Entwicklung und Aufbau eines Rechensystems inklusive Betriebssystem auf Basis eines 16-Bit Prozessors

UniBwM - ID 37/93

Diplomarbeit von
Oliver Henning
und
Uwe Reinhardt

Aufgabenstellung:
Prof. Dr. G. Regenspurg

Betreuung:
Dr. H. Kleebauer

Universität der Bundeswehr München
Fakultät für Informatik
Institut für Technische Informatik
Neubiberg, den 30. Dezember 1993

Inhaltsverzeichnis

1. Einführung

1.1. Der Hintergrund der Diplomarbeiten

Grundlage der Diplomarbeit ist ein Mikroprozessor mit Namen XProz, der am Institut für Technische Informatik an der Fakultät für Informatik der Universität der Bundeswehr München entwickelt wurde. Es sollte nun um diesen Mikroprozessor herum ein vollständiges Computersystem entwickelt und aufgebaut werden, im weiteren XSystem genannt. Auch für die Wahlpflichtvorlesung CAD-Praktikum, die jährlich in diesem Institut angeboten wird und in dessen Mittelpunkt die Entwicklung des Mikroprozessors steht, stellt das System dann eine vollständige Testumgebung für den XProz dar. Diese Aufgabe wurde aufgeteilt in die beiden Diplomarbeiten I:Hardware und II:Betriebssystem.

1.2. Aufgabenstellung der Diplomarbeit I: Hardware

Diese Diplomarbeit beinhaltet die Entwicklung und den Aufbau folgender Systeme für den Mikroprozessor XProz:

- Eine Tastaturschnittstelle.
- Eine Anschlußmöglichkeit für Massenspeicher.
- Eine parallele Schnittstelle zum Anschluss eines Druckers.
- Die serielle Schnittstelle XSer, welche an diesem Institut entwickelt wurde und ebenfalls auf einem XILINX-Chip implementiert ist.
- Wenn möglich, eine Videoeinheit.

Dabei sollten keine fertigen Bauteile verwendet werden, abgesehen von Speicherbausteinen, Treiberchips, Quarzoszillatoren und ähnlichem.

Dieser Teil wurde von Oliver Henning bearbeitet.

1.3. Aufgabenstellung der Diplomarbeit II: Betriebssystem

Diese Diplomarbeit beinhaltet die Programmierung eines Betriebssystems für den in Diplomarbeit I gebauten Rechner. Dieses soll folgende Komponenten beinhalten:

- Systeminitialisierung
- Dateiverwaltung
- Arbeitsspeicherverwaltung
- Eine Softwareschnittstelle zu den Geräten Tastatur, Drucker und serielle Schnittstelle.
- Einen Kommandointerpreter, um einfache Funktionen wie *Programm starten* und *Verzeichnisse anzeigen* u.s.w. ausführen zu können.

Dieser Teil wurde von Uwe Reinhardt bearbeitet.

1.4. Das XILINX-Chipssystem

Die komplette Hardware ist auf programmierbaren Logikbausteinen der Firma XILINX implementiert. Ein solcher Chip besteht im Wesentlichen aus einem Feld von gleichen Logikzellen (Logic-Cell-Array, LCA) mit dazwischenliegenden, schaltbaren Signalverbindungen (Verbindungsnetz) und einer Menge von Ein-Ausgabeleitungen.

Das besondere dieser Chips ist die Möglichkeit, die Funktionen der Logikzellen und das Verbindungsnetz benutzerspezifisch konfigurieren zu können. Dies geschieht, indem man nach dem Anlegen der Versorgungsspannung an den Chip, welcher zu diesem Zeitpunkt noch unkonfiguriert ist, über spezielle Anschlüsse die nötigen Konfigurationsdaten bitseriell überträgt. Diese Konfigurationsdaten sind ein Bitstrom einer bestimmten Länge und eines bestimmten Aufbaus und werden im XILINX-Chip mit S-RAM-Technologie gespeichert und bleiben bis zum Ausschalten oder Umkonfigurieren des Chips erhalten.

Bemerkung: Ein solcher programmierbarer Logikchip ist nicht zu verwechseln mit einem programmierbaren Microcontroller. Es wird wirklich die Hardwarefunktion programmiert, und dadurch ist eine sehr hohe Verarbeitungsgeschwindigkeit möglich.

Eine Logikzelle besteht im Wesentlichen aus einem Funktionsgenerator in Form eines Bitspeichers. Dadurch sind beliebige Funktionen von bis zu n Eingangsvariablen möglich. n ist hierbei abhängig vom der jeweiligen XILINX-Chip-Familie. Bei dieser Diplomarbeit wurden XILINX-Chips aus der 3000er-Serie verwendet, bei der n=5 ist. Desweiteren befinden sich noch m D-Flips-Flops in der Logikzelle, deren Dateneingang üblicherweise mit dem Ausgang des eben erwähnten Funktionsgenerators verbunden ist. Bei der 3000er-Serie sind zwei Flip-Flops in einer Logikzelle verfügbar.

1.5. Die Hardwareentwicklungsumgebung

Um eine bestimmte Hardware zu bekommen, muß man lediglich den Bitstrom erzeugen, der in den XILINX-Chip eingeschrieben wird. Diesen erzeugt man natürlich nicht *von Hand*. Vielmehr entwickelt man die gewünschte Schaltung komfortabel auf einem CAD-System und läßt dann daraus mit einem speziellen Programm den nötigen Bitstrom errechnen.

Das bei dieser Diplomarbeit verwendete CAD-System heißt WORKVIEW von der Firma VIEWLOGIC. Dieses CAD-System bietet auch die Möglichkeit, die entworfenen Schaltungen zu simulieren. Dadurch lassen sich schon viele Fehler ohne Hardwarerealisation beheben. Funktioniert die Schaltung auf dem Simulator ordungsgemäß, dann läßt man den Bitstrom errechnen. Mit diesem kann man seinen Hardwareentwurf real testen. Die Errechnung des Bitstroms kann je nach Schaltungsumfang mehrere Stunden in Anspruch nehmen.

1.6. Die Softwareentwicklungsumgebung

Da für den Prozessor XProz noch kein Compiler zur Verfügung stand, mußte das komplette Betriebssystem in der Assemblersprache dieses Prozessors geschrieben werden. Diese ist jedoch durch die wenigen Befehle und die fehlenden Arbeitsregister sehr unübersichtlich. Durch fehlende Sprung-, Stack und Move-Befehle ist sie sogar verwirrender als zum Beispiel die Assemblersprache des Motorola 68000 Prozessors. Die Entwicklung eines Betriebssystems in dieser Sprache wäre extrem zeitaufwendig gewesen.

Aus diesem Grund wurde für den XProz im Rahmen einer Trimesterarbeit [HUF93] ein komfortabler Makroassembler mit Namen XASS erstellt. Durch entsprechende Makros wurde die Assemblersprache so erweitert, daß die Assemblerprogramme einem 68000er-Assemblerprogramm sehr ähnlich sind.

Mit diesem Werkzeug konnte man also Software erstellen, aber nicht testen. Ein Testen der Software auf dem realen XSystem war nicht möglich, da dieses noch gar nicht existierte. Selbst wenn die Hardware schon fertig gewesen wäre, hätte man einen Debugger programmieren müssen, den man ohne Testmöglichkeit kaum hätte erstellen können. Erschwerend wäre hinzugekommen, daß der XProz über keine Einzelschrittbearbeitung (Tracing) verfügt.

Die eleganteste Lösung war ein Simulator auf einem PC. Dieser konnte aufgrund der einfachen, homogenen Struktur des XProz leicht implementiert werden. Er ist in der Hochsprache C von O. Henning programmiert, mit Borland C 3.1 für DOS übersetzt worden und heißt XSIM. Mit ihm hat man die Möglichkeit, Programme für den XProz schrittweise auszuführen und Speicherinhalte anzuzeigen.

Mit diesen beiden Werkzeugen wurde nahezu das komplette Betriebssystem entwickelt und getestet. Lediglich die sehr hardwarenahen Betriebssystemfunktionen konnten nicht bis ins letzte Detail simuliert werden.

Nachdem alles auf dem Simulator einwandfrei funktionierte, wurde ein kleines Datenkommunikationsprogramm für die serielle Schnittstelle erstellt, auf EPROM gebrannt und in das XSystem als minimales Debugsystem integriert. Damit war über einen angeschlossenen PC eine Übertragung von Speicherinhalten vom und zum XSystem möglich. So konnte das Betriebssystem zum XSystem übertragen und gestartet werden und man konnte zur Laufzeit Speicherinhalte aus dem XSystem auslesen.

1.7. Die Entwicklungsgeschichte

1.7.1. Ein erstes, minimales Testsystem

Im Rahmen des CAD-Praktikums wurde der Mikroprozessor um folgende Eigenschaften erweitert:

- Der neue Befehl ByteSwap (BSWP) wurde implementiert, um Operationen mit byteorientierten Geräten zu vereinfachen und auch zu beschleunigen. Dafür wurde der Befehl LSL (logical shift left) entfernt, da dieser die gleiche Funktionalität und Geschwindigkeit hat wie der Befehl ASL (arithmetical shift left).

- Die Anzahl der Unterbrechungsebenen wurde von zwei auf fünf erhöht. Damit wird eine effiziente Unterbrechungsbehandlung ermöglicht. Dies war nötig, da im voraus klar war, daß es mindestens zwei Unterbrechungsquellen gibt, nämlich die Tastatur und die serielle Schnittstelle. Der später nötig gewordene Zeitgeber, der auch eine Unterbrechungsquelle darstellt, konnte mit Hilfe der zusätzlichen Unterbrechungsebenen mühelos implementiert werden.

- Das einfache Ripple-Addierwerk des Prozessors wurde durch ein schnelleres 2-Bit-Carry-Lookahead-Addierwerk ersetzt. Damit wurde eine Taktfrequenz- und damit Geschwindigkeitssteigerung erreicht.

Mit diesem optimierten Prozessor wurde auf einer Lochrasterplatine in Wire-Wrap-Technik ein Minimalsystem mit Speicher (32 kWorte RAM + 32 kWorte EPROM), 8 Leuchtdioden und zwei Taster, die als Unterbrechungsquellen dienten, aufgebaut. Mit Hilfe verschiedener Testprogramme, die auf die EPROMs gebrannt wurden, wurde der Prozessor auf Korrektheit getestet. Ebenso wurde die maximal mögliche Taktfrequenz, bei der der Prozessor noch korrekt arbeitet, heuristisch ermittelt. Das Ergebnis dieses Tests: 32 MHz Prozessortakfrequenz, die intern durch 8 geteilt wird, also 4 MHz Phasenfrequenz und damit zirka 400.000 16-Bit-Befehle pro Sekunde.

Nachdem die Prozessortests erfolgreich abgeschlossen wurden, wurde die serielle Schnittstelle XSer auf die bestehende Lochrasterplatine hinzugefügt. Dank der Wire-Wrap-Technik konnten die nötigen Verbindungen einfach gelegt werden. Es galt nun, die XSer zu testen, da sie bisher nur auf einem Simulator ihre Funktionstüchtigkeit unter Beweis stellte. Dabei zeigten sich leider einige Probleme und Fehler, die dann behoben wurden.

Aufgrund der schlechten Erfahrung mit der XSer, die deutlich gezeigt hat, daß Realität und Simulation sich immer noch unterscheiden, wurde eine I/O-Einsteckkarte für einen PC entwickelt, auf der sich ein XILINX-Chip befindet. Dieser hat Zugriff auf 32 frei belegbare Eingabe/Ausgabeleitungen und kann mit einer beliebigen Schaltung programmiert werden. Mit ihrer Hilfe konnten die verschiedenen Schnittstellen, die in der Diplomarbeit I aufgebaut werden sollten, vollständig entwickelt und getestet werden, bevor sie dann für den XProz angepasst wurden.

Als Abschlußtest wurde eine intensive Datenübertragung zwischen einem PC und dem Testsystem über 24 Stunden hinweg durchgeführt, wobei zirka 82 MBytes an Daten in beide Übertragungsrichtungen transferiert und kontrolliert wurden. Da dabei kein einziger Fehler auftrat, konnte man davon ausgehen, daß sowohl der Prozessor als auch die serielle Schnittstelle stabil laufen.

1.7.2. Die ersten Gedanken zum Betriebssystem

Das Betriebssystem sollte sich klassischerweise in zwei Teile aufspalten: Das BIOS (Basic Input Output System) und das eigentliche Betriebssystem BS.

Das BIOS sollte zu den verschiedenen Geräten eine einfache und einheitliche Softwareschnittstelle darstellen. Das Betriebssystem sollte ausschließlich über das BIOS mit den Geräten kommunizieren. Diese Vorgehensweise hat den Vorteil, daß das BIOS unabhängig vom BS implementiert und getestet werden kann.

Das BIOS sollte für byteorientierte Geräte (Tastatur, Bildschirm, serielle Schnittstelle und Drucker) einfach eine Zeichenein- und eine Zeichenausgabefunktion zur Verfügung stellen. Da zu diesem Zeitpunkt noch nicht sicher war, ob es eine Videoeinheit geben wird oder nicht, konnte das BS erstmal seine Standardzeichenausgaben an die serielle Schnittstelle schicken. Ein dort angeschlossenes Zeichenterminal würde somit als virtuelle Videoeinheit dienen. Sollte die Grafikkarte dann doch möglich sein, genügt eine einfache BIOS-Erweiterung, um dem BS die reale Videoeinheit zugänglich zu machen.

Für blockorientierte Geräte, also die Massenspeicher, sollte das BIOS eine einfache Blocklese- und -schreiboperation dem BS zur Verfügung stellen sowie Möglichkeiten zur Abfrage der Speicherkapazität und Formatierung der Medien.

Das BS auf der anderen Seite bestand im Wesentlichen aus der Dateiverwaltung, mit deren Hilfe sich die blockorientierten Massenspeicher als zeichenorientierte Dateien zeigen. Der logische Aufbau des Dateisystems sollte ein Verzeichnis beinhalten, in dem alle Dateien des jeweiligen Massenspeichers aufgelistet sind. Ein hierarchisches Dateisystem, wie in UNIX, war nicht vorgesehen. Allerdings sollte jede Datei eine sogenannte Zugehörigkeit besitzen, eine Art Namenserweiterung, über die man differenzierter auf die Dateien zugreifen könnte. Dies entspricht im Wesentlichen einem 2-stufigen hierarchischem System.

Es war nun die Frage, wie die Datenblöcke einer Datei verkettet werden. Ein naheliegende Möglichkeit war die Implementierung eines FAT-Systems wie es von DOS benutzt wird. Dabei wird in einer zentralen, statischen Tabelle für jeden Block der Folgeblock eingetragen. Der Nachteil hierbei ist aber eben diese zentrale Tabelle. Wenn sie zerstört wird, kann sind die Daten des Datenträger verloren.

Eine Alternative wäre eine Verkettung ähnlich dem Dateisystem der 1541-Floppystation vom C64 von Commodore, wobei in jedem Block der Nachfolger eingetragen ist. Die Verkettungsinformationen sind also, wie die Daten selber, über den ganzen Datenträger verteilt. Der Nachteil ist, daß damit ein Block eine "krumme" Anzahl von Datenbytes enthält, eben nicht 512. Da der XProz über keinen Divisionsbefehl verfügt, wurde dieses Dateisystem verworfen.

Eine weitere Alternative war ein Dateisystem ähnlich dem von UNIX, bei dem Daten und Verwaltungsinformationen getrennt werden. Ein Block ist entweder vollständig ein Datenblock oder ein Zeigerblock, von dem aus auf weitere Blöcke verwiesen wird. Eine Datei ist somit wie ein Baum aufgebaut, die Knoten sind Verweise, die Blätter sind die Daten. Dieses System wurde schließlich implementiert

Die andere Aufgabe des Betriebssystems ist die Arbeitsspeicherverwaltung. Dafür sollte der Arbeitsspeicher in zwei Bereiche geteilt werden, einen Systembereich und einen Benutzerbereich. Der Systembereich besteht aus den Systemvariablen am Speicheranfang und einem Kellerspeicher fester Größe am Speicherende. Dazwischen liegt der Benutzerbereich, den ein Programm, welcher der Benutzer startet, komplett benutzen kann. Eine dynamische Speicherverwaltung schien zu diesem Zeitpunkt nicht notwendig. Im weiteren Verlauf der Entwicklung fiel die Entscheidung dann doch auf eine dynamische Speicherverwaltung in Form einer verketteten Liste von freien und belegten Speicherblöcken.

1.7.3. Hardwarerealisation

Nachdem der Prozessor XProz und der Chip für die serielle Schnittstelle XSer funktionstüchtig waren, galt es, das entgültige System aufzubauen. Dieses besteht aus einer einfachen Hauptplatine mit fünf Plätzen für Einsteckkarten, mit denen man das System erweitern kann. Die erste Aufgabe bestand also in Entwurf, Aufbau und Testen der Hauptplatine. Diese wurde so entworfen, daß sie in ein handelsübliches PC-Gehäuse mit Netzgerät eingebaut werden konnte. Damit war die Stromversorgung und die Befestigungsmöglichkeit für Massenspeicher sichergestellt.

Der Platinenentwurf wurde mit dem Layoutprogramm EAGLE von der Firma CadSoft gemacht. Nach dem Belichten, Entwickeln, Ätzen und Bohren der Platine wurde sie bestückt und verlötet. Mit Hilfe einer auf einer Lochrasterplatine aufgebauten Einsteckkarte mit 16 Leuchtdioden und diversen Testprogrammen, die wieder auf EPROM gebrannt wurden, wurde die Hauptplatine auf Funktionstüchtigkeit getestet. Lediglich ein paar kalte Lötstellen wurden entdeckt und durch Nachlöten beseitigt. Damit war der erste Meilenstein in der Systementwicklung erreicht.

Es stellte sich nun die Frage nach der Art des Massenspeicheranschlusses. Ein erster Gedanke war die schon oben erwähnte Diskettenstation 1541 des C64, welche über eine simple synchrone serielle Schnittstelle verfügt. Ein entsprechener Anschluß für das XSystem hätte man verhältnismäßig leicht realisieren können. Der große Nachteil dabei wäre jedoch die Unflexibilität dieser Schnittstelle gewesen. Weder Festplatten noch andere Diskettenstationen hätten dort Anschluß gefunden.

Eine weitere Möglichkeit wäre der Entwurf eines eigenen Floppycontrollers gewesen, an den man dann über den standardisierten SHUGART-Anschlusses eine bis vier handelsüblichen Diskettenstationen anschließen könnte. Da dieses Vorhaben im Rahmen des beschränkten Entwicklungszeitraumes wahrscheinlich ein zu großer Aufwand gewesen wäre, wurde diese Möglichkeit verworfen. Ein fertiger Controller, wie er zum Beispiel auf Standard-PC-Einsteckkarten zu finden ist, kam aufgrund der Vorgabe keine fertigen Bauteile nicht in Frage.

Die Wahl fiel schließlich auf den weit verbreiteten, gut dokumentierten und standardisierten SCSI-Bus (Small Computer Systems Interface). Diese Schnittstelle zeichnet sich aus durch Einfachheit der Hardware und Flexibilität bezüglich der Anschlußmöglichkeiten und der Software. Sie ist im wesentlichen ein 8-Bit-Parallelport mit doppeltem Handshaking, deshalb zeitunkritisch und für das XSystem gut geeignet.

Aufgrund dieser Entscheidung wurden für das XSystem eine SCSI-Festplatte mit 105 MBytes Kapazität von der Firma FUJITSU sowie ein SCSI-Diskettenlaufwerk mit 1.44 MBytes Kapazität der Firma TEAC gekauft. Die SCSI-Schnittstelle wurde mit Hilfe der oben erwähnten I/O-Einsteckkarte auf einem PC entwickelt und getestet.

Nun war das nächste Problem zu lösen: der Tastaturanschluß. Es war von Anfang an klar, daß eine handelsübliche MF2-Tastatur verwendet werden sollte, wie sie für einen normalen PC/AT verwendet wird. Bis auf ein undokumentiertes und merkwürdiges Verhalten beim Einschalten der angeschafften Tastatur bereitete die bidirektionale, synchrone serielle Schnittstelle keine größeren Probleme. Sie wurde, wie die SCSI-Schnittstelle, auf der PC-I/O-Karte entwickelt und getestet.

Nachdem dann diese beiden Schnittstellen voll funktionsfähig waren, kam der große Moment, diese auf einer Einsteckkarte für die Hauptplatine des XSystems in einem XILINX-LCA-Chip zu vereinen. Dabei mußte das Businterface für den XProz angepasst werden. Die Vorarbeit auf der PC-Karte erwies sich als erfolgreich, denn beide Schnittstellen arbeiteten auf Anhieb einwandfrei.

Da parallel zur Hardwareentwicklung auch das Betriebssystem Fortschritte machte, zeigte sich, daß es im System in irgendeiner Form eine Zeitbasis geben sollte, über die dann eine korrekte Timeout-Erkennung möglich ist. Ein Beispiel ist ein Drucker, der nicht eingeschaltet ist. Das System soll nämlich nach einer festen Zeit eine Fehlermeldung ausgeben. Implementiert wurde dies in Form eines Zeitgebers, der in konstanten Zeitabständen einen Unterbrechung auslöst und zur SCSI- und Tastaturkarte hinzugefügt wurde.

Zu diesem Zeitpunkt fehlte nur noch die parallele Schnittstelle zum Anschluß eines Druckers. Diese war einfach aufzubauen und wurde zusammen mit der schon getesteten XSer-Schnittstelle von Herrn Schmidt ebenfalls auf einer Einsteckkarte mit einem XILINX-Chip aufgebaut.

Nachdem auch diese Karte einwandfrei arbeitete, war der Anforderungskatalog der Diplomarbeit I erfüllt. Es fehlte eigentlich nur noch die Grafikkarte, die aber doch noch entwickelt wurde und so das XSystem zu einem eigenständigen Rechner machte.

1.7.4. Betriebssystementwicklung

Nach weiterführenden Überlegungen wurde die Idee des Betriebssystems geändert. Das BIOS als solches wurde abgeschafft und durch ein einheitliches Treiberkonzept ersetzt. Der Zugriff auf angeschlossene Geräte wurde komplett standardisiert und dadurch die Einbindung neuer Geräte erleichtert.

Da der XProz ein reiner 16-Bit-Prozessor ist und daher 8-Bit-Operationen sehr aufwendig und langsam sind, wurde konsequenterweise das gesamte Betriebssystem einschließlich der Treiber auf 16-Bit-Operationen ausgelegt. Das heißt zum Beispiel, daß auf Dateien nur wortweise operiert werden kann. Es werden also immer Vielfache von zwei Bytes gelesen oder geschrieben. Ebenso wird an einen Drucker immer ein Wort ausgegeben. Deshalb umfaßt ein Zeichen in diesem System 16 Bit.

Aufgrund der wortweisen Operation auf Dateien ist eine Pufferung der Sektoren der Massenspeicher nötig. Es wurden daher eine feste Anzahl von Sektorpuffern eingeführt, die sich alle geöffneten Dateien teilen. Um etwas mehr Flexibilität zu haben konnte ein Programm die Anzahl der Puffer explizit erhöhen und vermindern.

Entsprechend der Pufferung der Daten mußten aus Effizientsgründen Teile der Dateiverwaltung (Zuordnungstabelle, aktuelle Dateilänge, u.s.w.) ebenfalls im Arbeitsspeicher gepuffert werden. Dies führte schließlich zu dem sogenannten File-Desciptor-Block (FDB) in dem alle Informationen einer geöffneten Datei abgelgt sind. Ähnlich wie bei dem Betriebssystem DOS war die Anzahl der FDBs fest und damit die Anzahl der gleichzeitig geöffneten Dateien begrenzt. Später wurde diese statische Vorgehensweise abgeschafft. Ein FDB wird erst mit dem Öffnen einer Datei im Arbeistspeicher angelegt, und damit ist die Anzahl der offenen Dateien nur noch durch den Arbeitsspeicher begrenzt. Außerdem wird, falls keine Dateien geöffnet sind, kein Speicherplatz für nicht benutzte FDBs verschwendet.

Aus Geschwindigkeitsgründen und besseren Auslastung des Arbeitsspeichers wurde ein Massenspeicher-Cache implementiert, der eine Kopie der zuletzt angesprochenen Sektoren im Arbeitsspeicher verfügbar hält. Die Anzahl dieser Puffer ist dynamisch von der Speicherbelegung durch Programme abhängig, kann jedoch nie kleiner als eine bestimmte Untergrenze werden. Die Verwaltung dieser Cache-Puffer unterliegt dem SCSI-Treiber und ist damit völlig transparent für das Betriebsystem.

Nach erneutem Überdenken des Konzepts wurde klar, das eine mehrfache Pufferung der Sektoren, zum einen in den Dateipuffern, zum anderen im Cache, einen unnötigen Verwaltungsaufwand und Arbeitsspeicherverschwendung bedeutete. Deswegen übernahm der Cache die Aufgaben der Dateipufferung mit und das Betriebssystem übernahm keine Verwaltungsaufgaben mehr für Dateipuffer. Durch die schnelle Verfügbarkeit der Sektoren im Cache enthalten die FDBs keine Verweise mehr bezüglich der Datensektoren. Die FDBs wurden weitaus einfacher, denn die Informationen der Verwaltungssektoren wurden einfach aus dem Cache angefordert. Der SCSI-Treiber muß sich nun jedoch als wortorientierter anstatt als blockorientierter Treiber dem Betriebssystem zeigen.

Um auch Schreibzugriffe auf den Datenträger zu beschleuningen, wurde der Cache als Write-Back-Cache implementiert. Das heißt, ein Sektorpuffer in den Daten geschrieben wurden, wird erst zurückgeschrieben, wenn der Puffer für andere Sektoren verwendet werden muß. Problematisch sind bei dieser Vorgehensweise wechselbare Medien. Beim Wechsel eines Datenträgers besteht die Gefahr, daß für den alten Datenträger noch nicht geschriebene Puffer existieren und es so zu einer Inkonsistenz zwischen Cache und Datenträger kommen kann. Um dies zu vermeiden sollte der SCSI-Treiber im Hintergrund den Inhalt der Cache-Puffer wieder auf den Datenträger schreiben. Durch geschickte Programmierung des Treibers hätte dies für das laufende Programm nur einen geringen Geschwindigkeitsnachteil bedeutet.

Bei der Implementierung traten jedoch immer wieder neue Probleme im Zusammenhang mit der Hintergrundtätigkeit des SCSI-Treibers auf. Der logische Schluß war die Implementierung eines Taskswitchers, der in regelmäßigen, durch den Zeitgeber gesteuerten, Zeitabständen diese Aufgaben koordinierte. Von hier aus war der Schritt zu einem Multitasking-Betriebssystem nicht mehr weit.

Aufrund dieser Erkenntnis entstand der Betriebssystemkern, bestehend aus dynamischer Speicherverwaltung, Taskverwaltung mit preemtivem und kooperativem Multitasking, einer Treiberverwaltung und dem Dateisystem, welches über die Treiber auf die Geräte zugreift. Durch die Treiberverwaltung läßt sich eine mögliche Grafikkarte leicht in das Gesamtssystem einbinden.

Die letzte Hürde war die Implementierung der Benutzerschnittstelle. Die Wahl fiel auf einen Kommandozeilenintertpreter, ähnlich der COMMAND.COM von DOS, der jedoch kaum eigene Fähigkeiten besitzt. Die Aufgaben beschränken sich auf Ein-/ Ausgabeumlenkung und Programmstart. Die Betriebssystemfunktionen sich als externe Befehle, ausführbare Programme, realisiert. Um auf diese möglichst schnell zugreifen zu können wurden sie in einer ROM-Disk, welche ebenfalls über einen Treiber dem Betriebssystem zugänglich wurde, abgelegt.

1.7.5. Die Grafikkarte

Da noch Zeit zur Verfügung stand, wurde die Grafikkarte doch noch in Angriff genommen. Aufgrund der etwas beschränkten Rechenleistung des Prozessors XProz war nur Textmodus sinnvoll, bei dem im Bildschirmspeicher pro Zeichen lediglich ein Byte bzw. ein Wort abgelegt ist. Würde man ein pixelorientiertes Grafiksystem haben dann wäre das "Scrollen" des Bildschirminhalts zu träge gewesen, um mit dem System noch vernünftig arbeiten zu können. Auf der anderen Seite ließe sich die Hardware eines pixelorientierten Grafiksystems wesentlich leichter implementieren und wäre auch flexibler in der Grafikausgabe.

Die Wahl fiel schließlich auf eine farbfähige pixelorientierte Grafikkarte mit einer Auflösung von 640x400 Punkten, die jedoch Funktionen zum schnellen Bildschirm-Scrollen beinhaltet. Dieser sogenannte Auto-Copy-Mode entspricht ungefähr einer Teilfunktion heutiger Beschleuniger-Grafikkarten, wie sie für einen PC angeboten werden.

Um die Aufgaben des Betriebssystems zu erleichtern, wurde eine komplette zweite Grafikseite vorgesehen, auf die per Software umgeschaltet werden kann. Auf dieser Seite könnte das System zum Beispiel Fehlermeldungen ausgeben, ohne den Bildschirmaufbau des aktuell laufenden Programms zu beeinträchtigen. Da aber das Betriebssystem mittlerweile zu einem Multitasking-Betriebssystem gewachsen ist, wurde die zweite Bildschirmseite zusammen mit der Tastatur als vollständige virtuelle zweite Konsole implementiert. Der Benutzer kann nun jederzeit zwischen beiden Konsolen mit einem Tastendurck umschalten. Dies entspricht dem Virtuell-Konsolen-Konzept des Betriebssystems LINUX für PCs.

Ebenfalls vorgesehen ist ein Anschluß für eine Maus. Diese könnte einen Mauszeiger über den Bildschirm bewegen, ohne daß das Betriebssystem den Mauszeiger verwalten und darstellen müßte. Aufgrund des begrenzten Platzes des verwendeten XILINX-Chips konnte diese Idee leider nicht verwirklicht werden. Die Anschlüsse für die Maus sind aber schon vorhanden.

1.7.6. Das Ergebnis

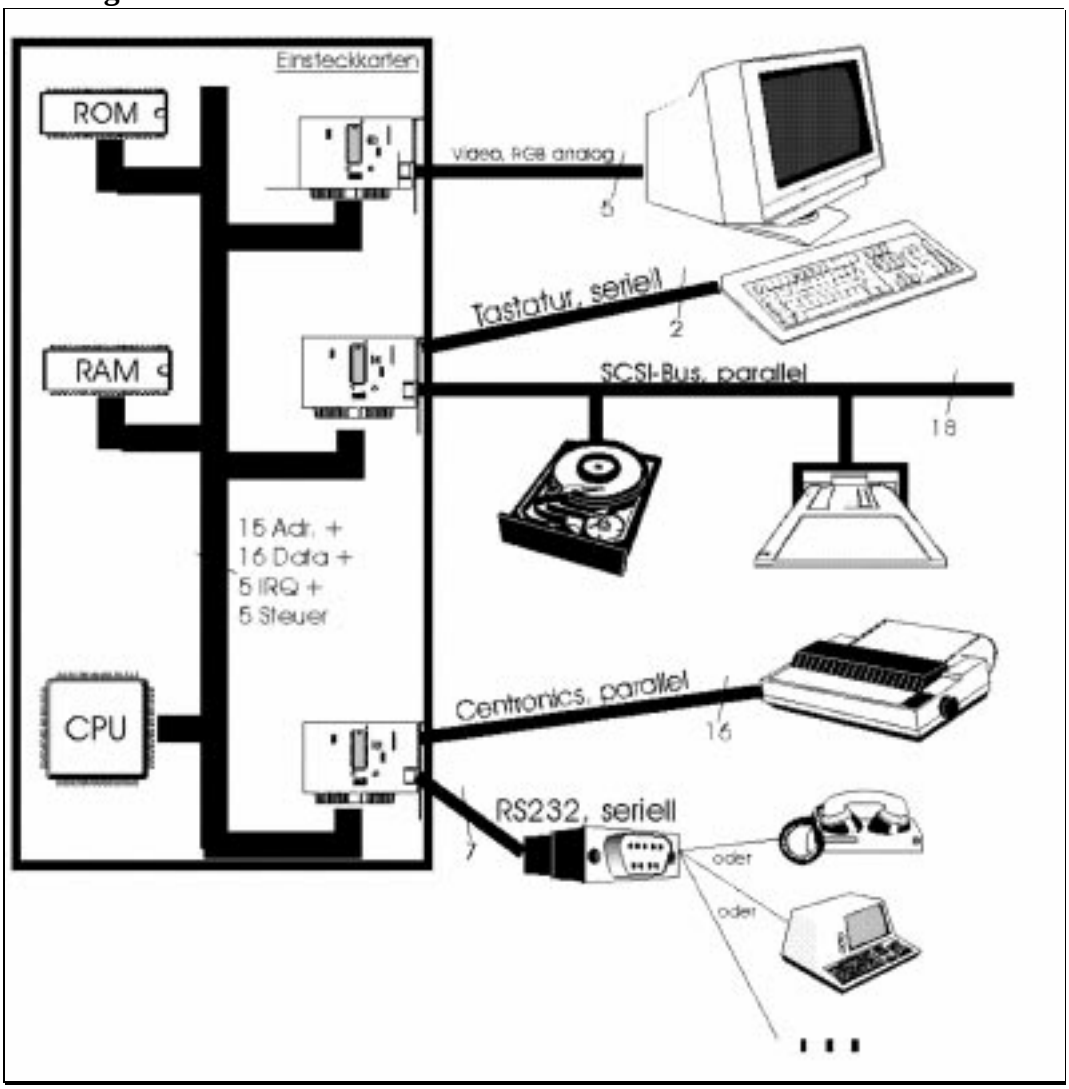


Bild 1: Aufbau des XSystems

Bild 2: XSystem

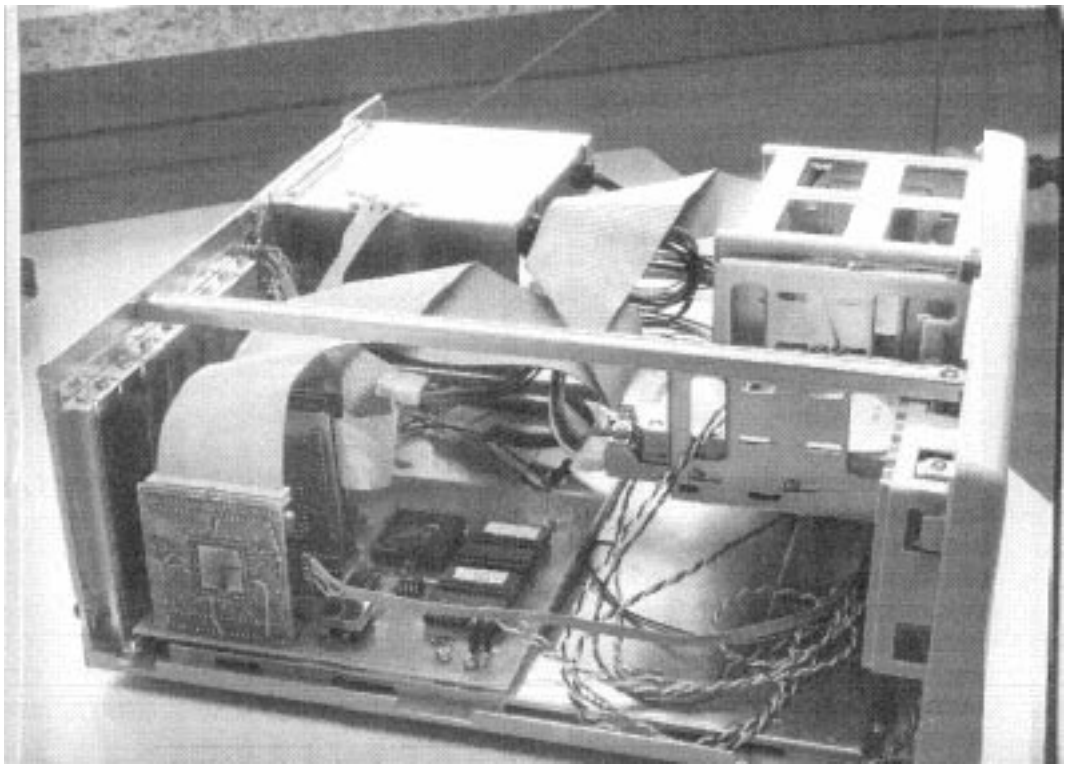


Bild 3: Blick ins Innere des XSystems

2. Prozessor XProz

2.1. Übersicht

Der Minimalprozessor XPROZ ist auf einem XILINX 3090PC84-70 implementiert. Es wurde hier versucht, einen einfachen Prozessor mit geringer Komplexität zu entwickeln. Dabei wurden einige Besonderheiten implementiert. So befinden sich der Programm-Zähler und die Register, bis auf das Statusregister, im RAM-Speicher. Es existiert kein expliziter Sprungbefehl, dieser wird durch eine Addition mit der Speicherzelle, die den Programm-Counter enthält, realisiert. Jeder Befehl kann bedingt und unbedingt ausgeführt werden. Auf diese Art lassen sich die entsprechenden bedingten und unbedingten Sprünge verwirklichen. Es werden 16-Bit Daten- und Adressworte verwendet. Somit können 32kWorte (=64kByte) RAM-Speicher, ROM-Speicher und I/O-Bereich adressiert werden. Der RAM- und ROM-Bereich werden in den Adressen durch das Adress-Bit 15 unterschieden. Ist dieses Bit gesetzt wird der ROM-Bereich adressiert, im anderen Fall der RAM-Bereich. Bei zugriffen auf den I/O-Adressbereich oder das Statusregister können die Befehle nur unbedingt ausgeführt werden. Ein Befehl umfaßt ein (z.B. and tr sf -,-,&- ;Statusregister(VCZN) löschen) bis vier Befehls-Worte. Es ist möglich bei einen Befehl bis zu drei Adressen anzugeben, zwei Operandenadressen und eine Zieladresse. In einem Befehl können zwei Speicherzellen verknüpft werden und in einer dritten abgelegt werden. Es werden mehrere Adressierungsarten für die Operanden und die Zieladresse zur Verfügung gestellt. Der Minimalprozessor XPROZ verfügt über fünf Interruptebenen (Interruptebene 1 bis Interruptebene 5), die gegeneinander priorisiert sind. Der Interrupt 5 hat dabei die höchste Priorität und der Interrupt 1 die niedrigste Priorität. Es können alle Interrupts zusammen gesperrt werden, gezieltes Sperren eines Interrupts ist nicht möglich. Die Interrupts können sowohl softwaremäßig durch Verändern des höherwertigen Bytes des Statusregister, als auch hardwaremäßig durch einen Low-Impuls an der entsprechenden Interruptleitung IRQ1~ bis IRQ5~ ausgelöst werden. Allerdings können sie nur softwaremäßig zurückgesetzt werden. Je nach Interruptebene x wird die Speicherzelle mit der Adresse x als Programm-Counter benutzt. Das heißt, daß die ersten sechs Worte des Arbeitsspeichers (Adresse 0 für User-Mode bis Adresse 5 für Interrupt-Mode 5) als Programm-Counter verwendet werden. Um die Arbeitsgeschwindigkeit zu erhöhen, wurden im Addierwerk vier 4-Bit-Volladdierer (jeweils 2-Bit-Carry-Look-Ahead; XILINX-Bezeichnung: HX83) verwandt.

2.2. Signalbeschreibung

Signal	Beschreibung
A_{14-0}	Adreßleitungen zur Adressierung von jeweils 32k Worten (=64k Bytes) RAM-, ROM- oder I/O-Bereich.
D_{15-0}	16 Datenleitungen
RAMSEL~	falls 0, ist die ausgegebene Adresse RAM-Adresse (Adresse $0000-$7fff)
ROMSEL~	falls 0, ist die ausgegebene Adresse ROM-Adresse (Adresse $8000-$ffff)
IOSEL~	falls 0, ist die ausgegebene Adresse eine I/O Adresse
R/W~	0: schreiben 1: lesen
O/E~	Ausgangstreiber der adressierten Einheit 0: aktiv, 1: inaktiv
RESET~	0: Sprung an Adresse $8000 im Interruptmodus 5 (SR=$0000) mit gesperrten Interrupts, 1: normaler Betrieb
IRQi~	0: Interrupt Request i, i=1...5, d.h. ab sofort wird Adresse i als Programmzeiger benutzt, 1: normaler Betrieb.
TAKT	maximal 28 MHz Prozessortakt (wird intern durch 8 geteilt)

Tabelle 1: Signalbeschreibung des Prozessors XProz

2.3. Programmzähler (PC)

Der PC befindet sich nicht als Register im Prozessor, sondern im normalen RAM-Speicher. Im User-Mode wird die Speicherzelle mit der Adresse 0 als pc verwendet. Im Interruptmodus 1 wird die Speicherzelle mit der Adresse 1 verwendet, entsprechendes gilt für die Interrupts 2 bis 5. Die Interrupts sind gegeneinander priorisiert. Der Interrupt 1 hat die geringste Priorität, der Interrupt 5 hat die höchste Priorität.

2.4. Statusregister

Bit#	Kürzel	Bedeutung
0	C	Carry-Flag
1	Z	Zero-Flag
2	N	Negaiv-Flag
3	V	Overflow-Flag
4	-	
5	-	
6	-	
7	-	
8	IE	Interrupt-Enable
9	I1~	0: Prozessor befindet sich in Interruptebene 1, falls kein Interrupt höherer Priorität aktiv.
10	I2~	dito, für Interruptebene 2
11	I3~	dito, für Interruptebene 3
12	I4~	dito, für Interruptebene 4
13	I5~	0: Prozessor befindet sich in Interruptebene 5
14	-	
15	-	

Tabelle 2: Statusregister des Prozessors XProz

I1~ bis I5~ können durch IRQ1~ bis IRQ5~ hardwaremäßig auf 0 gesetzt werden. Setzen (und damit Rückkehr in den User-Mode) ist nur per Software möglich. Das Umschalten in einen Interrupt-Modus erfolgt jeweils nach kompletter Abarbeitung eines Befehls. Eine Ausnahme davon ist gegeben, wenn dieser Befehl ein bedingter Befehl ist, der aufgrund der Bedingung nicht ausgeführt wird. In diesem Fall wird das Umschalten in den Interrupt-Modus solange verzögert, bis ein unbedingter Befehl, oder ein bedingter Befehl, dessen Bedingung erfüllt ist, abgearbeitet wurde. Dadurch konnte bei bedingten Befehlen, die nicht ausgeführt werden, ein Taktzyklus eingespart werden. Der Baustein IRQ-GEN ermittelt aus den anliegenden Ix~ -Signalen die aktuelle Interruptebene und übermittelt die Adresse des aktuellen Programm-Counters.

2.5. Befehlsformat

Der Prozessor stellt die Grundlage für eine Drei-Adress-Maschine dar. Das heißt, in einem Befehl werden zwei Operanden und ein Zieladresse angegeben. Ein Befehl besteht somit aus vier 16-Bit-Worten. Das erste Wort enthält Informationen über die Adressierungsarten der Operanden und die Zieladresse, Ausführungsbedingung des Befehls und ob das Statusregister Quelloperand oder Zieladresse ist. Es ist auch mögliche kürzere Befehle, je nach nicht benutzen eines Operanden oder der Zieladresse, zu benutzen. Es ist möglich fast jeden Befehl mit einer Ausführungsbedingung, die von den Flags (VNZC) abhängig ist, zu verknüpfen. Nur Befehle die sich auf den I/O-Bereich beziehen müssen unbedingt sein. Es besteht aber auch die Möglichkeit unbedingte Befehle auszuführen. Da die Programm-Counter im RAM liegen existiert kein Sprungbefehl. Ein Sprung wird durch den Additionsbefehl erreicht, bei dem die Zieladresse und (oder) ein Operand der jeweilige Programm-Counter ist. So sind bedingte, unbedingte, relative und absolute Sprünge möglich.

B_{15}	**B_{14}**	**B_{13}**	**B_{12}**	**B_{11}**	**B_{10}**	**B_9**	**B_8**	**B_7**	**B_6**	**B_5**	**B_4**	**B_3**	**B_2**	**B_1**	**B_0**
xsr	xio	zsr	zio	bed	sf	z1	z0	y1	y0	x1	x0	alu3	alu2	alu1	alu0
CC															
x-Operand															
y-Operand															
z-Adresse															

Tabelle 3: Befehlsformat des Prozessors XProz

alu 1 0	Befehlstyp	alu 3 2	Mnemonic
0 0	arithmetisch	0 0 0 1 1 0 1 1	add sub addc subc
0 1	logisch	0 0 0 1 1 0 1 1	nor and eor or
1 0	schiebe rechts	0 0 0 1 1 0 1 1	ror rorc asr lsr
1 1	schiebe links	0 0 0 1 1 0	rol rolc asl = lsl
1 1	höher- und niederwertiges Byte vertauschen	1 1	bswp

Tabelle 4: Befehle des Prozessors XProz

B5	B4	Beschreibung	Symbol
0	0	kein x-Operand (intern wird 1 verwendet	-
0	1	x ist als Direktoperand angegeben	#wert
1	0	Adresse von x ist angegeben	wert
1	1	Adresse von der Adresse von x ist angegeben	(wert)

Tabelle 5: X-Operand bestimmt durch x1, x0

B7	B6	Beschreibung	Symbol
0	0	kein y-Operand (intern wird 0 verwendet)	-
0	1	y ist als Direktoperand angegeben	#wert
1	0	Adresse von y ist angegeben	wert
1	1	Adresse von der Adresse von y ist angegeben	(wert)

Tabelle 6: Y-Operand bestimmt durch y1, y0

B9	B8	Beschreibung	Symbol
0	0	kein Ergebnis	-
0	1	Ergebnis hat gleiche Adresse wie y-Operand	=
1	0	Adresse von z ist angegeben	wert
1	1	Adresse von der Adresse von z ist angegeben	(wert)

Tabelle 7: Z-Adresse bestimmt durch z1, z0

Besonderheit: Bei y=(00) und z=(01) (d.h. kein y-Operand und Ergebnis gleich y-Operand) wird das Ergebnis im aktuellen PC abgespeichert. Dadurch sind 2-Befehlswort-Sprünge möglich, die sich immer auf den aktuelle PC beziehen, z.B:

```
add cs hf #label,-,=
```

Hinweis: Bei der Kombination y=(01) und z=(01) (d.h. y Operand ist als Direktoperand angegeben und Ergebnis ist gleich y Operand) wird der Direktoperand im Befehl mit dem Ergebnis überschrieben. Hiermit läßt sich beispielsweise ein Schleifenzähler realisieren:

```
       add tr hf #100,-,lab2+1 ;Zähler vorbesetzen
lab1:  ...
lab2:  sub tr sf -,#0,=        ;1 subtrahieren
       add ne hf #lab1,-,=     ;falls nicht 0 springen
```

B10	Beschreibung	Symbol
0	Flags werden durch diesen Befehl nicht verändert	hf
1	Flags werden entsprechend dem Ergebnis gesetzt	sf

Tabelle 8: Set Flags, bestimmt durch sf

B11	Beschreibung	Symbol
0	Befehl wird nur ausgeführt , wenn die Bedingung CC (B15-B12) erfüllt ist	gt, ne, ...
1	Befehl wird immer ausgeführt, B15-B12 modifizieren x-Operand und Ergebnisadresse	tr

Tabelle 9: Bedingt/Unbedingt, bestimmt durch bed

Falls B11=1:

B15	Beschreibung	Symbol
0	Der normal adressierte x-Operand wird verwendet	keines
1	Der x-Operand wird durch das Statusregister ersetzt	&

Tabelle 10: Statusregister als X-Operand, bestimmt durch xsr

B14	Beschreibung	Symbol
0	Der normal adressierte x-Operand wird verwendet	keines
1	Die Adresse des x-Operanden wird als I/O-Adresse behandelt (keine Wirkung falls kein x-Operand oder immidiate Wert)	@

Tabelle 11: X-Operand aus dem I/O-Adressraum, bestimmt durch xio

B13	Beschreibung	Symbol
0	Das Ergebnis wird normal abgespeichert	keines
1	Ergebnis wird zusätzlich zur normalen Speicherung auch im Statusregister abgespeichert. Die Flags werden dabei nur gespeichert falls auch sf=1. Es muß eine Zieladresse angegeben werden, z.B &100.	&

Tabelle 12: Ergebnis ins Statusregister, bestimmt durch zsr

B12	Beschreibung	Symbol
0	Das Ergebnis wird normal abgespeichert	keines
1	Die Adresse des Ergebnisses wird als I/O-Adresse behandelt (keine Wirkung falls kein Z-Operand)	@

Tabelle 13: Ergebnis in den I/O-Adressraum, bestimmt durch zio

Falls B11=0:

B15	B14	B13	B12	Symbol	Beschreibung	Funktion
0	0	0	0	me	minus or equal	Z + N
0	0	0	1	pe	plus and not equal	(~Z)(~N)
0	0	1	0	hi	higher	(~C)(~Z)
0	0	1	1	ls	lower or same	C + Z
0	1	0	0	cc	carry clear	~C
0	1	0	1	cs	carry set	C
0	1	1	0	ne	not equal	~Z
0	1	1	1	eq	equal	Z
1	0	0	0	vc	overflow clear	~V
1	0	0	1	vs	overflow set	V
1	0	1	0	pl	plus	~N
1	0	1	1	mi	minus	N
1	1	0	0	ge	greater or equal	NV + (~N)(~V)
1	1	0	1	lt	less then	N(~V)+(~N)V
1	1	1	0	gt	greater then	NV(~Z)+(~N)(~V)(~Z)
1	1	1	1	le	less or equal	Z+N(~V)+(~N)V

Tabelle 14: Condition Code, bestimmt durch CC

Bemerkung zum Assembler: Sind sowohl & als auch @ angegeben, so muß & vor @ stehen. Die Symbole für Statusregister (&) und I/O-Adresse (@) dürfen nur beim x-Operanden und dem Ergebnis stehen, nicht beim y-Operanden.

Befehl	Funktion	C-Flag	Z-Flag	N-Flag	V-Flag
ADD X,Y,Z	Z = X + Y	Carry	not(Z_i)	Z_{15}	Overflow
ADDC X,Y,Z	Z = X + Y + C	Carry	not(Z_i)	Z_{15}	Overflow
SUB X,Y,Z	Z = Y - X	Carry	not(Z_i)	Z_{15}	Overflow
SUBC X,Y,Z	Z = Y - X - C	Carry	not(Z_i)	Z_{15}	Overflow
NOR X,Y,Z	Z = X nor Y	0	not(Z_i)	Z_{15}	undefiniert
OR X,Y,Z	Z = X or Y	0	not(Z_i)	Z_{15}	undefiniert
AND X,Y,Z	Z = X and Y	0	not(Z_i)	Z_{15}	undefiniert
EOR X,Y,Z	Z = X eor Y	0	not(Z_i)	Z_{15}	undefiniert
ROR X,Y,Z	$Z_i = X_{i+1}$ für 0 <= i <=14 $Z_{15} = X_0$	X_0	not(Z_i)	Z_{15}	undefiniert
RORC X,Y,Z	$Z_i = X_{i+1}$ für 0 <= i <=14 $Z_{15} = C$	X_0	not(Z_i)	Z_{15}	undefiniert
LSR X,Y,Z	$Z_i = X_{i+1}$ für 0 <= i <=14 $Z_{15} = 0$	X_0	not(Z_i)	Z_{15}	undefiniert
ASR X,Y,Z	$Z_i = X_{i+1}$ für 0 <= i <=14 $Z_{15} = X_{15}$	X_0	not(Z_i)	Z_{15}	undefiniert
ROL X,Y,Z	$Z_i = X_{i-1}$ für 1 <= i <=15 $Z_0 = X_0$	X_{15}	not(Z_i)	Z_{15}	undefiniert
ROLC X,Y,Z	$Z_i = X_{i-1}$ für 1 <= i <=15 $Z_0 = C$	X_{15}	not(Z_i)	Z_{15}	undefiniert
LSL X,Y,Z	$Z_i = X_{i-1}$ für 1 <= i <=15 $Z_0 = 0$	X_{15}	not(Z_i)	Z_{15}	undefiniert
BSWP X,Y,Z	$Z_i = X_{i+8}$ für 0 <= i <= 7 $Z_i = Y_{i-8}$ für 8 <= i <= 15	X_{15}	not(Z_i)	Z_{15}	undefiniert

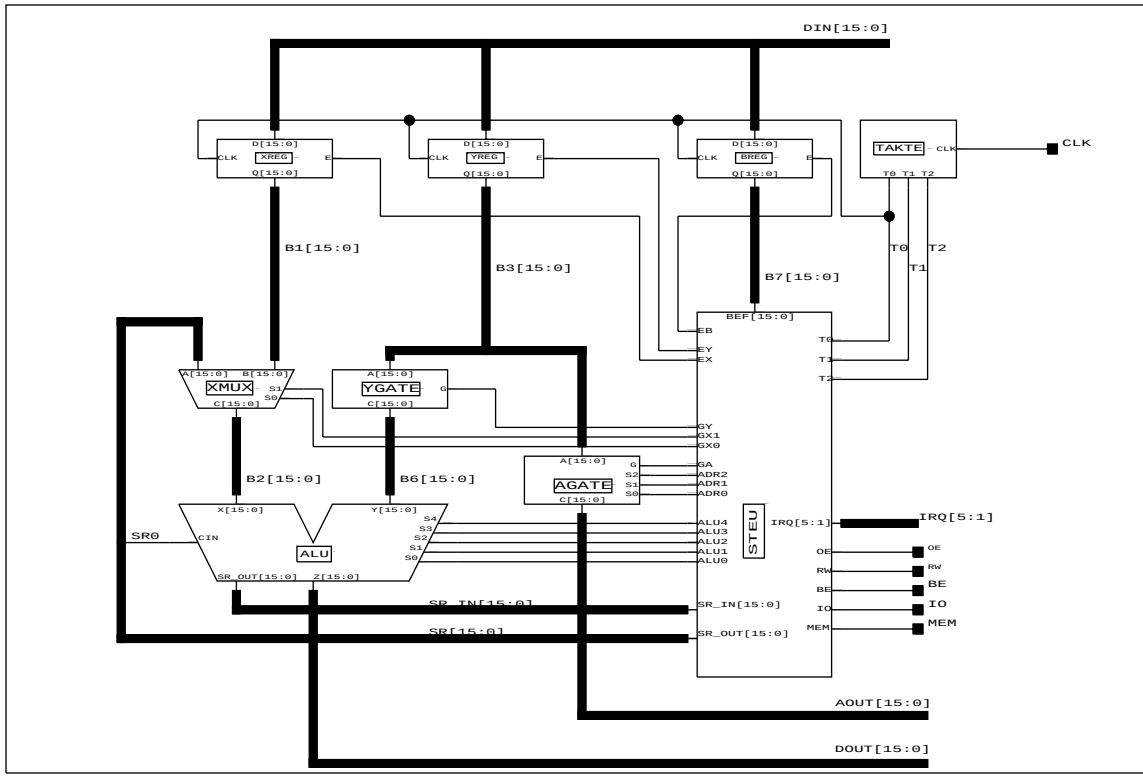
Tabelle 15: Beschreibung der Prozessorbefehle

Beispielprogramm:

```
        add   tr hf #100,xw,yw   ;addiere 100 zu xw und speichere
                                 ;Ergebnis in yw, Flags unverändert
        add   cs sf -,yw,=       ;addiere 1 zu yw, falls Carry gesetzt
                                 ;und setze Flags entsprechend dem
                                 ;Ergebnis
        add   mi hf #l1,-,=      ;springe nach l1, falls N-Flag gesetzt
        or    tr hf @$20,#$e,(30);Oder-Verknüpfung ohne Änderung der
                                 ;Flags, 1. Operand ist Wert unter der
                                 ;I/O-Adresse $20. 2. Operand ist $e
                                 ;(14 dezimal). Ergebnis in die
                                 ;Speicherstelle, deren Adresse in der
                                 ;Speicherstelle 30 steht.
l1:     or    tr hf &#$ffff,-,xw ;kopiere Statusregister in xw,
                                 ;dabei sind die Bits 5-7 und 14-15
                                 ;gesetzt.
l2:     add   tr hf #l2,-,=      ;Endlosschleife
xw:     dcw   100                ;Variable xw mit 100 belegen
yw:     dcw   10                 ;Variable yw mit 10 belegen
```

2.6. Funktionseinheiten des XPROZ

Der XPROZ besteht aus meheren Funktionsheiten. Dabei spielen die ALU (Arithmetic-Logic-Unit) und das STEU (Steuerwerk) die wichtigste Rolle. XREG, YREG und BREG sind jeweils 16-Bit Register. XMUX, YGATE und AGATE sind Multiplexer. Die Einheit Takte erzeugt die drei verschiedenen Takte, die im System benutzt werden.


DIN[15:0]
D[15:0]
CLK
XREG
E
Q[15:0]
YREG
BREG
TAKTE
CLK
T0 T1 T2
B1[15:0]
B3[15:0]
B7[15:0]
BEF[15:0]
EB
EY
EX
XMUX
YGATE
AGATE
GY
GX1
GX0
GA
ADR2
ADR1
ADR0
B2[15:0]
B6[15:0]
STEU
IRQ[5:1]
SR0
CIN
ALU
ALU4
ALU3
ALU2
ALU1
ALU0
OE
RW
BE
IO
MEM
SR_IN[15:0]
SR_OUT[15:0]
SR[15:0]
AOUT[15:0]
DOUT[15:0]

XREG und BREG

Beschreibung:	16-Bit Register
Eingänge:	D[15:0], CLK, E
Ausgänge:	Q[15:0]
Funktion:	falls E=1 werden bei der steigenden Taktflanke die anliegenden Signale D[15:0] in das Register übernommen
Reset:	Nach dem Reset enthalten die Register den Wert 0

YREG

Beschreibung:	16-Bit Register
Eingänge:	D[15:0], CLK, E
Ausgänge:	Q[15:0]
Funktion:	falls E=1 werden bei der steigenden Taktflanke die anliegenden Signale D[15:0] in das Register übernommen
Reset:	Nach dem Reset enthält das Register den Wert $8000

TAKTE

Beschreibung:	Erzeugt aus einen externen CLK die intern benötigten Takte T0, T1, T2
Eingänge:	CLK
Ausgänge:	T0, T1 T2
Funktion:	Erzeugt folgende Taktsignale:

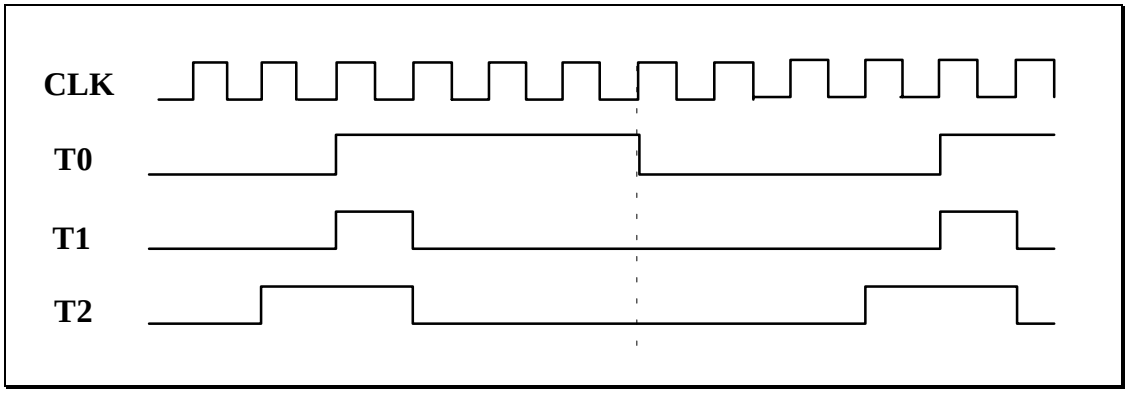


Bild 4: Ausgangssignale Modul TAKT vom XProz

Reset:	Während Reset sind T0, T1 und T2 0. Nach dem Reset starten die Signale an der markierten Stelle.

XMUX

Beschreibung:	Multiplexer, der wahlweise das XREG, das Statusregister oder den Wert $0001 durchschaltet.
Eingänge:	A[15:0], B[15:0], S1, S0
Ausgänge:	C[15:0]

Funktion:

S1	S0	C[15:14]	C[13:8]	C[7:4]	C[3:0]
0	0	0	A[13:8]	0	A[3:0]
0	1	B[15:14]	A[13:8]	B[7:4]	A[3:0]
1	0	0	0	0	1
1	1	B[15:14]	B[13:8]	B[7:4]	B[3:0]

Tabelle 16: Funktionstabelle Modul XMUX des XProz

YGATE

Beschreibung: Multiplexer, der wahlweise den Ausgang des YREG oder den Wert $0000 durchschaltet.

Eingänge: A[15:0], G

Ausgänge: C[15:0]

Funktion:

G	C[15:0]
0	$0000
1	A[15:0]

Tabelle 17: Funktionstabelle Modul YGATE des XProz

AGATE

Beschreibung: Multiplexer, der wahlweise den Ausgang des YREG oder die Werte $0000, $0001, $0002, $0003, $0004 oder $0005 (entsprechend des Program-Counter) durchschaltet.

Eingänge: A[15:0], G, S2, S1, S0

Ausgänge: C[15:0]

Funktion:

G	S2	S1	S0	C[15:0]
1	x	x	x	A[15:0]
0	0	0	0	$0000
0	0	0	1	$0001
0	0	1	0	$0002
0	0	1	1	$0003
0	1	0	0	$0004
0	1	0	1	$0005
0	1	1	0	$0006
0	1	1	1	$0007

Tabelle 18: Funktionstabelle Modul AGATE des XProz

ALU

Beschreibung: führt die arithmetischen und logischen Operationen aus

Eingänge: X[15:0], Y[15:0], CIN, S[4:0]

Ausgänge: Z[15:0], SR_OUT[15:0]

Funktion:

S1 S0 S3 S2	Z[15:0]	C-Flag	V-Flag
0 0 0 0	X+Y	C(X+Y)	V(X+Y)
0 0 0 1	Y-X	C(Y-X)	V(Y-X)
0 0 1 0	X+Y+CIN	C(X+Y+CIN)	V(X+Y+CIN)
0 0 1 1	Y-X-CIN	C(X-Y-CIN)	V(X-Y-CIN)
0 1 0 0	X nor Y	0	V(X+Y)
0 1 0 1	X and Y	0	V(X-Y)
0 1 1 0	X eor Y	0	V(X+Y+CIN)
0 1 1 1	X or Y	0	V(Y-X-CIN)
1 0 0 0	ror X	X[0]	V(X+Y)
1 0 0 1	rorc X	X[0]	V(Y-X)
1 0 1 0	asr X	X[0]	V(X+Y+CIN)
1 0 1 1	lsr X	X[0]	V(X-Y-CIN)
1 1 0 0	rol X	X[15]	V(X+Y)
1 1 0 1	rolc X	X[15]	V(X-Y)
1 1 1 0	asl X	X[15]	V(X+Y+CIN)
1 1 1 1	bswp X,Y	X[15]	V(Y-X-CIN)

Tabelle 19: Funktionstabelle Modul ALU des XProz

falls S4=1: R-OUT[15:0] = Z[15:0]
falls S4=0: SR_OUT[0] = C-Flag
SR_OUT[1] = nor(Zi) für i = 0..15
SR_OUT[2] = Z[15]
SR_OUT[3] = V-Flag
SR_OUT[15:4] = Z[15:4]

ADD/SUB, Teil der ALU

Beschreibung: Führt die Additions- und Subtraktionsbefehle aus
Eingänge: A[15:0], B[15:0], SUB, WC, CIN
Ausgänge: C[15:0], C, V
Funktion:

WC SUB	C[15:0]	C-Flag	V-Flag
0 0	X+Y	C(X+Y)	V(X+Y)
0 1	Y-X	C(Y-X)	V(Y-X)
1 0	X+Y+CIN	C(X+Y+CIN)	V(X+Y+CIN)
1 1	Y-X-CIN	C(Y-X-CIN)	V(Y-X-CIN)

Tabelle 20: Funktionstabelle Modul ADD/SUB des XProz

Zur Durchführung der Befehle werden vier 4 - Bit - Volladdierer (je 2-Bit Carry-Look-Ahead), die jeweils vier Bit verküpfen, benutzt.

LOG, Teil der ALU

Beschreibung: führt die logischen Befehle NOR, EOR, AND und OR aus
Eingänge: A[15:0], B[15:0], S1, S2
Ausgänge: C[15:0]
Funktion:

S1 S0	C[15:0]
0 0	A nor B
0 1	A and B
1 0	A eor B
1 1	A or B

Tabelle 21: Funktionstabelle Modul LOG des XProz

ZERO, Teil der ALU

Beschreibung: Berechnet das Z-Flag

Eingänge: A[15:0]

Ausgänge: Z

Funktion:

A[15:0]	Z
$0000	1
sonst	0

Tabelle 22: Funktionstabelle Modul ZERO des XProz

STEU

Beschreibung: Steuerwerk, steuert die Befehlsbearbeitung

Eingänge: BEF[15:0], T0, T1, T2, IRQ[5:1], SR_IN[15:0]

Ausgänge: EB, EY, EX, GY, GX1, GX0, GA, ADR2, ADR1, ADR0, ALU4, ALU3, ALU2, ALU1, ALU0, SR_OUT[15:0], OE, RW, BE, IO, MEM

Bemerkung: Ein Befehl kann bis zu 17 Phasen durchlaufen bis dieser abgearbeit ist. Das Steuerwerk enthält 17 Flip-Flops, die widerspiegeln in welcher Phase sich das Steuerwerk befindet.

Funktion:

Phase	Funktion	Beschreibung
0	PC -> y	PC laden
1	(y) -> Befehlsregister	Befehl lesen
2	y+1 -> PC,y	PC+1
3	y+1 -> PC	PC+1
4	(y) -> x,y	Adresse der Adresse von x-Operand lesen
5	(y) -> x,y	Adresse von x-Operand lesen
6	(y) -> x,y	x-Operand lesen
7	PC -> y	PC laden
8	y+1 -> PC	PC+1
9	(y) -> y	((Adresse von) Adresse von) y-Operand lesen
10	(y) -> y	Adresse von y-Operand lesen
11	(y) -> y	y-Operand lesen
12	ALU -> x	ALU-Operation ausführen, Ergebnis -> x
13	PC -> y	PC laden
14	y+1 -> PC -> y falls ~CC	PC+1 PC+1
15	(y) -> y	Adresse von Ergebnis laden
16	(y) -> y	Adresse von Ergebnis bei indirekter Adresse lesen
17	x -> (y)	Ergebnis speichern

Tabelle 23: Phasenbeschreibung Modul STEU des XProz

Mit:

x: X-Register
y: Y-Register
CC: 1 falls Condition-Code erfüllt, 0 sonst
ALU: ALU-Ausgang
PC: Programm-Zähler = Speicheradresse 0 für User-Mode
x für Interruptebene x

Phase	EB	EY	EX	GY	GX1	GX0	GA	ADR2	ADR1	ADR0	ALU4	ALU3	ALU2	ALU1	ALU0	RW	MEM	IO
0	-	1	-	-	-	-	0	f1	f5	f6	-	-	-	-	-	1	0	1
1	1	0	-	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
2	0	1	-	1	1	0	0	f1	f5	f6	-	0	0	0	0	0	0	1
3	0	0	-	1	1	0	0	f1	f5	f6	-	0	0	0	0	0	0	1
4	0	1	1	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
5	0	1	1	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
6	0	-	1	-	-	-	1	-	-	-	-	-	-	-	-	1	f8	~f8
7	0	1	0	-	-	-	0	f1	f5	f6	-	-	-	-	-	1	0	1
8	0	0	0	1	1	0	0	f1	f5	f6	-	0	0	0	0	0	0	1
9	0	1	0	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
10	0	1	0	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
11	0	1	0	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
12	0	-	1	f2	f3	f4	0	f1	f5	f6	f7	B3	B2	B1	B0	0	f10	1
13	0	1	0	-	-	-	0	f1	f5	f6	-	-	-	-	-	1	0	1
14	0	~CC	0	1	1	0	0	f1	f5	f6	-	0	0	0	0	0	0	1
15	0	1	0	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
16	-	1	0	-	-	-	1	-	-	-	-	-	-	-	-	1	0	1
17	-	-	-	0	1	1	1	-	-	-	-	0	0	0	0	0	f9	~f9

Tabelle 24: Funktionstabelle Modul STEU des XProz

Mit:

f1 = ~(I5 * I4)
f2 = B6 + B7f3 = ~(B11 * B15)
f4 = B4 + B5
f5 = ~(I2 * I 3) * I4 *I5
f6 = ~((I1 + ~(I2) + ~(I4)) * (~(I4) + I3) *I5)
f7 = B13 * B11
f8 = B11 * B14 * B5
f9 = B11 * B12
f10 = ~(CC)+~(B8)+B9+B6+B7
f1, f5 und f5 werden nur in Phase 0 ermittelt und behalten danach ihren Wert bei!

Ausgangsfunktion:

Pi = Phase i (Zustand i)
Bi = Befehlsbit i
CC = 1 falls Condition-Code erfüllt, 0 sonst
Ii = SR_{i+9} (0, falls Interruptebene i)
EB = P1
EY = ~(P1 + P3 + P8 + (P14 * CC))
EX = P4 + P5 + P6 + P12
GY = ~(P12 * (~B6) * (~B7) + P17)
GX1 = ~(P12 * B11 * B15)
GX0 = P12 * (B4 + B5) + P17
GA = ~(P0 + P2 + P3 + P7 + P8 + P12 + P13 + P14)
ADR2 = ~(I5 * I4)
ADR1 = ~(I2 * I 3) * I4 *I5
ADR0 = ~((I3 + ~(I2)) + ~(I4)) * (~(I4) + I3) *I5
ALU4 = P12 * B13 * B11
ALU3 = P12 * B3

ALU2 = P12 * B2
ALU1 = P12 * B1
ALU0 = P12 * B0
RW = ~(P2 + P3 + P8 + P12 + P14 + P17)
IO = ~(P6 * B11 * B14 * B5 + P17 * B11 * B12)
MEM =(P6*B11*B14*B5)+(P12*(~(CC)+~(B8)+B9+B6+B7))+(P17*B11*B12)

Ansteuertabelle der Flip-Flops:

D0	= P12 * (~C9) * (~C8) + P17	
D1	= P0 + P14 * (~CC)	
D2	= P1	
D3	= P2	R3 = P8 + P9 + P12
D4	= P3	R4 = P5 + P6
D5	= P4 + P3 * (~B4)	
D6	= P5 + P3 * (~B5)	
D7	= P6	R7 = P12
D8	= P7 + P2 * (~(B4 +B5))	R8 = P9 + P12
D9	= P8 + D8 * B8 * (~B9)	R9 = P12
D10	= P9	R10 = P11 + P12
D11	= P10 + P9 * (~B6)	R11 = P12
D12	= (P2 * (~(B4 + B5)) + P6) * (~(B6 + B7)) + P9 (~B7) + P11	
D13	= P12	R13 = P0
D14	= P13	
D15	= P14	R15 = P1 + P17
D16	= P15	R16 = P17
D17	= P14 * (~C9) + P15 * (~C8) + P16	R17 = P1

Zustandsübergangs-Tabelle:

C9,C8 = B7, B6 falls B9 = 0 und B8 = 1 (Z-Operand = Y-Operand)
= B9, B8 sonst
P0 -> P1
P1 -> P2
P2 -> P3, falls B4 + B5
-> P8, falls (~(B4 + B5)) * (B6 + B7) * ((~B8) + B9)
-> P9, falls (~(B4 + B5)) * (B6 + B7) * B8 * (~B9)
-> P12, falls (~(B4 + B5)) * (~(B6 + B7))
P3 -> P4, falls B4 * B5
-> P5, falls (~B4) * B5
-> P6, falls B4 * (~B5)
P4 -> P5
P5 -> P6
P6 -> P7, falls B6 + B7
-> P8, falls ~(B6 + B7)
P7 -> P8, falls (~B8) + B9
-> P9, falls B8 + (~B9)
P8 -> P9
P9 -> P10, falls B6 * B7
-> P11, falls (~B6) * B7
-> P12, falls (~B7)
P10 -> P11
P11 -> P12

P12 -> P13, falls C9 + C8
-> P0, falls (~C9) * (~C8)
P13 -> P14
P14 -> P15. falls CC * C9
-> P17, falls CC * (~C9)
-> P1, falls (~CC)
P15 -> P16, falls C8
-> P17, falls (~C8)
P16 -> P17
P17 -> P0

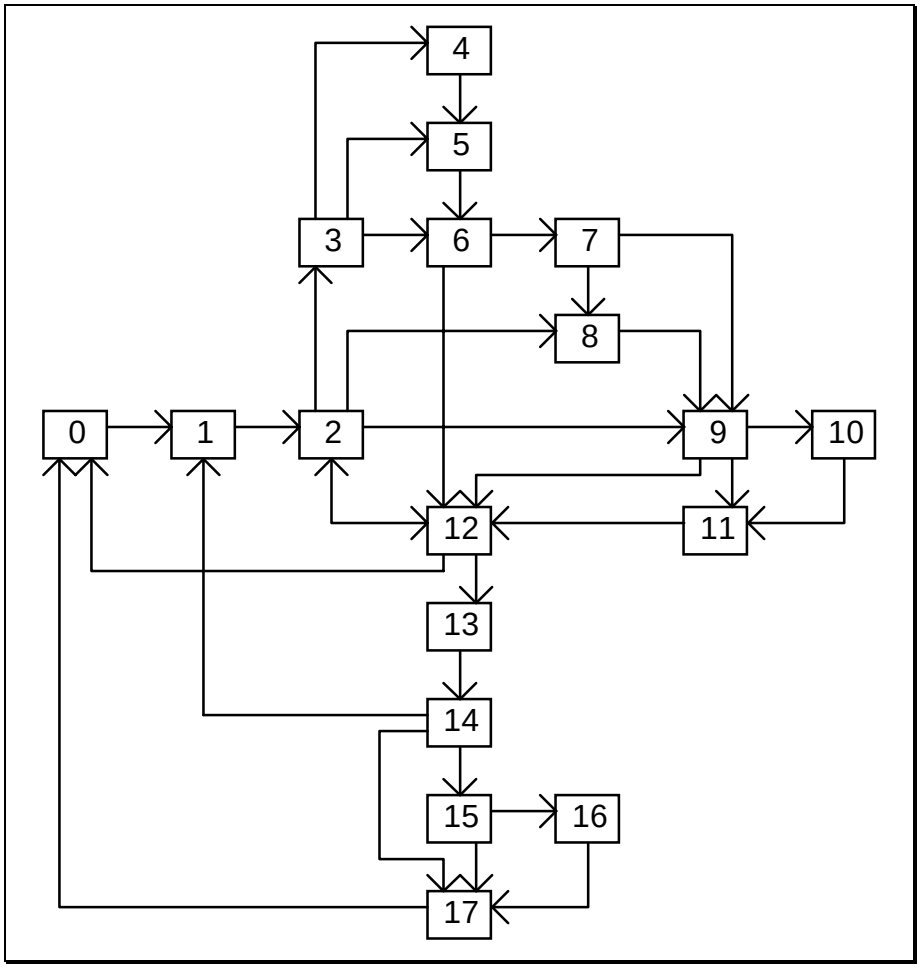


Bild 5: Zustands-Übergangsdiagramm des Steuerwerks

IRQ-GEN. Teil der Steuerwerkes

Beschreibung: Generiert die aktuelle Progarmm-Counter-Adresse
Eingänge: SR_OUT[13:9]
Ausgänge: A2, A1, A0

Funktion:

I5	I4	I3	I2	I1	A2	A1	A0
1	1	1	1	1	1	0	1
1	1	1	1	0	1	0	0
1	1	1	0	x	1	1	1
1	1	0	x	x	1	1	0
1	0	x	x	x	0	0	1
0	x	x	x	x	0	0	0

Tabelle 25: Funktionstabelle Modul IRQ-GEN des XProz

Mit:

I1: SR_OUT[9]
I2: SR_OUT[10]
I3: SR_OUT[11]
I4: SR_OUT[12]
I5: SR_OUT[13]
x:: Zustand ohne Bedeutung

Bemerkung: Die Ausgänge A2 und A0 passieren jeweils noch einen Inverter bevor sie zu den Signalen ADR2 und ADR0 werden. A1 entspricht direkt ADR1.

BED. Teil der Steuerwerkes

Beschreibung:	Überprüft Condition-Code
Eingänge:	BED3, BED2, BED1, BED0, V, N, Z, C
Ausgänge:	CC
Funktion:	Es wird entsprechend der Tabelle Condition-Code durch AND-Verküpfungen überprüft, ob die geforderten Flags gesetzt sind. Simmt die Flag-Folge mit der Befehlsbdingung überein hat CC den Wert 1.

3. Hauptplatine XBoard

3.1. Übersicht

Die Hauptplatine hat eine Größe von ca. 160 x 180 mm. Durch passende Bohrlöcher läßt sie sich in ein Standard-PC-Gehäuse einbauen. Die Stromversorgung erfolgt über eine Schraubklemme. Hier muß eine stabilisierte 5 Volt Spannungsquelle angeschlossen werden, zum Beispiel das eingebaute Netzteil des PC-Gehäuses. Die Hauptplatine versorgt alle weiteren Komponenten mit Strom.

Bemerkung: Der ebenfalls vorgesehene 12-Pin Versorungsanschluß, wie er auf handelsüblichen Hauptplatinen für INTEL-Prozessoren verwendet wird, konnte leider nicht benutzt werden, da ein entsprechender Anschlußstecker nicht besorgt werden konnte.

Auf der Platine befinden sich Sockel für den Prozessor, Quarzoszillatoren, Speicher- und Treiberbausteine sowie 5 Steckplätze für Erweiterungskarten. Ein Anschluß einschließlich Vorwiderstand für eine Leuchtdiode ist ebenso vorhanden wie die Anschluß für einen Reset-Taster.

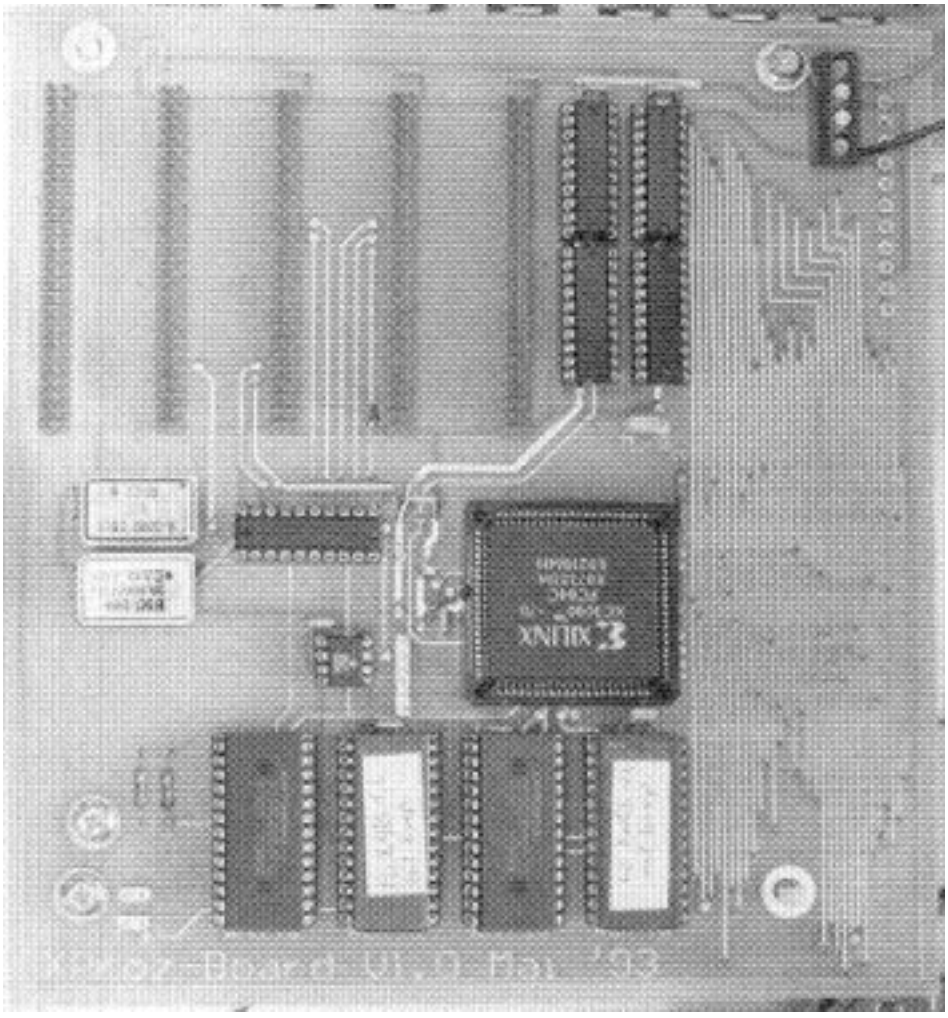


Bild 6: Hauptplatine

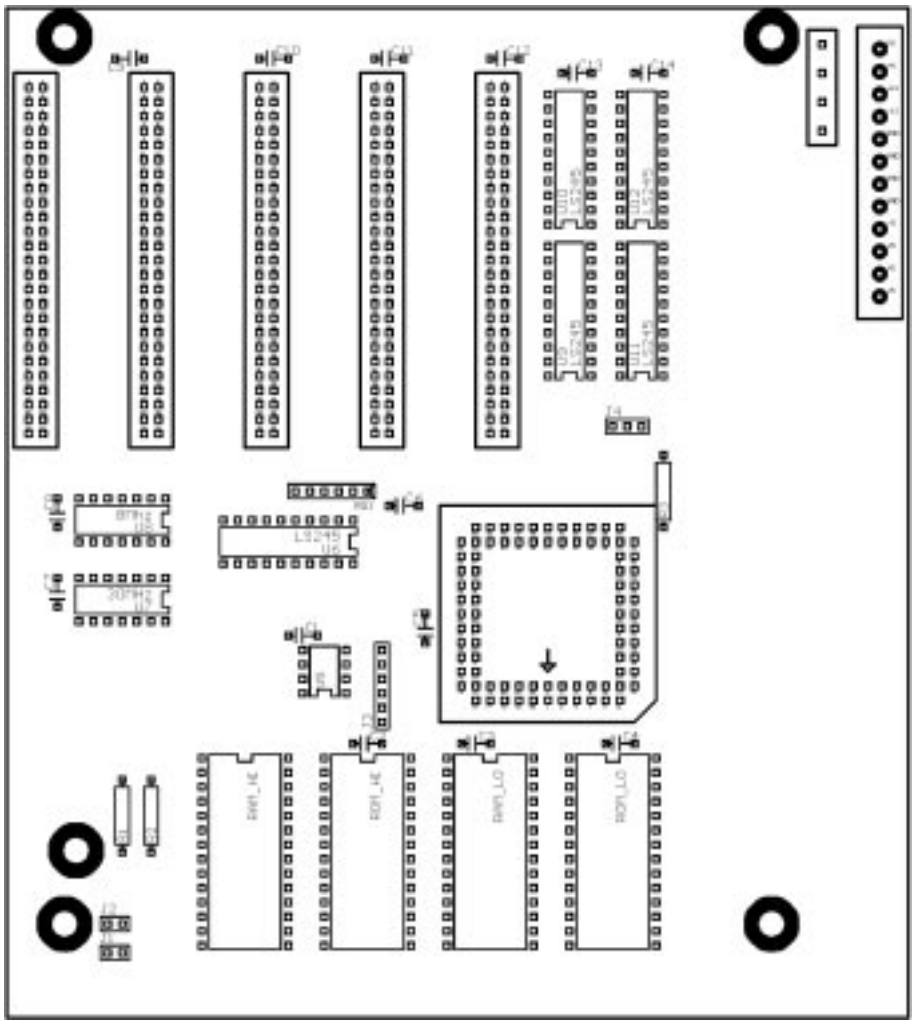


Bild 7: Bestückungsplan Hauptplatine

Nummer	Name	Typ
U1 / U3	RAM high / low	62256-70
U2 / U4	ROM high / low	27256-70
U5	Konfigurations-PROM	XC1765
U6, U9-U12	Treiberbausteine	74LS245
U7 / U8	Prozessorquarz / Slotquarz	28 MHz / 8 MHz
J1	Reset	
J2	Power LED	
J3	Download	
J4	Master/Slave	
J5	Power	
C1-C14	Entstörkondensatoren	100nF

Tabelle 26: Bestückungsliste der Hauptplatine

3.2. Prozessorsockel

Der Prozessorsockel nimmt ein XILINX-LCA-Chip vom Typ XC3090 in einem 84-poligen PLCC-Gehäuse auf. Für die Selbstkonfiguration des Chips im Master-Seriell-Mode beim Power-On befindet sich neben dem Prozessorsockel ein serielles PROM vom Typ XC1765.

Optional kann man den XILINX-Chip auch im Seriell-Slave-Mode betreiben, bei dem die Konfigurationsdaten über ein Downloadkabel von einem PC in den Chip eingeschrieben werden. Hierzu muß jedoch das serielle Prom (8-Pin-DIL-Gehäuse) aus seinem Sockel entfernt werden und der Jumper J1 auf Slave (S) gestellt werden. Nachfolgende Tabelle ist eine Übersicht über alle Prozessorsignale und der zugehörigen Pinnummern:

Signal	**In/Out**	**Pinnummer**
Stromversorgung:		
GND (Masse)	I	1, 21, 43, 65
Vcc (+5V)	I	2, 12, 22, 42, 64
Busse und Steuerung:		
A14, A13, ... , A0	O	3,82,77,78,70,60,61,52,53,45,40,35,5,27,26
D15, D14, ... , D0	IO	8,17,79,71,69,68,58,59,49,50,46,41,20,23,24,25
~IRQ5, ~IRQ4, ... , ~IRQ1	I	62, 63, 47, 48, 44
R/~W	O	4
~OE	O	6
~RAMSEL	O	81
~ROMSEL	O	83
~IOSEL	O	84
Clock (28 MHz)	I	57
~Reset	I	54
Konfiguration:		
Din	I	72
CClk	IO	74
D/~P	IO	55
M0, M1, M2	I	31, 32, 33

Tabelle 27: Prozessorsignale der Hauptplatine

3.3. Slots

Alle fünf Slots für die Erweiterungseinsteckkarten sind identisch. Sie bestehen aus dem Daten-, Adress- und Steuerbus sowie 5 Interruptleitungen und einer Leitung mit einem 8 MHz-Takt.

Bemerkung: Die Steuersignale ~RAMSEL und ~ROMSEL stehen an den Slots nicht zur Verfügung, da der gesamte Speicheradressbereich durch Speicherchips auf der Hauptplatine abgedeckt ist. Einsteckkarten können daher Register und ähnliches nur im IO-Adressbereich des Prozessors einblenden. Folgende Tabelle listet die Signale eines Slots auf:

Signal	Pinnummer
Stromversorgung:	
GND (Masse)	49,50
Vcc (+5V)	1, 2
Busse und Steuerung:	
D0, D1, ... , D15	3,4, ... , 18
~IRQ1, ~IRQ2, ... , ~IRQ5	19, 20, ... , 23
A0, A1, ... , A14	24, 25, ... , 38
~OE	39
R/~W	40
Clock (8 MHz)	41
~IOSEL	42
~RESET	43

Tabelle 28: Slotsignale der Hauptplatine

Eine Einsteckkarte muß den ganzen Adressbus (A0 bis A14) und ~IOSEL auskodieren, um Konflikte mit anderen Karten zu vermeiden. Benötigt eine Karte eine Interruptleitung, dann sollte sie mit einem TTL-kompatiblen Open-Kollektor-Ausgang auf GND gezogen werden. Somit können mehrere Karten denselben Interrupt benutzten. Die korrekte Einleseflanke für Schreibzugriffe ist der L-H-Übergang des Signals R/~W. Mit der fallenden Flanke sind Daten und Adresse noch nicht gültig!

3.4. Quarzoszillatoren

Ein Quarzoszillator versorgt direkt den Prozessor. Er arbeitet mit 28 MHz. Der andere Oszillator versorgt die 5 Slots mit einem 8 Mhz-Takt, so daß nicht jede Einsteckkarte einen eigenen Taktgenerator benötigt.

3.5. Speicherchips

Insgesamt 4 Speicherchips werden benötigt, um den ganzen Speicheradressraum des Prozessors abzudecken. Zwei davon sind RAMs, die beiden anderen sind (EP)ROMs. Es werden je zwei benötigt, da der Prozessor eine Datenbusbreite von 16 Bit hat, jeder Speicherchip aber nur 8 Bit breit ist. Die Adressen $0000 bis $7FFF werden durch die beiden RAMs (je ein 62256, 32k*8) belegt, von $8000 bis $FFFF liegen die EPROMs (je ein 27256, 32k*8).

3.6. Treiberbausteine

Auf der Hauptplatine befinden sich fünf 8 Bit breite, bidirektionale Treiberbausteine vom Typ 74LS245. Sie dienen zur Entkopplung der Slots vom Prozessor/Speicher.

Bemerkung: Lediglich zwei der fünf Treiberbausteine sind im bidirektionalen Betrieb, nämlich die für die 16 Datenleitungen. Die anderen drei sind durch feste Steuersignale nur unidirektional.

4. Grafikkarte

4.1. Technische Daten

- 640 x 400 Pixel Auflösung, 1 Bit Farbtiefe (monochrom).
- Jeder Pixel einzeln ansprechbar, echter Grafikmodus.
- 2 gleichwertige Bildschirmseiten, umschaltbar.
- Frei wählbare Vorder- und Hintergrundfarbe (Schwarz, Rot, Grün, Blau, Gelb, Lila, Türkis, Weiß)
- Hardwarescrolling (Autocopy).
- Bildwiederholfrequenz: ~73 Hz.
- Zeilenfrequenz: 30940.594 Hz, Standard-VGA.
- Pixelfrequenz: 25 MHz.
- Belegt nur 8 IO-Adressen.

4.2. Hardwarekomponenten

Das Herz der Grafikkarte bildet ein XILINX 3064-125 LCA und zwei 8-Bit breite, 15ns schnelle und 32kBytes große Speicherbausteine, einer für jede Bildschirmseite. Selbstverständlich befindet sich auf der Karte auch das obligatorische serielle PROM für die Selbstkonfiguration des XILINX-Chips sowie eine Anschlußmöglichkeit für ein Downloadkabel. Ein 25 MHz-Quarzoszillator dient zur Erzeugung des Pixeltakts.

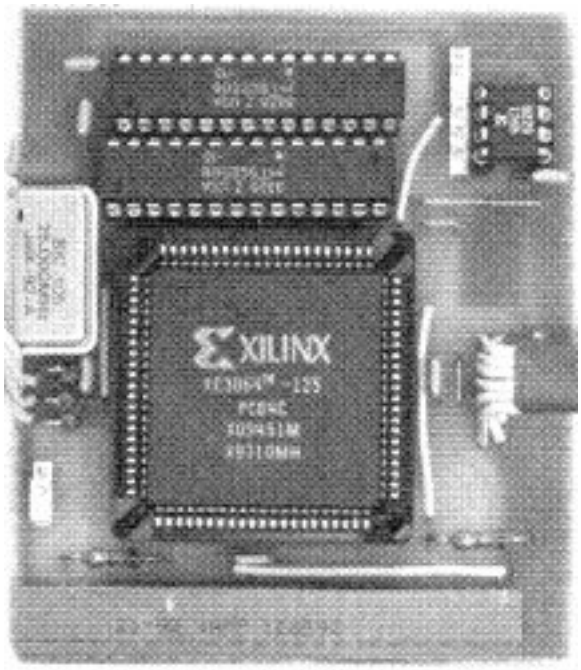


Bild 8: Grafikkarte XGraph

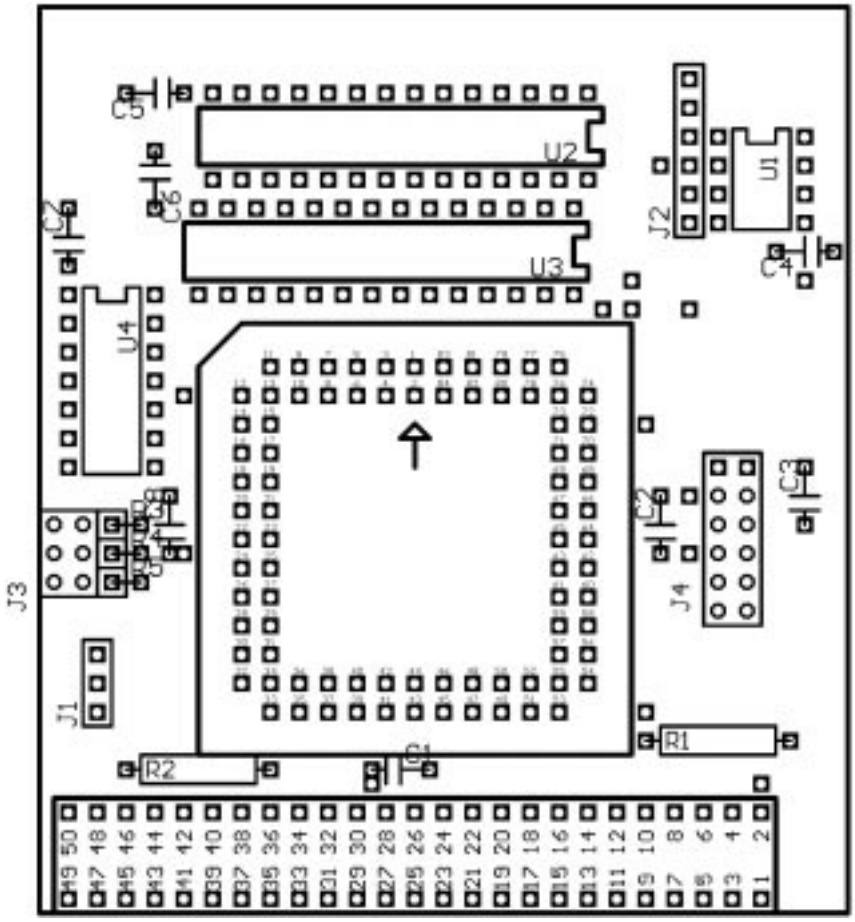


Bild 9: Bestückungsplan der Grafikkarte XGraph

Nummer	Name	Typ
U1	Konfigurationsprom	XC1765
U2, U3	Bildspeicher	5C2568-15
U4	Quarz-Ozilator	25 MHz
C1-C8	Entstörkondensatoren	100nF
R1, R2	Widerstand	2kOhm
R3-R5	Videosignal-Widerstand	900 Ohm
J1	Master/Slave	Jumper
J2	Download	6 pol. Stiftleiste, einreihig
J3	RGB-Anschluß	6 pol. Stiftleiste, doppelreihig
J4	H-, V-Sync-Anschluß	14 pol. Stiftleiste, doppelreihig
J5	XProz-Bus	50 pol. Buchsenleiste, doppelreihig

Tabelle 29: Bestückungsliste der Grafikkarte

4.3. Programmierung

Die Grafikkarte belegt 8 Register im IO-Adressbereich ab Adresse $7FE8:

#	IO-Adresse	Register beim Lesen	Register beim Schreiben
0	$7FE8	Status	Schreibzeiger low
1	$7FE9	Lesedatum	Schreibzeiger high
2	$7FEA	Status	Lesezeiger low
3	$7FEB	Lesedatum	Lesezeiger high
4	$7FEC	Status	Autocopy-Zähler low
5	$7FED	Lesedatum	Autocopy-Zähler high
6	$7FEE	Status	Control
7	$7FEF	Lesedatum	Schreibdatum

Tabelle 30: Registersatz Grafikkarte

Jedes Byte des Bildspeichers wird auf dem Bildschirm als 8 nebeneinanderliegende Punkte dargestellt, jedes Bit entspricht also einem Bildpunkt. Da eine Bildschirmzeile 640 Punkte hat belegt sie 80 Bytes im Bildspeicher. Mit den 400 Bildschirmzeilen ergibt das eine Bildspeichergröße von 32000 Bytes. Ein lineares Einblenden des Bildspeichers in den IO-Adressraum ist aufgrund dieser Größe nicht sinnvoll. Daher kann immer nur auf ein Byte zugegriffen werden. Welches Byte das ist steht im Register Schreibzeiger bei Schreibzugriffen bzw. Lesezeiger bei Lesezugriffen. Konkret sieht z.B. ein Schreibzugriff folgendermaßen aus:

```
move adress_low,@xgraph+0
move adress_high,@xgraph+1
move data,@xgraph+7
```

Bemerkung: xgraph ist die Basisadresse $7FE8 der Grafikkarte.

Nach der Übergabe des Datums wird der Schreibzeiger automatisch um 1 erhöht. Für einen Schreibzugriff auf die folgende Bildspeicheradresse ist eine Übergabe derselben dann nicht nötig.

Ein Lesezugriff erfolgt ähnlich: Leseadresse im Registerpaar 2/3 übergeben, dann das adressierte Byte aus dem Register 7 lesen. Hierbei ist zu beachten: Erst das Lowbyte der Leseadresse übergeben und dann das Highbyte (Erklärung siehe Kapitel 4.5).

Um z.B. den Bildschirminhalt um 16 Zeilen = 1280 Bytes zu verschieben, übergibt man die Schreibadresse 0 und die Leseadresse 1280. Nun muß nur noch (32000-1280)=30720 mal folgender Befehl ausgeführt werden um jeweils 1 Byte zu kopieren, die Adresspointer werden dann bei jedem move automatisch hochgezählt:

```
move @xgraph+7,@xgraph+7
```

Die Grafikkarte kann diese Datenschaufelei im Autocopy-Modus auch selbst durchführen. Dazu muß man ihr nach der Übergabe der Schreib- und der Leseadresse nur noch die Anzahl der zu kopierenden Bytes (weniger eins!) im Registerpaar 4/5 übergeben. Dabei ist wieder zu beachten, daß zuerst das Lowbyte in Register 4 und dann das Highbyte in Register 5 übergeben wird (Erklärung siehe Kapitel 4.6). Die Grafikkarte kopiert dann selbstständig den Speicherblock mit einer Geschwindigkeit von über 3 MBytes pro Sekunde. Im Statusregister (Register 0 lesend) in Bit 0 kann man abfragen, ob das Autokopieren beendet ist. Vorher sollte man keine Zugriffe auf die Grafikkartenregister machen. Ein Hardwarescrolling sieht demnach folgendermaßen aus:

```
        move        #0,@xgraph          ;Zieladr. 0 low
        move        #0,@xgraph+1        ;Zieladr. 0 high
        move        #0,@xgraph+2        ;Quelladr. 1280 low
        move        #5,@xgraph+3        ;Quelladr. 1280 high
        move        #255,@xgraph+4      ;Anzahl=30719 low
        move        #119,@xgraph+5      ;Anzahl=30719 high
wait:   and    sf   @xgraph,#1,-        ;Warten...
        jne         wait
```

Schließlich gibt es noch Kontrollregister, in denen man Vorder- und Hintergrundfarbe einstellen sowie die sichtbare und aktive Bildschirmseite wählen kann. Die aktive Seite ist die, auf die der Prozessor Schreib- und Lesezugriff hat. Für diese vielfältigen Funktionen liegen auf der Adresse $7FEE mehere Kontrollregister. Die oberen 4 Bit des Übergabebytes wählen das Kontrollregister aus, die unteren 4 Bit werden in dieses gewählte Register geschrieben.

#	gültige Werte	Register beim Schreiben
0	0 bis 7	Hintergrundfarbe
1	0 bis 7	Vordergrundfarbe
2	0 und 1	Sichtbare Bildschirmseite
3	0 und 1	Aktive Bildschirmseite

Tabelle 31: Kontrollregister der Grafikkarte

Um z.B. die Vordergrundfarbe auf Weiß zu setzen schreibt man:

```
        move #$17,$7FEE
```

4.4. Videosignale

Die Karte erzeugt 5 Signale zur Ansteuerung eines Standard-VGA-Monitors:

- HSync: Synchronisationsignal für horizontalen Strahlrücklauf mit TTL-Pegel.
- VSync: dasselbe für den vertikalen Strahlrücklauf.
- R: Das Videosignal für die Farbe Rot mit einem Pegel von 0 bis 0.7 Volt.
- G: dasselbe für Grün.
- B: und Blau.

Ein Horizontalzyklus setzt sich aus 3 Vorgängen zusammen:

- Display-Phase, in der die 640 Pixel dargestellt werden.
- Dunkeltast- oder Blankphase, in der die RGB-Signale abgeschaltet sind (0 Volt)

- Synchronisation, wobei das HSync-Signal kurz auf High gezogen wird.

Entsprechendes gilt für den Vertikalzyklus. Die Displayphase beträgt hier 400 Zeilen.

Die folgende Grafik zeigt den zeitlichen Verlauf der horizontalen (bzw. vertikalen) Synchronisations- und Blanksignale bezüglich Zeichennummer (bzw. Zeilennummer). Zur Veranschaulichung ist noch der mögliche Bereich der Videosignale während eines Zykluses dargestellt. In der HBlank- und der VBlank-Phase sind sie fest auf 0 (ergibt schwarz) gezogen:

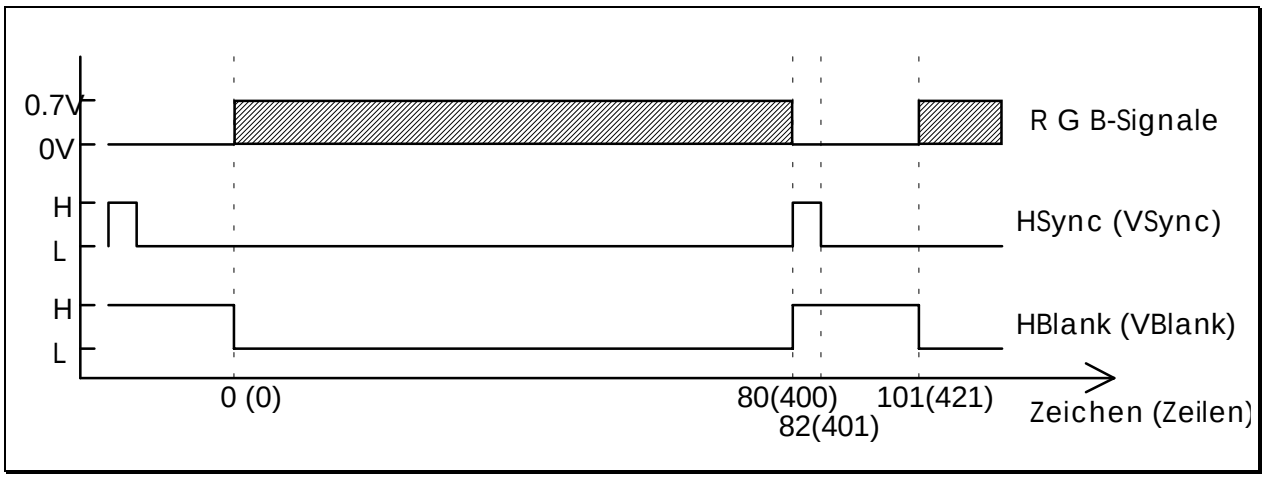


Bild 10:Videosignale

4.5. Speicherzugriffszyklen

Die Aufgabe des Grafik-Chips läßt sich prinzipiell zweiteilen: 1. Erzeugung der Videosignale und 2. Interface zwischen Bildspeicher und XProz. Über ein Zeitmultiplexverfahren können beide (Bildschirm und Prozessor) auf den Bildspeicher zugreifen. Dazu wird der Pixeltakt von 25 MHz durch Division durch 8 zum Zeichentakt, sein Frequenz beträgt also 3.125 MHz. Während der Darstellung der 8 Pixel eines Zeichentaktes erfolgen maximal 3 Zugriffe auf den Bildspeicher:

Pixel/Phase	Zugriffszyklus	Adressregister	Daten vom	Daten zum
0-2	0: Prozessor-Lesen	Lesezeiger	Bildspeicher	Datenregister
3-4	1: Prozessor-Schreiben	Schreibzeiger	Datenregister	Bildspeicher
5-7	2: Video-Lesen	Videozeiger	Bildspeicher	Videoshifter

Tabelle 32: Bildspeicherzugriffszyklen

Bemerkung: Die Schreibphase ist etwas kürzer, weil dabei Daten und Adressen nur zum Speicher müssen. Bei Lesezugriffen müssen die Adressen zum Speicher und erst dann kommen die Daten zurück.

Die Zugriffszyklen werden nach folgenden Regeln aktiviert:

- Wenn der Prozessor ein Byte in das Datenregister einschreibt, so wird einmal Zyklus 1 aktiv, um dieses Byte in den Bildspeicher zu schreiben und den Schreibzeiger um eins zu erhöhen, und dann wird einmal Zyklus 0 aktiv, um das Datenregister mit dem nächsten Lesebyte zu füllen und den Lesezeiger ebenfalls um eins zu erhöhen.

- Wenn der Prozessor das Highbyte einer Leseadresse in den Lesezeiger-High eingeschrieben hat, wird einmal Zyklus 0 aktiv, um das Datenregister mit dem gewünschten Datum aus dem Bildspeicher zu füllen und dann den Lesezeiger um eins zu erhöhen.

- Befindet sich der Grafikchip im Autocopymode, so sind Zyklus 0 und 1 immer aktiv, unabhängig von Prozessorzugriffen auf irgendwelche Register.

- Zyklus 2 ist immer aktiv.

Das Erhöhen des Videozeigers erfolgt in Phase 1, wenn HBlank Low ist. Das Einschreiben des Videobytes in den Videoshifter erfolgt in Phase 7. Das Erhöhen des Lesezeigers erfolgt gleichzeitig mit dem Einschreiben des Lesedatums in das Datenregister in Phase 2. Das Erhöhen der Schreibzeigers erfolgt gleichzeitig mit dem Einschreiben des Datums in den Bildspeicher in Phase 4.

4.6. Schaltungsaufbau

Die Schaltung ist bis auf zwei Filpflops als synchrones Schaltwerk aufgebaut. Diese beiden dienen zur Verlängerung des Prozessorschreibsignals, bis die dadurch aktivierten Zugriffszyklen ausgeführt werden können.

Phasengenerator Rot8: generiert vier Signale Q0-Q3, aus denen dann mittels AND-Gattern beliebige Phasenzeiten auskodiert werden können. Mit jeder steigenden Flanke des Clock-Signals (=Pixeltakt mit 25MHz) geht der Generator in den nächsten Zustand über.

Zustand n^t	Q0	Q1	Q2	Q3	n^{t+1}
0	0	0	0	0	1
1	1	0	0	0	2
2	1	1	0	0	3
3	1	1	1	0	4
4	1	1	1	1	5
5	0	1	1	1	6
6	0	0	1	1	7
7	0	0	0	1	0

Tabelle 33: Ausgangssignale des Phasengenerators

Sync-Signal-Generator Sync: erzeugt direkt die beiden Video-Ausgangssignale HSync und VSync sowie die Signale HBlank und VBlank zur Dunkeltastung der Videosignale RGB. Das Schaltwerk beinhaltet zwei ähnlichen Generatoren, einen für die HSync/HBlank-Signale (H-Generator) und einen für dir VSync/VBlank-Signale (V-Generator). Jeder Generator besteht aus einem Zähler (7- bzw. 9-Bit) mit synchronem Reset sowie mehreren Vergleichern, die bei bestimmten Zählerständen verschiedene Flipflops ein- und ausschalten bzw. den Zähler rücksetzen. Der H-Generator wird nur in Phase 7 aktiv, der V-Generator nur dann, wenn Phase 7 aktiv ist und der H-Generator sich in Zustand 100 befindet:

Zustand n^t	HSync	HBlank	n^{t+1}
0...79	0	0	n+1
80, 81	1	1	n+1
82...99	0	1	n+1
100	0	1	0

Tabelle 34: Ausgangssignale des H-Generators

Zustand n^t	VSync	VBlank	n^{t+1}
0...399	0	0	n+1
400	1	1	401
401...420	0	1	n+1
422	0	1	0

Tabelle 35: Ausgangssignale des V-Generators

Videozeiger VCounter: ein einfaches synchrones 15-Bit Zählregister mit asynchronem Reset. Das Inkrementwerk ist ein einfacher Ripple-Addierer, da das Register nur mit einer Frequenz von 3.125 MHz zählen muß. Er inkrementiert in Phase 1, aber nicht während einer horizontalen Dunkeltastphase. Da in dieser kein Bild dargestellt wird, soll sich der Videozeiger auch nicht ändern. Als Resetsignal wird das VBlank-Signal benutzt, damit bei jedem neuen Bildaufbau immer bei Adresse 0 angefangen wird.

Count	Reset	Q^{t+1}
0	1	0
1	0	Q^t+1

Tabelle 36: Übergangstabelle des Videozeigers VCounter

Adresszeiger ACounter: ein sychroner 15-Bit-Zähler mit der Möglichkeit, das Low- oder das Highbyte parallel zu laden. Dazu gibt es drei Steuerleitungen: Count, Loadl und Loadh. Maximal eine davon darf zu einem Zeitpunkt 1 sein. Mit der nächsten steigenden Taktflanke wird dann die gewählte Funktion ausgeführt. Das Flipflop zur Ansteuerung der 15 Multiplexer ist nötig, weil die Signale Loadl und Loadh asynchron zum FF-Takt kommen und die Flipflops in einen metastabilen Zustand fallen könnten. Dieses Register ist übrigens zweimal verwendet: einmal als Leseadresszeiger und einmal als Schreibadresszeiger.

Count	Loadl	Loadh	Q^{t+1} (bzw. Q_i^{t+1})
0	0	0	Q^t
1	0	0	Q^t+1
0	1	0	Q_i^t für i=8...14 D_i^t für i=0...7
0	0	1	D_{i-8}^t für i=8...14 Q_i^t für i=0...7

Tabelle 37: Übergangstabelle des Adresszeigers ACounter

Adressmultiplexer AMux: schaltet einen der drei 15-Bit breiten Eingänge A,B oder C abhängig von den Steuerleitungen S0 und S1 auf den Ausgang Q durch. Bei bestimmten Phasen werden mit diesem Mux entweder der Videozeiger oder einer der beiden Adresszeiger an den Adressbus des Bildspeichers gelegt:

Phase	durchgeschaltet	S1	S0	Q
5-7	Videozeiger	0	0	A
3-4	Schreibzeiger	0	1	B
0-2	Lesezeiger	1	0	C
-	-	1	1	0

Tabelle 38: Ausgangsfunktion des Adressmultiplexers

Datenregister MuxReg8: ein 8-Bit breites Datenregister mit vorgeschaltetem Multiplexer, um Daten von zwei Bussen in das Register schreiben zu können. Mit EA=1 wird mit der nächsten steigenden Taktflanke von Eingang A in das Register eingeschrieben, entsprechend für EB=1 von Eingang B. Sind EA=EB=0, dann hält das Register seinen Wert. Der Prozessor kann dieses Register jederzeit auslesen. Hininschreiben kann er über Eingang A. Über Eingang B kommen die Lesedaten vom Bildspeicher.

EA	EB	Q^{t+1}
0	0	Q^t
1	0	A^t
0	1	B^t

Tabelle 39: Übergangstabelle des Datenregisters

Videoshifter VShifter: ein 8-Bit breites unidirektionales Schieberegister mit der Möglichkeit des parallelelen Ladens. Ist Load=0, dann schiebt sich das Datum bei der nächsten steigenden Taktflanke um ein Bit nach rechts. Wenn Load=1 ist, dann werden mit der nächsten Taktflanke die Daten am Eingangsbus D in das Register eingeschrieben.

Load	Q_i^{t+1}
0	Q_{i+1}^t für i=0...6 0 für i=7
1	D_i^t

Tabelle 40: Übergangstabelle des Videoshifters

Kontrollregister Control: enthält 8 Flipflops, die mit der fallenden Flanke von Load gesetzt werden. Welche Flipflops mit welchen Daten bei bestimmten Eingangswerten belegt werden, zeigt folgende Tabelle:

D[7:4]	RB	GB	BB	RV	GV	BV	VISUAL	ACTIVE
0	D0	D1	D2	-	-	-	-	-
1	-	-	-	D0	D1	D2	-	-
2	-	-	-	-	-	-	D0	-
3	-	-	-	-	-	-	-	D0

Tabelle 41: Übergangstabelle des Kontrollregisters (- bedeutet unverändert)

Autocopyzähler Downcnt: ein parallel ladbarer 15-Bit Abwärtszähler. Sein Aufbau entspricht im Wesentlichen dem ACounter, nur daß er rückwärts zählt. Der Ausgang geht mit dem Einschreiben eines Datums in den High-Teil des Registers auf 1. Beim Zählerunterlauf (von 0 auf -1) geht der Ausgang dann wieder auf 0 zurück.

Count	Loadl	Loadh	Q^{t+1} (bzw Q_j^{t+1})	Go^{t+1}
0	0	0	Q^t	Go^t
1	0	0	Q^t -1	0 für Q^t=0 1 sonst
0	1	0	Q_i^t für i=8...14 D_i^t für i=0...7	Go^t
0	0	1	D_{i-8}^t für i=8...14 Q_i^t für i=0...7	1

Tabelle 42: Übergangstabelle des Autocopyzählers Downcnt

Schnittstellensysmbole XV_Mem, XDBus und XABus: hier sind nur IO-Puffer und IO-Pads definiert, um die Hauptschaltung übersichtlicher zu halten. Die folgende Tabelle zeigt die komplette Pinbelegung des Chips:

Signal	In/Out	Pinnummer
Stromversorgung:		
GND (Masse)	I	1, 21, 43, 65
Vcc (+5V)	I	2, 12, 22, 42, 64
Slotschnittstelle:		
A14, A13, ... , A0	I	29,28,36,30,35,38,37,40,39,46,41,47,44,48,45
D7, D6, ... , D0	IO	53,50,57,51,59,52,58,56
R/~W	I	27
~OE	I	26
~IOSEL	I	25
Bildspeicherschnittstelle:		
MA14, MA13, ... , MA0	O	73,77,76,83,5,81,79,78,80,82,84,4,6,7,9
MD7, MD6, ... , MD0	IO	8,10,15,17,18,16,14,11
~WE1	O	75
~WE2	O	71
~OE1	O	3
~OE2	O	70
Diverse:		
Clock (25MHz)	I	13
HSync, VSync	O	67, 66
R, G, B	O	20, 23, 24
Konfiguration:		
Din	I	72
CClk	IO	74
D/~P	IO	55
M0, M1, M2	I	31, 32, 33

Tabelle 43: Pinbelegung des Grafikchips

5. Timer-, SCSI- und Tastaturschnittstellenkarte

5.1. Aufbau

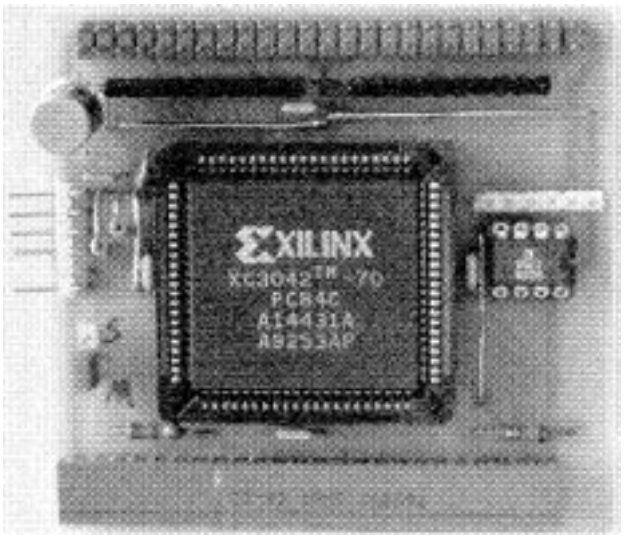


Bild 11: Timer, SCSI- und Tastaturkarte

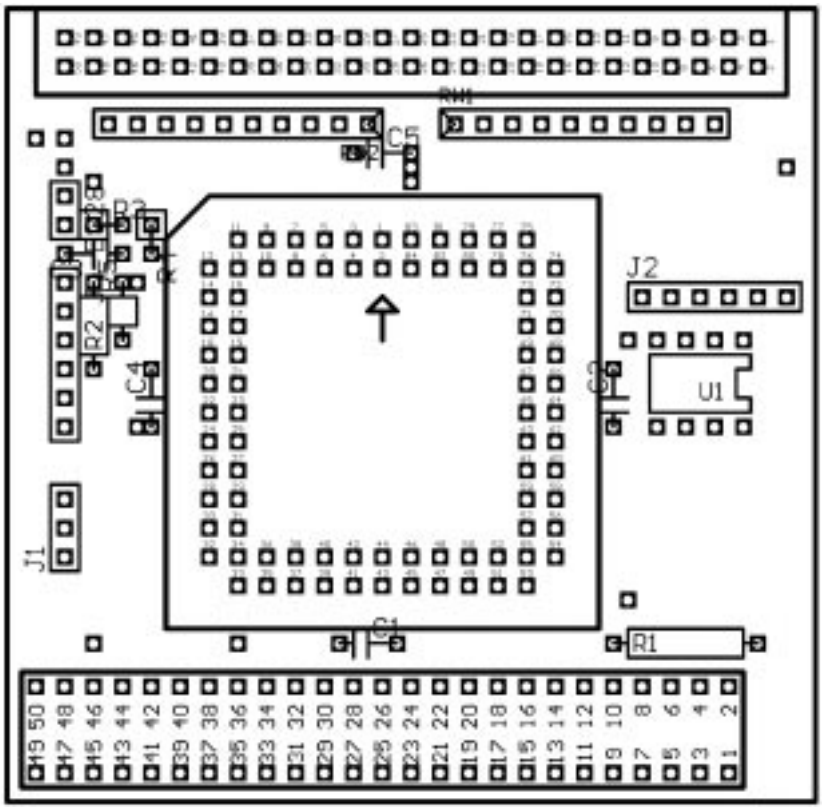


Bild 12: Bestückungsplan der Timer, SCSI- und Tastaturkarte

Nummer	Name	Typ
U1	Konfigurationsprom	XC1765
T1	Transistor	NPN-Darlington
C1-C5	Entstörkondensatoren	100nF
R1-R5	Widerstand	2kOhm
J1	Master/Slave	Jumper
J2	Download	6 pol. Stiftleiste, einreihig
J3	SCSI-Anschluß	50 pol. Stiftleiste, doppelreihig
J4	Tastaturanschluß	6 pol. Stiftleiste, einreihig
J5	XProz-Bus	50 pol. Buchsenleiste, doppelreihig
RW1, RW2	Widerstandsnetzwerk	2kOhm * 9

Tabelle 44: Bestückungsliste der Timer-, SCSI- und Tastaturschnittstellenkarte

5.2. SCSI-Schnittstelle XSCSI

5.2.1. Allgemeines

Das SCSI (Small Computer System Interface) ist ein Bussystem um bis zu 8 Geräten miteinander zu verbinden. Dies sind bei SCSI-1 blockorientierte Massenspeicher (Festplatte, Diskette) und Hostadapter zu Rechnersystemen. Der Datentransfer erfolgt beim SCSI 8-Bit-parallel und synchronisiert mit 2 Handshakeleitungen. Eine Übertragung findet immer nur zwischen 2 Geräten statt. Die Adressierung (man spricht bei SCSI von Selektierung) erfolgt vor dem eigentlichen Transfer und bleibt bis zu seinem Ende bestehen.

Jeder Transfer beginnt mit einem Kommando, gefolgt von der Datenübertragung und schließlich einer Ergebnisphase.

5.2.2. Hardwarespezifikation

Grundsätzlich unterscheidet man zwischen 2 Arten von SCSI-Geräten: Initiatoren und Targets. Ein Datentransfer findet immer zwischen einem Initiator und einem Target statt. Meistens startet ein Initiator einen Transfer. Üblicherweise gibt es in einem SCSI-System genau einen Initiator (Rechner mit Hostadapter) und mehrere Targets (Laufwerke). Jedes Target muß eine eindeutige Nummer (ID) zwischen 0 und 7 haben. Laufwerke haben zu diesem Zweck üblicherweise 3 Jumper, über die man binär kodiert deren ID einstellen muß. Auch Initiatoren benötigen eine eindeutige ID, aber nur wenn die optionale Disconnect-Reconnect-Fähigkeit des SCSI-Busses benutzt werden soll. Hierbei können auch Targets einen Transfer initiieren und müssen dazu einen Initiator über dessen ID selektieren. Diese Option ist beim XSystem nicht implementiert, da der Aufwand hierfür in keinem günstigen Verhältnis zum Nutzen steht. Daher benötigt die XSCSI-Karte keine ID.

Der SCSI-Bus selbst ist üblicherweise ein 50-poliges Flachbandkabel mit aufgequetschten, doppelreihigen Buchsenleisten.

Pinnummer	Signalname	Beschreibung
2	~DB0	Datenbit 0
4	~DB1	Datenbit 1
6	~DB2	Datenbit 2
8	~DB3	Datenbit 3
10	~DB4	Datenbit 4
12	~DB5	Datenbit 5
14	~DB6	Datenbit 6
16	~DB7	Datenbit 7
18	~DBP	Datenparität (ungerade)
26	TERMPWR	Terminatorpower +5V
32	~ATN	Attention
36	~BSY	Busy
38	~ACK	Acknowledge
40	~RST	Reset
42	~MSG	Message
44	~SEL	Select
46	~C/D	Command/Data
48	~REQ	Request
50	~I/O	Input/Output
25	-	Offen (no connection)
Rest	GND	Signalmasse

Tabelle 45: Pinbelegung SCSI-Bus

Es fällt auf, daß (fast) alle ungeraden Pins auf Masse liegen. Dadurch sind die Signalleitungen recht gut abgeschirmt.

Nach der SCSI-Norm müssen die beiden Endgeräte des Busses diesen mit einer 220 bzw. 330 Ohm-Terminierung gegen Vcc = 5V bzw. GND abschließen. Damit ergibt sich ein Ruhepegel von ca. 3V für High. Um ein Low auf eine Leitung zu bringen muß nun ein Gerät mit einem Open-Collector-Ausgang diese auf GND ziehen. Wegen der beiden Terminatoren muß dieser Treiber aber 50mA aufnehmen und darf dabei höchstens einen Pegel von 0.5V auf dem Bus lassen. Die Ausgangstreiber des XILINX-Chips erfüllen diese Bedingung leider nicht (maximal 9mA). Da aber ein solcher Chip als Hostadapter vorgesehen war, gab es nur 2 mögliche Lösungen: 1. Ein zusätzlicher Treiberbaustein auf dem Hostadapter oder 2. Die Abschlußwiderstände entgegen der SCSI-Norm vergrößern. Die Entscheidung fiel auf die einfachere 2. Lösung, da ihr einziger Nachteil eine Herabsetzung der maximalen Übertragungsrate war und diese vom XProz-Prozessor sowieso nicht erreicht werden kann. Der SCSI-Bus wird also im XSystem mit 2kOhm gegen V_{CC} abgeschlossen, gegen GND erfolgt keine Terminierung. Dieses Vorgehen hat einen weiteren Vorteil, nämlich die Erhöhung des Ruhepeges auf 5V, was den Eingangsstufen des XILINX-Chips eine sichere Erkennung von logisch High gewährleistet.

Interessant ist die Paritätsprüfung bei SCSI. Jeder, der ein Datenbyte auf den Bus legt, muß auch das Paritätssignal korrekt bedienen. Der Empfänger des Bytes kann damit Einzelbitfehler sofort erkennen.

5.2.3. Busphasen und Handshaking

Der SCSI-Bus befindet sich immer in einer der folgenden Phasen:

- Bus-Frei-Phase: In dieser Phase befindet sich der Bus nach einem Reset und nach Abschluß eines Transfers. Alle SCSI-Signale sind dabei inaktiv, das heißt durch die Abschlußwiderstände liegen sie auf logisch 1.

- Arbitrierungsphase: Möchte ein Initiator mit einem Target Daten austauschen, so muß er erst den Bus für sich in dieser Phase "gewinnen", da andere Initiatoren auch am Bus Interesse haben können. Gibt es jedoch (wie beim XSystem) nur einen Initiator, so kann diese Phase übersprungen und sofort zur folgenden Phase gegangen werden:

- Selektionsphase: hier wählt der Initiator ein Target aus, indem er die Datenleitung ~Di (mit i = gewünschte Target-ID) und das Select-Signal aktiviert (auf low zieht). Das Target muß nun seinerseits das Busy-Signal aktivieren. Wenn dies beim Initiator ankommt, deaktiviert er wieder ~SEL und die Datenleitung. ~BSY bleibt dann vom Target bis zum Transferende aktiviert und zeigt somit an, daß der SCSI-Bus belegt ist.

- Reselektionsphase: Diese Phase wird im XSystem nicht verwendet. Mit ihrer Hilfe könnte man einen SCSI-Befehl unterbrechen, der längere Zeit benötigt, um vom Target ausgeführt zu werden. Ist das Target fertig, selektiert es seinerseits den Initiator in dieser Phase. Der Initiator müßte also eine eigene ID haben und auf die Selektion rasch reagieren.

- Transferphase: Unmittelbar nach der Selektion beginnt die Transferphase. Es können mehrere Arten von Daten byteweise übertragen werden: Kommando, Daten, Status und Message. Was übertragen werden soll entscheidet immer das Target (!) mit den Signalen ~C/D, ~I/O und ~MSG, der Initiator reagiert nur auf diese Signale.

~MSG	~C/D	~I/O	Bezeichnung	Daten vom	zum
1	1	1	Data out	Initiator	Target
1	1	0	Data in	Target	Initiator
1	0	1	Command	Initiator	Target
1	0	0	Status	Target	Initiator
0	0	1	Message out	Initiator	Target
0	0	0	Message in	Target	Initiator
0	1	1	(nicht erlaubt)	-	-
0	1	0	(nicht erlaubt)	-	-

Tabelle 46: Informationstransferphasen bei SCSI

Die Übertragung eines Bytes funktioniert dann immer nach dem folgenden Schema: Der Initiator wartet, bis das Target das Request-Signal aktiviert. Dann holt er bei ~I/O = 0 das Byte vom Bus bzw legt bei ~I/O = 1 das nächste Byte auf den Bus. Danach zieht er Acknowledge auf low, um die Übertragung des Bytes zu bestätigen. Das Target nimmt dann Request zurück. Daraufhin muß der Initiator Acknowledge ebenfalls zurücknehmen. Dies wiederholt sich so Byte für Byte.

Wenn das Target das Busy-Signal zurücknimmt, dann ist das Kommando beendet und der Bus befindet sich wieder in der Bus-Frei-Phase.

5.2.4. Programmierung

Die XSCSI-Schnittstelle belegt lediglich zwei IO-Adressen:

Adresse	Register lesen	Register schreiben
$7FF4	SCSI-Daten in	SCSI-Daten out
$7FF5	Status	Control

Tabelle 47: Register der XSCSI-Schnittstelle

Bit	Bedeutung
0	Message
1	Command/Data
2	In/Out
3	Request
4	Busy
5	Parity-Error

Tabelle 48: Statusregister der XSCSI-Schnittstelle

Bit	Bedeutung
0	Reset
1	Attention

Tabelle 49: Kontrollregister der XSCSI-Schnittstelle

Über das Kontrollregister Bit 0 läßt sich das Reset-Signal vom SCSI-Bus setzen, um alle Targets zu initialisieren. Für einen Reset muß man das Bit setzen, einige Millisekunden warten und es dann wieder löschen. Entsprechend bedient Bit 1 direkt die Attention-Leitung. Sie hat jedoch beim XSystem keine Bedeutung und ist nur der Vollständigkeit halber implementiert und sollte deshalb inaktiv bleiben. Der Zweck des Kontrollregisters ist daher nur das Rücksetzen des SCSI-Busses bei der Systeminitialisierung.

Ausgehend von einer Bus-Frei-Phase erfolgt ein SCSI-Zugriff folgendermaßen:

1. Selektierung des Targets, indem man 2^{id} in das Datenregister schreibt (das Paritybit wird vom XSCSI-Adapter erzeugt) und damit eine der 8 Datenleitungen aktiviert. Da sich der SCSI-Bus in der Bus-Frei-Phase befindet, wird das Select-Signal vom Adapter automatisch mitaktiviert und das adapterinterne Parity-Error-Flag wird gelöscht.
2. Warten auf Belegung des SCSI-Bus durch das Target, indem man im Statusregister

Bit 4 auf eine 1 wartet (dieses Bit spiegelt das ~BSY-Signal wieder). Bemerkung: Wenn nach zirka einer Sekunde das Target immer noch nicht reagiert, dann ist es nicht vorhanden (oder kaputt) und man muß noch eine 0 ins Datenregister schreiben, um die aktivierte Datenleitung und das Select-Signal wieder zu deaktivieren.

3. Das Select-Signal wird durch das aktivierte Busy-Signal automatisch vom Adapter zurückgenommen.
4. Warten auf ein (Byte-)Request vom Target (eine 1 im Statusregister Bit 3).
5. Abhängig von Bit 0,1 und 2 im Statusregister ein Byte in das Datenregister schreiben bzw. ein Byte aus dem Datenregister lesen. Das nötige Byte-Handshaking übernimmt der Adapter.
6. Solang Busy aktiv ist, Schritte 4 und 5 wiederholen.
7. Abfrage des Parity-Error-Bit im Statusregister. Falls irgendwann bei der Übertragung ein Paritätsfehler aufgetreten ist, dann steht hier eine 1 und man sollte die ganze Übertragung wiederholen.

Nach erfolgreicher Selektion des Target fordert dieses erst mal ein Kommando an. Im Anhang befindet sich eine Übersicht der SCSI-Kommandos. Sie haben eine Länge von sechs oder zehn Bytes. Nach Übertragung des letzten Kommandobytes erfolgt die eigentliche Datenübertragung. Am Ende werden dann noch ein Status- und ein Messagebyte vom Target übermittelt, über die der Erfolg der Kommandoausführung kontrolliert werden kann.

5.2.5. Schaltungsaufbau

Die Schaltung des XSCSI-Adapters besteht im wesentlichen aus einem 8-Bit breiten Register (dreg8), welches das auszugebende Byte an den SCSI-Bus enthält. Dieses wird jedoch nur bei ~I/O=1 ausgegeben, bei ~I/O=0 sind die Ausgangstreiber des XILINX-Chips im hochohmigen Zustand. Es wurden keine flankengetriggerten D-Flip-Flops zur Speicherung der Daten verwendet, da mit der steigenden Flanke der R/~W-Signals des Prozessors das Acknowledge-Bit für das Target gesetzt wird. Zu diesem Zeitpunkt müssen aber die Datenbits und auch das Parity-Bit schon stabil beim Target anliegen. Die Daten müssen also schon früher ausgegeben werden. Daher werden die Datenbits des Prozessors während der gesamten Schreibphase (R/~W=0) direkt an den SCSI-Bus durchgeschleust.

Eine weitere Funktionseinheit bildet das Symbol PARITY. Diese Schaltnetz generiert zu den ausgegebenen bzw. eingelesenen Bytes das korrekte Parity-Bit. Bei Datenausgabe (~I/O=1) wird dieses dann auf die Parity-Leitung vom SCSI-Bus gelegt.

Bemerkung: Sowohl bei Dateneingabe als auch bei der Ausgabe wird dieses generierte Parity-Bit mit dem auf dem SCSI-Bus verglichen. Sollten diese einmal nicht übereinstimmen, wird das Parity-Error-Bit gesetzt. Gelöscht werden kann es dann erst wieder durch eine erneute Selektionsphase.

Die Schaltung ZERO dient dazu, bei mißlungener Selektion durch Einschreiben einer 0 in das Datenregister das Select-Signal wieder zurückzunehmen. Dieses wird nämlich vom Adapter automatisch gesetzt, wenn ein Wert ungleich 0 in das Datenregister geschrieben wird, während das Busy-Signal inaktiv ist, der SCSI-Bus also frei ist.

Die Schaltung FILTER dient zum ausfiltern kurzer 0-1-0 Störimpulse auf der Request-Leitung. Diese würden nämlich fälschlicherweise das Acknowledge-Signal zurücknehmen, ohne daß das Target das Request-Signal zurückgenommen hat.

5.3. Tastaturschnittstelle XKey

5.3.1. Allgemeines

An die Schnittstelle läßt sich eine handelsübliche MF2-Tastatur anschließen. Dieser Tastaturtyp ist leicht zu beschaffen, er findet sich nämlich an fast jedem PC wieder. Eine solche Tastatur besitzt einen eigenen Microcontroller, der selbstständig die Tastaturmatrix abfragt. Beim Drücken einer Taste sendet der Controller einen eindeutigen Tastencode, den sogenannten Scan-Code (kein ASCII-Code!). Sogar die automatische Tastenwiederholung erzeugt der Controller, er sendet nämlich den Tastencode in einer bestimmten Geschwindigkeit immer wieder, solange die Taste noch gedrückt ist.

Die Tastatur nimmt auch Kommandos entgegen, um z.B. die 3 Leuchtdioden (Caps-Lock, Scroll-Lock, Num-Lock) ein- und auschalten zu können oder aber die Tastenwiederholgeschwindigkeit einzustellen.

5.3.2. Die serielle Schnittstelle

Die Tastatur sendet und empfängt Daten über eine synchrone, bidirektionale, serielle Schnittstelle. Die Übertragungsrate beträgt zirka 20000 Baud, kann jedoch von Tastatur zu Tastatur variieren. Der Anschluß erfolgt über einen 5-poligen runden DIN-Stecker.

Pin	Bedeutung
1	Clock
2	Data
3	NC, nicht angeschlossen
4	GND, Ground
5	Vcc, +5 Volt
Rand	Gehäusemasse

Tabelle 50: Pinbelegung des Tastatursteckers

Der Ruhepegel von Clock und Data ist über 2kOhm-Pullup-Widerstände auf beiden Seiten des Kabels High. Sowohl Adapter als auch Tastatur können über Open-Kollektor-Ausgänge die Leitungen auf Low ziehen. Mit jeder steigenden Flanke von Clock wird ein Datenbit übertragen. Auch bei Senden von Daten gibt die Tastatur den Takt an, dann muß der Adapter mit jedem Takt das nächste Bit auf die Datenleitung legen. Eine SDU (Serial-Data-Unit) besteht bei der MF2-Tastatur aus 11 Bit: 1 Startbit (low), 8 Datenbit, 1 Paritybit (even) und 1 Stoppbit (high). Die Kommandos und die Tastaturtabelle befinden sich im Anhang.

5.3.3. Besonderheiten

Die bei der Diplomarbeit verwendete Tastatur hat eine besondere Eigenschaft, die sich erst nach dem Aufbau der Keyboard-Karte bemerkbar machte: Sie fällt nach einem Power-On selbstständig in den für XTs nötigen 9-Bit-Modus, das heißt eine SDU besteht nur noch aus Startbit und den 8 Datenbits. Dies kann nur verhindert werden, indem der Adapter die Clockleitung kurzzeitig auf low zieht, nachdem die Tastatur ihr Power-On-Byte ($aa) gesendet hat. Sie sendet dieses noch im 11-Bit-Modus. Prinzipell hätte der 9-Bit-Modus keine Nachteile. Die Tastatur läßt sich dann jedoch nicht mehr programmieren. Sie ist nur noch durch erneutes Aus- und Einschalten wieder in den 11-Bit-Modus zu bekommen.

Bemerkung: Fast alle anderen Tastaturen verfügen über einen kleinen Schalter auf der Rückseite, über den man die Tastatur fest in den 11- bzw. 9-Bit-Modus schalten kann. Solchen Tastaturen muß man nicht "zeigen", daß sie nicht an einem alten XT-Computer angeschlossen sind.

Leider braucht aber der XILINX-Chip auf der Keyboardkarte länger, um sich zu konfigurieren, als der Prozessor. Der Benutzer des Systems muß daher nach dem Einschalten kurz warten und dann einmal den Reset-Knopf betätigen. Zu diesem Zeitpunkt befindet sich aber die Tastatur schon im 9-Bit-Modus. Das Betriebssystem des Rechners muß also softwaremäßig die Möglichkeit haben, die Tastatur aus- und wieder einzuschalten. Dafür wurde einfach der Transistor benutzt, der ursprünglich für die Festplatten-LED im C-Gehäuses vorgesehen war. Diese LED sollte Aktivitäten auf dem SCSI-Bus anzeigen. Aus diesem Grund befinden sich auf der Platine einige Kabelbrücken, da die Stromversorgung der Tastatur nun über diesen Transistor geführt werden mußte.

5.3.4. Programmierung

Die Schnittstelle belegt lediglich zwei IO-Adressen:

Adresse	Register lesen	Register schreiben
$7FF6	Data in	Data out
$7FF7	-	Control

Tabelle 51: Register der XKEY-Schnittstelle

Möchte man der Tastatur ein Byte senden, dann muß man es nur in das Datenregister schreiben. Wenn ein Byte von der Tastatur angekommen ist, dann löst die Karte einen Interrupt 1 aus. In der Interruptbehandlungsroutine muß dann das Byte vom Datenregister abgeholt werden. Dies ist gleichzeitig die Interruptbestätigung an die Karte und sie nimmt das Interruptsignal zurück.

Das Kontrollregister wird lediglich beim Systemstart benötigt. Damit kann man die Tastatur aus- (Bit 1=low) und anschalten (Bit 1=high) sowie die Clockleitung auf low ziehen (Bit 0=high).

5.3.5. Schaltungsaufbau

Die zentrale Funktionseinheit der Schaltung ist das 11-Bit-Schieberegister, das man auch parallel Laden und asynchron löschen kann. Mit eine steigende Flanke am Anschluß PC (Parallel Clock) lädt das Register parallel mit 11-Bit vom Anschluß PIN[10:0]. Mit einer steigenden Flanke am Anschluß SC (Serial Clock) werden die Bits um eins nach rechts geschoben. Ganz links wird dann das Bit vom Eingang SIN eingetaktet. Es muß sichergestellt sein, daß zu einem Zeitpunkt nur einer der beiden Anschlüsse PC und SC auf low liegt! Am Ausgang POUT[10:0] liegen alle 11 gespeicherten Bits an. Zusätzlich liegt am Ausgang SOUT das Bit 0 an.

Die Datenbits der Tastatur sind im Schieberegister invertiert abgelegt. Da im "Ruhezustand" und nach einem Reset alle Schieberegisterflipflops auf low liegen, kann damit das Startbit (low) der SDU von der Tastatur als Interrupt-Anforderung (IRQ) benutzt werden. Kommt nämlich dieses Bit nach dem elften Takt im Schieberegister im rechten Flip-Flop an, geht dieses von low nach high über und löst den Interrupt aus.

Durch das Auslesen des Datenregisters wird dieses wieder auf 0 gesetzt und für ein neues Datenbyte von der Tastatur vorbereitet. Dadurch wird auch automatisch das IRQ-Signal zurückgenommen.

Wichtig ist auch das Flip-Flop, das entscheidet, ob die Schnittstelle im Sende- oder Empfangsmodus ist. Das FF ist normalerweise auf low. Dadurch ist der serielle Dateneingang hochohmig und die Schnittstelle befindet sich im Empfangsmodus. Nur, wenn ein Byte durch den XProz in das Datenregister geschrieben wurde, geht die Schnittstelle in den Sendemodus. Solange das Datenbyte noch nicht vollständig übertragen ist, muß die Interrupt-Anforderungsleitung gesperrt bleiben, da die ausgegeben Bits sonst IRQs auslösen würden. Zum Erkennen des Transferendes hilft die Tastatur mit. Das letzte Bit, das Stopbit, ist high, aber die Tastatur zieht für dieses Bit die Datenleitung auf low. Wenn also der Adapter high ausgibt, auf der Datenleitung aber low liegt, dann sind alle 11 Bit gesendet und es wird wieder in den Empfangsmodus geschaltet.

Bleibt nur noch die Frage, wie die Tastatur erkennt, daß man ein Byte an sie Senden möchte und sie elf mal auf der Clock-Leitung takten muß. Dies ist ganz einfach, denn beim Einschreiben eines Bytes durch den Prozessor werden Start- (low), Stop- (high) und Paritybit (even) vom Adapter automatisch mit in das Schieberegister geschrieben. Der Adapter legt dann sofort das Startbit auf die Datenleitung, zieht sie also auf low. Das erkennt die Tastatur und taktet sich die restlichen Bits zu.

5.4. Timer

5.4.1. Allgemeines

Der Timer dient zur Erzeugung von Unterbrechungsanforderungen im Interruptlevel 5, die ca. 15.25 (genau: $(8*10^6)/(2^{19})$) mal in der Sekunde ausgelöst werden. Wenn dann die Unterbrechungsbehandlungsroutine jedesmal einen Zähler erhöht, hat man eine einfache und genaue Zeitbasis, wie sie zum Lösen verschiedener Aufgaben im Betriebssystem nützlich sind. Ein Beispiel wäre eine Time-Out-Fehlermeldung nach einer definierten Zeit, wenn der Drucker keine Daten annimmt.

5.4.2. Programmierung

Die einzige Programmiermöglichkeit ist die, den Timer ein- und auszuschalten. Hierfür gibt es ein Kontrollregister an der IO-Adresse $7FF3. Schreibt man dort eine 0 hinein, wird der Timer abgeschaltet, eine 1 schaltet ihn ein. Zur Interruptbestätigung in der Interruptbehandlungsroutine muß ebenfalls eine 1 an diese Adresse geschrieben werden.

5.4.3. Schaltungsaufbau

Die Schaltung besteht im Wesentlichen aus einem einfachen 19-Bit-Ripple-Zähler. Mit ihm wird der 8 Mhz-Takt, der an einem Steckkartenplatz zur Verfügung steht, auf die nötige Timerfrequenz von 15.25 Hz herunterteilt. Desweiteren gibt es noch ein Flip-Flop, welches immer beim Null-Durchgang des Zählers gesetzt wird und so einen Interrupt auslöst. Dieses Flip-Flop kann man nur durch einen Schreibzugriff auf das Timer-Kontrollregister löschen. Um den Timer ein- und auszuschalten, gibt es ein weiteres Flip-Flop, über das die Interruptleitung fest auf high (inaktiv) gelegt werden kann.

6. Centronics- und RS-232-Karte

6.1. Aufbau

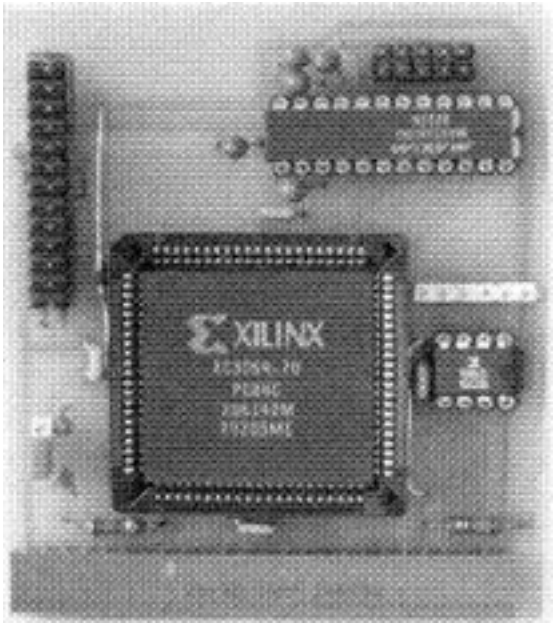


Bild 13: Centronics- und RS232-Karte

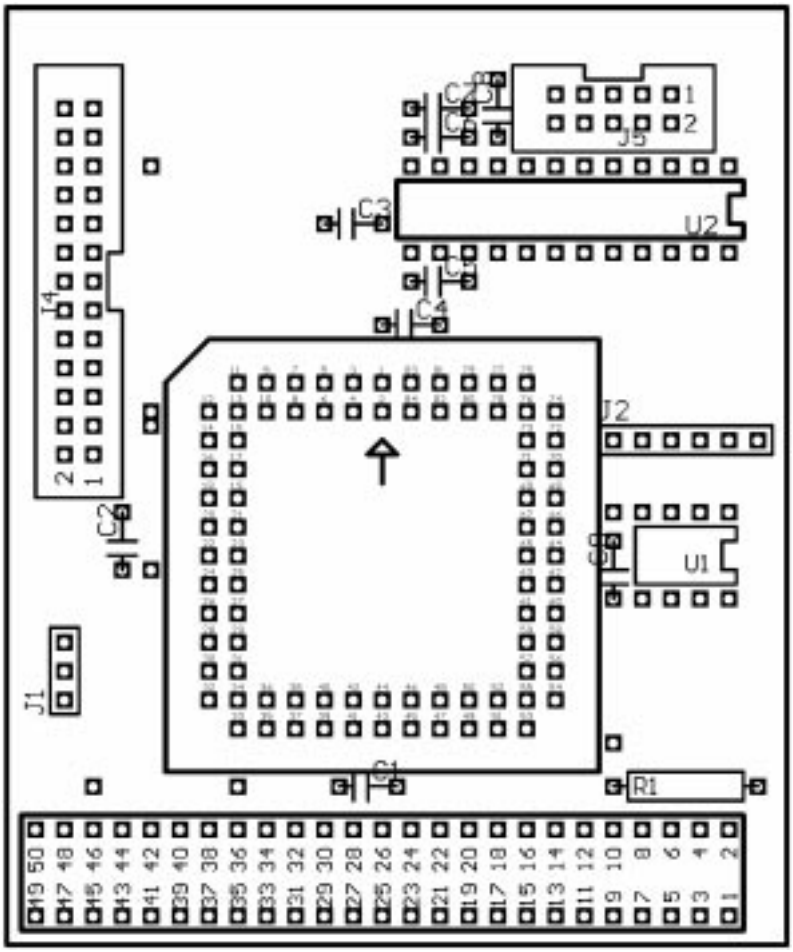


Bild 14: Bestückungsplan der Centronics- und RS232-Karte

Nummer	Name	Typ
U1	Konfigurationsprom	XC1765
U2	TTL-RS232-Pegelwandler	MAX238
C1,C2,C4,C9	Entstörkondensator	100 nF
C3,C5,C6,C7,C8	Tantal-Kondensator	1 µF
R1	Widerstand	2 kOhm
J1	Master/Slave	Jumper
J2	Download	6 pol. Stiftleiste, einreihig
J3	XProz-Bus	50 pol. Buchsenleiste, doppelreihig
J4	Parallelportanschluß	26 pol. Stiftleiste, doppelreihig
J5	Seriellportanschluß	10 pol. Stiftleiste, doppelreihig

Tabelle 52: Bestückungsliste der Centronics- und RS-232-Karte

6.2. Centronics-Schnittstelle

6.2.1. Allgemeines

Centronics ist eine eine 8-Bit parallele, unidirektionale Schnittstelle mit einfachem Handshaking, die vorwiegend zum Anschluß eines Druckers dient. Über spezielle Leitungen kann der Drucker seinen Status an den Rechner melden.

6.2.2. Hardwarespezifikation

Die Schnittstelle arbeit mit TTL-Pegeln, also 0V für low und 5V für high. Der Anschluß erfolgt rechnerseitig über einen 25-poligen SUB-D-Stecker. Auf der Druckerseite befindet sich ein 36-poliger Centronics-Stecker. Die Pinbelegung auf der Rechnerseite ist wie folgt:

Pinnummer	Richtung	Bezeichnung
1	Rechner => Drucker	-STROBE
2 bis 9	Rechner => Drucker	DATA 0 bis DATA 7
10	Drucker => Rechner	-ACK (Acknowledge)
11	Drucker => Rechner	BUSY
12	Drucker => Rechner	PE (Paper empty)
13	Drucker => Rechner	SELECTED
14	Rechner => Drucker	-AUTO FEED
15	Drucker => Rechner	-ERROR
16	Rechner => Drucker	-INT (Initialize)
17	nicht benutzt	
18 bis 26	GROUND	

Tabelle 53: Pinbelegung der Centronics-Schnittstelle

Mit einem Low-Impuls auf der Leitung -STROBE übernimmt der Drucker ein Byte von den Datenleitungen 0-7. Ein Low-Impuls an der Leitung -INT setzt den Drucker zurück. Die Leitung -AUTO FEED wird bei heutigen Druckern nicht mehr verwendet, sie ist nur der Vollständigkeit halber implementiert. Auf der Leitung -ACK bestätigt der Drucker mit einem Low-Impuls den Empfang eines Datenbytes.

Auf der Leitung BUSY zeigt der Drucker an, ob er Daten entgegennehmen kann oder nicht. Ist diese Leitung high, dann nimmt der Drucker keine Daten von der Centronics-Schittstelle an. Ist die Leitung PE auf high, dann kein Papier mehr im Drucker. Die Leitung SELECTED zeigt an, ob die Select- bzw. On-Line-Taste am Drucker betätigt ist. Und schließlich über die Leitung -ERROR zeigt der Drucker an, daß irgend ein Fehler aufgetreten ist, wie zum Beispiel ein Papierstau.

6.2.3. Programmierung

Der Druckerport belegt zwei Adressen im I/O-Adressbereich:

Adresse	lesen	schreiben
$7FF8	Druckerstatus	Datenregister
$7FF9	Druckerstatus	Kontrollregister

Tabelle 54: Register der Centronics-Schnittstelle

Bit1	Bit0	Funktion
0	0	keine auf -STROBE
0	1	-STROBE löschen
1	0	-STROBE setzen
1	1	-STROBE invertieren
Bit3	**Bit2**	
0	0	keine auf -INT
0	1	-INT löschen
1	0	-INT setzen
1	1	-INT invertieren
Bit5	**Bit4**	
0	0	keine auf -AUTO FEED
0	1	-AUTO FEED löschen
1	0	-AUTO FEED setzen
1	1	-AUTO FEED invertieren

Tabelle 55: Statusregister der Centronics-Schnittstelle

Bit#	Status
0	BUSY
1	-ACK (Acknowledge)
2	-ERROR
3	PE (Paper empty)
4	SELECTED

Tabelle 56: Kontrollregister der Centronics-Schnittstelle

Um ein Byte an den Drucker zu senden, muß man zuerst sicherstellen, daß der Drucker empfangsbereit ist, BUSY im Statusregister muß 0 sein. Dann schreibt man das Byte ins Datenregiste setzt das -STROBE-Signal, indem man eine 2 in das Kontrollregister schreibt. Frühestens nach 10 µs muß man dann das -STROBE-Signal wieder löschen, also eine 1 ins Kontrollregister schreiben.

6.2.4. Schaltungsaufbau

Die Schaltung besteht praktisch nur aus elf Flip-Flops, die die acht Datenbits und die drei Steuerbits darstellen. Um die Steuerbits einzeln ein- und ausschalten zu können, ohne daß sie auf verschiedenen I/O-Adressen liegen oder das Kontrollregister auslesbar ist, wurden hierfür keine D- sondern J-K-Flip-Flops verwendet. Dadurch hängt zwar jedes Steuerbit an zwei Datenleitungen, aber sie lassen sich unabhängig voneinander setzen, löschen und sogar invertieren.

6.3. RS-232-Schnittstelle

6.3.1. Allgemeines

Die RS-232-Schnittstelle (die entsprechende deutsche Norm heißt V.24) ist eine asynchrone, bidirektionale, serielle Schnittstelle mit vielen Status- und Handshakeleitungen. Sie dient vor allem zum Verbinden eines Datenendgeräts (Data Terminal, meist ein Rechner) und einer Datenübertragungseinheit (zum Beispiel ein Modem). Es lassen sich aber auch zwei Rechner über ein sogenanntes Nullmodemkabel direkt verbinden. Früher wurden auf diese Weise Terminals, zum Beispiel das VT-52, an einen Großrechner angeschlossen.

6.3.2. Hardwarespezifikation

Die Schnittstelle arbeitet nicht mit TTL-Pegeln. Vielmehr wird high als eine Spannung zwischen -3V und -15V bezüglich GROUND dargestellt und low als eine Spannung zwischen 3V und 15V. Durch diese ungewöhnlich großen Spannungsdifferenzen und den geringen Übertragungsraten der klassischen Terminals konnten die Verbindungskabel sehr lang werden.

6.3.3. Programmierung

Die Schnittstelle belegt im I/O-Adressraum vier Adressen:

I/O-Adresse	Zugriff	Register
$7FFC	Lesen/Schreiben	Statusregister SREG
$7FFD	Lesen	Empfangsregister RREG
$7FFE	Schreiben	Kontrollregister CREG
$7FFF	Schreiben	Senderegister WREG

Tabelle 57: Register der RS-232-Schnittstelle

Bit#	Funktion
1,0	Übertragungsrate: 00: 300 Baud 01: 4800 Baud 10: 9600 Baud 11: 19200 Baud
2	0: Ein Stoppbit 1: Zwei Stoppbits
3	0: Interrupt sperren 1: Interrupt freigeben
6	DTR
7	RTS

Tabelle 58: Kontrollregister CREG der RS-232-Schnittstelle

Bit#	Funktion
0	0: kein Paritätsfehler 1: Paritätsfehler
1	0: kein Fehler 1: Übertragungsfehler (zuwenig Stoppbits erkannt)
2	0: kein Fehler 1: Overrun (Empfangsbyte wurde nicht rechtzeitig abgeholt)
3	0: noch kein Byte angekommen 1: Byte angekommen
4	0: Sendepuffer leer 1: Sendepuffer noch voll
5	DCD
6	DSR
7	CTS

Tabelle 59: Statusregister SREG der RS-232-Schnittstelle

Um ein Byte zu Senden, muß man es einfach ins Senderegister schreiben. Vorher muß jedoch durch Abfrage des Statusregisters sichergestellt sein, daß das Senderegister bereit und nicht mehr mit dem Senden eines vorigen Bytes beschäftigt ist.

Wird ein Byte empfangen, dann löst die Schnittstelle eine Unterbrechung aus. In der Unterbrechungsbehandlungsroutine muß das Byte nur noch im Empfangsregister abgeholt werden. Außerdem sollte die Routine das Statusregister auslesen, um einen eventuellen Übertragungsfehler festzustellen. Über einen beliebigen Schreibzugriff auf das Statusregister wird dieses dann wieder gelöscht.

6.3.4. Schaltungsaufbau

ser ist in zwei Steuerwerke *stw_rec* zum Lesen bzw. *stw_tran* zum Schreiben von Daten, den zugehörigen Takt-/Signalgenerator-Moduln *bit_scan* bzw. *bit_tran* und dem gemeinsamen Registermodul *reg_mod* gegliedert.

Registermodul *reg_mod*:Wie schon oben angedeutet läuft über diesen Modul die gesamte Kommunikation sowohl über den acht Bit breiten Datenbus zum Prozessor als auch über die bitseriellen Leitungen *TD* bzw. *RD* und die Handshakeleitungen *RTS*, *DTR*, *CTS, DSR* und *DCD* zu einem angeschlossenen DV-Gerät. Zusätzlich zu den vier Registern (von oben nach unten gezeichnet) *CREG, SREG, WREG und RREG* sind noch zwei neun Bit breite Schieberegister *WSHR* (rechts unten mit "shifter" bezeichnet) und *RSHR* (links unten als 8-Bit breites und 1-Bit breites Register aufgebaut) implementiert, welche mit *WREG* bzw. mit *RREG* zusammenarbeiten, das höchstwertige Bit stellt jeweils das Paritybit dar. Aus Gründen der Störsicherheit ist *WSHR* nachträglich als Master-Slave-Register verändert worden.

Im Kontrollregister *CREG* sind von der höchstwertigsten Stelle links bis zur niederwertigsten Stelle rechts die Bits wie folgt definiert:

RTSF, *DTRF*, frei, frei, *IEF*, *STOPBITS*, *BAUDRATE* (zwei Bits)

Im Statusregister *SREG* sind die oberen drei Bits als Leitungen weitergeführt, nämlich von links nach rechts *CTS, DSR* und *DCD*. Dann folgen die gespeicherten Statuswerte

SENDEPUFFER, INTERRUPT, OVERRUN, LESEDATEN und *LESEPARITY*.

Lese-Modul *bit_scan*: Dieses Modul leistet die exakte Dekodierung der Bits aus dem über den Pin *RD* eintreffenden Datenstrom. Dazu muß zunächst die richtige Baudrate eingestellt sein, damit die Bits auch zeitlich richtig abgetastet und gespeichert werden können. Die Abtastung pro Bit geschieht dreimal, wobei der Wert als fehlerfrei erkannt und übernommen wird, der zweimal denselben Abtastwert liefert. Dieses Modul wird nach jedem übertragenen 11- bzw. 12-Bit-Datum (abhängig von der Anzahl der Stopbits) von dem Lese-Steuerwerk *stw_rec* initialisiert, damit die Abtastung immer synchron zum Datenstrom arbeitet.

Der an Pin 13 des Chips anliegende 8 MHz-Takt wird um den Faktor 2 erniedrigt einer Vorteilerkette zugeführt, welche entsprechend der Baudraten die Frequenz um

den Faktor 2 für 19200 Baud,
den Faktor 4 für 9600 Baud,
den Faktor 32 für 1200 Baud und
den Faktor 128 für 300 Baud

herunterteilt. Die so gewonnene Frequenz wird jetzt nochmals durch die Faktoren 13 und 8 geteilt, wobei der letzte Teiler die Abtastsignale erzeugt. Mit den ansteigenden Flanken der Abtastsignale *AB2, AB3* und *AB4* wird das ankommende Bit abgetastet und mit der abfallenden Flanke von *AB4* wird der resultierende Wert übernommen, der als Signal *bit_out* zum einen das eigentliche Datenbit und zum anderen als Bedingung in die Ablaufsteuerung *stw_rec* eingreift. Mittels *AB6* wird der negierte Wert *bit_out* ebenfalls als *STARTBIT* gespeichert und zur Steuerung in *stw_rec* verwendet. Die Zählung der acht Datenbits geschieht mittels des Zählers *c8dff*, welcher das Signal *BITCOUNT_0* generiert.

Der ankommende Datenstrom wird fortlaufend, d.h. solange das Steuerwerk *stw_rec* den Zustand *Z0* inne hat, mit einem 4 MHz-Takt durch den Zähler *c2dff* abgetastet. Liegt dieser Datenstrom für wenigstens zwei Taktzyklen lang auf 0-Potential, so wird das Signal *SCAN_START* erzeugt, welches den Beginn einer Datenübertragung signalisiert.

Lese-Steuerwerk *stw_rec*:Im Schaltbild sind die Zustände durch die Flipflops von oben nach unten dargestellt und mit *Z0, Z1, ..., Z14* bezeichnet.

Wie im Schaltungsbild gezeigt, werden die einzelnen Zustände durch die Bedingungen Bi verlassen und bleiben durch die Bedingungen Ei erhalten. Nachfolgend wird eine kurze Beschreibung der Zustände und Bedingungssignale gegeben:

In den Zuständen Z2..Z5 werden die Datenbits übernommen, im Zustand Z6 bzw. Z7 wird das Päritybit übernommen, und die restlichen Zustände sind zur Überprüfung der Stop-/des Startbits bzw. zur Fehlermeldung notwendig.

Z0: Grundzustand, Initialisierung verschiedener FFs.
Z1: BITCOUNT <-- 0, SREG(2:0) <-- 0
Z2:
Z3: SR(7) <-- BIT_OUT, SR <-- SR.SHR., BITCOUNT <-- BITCOUNT.INC
Z4:
Z5:
Z6:
Z7: PÄRITYFLAG <-- BIT_OUT, falls PF = 0
Z8: SREG(2) <-- 1 (Overrun), falls SREG(3) = 1
Z9: RREG <-- SR
Z10: 1.Stopbit
Z11:
Z12: 2.Stopbit
Z13:
Z14: SREG(1) <-- 1 (Fehlerhafte Datenübertragung, fehlerhafte Anzahl der Stopbits)

B1:	*SCAN_START*
B2:	*STARTBIT*
B3, B7, B9, B15:	*AB5*
B4, B8, B14:	*AB6*
B5:	*BITCOUNT_0-*

B6:	*BITCOUNT_0*
B10, B17:	*BIT_OUT * AB6*
B11, B16:	*BIT_OUT- * AB6*
B12:	*CREG(2)*
B13:	*CREG(2)-*
E1:	*SCAN_START-*
E2:	*STARTBIT-*
E3, E5, E7, E10:	*AB5-*
E4, E6, E8, E9, E11:	*AB6-*

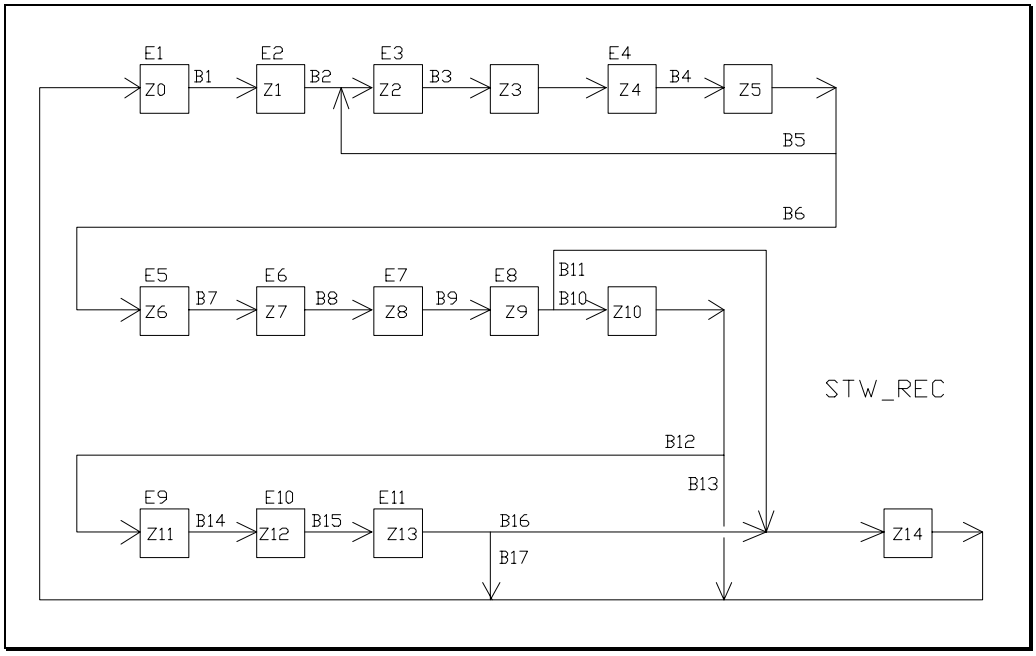


Bild 15: Zustandübergangsdiagramm des *stw_rec*

Schreib-Modul *bit_transmit*: Dieses Modul ist ähnlich aufgebaut wie das Modul *bit_scan*, es enthält jedoch nur die erste Teilerkette zur exakten Einstellung der Baudrate, den "Teiler durch 13" und den "Teiler durch 8", welcher die Steuersignale *TAST7* und *TAST8* erzeugt. Somit ist die zeitlich richtige Übertragung von Daten sichergestellt.

Im Gegensatz zum Lese-Modul *bit_scan* braucht *bit_transmit* während des Betriebes nicht initialisiert zu werden; denn eine Synchronisation ist in diesem Fall bei einer Datenübertragung nicht notwendig, da dieser Chip als Master arbeitet.

Schreib-Steuerwerk *stw_tran*: Im Schaltbild sind die Zustände durch die Flipflops von oben nach unten dargestellt und mit X*0, X1, ..., X10* bezeichnet, das oberste FF zählt nicht zum Steuerwerk, es generiert lediglich das Signal *CLOCK_SRW*.

Im Schaltungsbbild links unten ist der 9-Bit-Zähler *c9bcdr* gezeichnet, der das Signal *TRANSCOUNT* erzeugt, das zur Zählung der acht Datenbits und des Paritybits dient.

Wie im Schaltungsbild ebenfalls zu sehen ist, werden die einzelnen Zustände durch die Bedingungen Bi verlassen und bleiben durch die Bedingungen Ei erhalten. Nachfolgend wird eine kurze Beschreibung der Zustände und Bedingungssignale gegeben:

Das Start- und die Stopbits sind nicht in FFs gespeichert, sondern sie werden durch das "1"- bzw. "0"-Setzen der *TD*-Leitung erzeugt, dazu werden die Signale *SRW_EN* und *NOT_ACTIV* generiert. Das Laden des Sendepuffers *WREG* zur Datenübertragung setzt ebenfalls das Flag *SREG(4)*, wodurch eine Zustansänderung X0 --> X1 geschieht. In den Zuständen *X1* und *X2* muß sichergestellt werden, daß wenigstens die Zeit eines Stopbits eingehalten wird, bevor in den Zuständen *X3* und *X4* das Startbit übertragen wird. In den Zuständen *X5...X7* werden die acht Datenbits und das Paritybit übertragen und in den Zuständen *X8...X10* die eingestellte Anzahl der Stopbits.

X0: Grundzustand, *NOT_ACTIV*, Initialisierung verschiedener FFs.
X1: *LOAD, NOT_ACTIV, SP_LEER*
X2: TRANSCOUNT <-- 0, *NOT_ACTIV*
X3: *NOT_ACTIV_*
X4: *NOT_ACTIV-, SR_EN_*
X5: TRANSCOUNT <-- TRANSCOUNT.INC., *NOT_ACTIV-, SR_EN*
X6: *NOT_ACTIV-, SR_EN*
X7: SRW <-- SRW.SHR. falls X7 * TAST8, *NOT_ACTIV-, SR_EN*
X8: *NOT_ACTIV*
X9: *NOT_ACTIV*
X10: *NOT_ACTIV*

B1: *SREG(4)* - Sendepuffer voll
B2, B4, B10: *TAST8*
B3, B5, B11: *TAST7*
B6: *TAST8 * TRANSCOUNT-*
B7: *TAST8 * TRANSCOUNT*
B8: *TAST7 * STOPF*
B9: *TAST7 * STOPF-*

E1: *SREG(4)-*
E2, E4, E6, E8: *TAST8-*
E3, E5, E7, E9: *TAST7-*

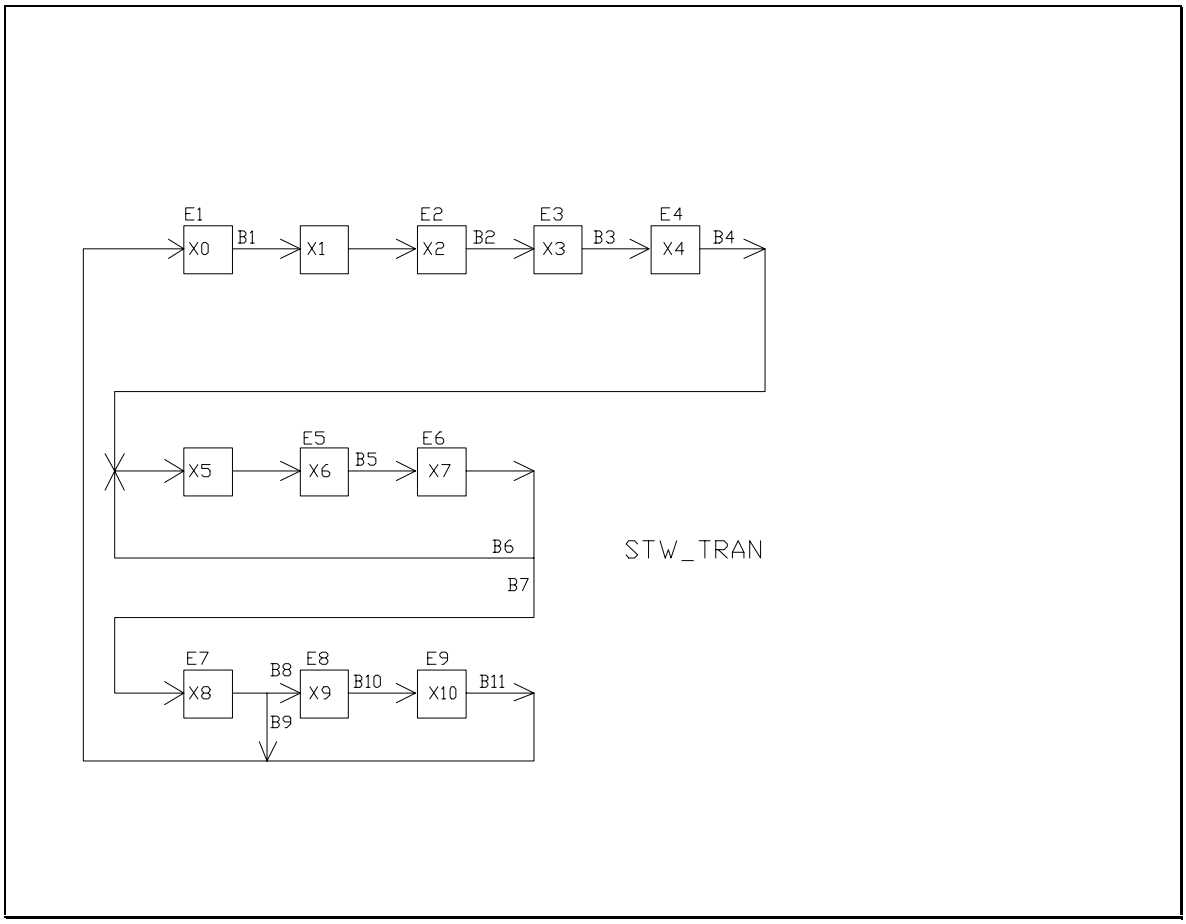


Bild 16: Zustandübergangsdiagramm des *stw_tran*

7. Betriebssystemreferenz XMOS 1.0

7.1. Vorwort

Das hier vorgestellte Betriebssystem weist einige nennenswerte Merkmale auf. Es handelt sich hier um ein Multitasking-Betriebssystem, das sowohl kooperativ als auch preemptiv arbeitet. Kooperativ wird jeweils beim Warten auf Systemresourcen gearbeitet. Für das preemptive Scheduling wird nach der Round-Robin-Methode jede 1/16 Sekunde eine laufende Task temporär suspendiert und die nächste Task in der Taskliste aktiviert. Die Anzahl der lauffähigen Tasks wird momentan nur durch den geringen Arbeitsspeicher begrenzt.

In diesem Betriebssystem werden alle Geräte, Dateien und Verzeichnisse gleich behandelt, als Dateien. Man kann also die Serielle Schnittstelle, eine ausführbare Datei oder ein Verzeichnis mit dem gleichen Systembefehl OPEN öffnen und mit den gleichen Operationen READ und WRITE bearbeiten. Dem System ist immer das Null-Device bekannt, daß alle Ausgaben aufnimmt und READ-Aufrufe mit dem Fehler End-Of-File beantwortet.

Das System arbeitet mit zwei virtuellen Konsolen, die durch die Taste [NUM] umgeschaltet werden können.

In dieser Version des Betriebssystems sind entsprechend der momentan angeschlossenen Hardware folgende Geräte dem System bekannt:

a:	SCSI-Device 0, Diskettenlaufwerk
b:	SCSI-Device 1, Festplattenlaufwerk
rom:	ROM-Drive, Blockdevice im ROM Speicher
ser:	Serielle Schnittstelle
par:	Parallelport
con0:	Konsole 0
con1:	Konsole 1
err:	Error-Konsole, die jeweils sichtbare Konsole.

Der SCSI-Treiber stellt einen dynamisch wachsenden Cache zur Verfügung.

Strings müssen alle mit einer Null terminieren, damit sie korrekt bearbeitet werden können. Für Systembefehle müssen Strings in ungepackter Form, daß heißt ein Zeichen pro Wort vorliegen.

Das Trennzeichen zwischen Verzeichnissen und Dateien ist hier ein /.

Es sind zwei einfache Formen von Wildcards zulässig. Ein **?** bedeutet, daß das Zeichen an dessen Stelle das **?** steht beliebig sein kann. Ein ***** bedeutet, daß alle nachfolgenden Zeichen beliebig sein können, daß heißt Hallo***** wird genauso bearbeitet wie Hallo*****World. Zulässig in Dateinamen sind Zeichen mit ASCII-Codes zwischen $21 und $7f.

Dateinamen dürfen eine Länge von maximal 16 Zeichen haben. Groß- und Kleinbuchstaben werden unterschieden. Treibernamen sind auf vier Zeichen begrenzt, Tasknamen auf acht Zeichen. Wählt man eine Datei eines anderen Gerätes muß der Pfadname folgende Gestalt haben:

```
/Treibername/Verzeichnisname/.../Dateiname
```

Verzeichnisse können beliebig tief verschaltet werden. Dateien sind auf eine Länge von 32MByte begrenzt. Adressierbar sind Massenspeicher mit bis zu 2^{32} Sektoren a 512 Bytes = 2 TeraBytes.

Der Betriebssystemkernel, bestehend aus Treiberverwaltung, Taskverwaltung und Speicherverwaltung (ohne Dateiverwaltung, Shell und Treibern), hat eine Größe von 3622 Bytes.

7.1.1. Namensgebung

In Anlehnung an die Baugruppennamen XProz, XSer u.s.w. und in Erinnerung daran, daß die Grundlage dieses Systems XILINX-Chips sind, wurde der Name XMOS 1.0 als Betriebssytemname gewählt. Er steht für

XILINX MULTITASKING OPERATING SYSTEM Version 1.0

In der Hoffnung, daß es keine urheberrechtlichen Konflikte gibt wurde der Name XILINX mit in den Betriebssystemnamen aufgenommen.

7.1.2. Abkürzungen

Im Weiteren werden folgende Abkürzungen benutzt:

sp:	Stackpointer
pc:	Aktueller Programcounter
sr:	Statusregister
d0-d7:	Register d0-d7
N-Flag:	Negative-Flag
Z-Flag:	Zero-Flag
C-Flag:	Carry-Flag
V-Flag:	Overflow-Flag
CWD:	Current Working Directory, aktuelles Verzeichnis
BAM:	Block Available Map
TDB:	Task-Descriptor-Block
FDB:	File-Descriptor-Block
MDB:	Media-Descriptor-Block

7.2. Betriebssystemkonzept

7.2.1. Aufbau

Das Betriebssystem besteht aus vier Einheiten:

1. Speicherverwaltung
2. Taskverwaltung
3. Dateiverwaltung
4. Treiberverwaltung

Eine Trennung von Basic Input Output System und Betriebssystem liegt hier nicht mehr vor. Die Aufgaben des BIOS werden von den Gerätetreibern übernommen. Software kann völlig geräteunabhängig entwickelt werden, da über die einheitlichen Dateioperation jedes Gerät angesprochen werden kann.

Das Betriebssystem ist modular aufgebaut. Ohne besondere Schwierigkeiten können Teile des Betriebssystems durch andere ersetzt werden oder andere Geräte eingebunden werden. Es gibt nur eine Verbindung zwischen SCSI-Treiber und Speicherverwaltung, die aber dokumentiert ist und leicht entfernt werden kann (siehe CACHE). Auch interne Operationen werden über die regulären Systembefehle bearbeitet.

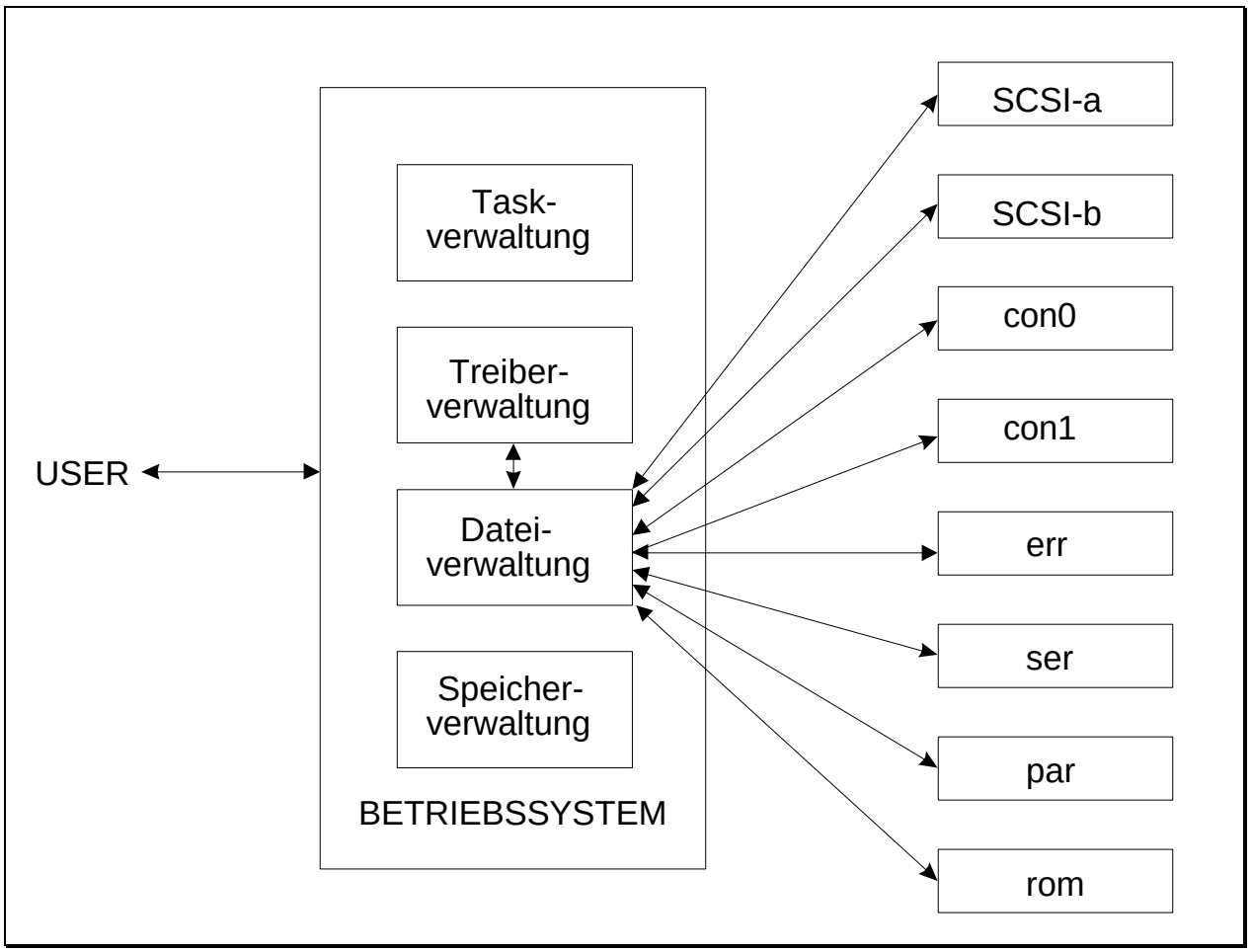


Bild 17: Systemsicht des Anwenders

7.2.2. Anpassung an andere Umgebungen

Das Betriebssystem ist an keine spezielle Adresse oder Hardware gebunden. Es kann auch in anderen Bereich des Speicher installiert werden. Sind die entsprechenden Treiber vorhanden kann praktisch mit jedem Gerät gearbeitet werden.

7.3. Multitasking

Das Betriebssystem ist multitaskingfähig. Eine Einschränkung der Anzahl der Tasks ergibt sich nur durch den zur Verfügung stehenden Arbeitspeicher des Systems. Alle Betriebssystemroutinen sind reentrant erstellt, sie arbeiten als Hoare'sche Monitore. Diesen Routinen steht ein Satz von acht Arbeitsregistern d0-d7 zur Verfügung, die bei einem Taskwechsel neben dem Stackpointer, den Prozessor-Flags und dem Programcounter gerettet und restauriert werden. Bei diesen Registern handelt es sich nicht um Hardware-Register, sondern um normale Speicherzellen des Arbeitsspeichers, die aufgrund ihrer besonderen Verwendung nur Register genannt werden. Das Betriebssystem arbeitet sowohl kooperativ als auch preemptiv. Die Tasks sind in einer zyklischen Liste angeordnet. Es werden nur zwei Taskzustände unterschieden, nämlich laufend und laufbereit.

7.3.1. Semaphore

Der Zugriff auf Systemresourcen wird über Semaphore geregelt. Als Semaphor wird ein ganzes Wort benutzt. Dieses hat den Wert Null, falls die Systemresource verfügbar ist. Andernfalls enthält das Wort die Tasknummer der belegenden Task. So läßt sich jederzeit ermitteln, welche Task welche Systemresource nutzt. In diesem Betriebssystem werden die Resourcen Arbeitsspeicher, File-Descriptor-Block-Liste, Konsole, SCSI und Datenträgerspeicher mit Hilfe von Semaphoren verwaltet. Während der Abfrage und der Veränderung eines Semaphores wird ein Taskwechsel durch eine temporäre Unterbrechungssperre unterbunden.

Beispiel einer Semaphorbehandlung:

```
driver: irqoff                              ;Unterbrechung unterbinden
        cmp     scsi_in_use,-               ;SCSI belegt?
        move eq akt_task,scsi_in_use        ;Nein, selber belegen
        irqon
        jeq     .semok
        move    akt_task,want_to_use_scsi   ;Fairness-Flag setzen
        jsr     SWITCHTASK                  ;Kooperatives Handeln!
        jmp     driver

.semok: ...
```

7.3.2. Kooperatives Multitasking

Kooperativ wird jeweils beim Warten auf Systemresourcen verfahren. Ist ein Semaphor belegt, wird ein Taskwechsel ausgelöst, da die aktuelle Task Rechenzeit nicht sinnvoll nutzen kann.

7.3.3. Preemptives Multitasking

Für das preemptive Scheduling wird nach der Round-Robin-Methode jede 1/16 Sekunde eine laufende Task temporär suspendiert und die nächste Task in der Taskliste aktiviert. Jedesmal bei der Aktivierung des Timers wird ein Taskwechsel ausgelöst, sofern nicht gerade einer durchgeführt wird. Das kooperative und preemptive Multitasking wird jedoch nicht koordiniert. Es kann also vorkommen das nachdem kooperativ verfahren wurde ein Taskwechsel aufgrund des Zeitplanes ausgelöst wird. Eine Beachtung dieses Falles wurde nicht implementiert um die Timer- und Taskwechselroutine möglichst kurz und einfach zu halten. Der Verwaltungsaufwand für einen Taskwechsel sollte viel geringer sein als das Arbeitszeit-Quantum einer Task. Die Timerroutine umfaßt neun Befehle und die Taskwechselroutine 52 Befehle, die linear nacheinander abgearbeitet werden. In einer Sekunde können bei jetziger Systemkonstellation ca. 400.000 Befehl abgearbeitet werden, das heißt in eine Zeitscheibe werden ca. 25.000 Befehle bearbeitet. Diese 9+52=61 Befehle Verwaltungsaufwand verbrauchen ca. 0,248% der Rechenzeit einer Zeitscheibe.

7.3.4. Fairness

Der Zugriff auf SCSI-Geräte ist mit einer einfachen Fairnessregel verbunden um zu verhindern, daß eine einzige Task durch ungünstige Taskwechselzeitpunkte immer diese Resource belegen kann, also andere andere Tasks *aushungert*. Ist die Resource SCSI nicht verfügbar, wird ein Flag gesetzt, welches anzeigt, daß eine andere Task diese Resource jetzt nutzen will. Nach Freigabe der Resource SCSI wird überprüft, ob dieses Flag gesetzt ist. Ist dies der Fall, wird ein Taskwechsel direkt ausgelöst. Da die anderen Resourcen in der Regel nur sehr kurz belegt sind, wurden keine weiteren Fairnessregeln implementiert. Zeigt die Anwendung des Systems, daß diese trotzdem von Nöten sind, lassen sich diese leicht nachträglich ergänzen. Die entsprechenden Stellen im Betriebssystemquelltext sind markiert. Dies könnte bei häufigen dynamischen Speicheroperationen der Fall sein, wenn kleine Speicherblöcke häufig allokiert und freigegeben werden.

7.4. Systembesonderheiten

7.4.1. SCSI-Cache für Festplatte und Floppy

Das Betriebssystem arbeitet mit einem durch den SCSI-Treiber zur Verfügung gestellt Software-Cache. Die Größe des Caches ändert sich dynamisch. Er verfügt über eine Mindestgröße 6*260 Worten Speicherplatz für sechs Sektoren à 256 Worte plus 4 Worte Verwaltungsinformation. Wird der Arbeitsspeicher nicht anderweitig benötigt, dann wird er als Cachespeicher benutzt. Wird ein Speicherblock angefordert, dann wird die Anzahl an Cache-Puffern reduziert und als Arbeitsspeicher verwendet. An dieser Stelle arbeiten Speicherverwaltung und SCSI-Treiber zusammen. Wenn Speicher frei wird, werden neue Cache-Puffer von der Speicherverwaltung zur Verfügung gestellt. Wird jedoch noch Speicher benötigt wird dem SCSI-Treiber ein bestimmter Job übergeben, der dann entsprechend viele Cache-Puffer freigibt. Der Cache liegt immer zusammenhängend am Ende des Arbeitsspeichers. Es verschiebt sich nur die Ende-Marke des freien Speichers, die gleichzeitig die Anfangs-Marke des Cache-Bereiches ist.

Es wird hier ein einfache Cache-Strategie verfolgt, die jeweils den Cache-Puffer verwirft, der am längsten nicht benutzt wurde. Diese Cache-Strategie ist bekannt unter der Bezeichnung LRU (least recently used). Es gibt keinen Zusammenhang zwischen der Adresse eines gepufferten Blocks und der Position im Cache-Speicher. Daher ist der hier implementierte Cache voll-assoziativ und man muß im Cache-Speicher nach einem angeforderten Block linear suchen. Bei Schreibzugriffen auf einen Massenspeicher werden Daten über den Cache verzögert geschrieben (Write-Back-Cache). Die Variable **accesses** wird immer um eins erhöht, wenn auf irgendeinen Cache-Puffer zugegriffen wird. Dieser Wert wird dann als Zurgiffszähler in den Verwaltungsbereich des Puffers eingetragen. Dies geschieht auch, wenn ein Sektor neu in einen Cache-Puffer geladen wird. Ist Bit_{15} des Gesamtzugriffszählers **accesses** gesetzt, werden die Zugriffszähler der einzelnen Cache-Puffer zurückgesetzt, indem 255 zu den einzelnen Zählern addiert und dann durch 256 geteilt wird. Wurde also auf einen Puffer schon mal zugegriffen, dann hat er jetzt noch den Wert 1. Der Gesamtzugriffszähler **accesses** erhält den Wert 257. Dies ist größer als der derzeit größtmögliche Wert in einem benutzten Puffer. Der Puffer mit dem geringsten Zurgiffszähler wird bei einem Cache-Miss mit dem neuen Sektor überschrieben. Ein Puffer wird durch ein gesetztes Bit im ersten Verwaltungswort als gültig gekennzeichnet.

Unpassend ist diese Strategie bei einem Kopiervorgang. In diesem Fall wird ein Sektor genau einmal gelesen und einmal geschrieben, danach wird nicht mehr auf den Sektor zugegriffen. Dann werden der Reihe nach alle anderen Cache-Puffer überschrieben und nicht dieser eine für den nächsten Sektor benutzt, da er dann den höchsten Zugriffszähler hat.

7.4.2. ROM-Disk

Der verbleibende Speicherbereich des ROM-Speicher, der nicht vom Betriebssystem genutzt wird, wird als Datenträger eines Blockdevices, der ROM-Disk, verwendet. Die Struktur, in der die Daten abgelegt sind, entspricht der vom Betriebssystem benutzten Dateistruktur. Es existiert ein Bootsektor und ein BAM-Sektor auf dieser ROM-Disk mit einer Sektorgröße von 256 Worten. So können bis auf Schreibzugriffe alle Dateioperationen auf dieser ROM-Disk durchgeführt werden. Die ROM-Disk hat momentan eine Kapazität von 14336 Worten, die für die Dienstprogramme genutzt werden. Da der Zugriff auf die ROM-Disk weitaus schneller ist als der Zugriff auf die anderen Geräte werden die Dienstprogramme mit nur geringer Zeitverzögerung ausgeführt. Der ROM-Disk-Treiber ist ein vereinfachter SCSI-Treiber.

7.5. Speicherverwaltung

7.5.1. Speicheraufbau

Der Arbeitsspeicher wird über eine verkette Liste der Speicherblöcke verwaltet. Speicherblöcke können in einer beliebigen Größe angefordert werden. Sie werden nach der First-Fit-Methode ermittelt, also der erste, ausreichend große, freie Speicherblock in der Liste wird ausgewält. Zu einem Speicherblock gehören zwei Worte Verwaltungsinformation. Das erste Verwaltungswort gibt die Länge dieses Speicherblocks inklusive der beiden Verwaltungsworte an, das zweite Wort gibt an wem der Speicherblock gehört. Ist dieses zweite Wort Null ist der Block frei, hat das Wort den Wert $ffff, gehört er dem Betriebssystem, andernfalls enthält dieses Wort die Tasknummer der Task, die diesen Speicherblock allokiert hat. Die Variable **bfs** zeigt auf das erste Verwaltungswort, die Länge des ersten Speicherblocks. Durch Addition der Länge zur Adresse dieses ersten Verwaltungswortes ergibt sich die Adresse des ersten Verwaltungswortes des nächsten Speicherblockes. Wird ein Speicherblock wieder freigegeben, dann wird nur das zweite Verwaltungswort auf Null gesetzt und dieser Block mit anliegenden freien Speicherblöcken *verschmolzen* zu einem großen Speicherblock. Der Zugriff auf die Speicherblockliste wird über den Semaphor **h_in_use** geregelt.

Variable	Adresse	Inhalt
bfs	6	Zeiger auf den ersten Speicherblock
efs	7	Zeiger auf das Ende des freien Speichers
h_in_use	26	Semaphor für den Zugriff auf die Speicherblockliste

Tabelle 60: Systemvariablen der Speicherverwaltung

Die Variable **efs** zeigt auf das erste Wort, das zum Cache-Bereich gehört. Durch die Speicherverwaltung wird nur der Bereich bis zu der Grenze **efs** verwaltet. Arbeitsspeicher hinter der Grenze efs wird von der Speicherverwaltung vom SCSI-Treiber, der den Cache verwaltet, angefordert und die Variable **efs** aktualisiert.

Die Varaible **bfs** zeigt hinter den Bereich der Systemvariablen. Zur Zeit werden 74 Systemvariablen benutzt, daß heißt **bfs** hat den Wert 80.

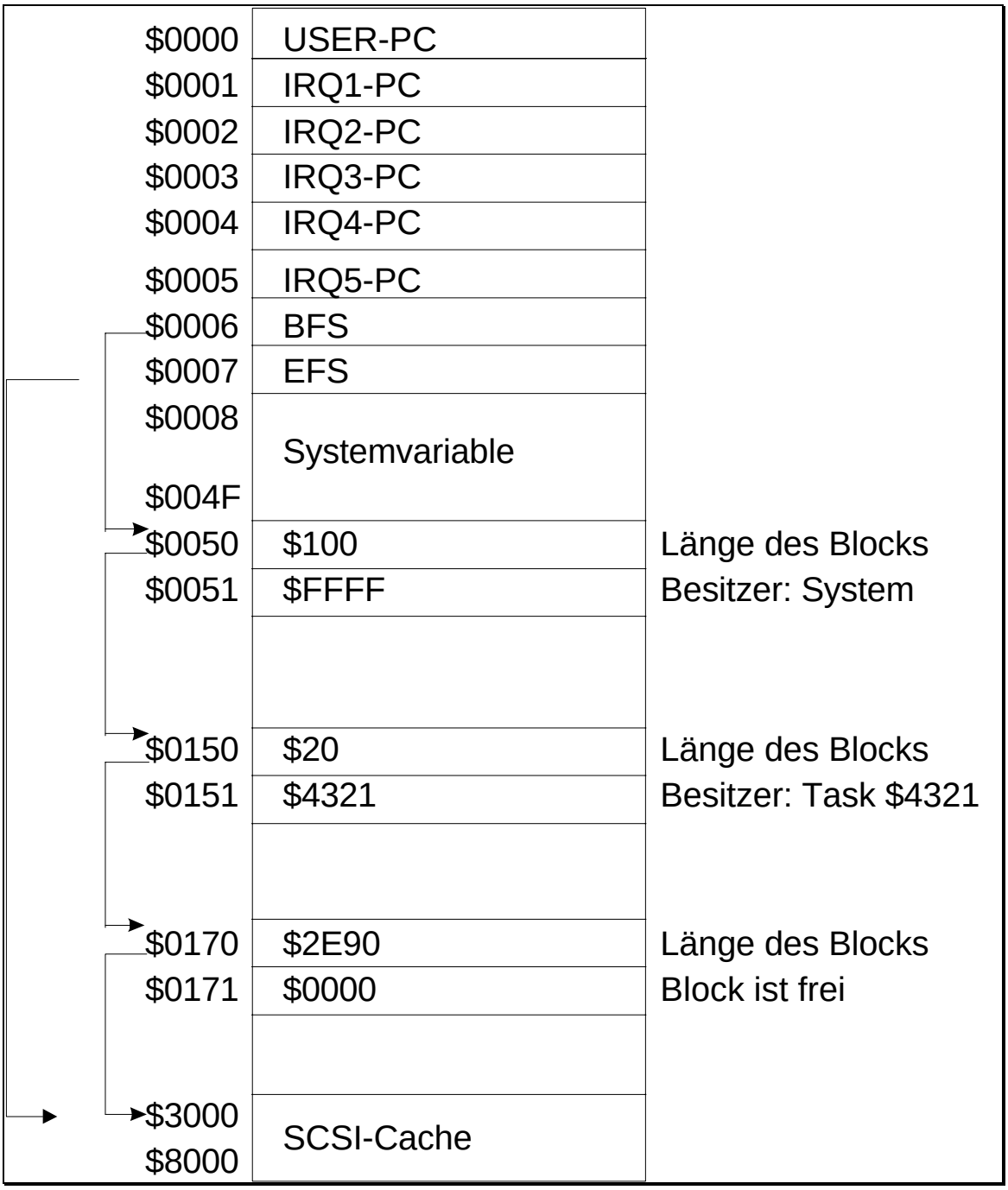


Bild 18: Organisation des Arbeitsspeichers

7.6. Taskverwaltung

7.6.1. Task-Descriptor-Blöcke

Die Taskliste ist als zyklische Liste von Task-Descriptor-Blöcken (auch Prozess-Leit-Blöcke genannt) organisiert. Ein Task-Descriptor-Block (TDB) bietet Platz für die zu sichernden Speicherzellen beim Taskwechsel, einen Verweis auf den nächsten Task-Descriptor-Block, den Tasknamen, die Adressen der File-Descriptor-Blöcke der StandardIn- und StandardOut-Dateien, und den Environmentbereich der Task. In diesem Environmentbereich wird die Currrent Working Directory, der Path und Parameter der Command Line abgelegt. Parameter der Command Line sind die Parameter die einem Programm durch die Shell übergeben werden. Beim Starten einer Task wird der Environment der startenden Task der zu startenden Task vererbt. Die Tasknummer ist die Anfangsadresse dieses TDBs im Speicher. Ein TDB mit einem Environmentbereich der Größe Null hat eine Länge von 22 Worten. Alle Betriebssystemroutinen operieren nur auf den Registern d0 bis d7. Diese müssen für jede Task neu lokal angelegt werden , dies geschieht im TDB. Bei einem Taskwechsel werden die Regsiterwerte des zu suspendierenden Task in den TDB gerettet, und die Werte der zu aktivierenden Task in die Register d0 bis d7 des Betriebssystem übertragen. So wird gewährleistet, das die Betreibssystemroutinen reentrant, und somit multitaskingfähig sind.

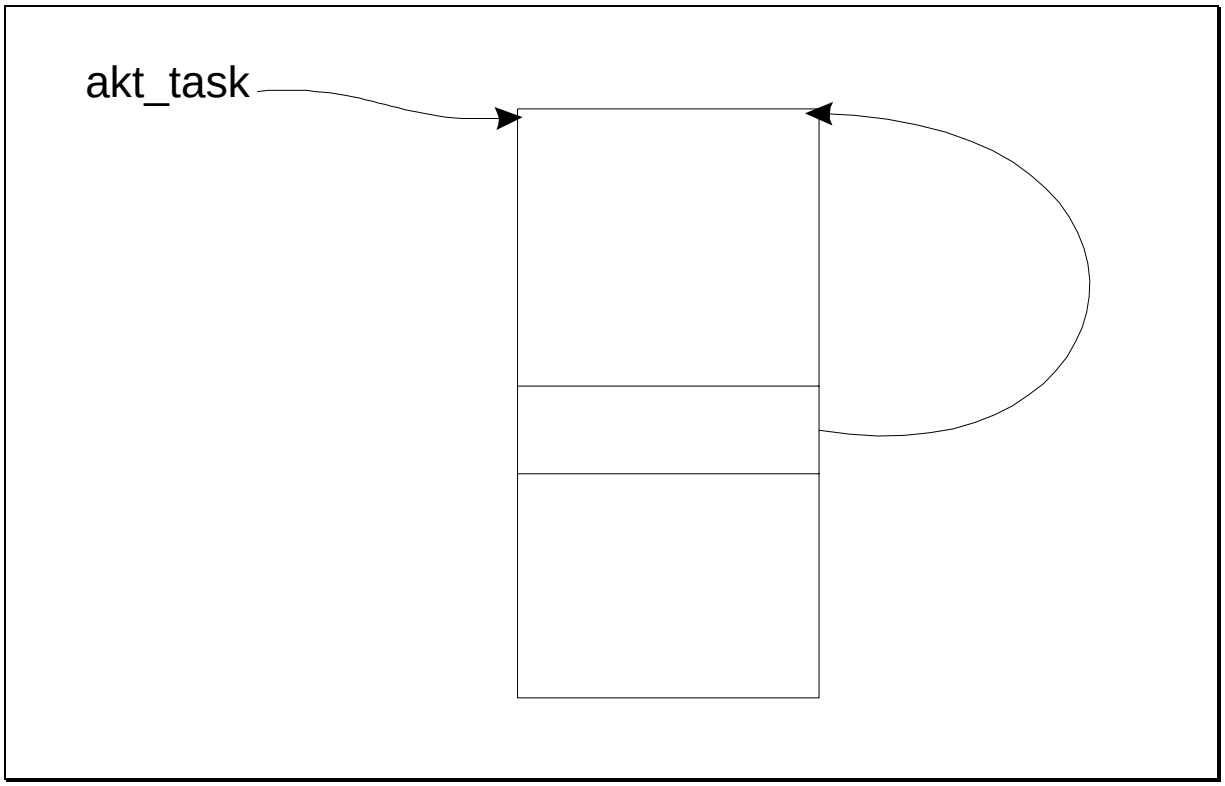


Bild 19: Taskliste mit einem TDB

Adresse	Inhalt
0	Gesicherter User-PC
1	Gesicherte Flags
2	Gesicherter Stackpointer
3	Gesichertes Register d0
4	Gesichertes Register d1
5	Gesichertes Register d2
6	Gesichertes Register d3
7	Gesichertes Register d4
8	Gesichertes Register d5
9	Gesichertes Register d6
10	Gesichertes Register d7
11	Adresse des nächsten TDB
12	Taskname, ein Zeichen
13	Taskname, ein Zeichen
14	Taskname, ein Zeichen
15	Taskname, ein Zeichen
16	Taskname, ein Zeichen
17	Taskname, ein Zeichen
18	Taskname, ein Zeichen
19	Taskname, ein Zeichen
20	FDB-Adresse der SatndardIn-Datei
21	FDB-Adresse der StandardOut-Datei
22	Environmentbereich, falls Environmentbereichsgröße gleich Null gewählt ist dies das letzte Wort des TDB und hat den Wert Null
23-...	Environmentbereich

Tabelle 61: Aufbau eines Task-Desciptor-Blockes

Der Systembefehl STARTTASK legt einen neuen TDB an und reiht ihn in die zyklische Liste ein. Das Einhängen in die Liste und das Entfernen aus der Liste erfolgt mit gesperrten Interrupts, damit dieser Vorgang nicht unterbrochen werden kann.

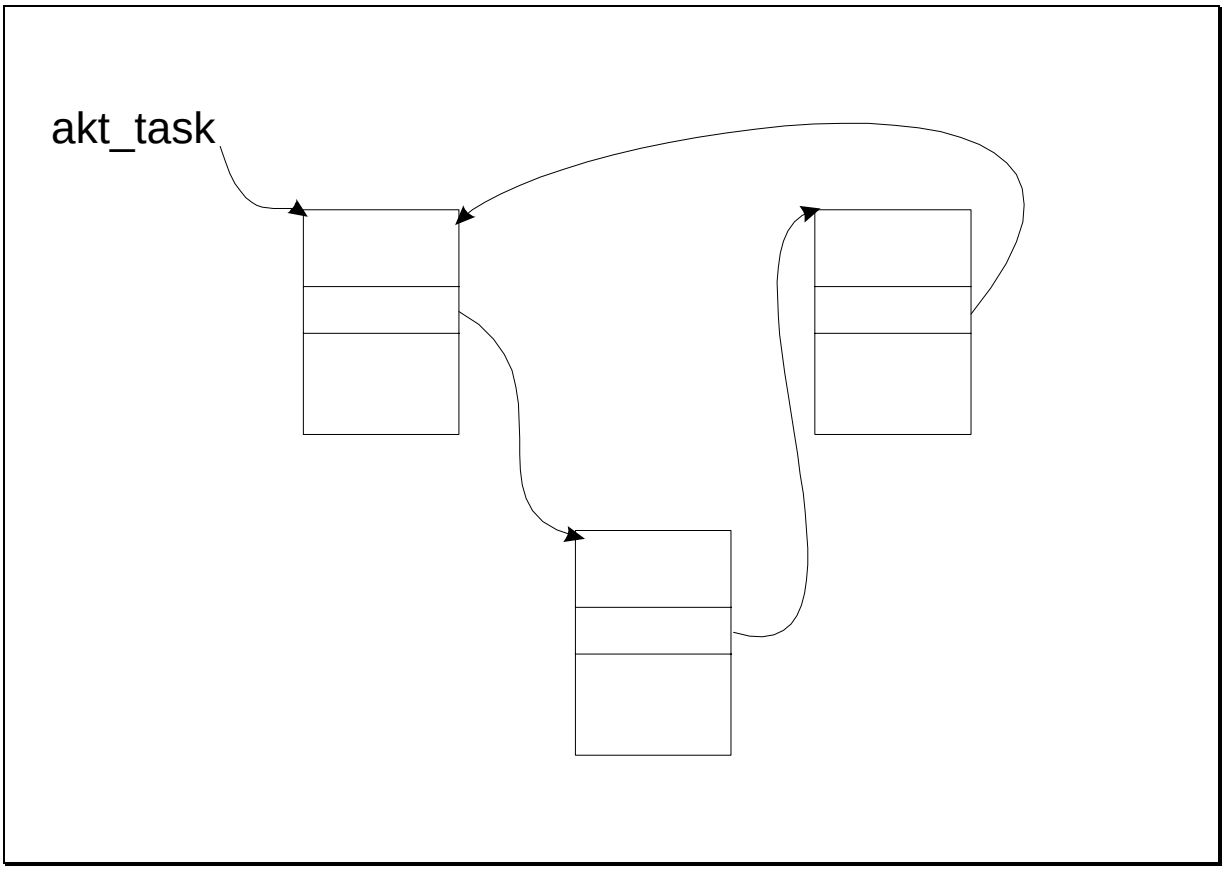


Bild 20: Zyklische TDB-Liste mit drei Tasks

7.6.2. Timer

Der Timer-Interrupt wird hardwaremäßig jede 1/16 Sekunde ausgelöst. Es wird dann die Variable **timerl** incrementiert, bei Überlauf auch die Variable **timerh**. Danach wird zur Taskwechselroutine SWITCHTASK gesprungen. Der Timer kann somit über 8 Jahre arbeiten, bis er wieder Null erreicht. Er besitzt die höchste Unterbrechungspriorität 5 und kann somit selber nicht unterbrochen werden.

Variable	Adresse	Inhalt
timerh	28	Zeitwert, High-Wort
timerl	29	Zeitwert, Low-Wort
flags5	30	gerettet Flags der Interruptroutine

Tabelle 62: Systemvariablen des Zeitgebers

7.6.3. Taskwechsel

Der Taskwechsel wird im Interruptlevel 5 durchgeführt, damit er nicht unterbrochen werden kann, auch nicht vom Timer. Beim Taskwechsel werden die Arbeitsregister d0 bis d7, die Flags, der Program-Counter und der Stackpointer in den TDB gerettet. Aus dem TDB der nächsten Task werden diese Speicherzellen wieder restauriert. Für diese Operation wird ein spezielles Arbeitsregister **task_switch_tmp** benutzt. Die Taskwechselroutine ist somit nicht reentrant, was aber auch nicht gefordert ist.

7.6.4. Task entfernen

Eine Task entfernt sich immer selbst. Durch den Befehl `rts` auf höchster Programmebene wird ein KILLTASK-Aufruf durchgeführt. Wird eine Task durch eine andere entfernt, wird der User-PC der zu entfernenden Task auf die Adresse der KILLTASK-Routine eingestellt. Auf dem Stack wird die Tasknummer der zu entfernenden Task abgelegt. Danach wird der Systembefehl beendet. Beim nächsten Taskwechsel begeht die Task "Selbstmord". Dabei werden folgende Aufgaben erledigt:

- Semaphore freigeben.
- Geöffnete Dateien schließen.
- Arbeitsspeicher freigeben.
- TDB aus Taskliste entfernen.

Variable	Adresse	Inhalt
akt_task	17	Aktuelle Tasknummer
task_switch_tmp	31	Arbeitsregister für Taskwechsel
resetvector	76	Adresse, die bei einem Reset durch KILLTASK angesprungen wird

Tabelle 63: Systemvariablen der Taskverwaltung

Wird die letzte Task entfernt wird durch die Routine KILLTASK ein RESET ausgelöst, dabei wird über die Variable **resetvector** gesprungen. Sie weist im Moment auf die Adresse der Betriebssysteminitialisierung, kann aber bei Bedarf verändert werden. So könnte zum Beispiel ein RAM-Disk-Treiber dort ein Programm eintragen, das bei einem RESET den Inhalt der RAM-Disk auf die Festplatte rettet und dann zur Betriebssysteminitialisierung springt.

7.7. Dateiverwaltung

7.7.1. Dateien und Treiber

Der Grundgedanke dieses Betriebssystem ist, dem Benutzer eine geräteunabhängige Systemsicht zu vermitteln. Geräte werden wie Dateien behandelt. Z.B. läßt sich die Datei `/b/Verz/Datei` mit den gleichen Operation behandeln wie das Gerät `/ser` (Serielle Schnittstelle). Es kann ein Wort, String oder Block in ein Zeichendevice oder in eine Datei geschrieben werden. Die Systembefehle übernehmen die Kommunikation mit den Treibern.

7.7.2. Dateistruktur

Als Dateistruktur wurde eine Baumstruktur gewählt. Ein Sektor muß eine Länge von 512 Bytes=256 Worte haben. In einer Datei wird logisch mit 16Bit Sektornummern gearbeitet, die eine mögliche Dateilänge von 65536 Sektoren = 32MByte zulasssen. Für die physikalischen Sektornummern stehen 32Bit zur Verfügung. Damit können 2^{32} Sektoren = 2 TeraBytes adressiert werden. Partitionierungen werden nicht unterstützt. Die freien, belegten und defekten Sektoren werden über eine BAM (Block Availible Map) verwaltet.

Es können beliebig viele Verzeichnisebenen erzeugt werden. Verzeichnisse werden wie Dateien behandelt.

Ist die Dateilänge kleiner oder gleich 256 Worte wird von der Datei ein Sektor, in dem die Daten abgelegt sind, belegt. Dieser erste Sektor, den eine Datei belegt, und auf den der Verzeichniseintrag verweist, heißt Primärsektor. Dieser ist hier Datensektor. Die Schachtelungstiefe, die angibt wieviel Verwaltungsebenen zur Datei gehören, ist Null.

Liegt die Dateilänge zwischen 2 Sektoren*256 Worten und 128 Sektoren*256 Worten=32768 Worten=64KByte, wird ein Verwaltungssektor benötigt. Dieser Verwaltungssektor ist nun der Primärsektor, auf den der Verzeichniseintrag zeigt. In diesem Sektor sind jeweils zwei Worte die Adresse eines Datensektors. Es existiert eine Verwaltungsebene, die Schachtelungstiefe ist Eins.

Bewegt sich die Dateilänge zwischen 128 Sektoren*256 Worten=64KByte und 128^2 Sektoren*256 Worten=8 MByte werden bis zu 129 Verwaltungssektoren in zwei Verwaltungsebenen benötigt, die Schachtelungstiefe ist Zwei. Der Primärsektor ist ein Verwaltungssektor, der wiederum mit 32Bit-Adressen auf bis zu 128 weitere Verwaltungssektoren zeigt. Diese wiederum zeigen mit 32Bit-Adressen auf die Datensektoren.

Für eine Datei mit einer Größe zwischen 8MByte und 32MByte werden drei Verwaltungsebenen benötigt, die Schachtelungstiefe ist Drei. Der Primärsektor ist ein Verwaltungssektor, der auf bis zu 128 Verwaltungssekotren zeigt, diese wiederum zeigen auf Verwaltungssektoren, die dann auf die Datensektoren zeigen.

Bei einer Schachtelungstiefe von Drei sind in der zweiten Verwaltungsebene allerdings durch die Beschränkung von 65536 logischen Sektoren nur vier Verwaltungssektoren möglich. Das heiß, bei dieser Schatelungstiefe existieren maximal 1+4+4*128=517 Verwaltungssektoren.

Es werden immer nur soviele Verwaltungsektoren benutzt, wie zur Verwaltung nötig sind. Das heißt, bei einer Schachtelungstiefe von Drei, werden nicht automatisch 517 Verwaltungssektoren eingetragen.

Selbst bei einer Dateigröße von 32MByte müssen nur drei Verwaltungssektoren gelesen werden, bis die Adresse des Datensektors ermittelt ist, egal ob es der erste oder der letzte der Datei ist.

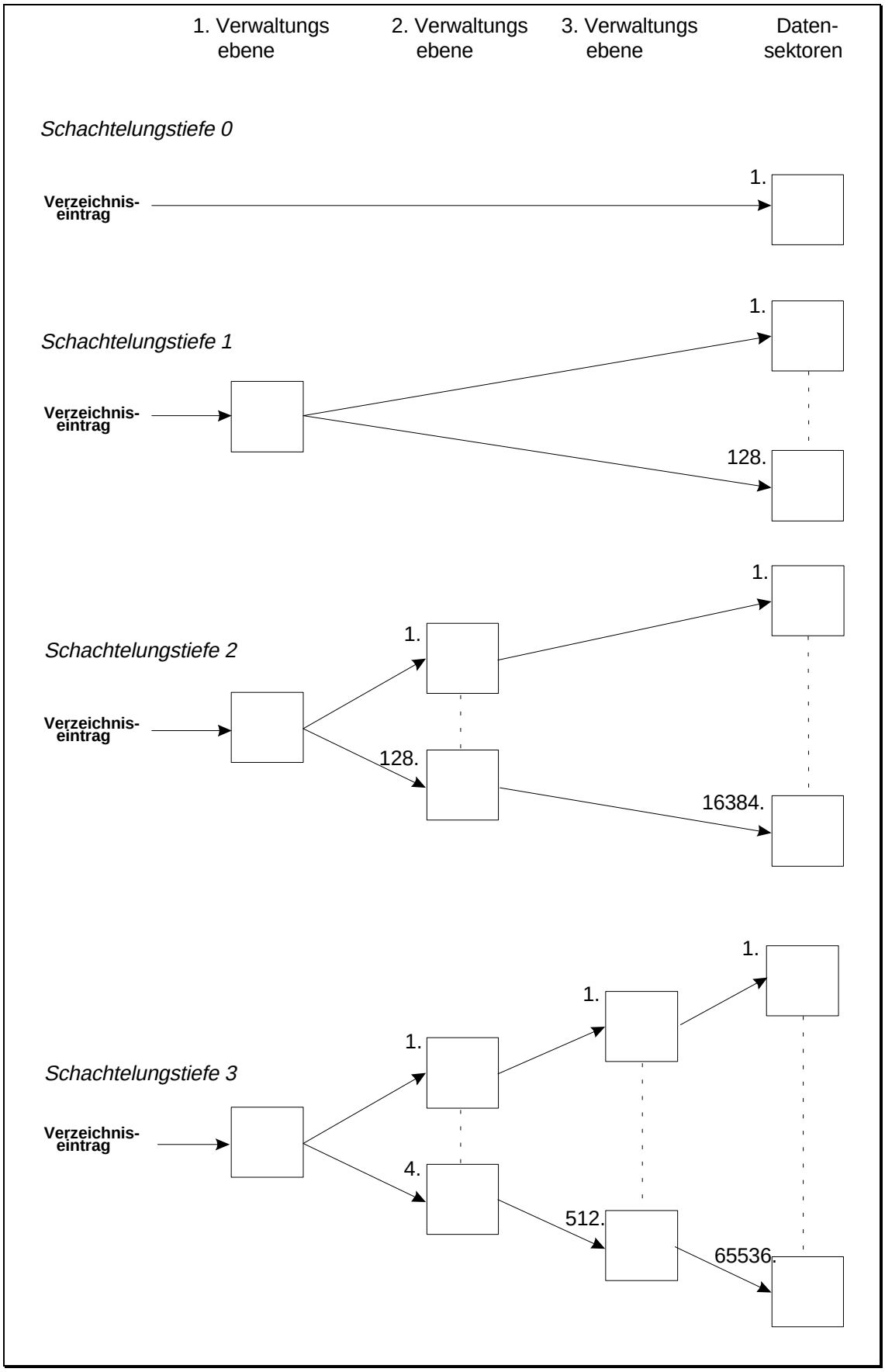


Bild 21: Dateistruktur

7.7.3. Verzeichniseinträge

Ein Verzeichniseintrag hat eine Länge von 16 Worten, so daß in ein Datensektor eines Verzeichnisses 16 Verzeichniseinträge passen. Ein Verzeichniseintrag enthält 8 Worte Verwaltungsinformation und 8 Worte Dateinamen. Der Dateiname liegt gepackt vor, er kann also bis zu 16 Zeichen lang sein. Und zwar wird das erste Zeichen im High-Byte und das zweite Zeichen im Low-Byte abgelegt. Zulässig sind Zeichen mit dem ASCII-Code $21 bis $7f. Die Dateilänge wird in Sektoren und Worten angegeben. Dabei beschreibt die Anzahl der Sektoren vollständig belegte Sektoren und die Anzahl der Worte eine Anzahl von Worten im letzten Dateisektor, die kleiner ist als 256. Bei einer Datei mit 200 Worten Länge ist die Dateilänge in Sektoren Null und die Anzahl der Worte 200.

Ein Verzeichniseintrag hat die folgende Form:

Adresse	Inhalt
0	Adresse des Primärsektors, High-Wort
1	Adresse des Primärsektors, Low-Wort
2	Dateilänge in Sektoren
3	Dateilänge in Worten <256, d.h. soviel Worte werden zusätzlich zu einem Sektor noch belegt.
4	Schachtelungstiefe (Anzahl der Verwaltungsebenen; zwischen 0 und 3)
5	Dateiattribute, bisher werden nur drei Bit benutzt (Bit_2: Writeprotect, Bit_1: Executable, Bit_0: Directory)
6	timerh, Erstellungszeitpunkt
7	timerl, Erstellungszeitpunkt
8	Dateiname, zwei Zeichen, insgesamt 16 Zeichen
9	Dateiname, zwei Zeichen
10	Dateiname, zwei Zeichen
11	Dateiname, zwei Zeichen
12	Dateiname, zwei Zeichen
13	Dateiname, zwei Zeichen
14	Dateiname, zwei Zeichen
15	Dateiname, zwei Zeichen

Tabelle 64: Aufbau eins Verzeichniseintrages

Ein Verzeichniseintrag ist ungültig, wenn das erste Wort des Dateinamens Null ist.

Es werden bisher nur drei Dateiattribute benutzt. Wenn der Bedarf nach weiteren Attributen besteht kann dieses Attribut-Wort dafür noch genutzt werden. Entsprechend muß dann das Dienstprogramm DIR und ATTRIB erweitert werden.

7.7.4. Verzeichnisstruktur

Ein Verzeichnis ist eine normale Datei, mit der alle Dateioperationen durchgeführt werden können und die die gleiche Struktur hat wie jede andere Datei. Als Dateiattribut ist das Directory-Flag gesetzt. So können in ein Verzeichnis beliebig viele Unterverzeichnisse eingebunden werden, und es können beliebig tiefe Unterverzeichnissebenen benutzt werden.

Zu beachten ist das die Dateilänge eines Verzeichnisses nicht die Summe der Längen der enthaltenen Dateien ist, sondern nur die Summe der durch die Verzeichniseinträge belegten Worte angibt. Wird ein Verzeichniseintrag gelöscht wird die Datei nicht verkürzt, sondern der Eintrag nur als ungültig erklärt. Die Verzeichnislänge ändert sich dann also nicht.

7.7.5. Bootsektor

Der Bootsektor eines Datenträgers enthält einen Verzeichniseintrag, eine 32Bit-Kennung, Informationen über die Länge der Block Available Map (BAM, siehe Kapitel 7.7.7) in Sektoren und die Position des ersten freien Sektors in der BAM. Der Rest des Sektors wird nicht belegt und ist mit Nullen gefüllt. Der Verzeichniseintrag enthält Informationen über das Hauptverzeichnis.

Adresse	Inhalt
0	Primärsektor der Rootdirectory, High-Wort
1	Primärsektor der Rootdirectory, Low-Wort
2	Länge der Rootdirectory in Sektoren
3	Länge der Rootdirectory in Worten <256, (siehe oben)
4	Schachtelungstiefe der Rootdirectory
5	Dateiattribute (Directory-Flag ist gesetzt)
6	timerh, Erstellungszeitpunkt
7	timerl, Erstellungszeitpunkt
8	Datenträgername, insgesamt 16 Zeichen
9	Datenträgername, zwei Zeichen
10	Datenträgername, zwei Zeichen
11	Datenträgername, zwei Zeichen
12	Datenträgername, zwei Zeichen
13	Datenträgername, zwei Zeichen
14	Datenträgername, zwei Zeichen
15	Datenträgername, zwei Zeichen
16	Kennung, momentan timerl, könnte durch einen Zufallswert ersetzt werden.
17	Länge der BAM in Sektoren, Startadresse der BAM ist immer Sektor 1, direct noch dem Bootsektor.
18	Sektornummer, des BAM-Sektors in dem der nächste freie Sektor eingetragen ist.
19	Wortposition im BAM-Sektor, in dem der nächste freie Sektor eingetragen ist.

Tabelle 65: Aufbau des Bootsektors

Der Bootsektor enthält keinerlei Information über die Kapazität und die Anzahl der Sektoren des Datenträgers. Dies können leicht über das SCSI-Kommando READ CAPACITY ermittelt werden. Die Länge eines Datensektors beträgt in diesem System immer 512 Bytes.

Zu beachten ist, daß der in diesem Verzeichnis angegeben Name als Datenträgername verwendet wird. Dieser Datenträgername wird in die Media-Descriptor-Blöcke (siehe SCSI-Treiber) übernommen, um einen Datenträger eindeutig zu identifizieren. An der Adresse 16 wird eine Datenträgerkenung oder Seriennummer abgelegt, die eine eindeutige Identifizierung des Datenträgers unterstützen soll. Zur Zeit ist dieser Wert der Inhalt des Zählers **timerl**. Falls das System um einen Zufallswertgenerierung erweitert wird, sollte dieser Wert als Kennnung eingetragen werden. An Adresse 18 wird nach der Formatierung eine 1 eingetragen, an Adresse 19 eine 0. Dort wird nach der Suche eines 0-Bits (freier Sektor) in der BAM begonnen. Nach der Allokierung eines Sektors werden die Adressen 18 und 19 neu gesetzt, auf die Adresse des gerade gefundenen Bits. Wird ein Sektor wieder freigegeben wird der Wert verglichen und bei Bedarf aktualisiert.

7.7.6. File-Descriptor-Blöcke

Geöffnete Dateien werden über einer Liste von File-Descriptor-Blöcke (FDB) verwaltet. Als Handlenummer wird die Startadresse des FDB im Arbeitsspeicher benutzt. Es ist daher möglich, beliebig viele Dateien zu öffnen und eindeutige Handlenummern zu vergeben. FDBs haben eines Länge von 20 Worten und werden für Block- und Zeichengeräte gleich aufgebaut und benutzt. Aufgrund dieser FDBs wird durch die OPEN-Funktion ein gemeinsamer Lese- und ein exklusiver Schreibzugriff geregelt.

Adresse	Inhalt
0	Adresse des nächsten FDB, falls Null war dies der letzte.
1	Flags, drei Bits benutzt (Bit_2: zum Schreiben geöffnet, Bit_1: zum Lesen geöffnet, Bit_0: 0=Blockdevice, 1=Zeichendevice).
2	Adresse des Gerätetreibers
3	Adresse des Verzeichniseintrages, Sektor, High-Wort
4	Adresse des Verzeichniseintrages, Sektor, Low-Wort
5	Adresse des Verzeichniseintrages, Wortposition im Sektor
6	Dateizeiger, Sektor
7	Dateizeiger, Worte <256
8	Magicwort 1, $1234, wird bei Dateioperationen überprüft
9	Magicwort 2, $5678, wird bei Dateioperationen überprüft
10	Adresse des Primärsektors, High-Wort
11	Adresse des Primärsektors, Low-Wort
12	Dateilänge in Sektoren
13	Dateilänge in Worten <256, d.h. soviel Worte werden zusätzlich zu einem Sektor noch belegt.
14	Schachtelungstiefe (Anzahl der Verwaltungsebenen; zwischen 0 und 3)
15	Dateiattribute, bisher werden nur drei Bit benutz (Bit_2: Writeprotect, Bit_1: Executable, Bit_0: Directory)
16	timerh, Erstellungszeitpunkt
17	timerl, Erstellungszeitpunkt
18	Arbeitswort für Dateioperationen
19	Arbeitswort für Dateioperationen

Tabelle 66: Aufbau eines File-Descriptor-Blockes

Da Dateioperationen häufig ein oder zwei Adressen als Arbeitsregister benötigen wurden diese in den FDB aufgenommen. So wurden keine extra Arbeitsregister benötigt, die bei einem Taskwechsel hätten gerettet und restauriert werden müßten. Eine andere Alternative wäre gewesen, das bei diesem Dateioperationen Arbeitsspeicher allokiert wird. Das hätte unnötige Zeit beansprucht und eine Fehlerquelle mehr aufgetan.

Ein FDB für ein Zeichendevice hat die gleiche Länge, benutzt werden allerdings nur die ersten drei Worte Verwaltungsinformation, die beiden Magicworte zur Erkennung und die beiden Arbeitsregister am Ende des FDB. Der Einheitlichkeit halber wurde keine zweite Form eines FDB geschaffen.

Die beiden Magic-Worte werden bei jedem Systembefehl, der mit Dateioperationen zu tun hat, überprüft, damit nicht auf ungültigen Daten operiert werden kann.

Beim Schließen einer Datei werden die Magic-Worte gelöscht und der FDB wieder freigegeben. Es kann nur eine Datei der eigenen Task geschlossen werden. Diese Information wird aus den Verwaltungsworten der Speicherverwaltung gewonnen, die dem FDB vorangestellt sind.

Die FDB werden am Anfang der Liste eingereiht, die Variable **fdblist** zeigt auf den ersten FDB. Existiert kein FDB hat die Variable **fdblist** den Wert Null. Der verändernde Zurgiff auf die FDB-Liste, daß heißt das Einhägen eines FDB oder das Entfernen ein es FDB wird über den Semaphor **fdblist_in_use** geregelt. Auch das Allokieren eines neuen Datenträgersektors wird über den Semaphor **allocate_in_use** geregelt, damit nicht zwei Task den gleichen Sektor als frei ermitteln und belegen.

7.7.7. Block-Available-Map

Die freien, belegten und defekten Sektoren werden über einen Bitvektor verwaltet. Ein Bit gibt Auskunft über einen Sektor:

0: Sektor ist frei

1: Sektor ist nicht verfügbar, belegt oder defekt.

Dadurch, das nur ein Bit Information pro Sektor benutzt wird, benötigt eine 1.44 MByte Diskette nur einen einzigen BAM-Sektor. Für die Datenträgerverwaltung werden also nur zwei Sektoren, Bootsektor und BAM-Sektor, benutzt. Das sind 0,0014% der Speicherkapazität des Datenträgers. Die nicht benötigten Bits des letzten BAM-Sektors werden mit 1 belegt, so ist keine Angabe über die Wortlänge der BAM nötig.

Die Worte mit den Adressen 18 und 19 im Bootsektor sollen die Suche nach freien Sektoren beschleunigen. An der dort angegebenen Adresse wird nach dem nächsten freien Sektor gesucht. Wird ein freier Sektor gefunden wird diese Adresse dort wieder eingetragen, da zu erwarten ist, daß das nächste Bit wieder einen freien Sektor kennzeichnet. Werden Sektoren wieder freigegeben wird dieser Wert aktualisiert.

Wird eine neuer Sektor angefordert wird testweise auf diesen lesend zugegriffen. Es wird davon ausgegangen, daß ein Sektor der ordnungsgemäß gelesen werden kann auch ordnungsgemäß geschrieben werden kann, er also nicht defekt ist. War der Zugriff nicht erfolgreich wird der Sektor als nicht verfügbar gekennzeichnet und ein neuer angefordert.

Variable	Adresse	Inhalt
fdblist	73	Zeiger auf den ersten FDB
fdblist_in_use	74	Semaphor
allocate_in_use	75	Semaphor

Tabelle 67: Systemvariablen der Dateiverwaltung

7.8. Treiberverwaltung

7.8.1. Treiberliste

Die im System angemeldeten Treiber werden in ein auf 32 Einträge begrenzte Liste eingetragen. Das Entfernen eines Treibers aus dieser Liste ist nicht vorgesehen. Die Anfangsadressen der Treiber werden in diese Liste eingetragen. Haben zwei Treiber den gleichen Namen, wird immer der zuerst im System angemeldete Treiber gefunden, da er als erste in der Liste auftaucht. Die Variable **driverlist** zeigt auf diese Liste. Beim Aufruf SEARCHDRIVER wird in dieser Liste gesucht. Das Eintragen eines Treibers in die Liste erfolgt mit gesperrten Interrupts, damit nicht zwei Task zwei Treiber auf den gleichen Listenplatz eintragen. Gerätename und Treibername bedeuten im Folgenden das Gleiche.

7.8.2. Aufbau

Zeichen- und Blockdevicetreiber haben den gleichen Aufbau, sie müssen nur andere Jobs verarbeiten. Die ersten vier Worte eines Treibers geben den Treibernamen ungepackt an. Das fünfte Wort gibt Auskunft darüber ob es ein Zeichen- oder Blockdevicetreiber ist.

Adresse	Inhalt
0	Treibername, ein Zeichen, insgesamt vier Zeichen
1	Treibername, ein Zeichen
2	Treibername, ein Zeichen
3	Treibername, ein Zeichen
4	0=Blockdevicetreiber/1=Zeichendevicetreiber
5	Anfang des Treibers

Tabelle 68: Declarationskopf eines Gerätetreibers

SEARCHDRIVER übergibt die relative Treiberadresse 5. Eingetragen in diese Treiberliste wird die relative Treiberadresse 0, diese muß auch der Funktion NEWDRIVER übergeben werden.

Variable	Adresse	Inhalt
driverlist	27	Zeiger auf die Treiberliste

Tabelle 69: Systemvariablen der Treiberverwaltung

7.8.3. Zeichendevice-Treiber

Ein Zeichendevicetreiber muß drei Jobs erfüllen.

1. Job: Ein Zeichen lesen
2. Job: Ein Zeichen schreiben
3. Job: Einen Null-terminierten String schreiben

Es wird ein Zeichen in einem Wort übergeben. Bei einer Stringausgabe wird jeweils nur das niederwertige Byte als Zeichen ausgegeben. Es ist Sache des Treibers, im gegebenen Fall die Fehler "End-Of-File erreicht!" und "Es liegt kein Zeichen an!" zu erzeugen. Er kann darüberhinaus über mehr Jobs verfügen, die direkt durch Treiberaufrufe und nicht durch Dateioperationen ausgeführt werden können. Bei Aufruf des Jobs 1 soll das gelesene Wort im Register d0 übermittelt werden. Für den Job2 muß das Zeichen als Parameter übergeben werden, für den Job 3 ein Zeiger auf den String. Fehlercodes werden immer über das Register d0 übermittelt. Bsp.: Treiberaufruf für ein Zeichendevice:

```
push   Parameter                    ;entfällt bei Job 1
push   Jobnummer
driver Startadresse des Treibers
add    #2,sp                        ;bei Job 1 inc sp
```

7.8.4. Blockdevice-Treiber

Ein Blockdevicetreiber muß vier Jobs erfüllen.

1. Job: Zwischen 1 und 256 Worten innerhalb eines Blocks lesen
2. Job: Zwischen 1 und 256 Worten innerhalb eines Blocks schreiben
3. Job: Kapazität des Datenträgers ermitteln
4. Job: Datenträger Low-Level-formatieren
5. Job: NOP, reserviert für SCSI-Treiber
6. Job: NOP, reserviert für SCSI-Treiber

Als Blockadresse wird die physikalische Adresse des Sektors übergeben. Er kann darüberhinaus über mehr Jobs verfügen, die direkt durch Treiberaufrufe und nicht Dateioperationen ausgeführt werden können.

Als Parameter müssen dem Treiber die Sektornummer, eine Pufferadresse, die Azahl der zu übertragenden Worte, und das erste Wort des Sektors, ab dem übertragen werden soll (zwischen 0 und 255), übergeben werden. Der Job 3 benötigt nur die Adresse eines 4-Wort-Puffers. Die Daten sind in der folgenden Weise in diesem 4-Wort-Puffer abgelegt:

0: Adresse des letzten Sektors, High-Wort
1: Adresse des letzten Sektors, Low-Wort
2: Bytes pro Sektor, High-Wort
3: Bytes pro Sektor, Low-Wort

Der Job 4 benötigt keine Parameter. Ein Treiberaufruf hat folgende Form:

```
push   Anzahl der zu übertragenden Worte  ;entfällt bei Job 3 und 4
push   Nummer des ersten Wortes in Sektor ;entfällt bei Job 3 und 4
push   Sektornummer, Low-Wort             ;entfällt bei Job 3 und 4
push   Sektornummer, High-Wort            ;entfällt bei Job 3 und 4
push   Pufferadresse                      ;entfällt bei Job 4
push   Jobnummer
driver Startadresse des Treibers
add    #6,sp            ;Bei Job 3: add #2,sp, Bei Job 4: inc sp
```

Die Pufferung der Sektoren muß durch den Treiber gewährleistet werden, da dieser Wortoperationen zulassen soll. Fehlercodes werden im Register d0 übermittelt.

7.8.5. SCSI-Treiber

Cache und weitere Jobs

Der SCSI-Treiber ist der aufwendigste Treiber des System. Da der Cache zum SCSI-Treiber gehört braucht sich das Betriebssystem nicht um eine Cache-Verwaltung zu kümmern. Alle Sektoren werden vom SCSI-Treiber angefordert, der Cache erhöht nur die Geschwindigkeit der Auftragserfüllung. Aufgrund des Caches sind deshalb zwei zusätzliche Jobs:

Job 5:Cache-Puffer reduzieren
Job 6:Cache-Flush

Der Job 5 wird von der Speicherverwaltung aufgerufen, wenn nicht genügend Arbeistspeicher vorhanden ist. Mit dem Job 5 wird als einziger Parameter eine Adresse im Arbeitsspeicher übergeben, bis zu der der Cachebereich reduziert werden muß. Die Speicherverwaltung überprüft, ob die Reduzierung möglich ist. Der SCSI-Treiber führt keine weitere Überprüfung durch, sondern führt den Job nur aus. Dieser Job 5 ist auch nur für die Speicherverwaltung gedacht, nicht für ein Anwendungsprogramm.

Der Aufruf hat die Form:

```
push   Speicheradresse
push   #5                          ;Jobnummer
driver Startadresse der Treibers
add    #2,sp
```

Der Job 6 wird von dem Systembefehl CLOSE aufgerufen, um beim Schließen einer Datei alle neu erstellten Informationen wieder auf dem Datenträger abzulegen. Dieser Job 6 wird zwar immer von dem Systembefehl CLOSE aufgerufen muß aber anderen Blockdevice-Treibern nicht bekannt sein, da keine Fehlerüberprüfung nach dem Aufruf erfolgt, darf jedoch auch nicht anderweitig belegt werden. Der Job 6 benötigt keinen Parameter. Der Flush bezieht sich auf alle SCSI-Geräte. Die Jobnummern 5 und 6 sind also reserviert. Der Aufruf hat die Form:

```
push   #6                          ;Jobnummer
driver Startadresse der Treibers
inc    sp
```

Ein Cache-Puffer enthält vier Worte Verwaltungsinformation. Das erste Wort enthält in den niederwertigsten drei Bit_{2-0} die SCSI-Device-ID, Bit_3 ist ein Schreibflag, das anzeigt ,ob es sich um einen Schreib-Cache-Puffer handelt und das vierte Bit_4 gibt an, ob der Inhalt eines Puffer gültig ist. Zwei Worte sind die physikalische Sektoradresse und das vierte Wort ist der Zugriffszähler. Danach folgen 256 Worte Daten.
Das erste Verwaltungswort und der Zugriffszähler werden gelöscht, wenn ein Cache-Puffer als ungültig erklärt wird. Die Cache-Puffer folgen alle aufeinander und beginnen ab der Adresse **efs**.

Adresse	Inhalt
0	Bit_{2-0}: SCSI-Device-ID Bit_3: Schreibflag (1=Schreibpuffer) Bit_4: Gültigkeitsbit (1=Puffer ist gültig)
1	Sektornummer, High-Wort
2	Sektornummer, Low-Wort
3	Zugriffszähler
4-259	Daten

Tabelle 70: Aufbau eines Cache-Puffers

Die Jobs 1 und 2 werden bei Zugriffen auf das Medium immer mit anschließendem Vergleich durchgeführt, es handelt sich um die standardisierten SCSI-Kommandos WRITE&VERIFY und EXTENDED READ.

Der Cache bietet für wechselbare Datenträger eine Gefahr, und zwar das auf dem Cache schreibend operiert wird, der Datenträger aber schon aus dem Laufwerk entfernt wurde. Der Fehler wird dann erst beim FLUSH entdeckt. Um diesen Fehler zu vermeiden wird bei jeder Schreibaufforderung auf ein wechselbares Medium der SCSI-Befehl TEST-UNIT-READY ausgeführt. Dieser antwortet mit dem Fehler UNIT ATTENTION, wenn das Medium gewechselt wurde. So ist eine Fehlerbehandlung beim Zugriff auf den Cache schon möglich, nicht erst beim Schließen der Datei. Bei dieser Fehlerbehandlung wird nicht nur überprüft, ob noch zu schreibende Cache-Puffer existieren, sondern auch ob noch geöffnete Dateien für dieses Device existieren. Die Anzahl der geöffnete Dateien für dieses Device wird über die FDB-Liste bestimmt. Lese-Cache-Puffer werden als ungültig erklärt. Tritt einer dieser Fälle ein wird der Benutzer aufgefordert den zuvor benutzten Datenträger wieder einzulegen oder eine Dateninkonsistenz hinzunehmen. Die Erkenunng eines Mediums erfolgt über Media-Descriptor-Blöcke, die beim Einlegen des Datenträgers verglichen werden. Der Befehl TEST-UNIT-READY ist ein sehr schnelles SCSI-Kommando, bei dem insgesamt nur 8 Bytes (6 Bytes Kommando, 1 Byte Statusmeldung, 1 Byte Message) übertragen werden.

Auch das Schreiben in den Cache für ein Schreibgeschütztes Medium ist nicht möglich. Beim Einlegen eines Datenträgers wird sofort die Beschreibbarkeit überprüft und in der Variablen **media_writeprot** in Form eines gesetzten Bits vermerkt. Diese Variable wird bei Schreibzugriffen überprüft.

Media-Descriptor-Blöcke

Ein Media-Descriptor-Block (MDB) wird für jedes SCSI-Device angelegt. Er hat eine Größe von 8 Worten (7 Worte Datenträgername, 1 Wort Kennung) und wird jedesmal neu erstellt, wenn ein Datenträger gewechselt wird oder schreibend auf den Sektor 0 zugegriffen wird. Dies ist nötig, damit beim High-Level-Format der neue Datenträgername und die neue Kennung in den MDB übernommen wird. Die Variable **media_descrpt** zeigt auf den ersten MDB. Die MDBs folgen im Arbeitsspeicher direkt aufeinander. Der Speicherplatz wird bei der Initialisierung angefordert.

Adresse	Inhalt
0	Datenträgername, 1 Zeichen
1	Datenträgername, 1 Zeichen
2	Datenträgername, 1 Zeichen
3	Datenträgername, 1 Zeichen
4	Datenträgername, 1 Zeichen
5	Datenträgername, 1 Zeichen
6	Datenträgername, 1 Zeichen
7	Seriennummer

Tabelle 71: Aufbau eines Media-Descriptor-Blockes

Zugriff und Fairness

Der Zugriff auf den SCSI-Treiber wird über den Semaphor **scsi_in_use** geregelt. Da SCSI eines der häufigst benutzten Betriebsmittel ist, gibt es eine einfache Fairness-Regel. Ist SCSI nicht verfügbar wird in der Variable **want_to_use_scsi** die Tasknummer der wartenden Task eingetragen. Vor dem Verlassen des SCSI-Treibers wird überprüft, ob diese Variable den Wert Null hat. Ist dies nicht der Fall wird die Variable gelöscht und ein Taskwechsel ausgelöst. So soll vermieden werden. daß bei ungünstigen Taskwechselzeitpunkten lange dieser Zugriff einer Task verwehrt bleibt. Wird die Schedulingmethode gewechselt besteht die Möglichkeit in dieser Variable zu erfahren, welche Task wartet und einen Taskwechsel gezielt zu dieser Task durch zuführen. Die Variable **want_to_use_scsi** darf dann erst von der Taskwechselroutine gelöscht werden.

Kooperation eines Low-Level-SCSI-Kommandos

Die SCSI-Kommandos können nach einem einfachen Schema bearbeitet werden. Das Schema ist hier um Kooperation zu ermöglichen geändert worden. Ein SCSI-Kommando löst selbständig einen Taskwechsel aus, falls einer der folgenden Situationen eintritt:

1. Nach Kommandoübermittlung beim Lesen (bis die Daten eintreffen).
2. Nach der Datenübermittlung beim Schreiben (denn der Status wird erst nach einer Überprüfung (WRITE&VERIFY) übermittelt).
3. Nach der Kommandoübermittlung beim Formatieren (denn der Status wird nach Beendigung des Formatiervorgangs übermittelt).

Variable	Adresse	Inhalt
status	64	SCSI-Status
accesses	65	Gesamtzugriffszähler für Cache
media_writeprot	66	Information über schreibgeschützte Medien Bit_0=1:SCSI-Device 0 ist schreibgeschützt Bit_1=1:SCSI-Device 1 ist schreibgeschützt ....
media_descr	67	Zeiger auf MDB-Liste
not_ready_active	68	Zeigt an, ob die Fehlerbehandlungsroutine bei Datenträgerwechsel mit noch geöffneten Datei aktive ist
scsi_in_use	69	Semaphor
scsi_buffer	70	Zeiger auf Arbeitspuffer des SCSI-Treibers
command_buffer	71	Zeiger auf 10-Wort-Kommandopuffer
want_to_use_scsi	72	Fairness-Flag
removable	76	Information über wechselbare Medien Bit_0=1: Medium in SCSI-Device 0 ist wechselbar Bit_1=1: Medium in SCSI-Device 1 ist wechselbar ....

Tabelle 72: Systemvariablen des SCSI-Treibers

Der SCSI-Treiber benötigt einen eigenen Sektor-Puffer als Arbeitspuffer. Er wird bei der Initialisierung benötigt um die Herstellerbezeichnung einzulesen. Viel wichtiger ist er bei der Fehlerbehandlung, wenn ein Datenträger gewechselt wurde, es aber noch geöffnete Dateien für diesen Datenträger gibt. Dann muß beim Einlegen eines Datenträgers der Bootsektor gelesen werden, um ihn mit den Informationen des MDB zu vergleichen und festzustellen, ob es der gleiche Datenträger ist. Für diese Operation darf kein Cache-Puffer benutzt werden.

Desweiteren wird ein 10 Wort Kommandopuffer benötigt in dem die SCSI-Kommandos zusammenkopiert werden. Die Kommandos sind 6 oder 10 Byte lang, und werden mit einem Byte pro Wort abgelegt.

Die Variable **not_ready_active** zeigt an, ob eine Fehlerbehandlung durch den SCSI-Treiber schon aktiv ist, damit nicht mehrere Tasks versuchen den gleichen Fehler zu behandeln.

Variable	I/O-Adresse	Inhalt
scsi_data	$7ff4	Datenregister des SCSI-Kontrollers
scsi_ctrl	$7ff5	Kontrollregister des SCSI-Kontrollers

Tabelle 73: SCSI-Kontroller-Adressen

Initialisiserung

In der Initialisierungphase ermittelt der Treiber selbständig wieviele und welche SCSI-Geräte angeschlossen sind. Ein neues SCSI-Gerät muß dem Treiber nicht angezeigt werden. Die ID's der angeschlossenen SCSI-Geräte müssen alle in aufsteigender Folge von Null an eingestellt sein, dann werden sie alle vom Treiber gefunden. Es werden alle Geräte von der ID Null an angesprochen und erst abgebrochen, falls eines nicht antwortet. In der Initialisierungsphase wird auch festgestellt, welches Gerät mit wechselbaren Medien arbeitet, dies ist bei der Cache-Bearbeitung von Bedeutung. Für jedes gefundene SCSI-Gerät wird ein MDB angelegt. Die Herstellerbezeichnungen werden auf der sichtbaren Konsole angezeigt.

Den SCSI-Geräten werden entsprechende den IDs Kleinbuchstaben zugeordnet:

SCSI-Device-ID	Gerätename
0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h

Tabelle 74: Zuordnung der SCSI-Gerätenamen entsprechend der ID

7.8.6. Konsolen-Treiber

Die Konsolen con0, con1 und err:

Das System arbeitet mit zwei virtuellen Konsolen, die alle die gleiche Hardware (Bildschirm und Tastatur) benutzten. Sie werden mit der Taste [Num] umgeschaltet. Leuchtet die Lampe ist die Konsole **con1** aktiv. Standardmäßig wird die Konsole **con0** geöffnet und eine Shell mit StandardIn- und StandardOut-Datei **con0** gestartet. Die beiden Konsolen benutzen getrennte Tastaturpuffer. Welche Konsole gerade sichtbar ist, wird in der Variablen **active_con** vermerkt. Auch der Zugriff auf das Betriebsmittel Konsole wird über den Semaphor **con_in_use** geregelt. Wird die sichtbare Konsole gewechselt werden alle Register (Cursor-Posititon, Hintergrundfarbe,...) der Konsole gerettet. Für Fehlerausgaben steht die Fehlerkonsole **err** zur Verfügung die Fehlermeldungen auf der sichtbarer Konsole ausgibt, und von dieser Tasteneingaben erwartet. Die Gerätenamen sind `con0`, `con1`, `err`.

VT52-Emulator:

Der Konsolentreiber versteht die wichtigsten VT52-Sequenzen. Diese sind:

Sequenz	Bedeutung
ESC A	Cursor hoch
ESC B	Cursor runter
ESC C	Cursor rechts
ESC D	Cursor links
ESC E	Clear Home
ESC H	Cursor Home
ESC J	Bildschirm ab Cursor löschen
ESC K	Zeile ab Cursor löschen
ESC j	Cursorposition speichern
ESC k	Cursorposition restaurieren
ESC L	Zeile einfügen
ESC b (x)	Schriftfarbe wählen, Farbwert als Zahlenwert zwischen ASCII-Code $30 und $37 (0-7)
ESC c (x)	Hintergrundfarbe wählen
ESC Y (x,y)	Cursorposition setzen, 32 muß zur X- und Y-Position addiert werden.

Tabelle 75: Sequnzen des VT52-Emulators

Dem VT52-Emulator ist noch eine besondere Sequenz ESC G bekannt. Mit dieser Sequenz können Graphikdaten angezeigt werden. Diese Graphikdaten enthalten Informationen für die Größe eines Zeichens. Der Cursor wird nach dieser Sequenz auch weitergesetzt. Der Sequenz müssen 16 Worte Daten folgenden. Aus einem Datenwort wird nur das niederwertige Byte ausgewertet. Dieses Byte enthält die Informationen für eine 8-Pixel-Zeile. Ist ein Bit gesetzt wird auch das Pixel in dieser Zeile gesetzt.

Durch diese Sequenz können Graphik und Daten beliebig gemischt werden. Das Dienstprogramm TYPE zeigt solche Graphiken an.

Tastatur:

Die Tastatur benutzt den Interruptlevel 1 um Zeichen zu übergeben. Die Tastatur wird im 11 Bit At-Modus Code-Set 3 betrieben. Betätigen der [Rollen]-Taste unterbindet das Scrollen auf der sichtbaren Konsole. Die Tasten [Druck] und [Pause] sind bisher ohne Funktion. Der Zahlenblock wird ausschließlich als Zahlenblock benutzt. Die Taste [Num] wird als Umschalttaste für die Konsolen **con0** und **con1** benutzt. Ein Tastaturpuffer ist als 16-Wort-Ringpuffer organisiert.

Wird ein Zeichen von der Tastatur gelesen wird der Tastencode im höherwertigen Byte und der ASCII-Code im niederwertigen Byte übermittelt. Die Funktions-, Cursor-,... Tasten erzeugen nur einen Tastencode, das niederwertige Byte ist Null. Die Tastenkombination [CTRL-z] erzeugt einen "`End-Of-File erreicht!`"-Fehler von der Tastatur. Ist kein Zeichen im Tastaturpuffer wird der Fehler "`Es liegt kein Zeichen an!`" übermittelt. Der Zustand der Sondertasten wird in der Variablen **umschaltflags** vermerkt.

Jobs:

Der Konsolentreiber stellt einen weiteren Job zur Verfügung. Und zwar kann ein Zeichen aus dem Tastaturpuffer gelesen werden ohne es aus dem Tastaturpuffer zu entfernen und den Lesezeiger zu verändern.

Job 4: Zeichen aus Tastaturpuffer lesen ohne Tastaturpuffer zu verändern.

Variable	Adresse	Inhalt
umschaltflags	32	Zustand der Umschalttasten
flags1	33	Gesicherte Flags der Tastatur-Interruptroutine
t_int_ar	34	Arbeitsregister der Tastatur- Interruptroutine
t_int_ar2	63	Arbeitsregister der Tastatur- Interruptroutine
con_in_use	35	Semaphor
t_read_ptr	36	Lesezeiger im Tastaturpuffer der aktiven Konsole
t_write_ptr	37	Schreibzeiger im Tastaturpuffer der aktiven Konsole
t_buffer	38	Anfang des Tastaturpuffers d. a. K.
t_buffer_end	39	Adresse des ersten Wortes, das nicht mehr zum Tastaturpuffer der a. Konsole gehört
cursor_adress	40	Adresse des Cursor der aktiven Konsole
cursor_x	41	X-Position der Cursors d. a. K
cursor_y	42	Y-Position des Cursor d. a. K.
vt52_seq	43	Variable die anzeigt, ob und welche VT52-Sequenz gerade bearbeitet wird
save_x	44	Gespeicherte X-Position bei VT52- Sequenz ESC j d. a. K.
save_y	45	Gespeicherte X-Position bei VT52- Sequenz ESC j d. a. K.
save_adr	46	Gespeicherte Cursor-Adresse bei VT52- Sequenz ESC j d. a. K.
fcolor	47	Schriftfarbe d. a. K.
bcolor	48	Hintergrundfarbe d.a. K.
Die gleichen Variablen für die nicht aktive Konsole:		
t_read_ptr2	49	...
t_write_ptr2	50	...
t_buffer2	51	...
t_buffer_end2	52	...
cursor_adress2	53	...
cursor_x2	54	...
cursor_y2	55	...
vt52_seq2	56	...
save_x2	57	...
save_y2	58	...
save_adr2	59	...
fcolor2	60	...
bcolor2	61	...
active_con	62	Zeigt an welche Konsole aktiv ist (0/1), zu dieser gehören die Variablen ohne die Endung 2

Tabelle 76: Systemvariablen des Konsolentreibers

Die Variable **umschaltflags** ist dabei folgendermaßen organisiert:

Bit_{10}: print/sysrq
Bit_9: pause/break
Bit_8: alt-rechts
Bit_7: alt-links
Bit_6: ctrl-rechts
Bit_5: ctrl-links
Bit_4: shift-rechts
Bit_3: shift-links
Bit_2: caps-lock

Bit_1: num-lock (Bei XMOS: Konsole 0/1 select)
Bit_0: scroll-lock

Bit_{3-0} werden als Lampenkombination der Tastatur übergeben.

Variable	I/O-Adresse	Inhalt
keyboard_data	$7ff6	Datenregister der Tastatur
keyboard_ctrl	$7ff7	Kontrollregister der Tastatur
xgraph	$7fe8	Basisadresse der Graphikkarte

Tabelle 77: Die I/O-Adresse der Konsole

7.8.7. Seriell-Treiber

Arbeitsweise und Fehlercodes

Bei diesen Seriell-Treiber handelt es sich um einen Text-Treiber mit Software-Handshaking. CTRL-Q teilt die Empfangsbereitschaft mit, CTRL-S teilt mit, daß der Empfänger nicht bereit ist. CTRL-Z erzeugt einen "`End-Of-File erreicht!`"-Fehler.

In niederwertigen Byte wird das empfangene Byte abgelegt, im höherwertigen Byte wird ein Fehlercode eingetragen. Drei Fehlertypen sind möglich:

Bit_0: Paritätsfehler
Bit_1: Stopbitfehler
Bit_2: Overrunfehler

Bitnummern beziehen sich auf das höherwertige Byte.

Die Empfangsroutine des Serielltreibers arbeitet im Interruptlevel 2. Sie benutzt einen 16-Wort-Ringpuffer. Im Empfangspuffer wird drei Worte vorausgeschaut, wenn das dritte Wort nicht mehr frei ist wird ein CTRL-S gesendet. Der Job 1 (Ein Zeichen Lesen) sendet ein CTRL-Q, wenn nach dem gelesenen Wort der Empfangspuffer leer ist. Der Gerätename ist `ser`.

Variable	Adresse	Inhalt
flags2	19	gerettete Flags der Seriell-Interruptroutine
s_write	20	Schreibzeiger im Empfangspuffer
s_read	21	Lesezeiger im Empfangspuffer
s_int_ar	22	Eingelesenes Zeichen, wird niederwertiges Byte im Wort des Empfansgpuffers
s_error	23	Fehlercode beim Empfang eines Zeichens, wird höherwertiges Byte im Wort des Empfangsregsiters
s_buffer	24	Zeiger auf den Pufferanfang des Empfangspuffers
s_buffer_end	25	Adresse des ersten Wortes, das nicht mehr zum Empfangspuffer gehört
cts	78	Clear-To-Send, hat den Wert Null, wenn der Empfänger empfangsbereit ist
rts	79	Request-To-Send, hat den Wert Null, falls der Seriell-Treiber empfangsbereit ist

Tabelle 78: Systemvariablen des Seriell-Treibers

Die Variablen **cts** und **rts** regeln das Software-Handshaking.

Variable	I/O-Adresse	Inhalt
empfangsregister	$7ffd	Empfangsregister der Seriellen Schnittstelle
senderegister	$7fff	Senderegister der Seriellen Schnittstelle
controllregister	$7ffe	Kontrollregister der Seriellen Schnittstelle
statusregister	$7ffc	Statusregister der Seriellen Schnittstelle

Tabelle 79: Die I/O-Adressen der seriellen Schnittstelle

7.8.8. Parallelport-Treiber

Der Parallelport-Treiber ermöglicht nur Werte an die parallele Schnittstelle auszugeben, nicht jedoch auch Werte einzulesen. Wird der Job 1 (Ein Zeichen Lesen) aufgerufen wird der Status der parallelen Schnittstelle übermittelt. Der Treibername ist `par`. Es werden keine speziellen Systemvariablen benötigt.

Variable	I/O-Adresse	Inhalt
parallel_data	$7ff8	Datenregister der parallelen Schnittstelle
parallel_status	$7ff9	Statusregister der parallelen Schnittstelle

Tabelle 80: I/O-Adressen des Parallelports

7.8.9. ROM-Disk-Treiber

Der ROM-Disk-Treiber ist ein vereinfachter SCSI-Treiber, der keinen eigenen Cache verwaltet und keine Systemvariablen benötigt. Der Job 2 und 4 wird mit einem "`Medium in SCSI-Laufwerk ist schreibgeschützt!`"-Fehler beantwortet. Als Kapazität wird die Größe des ROM-Disk-Datenbereiches übermittelt, derzeit $3A00 Worte. Der verbleibende ROM-Speicher, der nicht vom Betriebssystem genutzt wird, wird zur Steigerung der Geschwindigkeit mit den Dienstprogrammen belegt, die sich jederzeit auf einen anderen Datenträger auslagern lassen. Werden Dienstprogramme abgeändert und auf einem Datenträger x abgelegt, muß der Pfad geändert werden. Der standardmäßige Pfad ist derzeit **path=/rom/;/b/** . Der neue Pfad sollte dann **path=/x/;/rom/;/b/** sein.

7.9. Interruptbehandlung

Im System sind drei der fünf Interruptebenen belegt.

Interruptlevel	Verwendung
5	Timer
4	frei
3	frei
2	Serielle Schnittstelle
1	Tastatur

Tabelle 81: Interruptbelegung des Betriebssystems

Der Timer muß um eine annähernd korrekte Zeitbasis zu liefern und den Taskwechsel zu steuern den höchsten Interruptlevel haben. Tastatureingaben können etwas verzögert werden. Serieller Transfer mit höheren Baudraten müssen jedoch schneller beantwortet werden. Deswegen hat die serielle Schnittstelle ein höhere Priorität als die Tastatur. Jede Interruptroutine benutzt ihre eigenen Arbeitsregister und rettet die Flags.

7.10. Environmentvariablen

Das System arbeitet mit drei Environmentvariablen:

cdw: Current Working Directory
path: Pfad
cmd: Commandline

Die CWD muß mit einen / beginnen und aufhören. Die Standard-CWD ist

```
cwd=/b/
```

Der Pfad muß mit eine / beginnen und aufhören, das Trennzeichen ist ein ;. Der Standard-Pfad ist

```
path=/rom/;/b/
```

In der Variablen cmd werden Parameter an ein Programm übergeben, das Trennzeichen ist ein Space. Beispiel:

```
cmd=parameter1 parameter2 parameter3
```

Alle Environmentvariablen enden mit einem Null-Wort.

7.11. Initialisierungsphase des Betriebsystems

Die Initialisierung läuft im User-Mode ab. In der Initialisierungsphase wird zu erst der Timer deaktiviert und die Speicherverwaltung initialisiert. Dies geschieht durch setzen der Variablen **bfs** und **efs** und löschen des Semaphors **h_in_use**. Es wird ein freier Speicherblock, der den ganzen Speicher nach den Systemvariablen umfaßt, angelegt

Danach wird die Dateiverwaltung durch löschen der Variablen **fdblist** und der Semaphore **fdblist_in_use** und **allocate_in_use** initialisiert.

Jetzt wird "per Hand" eine Task erzeugt und die Variable **akt_task** gesetzt. Diese Task trägt den Namen XMOS 1.0 und wird nach der Intialisierung die Shell. Stack und Environmentbereich werden schon über den Systembefehl MALLOC angefordert. Danach ist die Taskverwaltung einsatzbereit.

Die Treiberverwaltung wird durch Anforderung eines Speicherblocks für die Treiberliste und setzen der Variable **driverlist** durchgeführt. Die Treiberliste wird mit Null initialisiert.

Dann werden die verschiedenen Initialisierungsroutinen der einzelnen Treiber aufgerufen, die mit MALLOC ihren benötigten Speicher allokieren und die Treiber mit NEWDRIVER anmelden. Reihenfolge des Aufrufes der Initialisierungroutinen:

1. Timer
2. Konsolen-Treiber
3. SCSI-Treiber
4. ROM-Disk-Treiber
5. Parallelport-Treiber
6. Seriell-Treiber

In den Environmentbereich der Task XMOS 1.0 werden die Environmentvariablen

```
cwd=/b/             und
path=/rom/;/b/
```

eingetragen. Die Konsole `con0` wird zum Lesen und Schreiben geöffnet und als StandardIn- und StandardOut-Datei in den TDB eingetragen. Eine Einschaltmeldung wird an `con0` ausgegeben und dann eine Shell gestartet.

7.12. Ausführbares Programmformat

Ein ausführbares Programm hat einen aus zwei Worten bestehenden Programmkopf, einem Programmrumpf und einer Reloziertabelle. Das erste Wort des Programmkopfes gibt den gesamten Speicherbedarf, sowohl initialisierter als auch nicht initialisierter Bereich, des Programms in Worten an. Das zweite Wort gibt die Länge des initialisierten Bereiches in Worten an. Dies ist die Anzahl an Worten die aus der Datei gelesen wird und als Programmrumpf interpretiert wird. Danach folgt eine Tabelle mit Relozierinformationen. Gestartet wird ein Programm immer an der ersten Adresse des Programms. Adressen werden nach dem Laden des Programms an die Startadresse angepaßt.

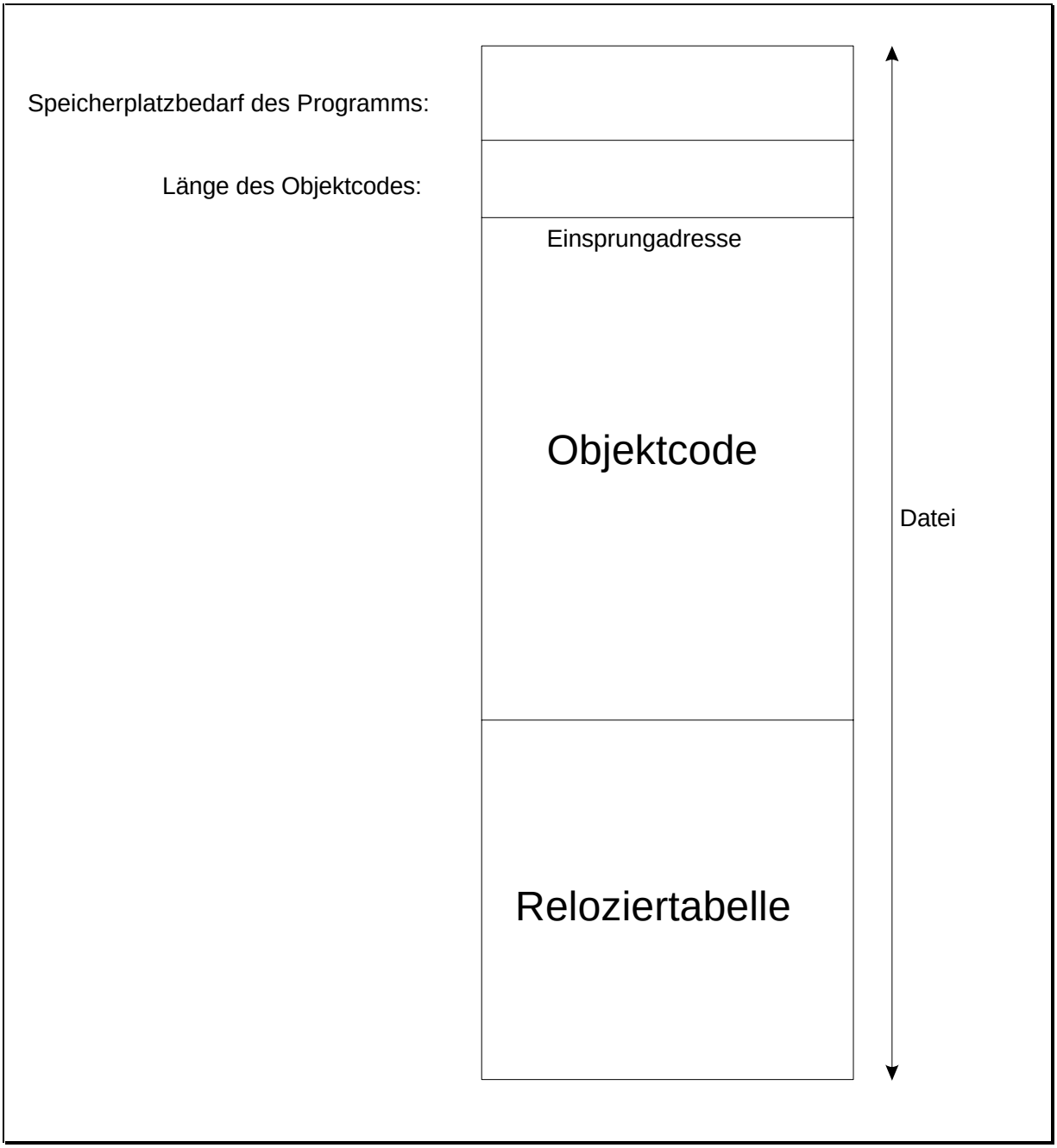


Bild 22: Format eines ausführbaren Programms

8. Betriebssystemfunktionen

8.1. Allgemeine Befehlsbehandlung

8.1.1. Befehlsformat

Um alle Systembefehle korrekt nutzen zu können sollte die Library LIB.ASM in das eigene Programm eingebunden werden. Die Library LIB.ASM enthält ein Makro, das den Zugang zu Systembefehlen einfach gestaltet. Jeder Systembefehl hat eine Befehlsnummer und wird über das Makro SYS #Befehlsnummer aufgerufen. Die Parameter, die der Systembefehl benötigt müssen über den Stack übergeben werden. Das Ergebnis eines Systembefehls wird in den Registern d0 und d1, die auch über die Library LIB.ASM zugänglich sind, geliefert. Kann ein Systembefehl nicht korrekt ausgeführt werden, so wird in dem Register d0 ein Fehlercode übergeben, und das N-Flag ist gesetzt. Bei korrekter Bearbeitung eines Systembefehls können die Flags Z, C oder V gesetzt sein, aber nicht das N-Flag. So kann durch einen bedingten Sprung jeweils nach einem Systembefehl eine Fehlerbehandlung erfolgen. Nach Ausführung eines Systembefehls hat der Stackpointer den gleichen Wert wie vor Aufruf des Systembefehls. Es ist also Sache des User-Programms den Stackpointer danach wieder so zu restaurieren, daß der Satckpointer wieder den gleichen Wert hat, wie vor einem Aufruf.

```
Beispiel für einen Systemaufruf:
push   #parameter1
push   #parameter2
sys    #Befehlsnummmer
add    #2,sp
jmi    fehler
```

Desweiteren ist zu beachten, daß Befehle ohne Angabe einer Befehlsbedingung und Flaganweisung als unbedingte Befehle ohne Flagveränderung interpretiert werden.

```
add    #a,b,c
```

wird interpretiert als

```
add    tr hf #a,b,c
```

8.1.2. Die Library LIB.ASM

Die Library LIB.ASM ist die Verbindung zwischen User-Programmen und dem Betriebssystem. Sie enthält außer einer Reihe sinnvoller Makros auch einige wichtige Vereinbarungen, ohne die ein Arbeiten mit diesem Betriebssystem nicht möglich ist.

Durch Makros zusätzlich zur Verfügung gestellte Befehle (bed.: kann nur bedingt ausgeführt werden, unbed: kann nur unbedingt ausgeführt werden, hf: verändert die Flags nicht, sf: verändert die Flags, keine Angabe: alles ist möglich):

```
move          x,z   ;z:=x
not           x,z   ;z:=not(x)
cmp           x,y   ;Verfahren: y-x, unbed., sf,
```

```
inc          x     ;x:=x+1
dec          x     :x:=x-1
clr          x     ;x:=0
irqoff             ;Sperrt alle Interrupts, unbed., hf
irqon              ;Gibt Interrupts wieder frei, unbed., hf
in           x,z   ;z:=I/O-Bereich(x), unbed.
out          x,z   ;I/O-Bereich(z):=x, unbed.
push         x     ;sp:=sp-1, (sp):=x, hf
pop          x     ;x:=(sp), sp:=sp+1, hf
pushf              ;sp:=sp-1, (sp):=Statusregister, hf
popf               ;Statusregister3-0:=(sp), sp:=sp+1, sf
pushz              ;sp:=sp-1, (sp):=0, hf
pushall            ;push d2, push d3, push d4, push d5,
                   ;push d6 ,push d7, hf
popall             ;pop d7, pop d6, pop d5, pop d4, pop d3, pop d2, hf
jsr          x     ;sp:=sp-1, (sp):=Rücksprungadresse, pc:=#x
                   ;sp:=sp+1, hf
rts                ;pc:=(sp), hf
driver       x     ;sp:=sp-1, (sp):=Rücksprungadresse, pc:=x
                   ;sp:=sp+1, hf
sys          x     ;Ruft Systembefehl x auf, unbed. hf
jmp          x     ;pc:=#x, unbed. hf
jeq          x     ;pc:=#x, bed., hf
jne          x     ;pc:=#x, bed., hf
jpl          x     ;pc:=#x, bed., hf
jmi          x     ;pc:=#x, bed., hf
jpe          x     ;pc:=#x, bed., hf
jcs          x     ;pc:=#x, bed., hf
jcc          x     ;pc:=#x, bed., hf
```

Einige Systemvariablen sind durch folgende Vereinbarungen Anwendungsprogrammen zugänglich:

```
d0:          Ergebnisregister eines Systembefehls
d1:          Ergebnisregister eines Systembefehls
d2-d7:       Arbeitsregister der Systembefehle
sp:          Stackpointer
akt_task:    Tasknummer der aktuellen Task
timerl:      Timer, Low-Wort
timerh:      Timer, High-Wort
```

Das Listing der Library LIB.ASM befindet sich im Anhang.

8.1.3. Notation

Die im folgenden beschrieben Systembefehle werden in einer nicht dem Aufruf entsprechenden Form beschrieben. Ein Befehlsaufruf der Gestalt

```
push Para1
push Para2
push Para3
sys  #Befehlsnummer
add  #3,sp
```

wird

```
Befehlsname(Para1, Para2, Para3):Rückgabewert1, Rückgabewert2
```

beschrieben.

Der Rückgabewert1 wird im Register d0, der Rückgabewert2 im Register d1 übergeben. Gibt es nur einen Rückgabewert, so wird dieser immer im Register d0 geliefert. Das Register d0 enthält im nicht erfolgreichen Fall einen Fehlercode. Als Fehlercodes werden nur Betriebssystemfehler genannt. Tritt bei einer Geräteoperation ein treiberspezifischer Fehler (z. B.: SCSI-Fehler) auf, wird dieser "durchgereicht" und auch im Register d0 übergeben.

8.2. Speicheroperationen

8.2.1. MALLOC

Der Systembefehl MALLOC stellt einen angeforderten Speicherblock, falls noch verfügbar, bereit. Es wird empfohlen nicht mehrere kleine Blöcke zu allokieren, sondern einen Größeren, da sonst viel Speicherplatz für Verwaltungsinformation der Speicherverwaltung verschwendet wird.

```
MALLOC(Gewünschte Blockgröße): Anfangsadresse des angeforderten
Blocks
```

Befehlsnummer: 0
Fehlercodes: 103 Zuwenig Speicherplatz!
Ein freier Block dieser Größe ist nicht mehr verfügbar.

8.2.2. MSHRINK

Der Systembefehl MSHRINK läßt einen Speicherblock auf eine angegebene Größe zusammen schrumpfen.

```
MSHRINK(Neue Blockgröße, Blockanfangsadresse)
```

Befehlsnummer: 1
Fehlercodes: 101 Ungültiges Handle!
Die Blockanfangsadressse war ungültig.

8.2.3. MFREE

Der Systembefehl MFREE gibt ein Speicherblock wieder frei.

```
MFREE(Blockanfangsadresse)
```

Befehlsnummer: 2
Fehlercodes: 101 Ungültiges Handle!
Die Blockanfangsadressse war ungültig.

8.2.4. MEMINFO

Der Systembefehl MEMINFO ermittelt die Größe des insgesamt noch freien Arbeitsspeichers und des größten freien Speicherblocks.

```
MEMINFO:Größe des freien Speichers insgesamt, Größe des längsten
freien Speicherblocks
```

Befehlsnummer: 3
Fehlercodes: -

8.3. Taskbehandlung

8.3.1. STARTTASK

STARTTASK meldet dem Betriebssystem eine neue Task an. Nach diesem Aufruf befindet sich die Task in der Taskliste und wird bearbeitet. Dem Systembefehl muß die gewünschte Größe des Environmentbereichs, die Startadresse der Task, das heißt die Anfangsadresse des Speicherblocks, der zur Task gehört übergeben werden. In diesem Environmentbereich wird die Currrent Working Directory, der Path und Parameter der Command Line abgelegt. Parameter der Command Line sind die Parameter die einem Programm durch die Shell übergeben werden. Der Environmentbereich der Task, der die Betriebssystemroutine STARTTASK aufruft, wird der zu startenden Task vererbt. Es empfiehlt sich daher den Environmentbereich der neuen Task genauso groß zu wählen, wie den der startenden Task. Reicht der Platz des Environmentbereiches der neuen Task nicht aus, ist er also kleiner als der der startenden Task, wird der Environmentbereich der neuen Task soweit wie möglich gefüllt und durch eine Null abgeschloßen. Dies kann dann mitten in einer Environmentvariable geschehen. Die Startadresse einer Task muß die erste Adresse eines Speicherblockes sein, der der Task gehört. Ab dieser Adresse wird die Task gestartet. Desweiteren muß ein Zeiger auf den Tasknamen, der entweder acht Zeichen lang sein oder mit einer Null abschließen muß, mitgeteilt werden. Der Zeiger auf den Tasknamen muß nur während des Aufrufes des Systembefehls STARTTASK gültig sein, der Taskname wird in den Taks-Descriptor-Block kopiert. Der Taskname darf ein Zeichen nur im Low-Byte enthalten, die High-Bytes müssen der Wert Null haben. Die Angabe der Handles der Dateien StandardIn und StandardOut ist auch gefordert. Die Betriebssystemroutine STARTASK stellt den Stackpointer auf das Ende des Speicherblocks der zur Task gehört ein und legt dort die Tasknummer und die Adresse der KILLTASK-Routine ab. Bei einem `rts` auf höchster Programmebene wird somit die Betriebssystemroutine KILLTASK aufgerufen und die Task entfernt. Eine Task muß deshalb noch Stackbereich besitzen, in dem zumindest die beiden genannten Werte abgelegt werden können. STARTTASK lädt und reloziert nicht, dies ist Aufgabe der Betriebssystemroutine PROGRAM LOAD.

```
 STARTTASK(StandardOut-Handle, StandardIn-Handle, Zeiger auf
Tasknamen, Startadresse, Größe des Environmentbereiches)
```

Befehlsnummer: 4

Fehlercodes:	103	Zuwenig Speicherplatz! Der benötigte Speicher für den Environmentbereich konnte nicht angefordert werden. Keine Nummer, nur N-Flag, falls die Startadresse der Task ungültig war.

8.3.2. SWITCHTASK

Durch den Systembefehl SWITCHTASK wird die aktuelle Task angehalten und die nächste Task in der Taskliste aktiviert. Existiert nur eine Task im System, so ist die aktuelle Task die nächste.

```
SWITCHTASK
```

Befehlsnummer: 5
Fehlercodes: -

8.3.3. KILLTASK

Der Systembefehl KILLTASK entfernt eine Task aus dem System. Vorher werden automatisch alle Systemresourcen, die von dieser Task belegt wurden, wieder freigegeben und alle geöffneten Dateien dieser Task ordnungsgemäß geschlossen. Dieser Aufruf wird implizit ausgeführt, wenn eine Task sich mit `rts` auf höchster Programmebene beendet. Wird die letzte Task im System entfernt erfolgt ein RESET.

```
KILLTASK(Tasknmummer)
```

Befehlsnummer: 6
Fehlercodes: 101 Ungültiges Handle!
Tasknummer war ungültig.

8.4. Dateioperationen

8.4.1. OPEN

Der Systembefehl OPEN öffnet eine Datei oder ein Gerät. Es wird überprüft, ob es durch mehrfach geöffnete Dateien zu einem Zugriffskonflickt kommen kann. Das ist dann der Fall, wenn eine Datei schon zum Lesen geöffnet ist und danach noch zum Schreiben geöffnet wird. Oder anders herum, wenn eine Datei schon zum Schreiben geöffnet ist und danach zum Lesen geöffnet werden soll. Unproblematisch ist eine Datei einmal zum Schreiben oder mehrmals zum Lesen zu öffnen. Geräte können nur einfach geöffnet sein.

Es stehen vier Öffnungsmodi zur Verfügung:

1: Datei zum Lesen öffnen
2: Datei zum Lesen und Schreiben öffnen
3: Datei zum Neuschreiben öffnen
4: Datei zum Anhängen öffnen

Bei den Öffnungsmodi 1 und 2 muß die angegebene Datei bereits existieren. Beim Öffnungsmodus 3 darf die angegebene Datei noch nicht existieren. Beim Öffnungsmodus 4 ist es irrelevant, ob die Datei existiert oder nicht. Falls die Datei nicht existiert wird eine neue Datei erzeugt. Beim Öffnungsmodus 2 ist zu beachten, das für Dateioperationen nur ein Dateizeiger existiert. Unter bestimmten Vorraussetzungen muß der Zeiger beim Wechsel der Zugriffsart neu positioniert werden. Dem Systembefehl OPEN kann ein kompletter Dateipfad übergeben werden oder nur ein Dateiname. Beim letzteren wird in der CWD nach der angegebenen Datei gesucht.

Bei Übergabe eines kompletten Dateipfades muß der String folgende Form haben:

```
/Verzeichnis1/Verzeichnis2/Verzeichnis3/Dateiname.
```

Soll aus der CWD eine Datei geöffnet werden muß der String folgende Gestalt haben:

```
Dateiname
```

Das Zeichen "..", eine Verzeichnisebene höher wechseln, wird von dem Systembefehl OPEN nicht erkannt. Der Dateiname muß mit einer Null enden. Zulässige Zeichen für den Dateinamen sind die ASCII-Codes $21 bis $7f.

```
OPEN(Öffnungsmodus, Zeiger auf Dateinamen):Handle
```

Befehlsnummer: 12

Fehlercodes:

103	Zuwenig Speicherplatz! Speicherplatz für Verwaltungsinformationen konnte nicht angefordert werden.
104	Dateiname hat falsches Format!
106	Device existiert nicht! Der angegebene Dateinamen bezog sich auf ein Gerät, daß dem System nicht bekannt ist.
108	Maximale Dateilänge (32MB) erreicht! Beim Öffnen zum Neuschreiben oder Anhängen ließ sich das Verzeichnis nicht mehr erweitern um den neuen Verzeichniseintrag zu erstellen.
109	Datei existiert nicht! Eine nicht existente Datei sollte zum Lesen geöffnet werden.
110	Datei existiert schon! Eine bereits existente Datei sollte zum Neuschreiben geöffnet werden.
111	Parameter falsch! Als Öffnungsmodus war eine nicht zulässige Zahl angegeben.
112	Datenträger voll! Beim Öffnen zum Neuschreiben oder Anhängen war nicht genügend Speicherplatz auf dem Datenträger verfügbar um den neuen Verzeichniseintrag zu erstellen.
113	Directory existiert nicht! Im Dateipfad befand sich ein Verzeichnis, das nicht existierte.
114	Datei ist schreibgeschützt! Es wurde versucht eine schreibgeschützte Datei zum Schreiben zu öffnen.
115	Zugriffkonflikt auf mehrfach benutzte Datei!

8.4.2. CLOSE

CLOSE schließt eine geöffnete Datei. Falls es sich um eine Datei auf einem SCSI-Device handelt werden alle Schreib-Cache-Puffer, die für dieses Device existieren, zurückgeschrieben.

```
CLOSE(Handle)
```

Befehlsnummer: 15

Fehlercodes:

101	Ungültiges Handle!

8.4.3. READ

READ ließ Informationen aus einer Datei. Es besteht die Möglichkeit ein Wort oder einen Block beliebiger Länge aus einer Datei zu lesen. Wird nur ein Wort gelesen, so wird dieses Wort in d0 übergeben. Es muß also kein Pufferbereich zur Verfügung gestellt werden. Da der Systembefehl READ immer den gleichen Aufbau hat muß auch beim wortweisen Lesen ein Puffer mit angegeben werden, obwohl dieser nicht benutzt wird. Es würde auch reichen den Stackpointer zu decrementieren. Wählt man als Dateihandle eine Null, so wird von der Datei StandardIn gelesen.

Wortlesen:

```
READ(#1, Dummy, Handle): Gelesenes Wort, Anzahl der gelesenen Worte
(hier=1)
```

Blocklesen:

```
READ(Anzahl, Pufferadresse, Handle):Fehlercode, Anzahl der gelesenen
Worte
```

Befehlsnummer: 13

Fehlercodes:	101	Ungültiges Handle!
	102	Befehlszugriff nicht erlaubt! Es wurde versucht aus einer nur zum Schreiben geöffneten Datei zu lesen.
	105	End-Of-File erreicht! Es wurde beim Lesen das Dateiende erreicht. d1 enthält die Anzahl der gelesenen Wort bis EOF erreicht wurde.
	118	Es liegt kein Zeichen an! Es wurde von einem Gerät gelesen, das im Moment kein Zeichen bereitstellen kann (z. B.Kein Zeichen im Tastaturpuffer)

8.4.4. WRITE

WRITE schreibt Informationen in eine Datei. Es besteht die Möglichkeit ein Wort , einen String, oder einen Block beliebiger Länge in eine Datei zu schreiben. Wird nur ein Wort geschrieben, so muß dieses Wort direkt angegeben werden, ansonsten muß eine Pufferadresse angegeben werden. Schreibt man einen String in eine Datei so muß als Anzahl eine Null übergeben werden. Wählt man als Dateihandle eine Null, so wird in die Datei StandardOut geschrieben.

Wortschreiben:

```
WRITE(#1,Auszugebendes Wort, Handle)
```

Blockschreiben:

```
WRITE(Anzahl, Pufferadresse, Handle)
```

Stringschreiben:

```
WRITE(#0, Zeiger auf den String, Handle)
```

Befehlsnummer: 14

Fehlercodes: 101 Ungültiges Handle!
108 Maximale Dateilänge (32MB) erreicht!
112 Datenträger voll!
114 Datei ist schreibgeschützt!

8.4.5. GET FLAGS

Der Systembefehl GET FLAGS ermittelt die Dateiflags einer Datei. Bisher werden die Dateiflags WRITEPROTECT (Bit_2), EXECUTABLE (Bit_1) und DIRECTORY (Bit_0) benutzt. Die Datei muß dazu schon geöffnet sein.

```
GET FLAGS(Handle):Dateiflags
```

Befehlsnummer: 16
Fehlercodes: 101 Ungültiges Handle!
102 Befehlszugriff nicht erlaubt!
Es wurde versucht Dateiflags für ein Zeichendevice zu ermitteln.

8.4.6. SET FLAGS

SET FLAGS setzt die Dateiflags einer Datei. Dabei wird das angegebene Wort in der Datei als Flags eingetragen, es können also auch andere Bits als die Bits2-0 gesetzt sein. Es wird dabei auch keine Zulässigkeit geprüft, ob es sich zum Beispiel wirklich um eine Directory handelt. Die Datei muß für diese Operation schon geöffnet sein.

```
SET FLAGS(Dateiflags, Handle)
```

Befehlsnummer: 17
Fehlercodes: 101 Ungültiges Handle!
102 Befehlszugriff nicht erlaubt!
Es wurde versucht Dateiflags für ein Zeichendevice zu ermitteln.

8.4.7. GET POSITION

Der Systembefehl GET POSITION ermittelt den momentanen Wert des Dateizeigers. Der Wert des Dateizeigers wird in Worten, nicht in Bytes, übermittelt.

```
GET POSITION(Handle):Dateizeiger in Worten (High), Dateizeiger in
Worten (Low)
```

Befehlsnummer: 18
Fehlercodes: 101 Ungültiges Handle!
102 Befehlszugriff nicht erlaubt!
Es wurde versucht den Dateizeiger für ein Zeichendevice zu ermitteln.

8.4.8. SET POSITION

SET POSITION setzt den Dateizeiger einer Datei neu. Es ist dabei möglich sowohl relativ zur momentanen Position als auch absolut zum Dateianfang zu positionieren. Der übergebene Dateizeiger muß auch wortbezogen angegeben werden. Wird das Dateiende einer Datei überschritten, das heißt, auf eine Wort positioniert das nicht mehr zur Datei gehört, so wird der Dateizeiger auf das Wort nach dem letzen zur Datei gehörenden Wort positioniert, sozusagen ein Wort nach dem letzten gültigen Wort.

Relative Positionierung:

```
 SET POSITION(#0, Handle, Dateizeiger in Worten(High), Dateizeiger in
Worten (Low))
```

Absolute Positionierung:

```
 SET POSITION(#1, Handle, Dateizeiger in Worten(High), Dateizeiger in
Worten (Low))
```

Befehlsnummer: 19

Fehlercodes:
- 101 Ungültiges Handle!
- 102 Befehlszugriff nicht erlaubt!
  Es wurde versucht den Dateizeiger für ein Zeichendevice zu setzen.
- 105 End-Of-File erreicht!
  Beim Positionieren wurde das Ende der Datei erreicht.

8.4.9. RENAME

Der Systembefehl RENAME benennt eine Datei um. Die Datei muß für diese Operation schon geöffnet sein. Der neue Dateiname muß ungepackt, das heißt ein Zeichen pro Wort, vorliegen und auf eine Null enden. Es wird dabei nicht überprüft, ob der Dateiname schon existiert!

```
 RENAME(Handle, Zeiger auf den neuen Dateinamen)
```

Befehlsnummer: 20

Fehlercodes:
- 101 Ungültiges Handle!
- 102 Befehlszugriff nicht erlaubt!
  Es wurde versucht den Dateinamen für ein Zeichendevice zu ändern.
- 103 Zuwenig Speicherplatz!
  Der Befehl RENAME allokiert einen 8 Wort großen Speicherplatz für diese Operation. Falls dieser Speicherplatz nicht allokiert werden kann wird der Systembefehl RENAME mit diesem Fehler abgebrochen.

8.4.10. DELETE

DELETE löscht eine Datei. Da in diesem System Verzeichnisse wie normale Dateien behandelt werden, können mit diesem Befehl auch Verzeichnisse gelöscht werden. Dies ist aber nur sinnvoll, wenn das Verzeichnis geleert ist, ansonsten werden die Sektoren, die Dateien belegen, nicht freigegeben. Es werden nämlich nur die Sektoren des Verzeichnisses gelöscht. Die Datei muß für diese Operation bereits geöffnet sein. Die Datei wird nach dem Enfernen implizit durch CLOSE geschlossen, obwohl die Datei nicht mehr existiert. Dies ist nötig, da der FDB aber noch existiert und wieder freigegeben werden muß. Das Handle ist also nach dem Aufruf des Betriebssystembefehls nicht mehr gültig.

```
 DELETE(Handle)
```

Befehlsnummer: 21

Fehlercodes:
- 101 Ungültiges Handle!
- 102 Befehlszugriff nicht erlaubt!
  Es wurde versucht ein Zeichendevice zu löschen.

8.4.11. MAKE DIRECTORY

Der Systembefehl MAKE DIRECTORY erzeugt ein neues Verzeichnis. Dieser Befehl öffnet die angegebene Datei, setzt das DIRECTORY-Flag und schließt die Datei wieder.

```
MAKE DIRECTORY(Zeiger auf den Verzeichnisnamen)
```

Befehlsnummer: 22
Fehlercodes: Alle Fehlercodes des Systemaufrufes OPEN.

8.4.12. REMOVE DIRECTORY

REMOVE DIRECTORY entfernt ein Verzeichnis. Das Verzeichnis muß vollständig geleert sein.

```
REMOVE DIRECTORY(Zeiger auf den Verzeichnisnamen)
```

Befehlsnummer: 23
Fehlercodes:
- 113 Directory existiert nicht!
  Es wurde versucht mit REMOVE DIRECTORY eine normale Datei zu löschen.
- 116 Directory ist nicht geleert!
  Das Verzeichnis enthält noch Dateien und ist nicht vollständig geleert.

Alle Fehlercodes der Systembefehle OPEN, READ, DELETE, SET POSITION und CLOSE.

8.4.13. CHANGE DIRECTORY

Der Systembefehl CHANGE DIRECTORY wechselt das aktuelle Verzeichnis oder Gerät. Durch diesen Befehl wird das CWD neu gesetzt. Es besteht die Möglichkeit eine Verzeichnisebene tiefer zu wechseln durch Angabe des Verzeichnisnamen, eine Verzeichnisebene höher zu wechseln durch Angabe von **..** als Pfadnamen, oder ein ganz neuen Pfad zu setzen durch Beginn des Pfadnamens mit /. Der letzte Verzeichnisname muß ohne ein anschließendes / angegeben werden. Der Pfadname muß mit einer Null enden.

```
CHANGE DIRECTORY(Zeiger auf den Pfadnamen)
```

Befehlsnummer: 24
Fehlercodes:
- 113 Directory existiert nicht!
  Bei dem angegeben Pfad ist eine Datei dabei, die kein Verzeichnis ist.

Alle Fehlercodes der Systembefehle OPEN, MALLOC, SET ENVIRONMENT.

8.4.14. PROGRAM LOAD

PROGRAM LOAD lädt ein Programm in den Arbeitsspeicher und reloziert es. Es muß sich bei der angegebenen Datei um eine ausführbare Datei, das heißt, daß EXECUTABLE-Flag muß gesetzt sein, handeln. Öffnen und Schließen der Datei und Speicherallokation erfolgt durch diesen Systembefehl. Als Rückgabewert erhält man die Startadresse des Programms im Arbeitsspeicher. Es wird kein Stackpointer eingestellt.

```
PROGRAM LOAD(Zeiger auf den Dateinamen):Startadresse der
Programms
```

Befehlsnummer: 25
Fehlercodes: 117 Datei ist nicht ausführbar!
Das EXECUTABLE-Flag der Datei ist nicht gesetzt.
Alle Fehlercodes der Systembefehle OPEN, READ und MALLOC.

8.4.15. PREPARE SEARCH

Durch den Systembefehl PREPARE SEARCH wird die Suche nach einem String in einem Verzeichnis vorbereitet. Der String wird in ein Format, das der Systembefehl SEARCH NEXT verarbeiten kann, konvertiert und überprüft ob er nur zulässige Zeichen enthält. Der Dateizeiger der Datei wird auf den Dateianfang gesetzt. Die Datei muß für diese Operation schon geöffnet sein und das Dateihandle übergeben werden, um auf die Datei zugreifen zu können. Der Suchstring muß ungepackt vorliegen und auf eine Null terminieren. Dem Systembefehl PREPARE SEARCH muß außerdem ein acht Worte großer Puffer übergeben werden, in dem sich nach diesem Aufruf der konvertierte String befindet, dieser Puffer muß dann dem Systembefehl SEARCH NEXT übergeben werden. Wildcards im Dateinamen sind erlaubt.

```
 PREPARE SEARCH(Handle des Verzeichnisses, Zeiger auf 8-Wort-Puffer,
Zeiger auf den Suchstring)
```

Befehlsnummer: 26
Fehlercodes: 101 Ungültiges Handle!
102 Befehlszugriff nicht erlaubt!
Es wurde versucht auf einem Zeichendevice zu operieren.
104 Dateiname hat falsches Format!
Der Suchstring enthält nicht zulässige Zeichen.

8.4.16. SCAN

Der Systembefehl SCAN ist im Grunde nur für Systembefehle gedacht, mußte aber für die Dienstprogramme von außerhalb des Betriebssystem zugänglich sein. Es handelt sich hierbei um eine abgewandelte Form des Systembegfehls PREPARE SEARCH. Es muß kein Handle übergeben werden und es wird auch kein Dateizeiger manipuliert. Der Suchstring darf keine Wildcards enthalten.

```
 SCAN(Zeiger auf 8-Wort-Puffer, Zeiger auf den Suchstring)
```

Befehlnummer: 28
Fehlercodes: 104 Dateiname hat falsches Format!
Der Suchtring enthält nicht zulässige Zeichen.

8.4.17. SEARCH NEXT

SEARCH NEXT operiert mit einem durch PREPARE SEARCH erstellten String auf einem Verzeichnis. Dem Systembefehl SEARCH NEXT muß der in PREPARE SEARCH bearbeitete 8-Wort-Puffer übergeben werden. Ein Verzeichnis wird auf einen zu diesem String passenden Eintrag hin durchsucht. Ist die Suche erfolgreich wird der Dateizeiger auf den Anfang des Verzeichniseintrages positioniert. War die Suche nicht erfolgreich wird ein End-Of-File übermittelt. SEARCH NEXT bearbeitet das Verzeichnis ab der aktuellen Dateizeigerposition. Die Datei muß für diese Operation schon geöffnet sein.

```
SEARCH NEXT(Zeiger auf 8-Wort-Puffer, Handle)
```

Befehlsnummer: 27

Fehlercodes: 101 Ungültiges Handle!
102 Befehlszugriff nicht erlaubt!
Es wurde versucht in einem Zeichendevice zu suchen.
Alle Fehlercodes der Systembefehle READ und MALLOC.

8.4.18. GET LENGTH

GET LENGTH ermittelt die Länge einer Datei in Worten. Die Datei muß schon geöffnet sein.

```
GET LENGTH(Handle):Dateilänge in Worten(High), Dateilänge in
Worten (low)
```

Befehlsnummer: 32

Fehlercodes: 101 Ungültiges Handle!
102 Befehlszugriff nicht erlaubt!
Es wurde versucht die Länge eines Zeichendevices zu bestimmen.

8.5. Treiberoperationen

8.5.1. NEWDRIVER

Der Systembefehl NEWDRIVER meldet dem System einen neuen Gerätetreiber an. Nach diesem Aufruf ist der Treiber über die Dateioperationen zugänglich. Zeichengeräte- und Blockgerätetreiber müssen bestimmte Aufgaben erfüllen können und ein bestimmtes Format besitzen um korrekt benutzt werden zu können. Nähere Informationen stehen im Kapitel BETRIEBSSYSTEMREFERENZ 7.8 Treiberverwaltung.

```
NEWDRIVER(Treibaranfangsadresse)
```

Befehlsnummer: 7

Fehlercodes: Keine Nummer, nur N-Flag, falls kein Platz in der Treiberliste (mehr als 32 Treiber) mehr verfügbar ist.

8.5.2. SEARCHDRIVER

SEARCHDRIVER ermittelt die Einsprungadresse eines Gerätetreibers. Der Treibername muß entweder vier Zeichen lang sein, oder mit einer Null abschließen.

```
SEARCHDRIVER(Treibername):Treibereinsprungadresse
```

Befehlsnummer: 8

Fehlercodes: 106 Device existiert nicht!
Unter dem angegebenen Namen ist dem System kein Treiber bekannt.

8.6. Environmentbereich

8.6.1. SET ENVIRONMENT

SET ENVIRONMENT trägt eine Null-terminierte Environmentvariable der Form **STRING=STRING** in den Environmentbereich der aktuellen Task ein. Existiert bereits eine Environmentvariable gleichen Namens, so wird diese zuvor entfernt.

Vorsicht! Wird eine Environmentvariable der Form STRING= angegeben, so wird diese in den Environmentbereich eingetragen, obwohl sie keinen Folgestring hat! Beim Suchen nach diesem String wird dann die Adresse des Trennzeichens (Nullwort) ermittelt!

```
SET ENVIRONMENT(Zeiger auf die Environmentvariable)
```

Befehlsnummer: 9
Fehlercodes: 107 Environmentbereich zu klein!
Der Environementbereich ist nicht mehr groß genug um diese Variable hinzu zufügen.

8.6.2. SEARCH ENVIRONMENT

SEARCH ENVIRONMENT ermittelt die Position einer Environmentvariable im Environmentbereich der aktuellen Task. Übermittelt wird die Adresse des ersten Zeichens nach dem Gleichheitszeichen. Die gesuchte Variable muß Null-terminiert in der Form **STRING=** angegeben sein.

```
SEARCH ENVIRONMENT(Zeiger auf Environmentvariable):Position der
      gesuchten Variable im Environmentbereich
```

Befehlsnummer: 10
Fehlercodes: N-Flag, falls die Variable nicht gefunden werden konnte.

8.6.3. CLR ENVIRONMENT

Der Systembefehl CLR ENVIRONMENT entfernt eine Environmentvariable aus dem Environmentbereich der aktuellen Task. Die zu entfernende Variable muß in der Form **STRING=** angegeben sein und mit Null terminieren.

```
CLR ENVIRONMENT(Zeiger auf die Variable)
```

Befehlsnummer: 11
Fehlercodes: -

8.7. Sonstige Systemkommandos

8.7.1. FEHLERAUSGABE

Der Systembefehl FEHLERAUSGABE bestimmt zu einem Fehlercode den zugehörigen Fehlerstring und gibt diesen mit führenden und folgenden CR/LF an die Datei StandardOut aus. Die gebräuchlichsten SCSI-Fehlermeldungen können auch erkannt und ausgegeben werden.

```
FEHLERAUSGABE(Fehlercode)
```

Befehlsnummer: 29
Fehlercodes: N-Flag Tritt ein Fehler bei der Ausgabe an StandardOut auf wird nur das N-Flag gesetzt.

8.7.2. SHELL

Der Systembefehl SHELL ist auch nur aus der Notwendigkeit, den Shell-Programmtext zugänglich zumachen, entstanden. Es erfolgt einfach ein Sprung zur Startadresse der Shell. Diese Funktion wird von dem Dienstprogramm SHELL aufgerufen. Die Shell beendet sich, falls von der Datei StandardIn ein End-Of-File gemeldet wird.

```
SHELL
```

Befehlsnummer: 30
Fehlercodes: 103 Zuwenig Speicherplatz!
Die Shell konnte ihren Arbeitsspeicher nicht allokieren.

8.7.3. DELAY

Der Systembefehl wartet eine angegebene Anzahl von Timer-Ticks (1 Timer-Tick ca. 1/16 Sekunde). Während des Wartens wird ein Taskwechsel ausgeführt.

```
DELAY(Anzahl der Timer-Ticks)
```

Befehlsnummer: 31
Fehlercodes: -

8.8. Shell

Die Shell ist die primäre Schnittstelle zwischen einem Benutzer und dem Betriebssystem. Die Shell zeigt als Eingabeaufforderung die aktuelle CWD an. Programme, bei denen das EXECUTABLE-Flag gesetzt ist, können Parameter angeben, die Ein-/Ausgabe umgelenkt und als Programm oder als Task gestartet werden. Ein Ein-/Ausgabeumlenkung bedeutet, daß für das zu startende Programm ein anderes Gerät oder eine andere Datei als Eingabequelle und Ausgabeziel gewählt werden kann, als das der aktuellen Shell. Die Programmnamen können mit oder ohne Pfad angegeben werden. Ohne Pfadangabe wird zuerst im aktuellen Verzeichnis (CWD), danach im `path` nach dem Programm gesucht. Es empfiehlt sich als erste Pfadangabe **/rom/** der Environmentvariable `path` zu benutzen, um schnell auf die Dienstprogramme zugreifen zu können. Der Programmname muß als erstes in der Eingabezeile erscheinen.

Programmaufruf mit Pfadangabe:

```
/Verzeichnis1/Verzeichnis2/Verzeichnis3/Programmname
```

Parameter können beliebig viele, solange es die Größe des Environmentbereiches zuläßt, dem Programm übergeben werden. Sie müssen durch ein Space getrennt sein. Ein Programmaufruf ohne Pfadangabe mit drei Parametern hat die folgende Gestalt:

```
Programmname Parameter1 Paremeter2 Parameter3
```

Ein-/Ausgabeumlenkungen können beliebig mit den Parametern gemischt werden. Umlenkung der Datei StandardIn erfolgt durch

```
Programmname <Dateiname
```

Entsprechend erfolgt die Umlenkung der Datei StandardOut durch

```
Programmname >Dateiname
```

Für Umlenkung der Datei StandardIn und StandardOut in die gleiche Datei bedient man sich der Zeichenfolge

```
Programmname <>Dateiname
```

Um ein Programm nicht als Unterprogramm der Shell sondern als Task zu starten muß dem Dateinamen ein **&** vorausgestellt sein. Der Task wird ein Environmentbereich von 256 Worten angelegt, sie erbt den Environmentbereich der Shell. Als Stackpointer wird die letzte Adresse der Programms benutzt. Findet keine Ein-/Ausgabeumlenkung statt wird der Task das Null-Device als Datei für StandardIn und StandardOut übergeben. Ein Unterprogramm erbt bei nicht erfolgter Umlenkung StandardIn und StandardOut der Shell.

Eine Shell beendet sich bei einem End-Of-File von der StandardIn-Datei. Von Tastatur wird ein End-Of-File mit der Tastenkombination [CTRL-z] erzeugt.

Die Taste [F1] löscht die Eingabezeile. Mit der Taste [F3] kann der letzte Befehl wieder in die Eingabezeile eingetragen werden, sofern nach erscheinen der Eingabeaufforderung keine weitere Taste gedrückt wurde.

8.9. Dienstprogramme

8.9.1. Allgemeines

Alle Dienstprogramme dieser Betriebssystemversion sind auf der ROM-Disk abgelegt. Die ROM-Disk hat die gleiche Struktur wie ein SCSI-Blockdevice und ist nur lesbar. Die Dienstprogramme werden in den Arbeitspeicher geladen und dort ausgeführt. Der Zugriff auf die ROM-Disk ist aber weitaus schneller als der Zugriff auf die anderen Geräte.

Die im Folgenden aufgeführten Dienstprogramme verarbeiten in der Regel die Wildcards ? und *.

Die Dienstprogramme COPY, DEL und TYPE sind durch die Tastenkombination [CTRL + c] unterbrechbar.

DEL, FORMAT und FMLL beinhalten eine Sicherheitsabfrage, um ungewollten Datenverlust zu vermeiden.

Zahlenausgaben der Dienstprogramme sind immer hexadezimal und wortbezogen. Im folgenden werden optionale Parameter in folgender Form angegeben

```
[optionaler Parameter]
```

8.9.2. DIR

Das Dienstprogramm DIR zeigt den Inhalt eines Verzeichnisses ganz oder teilweise an. Ist kein Parameter angegeben wird das gesamte Inhalt des aktuellen Verzeichnisses (CWD) angezeigt. Das gleiche geschieht bei * als Parameter. Es kann durch Pfadangabe auch ein beliebiges Verzeichnis angezeigt werden. Die Dateilänge ist eine Angabe in Worten. Die Dateilänge eines Verzeichnisses gibt Auskunft über das Verzeichnis selbst, nicht Gesamtlänge aller darin enthaltenen Dateien

```
dir [/Pfad/Suchstring]
```

8.9.3. COPY

COPY kopiert beliebig viele Dateien von einem Verzeichnis in ein anderes. Es besteht die Möglichkeit beim Kopieren Dateien umzubenennen. Quell- und Zielpfad müssen mit einem / enden. Ist das CWD der Quellpfad muß nur der Dateiname angegeben werden. Ist der Zielpfad das CWD muß als Zielpfad ein **.** angegeben werden. Verzeichnisse können nicht kopiert werden.

```
copy Quellpfad/Dateiname Zielpfad/[Neuer Dateiname]
```

Beispiel für Zielpfad gleich CWD:

```
copy /a/* .
```

Es werden alle Dateien des Verzeichnisses /a/ in das aktuelle Verzeichnis kopiert.

Beispiel für Quellpfad gleich CWD und Namensänderung:

```
copy Hallo /b/Servus
```

8.9.4. DEL

DEL entfernt Dateien. Verzeichnisse und schreibgeschützte Dateien können nicht entfernt werden. Ist kein Parameter angeben werden alle Datei des aktuellen Verzeichnisses entfernt. Es können bei Pfadangabe auch Dateien in anderen Verzeichnissen entfernt werden.

```
del [/Pfad/Dateiname]
```

8.9.5. CD

Mit dem Dienstprogramm CD wird das aktuelle Verzeichnis gewechselt. Ist nur ein Verzeichnisname angeben wird ein Verzeichnisebene tiefer gewechselt, ist .. als Pfad angegeben wird eine Verzeichnisebene höher gewechselt. Auch der Gerätewechsel wird mit diesen Befehl durchgeführt. Der letzte Verzeichnisname muß ohne / angegeben werden.

```
cd /Pfad
```

Beispiele:

```
cd ..
cd Unterverzeichnis
cd /rom
```

8.9.6. MD

MD erzeugt ein neues Verzeichnis. Wird kein Pfad angegeben wird im aktuellen Verzeichnis ein neues Verzeichnis erstellt. Es wird überprüft, ob schon eine Datei mit dem gewünschten Namen existiert.

```
md [/Pfad/]Verzeichnisname
```

8.9.7. RD

Mit dem Dienstprogramm RD wird ein Verzeichnis entfernt. Das Verzeichnis muß zu diesem Zweck geleert sein. Ein Pfad kann mit angegeben werden.

```
rd [/Pfad/]Verzeichnisname
```

8.9.8. REN

REN benennt Dateien um. Wildcards werden verarbeitet. Es wird überprüft, ob der gewünschte Dateiname schon existiert.

```
ren [/pfad/]AlterName NeuerName
```

8.9.9. FORMAT

FORMAT führt ein High-Level-Format durch. Es wird der Bootsektor und die BAM erstellt. Dem Dienstprogramm FORMAT muß der Gerätename und ein Datenträgername übergeben werden. Der Datenträgername wird beim Berabeiten des Datenträgers in einem Media-Descriptor-Block gespeichert und zur Identifikation des Mediums benutzt. Da die meisten Datenträger schon Low-Level-formatiert sind, ist dies weitaus schneller als pauschal ein Low-Level-Format durchzuführen

```
format Gerätename Datenträgername
```

8.9.10. FMLL

Das Dienstprogramm FMLL führt ein Low-Level-Format durch. Es muß nur der Gerätename angegeben werden.

```
fmll Gerätename
```

8.9.11. ATTRIB

Das Dienstprogramm ATTRIB verändert die Dateiflags. Es findet dabei keinerlei Überprüfung statt, ob die Attribute korrekt sind, es zum Beispiel wirklich ein Verzeichnis ist. Die Datei wird zur Veränderung der Dateiattribute zum Lesen geöffnet. Es besteht also die Möglichkeit, daß eine andere Task auf dieser Datei lesend operiert während die Dateiflags verändert werden. Um das jeweilige Dateiflag zu setzen muß es durch einen Buchstaben angegeben werden, ansonsten wird es gelöscht, die Reihenfolge der Flags ist ohne Bedeutung. Ist nichts angegeben werden alle Dateiflags gelöscht.

Vorsicht! Wird bei einer Directory das Directory-Attribut gelöscht kann auf die Dateidaten nicht mehr zugeriffen werden. Das Attribut kann jedoch wieder gesetzt werden. Wird dies nicht getan ist die BAM nicht mehr aktuell, da die Datensektoren der im Verzeichnis enthaltenen Dateien nicht freigegeben werden.

```
attrib Dateiname [w][x][d]
```

8.9.12. DISKINFO

DISKINFO ermittelt den Datenträgernamen, die Kapazität und den noch verbleibenden Speicherplatz auf einem Datenträger. Die Kapazität und der verbleibende Speicherplatz wird wortbezogen und hexadezimal ausgegeben.

```
diskinfo Gerätename
```

8.9.13. DRVLST

DRVLST zeigt alle dem System bekannten Gerätetreiber an, und teilt mit ob dieses Gerät ein Zeichen- oder Blockdevice ist.

```
drvlst
```

8.9.14. MEM

Das Dienstprogramm MEM ermittelt den noch freien Arbeitsspeicher insgesamt und die Größe des längsten noch existierenden Speicherblock.

```
mem
```

8.9.15. TASKLST

TASKLST zeigt alle in System befindlichen Tasks, mit ihrem Tasknamen, ihrer Tasknummer und ihren StandardIn- und StandardOut-Devices an. Die aktuelle, so zu sagen die eigene Task, wird als erstes genannt.

```
tasklst
```

8.9.16. KILL

KILL entfernt eine Task aus dem System. Identifiziert wird die Task mit ihrer Tasknummer. Wird die letzte Task entfernt wird ein RESET ausgelöst.

```
kill Tasknummer
```

8.9.17. TYPE

Mit dem Dienstprogramm TYPE kann man den Inhalt einer Datei anzeigen. Auch spezielle Grafiken mit der erweiterten VT52-Sequenz ESC G (siehe VT52-Emulator 7.8.6. Konsolen-Treiber) lassen sich mit diesem Dienstprogramm anzeigen. Es wird ein Wort an das Ausgabegerät übergeben, das heißt, das Texte die an die Konsole übergeben werden nur ein Zeichen pro Wort enthalten dürfen. Das höherwertige Byte wird nicht angezeigt.

```
type Dateiname
```

Bemerkung: Durch eine spezielle Form des TYPE-Dienstprogrammes mit Ausgabeumlenkung kann man auch in Dateien per Tastatur schreiben.

```
type /con1 >Dateiname
```

8.9.18. SET

SET ohne Parameter zeigt den Inhalt des Environmentbereiches an. SET mit einem Parameter der Form **STRING=STRING** trägt eine neue Variable in den Environmentbereich ein. Und ein Aufruf des Dientsprogrammes mit einem Parameter der Form **STRING=** löscht die entsprechende Environmentvariable.

```
set [String=[String]]
```

8.9.19. SHELL

Das Dientsprogramm SHELL startet ein weitere Shell. Um einfache Batchdateien zu verarbeiten kann man eine Shell mit StandardIn-Datei gleich der Batchdatei wählen. Die Shell beendet sich dann, sobald die Batchdatei zu Ende ist. Sinnvoll ist auch das Starten einer weiteren Shell als Task mit StandardIn- und StandardOut-Datei **/con1** oder **/ser** auf einem virtuellen Terminal.

```
shell
```

8.9.20. CLS

CLS löscht den Bildschirm mit der VT52-Sequenz ESC E.

```
cls
```

8.9.21. BCOLOR

BCOLOR setzt die Hintergrundfarbe mit der VT52-Sequenz ESC b Farbwert. Zulässig sind Farbwerte zwischen 0 und 7.

```
bcolor Farbwert
```

8.9.22. FCOLOR

FCOLOR setzt die Schriftfarbe mit der VT52-Sequenz ESC c Farbwert. Zulässig sind Farbwert zwischen 0 und 7.

```
fcolor Farbwert
```

8.10. Beachtenswertes bei eigenen Programmen

8.10.1. Stack

Wird ein Programm von der Shell aus gestartet, so wird der Stackpointer auf die letzte Adresse dieses Programms eingestellt, und die Rücksprungadresse dort abgelegt. Dieser Stack wird auch bei Systemaufrufen benutzt. Es spielt dabei keine Rolle ob das Programm als Task oder Unterprogramm der Shell ausgeführt wird. Es ist daher nötig das jedes Programm am Ende seines Bereiches Arbeitspseicher für Stack bereithält. Bei Fileoperation empfiehlt sich ein Stackbereich von 128 Worten, das ist auch der Wert mit denen die Dienstprogramme arbeiten.

8.10.2. Environmentbereich

Der Environmentbereich der von der Shell angelegt wird besitzt eine Größe von 256 Worten. So lassen sich längere Pfadangaben und ein umfangreiches CWD anlegen. Auch die Parameterangabe erfolgt über den Environmentbereich.

8.10.3. Parameterübergabe

Die Parameterübergabe von der Shell an Programme erfolgt über den Environmentbereich. Alle Parameter sind zugänglich über die Environmentvariable **cmd**. Die Parameter werden alle durch ein Space getrennt.

```
cmd=parameter1 parameter2 parameter3
```

8.10.4. Besondere Systemvariablen

Die Systemvariablen TIMERL, TIMERH und AKT_TASK sind über die Library LIB.ASM zugänglich. Werden Informationen über andere Systemvariablen benötigt, so werden diese im Kapitel SYSTEMREFERENZ genannt.

8.10.5. Assemblieren an eine bestimmte Adresse

Ein Assemblieren an eine bestimmte Adresse ist für User-Programme dieses Systems nicht geeignet, da die Programme an beliebige Positionen im Arbeitsspeicher geladen und dort reloziert werden.

8.10.6. Task und Unterprogramm

Bei der Programmierung eines Programm muß man keine Unterscheidung treffen, ob dieses Programm als Unterprogramm oder Task verwendet werden soll. Der Aufruf `rts` am Ende eines Programmes bewirkt beim Start als Unterprogramm der Shell ein Zurückkehren zur Shell, beim Start als Task einen Systemaufruf KILLTASK. Die Vorbereitung dieser beiden Sachen übernimmt jedoch die Shell. Sie pushed beim Start als Task die Tasknummer und die Adresse der KILLTASK-Routine auf den Stack des Programms.

8.11. Fehlercodes

Die Fehlercodes des Betriebssystems beginnen mit der Nummer 100.

Fehler# (dez)	Fehler# (hex)	Bedeutung
100	$64	Es gibt keine ausführbare Datei dieses Namens!
101	$65	Ungültiges Handle!
102	$66	Befehlszugriff nicht erlaubt!
103	$67	Zuwenig Speicherplatz!
104	$68	Dateiname hat falsches Format!
105	$69	End-Of-File erreicht!
106	$6a	Device existiert nicht!
107	$6b	Environmentbereich zu klein!
108	$6c	Maximale Dateilänge (32MB) erreicht!
109	$6d	Datei existiert nicht!
110	$6e	Datei existiert schon!
111	$6f	Parameter falsch!
112	$70	Datenträger voll!
113	$71	Directory existiert nicht!
114	$72	Datei ist schreibgeschützt!
115	$73	Zugriffskonflickt auf mehrfach benutzte Datei!
116	$74	Directory ist nicht geleert!
117	$75	Datei ist nicht ausführbar!
118	$76	Es liegt kein Zeichen an!

Tabelle 82: Betriebssystemfehlermeldungen

Fehler#	Bedeutung
$02xx	SCSI-Laufwerk nicht bereit!
$03xx	SCSI-Medienfehler!
$04xx	SCSI-Hardwarefehler!
$05xx	Ungültige SCSI-Anfrage!
$06xx	SCSI-Gerät wurde zurückgesetzt!
$07xx	Medium in SCSI-Laufwerk ist schreibgeschützt!
$08xx	Herstellerspezifischer SCSI-Fehler aufgetreten!
$09xx	SCSI-Schreibfehler!

Tabelle 83: SCSI-Fehlermeldungen

8.12. Erstellen eines Programms

Ein Programm kann mit dem in der Trimesterarbeit [HUF93] erstellten Assembler XASSIDE.EXE auf eine PC erstellt werden. Dabei darf der Befehl **origin** nicht verwendet werden, da sonst kein ausführbares Programm erstellt wird. Als erste Anweisung sollte das Programm

```
include lib.asm
```

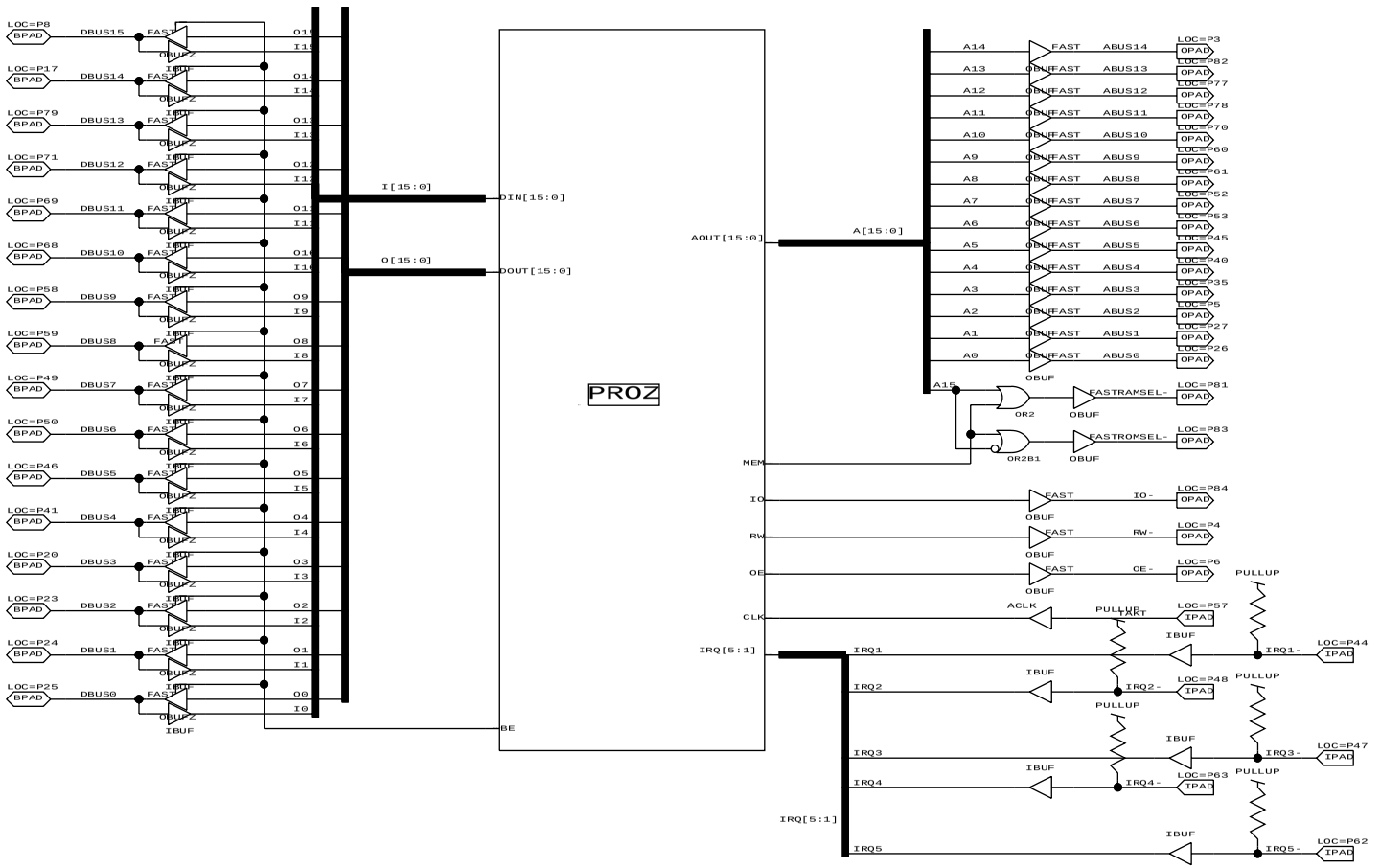
enthalten, um die Betriebssystemfunktionen nutzen zu können.

Nach dem Assemblieren existiert eine ausführbare Datei mit der Endung **.obj**, die von dem Hilfsprogramm XDISK.EXE verarbeitet werden kann. Nach starten des Hilfsprogramms XDISK.EXE kann man den Dateinamen der zu transportierenden Datei angeben. In das 3,5"-Laufwerk muß eine Low-Level-formatierte Diskette eingelegt werden. Auf dieser Diskette wird ein vom XSystem lesbares Dateisystem erzeugt. So ist dann das erzeugte Programm auf dem XSystem verfügbar.

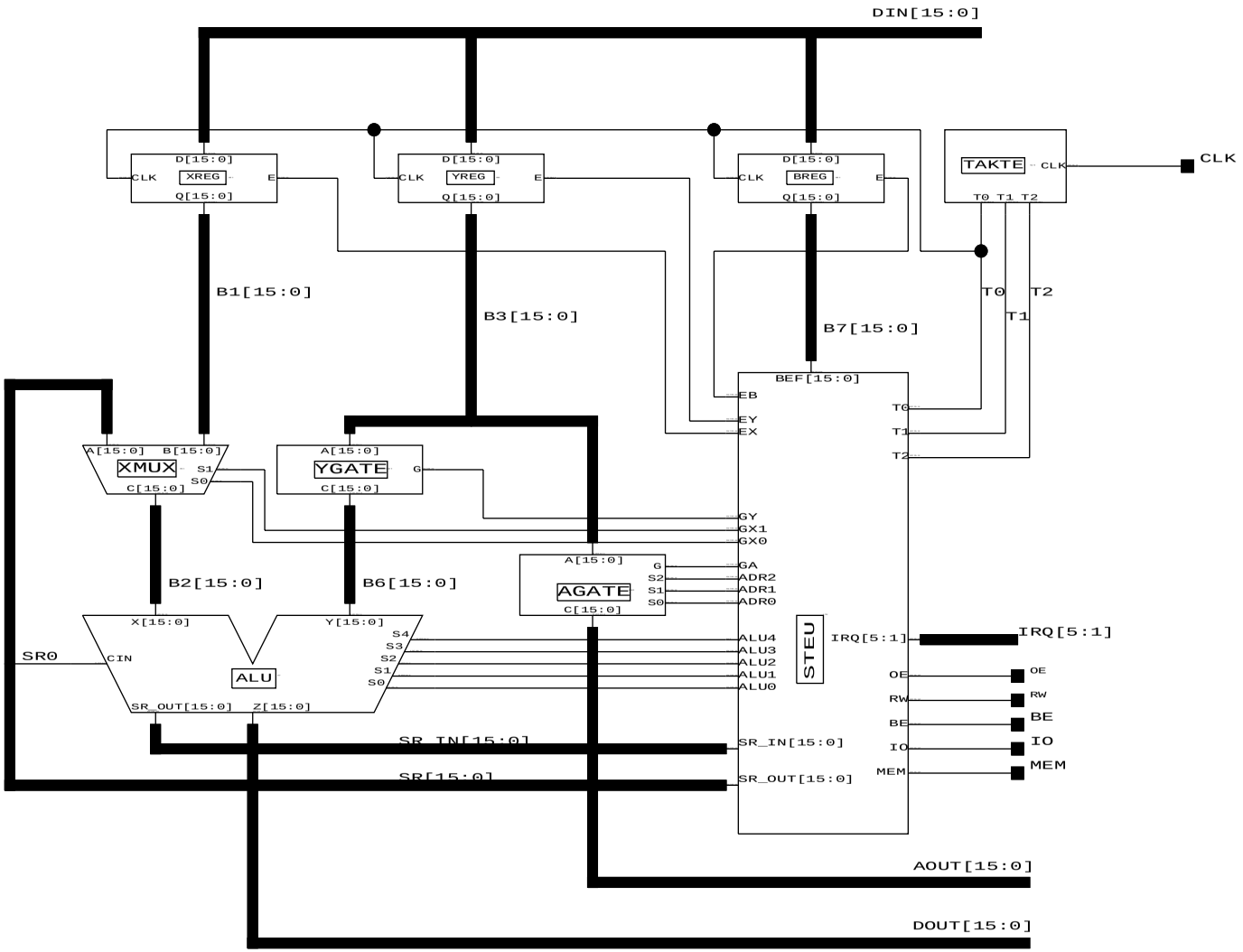
9. Anhang A: Schaltpläne

9.1. Der Prozessor XProz

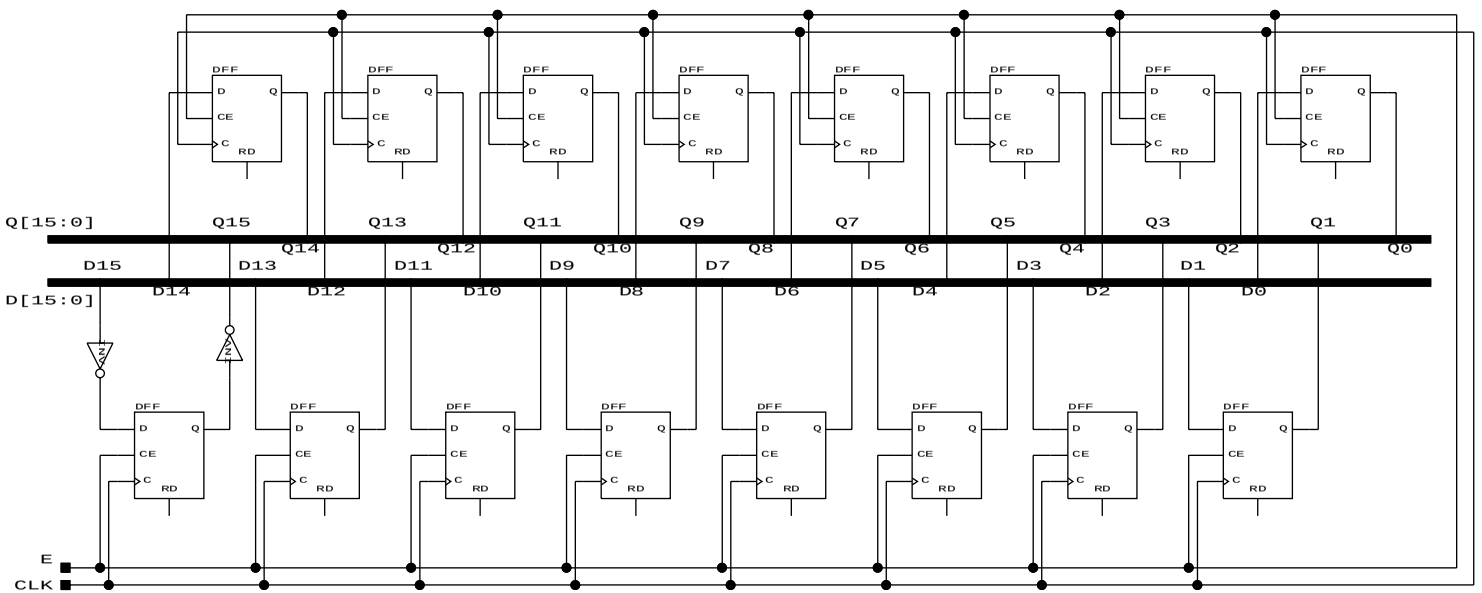
9.1.1. Pad-Locations (PROZ)


PROZ
I[15:0]
O[15:0]
DIN[15:0]
DOUT[15:0]
AOUT[15:0]
A[15:0]
MEM
IO
RW
OE
CLK
IRQ[5:1]
BE
DBUS15
DBUS14
DBUS13
DBUS12
DBUS11
DBUS10
DBUS9
DBUS8
DBUS7
DBUS6
DBUS5
DBUS4
DBUS3
DBUS2
DBUS1
DBUS0
ABUS14
ABUS13
ABUS12
ABUS11
ABUS10
ABUS9
ABUS8
ABUS7
ABUS6
ABUS5
ABUS4
ABUS3
ABUS2
ABUS1
ABUS0
EASTRAMSEL-
EASTROMSEL-
IO-
RW-
OE-
IRQ1-
IRQ2-
IRQ3-
IRQ4-
IRQ5-
ACLK
FAST
OBUF
IBUF
OBUFT
IPAD
OPAD
BPAD
OR2
OR2B1
PULLUP

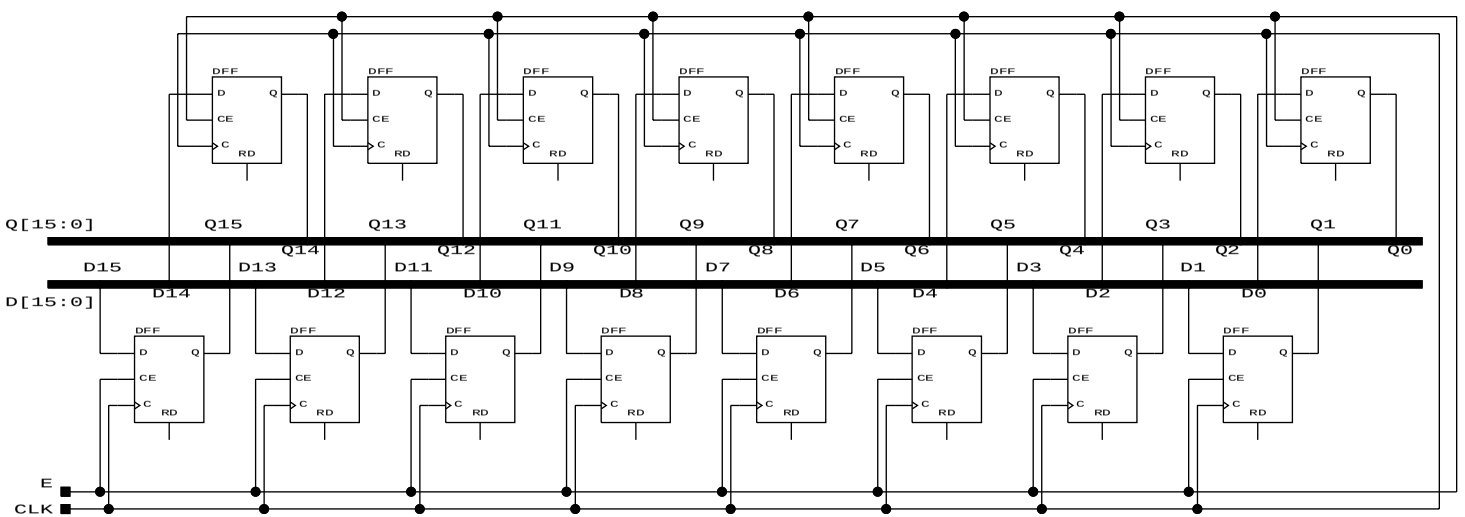
9.1.2. Hauptschaltung (PROZ)


DIN[15:0]
D[15:0]
CLK
XREG
E
Q[15:0]
YREG
BREG
TAKTE
CLK
T0 T1 T2
B1[15:0]
B3[15:0]
B7[15:0]
T0
T1
T2
BEF[15:0]
EB
EY
EX
A[15:0]
B[15:0]
XMUX
S1
S0
C[15:0]
YGATE
G
AGATE
GY
GX1
GX0
GA
ADR2
ADR1
ADR0
B2[15:0]
B6[15:0]
X[15:0]
Y[15:0]
SR0
CIN
ALU
S4
S3
S2
S1
S0
SR_OUT[15:0]
Z[15:0]
ALU4
ALU3
ALU2
ALU1
ALU0
STEU
IRQ[5:1]
OE
RW
BE
IO
MEM
SR_IN[15:0]
SR[15:0]
AOUT[15:0]
DOUT[15:0]

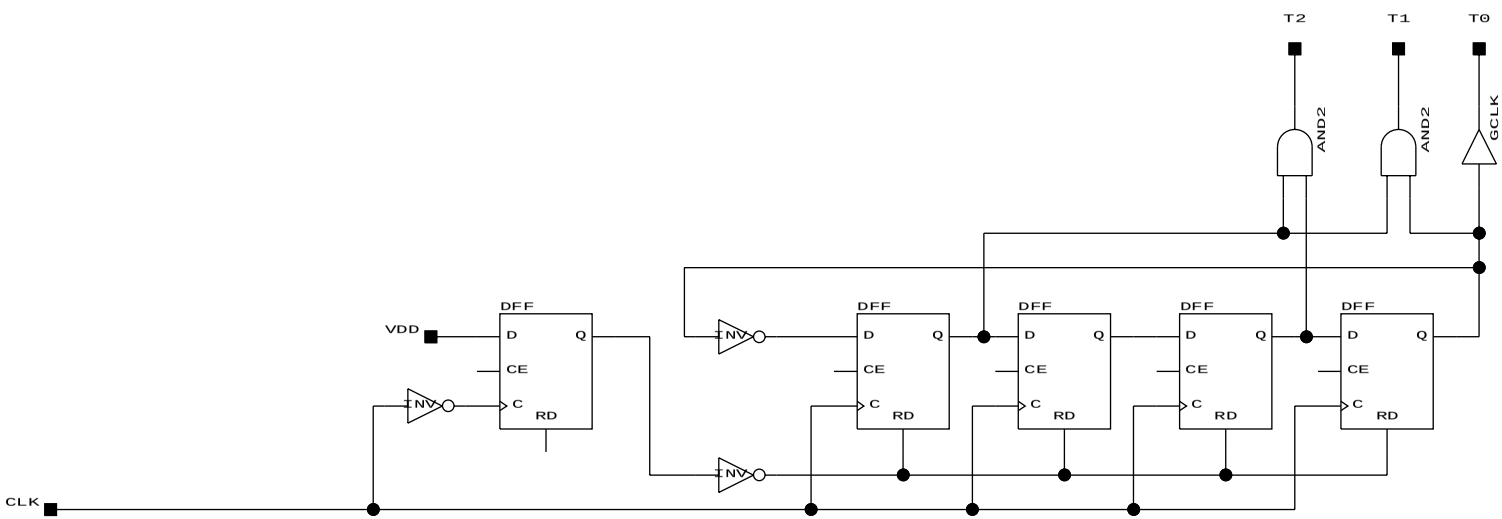
9.1.3. Y-Register (YREG)


DFF
D
Q
CE
C
RD
Q[15:0]
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
D[15:0]
INV
E
CLK

9.1.4. Befehlsregister und X-Register (BREG und XREG)


DFF
D
Q
CE
C
RD
Q[15:0]
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
D15
D14
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
D[15:0]
E
CLK

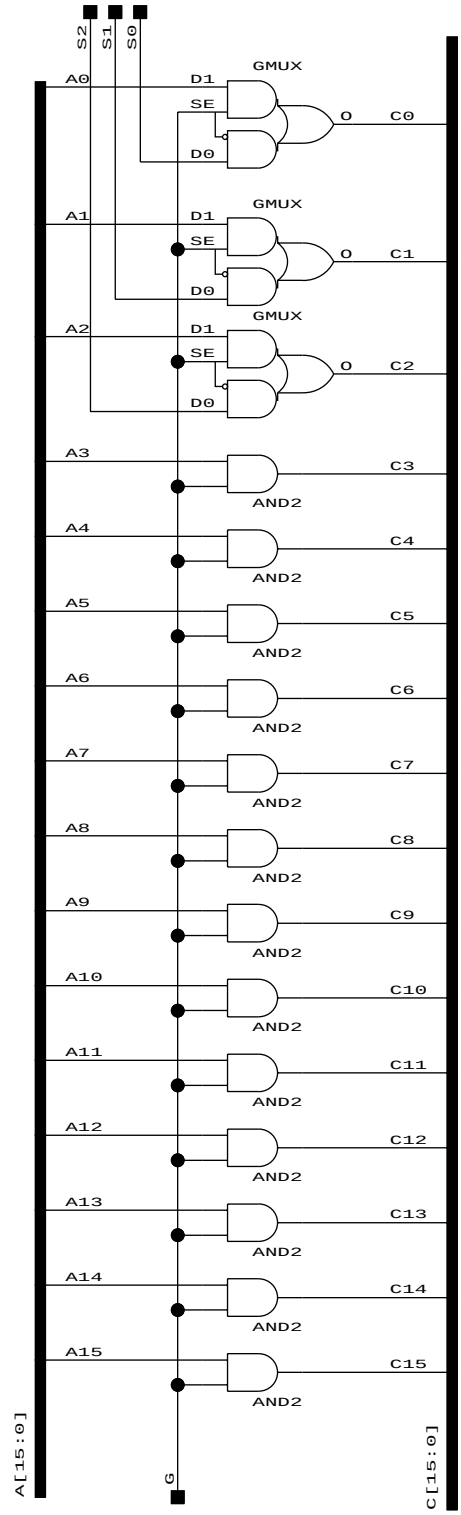
9.1.5. Taktgenerator (TAKTE)


T2
T1
T0
AND2
AND2
GCLK
VDD
CLK
DFF
D
Q
CE
C
RD
INV

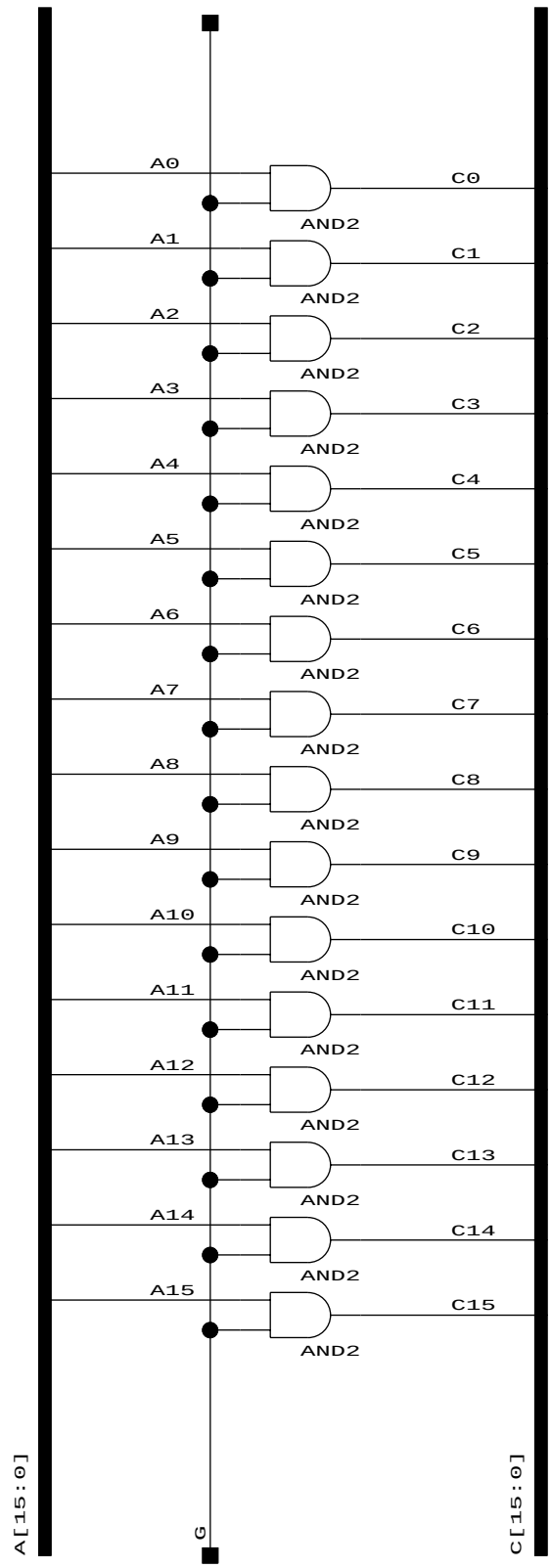
9.1.6. X-Multiplexer (XMUX)

B[15:0]
S0
A[15:0]
S1
C[15:0]
B0
OR2B1
B1
AND2
B2
AND2
B3
AND2
B4
AND2
B5
AND2
B6
AND2
B7
AND2
B8
AND2
B9
AND2
B10
AND2
B11
AND2
B12
AND2
B13
AND2
B14
AND2
B15
AND2
A0
A1
A2
A3
A8
A9
A10
A11
A12
A13
GMUX
D1
SE
D0
O
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15

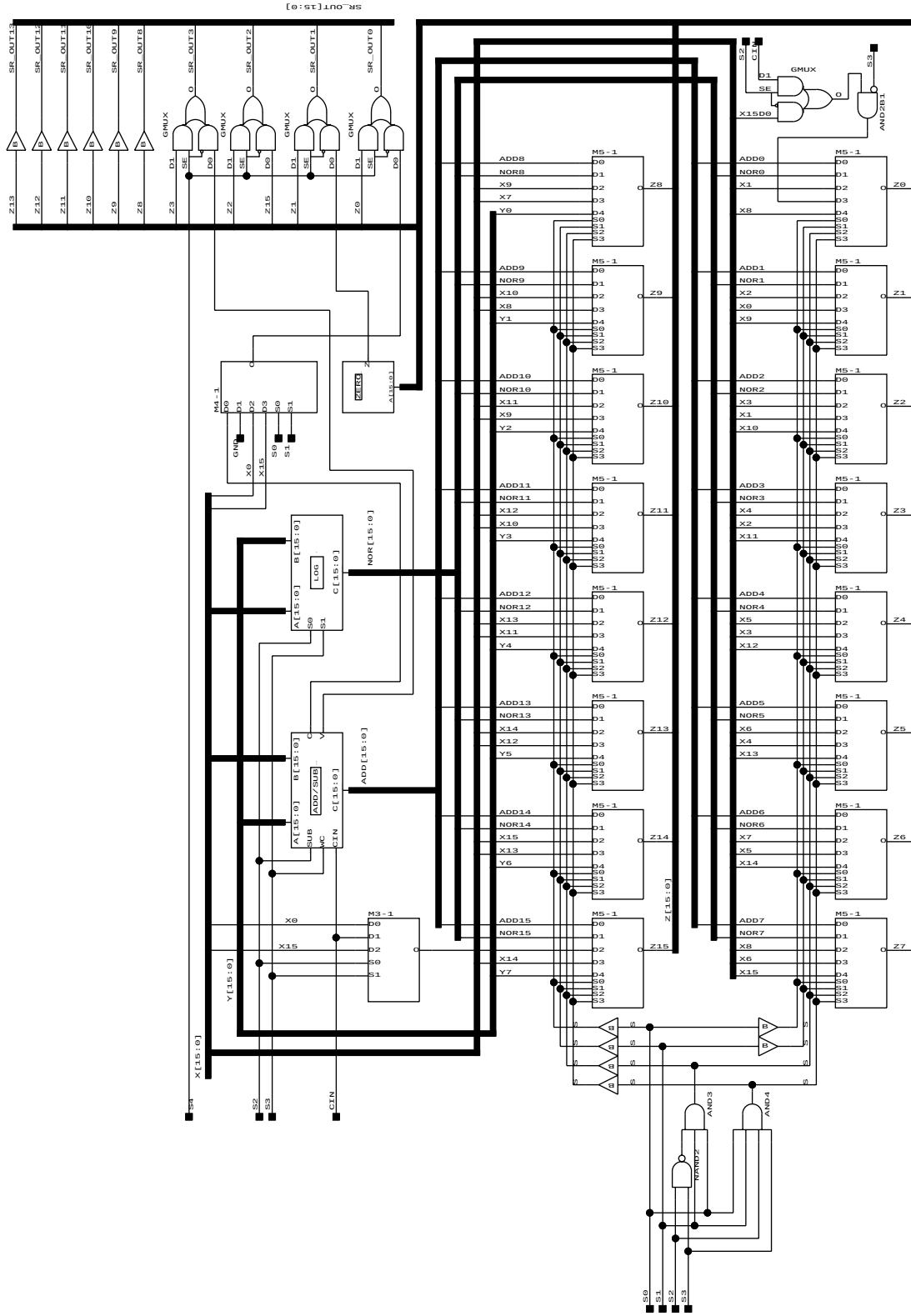
9.1.7. Adressmultiplexer (AGATE)


S2
S1
S0
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
GMUX
D1
SE
D0
O
C0
C1
C2
AND2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
A[15:0]
G
C[15:0]

9.1.8. Y-Multiplexer (YGATE)


A[15:0]
G
C[15:0]
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
AND2
C0
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15

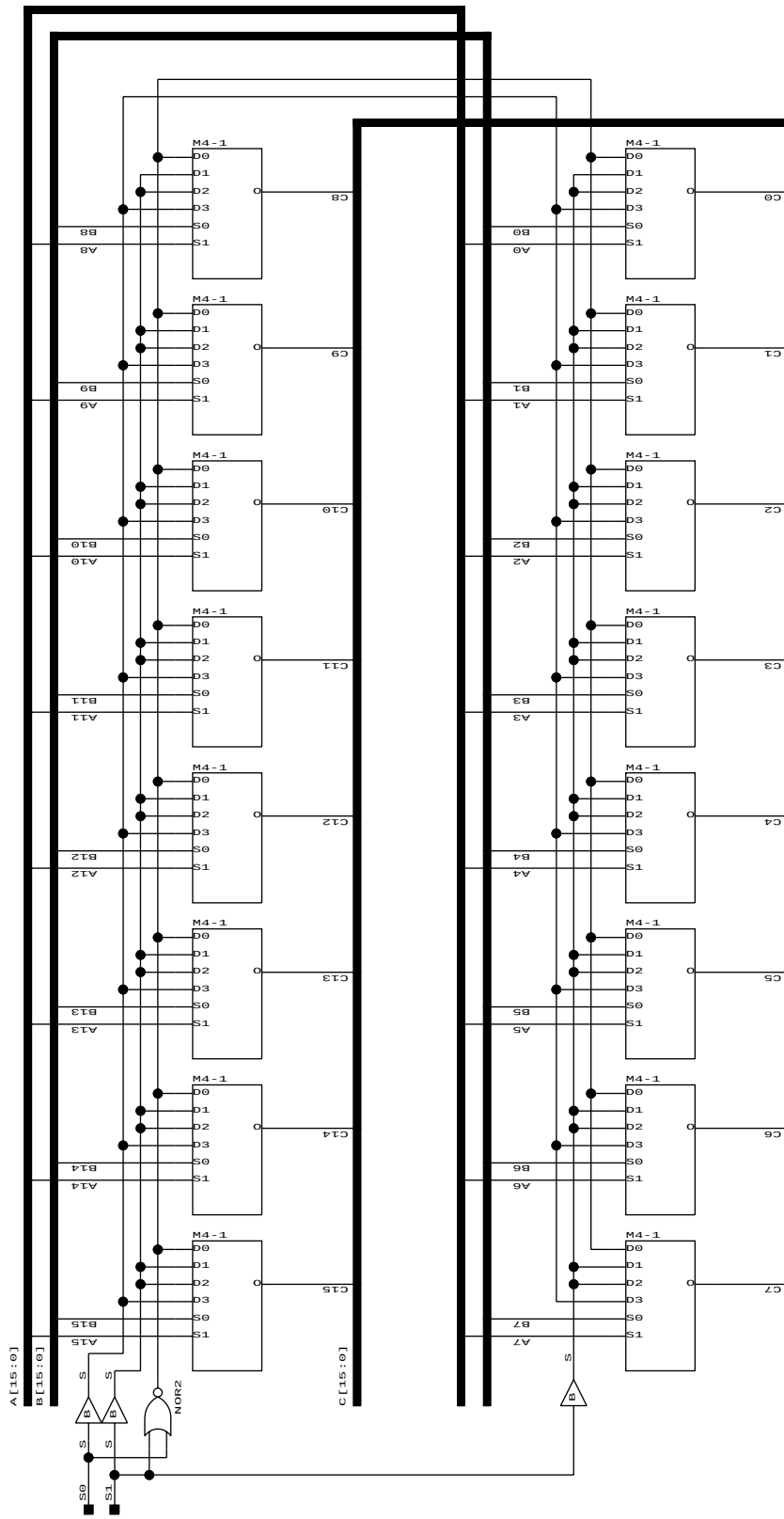
9.1.9. Arithmetic-Logical-Unit (ALU)


SR_OUT[15:0]
SR_OUT13
SR_OUT12
SR_OUT11
SR_OUT10
SR_OUT9
SR_OUT8
SR_OUT3
SR_OUT2
SR_OUT1
SR_OUT0
GMUX
Z13
Z12
Z11
Z10
Z9
Z8
Z3
Z2
Z15
Z1
Z0
M4-1
GND
X0
X15
S0
S1
ZERO
A[15:0]
NOR[15:0]
B[15:0]
C[15:0]
LOG
ADD[15:0]
ADD/SUB
SUB
WC
CIN
M3-1
Y[15:0]
X[15:0]
S4
S2
S3
M5-1
ADD8
NOR8
X9
X7
Y0
Z8
ADD9
NOR9
X10
X8
Y1
Z9
ADD10
NOR10
X11
X9
Y2
Z10
ADD11
NOR11
X12
X10
Y3
Z11
ADD12
NOR12
X13
X11
Y4
Z12
ADD13
NOR13
X14
X12
Y5
Z13
ADD14
NOR14
X15
X13
Y6
Z14
ADD15
NOR15
X14
Y7
Z15
Z[15:0]
ADD0
NOR0
X1
X8
Z0
ADD1
NOR1
X2
X0
X9
Z1
ADD2
NOR2
X3
X1
X10
Z2
ADD3
NOR3
X4
X2
X11
Z3
ADD4
NOR4
X5
X3
X12
Z4
ADD5
NOR5
X6
X4
X13
Z5
ADD6
NOR6
X7
X5
X14
Z6
ADD7
NOR7
X8
X6
X15
Z7
S2
CIN
D1
SE
X15D0
S3
AND2B1
AND3
AND4
NAND2
S0
S1
S2
S3

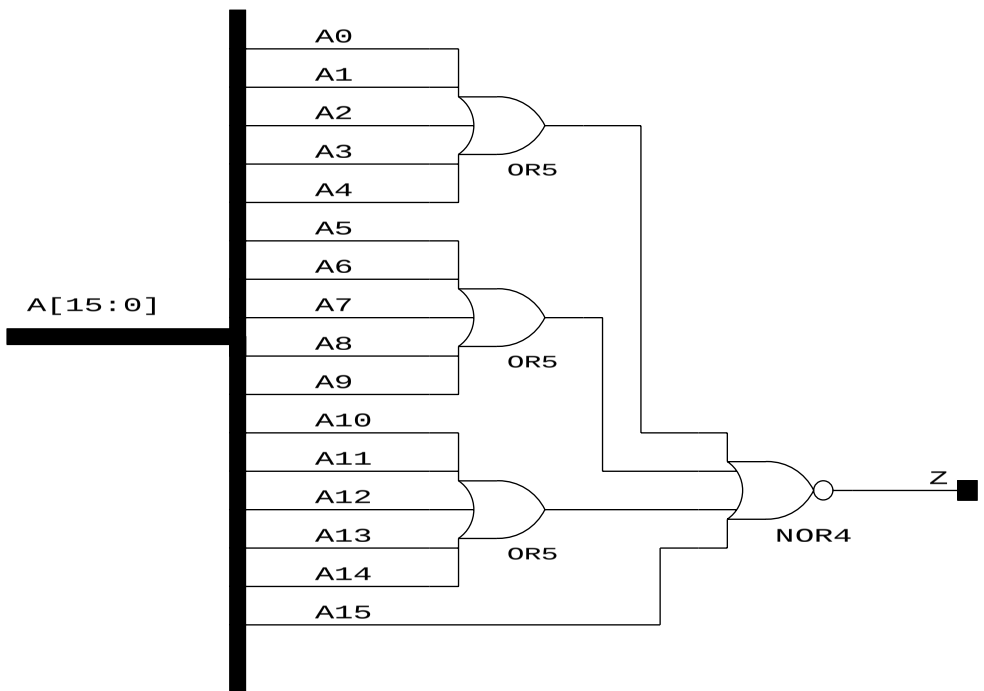
9.1.10. Addierwerk (ADDSUB)

CIN
WC
AND2
XOR2
HX83
C0
A1
A2
A3
A4
B1
B2
B3
B4
S1
S2
S3
S4
C4
A[15:0]
B[15:0]
AND3B1
AND3B2
OR2
V
S
B
SUB
XOR2
C
C[15:0]

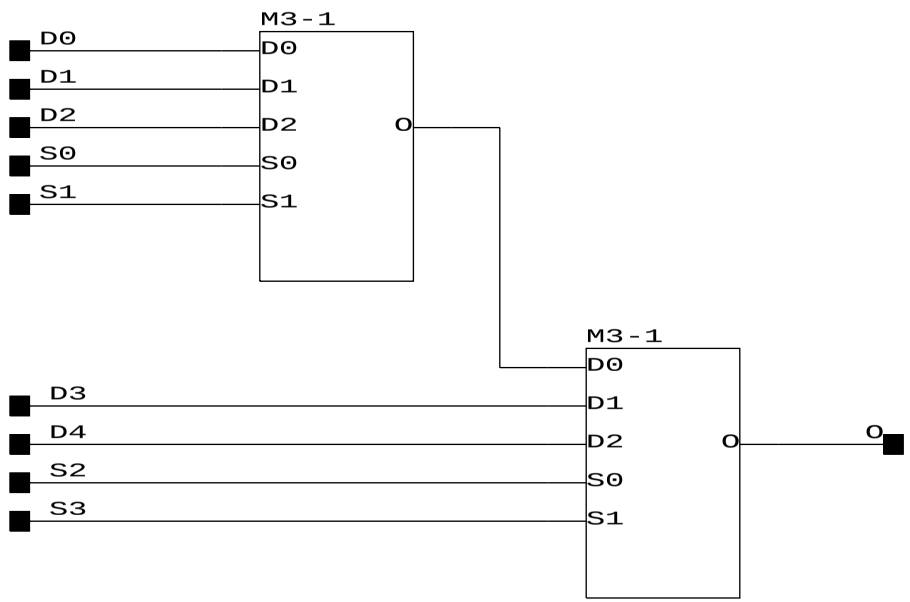
9.1.11. Logische Einheit (LOG)


M4-1
D0
D1
D2
D3
S0
S1
O
A[15:0]
B[15:0]
C[15:0]
NOR2
S0
S1

9.1.12. Zero-Flag (ZERO)

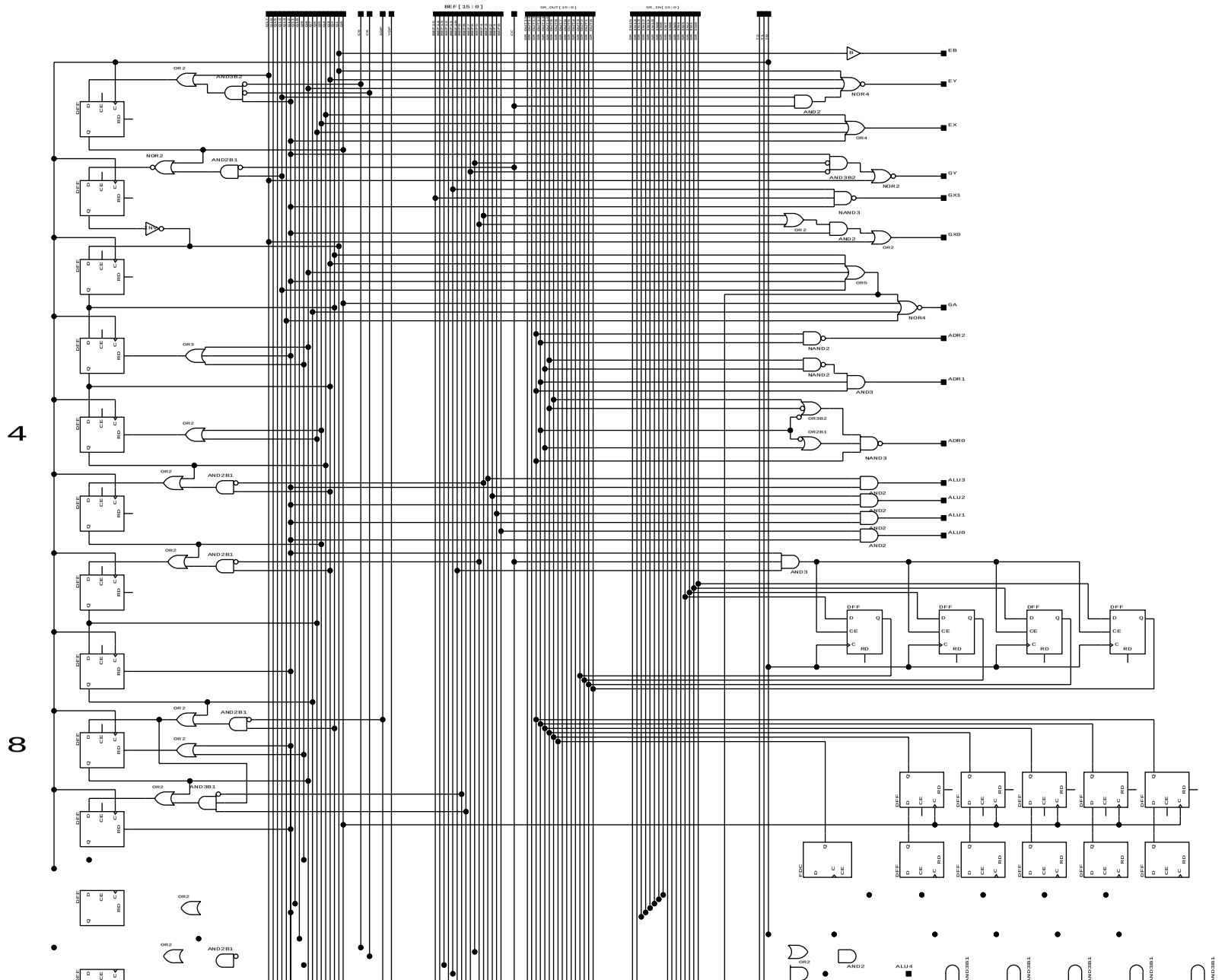

A[15:0]
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
OR5
OR5
OR5
NOR4
Z

9.1.13. 5-zu-1 Multiplexer (M5-1)

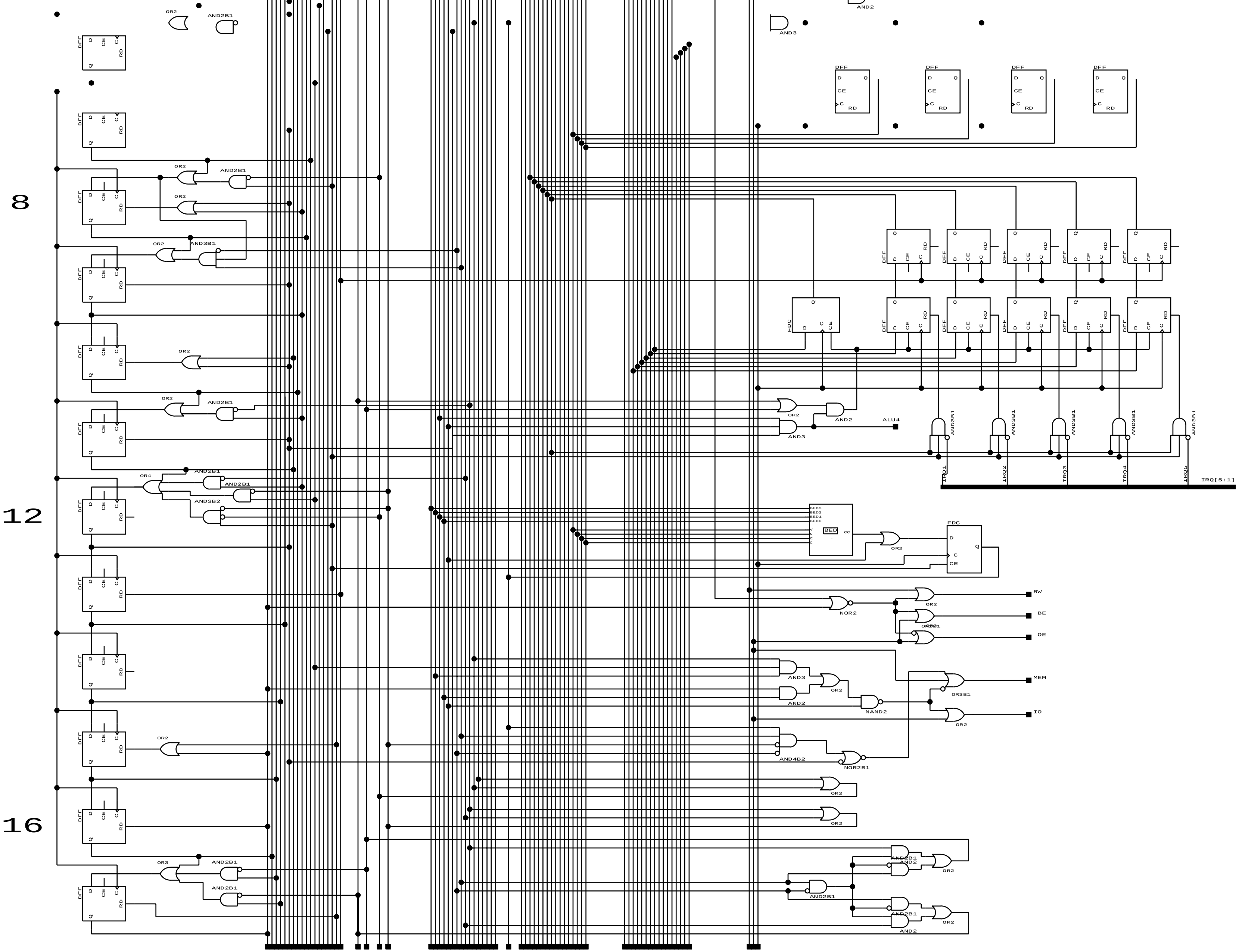

M3-1
D0
D1
D2
S0
S1
O
M3-1
D3
D4
S2
S3
D0
D1
D2
S0
S1
O
O

9.1.14. Steuersignaldecoder für M5-1 (MUXDECO)

9.1.15. Steuerwerk Teil1 (STEU)

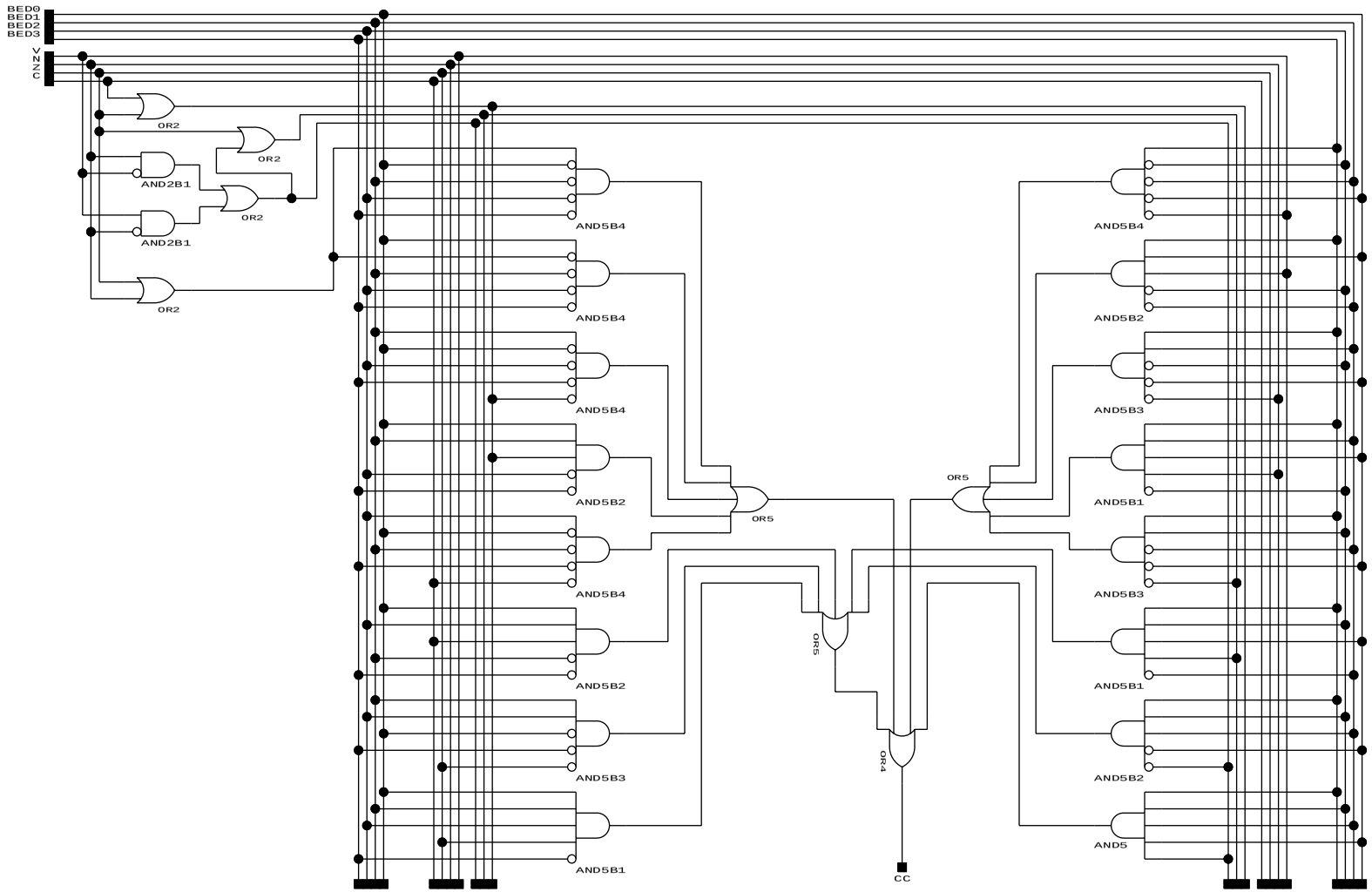

BEF[15:0]
SR_OUT[15:0]
SR_IN[15:0]
EB
EY
EX
GY
GX1
GX0
GA
ADR2
ADR1
ADR0
ALU3
ALU2
ALU1
ALU0
NOR4
NOR2
NAND3
NAND2
AND3
AND2
OR5
OR4
OR3
OR2
AND3B2
AND2B1
AND3B3
OR3B2
OR2B1
DFF
DFFC
ALU4
4
8

9.1.16. Steuerwerk Teil2 (STEU)



8
12
16
IRQ[5:1]
IRQ1
IRQ2
IRQ3
IRQ4
IRQ5
ALU4
RW
BE
OE
MEM
IO

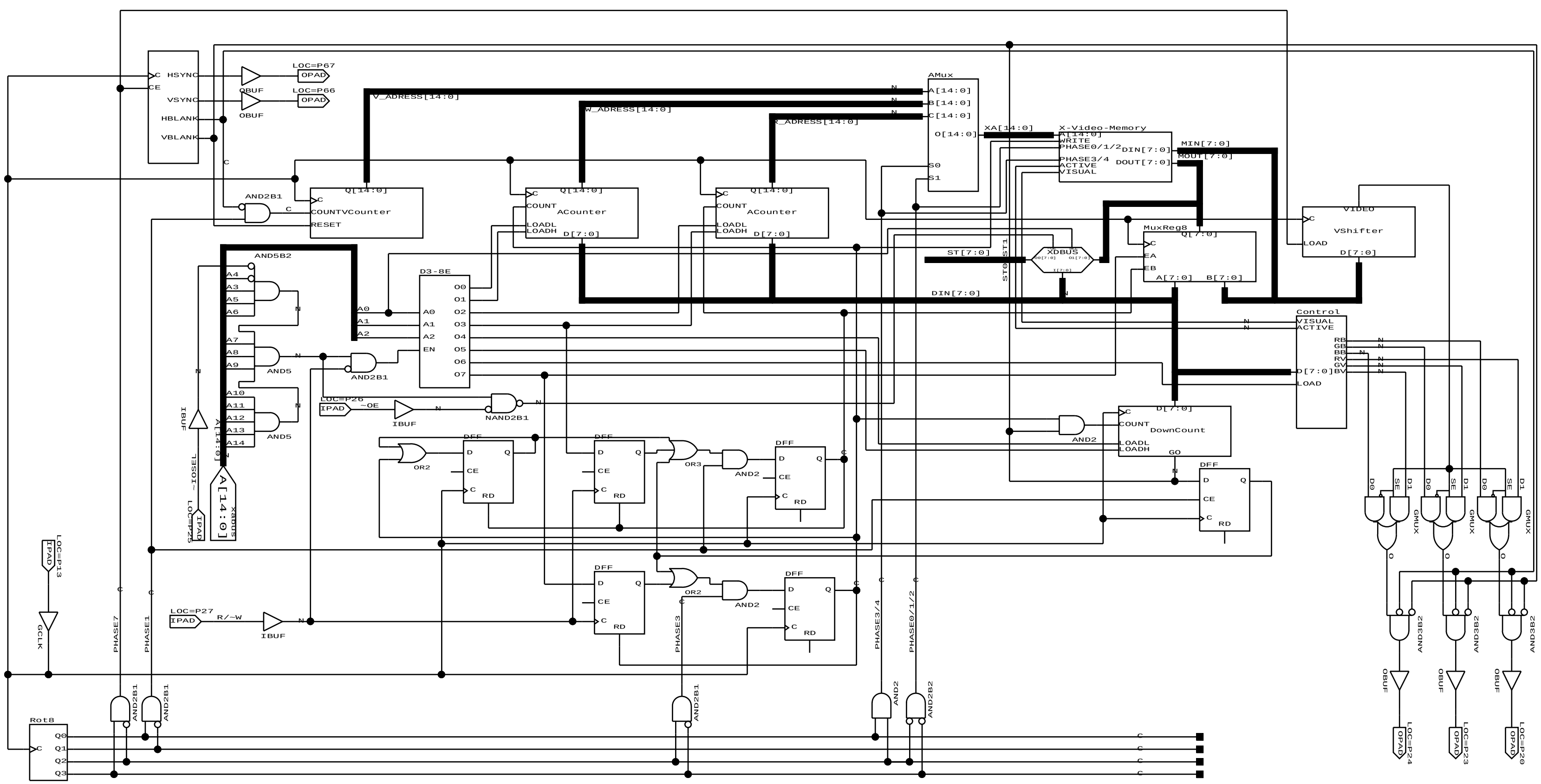
9.1.17. Condition-Code (BED)


BED0
BED1
BED2
BED3
VNC
OR2
OR2
AND2B1
OR2
AND2B1
OR2
AND5B4
AND5B4
AND5B4
AND5B2
AND5B4
AND5B2
AND5B3
AND5B1
OR5
OR5
OR4
CC
AND5B4
AND5B2
AND5B3
AND5B1
AND5B3
AND5B1
AND5B2
AND5

9.1.18. Programm-Counter Ermittlung (IRQ-GEN)

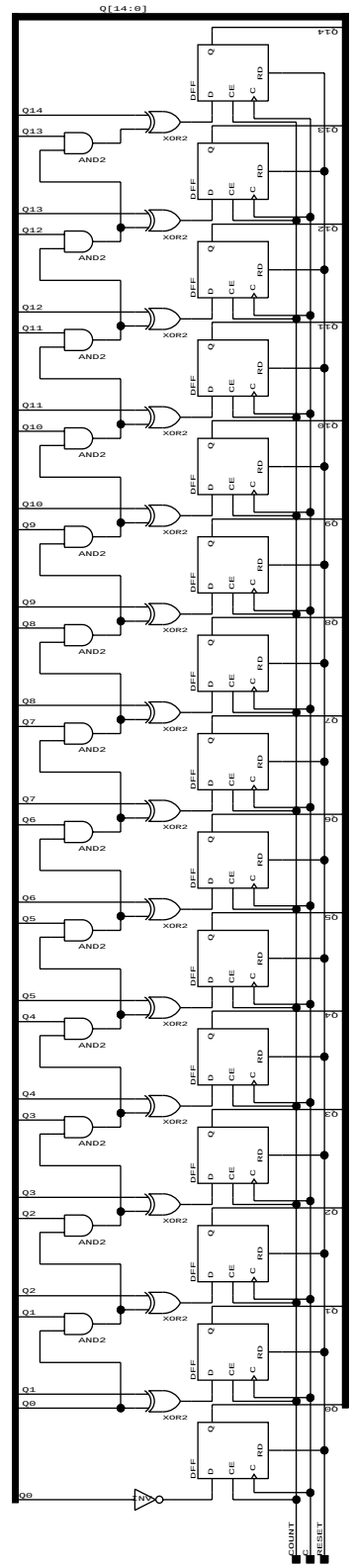
9.2. Die Grafikkarte XGraph

9.2.1. Hauptschaltung (XGRAPH)

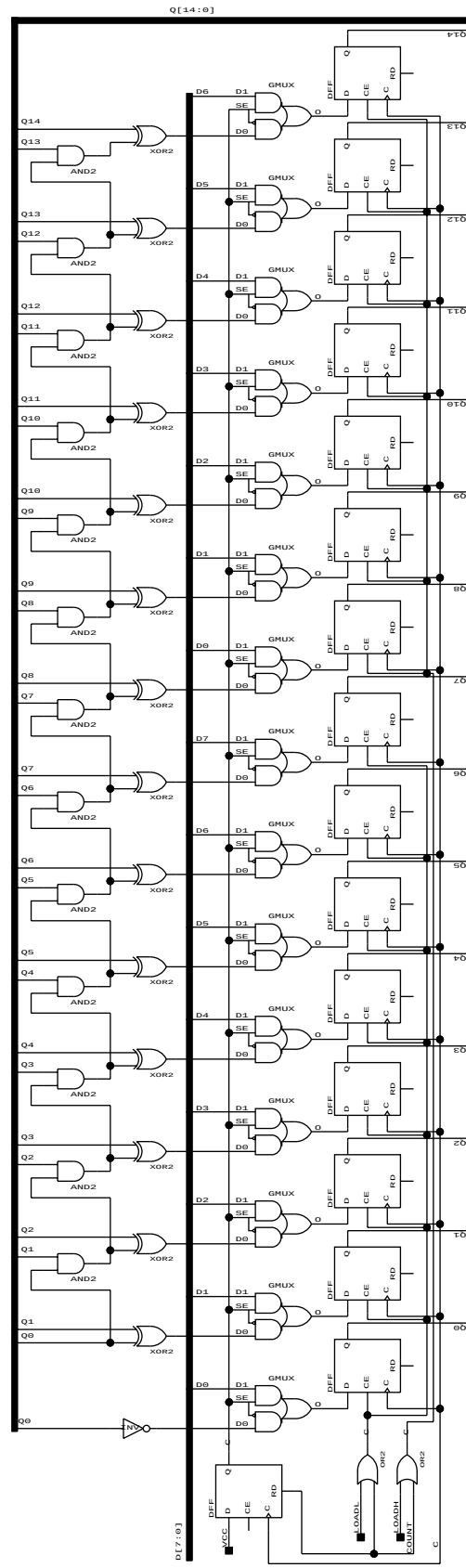


HSYNC
VSYNC
HBLANK
VBLANK
LOC=P67
LOC=P66
OPAD
OBUF
AMux
X-Video-Memory
XA[14:0]
MIN[7:0]
MOUT[7:0]
V_ADRESS[14:0]
W_ADRESS[14:0]
VCounter
ACounter
D3-8E
AND5B2
AND5
AND2B1
NAND2B1
LOC=P26
~OE
IBUF
IPAD
~IOSEL
LOC=P25
A[14:0]
xabus
MuxReg8
XDBUS
ST[7:0]
DIN[7:0]
VShifter
Control
DownCount
DFF
OR2
OR3
AND2
LOC=P27
R/~W
LOC=P13
GCLK
Rot8
PHASE7
PHASE1
PHASE3
PHASE3/4
PHASE0/1/2
AND2B2
GMUX
AND3B2
LOC=P24
LOC=P23
LOC=P20

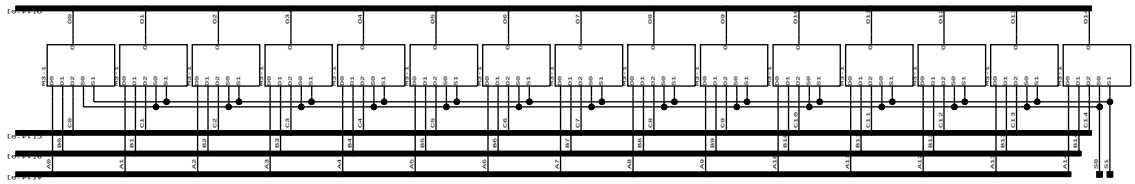
9.2.2. Videozeiger (VCOUNTER)


Q[14:0]
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
Q9
AND2
XOR2
INV
DFF
D
CE
C
RD
Q
COUNT
RESET

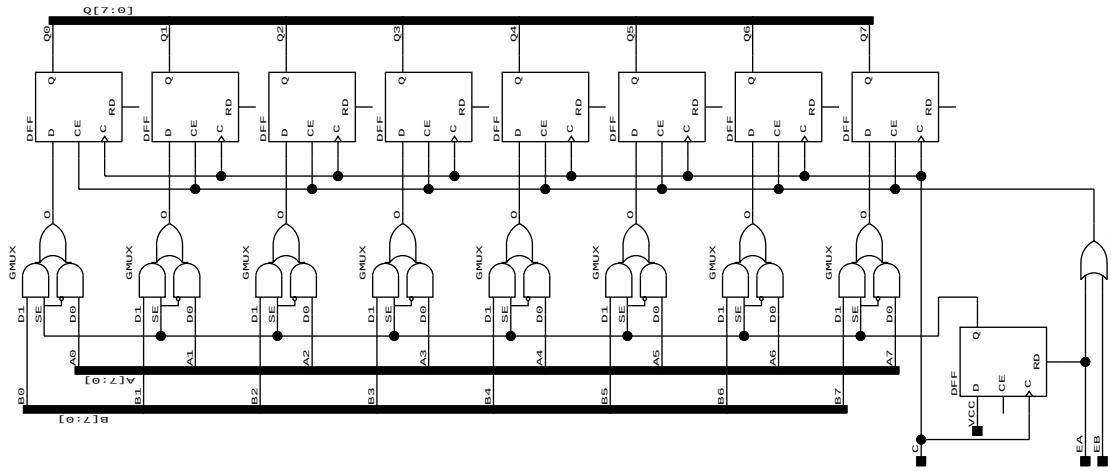
9.2.3. Schreib- und Leseadresszeiger (ACOUNTER)


Q[14:0]
D[7:0]
GMUX
DFF
XOR2
AND2
INV
OR2
OADL
OADH
COUNT
VCC

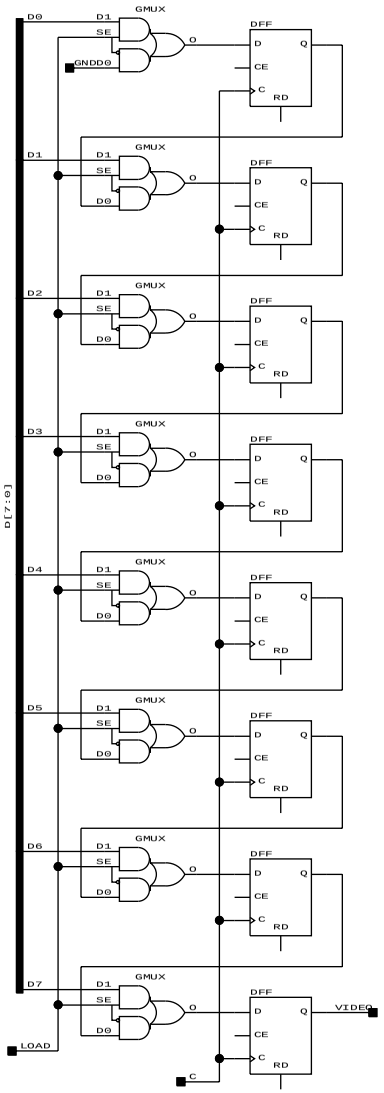
9.2.4. Adress-Multiplexer (AMUX)



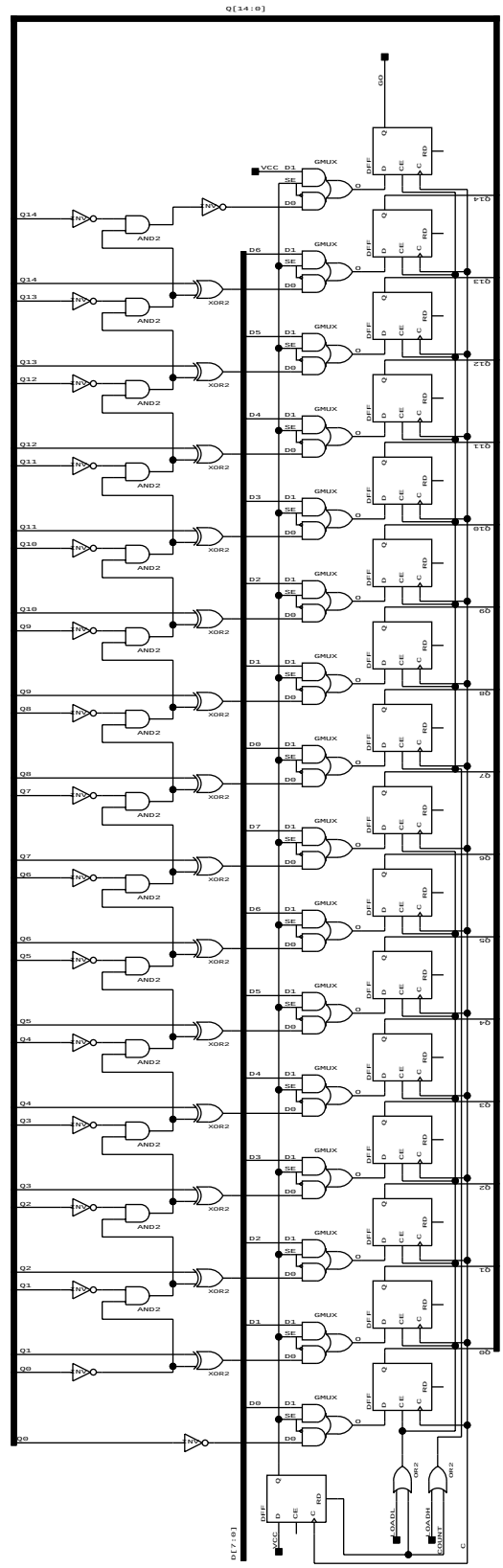
9.2.5. Datenregister (MUXREG8)


Q[7:0]
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
DFF
D
CE
C
RD
Q
GMUX
D1
SE
D0
O
A[7:0]
A0
A1
A2
A3
A4
A5
A6
A7
B[7:0]
B0
B1
B2
B3
B4
B5
B6
B7
VCC
OR2
EA
EB

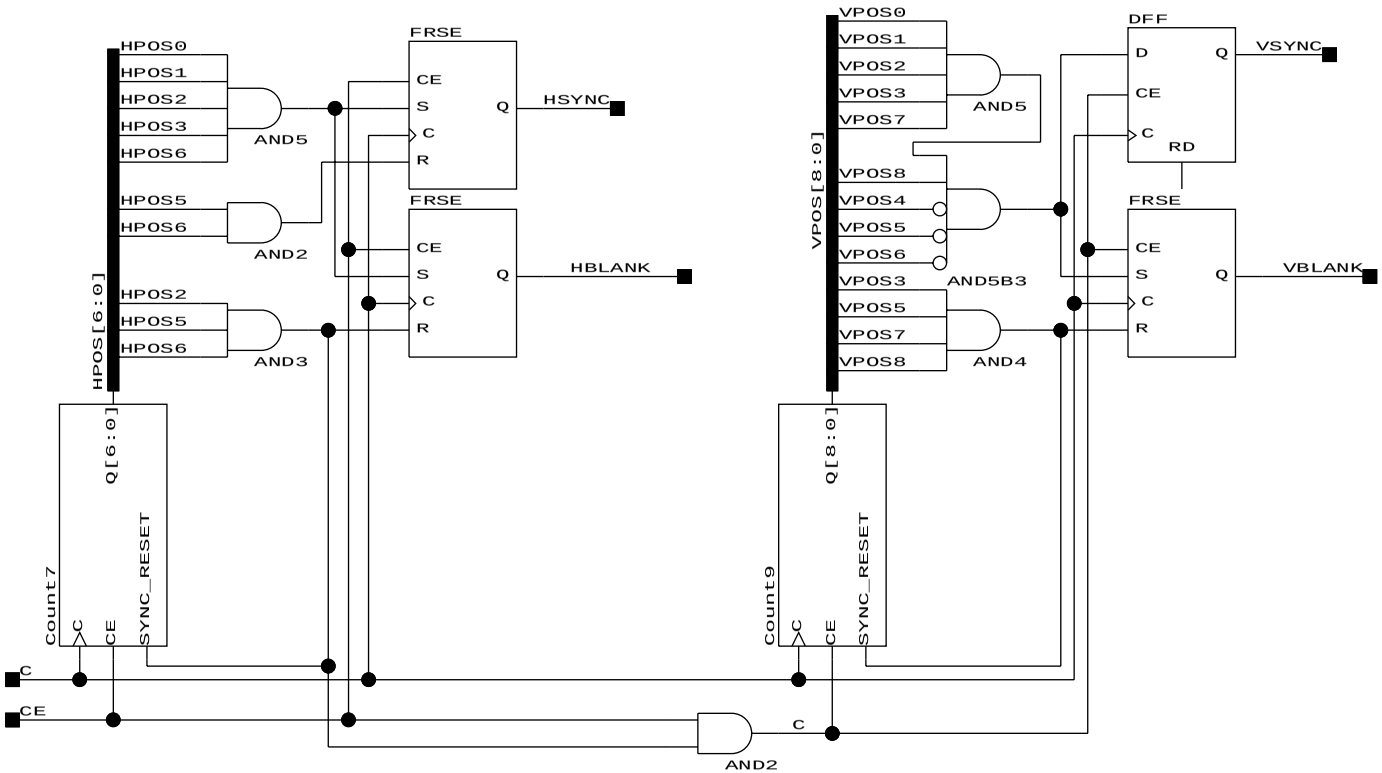
9.2.6. Videoshifter (VSHIFTER)


D[7:0]
D0
D1
D2
D3
D4
D5
D6
D7
GMUX
D1
SE
D0
O
GNDD0
DFF
D
Q
CE
C
RD
LOAD
C
VIDEO

9.2.7. Autocopycounter (DOWNCNT)


Q[14:0]
GO
VCC
D1
SE
D0
GMUX
O
DFF
Q
D
CE
C
RD
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
D6
D5
D4
D3
D2
D1
D0
INV
AND2
XOR2
OR2
OAOH
OAOH
COUNT
D[7:0]

9.2.8. Sync-Signal-Generator (SYNC)


HPOS0
HPOS1
HPOS2
HPOS3
HPOS6
AND5
HPOS5
HPOS6
AND2
HPOS2
HPOS5
HPOS6
AND3
HPOS[6:0]
FRSE
CE
S
C
R
Q
HSYNC
FRSE
CE
S
C
R
Q
HBLANK
Count7
C
CE
SYNC_RESET
Q[6:0]
C
CE
VPOS0
VPOS1
VPOS2
VPOS3
VPOS7
AND5
VPOS8
VPOS4
VPOS5
VPOS6
AND5B3
VPOS3
VPOS5
VPOS7
VPOS8
AND4
VPOS[8:0]
DFF
D
Q
CE
C
RD
VSYNC
FRSE
CE
S
C
R
Q
VBLANK
Count9
C
CE
SYNC_RESET
Q[8:0]
C
CE
C
AND2

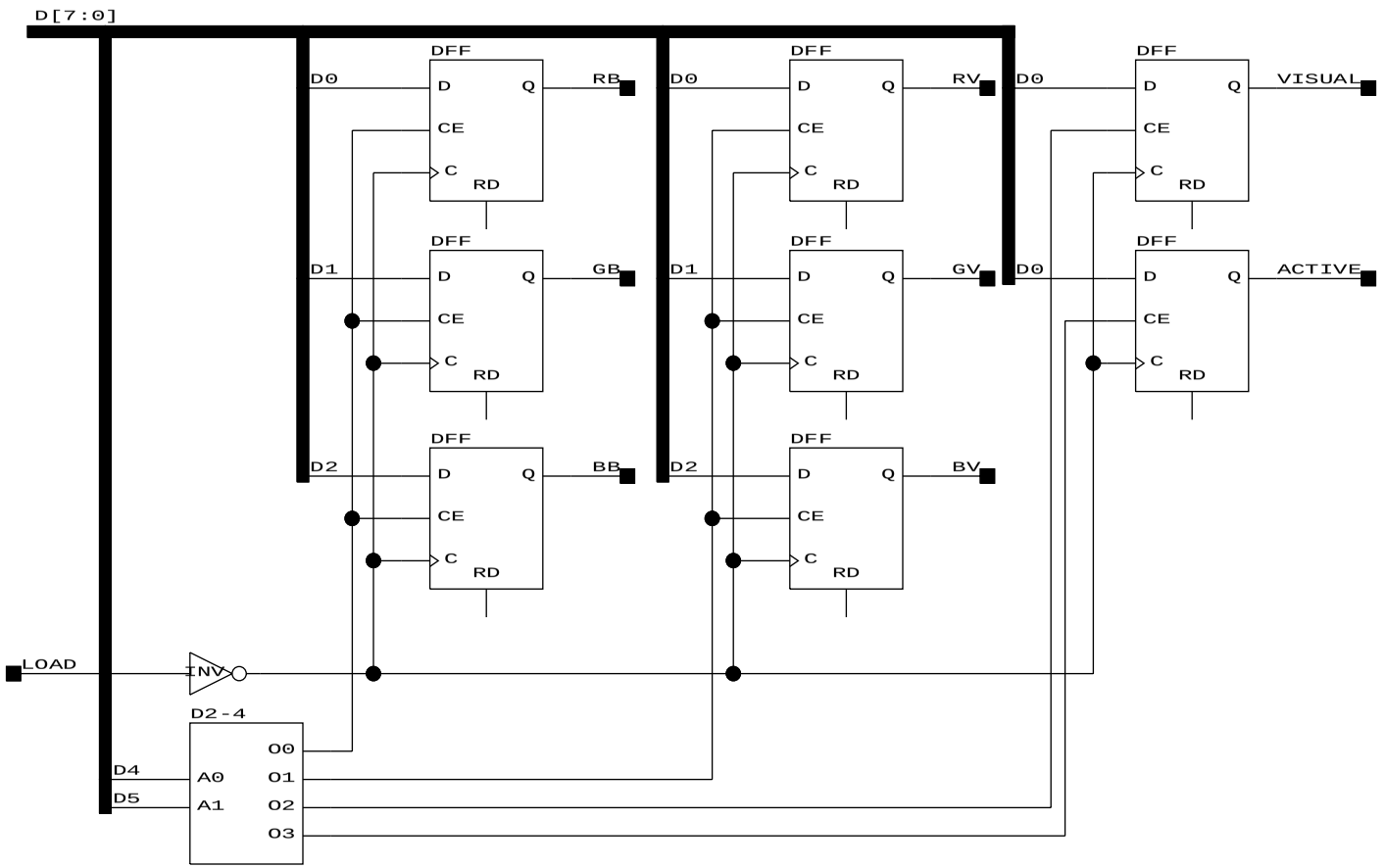
9.2.9. 7-Bit-Zähler (COUNT7)

Q[6:0]
Q6
Q5
Q4
Q3
Q2
Q1
Q0
DFF
Q
D
CE
C
RD
XOR2
AND2
AND2B1
INV
SYNC_RESET
CE
C

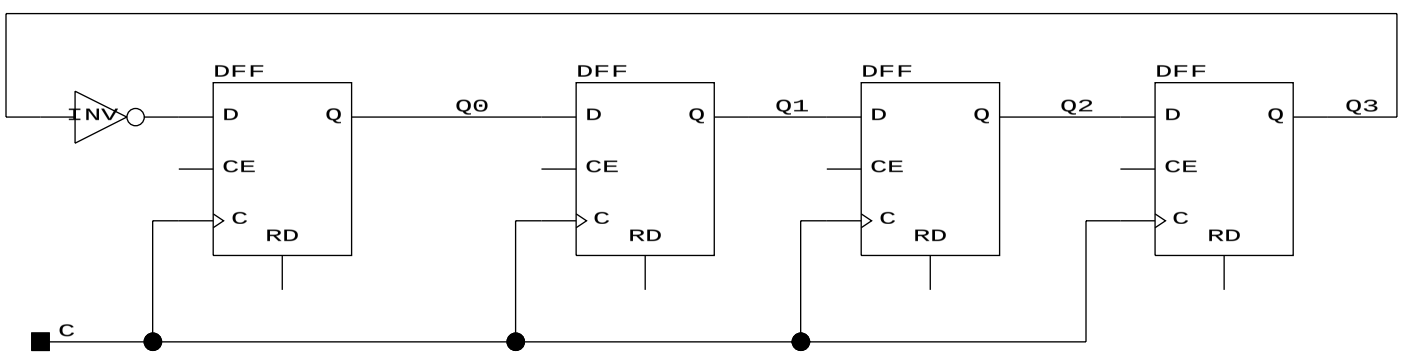
9.2.10. 9-Bit-Zähler (COUNT9)

Q[8:0]
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
DFF
Q
D
CE
C
RD
XOR2
AND2
AND2B1
INV
SYNC_RESET
CE
C

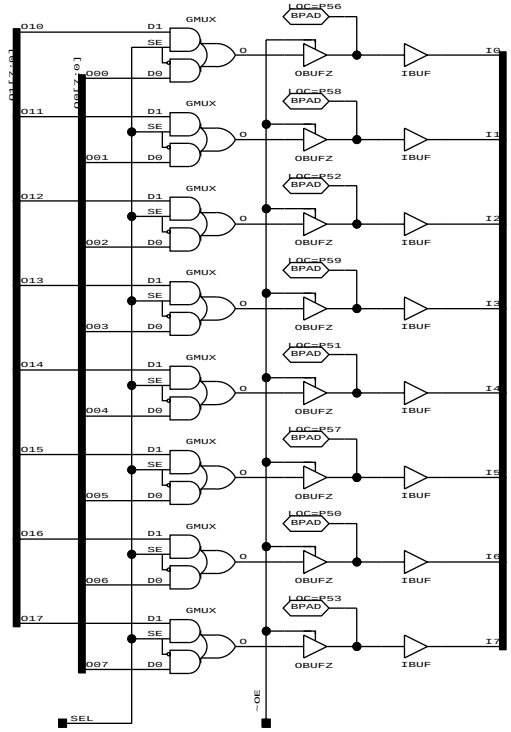
9.2.11. Kontrollregister (CONTROL)


D[7:0]
DFF
D0
D
Q
RB
CE
C
RD
DFF
D1
D
Q
GB
CE
C
RD
DFF
D2
D
Q
BB
CE
C
RD
DFF
D0
D
Q
RV
CE
C
RD
DFF
D1
D
Q
GV
CE
C
RD
DFF
D2
D
Q
BV
CE
C
RD
DFF
D0
D
Q
VISUAL
CE
C
RD
DFF
D0
D
Q
ACTIVE
CE
C
RD
LOAD
INV
D2-4
D4
D5
A0
A1
O0
O1
O2
O3

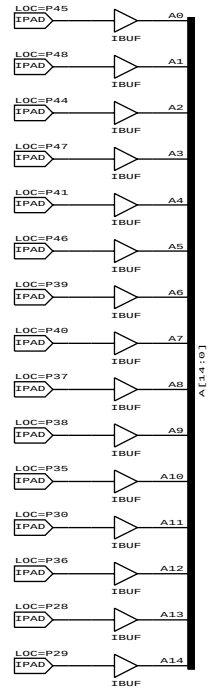
9.2.12. Phasengenerator (ROT8)


DFF
DFF
DFF
DFF
INV
D
Q
CE
C
RD
Q0
Q1
Q2
Q3
C

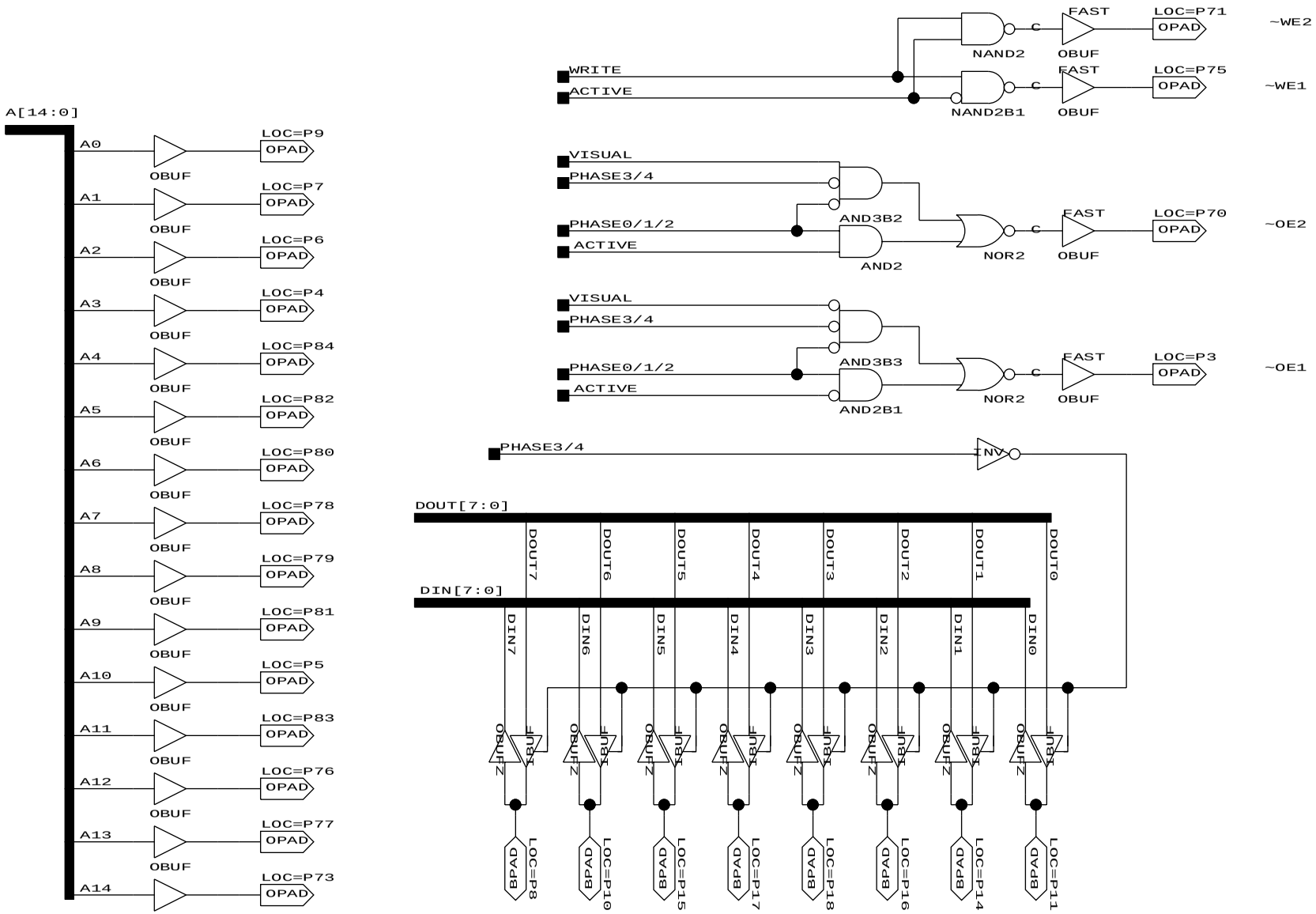
9.2.13. Datenbusschnittstelle (XDBUS)


O[7:0]
O10
O11
O12
O13
O14
O15
O16
O17
O[7:0]
O00
O01
O02
O03
O04
O05
O06
O07
D1
SE
D0
GMUX
O
LOC=P56
LOC=P58
LOC=P52
LOC=P59
LOC=P51
LOC=P57
LOC=P50
LOC=P53
BPAD
OBUFZ
IBUF
I6
I1
I2
I3
I4
I5
I7
I[7:0]
SEL
~OE

9.2.14. Adressbusschnittstelle (XABUS)


LOC=P45
IPAD
IBUF
A0
LOC=P48
IPAD
IBUF
A1
LOC=P44
IPAD
IBUF
A2
LOC=P47
IPAD
IBUF
A3
LOC=P41
IPAD
IBUF
A4
LOC=P46
IPAD
IBUF
A5
LOC=P39
IPAD
IBUF
A6
LOC=P40
IPAD
IBUF
A7
LOC=P37
IPAD
IBUF
A8
LOC=P38
IPAD
IBUF
A9
LOC=P35
IPAD
IBUF
A10
LOC=P30
IPAD
IBUF
A11
LOC=P36
IPAD
IBUF
A12
LOC=P28
IPAD
IBUF
A13
LOC=P29
IPAD
IBUF
A14
A[14:0]

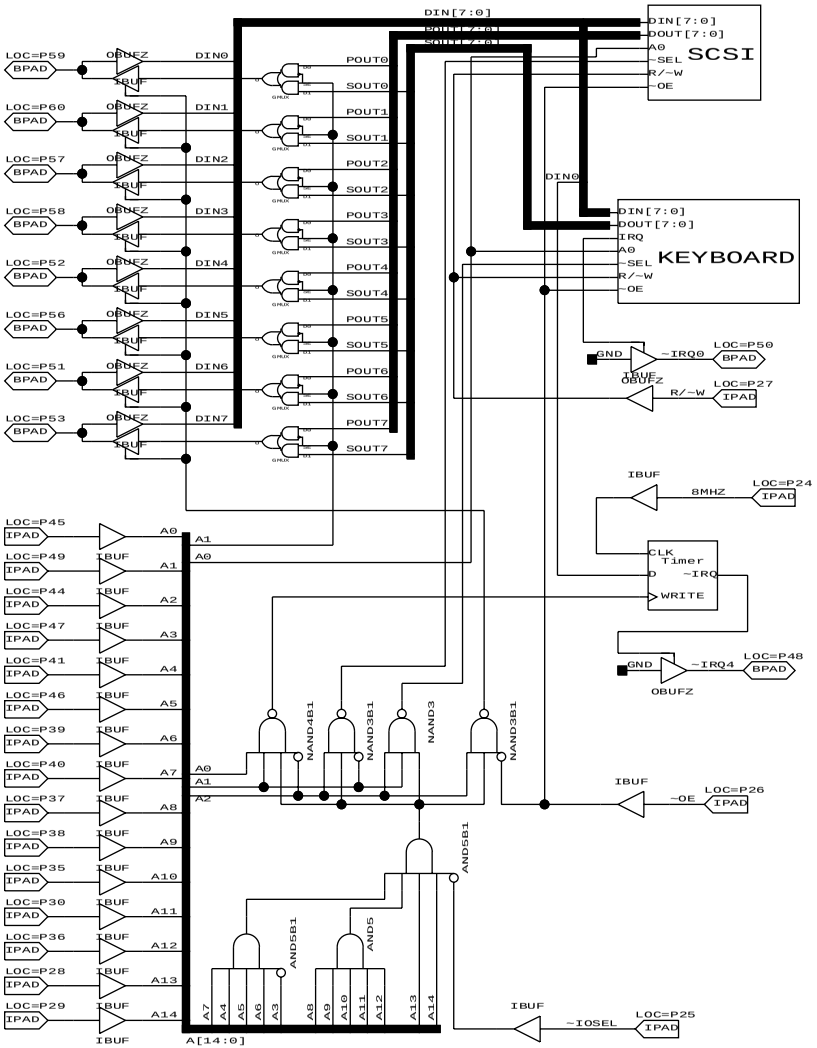
9.2.15. Bildspeicherschnittstelle (XV_MEM)


A[14:0]
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
OBUF
LOC=P9
LOC=P7
LOC=P6
LOC=P4
LOC=P84
LOC=P82
LOC=P80
LOC=P78
LOC=P79
LOC=P81
LOC=P5
LOC=P83
LOC=P76
LOC=P77
LOC=P73
OPAD
WRITE
ACTIVE
NAND2
NAND2B1
FAST
C
LOC=P71
LOC=P75
~WE2
~WE1
VISUAL
PHASE3/4
PHASE0/1/2
AND3B2
AND2
NOR2
LOC=P70
~OE2
AND3B3
AND2B1
LOC=P3
~OE1
INV
DOUT[7:0]
DIN[7:0]
DOUT7
DOUT6
DOUT5
DOUT4
DOUT3
DOUT2
DOUT1
DOUT0
DIN7
DIN6
DIN5
DIN4
DIN3
DIN2
DIN1
DIN0
OBUFT
I
T
O
BPAD
LOC=P8
LOC=P10
LOC=P15
LOC=P17
LOC=P18
LOC=P16
LOC=P14
LOC=P11

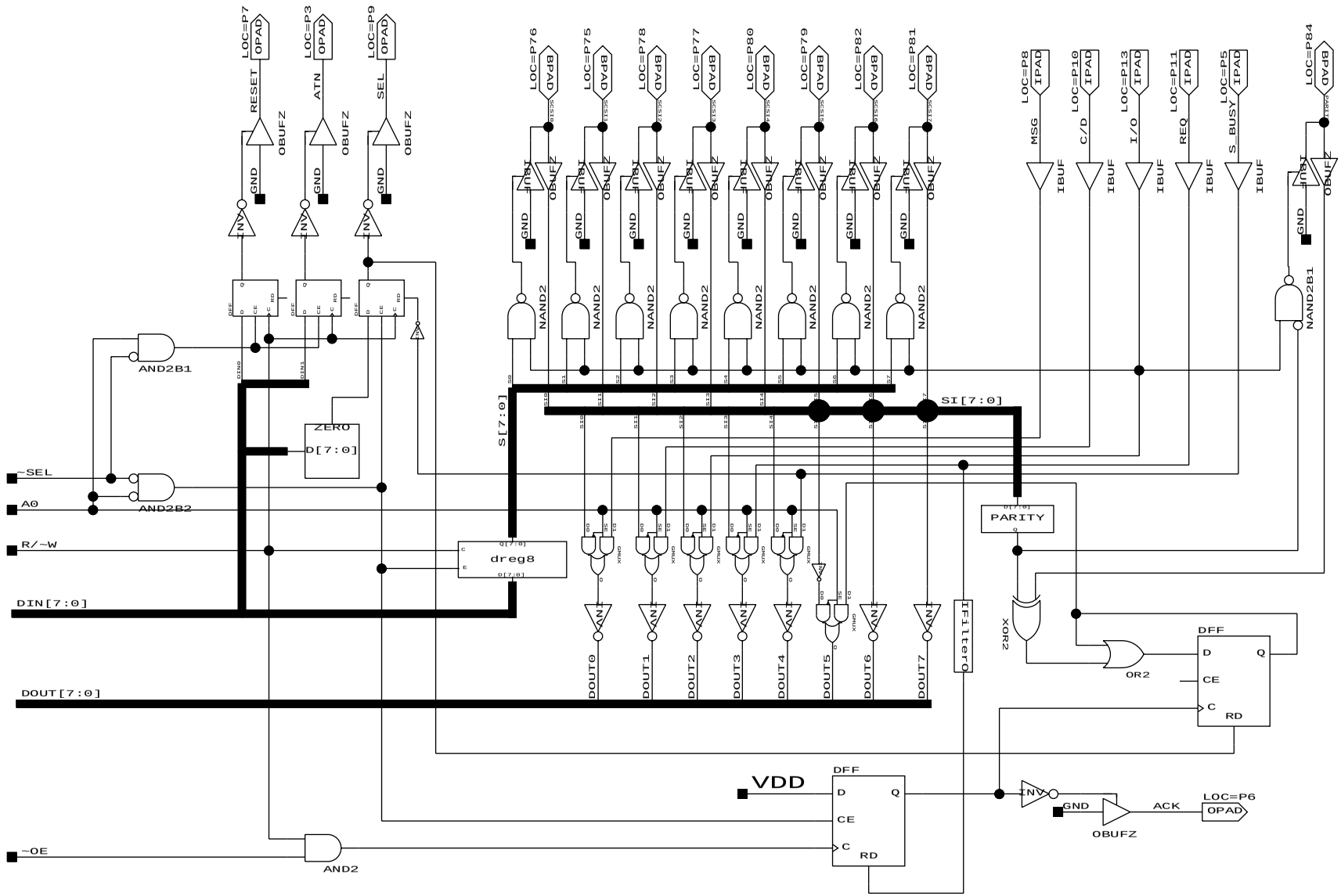
9.3. Die SCSI-, Keyboard- und Timer-Karte

9.3.1. Hauptschaltung (SCSI_KEY)

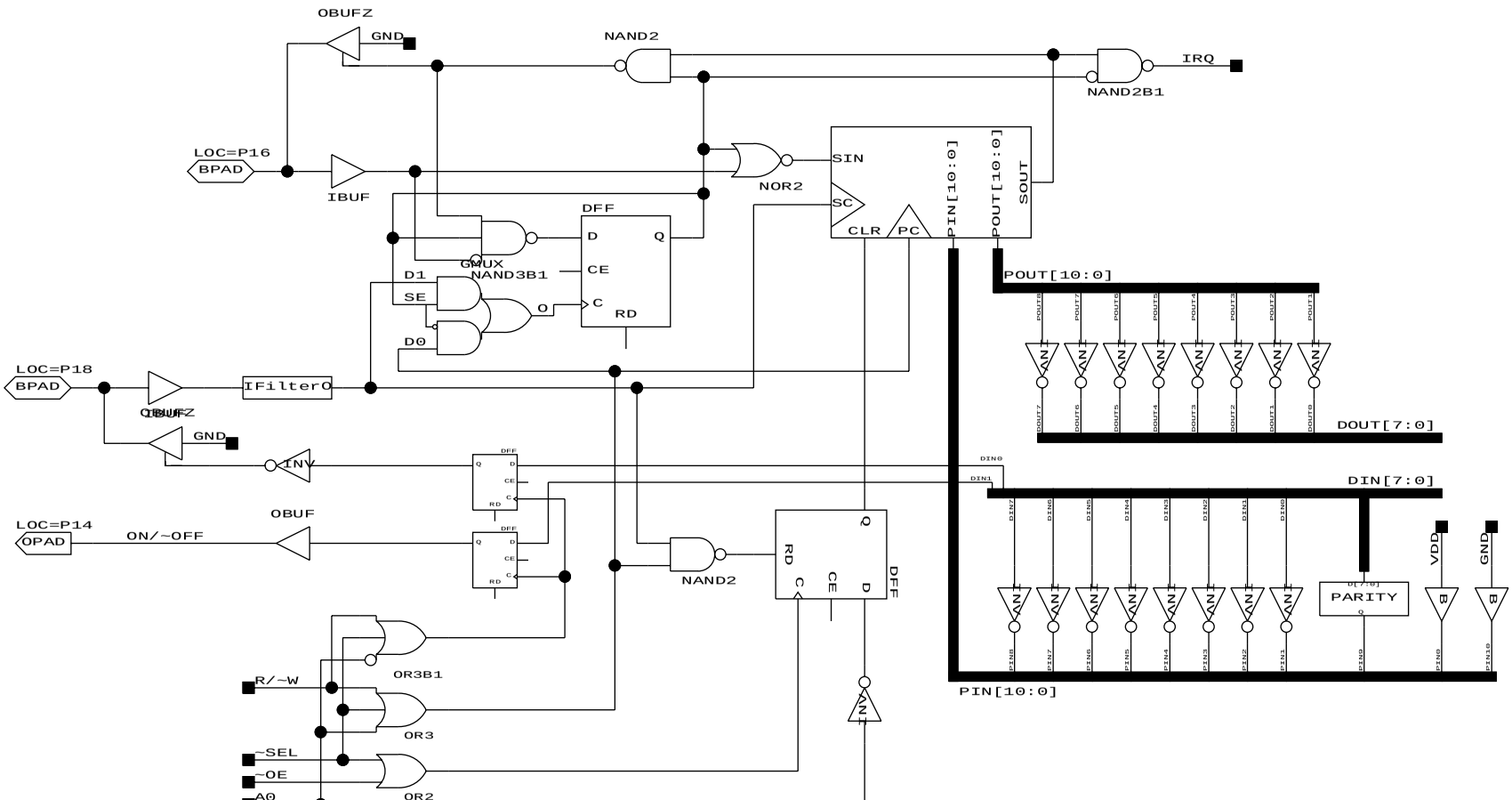
PART=3042PC84-70



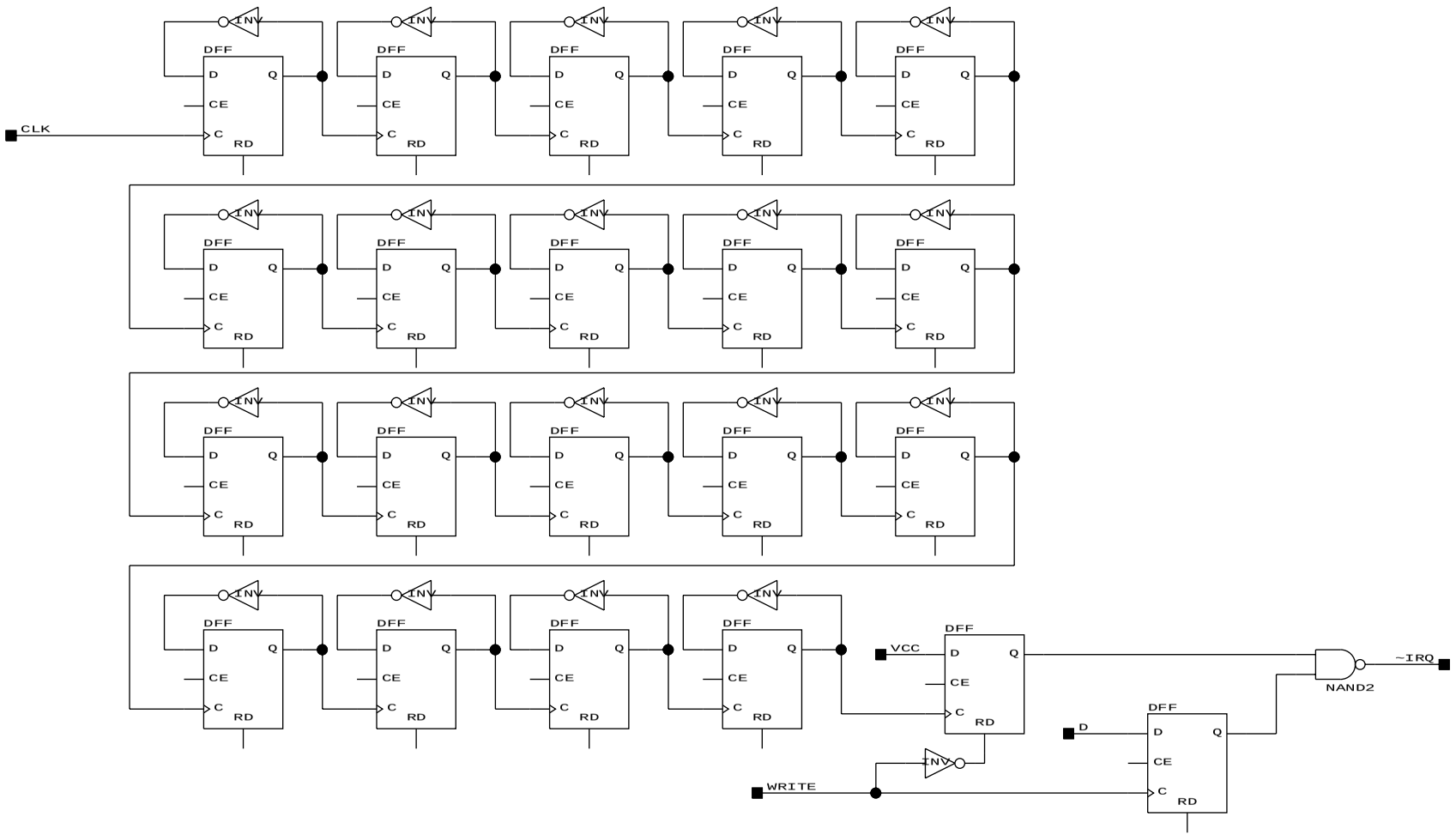
9.3.2. SCSI-Schnittstelle (SCSI)


LOC=P7
OPAD
RESET
OBUFZ
GND
INV
LOC=P3
ATN
LOC=P9
SEL
LOC=P76
LOC=P75
LOC=P78
LOC=P77
LOC=P80
LOC=P79
LOC=P82
LOC=P81
BPAD
IMUX
NAND2
LOC=P8
IPAD
MSG
IBUF
LOC=P10
C/D
LOC=P13
I/O
LOC=P11
REQ
LOC=P5
S_BUSY
LOC=P84
NAND2B1
DFF
AND2B1
AND2B2
ZERO
D[7:0]
S[7:0]
SI[7:0]
~SEL
A0
R/~W
DIN[7:0]
dreg8
PARITY
XOR2
OR2
CE
C
RD
DOUT[7:0]
DOUT0
DOUT1
DOUT2
DOUT3
DOUT4
DOUT5
DOUT6
DOUT7
rfilter0
VDD
~OE
AND2
ACK
LOC=P6

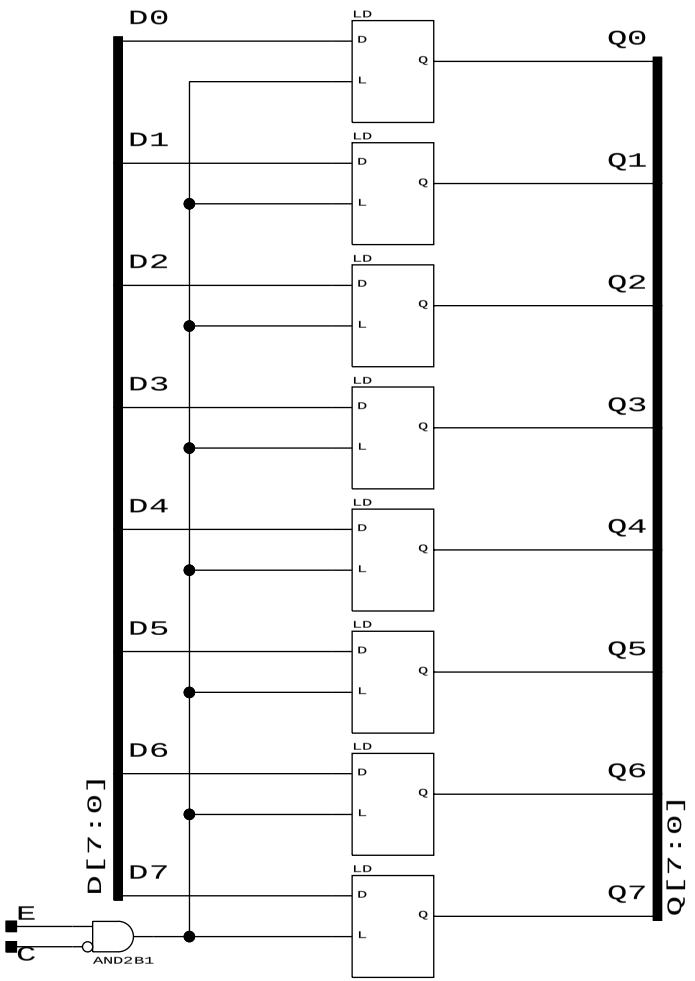
9.3.3. Keyboard-Schnittstelle (KEYBOARD)


OBUFZ
GND
NAND2
IRQ
NAND2B1
LOC=P16
BPAD
IBUF
NOR2
SIN
SC
CLR
PC
PIN[10:0]
POUT[10:0]
SOUT
DFF
D
Q
CE
C
RD
GMUX
NAND3B1
D1
SE
O
D0
POUT[10:0]
DOUT[7:0]
LOC=P18
BPAD
IFilterO
OBUFZ
GND
INV
DIN0
DIN1
DIN[7:0]
LOC=P14
OPAD
ON/~OFF
OBUF
NAND2
PARITY
VDD
GND
R/~W
OR3B1
OR3
~SEL
~OE
A0
OR2
PIN[10:0]

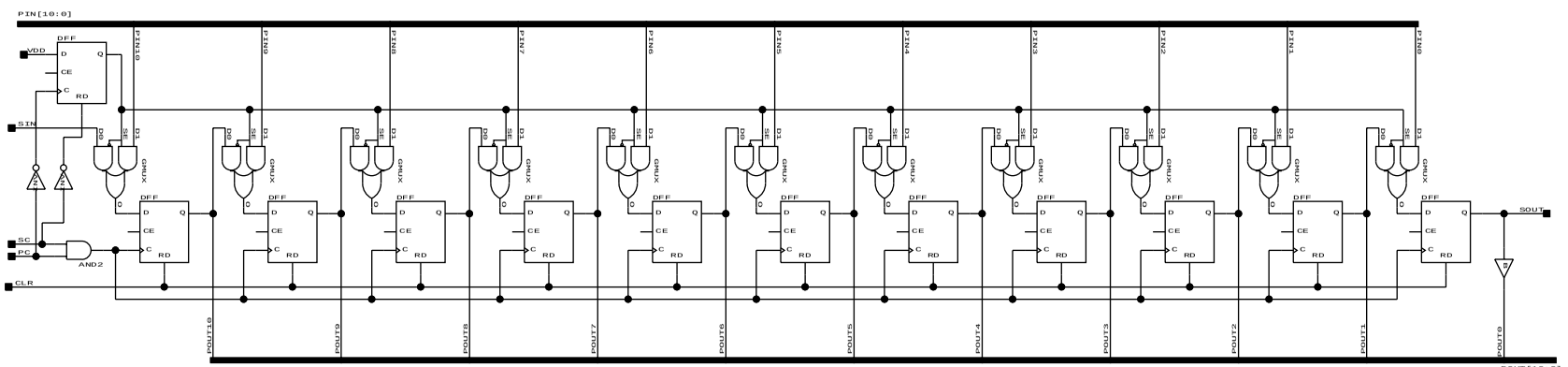
9.3.4. Timer (TIMER)


CLK
INV
DFF
D
Q
CE
C
RD
VCC
WRITE
D
NAND2
~IRQ

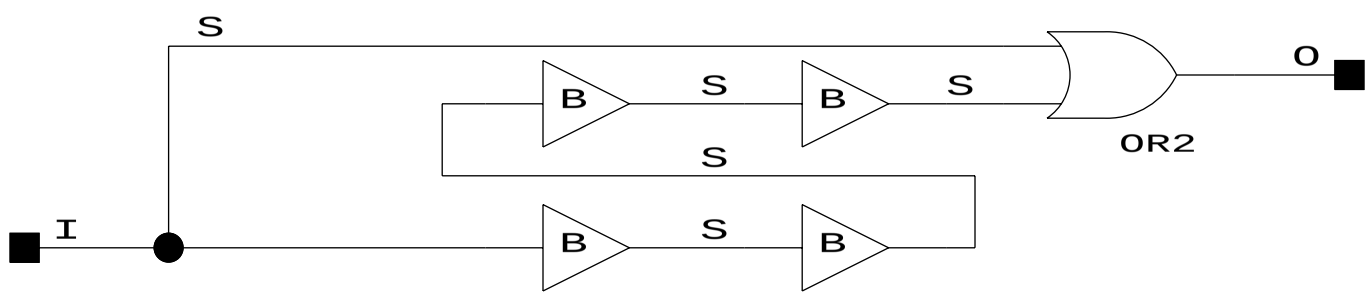
9.3.5. 8-Bit-Latch (DREG8)

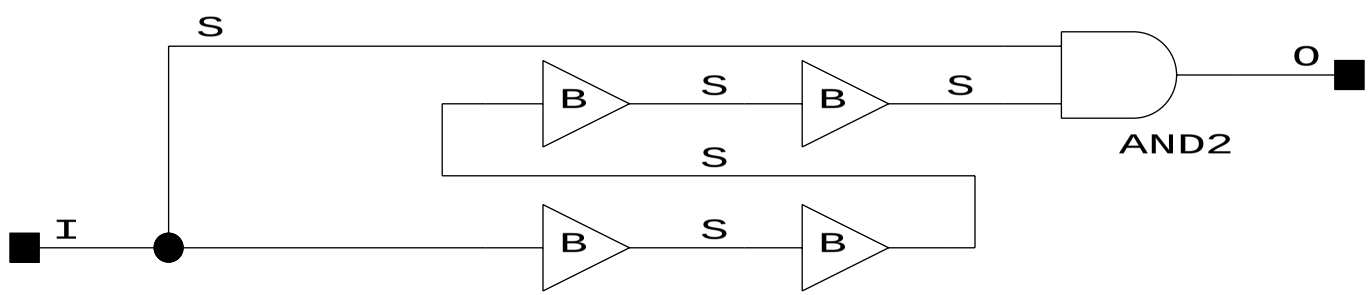

D0
D1
D2
D3
D4
D5
D6
D7
D[7:0]
LD
D
L
Q
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q[7:0]
E
C
AND2B1

9.3.6. 11-Bit-Schieberegister (SHIFT11)

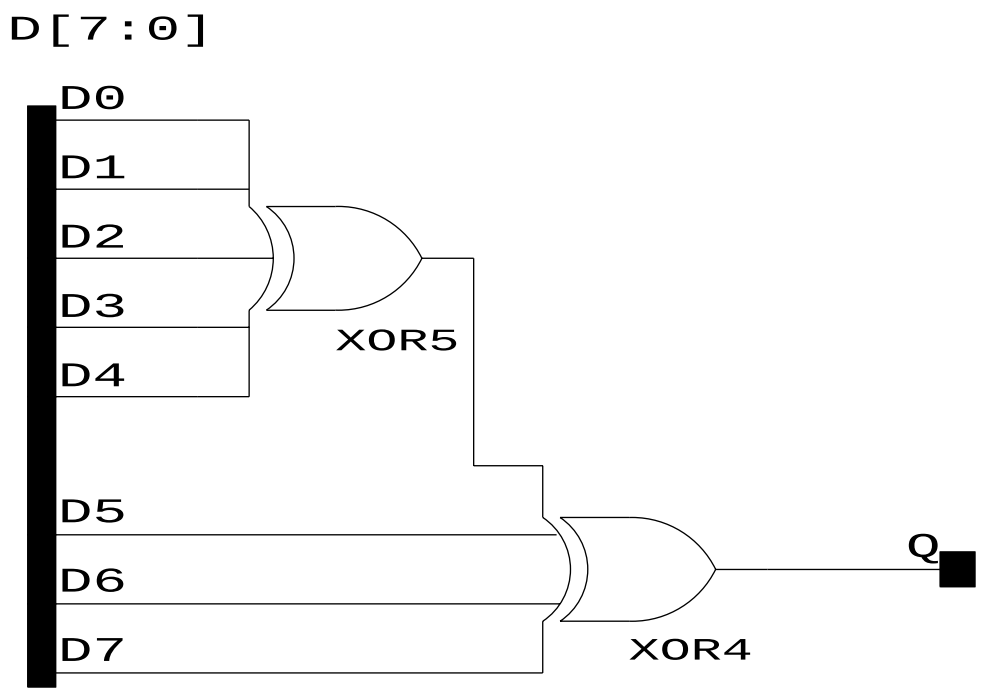

PIN[10:0]
VDD
SIN
SC
PC
CLR
AND2
DFF
GMUX
SOUT
POUT[10:0]

9.3.7. Impulsfilter (FILTER und FILTER2)

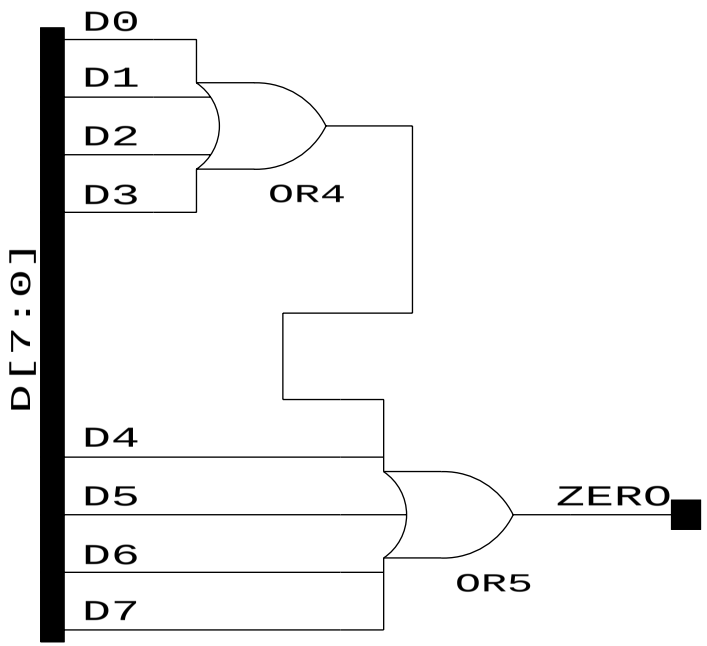

S
O
B
S
B
S
OR2
S
I
B
S
B


S
O
B
S
B
S
AND2
S
I
B
S
B

9.3.8. Parity-Generator (PARITY)


D[7:0]
D0
D1
D2
D3
D4
XOR5
D5
D6
D7
XOR4
Q

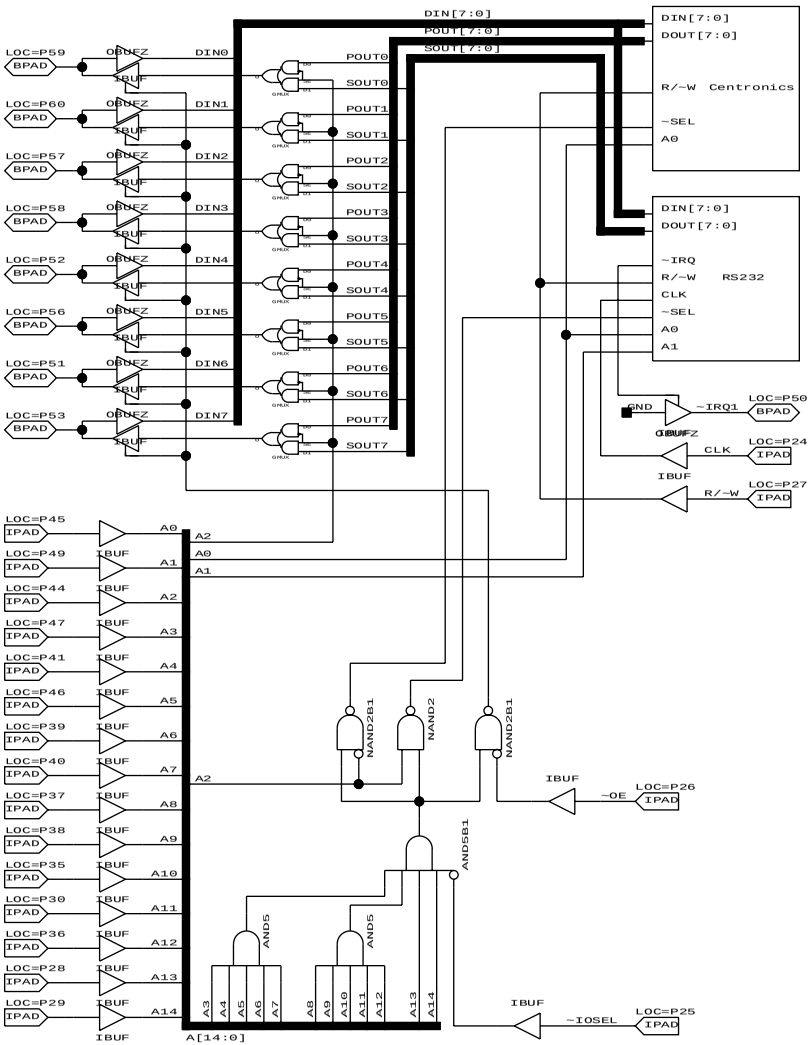
9.3.9. Nullbyteerkennung (ZERO)


D0
D1
D2
D3
OR4
D[7:0]
D4
D5
D6
D7
ZERO
OR5

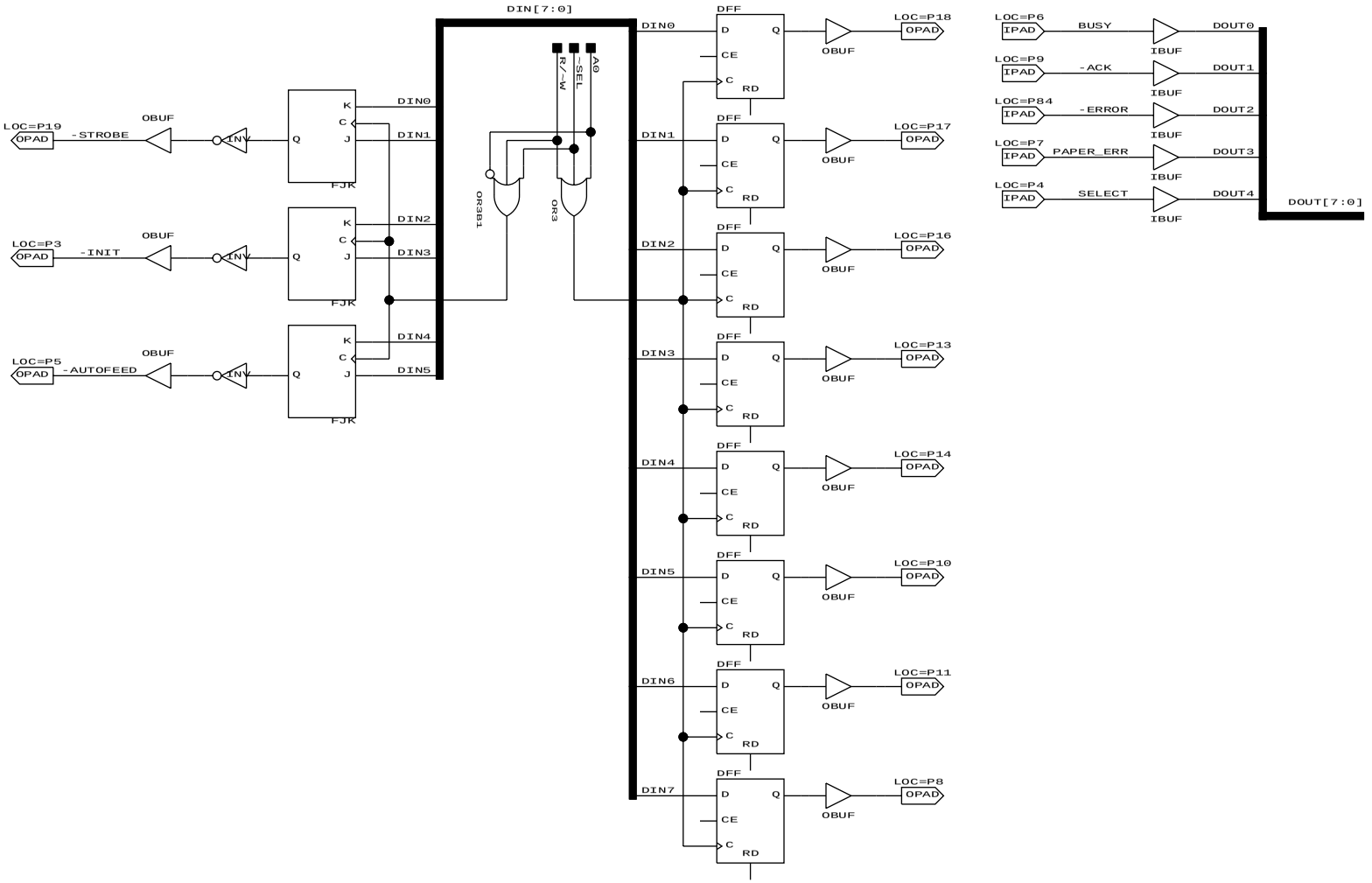
9.4. Die Centronics- und RS232-Karte

9.4.1. Hauptschaltung (SER_PAR)

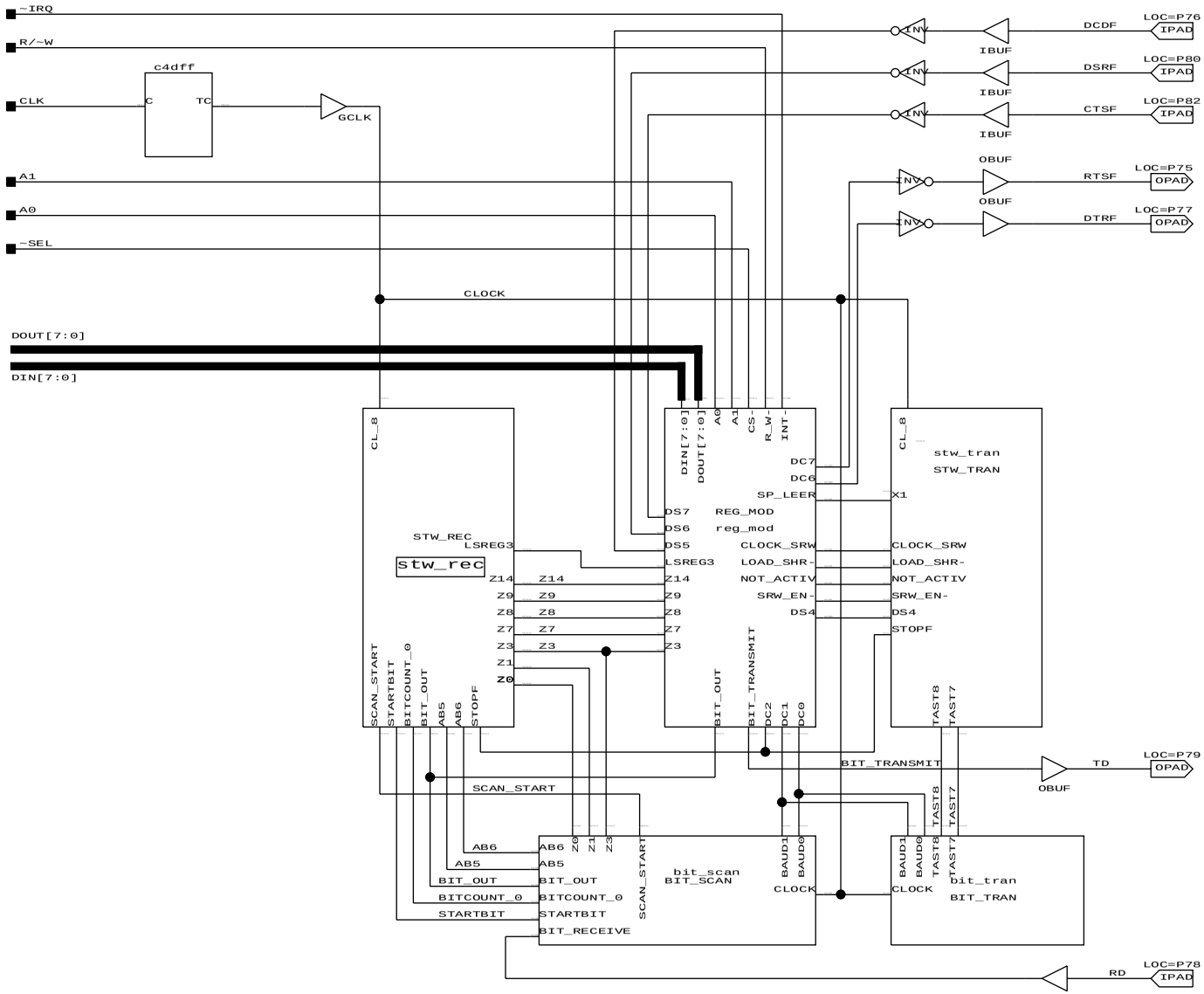
PART=3064PC84-70



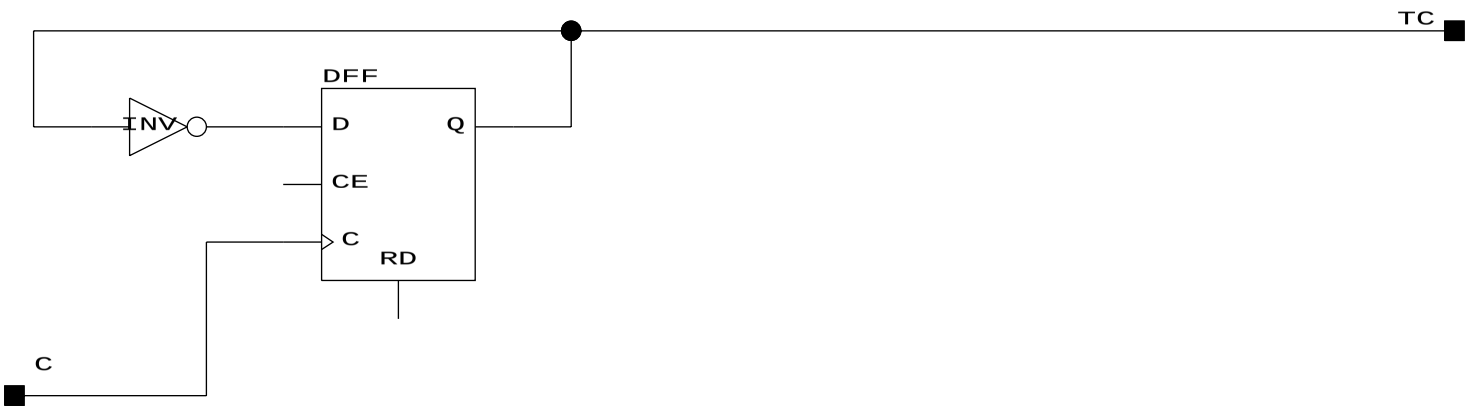
9.4.2. Parallelport (CENTRONI)


DIN[7:0]
R/-W
-SEL
A0
OR3B1
OR3
LOC=P19
OPAD
-STROBE
OBUF
INV
LOC=P3
-INIT
LOC=P5
-AUTOFEED
FJK
K
C
J
Q
DIN0
DIN1
DIN2
DIN3
DIN4
DIN5
DIN6
DIN7
DFF
D
CE
RD
LOC=P18
LOC=P17
LOC=P16
LOC=P13
LOC=P14
LOC=P10
LOC=P11
LOC=P8
LOC=P6
IPAD
BUSY
IBUF
DOUT0
LOC=P9
-ACK
DOUT1
LOC=P84
-ERROR
DOUT2
LOC=P7
PAPER_ERR
DOUT3
LOC=P4
SELECT
DOUT4
DOUT[7:0]

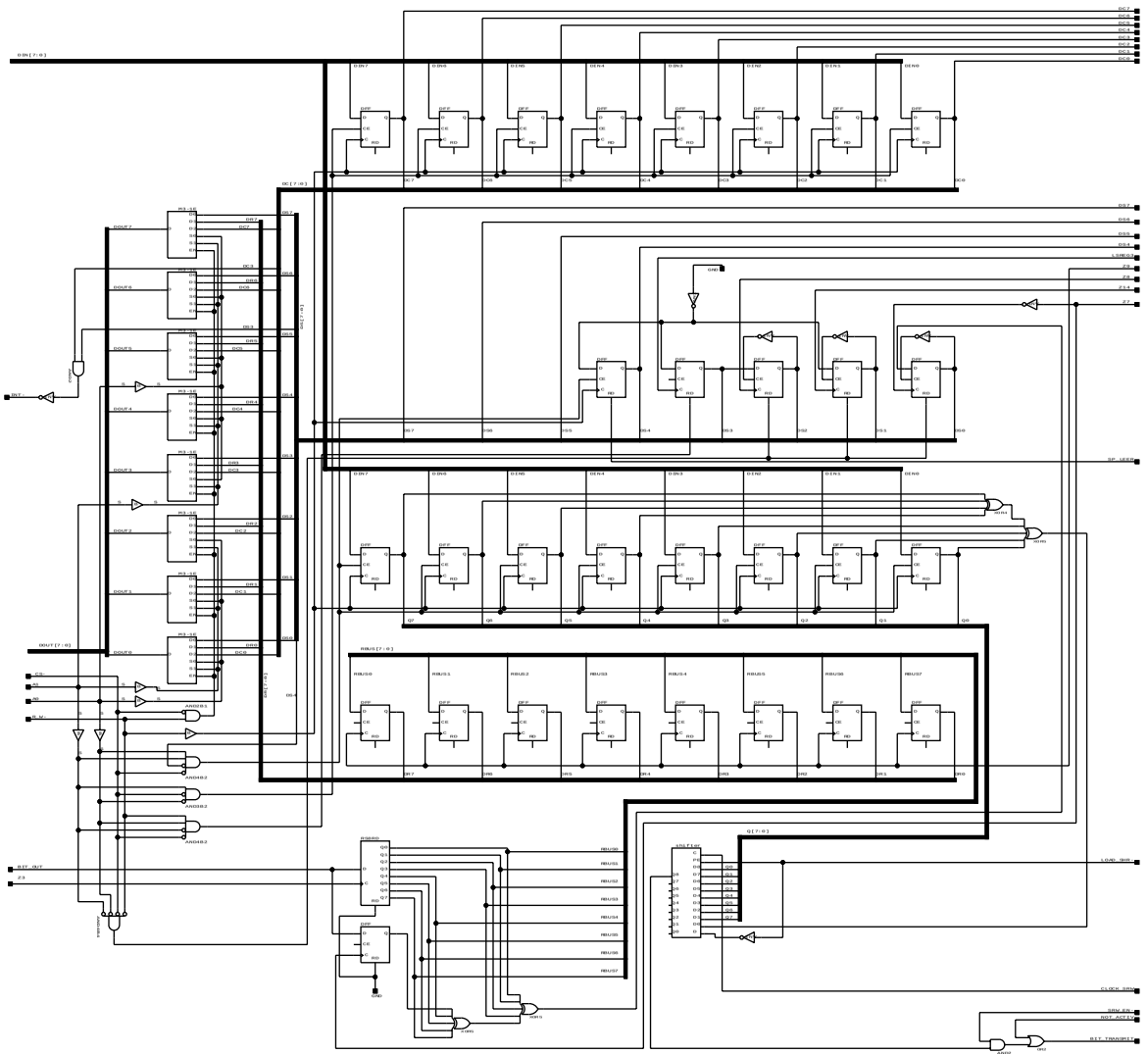
9.4.3. Hauptschaltung der seriellen Schnittstelle (RS232)


~IRQ
R/~W
CLK
c4dff
C
TC
GCLK
A1
A0
~SEL
CLOCK
DOUT[7:0]
DIN[7:0]
INV
IBUF
DCDF
LOC=P76
IPAD
DSRF
LOC=P80
CTSF
LOC=P82
OBUF
RTSF
LOC=P75
OPAD
DTRF
LOC=P77
CL_8
DIN[7:0]
DOUT[7:0]
A0
A1
CS-
R_W-
INT-
DC7
DC6
SP_LEER
DS7
REG_MOD
DS6
reg_mod
DS5
CLOCK_SRW
LOAD_SHR-
NOT_ACTIV
SRW_EN-
DS4
STOPF
X1
stw_tran
STW_TRAN
STW_REC
LSREG3
stw_rec
Z14
Z9
Z8
Z7
Z3
Z1
Z0
SCAN_START
STARTBIT
BITCOUNT_0
BIT_OUT
AB5
AB6
STOPF
BIT_TRANSMIT
DC2
DC1
DC0
TAST8
TAST7
TD
RD
bit_scan
BIT_SCAN
BAUD1
BAUD0
CLOCK
bit_tran
BIT_TRAN
BITCOUNT_0
BIT_RECEIVE

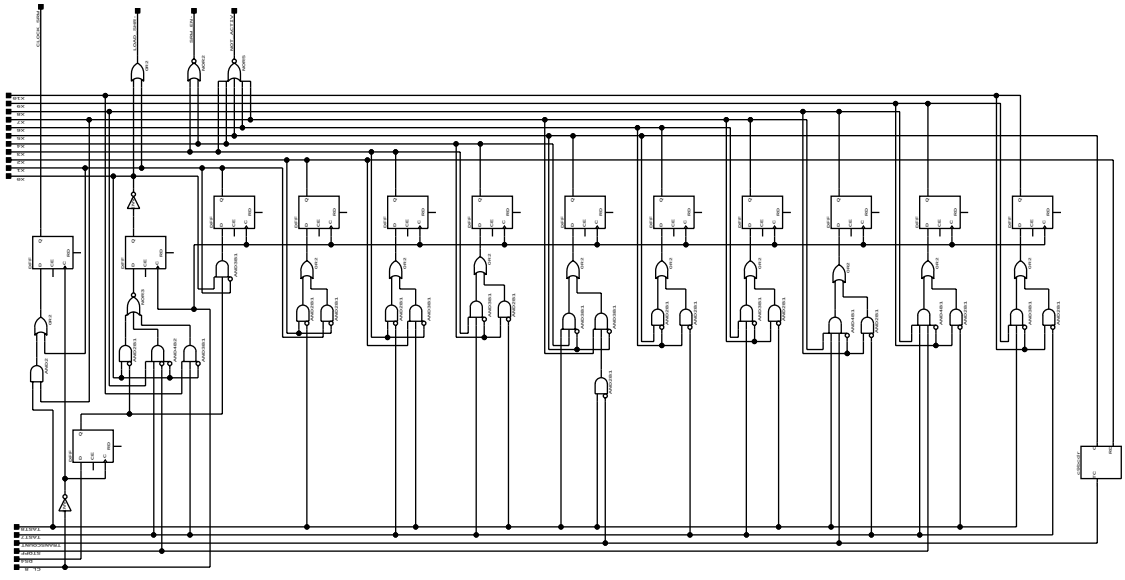
9.4.4. Takthalbierer (C4DFF)


TC
DFF
INV
D
Q
CE
C
RD
C

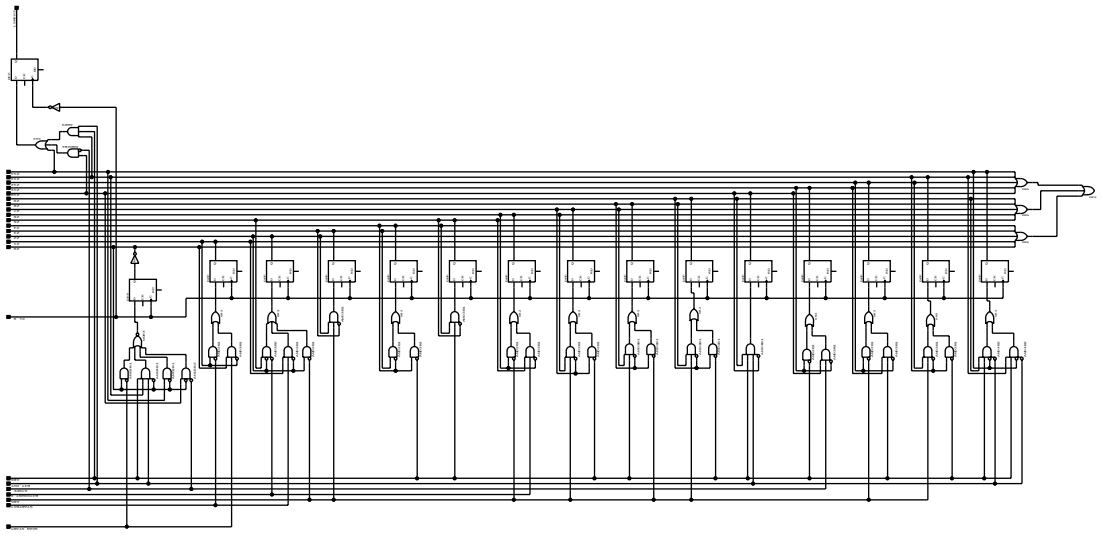
9.4.4. Registersatz (REG_MOD)



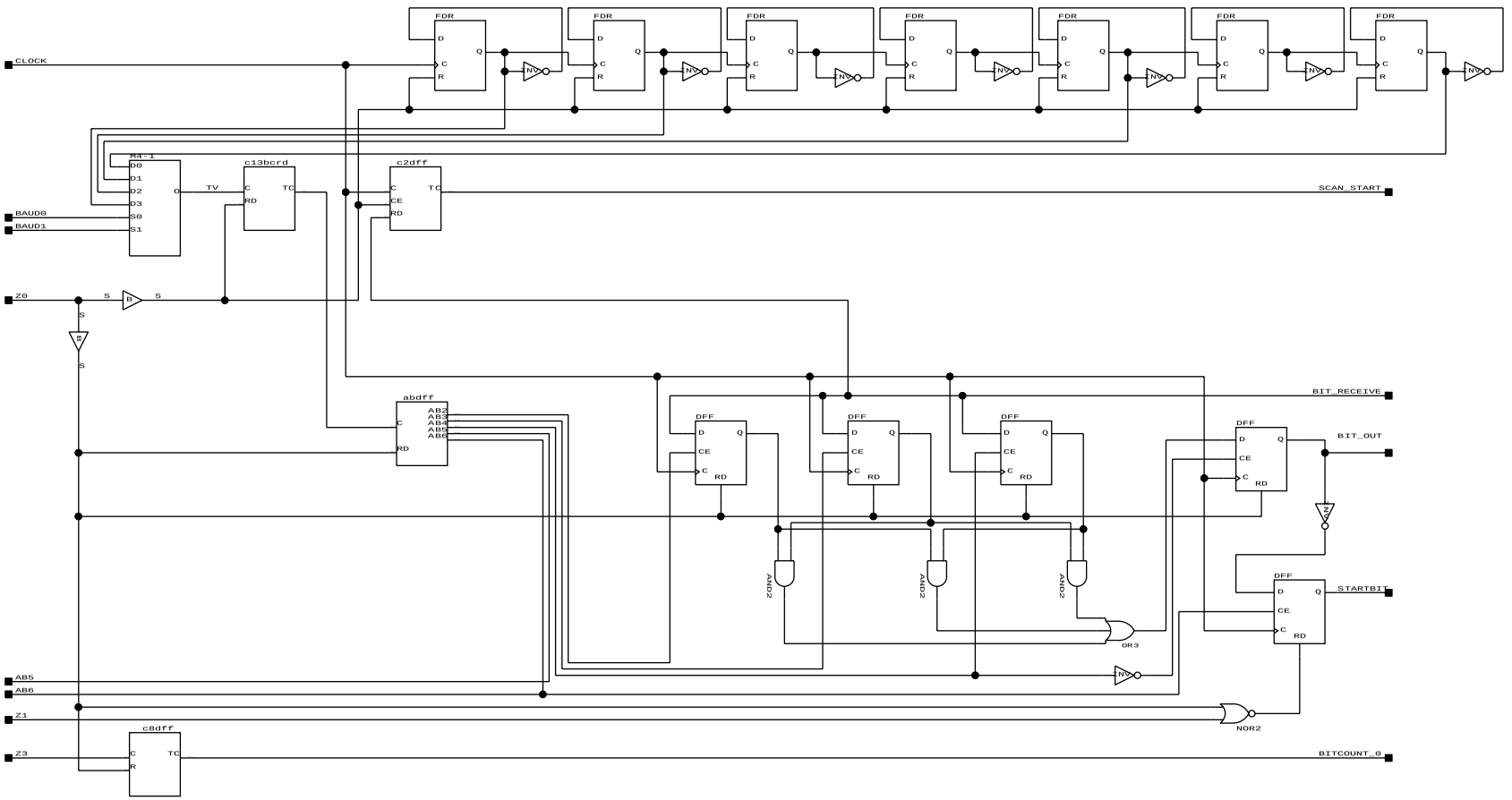
9.4.5. Steuerwerk Senden (STW_TRAN)



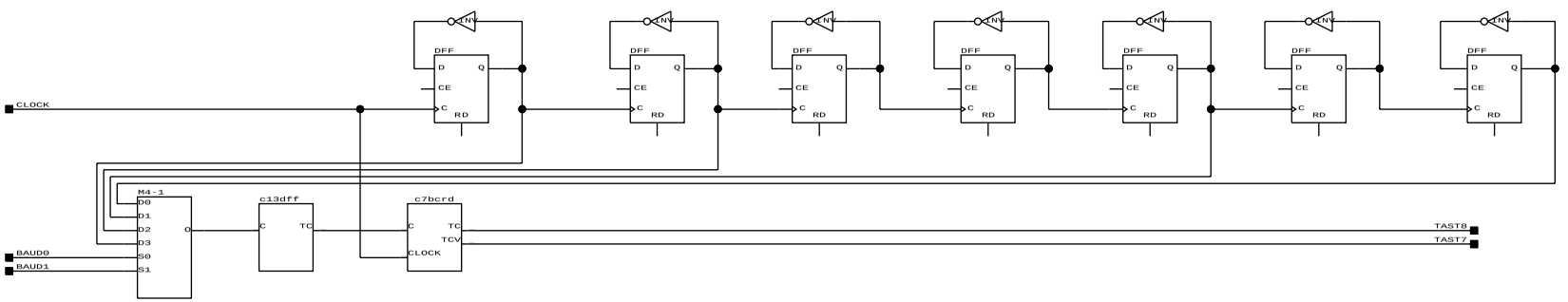
9.4.6. Steuerwerk Empfangen (STW_REC)



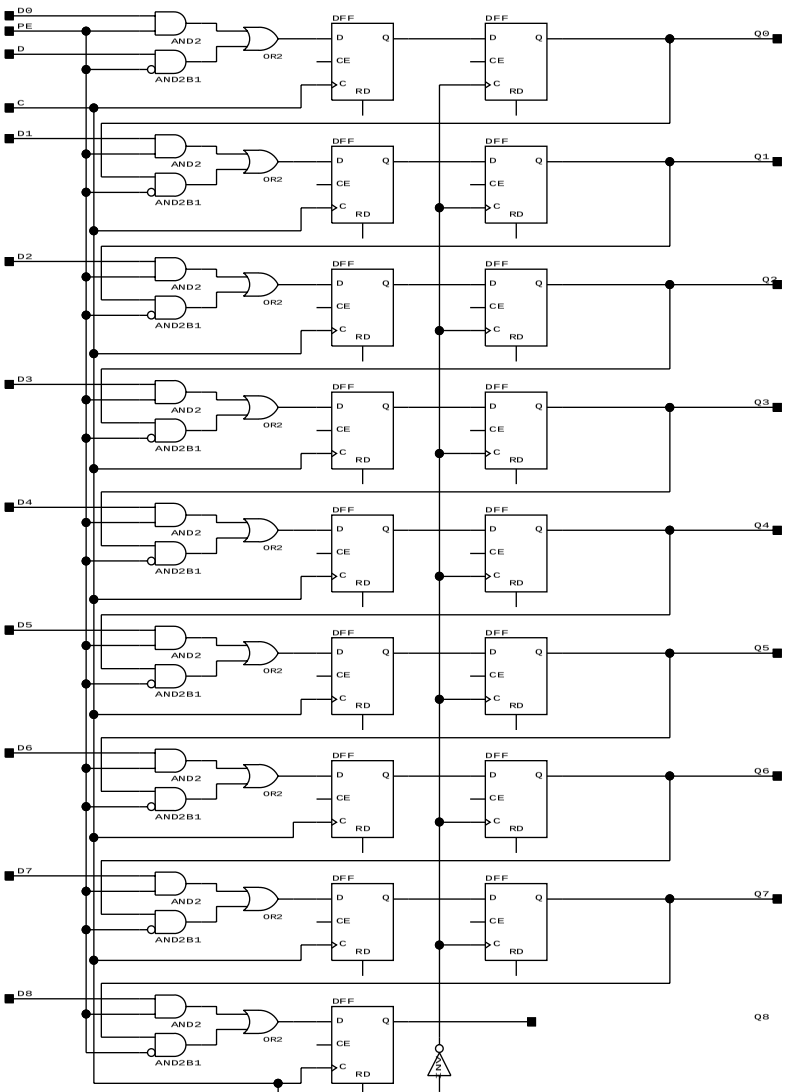
9.4.7. Empfangseinheit (BIT_SCAN)


CLOCK
FDR
D
Q
C
R
INV
M4_1
D0
D1
D2
D3
S0
S1
O
BAUD0
BAUD1
c13bcrd
TV
TC
RD
c2dff
CE
SCAN_START
Z0
S
B
abdff
AB2
AB3
AB4
AB5
AB6
DFF
BIT_RECEIVE
BIT_OUT
STARTBIT
AND2
OR3
NOR2
AB5
AB6
Z1
Z3
c8dff
BITCOUNT_0

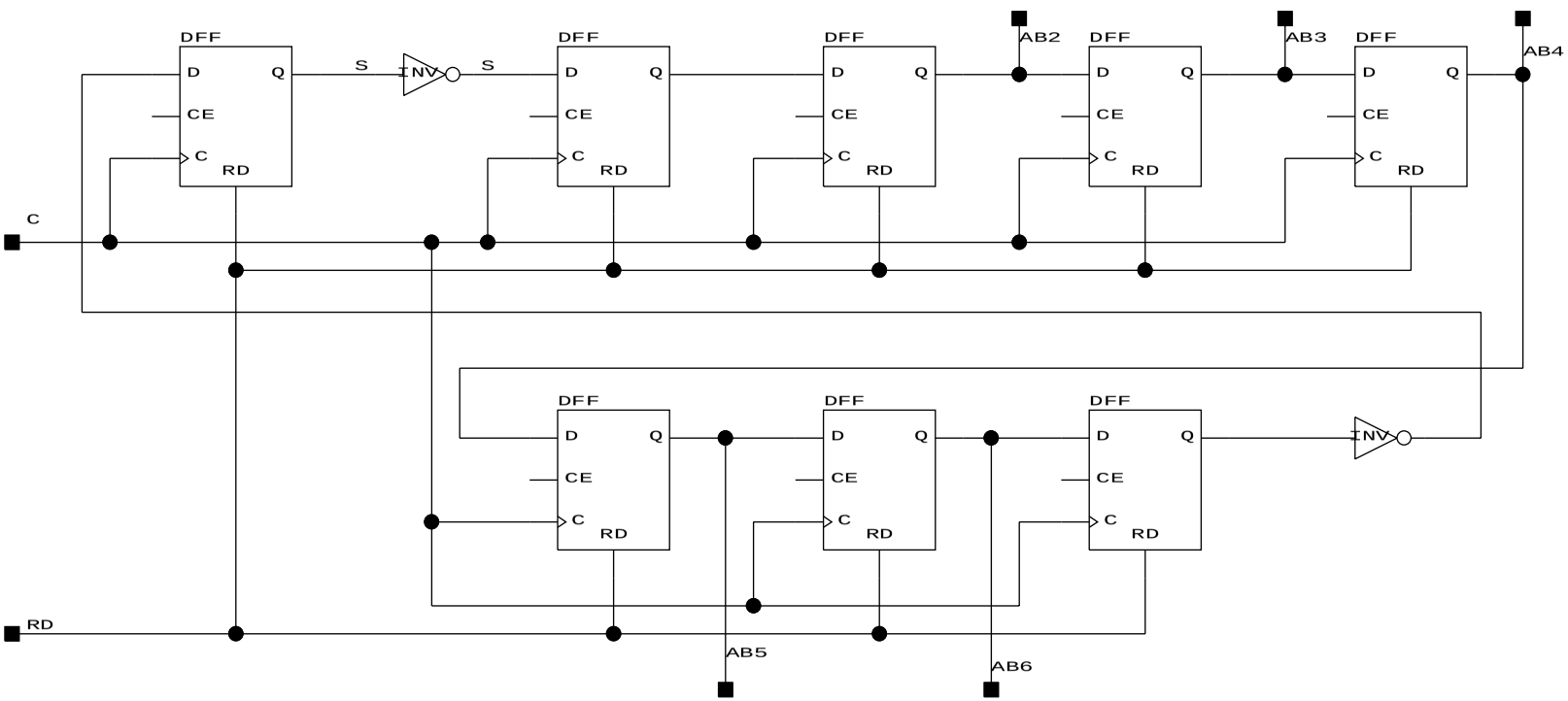
9.4.8. Sendeeinheit (BIT_TRAN)


CLOCK
INV
DFF
D
Q
CE
C
RD
M4-1
D0
D1
D2
D3
S0
S1
O
BAUD0
BAUD1
c13dff
C
TC
c7bcrd
TCV
CLOCK
TAST8
TAST7

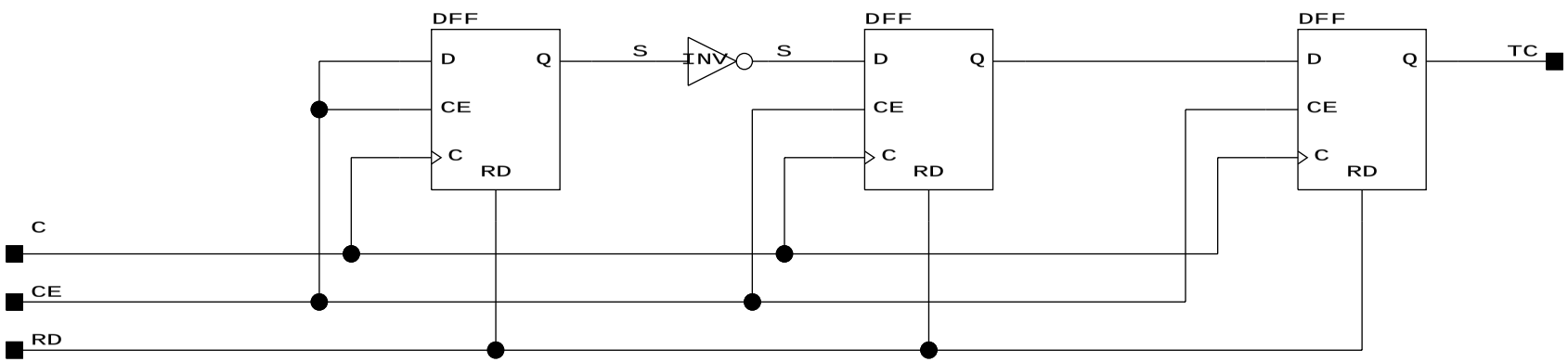
9.4.9. 8-Bit Schieberegister (SHIFTER)


D0
PE
D
C
D1
D2
D3
D4
D5
D6
D7
D8
AND2
AND2B1
OR2
DFF
D
CE
C
RD
Q
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8

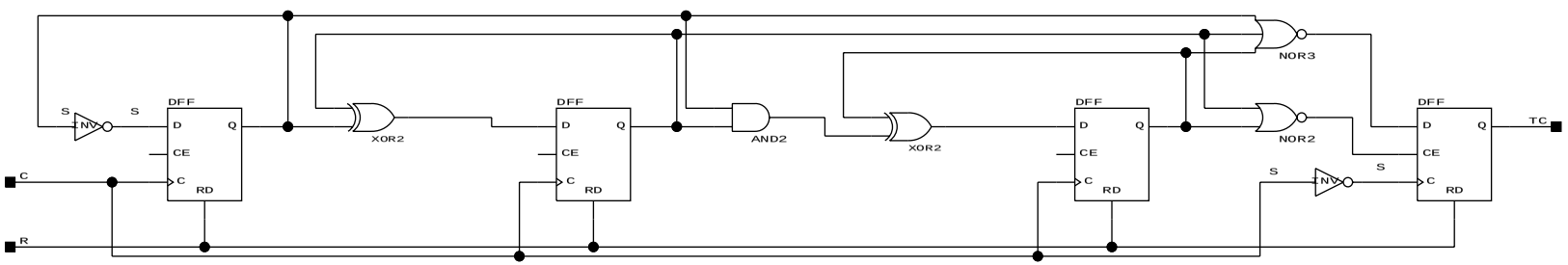
9.4.10. 8-Bit-Rotationsregister (ABDFF)


DFF
D
Q
CE
C
RD
S
INV
AB2
AB3
AB4
AB5
AB6
C
RD

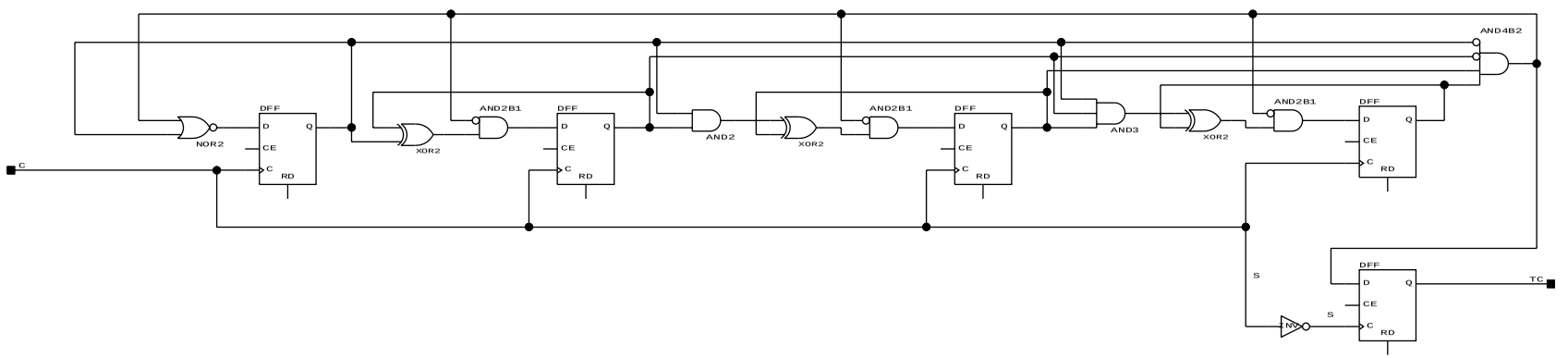
9.4.11. (C2DFF)


DFF
D
Q
CE
C
RD
S
INV
S
DFF
D
Q
CE
C
RD
DFF
D
Q
CE
C
RD
TC
C
CE
RD

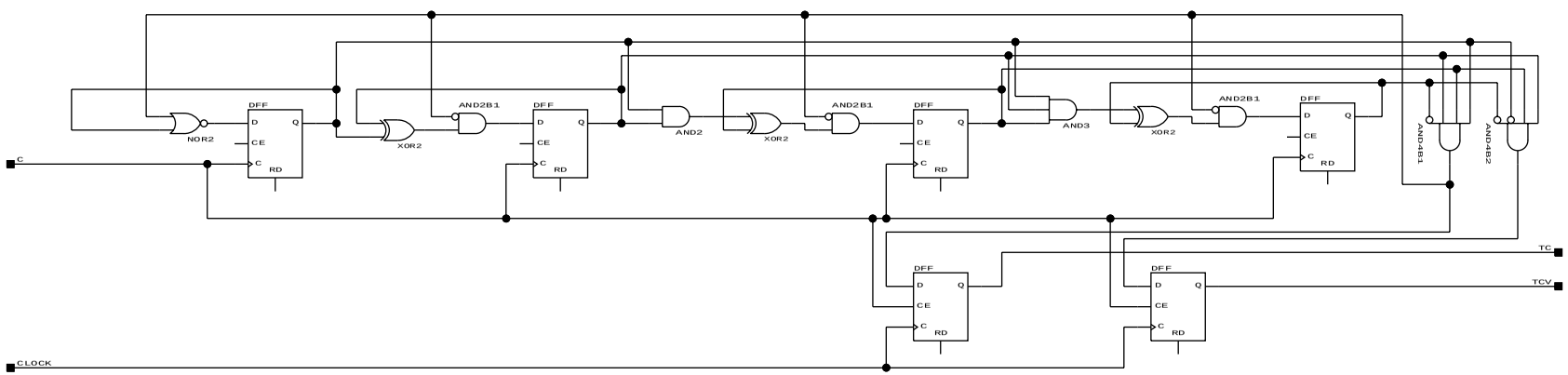
9.4.12. (C8DFF)


S
INV
S
DFF
D
Q
CE
C
RD
XOR2
DFF
D
Q
CE
C
RD
AND2
XOR2
DFF
D
Q
CE
C
RD
NOR3
NOR2
S
INV
S
DFF
D
Q
CE
C
RD
TC
C
R

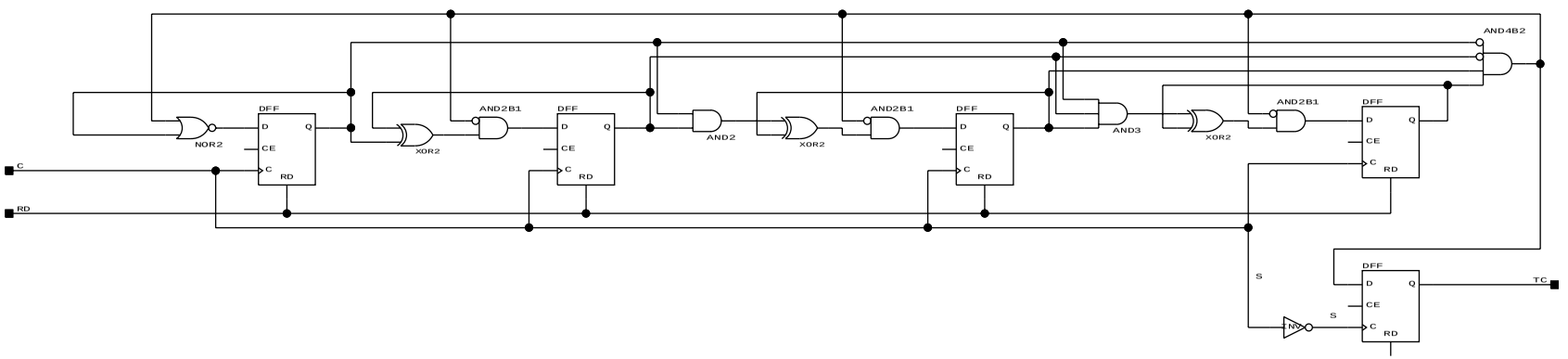
9.4.13. (C13DFF)


C
NOR2
DFF
D
Q
CE
C
RD
XOR2
AND2B1
DFF
D
Q
CE
C
RD
AND2
XOR2
AND2B1
DFF
D
Q
CE
C
RD
AND3
XOR2
AND2B1
AND4B2
DFF
D
Q
CE
C
RD
S
INV
S
DFF
D
Q
CE
C
RD
TC

9.4.14. (C7BCRD)


C
CLOCK
NOR2
XOR2
AND2B1
AND2
XOR2
AND2B1
AND3
XOR2
AND2B1
AND4B1
AND4B2
DFF
D
Q
CE
C
RD
TC
TCV

9.4.15. (C13BCRD)


C
RD
NOR2
DFF
D
Q
CE
C
RD
XOR2
AND2B1
DFF
D
Q
CE
C
RD
AND2
XOR2
AND2B1
DFF
D
Q
CE
C
RD
AND3
XOR2
AND2B1
AND4B2
DFF
D
Q
CE
C
RD
S
INV
S
DFF
D
Q
CE
C
RD
TC

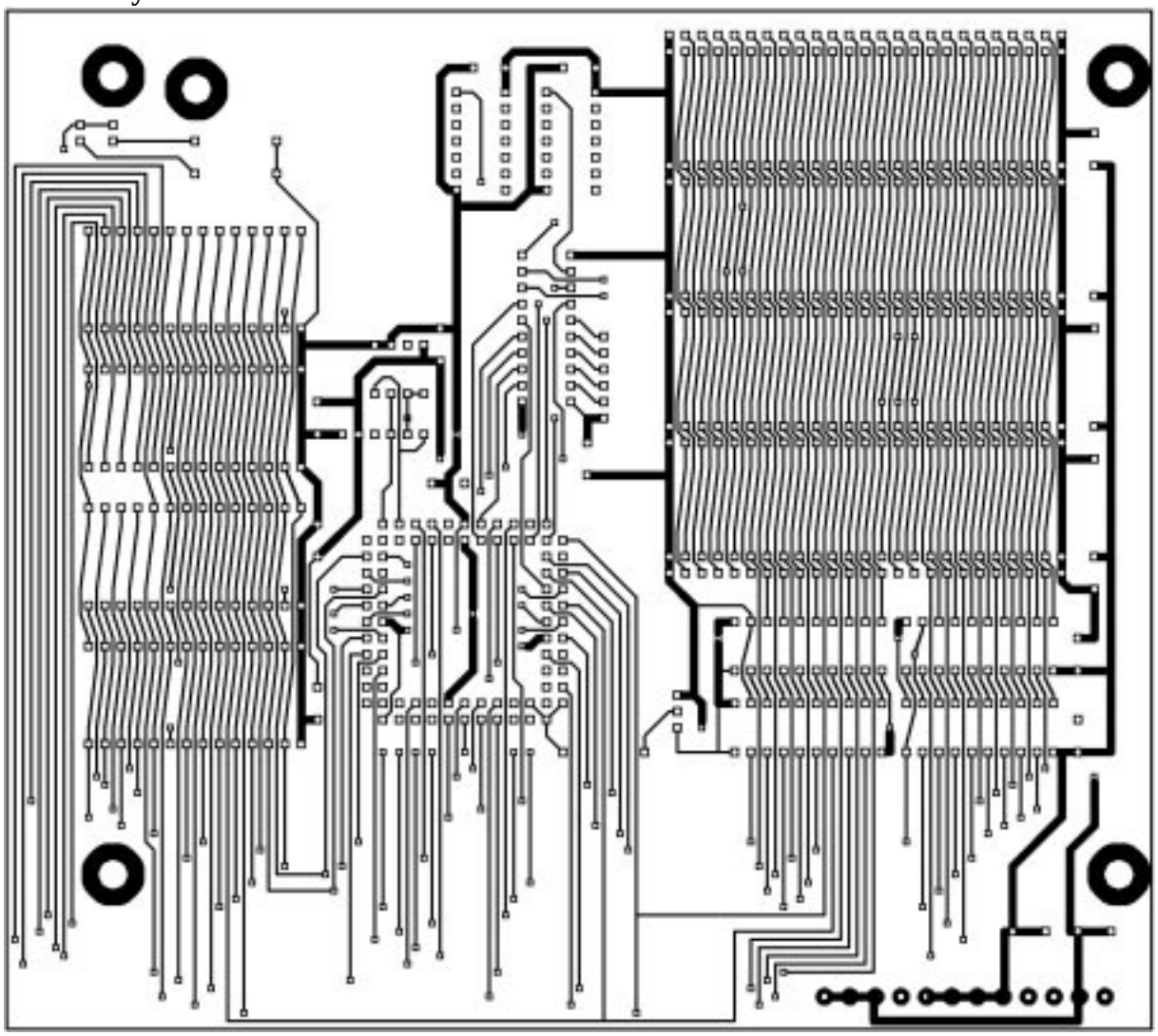
10. Anhang B: Platinenschaltpläne und Platinenlayouts

10.1. Hauptplatine

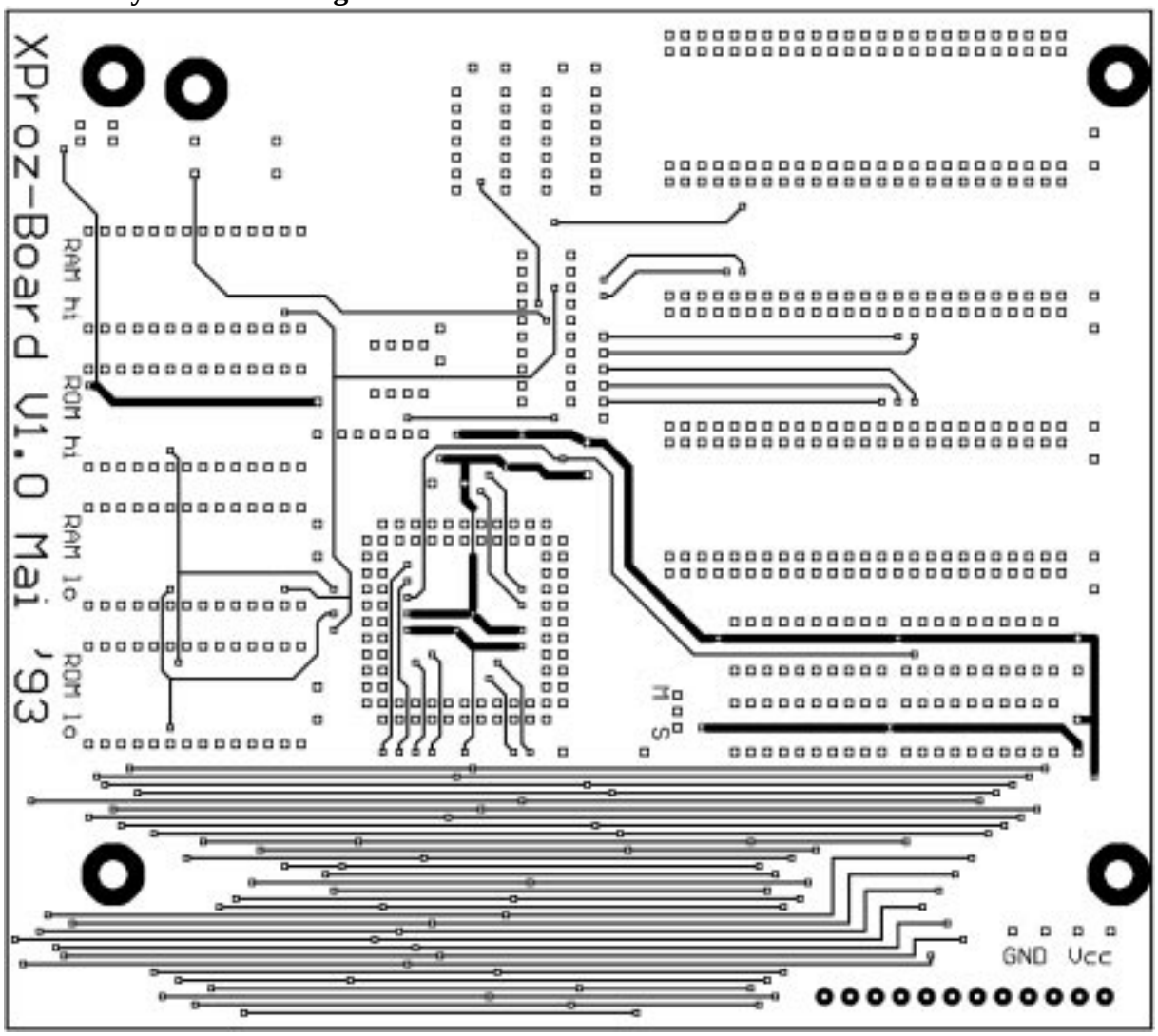
10.1.1. Schaltplan

Power-LED
Resettaster
Power
Slot
BD[15:0]
BA[14:0]
ROM32k*8
RAM32k*8
74LS245
XProz
Oszi
A[14:0]
D[15:0]
BI[1:5]

10.1.2. Layout Lötseite

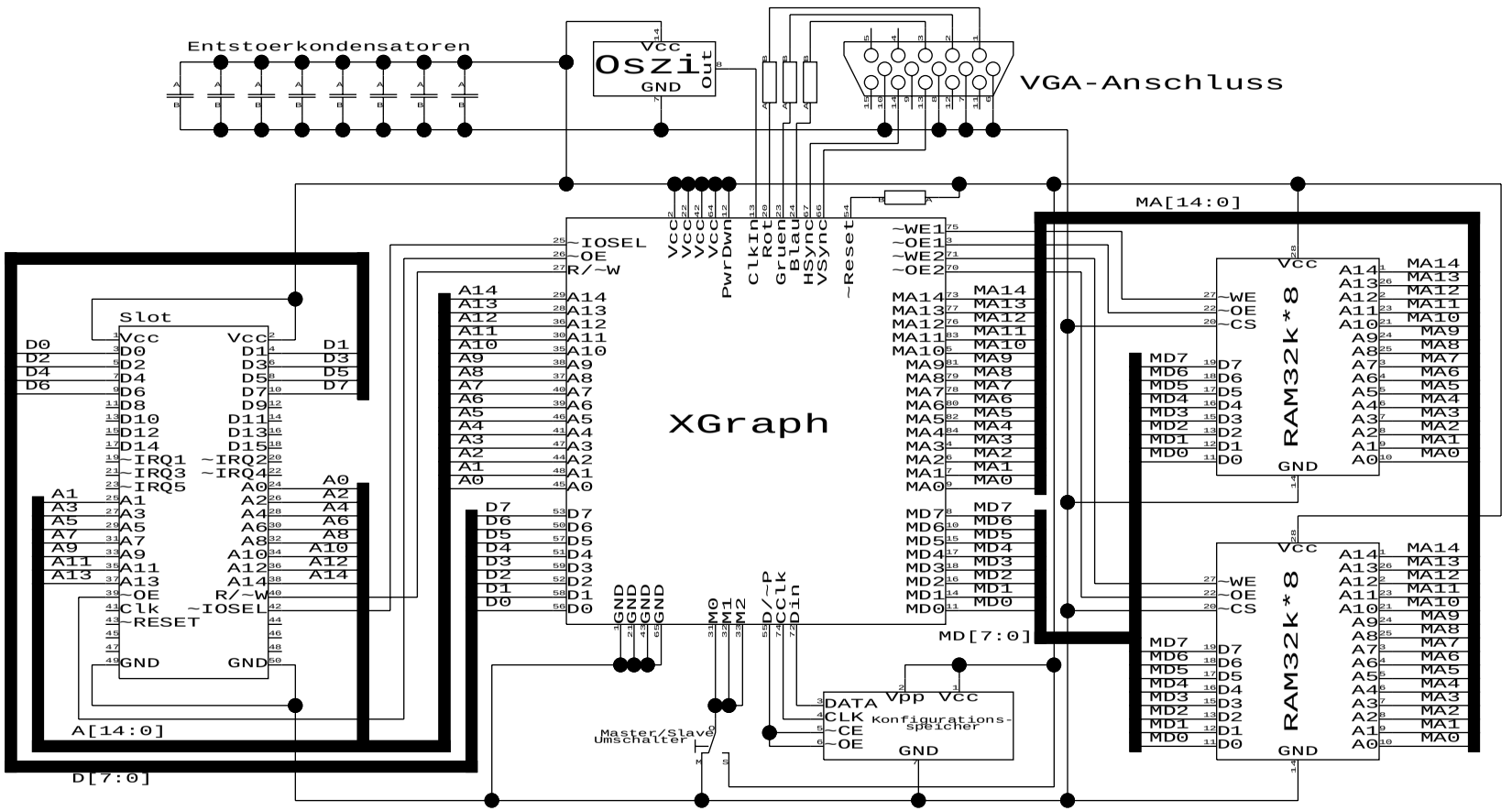


10.1.3. Layout Bestückungsseite

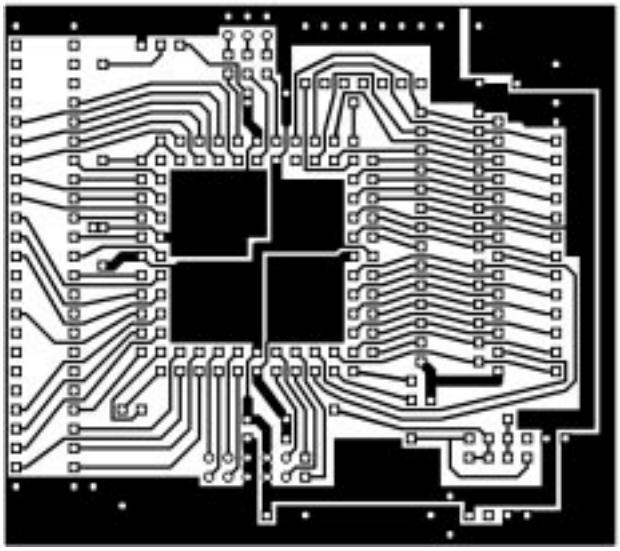


10.2. Grafikkarte

10.2.1. Schaltplan

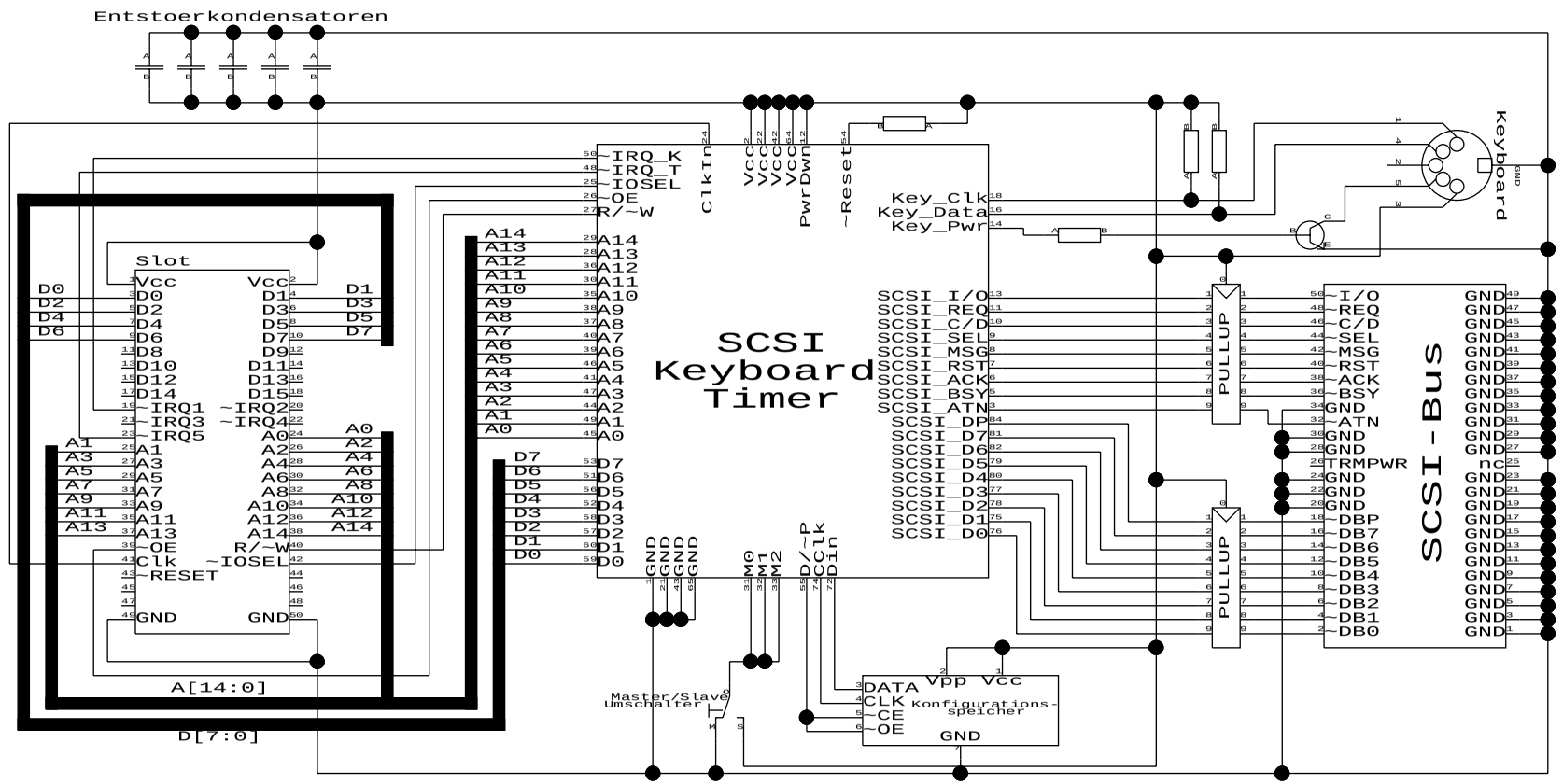

Entstoerkondensatoren
Oszi
Vcc
GND
Out
VGA-Anschluss
XGraph
Slot
RAM32k*8
Konfigurations-
speicher
Master/Slave
Umschalter
MA[14:0]
MD[7:0]
A[14:0]
D[7:0]

10.2.2. Layout Lötseite

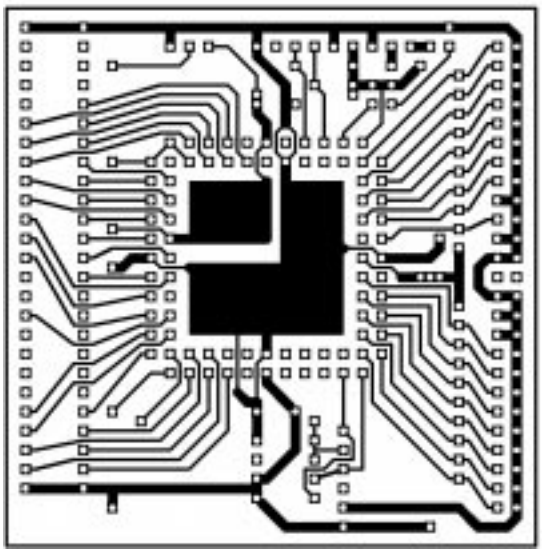


10.3. SCSI-Keyboard-Timer-Karte

10.3.1. Schaltplan

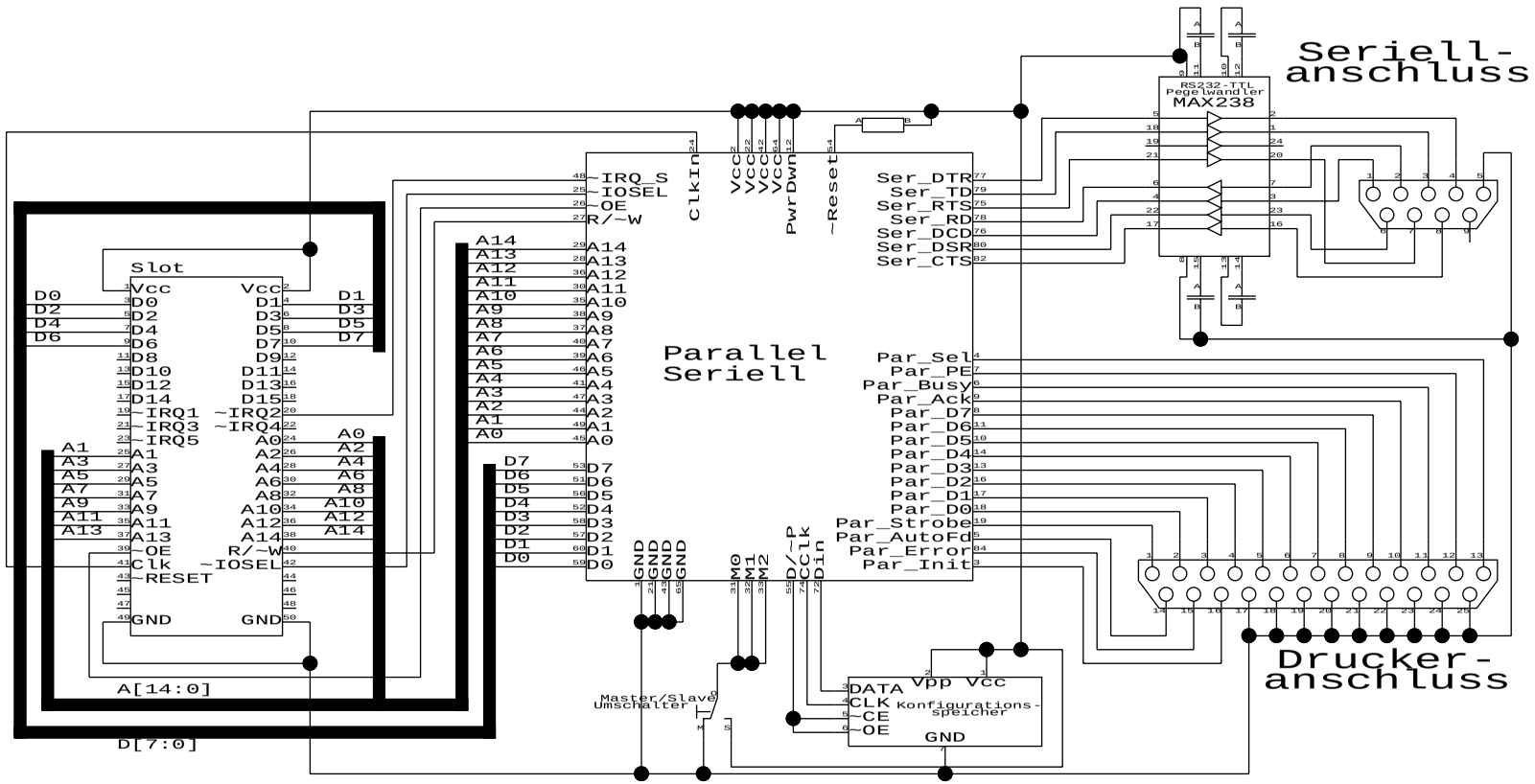

Entstoerkondensatoren
Slot
A[14:0]
D[7:0]
SCSI
Keyboard
Timer
~IRQ_K
~IRQ_T
~IOSEL
~OE
R/~W
ClkIn
Vcc
PwrDwn
~Reset
Key_Clk
Key_Data
Key_Pwr
SCSI_I/O
SCSI_REQ
SCSI_C/D
SCSI_SEL
SCSI_MSG
SCSI_RST
SCSI_ACK
SCSI_BSY
SCSI_ATN
SCSI_DP
SCSI_D7
SCSI_D6
SCSI_D5
SCSI_D4
SCSI_D3
SCSI_D2
SCSI_D1
SCSI_D0
PULLUP
Keyboard
SCSI-Bus
~I/O
~REQ
~C/D
~SEL
~MSG
~RST
~ACK
~BSY
~ATN
TRMPWR
nc
~DBP
~DB7
~DB6
~DB5
~DB4
~DB3
~DB2
~DB1
~DB0
GND
M0
M1
M2
D/~P
CClk
Din
Master/Slave
Umschalter
Konfigurations-
speicher
DATA
CLK
~CE
~OE
Vpp
Vcc

10.3.2. Layout Lötseite

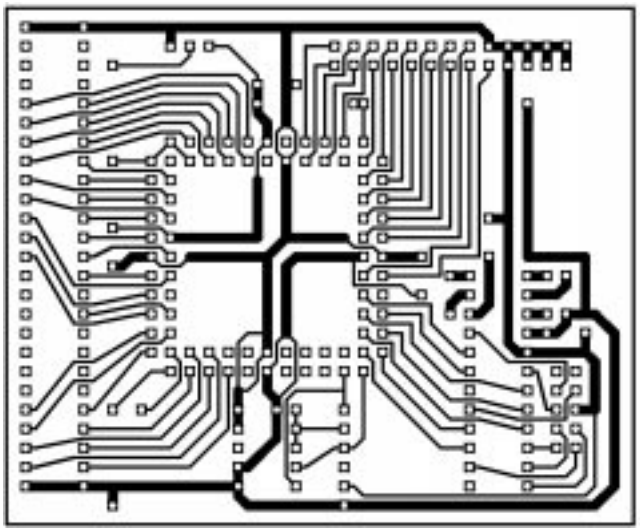


10.4. Centronics-RS232-Timer-Karte

10.4.1. Schaltplan


Seriell-
anschluss
RS232-TTL
Pegelwandler
MAX238
Parallel
Seriell
Slot
A[14:0]
D[7:0]
Master/Slave
Umschalter
Konfigurations-
speicher
Drucker-
anschluss

10.4.2. Layout Lötseite



11. Anhang C: Tastaturtabelle

Taste	Tastaturcode	Normal (ASCII, Zeichen)	Shift (ASCII, Zeichen)	Strg (ASCII, Zeichen)	Alt (ASCII, Zeichen)
A	28	$61,a	$41,A	$1,-	$61,a
B	50	$62,b	$42,B	$2,-	$62,b
C	33	$63,c	$43,C	$3,-	$63,c
D	35	$64,d	$44,D	$4,-	$64,d
E	36	$65,e	$45,E	$5,-	$65,e
F	43	$66,f	$46,F	$6,-	$66,f
G	52	$67,g	$47,G	$7,-	$67,g
H	51	$68,h	$48,H	$8,-	$68,h
I	67	$69,i	$49,I	$9,-	$69,i
J	59	$6a,j	$4a,J	$a,-	$6a,j
K	66	$6b,k	$4b,K	$b,-	$6b,k
L	75	$6c,l	$4c,L	$c,-	$6c,l
M	58	$6d,m	$4d,M	$d,-	$e6,µ
N	49	$6e,n	$4e,N	$e,-	$6e,n
O	68	$6f,o	$4f,O	$f,-	$6f,o
P	77	$70,p	$50,P	$10,-	$70,p
Q	21	$71,q	$51,Q	$11,-	$40,@
R	45	$72,r	$52,R	$12,-	$72,r
S	27	$73,s	$53,S	$13,-	$73,s
T	44	$74,t	$54,T	$14,-	$74,t
U	60	$75,u	$55,U	$15,-	$75,u
V	42	$76,v	$56,V	$16,-	$76,v
W	29	$77,w	$57,W	$17,-	$77,w
X	34	$78,x	$58,X	$18,-	$78,x
Y	26	$79,y	$59,Y	$19,-	$79,y
Z	53	$7a,z	$5a,Z	$1a,-	$7a,z
Ä	82	$84,ä	$8e,Ä	$84,ä	$84,ä
Ö	76	$94,ö	$99,Ö	$94,ö	$94,ö
Ü	84	$81,ü	$9a,Ü	$81,ü	$81,ü
0	69	$30,0	$3d,=	$1e,-	$7d,}
1	22	$31,1	$21,!	$31,1	$31,1
2	30	$32,2	$22,"	$32,2	$fd,²
3	38	$33,3	$dd,§	$33,3	$33,3
4	37	$34,4	$44,$	$34,4	$34,4
5	46	$35,5	$25,%	$35,5	$35,5
6	54	$36,6	$26,&	$36,6	$36,6
7	61	$37,7	$2f,/	$1b,-	$7b,{
8	62	$38,8	$28,(	$1c,-	$5b,[
9	70	$39,9	$29,)	$1d,-	$5d,]

Tabelle 84: Tastaturtabelle (Teil 1)

Taste	Tastaturcode	Normal (ASCII, Zeichen)	Shift (ASCII, Zeichen)	Strg (ASCII, Zeichen)	Alt (ASCII, Zeichen)
<>	19	$3c,<	$3e,>	$3c,<	$7c,|
,;	65	$2c,,	$3b,;	$2c,,	$2c,,
.:	73	$2e,.	$3a,:	$2e,.	$2e,.
-_	74	$2d,-	$5f,_	$2d,-	$2d,-
ß?	78	$9e,ß	$3f,?	$9e,ß	$5c,\
+*	91	$2b,+	$2a,*	$2b,+	$7e,~
^`	85	$27,´	$60,`	$27,´	$27,´
#'	83	$23,#	$27,'	$23,#	$23,#
^°	14	$5e,^	$f8,°	$5e,^	$5e,^
Space	41	$20,Space	$20,Space	$20,Space	$20,Space
ESC	8	$27,-	$27,-	$27,-	$27,-
Tab	13	$8,-	$8,-	$8,-	$8,-
F1	7	$00,-	$00,-	$00,-	$00,-
F2	15	$00,-	$00,-	$00,-	$00,-
F3	23	$00,-	$00,-	$00,-	$00,-
F4	31	$00,-	$00,-	$00,-	$00,-
F5	39	$00,-	$00,-	$00,-	$00,-
F6	47	$00,-	$00,-	$00,-	$00,-
F7	55	$00,-	$00,-	$00,-	$00,-
F8	63	$00,-	$00,-	$00,-	$00,-
F9	71	$00,-	$00,-	$00,-	$00,-
F10	79	$00,-	$00,-	$00,-	$00,-
F11	86	$00,-	$00,-	$00,-	$00,-
F12	94	$00,-	$00,-	$00,-	$00,-
Return	90	$0d,-	$0d,-	$0d,-	$0d,-
Enter	121	$0d,-	$0d,-	$0d,-	$0d,-
Backspace	102	$9,-	$9,-	$9,-	$9,-
Delete	100	$00,-	$00,-	$00,-	$00,-
Insert	103	$00,-	$00,-	$00,-	$00,-
Page up	101	$00,-	$00,-	$00,-	$00,-
Page down	110	$00,-	$00,-	$00,-	$00,-
Pos1	109	$00,-	$00,-	$00,-	$00,-
End	111	$00,-	$00,-	$00,-	$00,-
Crs up	99	$00,-	$00,-	$00,-	$00,-
Crs down	96	$00,-	$00,-	$00,-	$00,-
Crs left	97	$00,-	$00,-	$00,-	$00,-
Crs right	106	$00,-	$00,-	$00,-	$00,-

Tabelle 85: Tastaturtabelle (Teil 2)

Taste	Tastaturcode	Normal (ASCII, Zeichen)	Shift (ASCII, Zeichen)	Strg (ASCII, Zeichen)	Alt (ASCII, Zeichen)
0, Ziffernfeld	112	$30,0	$30,0	$30,0	$30,0
1, Ziffernfeld	105	$31,1	$31,1	$31,1	$31,1
2, Ziffernfeld	114	$32,2	$32,2	$32,2	$32,2
3, Ziffernfeld	122	$33,3	$33,3	$33,3	$33,3
4, Ziffernfeld	107	$34,4	$34,4	$34,4	$34,4
5, Ziffernfeld	115	$35,5	$35,5	$35,5	$35,5
6, Ziffernfeld	116	$36,6	$36,6	$36,6	$36,6
7, Ziffernfeld	107	$37,7	$37,7	$37,7	$37,7
8, Ziffernfeld	117	$38,8	$38,8	$38,8	$38,8
9, Ziffernfeld	125	$39,9	$39,9	$39,9	$39,9
/, Ziffernfeld	119	$2f,/	$2f,/	$2f,/	$2f,/
*, Ziffernfeld	126	$2a,*	$2a,*	$2a,*	$2a,*
-, Ziffernfeld	132	$2d,-	$2d,-	$2d,-	$2d,-
+, Ziffernfeld	124	$2b,+	$2b,+	$2b,+	$2b,+
,, Ziffernfeld	113	$2c,,	$2c,,	$2c,,	$2c,,
Caps-Lock	20	-	-	-	-
Shift-Links	18	-	-	-	-
Shift-Rechts	89	-	-	-	-
Num-Lock	118	-	-	-	-
Ctrl-Links	17	-	-	-	-
Ctrl-Rechts	88	-	-	-	-
Scroll-Lock	95	-	-	-	-
Alt-Links	25	-	-	-	-
Alt-Rechts	57	-	-	-	-
Pause	98	-	-	-	-
Print	87	-	-	-	-

Tabelle 86: Tastaturtabelle (Teil 3)

12. Anhang D: Listings

12.1. Initialisierung

```
origin $8000
include lib

;================================================================================
;SYSTEMVARIABLEN
;================================================================================
equ bfs,6               ;Pointer auf Heap
equ efs,7               ;Adresse des 1. Wortes, welches nicht mehr zum Heap
                        ;gehört (=Adresse des 1. Cachebuffers)
equ dummy,18            ;Dummyregister zur Einstellung des Statusregisters

;Serielle Schnittstelle
equ flags2,19           ;Reg. zur Sicherung der Flags
equ s_write,20          ;Schreibzeiger im Empfangspufferbereich
equ s_read,21           ;Lesezeiger im Empfangspufferbereich
equ s_int_ar,22         ;Arbeitregister der Interruptroutine
equ s_error,23          ;Fehlerregister für Serielle-Schnittstelle,Statusregister wird im
höherweritigen Byte
                        ;eines empfangenen Zeichens (1 Wort im Puffer) abgelegt
equ s_buffer,24         ;Zeiger auf den Pufferanfang, kann bei speziellen
                        ;Kommunikationsprogrammen verändert werden
equ s_buffer_end,25     ;Zeiger auf Pufferende, erstes Wort welches nicht mehr zum Puffer
                        ;gehört
equ cts,78              ;Variablen für Software-Handshaking
equ rts,79

;Heap- ,Driver- und Taskverwaltung
equ h_in_use,26         ;Semaphor
equ driverlist,27       ;Pointer auf Driverliste
equ resetvector,77      ;Über diese Variable wird beim entfernen der letzten
                        ;Task gesprungen
equ akt_task,17         ;Pointer auf TDB (Task-Desktriptor-Block)

;Timer:
equ timerh,28           ;Timer-High-Wert
equ timerl,29           ;Timer-Low-Wert
equ flags5,30           ;Enthält die Flags beim Interrupt
equ task_switch_tmp,31  ;Temporäres Arbeitsregister beim Taskswitchen

;Konsole (Tastatur und Bildschirm)
equ umschaltflags,32    ;enthält Flags der Umschalttasten
equ flags1,33           ;Enthält gesicherte Flags
equ t_int_ar,34         ;Arbeitsregister der Interruptroutine
equ con_in_use,35       ;Semaphor
equ t_read_ptr,36       ;Lesezeiger
equ t_write_ptr,37      ;Schreibzeiger
equ t_buffer,38         ;Zeiger auf den Pufferanfang
equ t_buffer_end,39     ;Zeiger auf Pufferende
equ cursor_adress,40
equ cursor_x,41
equ cursor_y,42
equ vt52_seq,43
equ save_x,44
equ save_y,45
equ save_adr,46
equ fcolor,47
equ bcolor,48

equ t_read_ptr2,49       ;Dasselbe für die gerade nicht aktive Konsole
equ t_write_ptr2,50
equ t_buffer2,51
equ t_buffer_end2,52
equ cursor_adress2,53
equ cursor_x2,54
equ cursor_y2,55
equ vt52_seq2,56
equ save_x2,57
equ save_y2,58
equ save_adr2,59
equ fcolor2,60
equ bcolor2,61

equ active_con,62       ;Welche Konsole (0/1?) ist aktive?
equ t_int_ar2,63        ;2. Arbeitsregister der Tastaturinterruptroutine

;SCSI:
```

```
equ status,64           ;Status
equ accesses,65         ;Zählt Zugriffe auf Sektoren
equ media_writeprot,66  ;...
equ media_descr,67      ;Pointer auf Media-Deskriptor-Blöcke (je 8 Worte)
equ not_ready_active,68 ;Zeigt an, ob die Drive-Not-ready Routine schon arbeitet
equ scsi_in_use,69      ;SCSI-Semaphor
equ scsi_buffer,70      ;Zeiger auf Arbeitspufferanfang
equ command_buffer,71   ;Zeiger auf Kommandopuffer
equ want_to_use_scsi,72 ;Fairness-Flag
equ removable,76        ;Welche Devices haben wechselbare Medien?

;Fileverwaltung
equ fdblist_in_use,73   ;Semaphor
equ fdblist,74          ;Pointer auf die FDB-Liste
equ allocate_in_use,75  ;Semaphor

;============================================================================
;HARDWARERESET-EINSPRUNG
;============================================================================
bs_start:       jmp       reset        ;Weiter zur eigentlichen Reset-Routine

;============================================================================
;SYSTEMAUFRUF-DISPATCHER (Einsprungadresse $8002)
;============================================================================
                pop       d0                  ;Funktionsnummer vom Stack
                cmp       d0,#32              ;32 ist größte Funktionsnummer
                rts mi                        ;Fertig, falls ungültige Funktionsnummer
                add       #.table,d0          ;Funktionsadresse berechnen
                add       (d0),-,=            ;Funktion anspringen

;Tabelle mit Adressen aller BS-Funktionen
.table:         dw        malloc              ;0:Speicherblock allokieren
                dw        mshrink             ;1:Speicherblock verkleinern
                dw        mfree               ;2:Speicherblock freigeben
                dw        meminfo             ;3:Freien Speicherplatz ermitteln
                dw        starttask           ;4:Task anmeldem
                dw        switchtask          ;5:Task umschalten
                dw        killtask            ;6:Task beenden
                dw        newdriver           ;7:Neuen Devicedriver anmelden
                dw        searchdriver        ;8:Devicedriver suchen
                dw        set_env             ;9:Env.-Variable setzen
                dw        search_env          ;10:Suche Environmentvariable
                dw        clr_env             ;11:Env.-Variable löschen
                dw        open                ;12:Datei öffnen
                dw        read                ;13:Lesen aus Datei
                dw        write               ;14:Schreiben in Datei
                dw        close               ;15:Datei schließen
                dw        get_flags           ;16:Dateiflags ermitteln
                dw        set_flags           ;17:Dateiflags setzen
                dw        get_pos             ;18:Fileposition setzen
                dw        set_pos             ;19:Fileposition setzen
                dw        rename              ;20:Rename
                dw        delete              ;21:Delete
                dw        md                  ;22:Make Directory
                dw        rd                  ;23:Remove Directory
                dw        cd                  ;24:Change Directory
                dw        prg_load            ;25:Programm laden
                dw        prepare_search      ;26:Suchstring vorbereiten
                dw        search_next         ;27:nächsten DirEintrag suchen
                dw        scan                ;28:Dateinamen einlesen
                dw        fehler              ;29:Fehlerstring an StandardOut ausgeben
                dw        shell               ;30:Shell
                dw        delay               ;31:Zeit verbraten
                dw        get_lng             ;32:Filelänge ermitteln

;============================================================================
;INITIALISIERUNG
;============================================================================
reset:          out       #0,$7ff3            ;Timer off
                move      #user_pc,0          ;Initialisierung läuft im User-Mode
                move      #$fe00,~dummy       ;Switch to User-Mode

;Heapverwaltung initialisieren
user_pc:        move      #reset,resetvector  ;Resetvector einstellen
                move      #80,bfs             ;Anfang des freien Speichers
                move      #bs_start-80,(bfs)  ;Länge dieses Speicherblocks
                add       -,bfs,d0
                clr       (d0)                ;Speicherbereich als frei kennzeichnen
                move      #bs_start,efs       ;Ende des freien Speichers (hier=$8000)
                clr       h_in_use            ;Heapverwaltung verfügbar

;Dateiverwaltung initialisieren
```

```
                clr         fdblist             ;Semaphore für Fileverwaltung
                clr         fdblist_in_use
                clr         allocate_in_use

;Taskverwaltung initialisieren, 1. Task erzeugen
                ror         -,-,sp              ;Temporärer Stack an Adresse $8000
                push        #278                ;Wir wollen 256 Worte Environment
                jsr         malloc              ;TDB allokieren
                inc         sp
                move        d0,akt_task         ;Adresse des TDB wird Tasknummer
                dec         d0
                move        akt_task,(d0)       ;Speicherblock gehört uns
                add         #12,d0              ;Nächster Task sind wir selbst
                move        akt_task,(d0)
                move        #.taskname,d1       ;Taskname eintragen
.nameloop:      inc         d0                  ;Name des Tasks: "XMOS 1.0"
                move    sf (d1),(d0)
                inc         d1
                jpl         .nameloop

                push        #96                 ;Speicher für Stack
                jsr         malloc
                inc         sp
                sub         #2,d0
                add         (d0),d0,sp          ;Stackpointer init
                push        akt_task            ;Tasknummer für Killtask
                push        #killtask           ;Killtaskadresse für Beendigung

;Hardwaretreiber initialisieren
                push        #32         ;Driverlist init und löschen
                jsr         MALLOC
                inc         sp
                move        d0,driverlist
                sub         -,d0,d1
                sub         -,-,(d1)    ;Sys-Speicherblock (Tasknummer $ffff)
                move        #32,d1      ;Treiberliste mit Null initialisieren
.clear:         clr         (d0)
                inc         d0
                dec     sf d1
                jne         .clear

                jsr         timer       ;Timer init
                jsr         konsole     ;Diverse Treiber init
                jsr         scsi
                jsr         romdisk
                jsr         parallel
                jsr         seriell

;Standard CWD, Pfad setzen, StandardIn/Out öffnen
                push        #.defaultcwd        ;Std-Umgebungsvariablen setzen
                jsr         set_env
                push        #.defaultpath
                jsr         set_env
                push        #2                  ;Zum lesen/schreiben
                push        #.defaultio         ;Stdio öffnen
                jsr         OPEN
                add         #4,sp
                add         #20,akt_task,d1     ;In TDB eintragen
                move        d0,(d1)
                inc         d1
                move        d0,(d1)

                pushz                           ;Einschaltmeldung
                push        #.remark
                pushz
                jsr         WRITE
                add         #3,sp

                jmp         shell               ;Shell starten

.taskname:      dw      "XMOS 1.0",0,0,0,$ffff
.defaultio:     dw      "/con0",0
.defaultcwd:    dw      "cwd=/b/",0
.defaultpath:   dw      "path=/rom/",$3b,"/b/",0
.remark:        dw 10,13,"Welcome to the "
                dw $1b,$47,$00,$00,$1f,$3e,$7c,$f8,$7c,$3e,$1f,$3e,$7c,$f8,$7c,$3e,$1f,$00
                dw $1b,$47,$00,$00,$7c,$f8,$70,$20,$00,$00,$00,$00,$00,$20,$70,$f8,$7c,$00
                dw "Xilinx Multitasking Operating System V1.0, November 1993",13,10,0

;Komponeneten einbinden
include konsole         ;Konsolentreiber, Tastaturinterruptroutine, Zeichensatz
include scsi            ;SCSI-Treiber und Cache
```

```
include shelltxt        ;Shell
include seriell         ;Text-Treiber für serielle Schnittstelle
include parallel        ;Parallelport-Treiber
include romdisk         ;ROM-Disk Treiber mit Datenbereich
include file            ;Dateiverwaltung
```

12.2. Treiberverwaltung

```
;================================================================================
;GERÄTETREIBER-VERWALTUNG
;================================================================================
```

12.2.1. NEWDRIVER

```
;--------------------------------------------------------------------------------
;NEWDRIVER
;--------------------------------------------------------------------------------
;Anmelden eines neuen Treibers
;d0: Zeiger in der Treiberliste
;d1: Zähler
newdriver:      irqoff                          ;Keine Unterbrechung zulassen
                move       driverlist,d0        ;Suche freien Eintrag
                move       #32,d1               ;Zähler, max. 32 Einträge
.listsearch:    dec     sf d1
                jmi        .full                ;Driverliste leider voll
                cmp        (d0),-               ;Eintrag frei?
                inc  ne    d0                   ;Nein weiter
                jne        .listsearch

;d1: Zeiger im Stack
                add        -,sp,d1
                move       (d1),(d0)    ;Treiberanfangsadresse eintragen
.ende:          irqon
                rts

.full:          ror     sf -,-,-        ;N-Flag setzen, Liste ist voll
                jmp         .ende
```

12.2.2. SEARCHDRIVER

```
;--------------------------------------------------------------------------------
;SEARCHDRIVER
;--------------------------------------------------------------------------------
;Suchen nach einem angemeldeten Treiber
;d0, d1: Zeiger auf zu suchenden Treibernamen
;d2: Zeiger auf Treiberliste
;d3: Zeiger auf Treibernamen im Treiber
searchdriver:   add        -,sp,d0
                push       d2
                push       d3
                move       (d0),d0              ;Pointer auf Name des gesuchten Drivers
                move       driverlist,d2        ;Adresse der Treiberliste
.loop:          move       d0,d1
                move    sf (d2),d3              ;Adresse des Treibers holen
                jeq        .not_found           ;Es gibt keine Treiber mehr
                cmp        (d1),(d3)            ;1.Zeichen gleich?
                jne        .notyet              ;Nein, Treiber ist nicht der gesuchte
                inc         d1
                inc         d3
                cmp        (d1),(d3)            ;2. Zeichen gleich?
                jne        .notyet              ;Nein, Treiber ist nicht der gesuchte
                cmp        (d1),-               ;Ende des Namens?
                jeq        .found
                inc         d1
                inc         d3
                cmp        (d1),(d3)            ;3. Zeichen gleich?
                jne        .notyet              ;Nein, Treiber ist nicht der gesuchte
                cmp        (d1),-               ;Ende des Namens?
                jeq        .found
                inc         d1
                inc         d3
                cmp        (d1),(d3)            ;4. Zeichen  gleich?
                jne        .notyet              ;Nein, Treiber ist nicht der gesuchte
.found:         add        #5,(d2),d0           ;Treiber gefunden, Adresse in d0
                clr     sf ~-                   ;N-Flag löschen
.end:           pop        d3
                pop        d2
                rts

;Dieser Treiber war unpassend, nächsten Treiber betrachten.
.notyet:        inc        d2                   ;Nächster Treiber
                jmp        .loop
```

```
;Liste zu Ende, Treiber nicht gefunden.
.not_found:     move       #106,d0             ;Device exitiert nicht!
                ror     sf -,-,-               ;N-Flag setzen
                jmp        .end
```

12.3. Taskverwaltung

```
;===========================================================================
;TASKVERWALTUNG
;===========================================================================
```

12.3.1. STARTTASK

```
;---------------------------------------------------------------------------
;STARTTASK
;---------------------------------------------------------------------------
;Task im System anmelden und TDB erzeugen
;d0, d2: Adresse der Task
starttask:      add        #2,sp,d0
                move       (d0),d0      ;Startadresse der Task holen
                jsr        find_block   ;zugehörigen Speicherblock ermitteln
                rts  mi
                push       d2
                move       d0,d2        ;Speicherblockadresse retten

                add        #2,sp,d0
                add        #22,(d0),d0  ;Größe des Environmentbereiches

;Speicher für TDB mit Environmentbereich anfordern
                push       d0           ;Neuen TDB anlegen
                jsr        MALLOC
                inc        sp
                pop  mi    d2
                rts  mi                 ;Bei Fehler: Task kann nicht gestartet werden

;d0: Adresse des TDB
;d1: Adresse des zweiten Verwaltungswortes dieses Speicherblocks (Besitzer).
                sub        -,d0,d1      ;Speicherblock gehört dem neuen Task
                move       d0,(d1)
                add        #3,sp,d1     ;d1: Startadresse der neuen Task holen...
                move       (d1),(d0)    ;...und in TDB eintragen

;Taskname eintragen
;d0: Zeiger in TDB
;d1: Zeiger auf Tasknamen
;d2: Zähler
                push       d2
                inc        d1           ;Taskname in TDB eintragen
                move       (d1),d1      ;Pointer auf Taskname holen
                add        #12,d0
                move       #8,d2        ;8 Zeichen kopieren
.copyloop:      move    sf (d1),(d0)
                inc  ne    d1
                inc        d0
                dec     sf d2
                jne        .copyloop
                pop        d2

;StandardIn und StandardOut-FDB eintragen und diese Task als Besitzer dort eintragen
;d0: Zeiger in TDB
;d1: Zeiger auf StdIn im Stack
;d2: Adresse der Task
                add        #5,sp,d1     ;Stdin verwalten
                move       (d1),d1
                move    sf d1,(d0)
                dec        d1
                sub ne     #20,d0,(d1)  ;Stdin-FDB gehört jetzt uns
                inc        d0
                add        #6,sp,d1     ;Ebense Stdout
                move       (d1),d1
                move    sf d1,(d0)
                dec        d1
                sub ne     #21,d0,(d1)

;Environmentbereich der Erzeugertask kopieren.
;Diese Task erbt den Env.-Bereich
;d1: Zeiger auf Ende des TDB
;d2: Zeiger auf Environmentbereich der aufrufenden Task
                push       d0
                push       d2
                inc        d0                   ;d0 zeigt auf neues Environment
```

```
                sub         #24,d0,d1
                add         (d1),d1              ;d1 zeigt auf Ende des TDB
                add         #22,akt_task,d2      ;d2 zeigt auf aktuelles Environment
                cmp         d0,d1                ;Environment leer?
                jeq         .noenv               ;Ja, nichts kopieren
.copyenv:       move       (d2),(d0)            ;Environment vererben
                inc         d2
                inc         d0
                cmp         d0,d1
                jne         .copyenv
                dec         d0                   ;Environmentende setzen
                clr         (d0)
.noenv:         pop        d2
                pop         d0

;gerettet Falgs mit $ffff vorbesetzen
                sub         #20,d0
                sub         -,-,(d0)             ;Flags sind $ffff

;Stackpointer auf Ende der Task einstellen.
;Auf den Stack Tasknummer und Adresse der Killtaskroutine pushen
;d1: Zeiger auf Ende der Task (Stack)
;d2: Zeiger auf Anfangsadresse der Task
                add         (d2),d2,d1           ;d1 ist Stackpointer der neuen Task
                dec         d1
                sub         -,d0,(d1)            ;Eigene Tasknummer auf den Stack pushen
                inc         d2
                move        (d1),(d2)            ;Programmspeicherblock gehört der neuen Task
                dec         d1                   ;ebenso Taskrücksprungadresse
                move        #KILLTASK,(d1)
                inc         d0
                move        d1,(d0)              ;Stackpointer in TDB eintragen

;TDB in Taskliste einhängen, darf nicht unterbrochen werden
;d1: Zeiger auf TDB des Nachfolgers
                irqoff
                add         #9,d0                ;Nachfolger der aktuellen...
                add         #11,akt_task,d1      ;...Task wird Nachfolger...
                move        (d1),(d0)            ;...der neuen Task
                sub         #11,d0,(d1)          ;Neue Task wird Nachfolger der aktuellen Task
                irqon

                clr     sf ~-                    ;N-Flag löschen
                pop         d2
                rts
```

12.3.2. SWITCHTASK

```
;------------------------------------------------------------------------------
;SWITCHTASK
;------------------------------------------------------------------------------
;Taskwechselroutine, kann nicht unterbrochen werden.
;Die Routine ist um die Geschwindigkeit zu optimieren linear und ohne
;Schleifen programmiert. Die Routine läuft im Interruptlevel 5.
switchtask:     irqoff                                  ;Unterbrechung verhindern
                or          ~-,#256,flags5               ;Flags retten, IRQ enable
                move        #.do_it,5
                and         ~-,#$dfff,~dummy             ;In IRQ5 schalten
.do_it:         move    sf task_switch_tmp,(akt_task)   ;Autoswitch: User-PC retten
                move eq     (sp),(akt_task)              ;Syscall: User-PC
                add         -,akt_task,task_switch_tmp
                move        flags5,(task_switch_tmp)     ;Flags retten
                inc         task_switch_tmp
                move        sp,(task_switch_tmp)         ;Stackpointer retten
                inc         task_switch_tmp
                move        d0,(task_switch_tmp)         ;Arbeitsregister retten
                inc         task_switch_tmp
                move        d1,(task_switch_tmp)
                inc         task_switch_tmp
                move        d2,(task_switch_tmp)
                inc         task_switch_tmp
                move        d3,(task_switch_tmp)
                inc         task_switch_tmp
                move        d4,(task_switch_tmp)
                inc         task_switch_tmp
                move        d5,(task_switch_tmp)
                inc         task_switch_tmp
                move        d6,(task_switch_tmp)
                inc         task_switch_tmp
                move        d7,(task_switch_tmp)
                inc         task_switch_tmp
```

```
;Nächste Task aktivieren
                move       (task_switch_tmp),task_switch_tmp    ;Nächste Tasknummer holen
                move       task_switch_tmp,akt_task
                move       (task_switch_tmp),0          ;USER-PC der neuen Task
                inc        task_switch_tmp
                move       (task_switch_tmp),flags5     ;Flags zurück
                inc        task_switch_tmp
                move       (task_switch_tmp),sp         ;Stackpointer zurück
                inc        task_switch_tmp
                move       (task_switch_tmp),d0         ;Arbeitsregister zurück
                inc        task_switch_tmp
                move       (task_switch_tmp),d1
                inc        task_switch_tmp
                move       (task_switch_tmp),d2
                inc        task_switch_tmp
                move       (task_switch_tmp),d3
                inc        task_switch_tmp
                move       (task_switch_tmp),d4
                inc        task_switch_tmp
                move       (task_switch_tmp),d5
                inc        task_switch_tmp
                move       (task_switch_tmp),d6
                inc        task_switch_tmp
                move       (task_switch_tmp),d7
                clr        task_switch_tmp
                move    sf flags5,~dummy        ;Flags zurück
```

12.3.3. TIMER-Interrupt-Routine

```
;-------------------------------------------------------------------------
;TIMER-IRQ-ROUTINE
;-------------------------------------------------------------------------
;Die Timerroutine incrementiert die Zeitzähler lößt einen Taskwechsel aus.
timer_irq:      out        -,$7ff3              ;Timer-IRQ-Bestätigung
                or         ~-,#8192,flags5      ;Flags retten
                inc     sf timerl               ;Timer-Low inkrement
                inc  cs    timerh               ;Timer-High inkrement
                cmp        task_switch_tmp,-    ;Bei Störungen auf der IRQ-Leitung: Kein
Autotaskswitch
                move eq    0,task_switch_tmp    ;Autotaskswitch prepare
                move eq    #switchtask,0        ;Autotaskswitch
                move    sf flags5,~dummy        ;Flags zurück
                jmp        timer_irq            ;u.s.w.
```

12.3.4. KILLTASK

```
;-------------------------------------------------------------------------
;KILLTASK
;-------------------------------------------------------------------------
;Killtask entfernt einen Task aus dem System
;Speicher wird zurückgegeben und Dateien geschlossen
killtask:       irqoff                          ;Keine Unterbrechung
                add        -,sp,d0
                move       (d0),d0              ;d0 zeigt auf TDB des zu löschenden Tasks
                cmp        akt_task,d0          ;Ist dies ein Selbstmord?
                jeq        .suicide             ;ja

;Ist eine gültige Tasknummer angegeben?
;d0: gesuchte Tasknummer
;d1: Zeiger in TDB-Liste
                move       akt_task,d1          ;Suche nach Vorgängertask
.gueltig:       add        #11,d1
                cmp        d0,(d1)              ;Ist nächste Task unsere gesuchte Task?
                jeq        .found               ;ja
                move       (d1),d1
                cmp        d1,akt_task          ;Haben wir schon alle Tasks durchgeschaut?
                jne        .gueltig
                move       #101,d0              ;ja
.fehler:        irqon
                ror     sf -,-,-                ;N-Flag setzen, Tasknummer war ungültig
                rts

;Tasknummer ist gültig. User-Pc der zu entfernenden Task auf die Killtaskroutine
;einstellen und die Tasknummer auf deren Stack pushen. Beim nächsten Taskwechsel
;begeht diese Routine einen Selbstmord
.found:         move       #killtask,(d0)       ;User PC auf Killtaskroutine stellen
                add        #2,d0,d1             ;Adresse des Stackpointers
                move       (d1),d1              ;Wert des Stackpointers
                move       d0,(d1)              ;Tasknummer auf Stack ablegen
                add        #2,d0,d1             ;Adresse des Stackpointers
                dec        (d1)
                irqon
```

```
                clr     sf ~-
                rts

;Selbstmord einer Task, Semaphore löschen
.suicide:       cmp         akt_task,h_in_use    ;benutzt Task die Heapverwaltung?
                clr eq      h_in_use             ;Ja, Heapverwaltung freigeben
                cmp         akt_task,con_in_use  ;...ebenso Konsole...
                clr eq      con_in_use
                cmp         akt_task,fdblist_in_use
                clr eq      fdblist_in_use
                cmp         akt_task,allocate_in_use
                clr eq      allocate_in_use
                cmp         akt_task,scsi_in_use
                irqon
                jne         .fdbstart
                out         -,scsi_ctrl                  ;SCSI-BUS-Reset
                push        #2                           ;kurz warten
                jsr         DELAY
                clr         @scsi_ctrl                   ;SCSI-BUS-Reset clear
                clr         scsi_in_use

;geöffnete Dateien ordnungsgemäß schließen
;d1: Zeiger auf FDB-Liste
.fdbstart:      move        #fdblist,d1          ;Alle geöffneten Dateien des Tasks löschen
.fdblist:       cmp         (d1),-
                jeq         .go_on
                move        (d1),d1
                dec         d1
                cmp         (d1),akt_task        ;gehört die Datei dieser Task?
                inc         d1
                jne         .fdblist
                push        d1                   ;Datei schließen
                jsr         CLOSE
                inc         sp
                jmp         .fdbstart

;RESET auslösen?
.go_on:         irqoff                           ;Unterbrechung unterbinden
                add         #11,akt_task,d0      ;Ist dies die letzte Task?
                cmp         akt_task,(d0)
                add  eq     resetvector,-,=      ;Reset auslösen

;Task aus der Taskliste entfernen
;d1: Zeiger auf Vorgängertask
                move        akt_task,d1          ;Suche nach Vorgängertask
.search:        add         #11,d1
                cmp         akt_task,(d1)        ;Ist nächste Task unsere gesuchte Task?
                move ne     (d1),d1
                jne         .search
                add         #11,akt_task,d0      ;Task aus Taskliste streichen
                move        (d0),(d1)

;Durch Task belegt Speicherbereich freigeben
;d1: Zeiger auf Speicherliste
                move        bfs,d1
.heapsearch:    inc         d1           ;Ist Speicherblock durch diese Task belegt?
                cmp         (d1),akt_task
                clr eq      (d1)         ;Ja, Speicherblock freigeben
                dec         d1
                add         (d1),d1      ;nächster Block
                cmp         d1,efs       ;letzter Block?
                jne         .heapsearch  ;Noch nicht
                jsr         mfree.melt   ;...Garbage-Collection in Speicherliste
                irqon
                jmp         switchtask   ;Ja, zur nächsten Task
```

12.3.5. Timerinitialisierung

```
;=============================================================================
;TIMER
;=============================================================================
;Der Timer-Interrupt wird 8000000/(2^19)-mal in der Sekunde ausgelößt.

;-----------------------------------------------------------------------------
;INITIALISIERUNG
;-----------------------------------------------------------------------------
timer:          clr         timerl       ;Register löschen
                clr         timerh
                clr         task_switch_tmp
                move        #timer_irq,5 ;Interruptroutine einstellen
                out         -,$7ff3      ;Timer on
                rts
```

12.3.6. DELAY

```
;------------------------------------------------------------------------------
;DELAY
;------------------------------------------------------------------------------
;d0: Zeitpunkt der Rückkehr, low
;d1: Zeitpunkt der Rückkehr, high
delay:          add        -,sp,d0              ;Timer-hi und -lo holen und...
                irqoff                           ;... gewünschte Wartezeit
                add     sf timerl,(d0),d0       ;addieren
                addc       timerh,-,d1
                irqon

.wait_hi:       cmp        d1,timerh            ;Warten, bis timer-high...
                jpl        .wait_low            ;geforderten Wert erreicht hat
                jsr        switchtask
                jmp        .wait_hi

.wait_low:      cmp        d0,timerl            ;ebenso für timer-low
                rts pl
                jsr        SWITCHTASK
                jmp        .wait_low
```

12.4. Speicherverwaltung

```
;==============================================================================
;HEAPVERWALTUNG
;==============================================================================
```

12.4.1. MALLOC

```
;------------------------------------------------------------------------------
;MALLOC
;------------------------------------------------------------------------------
;Arbeitsspeicher zur Verfügung stellen
;d0: Zeiger auf Speicherliste
;d1: Zeiger auf angeforderte Speichergröße
;d2: Gesuchte Speichergröße (+2 Worte Verwaltung)
;d3: Adresse des letzten freien Blocks
malloc:         irqoff                           ;Semaphor überprfüfung darf nicht unterbrochen
werden
                move    sf h_in_use,-            ;Heapverwaltung schon aktiv?
                move eq    akt_task,h_in_use     ;Falls nicht, belegen
                irqon
                jeq        .semok
;(Hier könnte ein Fairnessflag gesetzt werden!)
                jsr        SWITCHTASK
                jmp        malloc               ;Sonst warten

.semok:         push       d2
                push       d3
.retry:         move       bfs,d0       ;Adresse des 1. Blocks holen
                add        #3,sp,d1
                add        #2,(d1),d2   ;Soviel will der User (+2 Worte für Verwaltung)

;Block der angeforderten Größe suchen
.search:        cmp        d2,(d0)      ;Ist aktueller Block groß genug?
                jmi        .toosmall    ;Nein
                add        -,d0,d1      ;Ja, aber ist er frei?
                move    sf (d1),-
                jeq        .found       ;Auch das -> Block gefunden

.toosmall:      move       d0,d3        ;Adresse des letzten Blocks merken
                add        (d0),d0      ;Pointer auf nächsten Block...
                cmp        efs,d0       ;Gibt es gar keinen mehr?
                jne        .search      ;Doch, weitersuchen

;.............................................................................
;Auch der letzte Block war zu klein oder belegt -> Speicher vom SCSI-Cache
;zurückfordern
;Achtung! Hier ist eine Verbindung zum SCSI-Treiber!
                cmp        (d1),-       ;Ist dieser letzte Block frei?
                sub eq     (d3),d2      ;Ja, nur den fehlenden Speicher vom Cache anfordern
                add        efs,d2
                cmp        d2,#bs_start-6*260 ;Würde der Speicher reichen?
                move mi    #103,d0
                jmi        .errorend    ;Nein, Speicheranforderung nicht erfüllbar

;Treiberaufruf an SCSI um Cache freizugeben!
                push       d2           ;Sonst: Dec-Buffer-Job für einen SCSI-Driver aufbauen
                push       #5
                jsr        scsidriver0+5
```

```
                add         #2,sp
                add         -,d3,d0      ;Falls letzter Block belegt war, neuen Block anlegen
                cmp         (d0),-
                add ne      (d3),d3
                sub         d3,efs,(d3)
                inc         d3
                clr         (d3)         ;Block ist frei
                jmp         .retry       ;und nochmal versuchen
;.....................................................................
;Erst ab 16 verbleibenden Worten wird ein Block geteilt um keinen Verwaltungs-
;Overhead hervorzurufen.
;d1: Zeiger auf die Verwaltungsworte
;d3: Größe des freien Speicherblocks
.found:         sub         #16,(d0),d3          ;Soll Block geteilt werden?
                cmp         d2,d3
                jmi         .allocateblock       ;Nein, nur belegen

                move        (d0),d3              ;Bisherige Blockgröße holen
                move        d2,(d0)              ;Block wird so groß wie angefordert wurde
                add         d2,d0,d1             ;Adresse des neu entstandenen Blocks
                sub         d2,d3,(d1)           ;Dessen neue Größe eintragen
                inc         d1
                clr         (d1)                 ;neuer Block ist frei

;Als Blockbesitzer ist aktuelle Task eintragen
.allocateblock: inc         d0
                move        akt_task,(d0)        ;Aktuelle Tasknummer eintragen
                inc         d0                   ;Adresse des freien Bereiches
                clr     sf -                     ;N-Flag löschen

.errorend:      pop         d3
                pop         d2
                clr         h_in_use             ;Semaphor löschen
;(Hier müßte Fairnessüberprüfung eingesetzt werden!)
                rts
```

12.4.2. MFREE

```
;---------------------------------------------------------------------------
;MFREE
;---------------------------------------------------------------------------
;MFREE gibt Speicher frei
mfree:          irqoff                          ;Kein Unterbrechung bei Semaphorbehandlung
                move    sf h_in_use,-           ;Heapverwaltung schon aktiv?
                move eq     akt_task,h_in_use   ;Falls nicht, aktivieren
                irqon
                jeq         .semok
;(Hier könnte ein Fairnessflag gesetzt werden!)
                jsr         SWITCHTASK
                jmp         mfree        ;Sonst warten

.semok:         add         -,sp,d0
                sub         #2,(d0),d0   ;Adresse des freizugebenden Blocks

;Speicherblock suchen
;d0: Adresse des freizugebenden Blocks
;d1: Zeiger auf Speicherliste
                move        bfs,d1
.suchloop:      cmp         d1,d0        ;Block im Heap gefunden?
                jeq         .free        ;Ja, freigeben
                add         (d1),d1      ;Nächster Block
                cmp         d1,efs       ;War das der letzte?
                jne         .suchloop    ;Noch nicht
.illegal:       ror     sf -,-,-         ;Blockadresse war ungültig
                move        #101,d0
                jmp         .ende

;Gehört zu löschender Block der Task?
.free:          inc         d0
                cmp         (d0),akt_task ;Eigener Block?
                jne         .illegal     ;Nein, nicht löschen
                clr         (d0)         ;Block freigeben

;Hier werden nun 2 aufeinanderfolgende freie Blöcke zu einem verschmolzen
;.....................................................................
.melt:          move        bfs,d0       ;d0: Adresse des 1. Blocks
.meltloop:      add         d0,(d0),d1   ;d1: Adresse des 2. Blocks
                cmp         d1,efs       ;Gibt es den 2. Block gar nicht?
                jeq         .inc_buffer
                inc         d0           ;Zeiger auf Block-Frei-Wort
                cmp         (d0),-       ;1. Block frei?
                dec         d0
```

```
                jne         .notfree     ;nein
                inc         d1           ;Zeiger auf Block-Frei-Wort
                cmp         (d1),-       ;2. Block frei?
                dec         d1
                add eq      (d1),(d0)    ;Ja: Länge des 1. Blocks wird Summe der beiden
Blocklängen
.notfree:       add ne      (d0),d0      ;1. Zeiger weiterschalten
                jmp         .meltloop

;.....................................................................
;Hier ist eine Verbindung der Speicherverwaltung mit dem SCSI-Treiber!
;Falls letzter Block sehr groß ist, ein Teil davon als Cachebuffers freigeben
.inc_buffer:    cmp         (d0),#3000   ;Letzter Block sehr groß?
                jpl         .ende        ;Nein, keinen neuen Cachebuffer einrichten

                sub         #257,efs     ;Neuer Buffer
                clr         (efs)        ;Zugriffzähler=0
                sub         #3,efs
                clr         (efs)        ;Buffer ist frei
                sub         #260,(d0)    ;Block ist um soviel kleiner
                jmp         .inc_buffer
;.....................................................................
.ende:          clr         h_in_use
;(Hier müßte Fairnessüberprüfung eingesetzt werden!)
                rts
```

12.4.3. MSHRINK

```
;---------------------------------------------------------------------------
;MSHRINK
;---------------------------------------------------------------------------
mshrink:        irqoff                          ;Keine Unterbrechung
                move    sf h_in_use,-           ;Heapverwaltung schon aktiv?
                move eq    akt_task,h_in_use    ;Falls nicht, aktivieren
                irqon
                jeq         .semok
;(Hier könnte ein Fairnessflag gesetzt werden!)
                jsr         SWITCHTASK
                jmp         mshrink      ;Sonst warten

.semok:         add         -,sp,d1
                sub         #2,(d1),d0   ;Adresse des Blocks holen

;Block suchen
;d0: Adresse des zu verkleinernden Blocks
;d1: Zeiger auf Speicherliste
                move        bfs,d1
.suchloop:      cmp         d1,d0        ;Block im Heap gefunden?
                jeq         .shrink      ;Ja, verkleinern
                add         (d1),d1      ;Nächster Block
                cmp         d1,efs       ;War das der letzte?
                jne         .suchloop    ;Noch nicht
.illegal:       ror     sf -,-,-         ;Blockadresse war ungültig
                move        #101,d0
                jmp         mfree.ende

;Block zusammenschmelzen
;d1: neue Blockgröße
.shrink:        add         #2,sp,d1
                move        (d1),d1      ;neue Blockgröße holen
                inc         d0
                cmp         (d0),akt_task ;Eigener Block?
                jne         .illegal     ;Nein, nicht verkleinern
                dec         d0

                add         #3,d1        ;Soll Block geteilt werden?
                cmp         (d0),d1
                sub         #3,d1
                jpl         mfree.ende   ;Nein, der entstehende Block wäre zu klein

;d2: Größe des neu entstandenen Blocks
                push        d2
                sub         d1,(d0),d2
                add         #2,d1,(d0)   ;Neue Größe des verkleinerten Blocks
                add         (d0),d0
                sub         #2,d2,(d0)   ;Größe des neu entstehenden Blocks
                pop         d2
                inc         d0
                clr         (d0)         ;Neuer Block ist frei
                jmp         mfree.melt   ;Meltloop aufrufen

;---------------------------------------------------------------------------
```

```
;FIND_BLOCK
;---------------------------------------------------------------------------
;Nur interner Aufruf.
;Findet zu einer Adresse in d0 den zugehörigen Speicherblock
;d0: Zeiger auf Speicherliste
;d1: Adresse des zu suchenden Blocks
;d2: Speicherblockadresse gegen die geprüft wird
find_block:     push       d1
                push       d2
                move       d0,d1        ;Adresse im zu suchenden Block
                move       bfs,d0       ;Heapliste
.loop:          add        (d0),d0,d2   ;Ist aktueller Block der gesuchte?
                cmp        d1,d2
                jpe        .found
                add        (d0),d0      ;Nächsten Puffer betrachten
                cmp        efs,d0       ;letzter Block?
                jmi        .loop        ;Nein, weitersuchen
                ror     sf -,-,-        ;Speicherblock nicht gefunden
.found:         pop        d2
                pop        d1
                rts
```

12.4.4. MEMINFO

```
;---------------------------------------------------------------------------
;MEMINFO
;---------------------------------------------------------------------------
Ergebnisse:
;d0: Länge des größten Speicherblocks
;d1: Länge des freien Speichers insgesamt
;Registerverwendung:
;d2: Zeiger auf Speicherliste
meminfo:        push       d2           ;Ermittelt noch zur Verfüfgung stehenden Speicher
                push       d3
                push       d4
                push       d5
                push       d6
                clr        d0
                clr        d1
                clr        d5
                move       bfs,d2       ;Zeiger auf ersten Speicherblock

;Freie Blöcke suchen
;d3: Adresse des 2. Verwaltungswortes (Besitzer)
;d4: Adresse eines freien Blocks
;d5: Adresse des letzten Blocks der Liset
;d6: Adresse des nächsten Blocks
.loop:          add        -,d2,d3
                cmp        (d3),-       ;Speicherblock frei?
                jne        .next

;d3: Länge des Blocks
                sub        #2,(d2),d3
                add        d3,d1        ;ja,Gesmatspeicherfreizähler erhöhen
                cmp        d3,d0        ;Gefundener Block größer als bisher größter
                move mi    d3,d0        ;ja, eintragen
                move mi    d2,d4        ;Adresse des Blocks merken              add
d2,(d2),d6
                cmp        d6,efs       ;Ist dieser Block der letzte freie?
                move eq    d3,d5        ;ja, Länge merken

.next:          add        (d2),d2
                cmp        d2,efs       ;Ende des Speichers erreicht?
                jpe        .loop

;.........................................................................
;Speicherplatz bis zum Minimal-Cache hinzu addieren
;Achtung! Verbindung der Speicherverwaltung mit dem SCSI-Treiber (Cache)!
;d2: Verbleibender Speicher ohne Cache
;d3: Adresse des letzten freien Blocks
                sub        efs,#bs_start-6*260,d2       ;Speicherplatz ohne Cache-Puffer
                add        d2,d1                        ;Insgesamt freier Speicher
                add        d4,(d4),d3
                cmp        d3,efs       ;Schloß der längste freie Block an efs an?
                add eq     d2,d0        ;ja, dann Cache-Speicher hinzuaddieren
                jeq        .ende

                add        d5,d2
                cmp        d2,d0        ;Ist der letzte freie+Cache-Speicher größer als der
bisher größte Block?
                move mi    d2,d0        ;Ja
;.........................................................................
```

```
.ende:          pop        d6
                pop        d5
                pop        d4
                pop        d3
                pop        d2
                clr     sf ~-
                rts

dw "Hardware und Software entwickelt von Uwe Reinhardt und Oliver Henning, 1993"
```

12.5. Dateiverwaltung

```
;===============================================================================
;FILE VERWALTUNG
;===============================================================================
include file_sub

equ primaersektor_hi,0
equ primaersektor_lo,1
equ dateilng_sektor,2
equ dateilng_offset,3
equ tiefe,4
equ flags,5
equ kennung_hi,6
equ kennung_lo,7
equ dateiname,8

equ kontrollflags,1
equ driver,2
equ verweis_hi, 3
equ verweis_lo,4
equ verweis_worte,5
equ dateizeiger_sektor,6
equ dateizeiger_offset,7
```

12.5.1. OPEN

```
;-------------------------------------------------------------------------------
;DATEI ÖFFNEN
;-------------------------------------------------------------------------------
;DATEI ÖFFNEN öffnet eine Datei und überprfüt Zugriffskonflikte. Ein FDB wird
;angelegt.
;Typ:   1: lesen
;       2: lesen und schreiben
;       3: neuschreiben
;       4: anhängen

;Ressource "Fileverwaltung" allokieren
open:           irqoff                          ;Unterbrechung verhindern
                cmp        fdblist_in_use,-     ;Semaphor überprüfen
                move eq    akt_task,fdblist_in_use       ;
                irqon
                jeq        .semok
;(Hier könnte ein Fairnessflag gesetzt werden!)
                jsr        SWITCHTASK
                jmp        open

;Parameter holen
.semok:         add        -,sp,d0
                pushall
                move       (d0),d6      ;d6: Zeiger auf Dateinamen holen
                inc        d0
                move       (d0),d5      ;d5: Öffnungsmodus holen


;Standard-FDB aufbauen (Teile der Flags, Magic und Rest auf 0 setzen)
                push       #36           ;Speicher für FDB und temporären...
                jsr        MALLOC        ;...Arbeitsspeicher holen
                inc        sp
                jmi        .ende
                move       d0,d7         ;d7:Zeiger auf neuen FDB

;Angegebener Modus zwischen 1 und 4?
                cmp        -,d5
                move mi    #111,d0
                jmi        .memfree
                cmp        d5,#4
                move mi    #111,d0       ;Fehler 111: Parameter falsch!
                jmi        .memfree

                inc        d0
```

```
                move       #6,(d0)      ;Erst mal Read- und Writeflag setzen setzen
                bswp       -,d5
                or         d5,(d0)      ;Im höherwertigen Flagteil: Öffnungsmodus
                move       #6,d1        ;6 Wörter im FDB löschen
                inc        d0
.clrloop:       clr        (d0)
                inc        d0
                dec sf     d1
                jne        .clrloop
                move       #$1234,(d0)  ;Magic-Worte setzen
                inc        d0
                move       #$5678,(d0)

;Dateinamen vorbereiten (evt. gehört CWD zum Pfad)
;d2: Flag ob CWD mit betrachtet wird oder nicht
                clr        d2           ;Flag "Autoerweiterung" löschen
                cmp        #$2f,(d6)    ;Dateiname um cwd erweitern ?
                jeq        .start               ;Nein, Pfad angegeben (Start mit "/")

;d2 wird jetzt Zeiger auf den Dateinamen und d6 Zeiger auf CWD
                move       d6,d2        ;Zeiger auf Dateiname retten (=Flag setzen)
                push       #cwd         ;CWD= im Environment suchen
                jsr        SEARCH_ENV
                inc        sp
                jmi        .memfree     ;Gibt es nicht -> Speicher wieder frei und Ende
                move       d0,d6        ;Erst mal CWD als Dateinamen benutzen

;Suche nach dem zugehörigen Gerätetreiber
;d0: Zähler
;d1: Zeiger auf Arbeitsspeicher
.start:         inc        d6           ;Treibernamen in Puffer kopieren
                add        #18,d7,d1    ;Dateinamen in letzten Teil des Arbeitsbereiches
eintragen
                move       #5,d0        ;Maximal 5 Zeichen kopieren
.drvloop:       move    sf (d6),(d1)    ;1 Zeichen kopieren
                jeq        .searchdrv   ;Stringende->Treiber suchen
                cmp        #$2f,(d1)    ;War das Zeichen ein "/"?
                clr eq     (d1)         ;Ja, dort ist nun Stringende...
                jeq        .searchdrv   ;...und Treiber suchen
                inc        d6
                inc        d1
                dec     sf d0
                jne        .drvloop

.drvnotfound:   move       #106,d0      ;Device existiert nicht!
                jmp        .memfree

.searchdrv:     add        #18,d7,d1    ;Treiber suchen
                push       d1
                jsr        SEARCHDRIVER
                inc        sp
                jmi        .drvnotfound ;Device nicht gefunden?
                add        #2,d7,d1
                move       d0,(d1)      ;Treiberadresse in FDB eintragen

;Dispatchen: Zeichen- oder Blockdriver
                dec        d0           ;Zeiger auf Devicetype
                dec        d1           ;Typ in FDB-Flags eintragen
                or         (d0),(d1)
                move    sf (d0),-       ;Blockdevice?
                jeq        .blockdevice ;Ja, Directorybaum durchlaufen
                cmp        (d6),-       ;Nein, Dateiname muß hier enden
                move ne    #104,d0      ;Sonst: Dateiname war falsch
                jne        .memfree

;Es handelt sich hier um ein Zeichendevice.
;Zugriffskonflikt überprüfen.
                push       d7           ;Gerät schon geöffnet?
                jsr        FDB_CHECK_X
                inc        sp
                and     sf #6,d0,-
                jeq        .append      ;Nein, alles ok
                move       #115,d0      ;Ja, Zugriffskonflikt
                jmp        .memfree

;.....................................................................
;Blockdevice -> Directorybaum durchlaufen
;d0: Zeiger auf Verweis
.blockdevice:   add        #5,d7,d0     ;Laden des Directory-Entrys in FDB
                push       #8           ;Wir wollen 8 Worte (1/2 Direcory-Entry)
                push       (d0)         ;Ab Wort Nummer
                dec        d0
```

```
                push       (d0)          ;Sektornummer lo
                dec        d0
                push       (d0)          ;Sektornummer hi
                add        #10,d7,d1
                push       d1            ;Pufferadresse
                push       -             ;Job 1:lesen
                dec        d0
                driver     (d0)
                add        #6,sp
                jmi        .memfree

;Pfadende erreicht?
                cmp        (d6),-        ;Pfadname zu Ende?
                jeq        .exist        ;Ja, fertig
                add        #15,d7,d0     ;Ist das überhaupt eine Directory?
                and     sf -,(d0),-
                move eq    #113,d0       ;Dir-not-exist Fehler
                jeq        .memfree
                inc        d6
                cmp        (d6),-        ;War das vielleicht erst die CWD?
                move eq    d2,d6         ;Ja, jetzt kommt dann der Dateiname

;Nein, nächste Pfadstück bearbeiten
                add        #28,d7,d3     ;Nächstes Pfadstück in Arbeitsbereich packen
                push       d3           ;Zeiger auf Zielpuffer
                push       d6           ;Zeiger auf Dateinamen
                jsr        SCAN          ;naechsten Teil des Dateinamens lesen
                add        #2,sp
                jmi        .memfree
                move       d0,d6         ;Zeiger auf nächsten Teilpfad setzen

;Pfadstück in Verzeichnis suchen
                push       d3            ;Zeiger auf den Namen
                push       d7           ;FDB-Adresse
                jsr        SEARCH_NEXT   ;sucht Dateinamen im Directory
                add        #2,sp
                jmi        .exist_nicht ;Dateiname nicht gefunden

;Adresse des Verweises bestimmen
;d3: Zeiger auf Verweis im FDB
.verweis:       push       d7            ;Bestimmung des Verweises
                jsr        LOG_ABS       ;Absolute Sektornummer bestimmen
                inc        sp
                add        #3,d7,d3
                move       d0,(d3)       ;SektorNummer-high im FDB eintragen
                inc        d3
                move       d1,(d3)       ;ebenso SN-low
                add        #3,d3
                and        #255,(d3),d0  ;ebenso relative Pos im Sektor
                clr        (d3)          ;Dateizeiger auf 0 setzen
                dec        d3
                clr        (d3)
                dec        d3
                move       d0,(d3)
                jmp        .blockdevice  ;Und weiter im Pfad

;Datei existiert: Fehler melden bei Neuschreiben
;d0: Flags im FDB
.exist:         add        -,d7,d0       ;Adresse der Flags
                and     sf #$300,(d0),d0
                cmp        #$300,d0      ;Datei neuschreiben?
                move eq    #110,d0       ;Datei-existiert-schon Fehler
                jeq        .memfree

                add        -,d7,d0       ;Datei zum schreiben öffnen?
                and     sf #$100,(d0),-
                jne        .sharetest    ;nein
                add        #15,d7,d0
                and     sf #4,(d0),-     ;Ja, ist Datei schreibgeschützt?
                jeq        .sharetest    ;nein
                move ne    #114,d0       ;Ja, Fehler: Datei schreibgeschützt
                jne        .memfree

;Test auf Zugriffskonflikt mit anderen Tasks
.sharetest:     push       d7            ;Ist diese Datei schon geöffnet?
                jsr        FDB_CHECK_X
                inc        sp
                and     sf #4,d0,-       ;zum Schreiben?
                move ne    #115,d0       ;Ja: Zugriffskonflikt
                jne        .memfree
                and     sf #2,d0,-       ;zum Lesen?
                jeq        .append       ;Nein, alles ok
```

```
                add         -,d7,d1
                bswp        (d1),-,d1     ;Ja, aber wollen wir schreiben?
                cmp         -,d1
                move ne     #115,d0       ;Auch das: Zugriffskonflikt
                jne         .memfree

;Read/Write-Flags und Filepos initialisieren
.append:        add         -,d7,d1       ;d1: Pointer auf Flags
                bswp        (d1),-,d0     ;Öffnungsmodus nach d0
                cmp         #2,d0         ;Welcher Öffnungsmodus?
                and pe      #5,(d1)       ;Nur Schreiben -> Readflag löschen
                and mi      #3,(d1)       ;Nur lesen -> Schreibflag löschen

                and     sf #4,d0,-        ;Soll angehängt werden?
                jeq         .insert       ;Nein

                add         #6,d7,d0      ;Filepos auf Dateilänge setzen
                add         #11,d1
                move        (d1),(d0)     ;Maximale Sektornummer
                inc         d0
                inc         d1
                move        (d1),(d0)     ;Maximale Wortnummer in diesem Sektor

;Neuen FDB in FDB-List eintragen
.insert:        move        fdblist,d0    ;FDB am Anfang der Liste eintragen
                move        d7,fdblist
                move        d0,(d7)

                push        #20           ;FDB auf 20 Worte reduzieren
                push        d7
                jsr         MSHRINK
                add         #2,sp

                clr     sf ~-
                move        d7,d0

.ende:          clr         fdblist_in_use        ;Semaphor löschen
;(Hier müßte Fairnessüberprüfung eingesetzt werden!)
                popall
                rts

;.....................................................................
;Datei existiert nicht und muß eventuell neu erstellt werden
.exist_nicht:   cmp        (d6),-                 ;Nein, Fehler
                jne         .memfree

                add         -,d7,d0       ;d0: Zeiger auf Flags in FDB
                bswp        (d0),-,d0     ;Nur zum Lesen öffnen?
                cmp         -,d0
                move eq     #109,d0       ;Ja, Datei nicht gefunden
                jeq         .memfree

                cmp         #3,d0         ;Neuschreiben?
                jne         .anfang       ;Nein
                add         -,d7,d0
                and         #$ff,(d0)
                or          #$0400,(d0)   ;JA, jetzt nur Anhängen eintragen

;Directory-Eintrag erstellen (Directory ist momentan geöffnet)
.anfang:        add         #6,d7,d0      ;Auf Dateianfang+8 positionieren
                clr         (d0)
                inc         d0
                move        #8,(d0)

;Freien Platz im Verzeichnis suchen
.lesen:         push        -             ;1 Wort aus Directory lesen
                dec         sp            ;Dummy beim Wort lesen
                push        d7            ;FDB-Adresse
                jsr         READ
                add         #3,sp
                jmi         .found                ;EOF? Kein Eintrag gefunden, Dir vergrößern
                cmp         d0,-          ;Eintrag frei?
                jeq         .found                ;Ja freien Eintrag gefunden

                pushz                     ;relativ positionieren
                push        d7            ;FDB-Adresse
                pushz                     ;Worte hi
                push        #15           ;Worte lo
                jsr         SET_POS
                add         #4,sp
                jpl         .lesen        ;Nächsten Eintrag überprüfen
```

```
.found:         add        #7,d7,d0      ;An den Anfang des Entrys positionieren
                and        #$fff0,(d0)

;Verzeichniseintrag erstellen
                add        #20,d7,d0     ;Eintrag für neue Datei erstellen
                clr        (d0)          ;Primaersektor hi
                inc        d0
                clr        (d0)          ;Primaersektor lo
                inc        d0
                clr        (d0)          ;Dateilaenge hi
                inc        d0
                clr        (d0)          ;Dateilaenge lo
                inc        d0
                clr        (d0)          ;Schachtelungstiefe
                inc        d0
                clr        (d0)          ;Flags
                inc        d0
                move       timerh,(d0)   ;Kennung
                inc        d0
                move       timerl,(d0)   ;Kennung
                push       #16           ;Anzahl der Worte
                add        #20,d7,d0
                push       d0            ;Pufferzeiger
                push       d7            ;FDB-Adresse
                jsr        WRITE         ;Dateieintrag ablegen
                add        #3,sp
                jmi        .memfree

                add        #7,d7,d0      ;An den Anfang des Entrys positionieren
                cmp        (d0),-        ;Wurde ein neuer Sektor erreicht?
                sub ne     #16,(d0)      ;Nein
                move eq    #$f0,(d0)     ;doch
                dec  eq    d0
                dec  eq    (d0)          ;Sektornummer dec

                jmp        .verweis      ;Pfadschleife erneut betreten

;Fehler aufgetreten, Speicher wieder freigeben
.memfree:       move       d0,d6         ;Fehlercode retten
                push       d7            ;Arbeitsspeicher wieder freigeben
                jsr        MFREE
                inc        sp
                move       d6,d0         ;Fehlercode zurück
                ror     sf -,-,-

                jmp        .ende
```

12.5.2. CLOSE

```
;---------------------------------------------------------------------------
;DATEI SCHLIEßEN
;---------------------------------------------------------------------------
;Ressource "Fileverwaltung" allokieren
close:          irqoff                          ;Semaphor setzen
                cmp        fdblist_in_use,-
                move eq    akt_task,fdblist_in_use
                irqon
                jeq        .semok
;(Hier könnte ein Fairnessflag gesetzt werden!)
                jsr        SWITCHTASK
                jmp        close

.semok:         add        -,sp,d0
                push       d2
                move       (d0),d2       ;d2: FDB-Adresse

;FDB in Liste suchen, dabei Nachfolger bestimmen
;d1: Zeiger auf FDB-Liste
                move       #fdblist,d1   ;Zeiger auf Zeiger auf 1. FDB
.search:        cmp        (d1),-        ;Adresse des nächsten FDB testen
                jeq        .error                ;Das war der letzte, gesuchten FDB nicht
gefunden
                cmp        (d1),d2       ;Ist Nachfolger der gesuchte FDB?
                move ne    (d1),d1       ;Nein, nächsten FDB untersuchen
                jne        .search

                sub        -,d2,d0       ;Gehört uns der FDB überhaupt?
                cmp        (d0),akt_task
                jne        .error        ;Nein, nichts tun

                move       (d2),(d1)     ;Unser Nachfolger wird Nachfolger unseres Vorgänger
                add        #2,d2,d0
```

```
;Alle Schreibpuffer für dieses Device zurückschreiben
                push       #6           ;Flush
                driver     (d0)         ;Treiberaufruf
                inc        sp            ;Keine Fehlerüberprüfung, da Job nicht
                                         ;allen Treibern bekannt

;Magicworte löschen
                add        #8,d2,d0
                clr        (d0)
                inc        d0
                clr        (d0)

;FDB wieder freigeben
                push       d2
                jsr        MFREE
                inc        sp
                clr     sf ~-

.ende:          clr        fdblist_in_use
;(Hier müßte Fairnessüberprüfung eingesetzt werden!)
                pop        d2
                rts

.error:         ror     sf -,-,-
                move       #101,d0      ;Ungültiges Handle
                jmp        .ende
```

12.5.3. DELETE

```
;-------------------------------------------------------------------------
;DATEI LÖSCHEN
;-------------------------------------------------------------------------
;Löscht eine Datei in einem Verzeichnis. Es wird eine 0 als 1. und 2.
;Dateinamenszeichen geschrieben. Die Größe der Dir wird nicht verändert.
;Beim Öffnen einer neuen Datei wird dieser Eintrag dann neu beschrieben.
;Es kann dadurch zu uneindeutigen Aussagen über die Länge der Directory kommen.
;Es werden der Reihe nach alle Datensektoren gelöscht, danach die Tiefe
;decrementiert. Somit sind die Verwaltungsektoren der letzten Stufe jetzt die
;neuen Datensektoren. Dieser Vorgang wird solang wiederholt bis die Tiefe
;gleich Null ist. Dann wird der DIR-Eintrag und der Primärsektor
;gelöscht. Vorsicht! Beim Löschen einer DIR müssen vorher alle Datei gelöscht
;werden. Sonst werden die durch die Dateien belegten Sektoren nicht wieder
;frei gegeben.
delete:         add        -,sp,d0
                pushall
                move       (d0),d0      ;d0:FDB-Adresse

;Ist die FDB-Adresse gültig?
                jsr        FDB_ADRESS
                jmi        .ende
                move       d0,d7        ;d7:FDB-Adresse

;Ist dies auch eine Blockdevice?
                add        -,d0
                and     sf -,(d0),-     ;Zeichendevice?
                move ne    #102,d0      ;Ja, Befehlszugriff nicht erlaubt
                jne        .fehler

;Zeiger intialisieren (Dateizeiger, Schachtelungstiefe,..)
                add        #2,d7,d3     ;d3=Treiberadresse
                add        #dateizeiger_sektor,d7,d6

                add        #14,d7,d2    ;d2: Zeiger auf Schachtelungstiefe
                move       (d2),d2

                push       -            ;Auf Dateianfang positionieren
                push       d7
                pushz
                pushz
                jsr        SET_POS
                add        #4,sp

                add        #12,d7,d5    ;d5:Zeiger auf dateilng_sektor
                inc        (d5)

;.........................................................................
;Dateilänge der reduzierten Schachtelungstiefe bestimmen
.tiefe:         move    sf d2,-
                jmi        .clr_v
                add        #12,d7,d0    ;d5:Zeiger auf dateilng_sektor
                move       (d0),d4
```

```
                lsr        d4           ;/2
                lsr        d4           ;/4
                lsr        d4           ;/8
                lsr        d4           ;/16
                lsr        d4           ;/32
                lsr        d4           ;/64
                lsr        d4           ;/128
                inc        d4           ;d4: Neue Dateilaenge
                move       (d0),d5
                move       d4,(d0)      ;Neue Dateilänge eintragen

;Alle Datensektoren dieser Schachtelungstiefe löschen
.lng:           cmp        d5,-          ;aktuellle Dateilaenge=0?
                jeq        .exit

;Absolute Sektornummer bestimmen
                push       d7           ;FDB-Adresse
                jsr        LOG_ABS
                inc        sp
                jmi        .clr_v

;Sektor freigeben
                push       d0           ;high
                push       d1           ;low
                push       d7           ;FDB-Adresse
                jsr        BLOCK_FREE   ;Sektor freigeben
                add        #3,sp
                jmi        .fehler
                inc        (d6)         ;Dateizeiger_Sektor dec
                dec        d5           ;Sektorzähler dec
                jmp        .lng

;Schachtelungstiefe reduzieren
.exit:          add        #14,d7,d2     ;d2: Zeiger auf Schachtelungstiefe
                dec        (d2)
                move       (d2),d2
                clr        (d6)
                jmp        .tiefe

;.....................................................................
;Directoryeintrag löschen
.clr_v:         add        #verweis_hi,d7,d0    ;Eintrag des Primaersektors suchen
                move       (d0),d2
                inc        d0
                move       (d0),d5
                inc        d0
                add        #8,(d0),d0    ;Adresse des zu löschenden Wortes
                push       -             ;Anzahl der Worte
                push       d0           ;Ab Position
                push       d5           ;Sektornummer lo
                push       d2           ;Sektornummer hi
                push       #.null                ;Pufferadresse
                push       #2           ;Job 2:schreiben
                driver     (d3)
                add        #6,sp
                jmi        .ende

                clr     sf ~-
.ende:          popall
                rts

.fehler:        ror     sf -,-,-
                jmp        .ende

.null:  dw 0
```

12.5.4. RENAME

```
;---------------------------------------------------------------------------
;DATEI UMBENNEN
;---------------------------------------------------------------------------
;DATEI UMBENENNEN
;(aktueller Dateiname, neuer Dateiname):ok
rename:         add        -,sp,d0
                push       d2
                push       d3
                push       d4
                move       (d0),d3      ;d3: Zeiger auf neuen Namen
                inc        d0
                move       (d0),d0       ;d0: FDB-Adresse

;Ist FDB-Adresse gültig?
```

```
                jsr         FDB_ADRESS
                jmi         .ende
                move        d0,d2

;Ist dies auch eine Blockdevice?
                inc         d0              ;d0: Zeiger auf FDB, Flags
                and     sf -,(d0),-     ;Blockdevice?
                move ne     #102,d0         ;Nein, Zugriff nicht erlaubt
                ror  ne sf -,-,-
                jmi         .ende ;nein!

;Speicher für neuen Namen allokieren
                push        #8
                jsr         MALLOC
                inc         sp
                jmi         .ende
                move        d0,d4

;Neuen Namen auf zulässige Zeichen überprüfen und kompremieren
                push        d4              ;Zeiger auf 8-Wort Namensfeld
                push        d3              ;Zeiger auf neuen Namen
                jsr         SCAN            ;Dateiname überprüfen
                add         #2,sp
                jmi         .fehler

;neuen Dateinamen eintragen
                push        #8              ;8 Worte schreiben
                add         #verweis_worte,d2,d0
                add         #8,(d0),d1
                push        d1              ;Ab Position
                dec         d0
                push        (d0)            ;Sektornummer lo
                dec         d0
                push        (d0)            ;Sektornummer hi
                push        d4              ;Pufferzeiger
                push        #2              ;Job 2:schreiben
                dec         d0
                driver      (d0)            ;Treiberaufruf
                add         #6,sp
                jmi         .fehler

                clr         sf ~-           ;Flags löschen
                push        d4
                jsr         MFREE
                inc         sp

.ende:          pop         d4
                pop         d3
                pop         d2
                rts

.fehler:        move        d0,d2           ;Fehlercode retten
                push        d4
                jsr         MFREE
                inc         sp
                move        d2,d0           ;fehlercode restaurieren
                ror      sf -,-,-           ;N-Flag setzen
                jmp         .ende
```

12.5.5. SET FLAGS

```
;---------------------------------------------------------------------------
;DATEIFLAGS
;---------------------------------------------------------------------------
;DATEIFLAGS SETZEN
set_flags:      add         -,sp,d0
                move        (d0),d0         ;d0:Zeiger auf Dateinamen

;Ist FDB-Adresse gültig?
                jsr         FDB_ADRESS
                jmi         .ende

;Ist dies auch eine Blockdevice?
                add         -,d0,d1         ;d0 zeigt auf Kontrollflags
                and     sf -,(d1),-         ;Block-Oder Zeichendevice
                move ne     #102,d0         ;Zeichendevice=Fehler
                ror  ne sf -,-,-
                jmi         .ende

;Dateiflags schreiben
                push        -                       ;1 Wort
                add         #verweis_worte,d0       ;d0:Zeiger auf Verweis_worte
```

```
                add         #5,(d0),d1            ;Zeiger auf Dateiflags
                push        d1                    ;Ab Position
                dec         d0
                push        (d0)          ;Sektornummer lo
                dec         d0
                push        (d0)          ;Sektornummer hi
                add         #6,sp,d1      ;Adresse der FLags auf dem Stack
                push        d1            ;Pufferadresse
                push        #2            ;Job 2:schreiben
                dec         d0
                driver      (d0)
                add         #6,sp
                jmi         .ende

                clr     sf ~-             ;Flags löschen
.ende:          rts
```

12.5.6. GET FLAGS

```
;-------------------------------------------------------------------------------
;GET_FLAGS
;-------------------------------------------------------------------------------
;DATEIFLAGS laden
get_flags:      add         -,sp,d0
                move        (d0),d0      ;d0:Zeiger auf Dateinamen

;Ist FDB-Adresse gültig?
                jsr         FDB_ADRESS
                jmi         .ende

;Ist dies auch eine Blockdevice?
                add         -,d0          ;d0 zeigt auf Kontrollflags
                and     sf -,(d0),-       ;Block-Oder Zeichendevice?
                move ne     #102,d0       ;Zeichendevice=Fehler
                ror  ne sf -,-,-
                jmi         .ende

                add eq      #14,d0        ;Zeiger auf Dateiflags
                move eq     (d0),d0       ;Dateiflags nach d0
.ende:          rts
```

12.5.7. SET POSITION

```
;-------------------------------------------------------------------------------
;DATENZEIGERMANIPULATION
;-------------------------------------------------------------------------------
;POSITIONSAUFRUF(0=REL,1=ABS;FDB-ADRESSE;ANZAHL DER WORTE HIGH; LOW)
set_pos:        add         -,sp,d0
                push        d2
                push        d3
                push        d4
                push        d5
                push        d6
                move        (d0),d4       ;d4: Anzahl der worte,l
                inc         d0
                move        (d0),d3       ;d3: Anzahl der worte,h
                inc         d0
                move        (d0),d2       ;d2: FDB-Adresse holen
                inc         d0
                move        (d0),d5       ;d5: Rel oder Abs?

;Ist FDB-Adresse gültig?
                move        d2,d0
                jsr         FDB_ADRESS
                jmi         .ende

;Ist dies auch eine Blockdevice?
                add         -,d2,d0       ;Zeiger auf Flags
                and     sf -,(d0),-       ;Zeichen-Device?
                move ne     #102,d0       ;Ja, kein Positionieren möglich
                ror  ne sf -,-,-
                jmi         .ende

                add         #6,d2,d0      ;Dateizeiger Sektor
                add         -,d0,d1       ;Dateizeiger Wort

                cmp         d5,-          ;Relative Positionierung
                jne         .calc         ;Nein, absolut

;POSITIONIEREN RELATIV VORWÄRTS;
                bswp        -,(d0),d5
                or          (d1),d5       ;Zeiger lo in Worten
```

```
                add     sf d5,d4 ;low
                bswp       (d0),-,d5     ;Zeiger hi in Worten
                addc       d5,d3

;POSITIONIEREN ABSOLUT DATEIZEIGER (handle, anzahl der worte)
.calc:          bswp       d4,d3,d5      ;d5=Sektoren
                and        #$ff,d4,d6    ;d6=Worte

;POSITIONIEREN ABSOLUT DATEIZEIGER (handle, Sektoren, Worte)
.abs:           add        #12,d2,d3     ;Zeiger auf Dateilänge
                add        -,d3,d4
                cmp        d5,(d3)       ;Sektorlänge möglich?
                jpe        .pos          ;ja
                cmp        d6,(d4)       ;Wortlänge möglich?
                jpl        .pos          ;ja

                move       (d3),(d0)     ;Dateizeiger auf Fileende
                move       (d4),(d1)
                ror     sf -,-,-         ;N-Flag setzen
                move       #105,d0       ;EOF erreicht
                jmp        .ende

.pos:           move       d5,(d0)       ;Dateizeiger positionieren
                move       d6,(d1)       ;Dateizeiger positionieren
                clr     sf ~-            ;Flags löschen
.ende:          pop        d6
                pop        d5
                pop        d4
                pop        d3
                pop        d2
                rts
```

12.5.8. GET POSITION

```
;---------------------------------------------------------------------------
;GET_POS
;---------------------------------------------------------------------------
;POSITON DATEIZEIGER IN WORTEN ERMITTELN (FDB-ADRESSE)
;       :d0 Position in Worten high
;       :d1 Position in Worten low
get_pos:add        -,sp,d1
                push       d2
                move       (d1),d2       ;d2: FDB-Adresse

;Ist FDB-Adresse gültig?
                move       d2,d0
                jsr        FDB_ADRESS
                jmi        .ende

;Ist dies auch eine Blockdevice?
                add        -,d2,d0       ;Zeiger auf Flags
                and     sf -,(d0),-      ;Blockdevice?
                move ne    #102,d0       ;Nein, Zugriff nicht erlaubt
                ror  ne sf -,-,-         ;N-Flag setzen
                jmi        .ende

;Dateiposition übermitteln
;Z-Flags ist hier gesetzt
                add        #dateizeiger_sektor,d2
                bswp       (d2),-,d0     ;Hi-Wort
                bswp       -,(d2),d1     ;Lo-Wort
                inc        d2
                or         (d2),d1
.ende:          pop        d2
                rts
```

12.5.9. GET LENGTH

```
;---------------------------------------------------------------------------
;GET_LNG
;---------------------------------------------------------------------------
;DATEILÄNGE ERMITELN(FDB-ADRESSE):d0 Länge in Worten high
;                                :d1 Länge in Worten low
get_lng:        add        -,sp,d0

;Ist FDB-Adresse gültig?
                move       (d0),d0
                jsr        FDB_ADRESS
                jmi        .ende

;Ist dies auch eine Blockdevice?
                add        -,d0,d1       ;d1:Zeiger auf Treiberadresse
```

```
                and     sf -,(d1),-     ;BLock- oder Zeichendevice?
                move ne    #102,d0      ;Zeichendevice!, Zugriff nicht erlaubt!
                ror  ne sf -,-,-        ;N-Flag setzen
                jmi        .ende        ;Nein!

;Dateilänge übermitteln.
;Z-Falg ist hier gesetzt.
                add        #12,d0,d1    ;Zeiger auf Dateilänge_sektor
                bswp       (d1),-,d0    ;Hi-Wort
                bswp       -,(d1),d1    ;Lo-Wort
                inc        d0
                or         (d0),d1
.ende:          rts
```

12.5.10. READ

```
;---------------------------------------------------------------------------
;TRANSFER
;---------------------------------------------------------------------------
;READ(FDB-ADRESSE,ZEIGER AUF DATEN,ANZAHL=0:STRING,=1 WORT, x=BLOCK)
read:           add        -,sp,d0
                pushall
                move    sf (d0),d7              ;d7:FDB-Adresse (0=Stdout)
                add eq     #20,akt_task,d1      ;Adresse des Stdin-Handles
                move eq sf (d1),d7
                move eq    #105,d0              ;Nulldevice -> EOF und Ende
                jeq        transferende.fehler
                move       d0,d2

;Ist FDB-Adresse gültig?
                move       d7,d0                ;FDB-Adresse zulässig?
                jsr        FDB_ADRESS
                jmi        transferende.fehler

;Zugriff erlaubt? (Datei zum lesen geöffnet?)
                add        -,d7,d1              ;Read-Flag gesetzt?
                and     sf #2,(d1),-
                move eq    #102,d0              ;Nein, Zugriff nicht erlaubt
                jeq        transferende.fehler
                add        #2,d7,d5             ;d5:Treiberadresse
                move       (d5),d5

                inc        d2
                move       (d2),d6              ;d6:Pufferadresse
                inc        d2
                move    sf (d2),d2              ;d2:Anzahl
                move eq    #111,d0              ;Anzahl=0: Falscher Parameter
                jeq        transferende.fehler
                cmp        -,d2
                jeq        wortread             ;=1 WORT

;.........................................................................
;sonst BLOCK EINLESEN
blockread:      and     sf -,(d1),-     ;Block oder Zeichendevice?
                jeq        .blockdevice
                move       d2,d4        ;Anzahl merken

.zeichendevice: push       -            ;1 Zeichen vom Device holen
                driver     d5
                inc        sp
                sub mi     d2,d4,d1     ;d1 enthält Anzahl der übermittelten Zeichen
                jmi        transferende.fehler
                move       d0,(d6)      ;Alles ok, Wort in Puffer schreiben
                inc        d6           ;Pufferzeiger++
                dec     sf d2           ;Anzahl--
                jne        .zeichendevice
                move       d4,d1        ;es wurden alle Zeichen übertragen
                jmp        transferende

.blockdevice:   clr        d1
.loop:          cmp        d2,-         ;letztes Zeichen kopiert
                jeq        transferende ;Ja, fertig
                add        #7,d7,d0
                sub        (d0),#256,d4 ;Anzahl der übertragbaren Worte im aktuellen Sektor
                cmp        d2,d4        ;Reicht der Platz im aktuellen Sektor?
                sub  mi    d4,d2        ;Nein, neue Länge=Alte Länge-Anzahl übertragener
Worte
                move pl    d2,d4        ;Ja, nur soviel übertragen wie nötig
                clr  pl    d2           ;...und dann nichts mehr
                push       d1
                jsr        IN
                pop        d3
```

```
                add         d3,d1         ;Aufsummieren der übertragenen Wörter
                jmi         transferende.e2
                add         d4,d6         ;Pufferzeiger um Anzahl übertragener Worte
weiterschalten
                jmp         .loop

;.....................................................................
;WORTLESEN
wortread:       and     sf -,(d1),-             ;Block oder Zeichendevice?
                jeq         .blockdevice        ;Block!

                push        -                   ;Job 1:Zeichen (Status) lesen
                driver      d5                  ;Teiberaufruf
                inc         sp
                jmp         transferende.e2     ;Zeichen in d0

.blockdevice:   move        -,d4          ;Anzahl der zulesenden Worte=1
                add         #18,d7,d6     ;Pufferbereich (im FDB)
                jsr         IN            ;Read-Routine
                move pl     (d6),d0       ;Ergebnis aus Puffer nach d0
                jmp         transferende.e2
```

12.5.11. WRITE

```
;---------------------------------------------------------------------------
;WRITE(FDB-ADRESSE,ZEIGER AUF PUFFER,ANZAHL=0:STRING,=1 WORT, x=BLOCK)
write:          add         -,sp,d0
                pushall
                move    sf (d0),d7              ;d7:FDB-Adresse (0=Stdout)
                add eq      #21,akt_task,d1     ;Adresse des Stdout-FDB holen
                move eq sf (d1),d7
                jeq         transferende        ;Nulldevice? -> Fertig
                move        d0,d2

;Ist FDB-Adresse gültig?
                move        d7,d0               ;FDB-Adresse zulässig?
                jsr         FDB_ADRESS
                jmi         transferende.fehler

                add         -,d7,d1             ;Schreib-Flag gesetzt?
                and     sf #4,(d1),-
                move eq     #102,d0             ;Nein: Unerlaubter Zugriff
                jeq         transferende.fehler
                add         #2,d7,d5            ;d5:Treiberadresse
                move        (d5),d5

                inc         d2
                move        (d2),d6             ;d6:Pufferadresse
                inc         d2
                move    sf (d2),d2              ;d2:Anzahl
                jeq         stringwrite         ;0: STRING
                cmp         -,d2
                jeq         wortwrite           ;1: WORT

;.....................................................................
;sonst BLOCK TRANSFERIEREN
blockwrite:     and     sf -,(d1),-     ;Block oder Zeichendevice?
                jeq         .blockdevice  ;Block!

.zeichendevice: push        (d6)          ;Zeichen an Driver ausgeben
                push        #2
                driver      d5            ;Teiberaufruf
                add         #2,sp
                jmi         transferende.e2
                inc         d6            ;Pufferzeiger++
                dec     sf d2             ;Anzahl--
                jne         .zeichendevice
                jmp         transferende

.blockdevice:   cmp         d2,-          ;letztes Zeichen kopiert
                jeq         transferende  ;Ja, fertig
                add         #7,d7,d3
                sub         (d3),#256,d4  ;Anzahl der übertragbaren Worte im aktuellen Sektor
                cmp         d2,d4         ;Reicht der Platz im aktuellen Sektor?
                sub  mi     d4,d2         ;Nein, neue Länge=Alte Länge-Anzahl übertragener
Worte
                move pl     d2,d4         ;Ja, nur soviel übertragen wie nötig
                clr  pl     d2            ;...und dann nichts mehr
                jsr         OUT
                jmi         transferende.e2
                add         d4,d6         ;Pufferzeiger um Anzahl übertragener Worte
weiterschalten
```

```
                jmp         .blockdevice

;.....................................................................
;WORT TRANSFERIEREN
wortwrite:      and     sf -,(d1),-     ;Blockdevice?
                jeq         .blockdevice  ;Ja

                push        d6            ;Zeichen an Zeichendevice ausgeben
                push        #2
                driver      d5
                add         #2,sp
                jmp         transferende.e2

.blockdevice:   move        -,d4          ;Anzahl der zu schreibenden Worte=1
                add         #8,sp,d6      ;Adresse des Zeichens im Stack
                jsr         OUT           ;Read-Routine
                jmp         transferende.e2

;.....................................................................
;STRINGWRITE
stringwrite:    and     sf -,(d1),-     ;Block oder Zeichendevice?
                jeq         .blockdevice  ;Block!

                push        d6            ;String an Treiber ausgeben
                push        #3
                driver      d5
                add         #2,sp
                jmp         transferende.e2

.blockdevice:   move        d6,d0         ;Temporärpointer, d2 ist schon 0
.length:        cmp         (d0),-        ;Stringlänge ermitteln
                jeq         blockwrite.blockdevice
                inc         d2            ;Anzahl++
                inc         d0            ;Pointer++
                jmp         .length

;---------------------------------------------------------------------------
;EXIT für beide Übertragungsroutinen
transferende:   clr     sf ~-           ;Flags löschen
.e2:            popall
                rts
.fehler:ror     sf -,-,-;N-Flag setzen
                jmp         .e2

;---------------------------------------------------------------------------
;READAUFRUF nur durch BS zugänglich
;EOF überprüfung
in:             push        d2
                add         #6,d7,d0      ;Sind wir im letzten Sektor?
                add         #12,d7,d1     ;d1: Dateilänge in Sektoren
                cmp         (d0),(d1)
                clr         d2            ;Erst mal kein Fehler
                jne         .noeof        ;Nein,  Fileende wird nicht überschritten
                inc         d0
                inc         d1
                add         (d0),d4,d3    ;Wortpos nach Leseoperation
                cmp         d3,(d1)
                sub mi      (d0),(d1),d4  ;EOF würde überschritten werden -> weniger lesen
                move mi     #105,d2       ;Fehler: EOF

;Absolute Sektornummer ermitteln
.noeof:         push        d7            ;Lesesektorposition berechnen
                jsr         LOG_ABS
                inc         sp
                jmi         .ende
;Daten einlesen
                add         #7,d7,d3
                push        d4            ;Anzahl der  Worte
                push        (d3)          ;Ab Position d0
                push        d1            ;Sektornummer lo
                push        d0            ;Sektornummer hi
                push        d6            ;Zeiger auf Puffer
                push        -             ;Job 1:lesen
                driver      d5            ;Treiberaufruf
                add         #6,sp
                jmi         .ende

;Dateizeiger erhöhen
                add         d4,(d3)       ;Dateizeiger lo erhöhen
                cmp         #256,(d3)
                and         #$ff,(d3)
                dec         d3
```

```
                inc eq     (d3)         ;Dateizeiger hi erhöhen

                move       d4,d1        ;Anzahl der gelesenen Zeichen
                move    sf d2,d0        ;Falls kein EOF ist d2=0
                ror ne  sf -,-,-        ;N-Flag bei EOF
.ende:          pop        d2
                rts

;-------------------------------------------------------------------------
;WRITEAUFRUF nur durch BS zugänglich
;Ist Datei länger geworden?
out:            push       d2
                add        #7,d7,d0     ;Stehen wir am Anfang eines neuen Sektors?
                cmp        (d0),-       ;Nein, wir stehen mitten in einem Sektor
                jne        .calc
                add        #6,d0        ;Steht das Dateiende an einer Sektorgrenze?
                cmp        (d0),-       ;Nein, wir brauchen keinen neuen Sektor
                jne        .calc
                dec        d0           ;Stehen wir am EOF?
                add        #6,d7,d1     ;Dateizeiger Worte
                cmp        (d1),(d0)
                jne        .calc;Nein

;Dateilänge hat sich verändert, wir brauchen einen neuen Sektor
                push       d7           ;Einen neuen Datensektor an die Datei anfügen
                jsr        SEKTOR_FORDERN
                inc        sp
                jmi        .ende

;Hier ist nun sichergestellt, daß an der Schreibposition ein gültiger Datensektor liegt
.calc:          push       d7           ;Schreibsektorposition berechnen
                jsr        LOG_ABS
                inc        sp
                jmi        .ende
;Daten ablegen
                push       d4           ;Anzahl der  Worte
                add        #7,d7,d3
                push       (d3)         ;Ab Position
                push       d1           ;Sektornummer lo
                push       d0           ;Sektornummer hi
                push       d6           ;Zeiger auf Puffer
                push       #2           ;Job 2:schreiben
                driver     d5
                add        #6,sp
                jmi        .ende

;Filepos weiterschalten
                add        #7,d7,d0     ;Zeiger auf Dateipos-Offset
                add        d4,(d0)      ;Dateizeiger lo erhöhen
                cmp        #256,(d0)    ;Sektorgrenze erreicht?
                and        #$ff,(d0)    ;Offset normieren
                dec        d0
                inc eq     (d0)

;Dateilänge anpassen
                add        #12,d7,d1
                cmp        (d1),(d0)    ;Sektoranzahl<Aktueller Sektor
                jmi        .notbigger   ;Ja, Datei hat sich nicht vergrößert
                inc        d0
                inc        d1
                jpe        .bigger      ;Umgekehrt -> Datei hat sich vergößert
                cmp        (d0),(d1)    ;Dateizeigeroffset > Dateilängenoffset?
                jpl        .notbigger   ;Nein

.bigger:        move       (d0),(d1)    ;Neue Dateilänge im FDB eintragen
                dec        d0
                dec        d1
                move       (d0),(d1)

;neue Dateilänge abspeichern
                push       #5           ;Neuen Directoryentry schreiben
                add        #5,d7,d0
                push       (d0)         ;Dir-Verweis-Offset
                dec        d0
                push       (d0)         ;Dir-Verweis-Sektornummer lo
                dec        d0
                push       (d0)         ;Dir-Verweis-Sektornummer hi
                add        #10,d7,d1    ;Zeiger auf neuen Entry
                push       d1
                push       #2           ;Job 2:schreiben
                driver     d5
                add        #6,sp
```

```
                jmi         .ende
.notbigger:     clr     sf ~-           ;Flags löschen
.ende:          pop         d2
                rts
```

12.5.12. CHANGE DIRECTORY

```
;---------------------------------------------------------------------------
;CD
;---------------------------------------------------------------------------
;Change Directory
cd:             push        d2
                push        d3
                push        d4
                add         #4,sp,d3    ;Zeiger auf Stringzeiger
                move        (d3),d3     ;d3: Stringzeiger

;Arbeitsspeicher allokieren
                push        #256
                jsr         MALLOC      ;Speicher anfordern
                inc         sp
                jmi         .ende
                move        d0,d2

;Welche Form des CD?
                cmp         (d3),#$2e   ;"..",cwd kürzen?
                jne         .new        ;Nein
                add         -,d3,d0
                cmp         (d0),#$2e
                jeq         .cut        ;ja

;Existiert dieses Verzeichnis?
.new:           push        -           ;Ist Dir zulässig?
                push        d3          ;Zum lesen öffnen
                jsr         OPEN        ;Überprüfen ob Dir existiert
                add         #2,sp
                jmi         .fehler     ;neue cwd exisitiert nicht

                add         #15,d0,d1   ;Handelt es sich bei der Datei um eine Dir?
                and     sf -,(d1),d4    ;Dir-Flag gesetzt

                push        d0          ;Datei wieder schließen
                jsr         CLOSE
                inc         sp

                move    sf d4,-         ;D-Flag gesetzt?
                move eq    #113,d0
                jeq         .fehler     ;Nein

;Im Arbeitspuffer das neue CWD aufbauen
.cut:           move        d2,d4
                move        #$63,(d2)   ;"cwd=" in Puffer eintragen
                inc         d2
                move        #$77,(d2)
                inc         d2
                move        #$64,(d2)
                inc         d2
                move        #$3d,(d2)
                inc         d2

                cmp         (d3),#$2f   ;"/", Neue cwd?
                jeq         .copy;ja

                push        #cwd        ;cwd bestimmen
                jsr         SEARCH_ENV
                inc         sp
                jmi         .fehler

.eintragen:     cmp         (d0),-      ;Alte cwd hinzufügen
                move ne    (d0),(d2)
                inc  ne     d0
                inc  ne     d2
                jne         .eintragen

                cmp         (d3),#$2e   ;".",cwd kürzen?
                jne         .copy       ;Nein, Srring kopieren
                add         -,d3,d0
                cmp         (d0),#$2e
                move eq    d2,d3
                dec  eq     d2
                clr         d0
                jeq         .check      ;ja
```

```
.copy:          cmp        (d3),-       ;String in cwd kopieren
                move ne    (d3),(d2)
                inc  ne    d3
                inc  ne    d2
                jne        .copy
.last:          move       #$2f,(d2)    ;"/",Trennzeichen anhängen
                inc        d2
                clr        (d2)

.new_set:       push       d4           ;geänderte cwd in Env.-Bereich eintragen
                jsr        SET_ENV
                inc        sp
                jmi        .fehler

.ende:          push       d4
                jsr        MFREE
                inc        sp
                clr     sf ~-
.ende2:         pop        d4
                pop        d3
                pop        d2
                rts
.fehler:        push       d0           ;Fehlercode retten
                push       d4
                jsr        MFREE
                inc        sp
                pop        d0           ;Fehlercode restaurieren
                ror     sf -,-,-        ;N-Flag setzen
                jmp        .ende2

;cwd soll gekürzt werden.
.check:         dec        d3           ;Überprüfung ob Kürzung zulässig
                cmp        (d3),#$2f    ;d3 steht am Ende der Cwd,"/"?
                inc eq     d0
                cmp        d0,#2
                jmi        .delete
                cmp        (d3),#$3d
                jeq        .ende
                jmp        .check

;Es exitiert noch min. ein Eintrag in der cwd, diesen vom Ende her bis zu
;einem / löschen.
.delete:        clr        (d2)         ;Rückwärts löschen
                dec        d2
                cmp        (d2),#$2f    ;"/"?
                jeq        .new_set
                jmp        .delete

cwd:            dw "cwd=",0             ;Suchstring im Env.-Bereich
```

12.5.13. LOAD PROGRAM

```
;---------------------------------------------------------------------------
;LOAD PROGRAM
;---------------------------------------------------------------------------
;Programm wird geladen und reloziert. d0 enthält die Startadresse des Programms.
prg_load:       add        -,sp,d0
                push       d2
                push       d3

;Datei zum Lesen öffnen
                push       -            ;zum lesen öffnen
                push       (d0)         ;Pointer auf Dateinamen
                jsr        OPEN
                add        #2,sp
                jmi        .exit
                move       d0,d2       ;d2:File-Handle

;Ist diese Datei ausführbar?
                add        #15,d0       ;Zeiger auf Dateiflags im FDB
                and     sf #2,(d0),-    ;Executable-Flag gesetzt?
                move eq    #117,d0      ;Nein
                ror  eq sf -,-,- ;N-Flags setzen
                jmi        .fehler

;Länge des PRGs im Speicher ermitteln
                push       -            ;1 Wort lesen (Speicherbedarf des Programes)
                dec        sp           ;Dummy, wegen 1 Wort lesen
                push       d2
                jsr        READ
                add        #3,sp
```

```
                jmi         .fehler

;Entsprechend großen Speicher anfordern
                push        d0              ;Speicher anfordern für Programm
                jsr         MALLOC
                inc         sp
                jmi         .fehler
                move        d0,d3           ;d3: Speicheradresse des Programms

;Länge des initialisierten Bereich betsimmen
                push        -               ;Nächstes Wort: Länge des zu ladenden Programmteils
                dec         sp              ;Dummy, wegen 1 Wort lesen
                push        d2
                jsr         READ
                add         #3,sp
                jmi         .memfree

;Diesen Bereich laden
                push        d0             ;Programmtext laden
                push        d3
                push        d2
                jsr         READ
                add         #3,sp
                jmi         .memfree

;Relozieren
.wortweise:     push        -               ;1 Wort 'Relo'-Informationen lesen
                dec         sp              ;Dummy
                push        d2
                jsr         READ
                add         #3,sp
                jmi         .ende           ;Fertig reloziert
                add         d3,d0           ;Wort relozieren
                add         d3,(d0)
                jmp         .wortweise

;Bei Fehler Speicher wieder freigeben und Datei schließen
.memfree:       push        d3
                move        d0,d3           ;Fehlercode retten
                jsr         MFREE
                inc         sp
                move        d3,d0           ;Fehlercode restaurieren
.fehler:        move        d3,d0           ;Fehlercode restaurieren
                ror     sf -,-,-            ;N-Flag setzen
                jmp         .exit

.ende:          push        d2
                jsr         CLOSE           ;Geöffnete Datei wieder schließen
                inc         sp
                clr     sf ~-               ;Flags löschen
                move        d3,d0           ;Startadresse vom Prg zurückgeben
.exit:          pop         d3
                pop         d2
                rts
```

12.5.14. MAKE DIRECTORY

```
;---------------------------------------------------------------------------
;MAKE DIRECTORY
;---------------------------------------------------------------------------
md:             add         -,sp,d0
                push        d2
                move        (d0),d2         ;Zeiger auf Dateinamen

;Verzeichnis öffnen
                push        #3              ;Zum Neuschreiben öffnen
                push        d2              ;Dateiname
                jsr         OPEN            ;Datei erzeugen
                add         #2,sp
                jmi         .ende
                move        d0,d2

;D-Flag setzen
                push        -
                push        d0
                jsr         SET_FLAGS       ;Set-D-Flag
                add         #2,sp

;Datei wieder schließen
                push        d2              ;Datei wieder schließen
                jsr         CLOSE
                inc         sp
```

```
.ende:          pop         d2
                rts
```

12.5.15. REMOVE DIRECTORY

```
;------------------------------------------------------------------------------
;REMOVE DIRECTORY
;------------------------------------------------------------------------------
rd:             add         -,sp,d0
                push        d2
                move        (d0),d2       ;d2: Zeiger auf Dateinamen

;Datei zum Lesen öffnen
                push        #2            ;zum lesen und schreiben öffnen
                push        d2            ;Dateiname
                jsr         OPEN          ;Directory öffnen
                add         #2,sp
                jmi         .ende
                move        d0,d2

;Ist dies auch eine Blockdevice?
                add         -,d0
                and     sf -,(d0),-       ;Block oder Zeichen Device?
                move ne     #102,d0
                jne         .fehler

;Ist es eine Directory?
                add         #14,d0        ;Zeiger auf Dateiflags
                and     sf -,(d0),-       ;Ist D-Flag gesetzt?
                move eq     #113,d0
                jeq         .fehler

;Ist dieses Verzeichnis auch leer?
                push        -             ;absolut Positionieren
                push        d2
                pushz
                push        #8
                jsr         SET_POS
                add         #4,sp
                jmi         .del

.read:          push        -             ;Anzahl
                dec         sp
                push        d2            ;FDB-Adresse
                jsr         READ          ;READ
                add         #3,sp
                jmi         .fehlercheck
                cmp         d0,-
                move ne     #116,d0
                jne         .fehler

                pushz
                push        d2
                pushz
                push        #15
                jsr         SET_POS
                add         #4,sp
                jpl         .read

.fehlercheck:   cmp         d0,#105       ;EOF-Fehler?
                jne         .fehler       ;Nein

;Verzeichnis löschen
.del:           push        d2
                jsr         DELETE
                inc         sp
                jmi         .fehler

;Verzeichnis schließen, um FDB freizugeben
                push        d2
                jsr         CLOSE
                inc         sp
                clr     sf ~-
.ende:          pop         d2
                rts

.fehler:        push        d0            ;Fehlercode retten
                push        d2
                jsr         CLOSE
                inc         sp
                pop         d0            ;Fehlercode restaurieren
```

```
                ror     sf -,-,-        ;N-Flag setzen
                jmp        .ende
```

12.5.16. PREPARE SEARCH

```
;---------------------------------------------------------------------------
;PREPARE_SEARCH
;---------------------------------------------------------------------------
;Datei muß geöffnet sein
;Es wird auf Anfang der Datei positioniert
prepare_search: add        -,sp,d0
                push       d2
                push       d3
                move       (d0),d3      ;d3:Zeiger auf Quelltring, ungepackt
                inc        d0
                move       (d0),d2      ;d2:Ergebnispuffer, 8 Worte groß
                inc        d0
                move       (d0),d0      ;d0: FDB-Adresse

;Ist FDB-Adresse gültig?
                jsr        FDB_ADRESS
                jmi        .ende

;Dateizeiger auf Dateianfang positionieren.
                add        #6,d0        ;Dateizeiger auf 0 positionieren
                clr        (d0)
                inc        d0
                clr        (d0)

                move       d2,d0;Arbeitspuffer löschen
                move       #8,d1
.clr:           clr        (d0)
                inc        d0
                dec     sf d1
                jne        .clr

;Namen uaf zulässige Zeichen untersuchen und kompremieren
                move       #16,d0       ;Maximal 16 Zeichen erlaubt
                clr        d1           ;Hi-Lo-Flag auf High
.copyloop:      cmp        (d3),-       ;Quellstring zu Ende?
                jeq        .fertig      ;Ja, Aktion erfolgreich
                cmp        #$2f,(d3)    ;Ein / ist auch eine Endekennung
                jeq        .fertig
                cmp        #$2a,(d3)    ;* Rest des Namens ohne Bedeutung
                jeq        .fill
                cmp        #$3c,(d3)    ;<
                jeq        .fehler
                cmp        #$3b,(d3)    ;; (Semikolon)
                jeq        .fehler
                cmp        #$3e,(d3)    ;>
                jeq        .fehler
                cmp        #$21,(d3)    ;Begin zulässiger Bereich
                jmi        .fehler
                cmp        #$80,(d3)    ;Ende zulässiger Bereich
                jpl        .fehler

                cmp        d1,-         ;Zeichen gültig -> kopieren
                bswp eq    -,(d3),(d2)  ;Zeichen ins Hibyte kopieren
                or   ne    (d3),(d2)    ;Bzw. ins Lowbyte...
                inc  ne    d2           ;...und Zielpointer erhöhen
                inc        d3           ;Quellpointer weiterschalten
                eor        -,d1         ;Hi-Lo-Flag togglen
                dec     sf d0           ;Maximal 16 Zeichen kopieren
                jne        .copyloop
                cmp        (d3),-
                jeq        .fertig
                jmp        .fehler      ;Suchstring hat mehr als 16 Zeichen


;* wird in ????... expandiert
.fill:          cmp        d1,-         ;Zeichen gültig -> kopieren
                bswp eq    -,#$3f,(d2)  ;Zeichen ins Hibyte kopieren
                or   ne    #$3f,(d2)    ;Bzw. ins Lowbyte...
                inc  ne    d2           ;...und Zielpointer erhöhen
                inc        d3           ;Quellpointer weiterschalten
                eor        -,d1         ;Hi-Lo-Flag togglen
                dec     sf d0
                jne        .fill

.fertig:        cmp        d0,#16       ;Wenn Dateinamenlänge=0: Fehler
                jne        .ende
.fehler:       move        #104,d0      ;Dateiname hat falsches Format
```

```
                ror     sf -,-,-        ;N-Flag setzen
.ende:          pop        d3
                pop        d2
                rts
```

12.5.17. SCAN

```
;------------------------------------------------------------------------
;SCAN
;------------------------------------------------------------------------
;Packt einen Teilpfad (Zeichen zwischen 2 /'s) zusammen in einen Zielbereich
;Arbeitet wird PREPAR_SEARCH, Wildcards sind nicht zulässig, Dateizeiger
;wird nicht positioniert.
scan:           add        -,sp,d0
                push       d2
                push       d3
                move       (d0),d3      ;d3:Zeiger auf Quelltring, ungepackt
                inc        d0
                move       (d0),d2      ;d2:Ergebnispuffer, 8 Worte groß

                move       d2,d0        ;Arbeitspuffer löschen
                move       #8,d1
.clr:           clr        (d0)
                inc        d0
                dec     sf d1
                jne        .clr

;Namen uaf zulässige Zeichen untersuchen und kompremieren, keine Wildcards
                move       #16,d0       ;Maximal 16 Zeichen erlaubt
                clr        d1           ;Hi-Lo-Flag auf High
.copyloop:      cmp        (d3),-       ;Quellstring zu Ende?
                jeq        .fertig      ;Ja, Aktion erfolgreich
                cmp        #$2f,(d3)    ;Ein / ist auch eine Endekennung
                jeq        .fertig
                cmp        #$3f,(d3)    ;? nicht erlaubt
                jeq        .fehler
                cmp        #$2a,(d3)    ;ebenso *
                jeq        .fehler
                cmp        #$3c,(d3)    ;<
                jeq        .fehler
                cmp        #$3b,(d3)    ;; (Semikolon)
                jeq        .fehler
                cmp        #$3e,(d3)    ;>
                jeq        .fehler
                cmp        #$21,(d3)    ;Begin zulässiger Bereich
                jmi        .fehler
                cmp        #$80,(d3)    ;Ende zulässiger Bereich
                jpl        .fehler

                cmp        d1,-         ;Zeichen gültig -> kopieren
                bswp eq    -,(d3),(d2)  ;Zeichen ins Hibyte kopieren
                or   ne    (d3),(d2)    ;Bzw. ins Lowbyte...
                inc  ne    d2           ;...und Zielpointer erhöhen
                inc        d3           ;Quellpointer weiterschalten
                eor        -,d1         ;Hi-Lo-Flag togglen
                dec     sf d0           ;Maximal 16 Zeichen kopieren
                jne        .copyloop
                cmp        (d3),-
                jeq        .fertig
                cmp        (d3),#$2f
                jeq        .fertig

.fehler:        move       #104,d0      ;Dateiname hat falsches Format
                ror     sf -,-,-        ;N-Flag setzen
                pop        d3
                pop        d2
                rts

.fertig:        cmp        d0,#16       ;Wenn Dateinamenlänge=0: Fehler
                jeq        .fehler
                move       d3,d0;Neuen Quellzeiger zurückgeben
                pop        d3
                pop        d2
                rts
```

12.5.18. SEARCH NEXT

```
;------------------------------------------------------------------------
;SEARCH_NEXT
;------------------------------------------------------------------------
;Nächsten passenden Eintrag suchen
search_next:    add        -,sp,d0
```

```
                pushall
                move       (d0),d7      ;d7: FDB-Adresse der Dir
                inc        d0
                move       (d0),d6      ;d6: Zeiger auf Suchstring

;Ist FDB-Adresse gültig?
                move       d7,d0
                jsr        FDB_ADRESS
                jmi        .ende

;Arbeitsspeicher zum Einlesen eines Directoryeintrages
                push       #16
                jsr        MALLOC       ;Arbeitsspeicher anfordern
                inc        sp
                jmi        .ende
                move       d0,d5        ;d5: Zeiger auf Arbeitsbereich

;Einen Directoryeintrag einlesen
.read:          push       #16          ;Anzahl
                push       d5           ;Pufferzeiger
                push       d7           ;FDB-Adresse
                jsr        READ         ;liest entsprechend des Dateizeigers 16 Worte
                add        #3,sp
                jmi        .memfree     ;Fehler aufgetreten!

                add        #8,d5,d2     ;d2:Zeiger auf eingelesenen Dateinamen
                cmp        (d2),-       ;Eintrag gültig
                jeq        .read;Nein
                move       d6,d4        ;Adresse des Dateinamens nach d4
                move       #8,d3        ;8*2 Zeichen vergleichen

;Entspricht der Name dem Suchkriterium?
.vergleich:     bswp       (d4),-,d0    ;höherwertiges Zeichen vergleichen
                cmp        d0,#$3f
                jeq        .lo
                bswp       (d2),-,d1
                cmp        d0,d1        ;Zeichen vergleichen
                jne        .read        ;ungleich, nächsten eintrag überprüfen
.lo:            and        #$ff,(d4),d0 ;niederwertiges Zeichen vergleichen
                cmp        d0,#$3f
                jeq        .ok
                and        #$ff,(d2),d1
                cmp        d0,d1        ;Zeichen vergleichen
                jne        .read        ;ungleich, nächsten eintrag überprüfen
.ok:            inc        d2           ;gleich,...
                inc        d4           ;...nächstes Zeichen vorbereiten
                dec     sf d3           ;Zeichenzähler dec
                jne        .vergleich

;Eintrag ist passend, -> Positionieren auf Anfang des Directoryeintrages
                add        #7,d7        ;Filepos auf Entryanfang zurücksetzen
                sub     sf #16,(d7)
                move mi    #240,(d7)    ;Dateizeiger low korrigieren, Bereich [0..255]
                dec        d7
                dec mi     (d7)
                clr     sf ~-           ;Flags löschen

.memfree:       pushf
                move       d0,d2        ;Fehlercode retten
                push       d5
                jsr        mfree
                inc        sp
                move       d2,d0        ;Fehlercode restaurieren
                popf

.ende:          popall
                rts
```

12.5.19. Hilfsroutinen

```
;---------------------------------------------------------------------------
;BLOCK BELEGUNG UND FREIGABE
;---------------------------------------------------------------------------
;SEKTOR ANFORDERN
;Fordert einen neuen Sektor zum Schreiben an.
;Die Sektoren werden nicht initialisisert, sondern
;einfach zugewiesen und in den Verwaltungsektoren eingetragen.
;Verwaltungssektoren werden initialisiert.
;Maximale Dateilänge: 65536 Sektoren * 512 Bytes = 32 MByte

sektor_fordern: add        -,sp,d0      ;FDB-Adresse holen
                pushall
```

```
                move       (d0),d7       ;d7: FDB-Adresse
                add        #2,d7,d5      ;Treiberadresse holen
                move       (d5),d5       ;d5: Treiberadresse
                add        #18,d7,d4     ;d4: Pufferadresse

                add        #6,d7,d0      ;Wird der allererste Sektor der Datei angefortert?
                move       (d0),d0
                cmp        d0,-
                jeq        .erster       ;Ja, schnelle Spezialbehandlung

                add     sf -,d0,-        ;Den $ffff-ten Sektor nicht mehr holen
                move eq    #108,d0       ;Sonst: Maximale Dateilänge erreicht
                jeq        .errorend

                add        #14,d7,d1     ;d1: Aktuelle Schachtelungstiefe
                add        #.exp128,(d1),d1
                move       (d1),d1
                cmp        d0,d1         ;Erhöht sich die Schachtelungstiefe?
                jne        .samelevel            ;Nein

                jsr        BLOCK_ALLOCATE
                jmi        .end

;Neuen Sektor initialisieren
                move       d0,d2
                move       d1,d3
                add        #10,d7,d0
                push       #2            ;2 Worte
                pushz                    ;Ab Wortpos
                push       d3            ;Sektor low
                push       d2            ;Sektor high
                push       d0            ;Pufferadresse
                push       #2            ;Job 2: Schreiben
                driver     d5
                add        #6,sp
                jmi        .end

;Neuen Sektor als Primärsektor eintragen
                add        #10,d7,d0
                move       d2,(d0)       ;Neuer Sektor wird Primärsektor
                inc        d0
                move       d3,(d0)
                add        #3,d0
                inc        (d0)          ;Schachtelungstiefe++

.samelevel:     add        #10,d7,d1     ;Nummer des Primärsektors holen
                move       (d1),d0
                inc        d1
                move       (d1),d1
                add        #6,d7,d3      ;Relative Sektornummer der neuen Filepos holen
                move       (d3),d3
                add        #14,d7,d2     ;d2: Schachtelungstiefe
                cmp        #2,(d2)       ;Wie oft muß dereferenziert werden?
                jmi        .tiefe1
                jeq        .tiefe2

.tiefe3:        rol        d3,-,d2
                rol        d2,-,d2
                and        #3,d2         ;d4=(d3 div 16384) and 3;
                jsr        .deref

.tiefe2:        asl        d3,-,d2
                bswp       d2,-,d2
                and        #127,d2       ;d2=(d3 div 128) and 127;
                jsr        .deref

.tiefe1:        and        #127,d3,d2    ;d2=d3 and 127;
                jsr        .deref
                clr     sf ~-
                jmp        .end

.errorend:      ror     sf -,-,-         ;N-Flag setzen
.end:           popall
                rts

.erster:        jsr        BLOCK_ALLOCATE        ;Neuen Sektor holen
                jmi        .end
                add        #10,d7,d2
                move       d0,(d2)       ;Neuer Sektor wird Primärsektor
                inc        d2
                move       d1,(d2)
                jmp        .end
```

```
;Einmal dereferenzieren
.deref:         asl         d2,-,d2       ;d2=d2*2;
                push        #2            ;2 Worte lesen
                push        d2            ;Ab Position d2
                push        d1            ;Sektornummer lo
                push        d0            ;Sektornummer hi
                push        d4             ;Pufferadresse
                push        -              ;Job 1:lesen
                driver      d5            ;Treiberaufruf
                add         #2,sp
                pop         d0             ;Alte Sektornummer wieder holen
                pop         d1
                add         #2,sp
                inc mi      sp
                jmi         .end

                cmp         (d4),-        ;Neuer Verweis=0?
                jne         .nextref      ;Nein, dereferenzieren
                inc         d4
                cmp         (d4),-
                dec         d4
                jne         .nextref      ;Nein, dereferenzieren

;Falls beim Dereferenzieren ein Verweis 0 auftritt muß eine weiterer
;Sektor allokiert werden
                push        d0             ;Wir haben einen 0-Verweis gefunden
                push        d1             ;--> Neuen Block holen
                jsr         BLOCK_ALLOCATE
                move        d0,(d4)
                inc         d4
                move        d1,(d4)
                dec         d4
                add mi      #3,sp
                jmi         .end
                pop         d1
                pop         d0

;Neuen Sektor eintragen
                push        #2            ;2 Worte schreiben
                push        d2            ;Ab Position d2
                push        d1            ;Sektornummer lo
                push        d0            ;Sektornummer hi
                push        d4             ;Pufferadresse
                push        #2             ;Job 2:Schreiben
                driver      d5            ;Treiberaufruf
                add         #6,sp
                inc mi      sp
                jmi         .end

;Verweis war nicht 0: holen
.nextref:       move ne     (d4),d0       ;d0: Sektornummer, high
                add         -,d4,d1
                move        (d1),d1       ;d1: Sektornummer, low
                rts

;Anzahl der logischen Sektoren pro Schachtelungstiefe
.exp128:dw          1,128,16384,$ffff

;---------------------------------------------------------------------------
;BLOCK ALLOKIEREN
;---------------------------------------------------------------------------
;Sektoradresse wird über dei Register d0,d1 übergeben. Alookierte Sektoren
;werden mit Null initialisiert.
;Diese Routine wird nur von der Routine SEKTRO_FORDERN aufgerufen, deswegen
;arbeiten diese beiden Routinen auch zusammen. Folgende Register enthalten
;dann Arbeitswerte:
;d4: Pufferbereich im FDB
;d5: Treiberadresse

;d0: Sektornummer high
;d1: Sektornummer low
block_allocate:  irqoff                                   ;Semaphor überprüfen
                cmp         allocate_in_use,-
                move eq     akt_task,allocate_in_use
                irqon
                jeq         .semok
;(Hier könnte ein Fairness-Flag gesetzt werden! Konkurrierendes Block-Allokieren
;ist jedoch sehr unwahrscheinlich, es sei denn mehere Tasks kopieren Dateien
;oder schreiben blockweise Daten.)
                jsr         SWITCHTASK
                jmp         block_allocate
```

```
.semok:         push       d2
                push       d3
                push       d6

;länge der BAM bestimmen
.retry:         push       -            ;Länge der BAM nach d6 holen
                push       #17          ;(=Sektornummer des letzen BAM-Sektors)
                pushz
                pushz
                push       d4           ;Ziel ist erstmal FDB-Pufferwort
                push       -
                driver     d5
                add        #6,sp
                jmi        .ende
                move       (d4),d6

;Position des letzten freien Sektors in BAM bestimmen
                push       #2           ;Pointer auf nächsten freien Eintrag
                push       #18          ;in BAM holen
                pushz
                pushz
                push       d4
                push       -
                driver     d5
                add        #6,sp
                jmi        .ende
                move       (d4),d2      ;d2/d3 zeigt darauf
                add        -,d4,d3
                move       (d3),d3

.loop:          push       -            ;BAM-Entry lesen
                push       d3
                push       d2
                pushz
                push       d4
                push       -
                driver     d5
                add        #6,sp
                jmi        .ende

                add     sf -,(d4),-     ;Gibt es einen freien Sektor?
                jne        .found       ;Ja

                inc        d3           ;Kein Glück: Nächstes Wort im BOM-Sektor
                bswp    sf d3,-,-       ;Sektorgrenze überschritten?
                jeq        .loop        ;Nein, im aktuellen Sektor weitersuchen
                inc        d2           ;Nächster Sektor
                clr        d3           ;Dort mit dem 1. Wort anfangen
                cmp        d2,d6        ;War das der letze BOM-Sektor?
                jpl        .loop        ;Nein, weitersuchen
                move       #112,d0      ;Ja, Datenträger voll
                jmp        .ende

.found:         clr        d6           ;Bitnummer der ersten 0 ermitteln
                move       -,d1         ;d1: Bitmaske
.bitloop:       and     sf (d4),d1,-    ;Bit clear?
                asl     ne d1           ;Nein, nächstes Bit untersuchen
                inc     ne d6
                jne        .bitloop

;Sektor in BAM belegen
                or         d1,(d4)      ;Sektor belegen
                push       -            ;Neuen BAM-Entry schreiben
                push       d3
                push       d2
                pushz
                push       d4
                push       #2
                driver     d5
                add        #6,sp
                jmi        .ende

                move       d2,(d4)      ;BAM-Entry-Adresse im Bootsektor merken
                add        -,d4,d0
                move       d3,(d0)
                push       #2
                push       #18
                pushz
                pushz
                push       d4
                push       #2
```

```
                driver     d5
                add        #6,sp
                jmi        .ende

                asl        d3            ;absolute Sektornummer bestimmen
                asl        d3
                asl        d3
                asl        d3
                add        d6,d3         ;d3=Bit+16*Wort
                dec        d2
                ror        d2,-,d0
                ror        d0
                ror        d0
                ror        d0
                and        #$f000,d0
                add        d0,d3         ;d3=Bit+16*Wort+(4096*Sektor)&$f000
                lsr        d2
                lsr        d2
                lsr        d2
                lsr        d2            ;d2=Sektor/16

                move       #248,d6
.clearloop:     push       #8            ;Neuen Sektor mit 0 füllen
                push       d6
                push       d3
                push       d2
                push       #.nullen
                push       #2
                driver     d5
                add        #6,sp
                jmi        .retry        ;Sektorfehler -> neuen Sektor
                sub     sf #8,d6
                jpl        .clearloop

                clr     sf ~-            ;Flags löschen
                move       d3,d1         ;Rückgabewerte
                move       d2,d0
.ende:          clr        allocate_in_use
;(Fairness-Flag überprüfung hier!)
                pop        d6
                pop        d3
                pop        d2
                rts

.nullen:dw 0,0,0,0,0,0,0,0

;-------------------------------------------------------------------------------
;BLOCK FREE
;-------------------------------------------------------------------------------
;Gibt eine Sektor wieder frei. Die Sektornummer muß über den Stack übergeben werden.
block_free:     add        -,sp,d0
                pushall                  ;Register retten
                move       (d0),d7       ;d7: FDB-Adresse
                inc        d0
                move       (d0),d1       ;d1: Sektornummer low
                inc        d0
                move       (d0),d2       ;d2: Sektornummer high

                asl        d2            ;Sektornummer h *16
                asl        d2
                asl        d2
                asl        d2
                bswp       d1,-,d0       ;d0=Sektornummer low/4096
                lsr        d0
                lsr        d0
                lsr        d0
                lsr        d0
                add        d0,d2         ;d2:Sektornummer low des BAM-Eintrages
                inc        d2
                and        #$0fff,d1
                and        #15,d1,d0     ;Bit Nummer des BAM-Eintrages
                lsr        d1
                lsr        d1
                lsr        d1
                lsr        d1            ;Wortnummer des BAM-Eintrages

                move       -,d3          ;d1: Bitmaske
                inc        d0
.bitloop:       dec     sf d0
                asl ne     d3            ;Nein, nächstes Bit untersuchen
                jne        .bitloop
                move       d1,d5
```

```
                add         #18,d7,d4     ;Pufferadresse in FDB
                push        -             ;Anzahl der Worte
                push        d1            ;Ab welchen Wort
                push        d2            ;Sektornummer lo
                pushz                     ;Sektornummer hi
                push        d4            ;Pufferadresse
                push        -             ;Job 1:lesen
                add         #2,d7,d0
                driver      (d0)
                add         #6,sp
                jmi         .ende

                sub         d3,#$ffff,d3
                and         d3,(d4)
                push        (d4)

                push        -             ;Anzahl der Worte
                push        d5            ;Ab welchen Wort
                push        d2            ;Sektornummer lo
                pushz                     ;Sektornummer hi
                add         #4,sp,d0
                push        d0            ;Pufferadresse;Wort d2 wieder abspeichern
                push        #2            ;Job 2:schreiben
                add         #2,d7,d0
                driver      (d0)
                add         #7,sp
                jmi         .ende

                clr     sf ~-             ;Flags löschen
.ende:          popall
                rts



;-------------------------------------------------------------------------------
;LOG_ABS
;-------------------------------------------------------------------------------
;Nur interner Aufruf.
;Umwandlung der logischen Sektornummer in die absolute(HI & LO)
;do, d1 enthalten die gesuchten Nummern
log_abs:        add         -,sp,d1
                push        d2
                push        d3
                push        d4
                push        d5
                move        (d1),d4       ;d4: FDB-Adresse

;Primärsektor bestimmen
                add         #10,d4,d1
                move        (d1),d0       ;Primärsektor high
                inc         d1
                move    sf (d1),d1        ;Primärsektor low
                jne         .noerror
                cmp         d0,-
                jne         .noerror

.errorend:      sub         -,-,d0        ;Es gibt gar keine Datensektoren
                sub     sf -,-,d1         ;Vorsichtshalber -1 zurückgeben mit N-Flag
                jmp         .ende

.noerror:       add         #14,d4,d2     ;d2: Schachtelungstiefe holen
                move    sf (d2),d2
                jeq         .ende         ;Falls sie 0 ist, dann ist der Primärsektor der
gesuchte

                add         #6,d4,d3      ;Relative Sektornummer der akt Filepos holen
                move        (d3),d3

                add         #2,d4,d5
                move        (d5),d5       ;Driveradresse
                add         #18,d4        ;Pufferadresse

                cmp         #2,d2         ;Wie oft muß dereferenziert werden?
                jmi         .tiefe1
                jeq         .tiefe2

;entsprechend der Schachtelungstiefe dereffenrenzieren
.tiefe3:        rol         d3,-,d2
                rol         d2,-,d2
                and         #3,d2         ;d4=(d3 div 16384) and 3;
                jsr         .deref
```

```
.tiefe2:        asl        d3,-,d2
                bswp       d2,-,d2
                and        #127,d2       ;d2=(d3 div 128) and 127;
                jsr        .deref

.tiefe1:        and        #127,d3,d2    ;d2=d3 and 127;
                jsr        .deref

                clr     sf ~-
.ende:          pop        d5
                pop        d4
                pop        d3
                pop        d2
                rts

.deref:         asl        d2,-,d2       ;d2=d2*2;
                push       #2            ;2 Worte lesen
                push       d2            ;Ab Position d2
                push       d1            ;Sektornummer lo
                push       d0            ;Sektornummer hi
                push       d4             ;Pufferadresse
                push       -              ;Job 1:lesen
                driver     d5            ;Treiberaufruf
                add        #6,sp
                inc mi     sp
                jmi        .errorend
                move       (d4),d0       ;1 mal dereferenzieren
                add        -,d4,d1
                move       (d1),d1
                rts

;-------------------------------------------------------------------------------
;FDB_CHECK_X
;-------------------------------------------------------------------------------
;Überprüft, ob eine Datei schon zum lesen/schreiben geöffnet ist
fdb_check_x:    add        -,sp,d0
                push       d2
                push       d3
                push       d4
                add        #5,(d0),d1    ;Adresse des zu überprüfenden FDB holen
                clr        d0            ;Flags erst mal löschen
                move       #fdblist,d2   ;Adresse des 1. FDB in FDB-List holen

.loop:          move    sf (d2),d2       ;Nächsten FDB in Liste
                jeq        .exit         ;Fertig?
                move       d1,d3
                add        #5,d2,d4
                cmp        (d3),(d4)     ;Verweis-Wortnummer gleich?
                jne        .loop
                dec        d3
                dec        d4
                cmp        (d3),(d4)     ;Verweis-Sektornummer-low gleich?
                jne        .loop
                dec        d3
                dec        d4
                cmp        (d3),(d4)     ;Verweis-Sektornummer-high gleich?
                jne        .loop
                dec        d3
                dec        d4
                cmp        (d3),(d4)     ;Driver-Adresse gleich?
                jne        .loop
                dec        d4            ;Doppelten FDB gefunden
                move       (d4),d0

.exit:          pop        d4
                pop        d3
                pop        d2
                rts

;-------------------------------------------------------------------------------
;FDB_ADRESS
;-------------------------------------------------------------------------------
;Überprüfung, ob Adresse ein Zeiger auf einen gültigen FDB ist
fdb_adress:     add        #8,d0         ;Magic1 korrekt?
                cmp        #$1234,(d0)
                jne        .fehler               ;Nein
                inc        d0                    ;Magic2 korrekt?
                cmp        #$5678,(d0)
                jne        .fehler               ;Nein
                sub        #9,d0                 ;d0 restore
                clr     sf ~-
```

```
                rts
.fehler:        move       #101,d0              ;Fehler: Ungültiges Handle
                ror     sf -,-,-
                rts

;---------------------------------------------------------------------------
;OPENED_FILES
;---------------------------------------------------------------------------
;Überprfüfung ob für angegebenes SCSI-Device noch zum Schreiben geöffnete Dateien existieren
;Dies ist nur ein interner Aufruf.
opened_files:   add        -,sp,d0
                push       d2
                clr        d2           ;Anzahl mit Null initialisieren

;Aus Treibernamen Treiberadresse bestimmen
                push       (d0)
                jsr        SEARCHDRIVER
                inc        sp
                jmi        .ende

;FDB-Liste durchsuchen und überprüfen ob es noch zum Schreiben geöffnete Dateien gibt
                move       #fdblist,d1
.fdblist:       cmp        (d1),-
                jeq        .ende
                move       (d1),d1
                add        #2,d1
                cmp        (d1),d0      ;Richtiges Device?
                sub        #2,d1
                jne        .fdblist
                inc        d1
                and     sf #4,(d1),-    ;Zum Schreiben geöffnet?
                inc ne     d2           ;Anzahl++
                dec        d1
                jmp        .fdblist

.ende:          move       d2,d0        ;d0: Anzahl
                pop        d2
                rts
```

12.5.20. Fehlerausgabe

```
;---------------------------------------------------------------------------
;FEHLERAUSGABE AN STANDARDOUT
;---------------------------------------------------------------------------
fehler:         add        -,sp,d0
                move       (d0),d0      ;Fehlercode nach d0 holen

                bswp    sf d0,-,d1      ;Treiberfehler
                jne           .treiberfehler
                sub           #100,d0       ;Fehlercode -100
                add           #.fehlertabelle,d0
.out:           pushz                       ;Fehler-String ausgeben
                push       (d0)
                pushz
                jsr        WRITE
                add        #3,sp
                rts

;Es ist ein SCSI-Fehler, Statuscode betrachten
.treiberfehler: and           #$ff,d1,d0
                sub           #2,d0
                add           #.treibererrors,d0
                jmp           .out

.fehlertabelle: dw      .noexe
                dw      .geschlossen
                dw      .zugriff
                dw      .speicher
                dw      .form
                dw      .eof
                dw      .device
                dw      .env
                dw      .laenge
                dw      .exist_not
                dw      .exist
                dw      .parameter
                dw      .disk_full
                dw      .nodir
                dw      .writeprotect
                dw      .konflikt
                dw      .dir
                dw      .executable
```

```
                dw      .keinzeichen

.noexe:         dw 10,13,"Es gibt keine ausführbare Datei dieses Namens!",13,10,0
.geschlossen:   dw 10,13,"Ungültiges Handle!",13,10,0
.zugriff:       dw 10,13,"Befehlszugriff nicht erlaubt!",13,10,0
.speicher:      dw 10,13,"Zuwenig Speicherplatz!",13,10,0
.form:          dw 10,13,"Dateiname hat falsches Format!",13,10,0
.eof:           dw 10,13,"End-Of-File erreicht!",13,10,0
.device:        dw 10,13,"Device existiert nicht!",13,10,0
.env:           dw 10,13,"Environmentbereich zu klein!",13,10,0
.laenge:        dw 10,13,"Maximale Dateilänge (32MB) erreicht!",13,10,0
.exist_not:     dw 10,13,"Datei existiert nicht!",13,10,0
.exist:         dw 10,13,"Datei existiert schon!",13,10,0
.parameter:     dw 10,13,"Parameter falsch!",13,10,0
.disk_full:     dw 10,13,"Datenträger voll!",13,10,0
.nodir:         dw 10,13,"Directory existiert nicht!",13,10,0
.writeprotect:  dw 10,13,"Datei ist schreibgeschützt!",13,10,0
.konflikt:      dw 10,13,"Zugriffskonflikt auf mehrfach benutzte Datei!",13,10,0
.dir:           dw 10,13,"Directory ist nicht geleert!",13,10,0
.executable:    dw 10,13,"Datei ist nicht ausführbar!",13,10,0
.keinzeichen:   dw 10,13,"Es liegt kein Zeichen an!",13,10,0

.treibererrors: dw .nr,.me,.he,.ua,.rs,.wp,.z,.hs,.z,.z,.z,.z,.se

.z:             dw 0
.nr:            dw 10,13,"SCSI-Laufwerk nicht bereit!",13,10,0
.me:            dw 10,13,"SCSI-Medienfehler!",13,10,0
.he:            dw 10,13,"SCSI-Hardware-Fehler!",13,10,0
.ua:            dw 10,13,"Ungültige SCSI-Anfrage!",13,10,0
.rs:            dw 10,13,"SCSI-Gerät wurde zurückgesetzt!",13,10,0
.wp:            dw 10,13,"Medium in SCSI-Laufwerk schreibgeschützt!",13,10,0
.hs:            dw 10,13,"Herstellerspezifischer SCSI-Fehler aufgetreten!",13,10,0
.se:            dw 10,13,"SCSI-Schreibfehler!",13,10,0
```

12.6. Umgebungsvariablenverwaltung

12.6.1. SET ENVIRONMENT

```
;---------------------------------------------------------------------------
;SET_ENV
;---------------------------------------------------------------------------
;Set environment
;Setzt eine Enivronementvariable.
set_env:add        -,sp,d0              ;Zeiger auf Strg-Pointer
                push       d2
                push       d3
                move       (d0),d2      ;d2:Stringpointer

;Falls Env.-Var. existiert -> löschen
                push       d2           ;alte gleichnamige Variable...
                jsr        CLR_ENV      ;...löschen
                inc        sp

;Länge der Variablen bestimmen
                move       d2,d0        ;Länge des neuen Strings berechnen
.length:        cmp        (d0),-       ;Ende des Strings?
                inc        d0
                jne        .length

;Größe des Env.-Bereiches bestimmen
                sub        d2,d0,d3             ;d3:Länge des Strings (mit der 0)
                inc        d3                   ;Wir brauchen aber 1 Wort mehr für
zusätzliche 0
                sub        #2,akt_task,d1               ;Länge des TDB
                add        (d1),d1              ;Pointer auf 1. Wort nach ENV-Bereich
                add        #22,akt_task,d0      ;Environmentbereichsadresse

;Freie Position im Env.-Bereich suchen und Var. eintragen, falls Länge reicht
.search:        cmp        (d0),-       ;Freier Platz?
                jeq        .found       ;Ja,gefunden
.loop:          cmp        (d0),-       ;Zur nächsten Variablen
                inc        d0
                jne        .loop
                jmp        .search

.found:         sub        d0,d1        ;Reicht der Platz noch aus?
                cmp        d3,d1
                move mi    #107,d0
                jmi        .ende        ;Nein, Fehler

.copy:          move    sf (d2),(d0)    ;Environmentvariable kopieren
```

```
                inc         d0
                inc         d2
                jne         .copy
                clr         (d0)          ;Und 1 Null mehr

                clr     sf ~-             ;Kein Fehler
.ende:          pop         d3
                pop         d2
                rts
```

12.6.2. CLEAR ENVIRONMENT

```
;---------------------------------------------------------------------------
;CLR_ENV
;---------------------------------------------------------------------------
;Clear Environment
;Löscht eine Environmentvariable
clr_env:        push        d2
                add         #22,akt_task,d0       ;Environmentbereichsadresse

;Var. suchen
.search:        cmp         (d0),-        ;Eintrag suchen
                jeq         .ende         ;Kein String gefunden
                move        d0,d2         ;Position merken für clear
                add         #2,sp,d1      ;Zeiger auf Strg-Pointer
                move        (d1),d1       ;d1:Stringpointer

.found:         cmp         (d0),(d1)     ;Gleiche Zeichen?
                jne         .go_on        ;Nein, zum nächsten Env-String
                cmp         #$3d,(d0)     ;"=", Abbruch
                jeq         .remove       ;Variable gefunden, löschen
                inc         d0
                inc         d1
                jmp         .found
.go_on:         cmp         (d0),-        ;Zum nächsten String weiterschalten
                inc         d0
                jne         .go_on
                jmp         .search

;Eintrag löschen, indem der Rest des Env. bereiches an diese Position aufrückt
.remove:        sub         #2,akt_task,d1 ;Länge des TDB
                add         (d1),d1         ;Adresse des 1. Worts nach ENV-Bereich

.nextvar:       cmp         (d0),-        ;Suche nach der nächsten Variablen
                inc         d0
                jne         .nextvar
.copy:          move        (d0),(d2)     ;Variable gefunden, verschieben!
                inc         d0
                inc         d2
                cmp         d0,d1         ;Env-Ende?
                jne         .copy         ;Nein, weiterverschieben

.clear:         clr         (d2)          ;Rest des Env-Bereichs löschen
                inc         d2
                cmp         d2,d1
                jne         .clear

.ende:          pop         d2
                rts
```

12.6.3. SEARCH ENVIRONMENT

```
;---------------------------------------------------------------------------
;SEARCH_ENV
;---------------------------------------------------------------------------
;Search Environment
;Sucht eine Environementvariable
search_env:     add         #22,akt_task,d0       ;Environmentbereichsadresse
.search:        cmp         (d0),-                ;Eintrag leer?
                jeq         .errorend             ;Ja, String nicht gefunden
                add         -,sp,d1               ;Zeiger auf Strg-Pointer
                move        (d1),d1               ;d1:Stringpointer

.found:         cmp         (d0),(d1)     ;Gleiche Zeichen?
                jne         .go_on        ;Nein, zum nächsten Env-String
                cmp         (d0),#$3d     ;"=", Abbruch
                inc         d0
                rts eq                    ;Variable gefunden, fertig
                inc         d1
                jmp         .found
.go_on:         cmp         (d0),-        ;Nächste Variable suchen
                inc         d0
```

```
                jne         .go_on
                jmp         .search

.errorend:      ror     sf -,-,-        ;Variable nicht gefunden-> N-Flag setzen
                rts
```

12.7. Shell

```
;---------------------------------------------------------------------------
;SHELL
;---------------------------------------------------------------------------
shell:          push        #258        ;2*128 Zeichen Arbeitspuffer +2 gerettete
Handlenummern
                jsr         MALLOC
                inc         sp
                move        d0,d7       ;d7:Zeiger auf Arbeitspuffer!
                jpl         .memok

                push        d0
                jsr         FEHLER      ;Kein Speicherplatz verfügbar!
                inc         sp
                move        #103,d0
                ror     sf -,-,-
                rts

;Speicher konnte allokiert werden
;.........................................................................
.memok:         pushz                   ;Einschaltmeldung ausgeben
                push        #initstring
                pushz
                jsr         WRITE
                add         #3,sp

.new:           move       d7,d5        ;Beide Arbeitspuffer & stdin/out-Umlenkhilfvariablen
löschen
                move        #258,d3
                move        -,d6        ;Puffergelerrt Flag setzen
.clrbuf:        clr         (d5)        ;Puffer leeren
                inc         d5
                dec     sf d3
                jne         .clrbuf

                move        #127,d3     ;Zählt die freien Zeichen im Arbeitspuffer 1
                move       d7,d5        ;d5: Zeiger im Eingabepuffer

.old_string:    push        -
                push        #13         ;CR ausgeben...
                pushz                   ;... an StandardOut
                jsr         WRITE
                add         #3,sp
                jmi         .fehler

                pushz
                push        #clr_z      ;Zeile löschen...
                pushz                   ;... an StandardOut
                jsr         WRITE
                add         #3,sp
                jmi         .fehler

                push        #cwd        ;CWD-String im ENV-Bereich suchen
                jsr         search_env
                inc         sp

                pushz                   ;String ausgeben...
                push        d0          ;...und zwar die Cwd...
                pushz                   ;... an StandardOut
                jsr         WRITE
                add         #3,sp
                jmi         .fehler

                push        -           ;Prompt ausgeben...
                push        #$3e
                pushz                   ;... an StandardOut
                jsr         WRITE
                add         #3,sp
                jmi         .fehler

;Eingaben einlesen
;.........................................................................
.getloop:       push        -           ;Ein Zeichen
                dec         sp          ;Dummywert, wegen 1 Zeichen lesen
                pushz                   ;von Stdin
```

```
                jsr         READ            ;einlesen
                add         #3,sp,=
                jpl         .keystroke      ;Ja, Zeichen angekommen
                cmp         d0,#105
                jeq         .finish
                jsr         SWITCHTASK      ;Nein, warten
                jmp         .getloop

.keystroke:     bswp        d0,-,d2         ;d2:Scancode
                and         #$ff,d0         ;d0:ASCII-Code
                cmp         d2,#23          ;f3:alten inhalt der Eingabezeile anzeigen
                jne         .nicht_f3

                cmp         d6,-            ;Wurde Puffer schon verändert?
                jne         .getloop        ;ja
                move        d7,d5
                move        #127,d3
.bufloop:       cmp         (d5),-          ;Ende des Strings?
                inc ne      d5
                dec ne      d3
                jne         .bufloop

                pushz                       ;String ausgeben...
                push        d7
                pushz                       ;... an StandardOut
                jsr         WRITE
                add         #3,sp
                jmi         .fehler
                move        -,d6
                jmp         .getloop

.nicht_f3:      cmp         d6,-            ;Ist Puffer schon gelerrt?
                jne         .f1

;Puffer leeren falls erste Taste nicht F3
                move        d7,d5           ;Beide Arbeitspuffer & stdin/out-Umlenkhilfvariablen
löschen
                move        #258,d3
                move        -,d6            ;Puffergelerrt Flag setzen
.clrbuf2:       clr         (d5)
                inc         d5
                dec     sf  d3
                jne         .clrbuf2
                move        #127,d3         ;Zählt die freien Zeichen im Arbeitspuffer 1
                move        d7,d5           ;d5: Zeiger im Eingabepuffer

.f1:            cmp         d2,#7           ;f1:Eingabezeile löschen
                jeq         .new
                cmp         d0,#13          ;Ende der Eingabe?
                jeq         .auswertung
                cmp         d0,#8           ;Backspace?
                jeq         .backspace
                cmp         d2,#97          ;Crs left?
                jeq         .left
                cmp         d2,#106
                jeq         .right          ;Crs right?
                cmp         d2,#100
                jeq         .delete         ;Delete?
                cmp         #$20,d0
                jmi         .getloop        ;Zeichencode ohne Interesse!
                cmp         d0,#$7f
                jmi         .getloop        ;Zeichencode ohne Interesse!

                cmp         d3,-            ;Eingabepuffer voll?
                jeq         .getloop        ;Ja

                add         #126,d7,d1      ;1 Zeichen einfügen
                add         -,d1,d2
.insert:        move        (d1),(d2)
                dec         d2
                cmp         d1,d5
                dec         d1
                jne         .insert

                dec         d3              ;Eingabepuffer um 1 verkleinern
                move        d0,(d5)         ;Zeichen in Eingabepuffer schreiben
                inc         d5
                push        -               ;Das Zeichen an StdOut
                push        d0
                pushz
                jsr         write
                add         #3,sp
```

```
.cmdlineout:    pushz
                push        #crs_mark     ;Cursorposition  merken
                pushz
                jsr         WRITE

                pushz                     ;String ab Cursorpos an Stdout ausgeben
                push        d5
                pushz
                jsr         WRITE

                pushz                     ;Zeile bis Ende löschen
                push        #clr_z
                pushz
                jsr         WRITE

                pushz
                push        #crs_set      ;Cursor auf alte Position zurücksetzen
                pushz
                jsr         WRITE
                add         #12,sp
                jmi         .fehler
                jmp         .getloop      ;Nächstes Zeichen

;Spezialtasten
;.....................................................................
.backspace:     cmp         d5,d7         ;Cursor schon ganz links?
                jeq         .getloop      ;Ja, kein Backspace möglich

                pushz
                push        #crs_l        ;Cursor 1 mal links
                pushz
                jsr         write
                add         #3,sp
                dec         d5            ;Eingabepos eins nach links

.delete:        cmp         (d5),-        ;Cursor ganz rechts?
                jeq         .getloop      ;Ja, nichts deleten

                inc         d3            ;Eingabezeile wird 1 kleiner
                add         -,d5,d1
                move        d5,d0
.back:          move    sf (d1),(d0)      ;Rest des Eingabestrings verschieben
                inc         d0
                inc         d1
                jne         .back
                jmp         .cmdlineout   ;Kommandostring neu ausgeben

.left:          cmp         d5,d7         ;Cursor schon ganz links?
                jeq         .getloop      ;Ja

                pushz
                push        #crs_l
                pushz
                jsr         WRITE         ;Cursor verschieben
                add         #3,sp
                dec         d5            ;Eingabepos eins nach links
                jmp         .getloop

.right:         cmp         (d5),-        ;Schon ganz rechts?
                jeq         .getloop      ;Ja

                pushz
                push        #crs_r
                pushz
                jsr         WRITE         ;Cursor verschieben
                add         #3,sp,=
                inc         d5            ;Eingabepos eins nach rechts
                jmp         .getloop

.entry:         clr         d6
                jmp         .old_string


;User hat Return gedrückt
;.....................................................................
.auswertung:    pushz
                push        #newline      ;CR/LF ausgeben...
                pushz                     ;... an StandardOut
                jsr         WRITE
                add         #3,sp,=
```

```
;Parameter in Kommandozeile in Variable CMD= übertragen
                add         #128,d7,d4    ;Arbeitsbereich für execute
                move        d4,d0
                move        #$63,(d0)     ;c
                inc         d0
                move        #$6d,(d0)     ;m
                inc         d0
                move        #$64,(d0)     ;d
                inc         d0
                move        #$3d,(d0)     ;=
                inc         d0

                cmp         (d7),-        ;Eingabezeile leer?
                jeq         .new
                move        d7,d2         ;d2: Anfang des Strings
                jsr         SCANCMD       ;Programmname überspringen

;Parameter suchen und in cmd eintragen
;.....................................................................
.scanloop:      cmp         (d2),#$3c     ;"<"?
                jeq         .nextparam
                cmp         (d2),#$3e     ;">"?
                jeq         .nextparam
.paramcopy:     move    sf (d2),(d0)      ;1 Zeichen kopieren
                jeq         .set_env      ;Stringende-> Alle Parameter gefunden
                cmp         #32,(d2)      ;Parameter vollständig kopiert?
                inc ne      d2
                inc         d0
                jne         .paramcopy
.nextparam:     jsr         SCANCMD       ;Pointer auf nächsten Parameter stellen
                jmp         .scanloop

.set_env:       dec         d0
                cmp         #32,(d0)      ;Wenn das letzte Zeichen ein Space war
                clr     eq (d0)           ;Zeichen löschen
                push        d4
                jsr         SET_ENV       ;Environment-Variable CMD setzen
                inc         sp
                jmi         .fehler

;Ein-/Ausgabeumlenkung bearbeiten
;.....................................................................
                move        d7,d2         ;d2: Anfang des Strings
                jmp         .nextscan2

.scanloop2:     cmp         #$3c,(d2)     ;< Eingabeumlenkung
                jne         .nur_out      ;Nein, aber vielleicht Ausgabeumlenkung
                inc         d2
                cmp         #$3e,(d2)     ;> Ausgabeumlenkung?
                jne         .nur_in       ;Auch nicht

;Ein-und Ausgabeumlenkung
                inc         d2            ;<> Behandlung
                add         #256,d7,d3    ;Hat schon eine Umlenkung stattgefunden?
                cmp         (d3),-
                jne         .nextscan2    ;Ja, diese interessiert nicht mehr
                inc         d3
                cmp         (d3),-
                jne         .nextscan2

                jsr         SCANNAME      ;Namen einlesen

                push        #2            ;Datei zum lesen & schreiben öffnen
                push        d4
                jsr         OPEN
                add         #2,sp
                jmi         .dateifehler

                add         #20,akt_task,d1       ;Stdin der akt. Task
                add         #256,d7,d3
                move        (d1),(d3)             ;retten
                move        d0,(d1)               ;Geöffnete Datei wird neues Stdin
                inc         d3                    ;Ebenso für Stdout
                inc         d1
                move        (d1),(d3)
                move        d0,(d1)
                jmp         .nextscan2

;Eingabeumlenkung
.nur_in:        add         #256,d7,d3
                cmp         (d3),-        ;Hat schon eine Eingabeumlenkung stattgefunden?
                jne         .nextscan2    ;Ja, diese ignorieren
```

```
                jsr         SCANNAME

                push        -             ;Datei zum lesen öffnen
                push        d4
                jsr         OPEN
                add         #2,sp,=
                jmi         .dateifehler

                add         #20,akt_task,d1       ;Stdin der akt. Task
                add         #256,d7,d3
                move        (d1),(d3)             ;retten
                move        d0,(d1)               ;Geöffnete Datei wird neues Stdin
                jmp         .nextscan2

;Ausgabeumlenkung
.nur_out:       cmp         #$3e,(d2)     ;> Ausgabeumlenkung ?
                jne         .nextscan2    ;Auch nicht
                inc         d2

                add         #257,d7,d3
                cmp         (d3),-        ;Hat schon eine Ausgabeumlenkung stattgefunden?
                jne         .nextscan2

                jsr         SCANNAME

                push        #4            ;Datei zum anhängen öffnen
                push        d4
                jsr         OPEN
                add         #2,sp
                jmi         .dateifehler

                add         #21,akt_task,d1       ;Stdout der akt. Task
                add         #257,d7,d3
                move        (d1),(d3)             ;retten
                move        d0,(d1)               ;Geöffnete Datei wird neues Stdout

.nextscan2:     jsr         scancmd               ;Nächsten Parameter
                cmp         (d2),-
                jne         .scanloop2

;Programm laden
;.....................................................................
.loadprg:       move        d7,d2
                cmp         (d2),#38      ;"&"-Vergleich: Taskstart erwünscht?
                move ne     -,d5          ;d5 Flag, ob Unterprogramm oder Task
                clr  eq     d5
                inc  eq     d2

                jsr         SCANNAME

                push        d4
                jsr         PRG_LOAD
                inc         sp
                jpl         .startprg

;Pfad benutzen
                push        #path         ;Dateiname war nicht in der Cwd,...
                jsr         SEARCH_ENV    ;...deshalb Path benutzen
                inc         sp
                move mi     #100,d0       ;Falls kein Pfad existiert: Datei nicht gefunden
                jmi         .fehler
                move        d0,d3         ;d3: Pointer auf Pfad

.pfadloop:      cmp         (d3),-        ;Pfad zuende?
                move eq     #100,d0       ;Ja, Datei nicht gefunden
                jeq         .fehler
                move        #127,d6       ;Zähler für Zeichenanzahl max. 128
                move        d4,d1         ;Pufferzeiger

.copypfad:      cmp         (d3),-        ;Pfadende?
                jeq         .pfadok       ;Fertig bei Stringende
                cmp         #$3b,(d3)    ;Ende des Teilpfades (;)?
                inc  eq     d3
                jeq         .pfadok       ;Fertig bei Stringende
                move        (d3),(d1)     ;1 Zeichen kopieren
                inc         d3
                inc         d1
                dec      sf d6
                jpl         .copypfad
                move        #107,d0
                jmp         .fehler
```

```
.pfadok:        move       d2,d0        ;Programmnamen mit Pfad verbinden
.space5:        cmp        (d0),-       ;Stingende?
                jeq        .trytoload   ;Ja, gleich Programm ausführen
                cmp        (d0),#$20    ;Ende des Dateinamens suchen
                jeq        .trytoload   ;Ja, gleich Programm ausführen
                move       (d0),(d1)
                inc        d1
                inc        d0
                dec     sf d6
                jpl        .space5
                move       #107,d0
                jmp        .fehler

.trytoload:     clr        (d1)         ;0 Eintragen als Endekennung für open

                push       d4
                jsr        PRG_LOAD
                inc        sp
                jmi        .pfadloop

;Das Programm wird nun gestartet
;.....................................................................
.startprg:      move       d0,d6;d6:Speicherblock des Programms

                push       d7
                push       d6
                cmp        d5,-         ;Task oder Subroutine?
                jeq        .task
                sub        #2,d6,d0     ;
                move       sp,d1        ;Eigenen Stackppointerretten
                sub        #2,(d0),d0
                add        d0,d6,sp     ;Neuen Stackpointerbenutzen
                push       d1           ;aktuellen Stackpointer retten auf Subroutinen Stack
                push       #.return     ;Subroutine
                add        d6,-,=       ;jsr Programm

;Programm als neue Task starten
;.....................................................................
.task:          add        #21,akt_task,d1
                add        #257,d7,d0
                cmp        (d0),-       ;Wurde Ausgabe umgelenkt?
                push ne    (d1)         ;Ja, vererben...
                move ne    (d0),(d1)    ;...und wieder das alte Stdout benutzen
                pushz eq                ;Nein, das Programm macht keine Ausgaben
                dec        d1
                dec        d0
                cmp        (d0),-       ;Wurde eine Eingabeumlenkung vorgenommen?
                push ne    (d1)         ;Ja, neues Stdin vererben...
                move ne    (d0),(d1)    ;...und wieder das alte Stdin benutzen
                pushz eq                ;Nein, keine Std-Eingabe
                push       d4           ;Taskname
                push       d6           ;Startadresse
                push       #256         ;Größe des Environmentbereiches
                jsr        STARTTASK    ;Task starten
                add        #5,sp
                pop        d6
                pop        d7
                add pl     #257,d7,d0
                clr pl     (d0)
                dec pl     d0
                clr pl     (d0)
                jpl        .entry
                jmp        .restore

;Bei Fehler oder Ende: Alles rückgängig machen (Stdin/out/Memory)
;.....................................................................
.return:        inc        sp
                move       (sp),sp      ;Stackpointer restaurieren
                pop        d6
                pop        d7
.restore:       push       d6
                jsr        MFREE        ;Programmspeicher wieder freigeben
                inc        sp           ;Einer mehr, wegen rts des Programms
                jsr        .iorestore
                jmp        .entry

.dateifehler:   push       d0
                jsr        .iorestore
                pushz
                push       d4
                pushz
```

```
                jsr        write
                add        #3,sp
                pop        d0

.fehler:push       d0
                jsr        .iorestore
                jsr        fehler
                inc        sp
                jmp        .entry

;StandardIn+Out-Wechsel rückgängig machen
.iorestore:     add        #20,akt_task,d2
                add        #256,d7,d3
                cmp        (d3),-
                jeq        .rest_out
                push       (d2)
                jsr        CLOSE
                inc        sp
                move       (d3),(d2)
                clr        (d3)
.rest_out:      inc        d2
                inc        d3
                cmp        (d3),-
                rts eq
                push       (d2)
                jsr        CLOSE
                inc        sp
                move       (d3),(d2)
                clr        (d3)
                rts

;Shell beenden
.finish:        pushz                      ;Endemeldung ausgeben
                push       #bye
                pushz
                jsr        write
                add        #3,sp

                push       d7
                jsr        mfree
                inc        sp

                rts

;Weiterschalten von d2 (Aktuelle Kommandlineposition)
;.....................................................................
scancmd:        cmp        (d2),-          ;Stringende?
                rts eq                     ;Ja, fertig mit scannen
                cmp        (d2),#$20       ;Nicht-Spaces überlesen
                inc ne     d2
                jne        scancmd

.space:         cmp        (d2),#$20       ;Spaces Überlesen
                inc eq     d2
                jeq        .space
                rts

;Dateinamen für IO-Umlenkung lesen
;.....................................................................
SCANNAME:       move       d2,d3
                move       d4,d0           ;Pufferzeiger
.space:         cmp        (d3),-          ;Stingende?
                jeq        .redir          ;Ja, gleich Datei öffnen
                cmp        #$20,(d3)       ;Ende des Dateinamens suchen
                move ne    (d3),(d0)
                inc  ne    d3              ;Lesezeiegr weiterschalten
                inc  ne    d0              ;Pufferzeiger weiterschalten
                jne        .space

.redir:         clr        (d0)            ;Stringende erzeugen
                rts



initstring:     dw 13,10,"XShell 1.0 bereit.",13,10,0
path:           dw "path="
crs_l:          dw 27,"D",0
crs_r:          dw 27,"C",0
crs_mark:       dw 27,"j",0
crs_set:        dw 27,"k",0
clr_z:          dw 27,"K",0
newline:        dw 13,10,0
```

```
bye:            dw "Bye!",13,10,0
```

12.8. Konsolen-Treiber

```
;================================================================================
;KONSOLEN-TREIBER (Tastatur+2 Bildschirme)
;================================================================================
;Umschaltflags höchstwertiges Byte -> niederwertiges Byte, x nicht belegt
;Bit 10:print/sysrq
;Bit 9:pause/break
;Bit 8:alt-r
;Bit 7:alt-l
;Bit 6:ctrl-r
;Bit 5:ctrl-l
;Bit 4:shift-r
;Bit 3:shift-l
;Bit 2:caps-lock
;Bit 1:num-lock (Bei XMOS: Konsole 1/2 select)
;Bit 0:scroll-lock

equ keyboard_data,$7ff6
equ keyboard_ctrl,$7ff7
equ xgraph,$7fe8

;Konsolentreiber initialisieren
;--------------------------------------------------------------------------------
konsole:        move       #$20,@xgraph+6        ;CON 1 Sichtbar
                clr        vt52_seq              ;Nicht im VT52-Mode
                clr        vt52_seq2
                clr        con_in_use
                clr        active_con
                clr        umschaltflags         ;Sondertasten

                push       #.driver0             ;Treiber anmelden
                jsr        NEWDRIVER
                push       #.driver1
                jsr        NEWDRIVER
                push       #.error               ;Treiber für Fehlermeldungen
                jsr        NEWDRIVER
                add        #3,sp,=

                push       #.initstr1            ;Initstrings ausgeben
                push       #3
                jsr        .driver1+5
                push       #.initstr0
                push       #3
                jsr        .driver0+5
                add        #4,sp

;Tastaturpuffer init
                push       #32                   ;32-Worte-Ringpuffer
                jsr        MALLOC
                inc        sp
                sub        -,d0,d1
                sub        -,-,(d1)              ;Systemspeicherblock
                move       d0,t_buffer           ;Empfangspuffer
                add        #16,d0,t_buffer_end   ;soll 16 Zeichen enthalten
                move       t_buffer_end,t_buffer2
                add        #32,d0,t_buffer_end2

.buffer_clear:  sub        -,-,(d0)              ;#$ffff Schreiben
                inc        d0                    ;nächstes Wort
                cmp        d0,t_buffer_end2      ;Bis alle Worte in beiden Puffern...
                jne        .buffer_clear         ;... voll sind

                move       t_buffer,t_read_ptr   ;Lesezeiger init
                move       t_buffer,t_write_ptr  ;Schreibzeiger init
                move       t_buffer2,t_read_ptr2
                move       t_buffer2,t_write_ptr2

.keyonoff:      in         keyboard_data,-       ;Empfangsregister löschen
                move       #.tastatur_aus,1      ;IRQ1-Vektor einstellen
                move       #.t_commands,t_int_ar2 ;Zeiger auf auszugebende Bytes

                out        #0,keyboard_ctrl      ;Tastatur zurücksetzen
                add        #10,timerl,d0         ;(Aus-und-Anschalten)
.delay1:        cmp        timerl,d0
                jpl        .delay1
                in         keyboard_data,-       ;Empfangsregister löschen
                out        #2,keyboard_ctrl      ;Spannung an

                add        #16,timerl,d0
```

```
.wait_init:      cmp        timerl,d0            ;Timeout?
                 jmi        .keyonoff            ;Ja, Tastatur nochmal reseten
                 cmp        #.done,1             ;Fertig?
                 jne        .wait_init           ;Noch nicht
                 rts

.initstr0:       dw         27,"E",27,"j",27,"b2",27,"c0Konsole 0 bereit.",13,10,0
.initstr1:       dw         27,"E",27,"j",27,"b1",27,"c7Konsole 1 bereit.",13,10,0

;IRQ-Routine: 11-Bit-Modus beibehalten
.retry:          move    sf flags1,~dummy        ;Interruptflag löschen
.tastatur_aus:   out        #3,keyboard_ctrl     ;Clockleitung auf low
                 or         ~-,#512,flags1               ;Flags sichern
                 add        #2,timerl,t_int_ar   ;kurz warten
.delay2:         cmp        timerl,t_int_ar
                 jpl        .delay2
                 and        @keyboard_data,#$ff,t_int_ar ;Tastaturbyte holen
                 out        #2,keyboard_ctrl       ;Clockleitung auf tri-state
                 cmp        #$aa,t_int_ar        ;'alles ok' Byte?
                 jne        .retry                       ;Nein, nochmal versuchen

;IRQ-Routine: Kommandobytes senden
.next_byte:     cmp        (t_int_ar2),-         ;Fertig?
                 jeq        .lamps                       ;ja, zur normalen IRQ-Routine
                 out        (t_int_ar2),keyboard_data    ;Kommando senden
                 inc        t_int_ar2                    ;Zeiger inc
                 move    sf flags1,~dummy        ;Interruptflag löschen
                 or         ~-,#512,flags1               ;Flags sichern
                 and        @keyboard_data,#$ff,t_int_ar ;Byte holen
                 cmp        #$fa,t_int_ar        ;Letztes Byte ok?
                 jeq        .next_byte           ;Ja, nächstes
                 cmp        t_int_ar,#$fe        ;Letztes Byte nochmal?
                 jne        .retry                       ;nein, ALLES nochmal

                 and     sf (t_int_ar2),#$8000,t_int_ar   ;MSB betrachten
                 dec  ne    t_int_ar2                    ;Zeiger decrementieren
                 dec        t_int_ar2                    ;Zeiger decrementieren
                 jmp        .next_byte

;Initialisierungskommandos
.t_commands:     dw      $f0,32768+3      ;Code-Set 3
                 dw      $fc,32768+57     ;ALT-GR auf Make/Break
                 dw      $fc,32768+88     ;Rechte CTRL auf Make/Break
                 dw      $fb,32768+111    ;Page-up auf typematic
                 dw      $fb,32768+109    ;Page-down auf typematic
                 dw      $fd,32768+124    ;Große +Taste auf Make
                 dw      $fd,32768+20     ;CapsLock auf Make
                 dw       $f3,32768       ;Schnelles Typematic
                 dw      $f4              ;Arbeit wieder aufnehmen
                 dw      0                ;Endekennung

;Normale Tastatur-Interruptroutine IRQ-Level 1
;-------------------------------------------------------------------------------
.lamps:          out        #$ed,keyboard_data    ;Lampen-Kommando senden
                 move    sf flags1,~dummy                ;Umschalten in User Mode
                 or         ~-,#512,flags1               ;Interruptflag löschen
                 and        @keyboard_data,#$ff,t_int_ar ;Byte holen
                 cmp        t_int_ar,#$fe        ;Nicht verstanden?
                 jeq        .lamps                       ;Kommando wiederholen
                 and        #7,umschaltflags,@keyboard_data ;Lampen einstellen
                 move    sf flags1,~dummy        ;Umschalten in User Mode
                 or         ~-,#512,flags1               ;Interruptflag löschen
                 and        @keyboard_data,#$ff,t_int_ar ;Byte holen
                 cmp        t_int_ar,#$fe        ;Nicht verstanden?
                 jeq        .lamps                       ;Kommando wiederholen
.t_clr_int:     move    sf flags1,~dummy         ;Umschalten in User Moden
.done:           or         ~-,#512,flags1             ;Flags sichern
                 and        @keyboard_data,#$ff,t_int_ar ;Zeichen holen
                 cmp        t_int_ar,#$f0        ;Breakcode?
                 jeq        .t_clr_int           ;ja, ignorieren
                 cmp        t_int_ar,#20         ;caps-lock?
                 jeq        .caps_lock
                 cmp        t_int_ar,#18         ;shift-links?
                 jeq        .shift_links
                 cmp        t_int_ar,#89         ;shift-rechts?
                 jeq        .shift_rechts
                 cmp        t_int_ar,#25         ;alt-links?
                 jeq        .alt_links
                 cmp        t_int_ar,#57         ;alt-rechts?
                 jeq        .alt_rechts
                 cmp        t_int_ar,#17         ;ctrl-links?
                 jeq        .ctrl_links
```

```
                cmp         t_int_ar,#88          ;ctrl-rechts?
                jeq         .ctrl_rechts
                cmp         t_int_ar,#95          ;scroll-lock?
                jeq         .scroll_lock
                cmp         t_int_ar,#98          ;pause/break?
                jeq         .pause_break
                cmp         t_int_ar,#87          ;print/sysrq?
                jeq         .print_sysrq
                cmp         t_int_ar,#118         ;num-lock?
                jeq         .num_lock

;Es ist keine Sondertaste, sondern eine Zeichenerzeugende Taste
                bswp        -,t_int_ar,t_int_ar2          ;Tastencode in AR2 merken
                and     sf #508,umschaltflags,-
                add  eq    #.tabelle-7,t_int_ar           ;keine Sondertaste
                jeq         .zeichen_holen
                and     sf #28,umschaltflags,-           ;shift od. capslock?
                add  ne    #.tab_shift-7,t_int_ar         ;ja
                jne         .zeichen_holen
                and     sf #96,umschaltflags,-           ;ctrl?
                add  ne    #.tab_ctrl-7,t_int_ar ;ja
                jne         .zeichen_holen
                add         #.tab_alt-7,t_int_ar           ;sonst:Alt

.zeichen_holen: or         (t_int_ar),t_int_ar2           ;Tastaturpuffereintrag berechnen
                lsr         umschaltflags,-,t_int_ar      ;Konsole 1 aktiv?
                and         -,t_int_ar
                cmp         active_con,t_int_ar
                jne         .satz2                        ;Ja, im 2. Tastaturpuffer eintragen
                add     sf -,(t_write_ptr),-              ;Puffer frei?
                jne         .t_clr_int                    ;nein, fertig
                move        t_int_ar2,(t_write_ptr)               ;Taste in Puffer eintragen
                inc         t_write_ptr                  ;Schreibzeiger inc
                cmp         t_write_ptr,t_buffer_end     ;Wrap_around?
                move eq     t_buffer,t_write_ptr         ;ja
                jmp         .t_clr_int                   ;wieder von vorn

.satz2:         add     sf -,(t_write_ptr2),-             ;Puffer frei?
                jne         .t_clr_int                    ;nein, fertig
                move        t_int_ar2,(t_write_ptr2)      ;Taste in Puffer eintragen
                inc         t_write_ptr2                  ;Schreibzeiger inc
                cmp         t_write_ptr2,t_buffer_end2    ;Wrap_around?
                move eq     t_buffer2,t_write_ptr2        ;ja
                jmp         .t_clr_int                    ;wieder von vorn

.num_lock:      eor         #2,umschaltflags              ;Numlock togglen
                lsr         umschaltflags,-,t_int_ar      ;Konsole umschalten
                and         -,t_int_ar
                or          #$20,t_int_ar,@xgraph+6
                cmp         t_int_ar,active_con
                jne         .x2
                move        bcolor,@xgraph+6
                or          #$10,fcolor,@xgraph+6
                jmp         .lamps
.x2:            move        bcolor2,@xgraph+6
                or          #$10,fcolor2,@xgraph+6
                jmp         .lamps

.caps_lock:     eor        #4,umschaltflags               ;Caps-Flag toggle
                jmp         .lamps                        ;und Lampen anpassen

.shift_links:   eor        #8,umschaltflags               ;Flag togglen
                and         #$fffb,umschaltflags  ;CapsLock aus
                jmp         .lamps                        ;weiter

.shift_rechts:  eor        #16,umschaltflags              ;Flag togglen
                and         #$fffb,umschaltflags  ;CapsLock aus
                jmp         .lamps                        ;weiter

.ctrl_links:    eor        #32,umschaltflags              ;Flag toggeln
                jmp         .t_clr_int           ;PC neu einstellen

.ctrl_rechts:   eor        #64,umschaltflags              ;Flag toggeln
                jmp         .t_clr_int           ;PC neu einstellen

.alt_links:     eor        #128,umschaltflags     ;Flag toggeln
                jmp         .t_clr_int           ;PC neu einstellen

.alt_rechts:    eor        #256,umschaltflags     ;Flag toggeln
                jmp         .t_clr_int           ;PC neu einstellen

.scroll_lock:   eor        -,umschaltflags                ;Flag toggle
```

```
                jmp         .lamps                          ;PC neu einstellen

.pause_break:   eor         #512,umschaltflags     ;Flag togglen
                jmp         .t_clr_int            ;PC neu einstellen

.print_sysrq:   eor         #1024,umschaltflags    ;Flag togglen
                jmp         .t_clr_int            ;PC neu einstellen

.tabelle:   dw 0,27,0,0,0,0,9,94,0,0                                        ;7-16
            dw 0,0,60,0,$71,49,0,0,0,121,115,97,119,50,0,0                  ;17-32
            dw 99,120,100,101,52,51,0,0,32,118,102,116,114,53,0,0           ;33-48
            dw 110,98,104,103,122,54,0,0,109,$6d,$6a,$75,$37,$38,0          ;49-63
            dw 0,$2c,$6b,$69,$6f,$30,$39,0,46,$2e,$2d,$6c,$94,$70,$e1,0 ;64-79
            dw 0,132,$84,$23,$81,$27,0,0,0,0,$D,$2b,0,0,0,0                 ;80-95
            dw 0,0,0,0,0,0,8,0,49,$31,0,$34,$37,0,0,0                       ;96-111
            dw $30,$2c,$32,$35,$36,$38,47,$2F,0,13,$33,43,$2b,$39,$2a       ;112-126
            dw 0,0,0,0,0,$2D                                                ;127-132
.tab_shift: dw 0,27,0,0,0,0,9,248,0,0                                      ;7-16
            dw 0,0,62,0,81,33,0,0,0,89,83,65,87,34,0,0                      ;17-32
            dw 67,88,68,69,36,21,0,0,32,86,70,84,82,$25,0,0                 ;33-48
            dw 78,66,72,71,90,38,0,0,77,$4d,$4a,$55,$2f,$28,0               ;49-63
            dw 0,$3b,$4b,$49,$4f,$3d,$29,0,58,$3a,$5f,$4c,$99,$50,$3f,0 ;64-79
            dw 0,142,$8e,$60,$9a,$27,0,0,0,0,13,$2a,0,0,0,0                 ;80-95
            dw 0,0,0,0,0,0,8,0,0,$31,0,$34,$37,0,0,0                        ;96-111
            dw $30,$2c,$32,$35,$36,$38,47,$2f,0,13,$33,43,$2b,$39,$2a       ;112-126
            dw 0,0,0,0,0,$2d                                                ;127-132
.tab_alt:   dw 0,27,0,0,0,0,9,94,0,0                                        ;Scancodes 7-16
            dw 0,0,$7c,0,$40,49,0,0,0,121,115,97,119,$fd,0,0                ;17-32
            dw 99,120,100,101,52,51,0,0,32,118,102,116,114,53,0,0           ;33-48
            dw 110,98,104,103,122,54,0,0,109,$e6,$6a,$75,$7b,$5b,0          ;49-63
            dw 0,$2c,$6b,$69,$6f,$7d,$5d,0,46,$2e,$2d,$6c,$94,$70,$5c,0 ;64-79
            dw 0,132,$84,$23,$81,$27,0,0,0,0,$D,$7e,0,0,0,0                 ;80-95
            dw 0,0,0,0,0,0,8,0,49,$31,0,$34,$37,0,0,0                       ;96-111
            dw $30,$2c,$32,$35,$36,$38,47,$2F,0,13,$33,43,$2b,$39,$2a       ;112-126
            dw 0,0,0,0,0,$2D                                                ;127-132
.tab_ctrl:  dw 0,27,0,0,0,0,9,30,0,0                                        ;Scancodes 7-16
            dw 0,0,60,0,17,49,0,0,0,25,19,1,23,50,0,0                       ;17-32
            dw 3,24,4,5,52,51,0,0,32,22,6,20,18,53,0,0                      ;33-48
            dw 14,2,8,7,26,54,0,0,109,13,10,21,$37,27,0                     ;49-63
            dw 0,$2c,11,$9,15,$30,29,0,46,$2e,31,12,$94,16,28,0             ;64-79
            dw 0,132,$84,$23,$81,$27,0,0,0,0,$D,$2b,0,0,0,0                 ;80-95
            dw 0,0,0,0,0,0,8,0,49,$31,0,$34,$37,0,0,0                       ;96-111
            dw $30,$2c,$32,$35,$36,$38,47,$2F,0,13,$33,43,$2b,$39,$2a       ;112-126
            dw 0,0,0,0,0,$2D                                                ;127-132


;Konsolentreiber
;-------------------------------------------------------------------------------
.error:         dw          "err",0             ;Treibername "err"
                dw          1                   ;Zeichentreiber
                and     sf #2,umschaltflags,-   ;Welche Konsole ist sichtbar?
                jeq         .semwait0           ;Konsole 0
                jne         .semwait1           ;Konsole 1

.driver0:       dw          "con0"              ;Treibername "con0"
                dw          1                   ;ist ein Zeichentreiber
.semwait0:      irqoff
                move    sf con_in_use,-         ;Konsole schon aktiv?
                move eq     akt_task,con_in_use ;Falls nicht, belegen
                irqon
                jeq         .semok0
;(Hier könnte ein Fairness-Flag gestzt werden!)
                jsr         SWITCHTASK          ;Sonst warten
                jmp         .semwait0
.semok0:        move        #$30,@xgraph+6              ;CON 0 Schreibbar
                move    sf active_con,-         ;Konsole 0 schon aktiv?
                jeq         .driversame         ;Ja
                clr         active_con          ;Konsole 0 aktivieren
                jmp         .othercon           ;Variablenfeld wechseln

.driver1:       dw          "con1"              ;Treibername "con1"
                dw          1                   ;ist ein Zeichentreiber
.semwait1:      irqoff
                move    sf con_in_use,-         ;Konsole schon aktiv?
                move eq     akt_task,con_in_use ;Falls nicht, belegen
                irqon
                jeq         .semok1
;(Hier könnte ein Fairness-Flag gestzt werden!)
                jsr         SWITCHTASK          ;Sonst warten
                jmp         .semwait1
.semok1:        move        #$31,@xgraph+6              ;CON 1 Schreibbar
                move    sf active_con,-         ;Konsole 1 schon aktiv?
```

```
                jne         .driversame           ;Ja
                move        -,active_con          ;Konsole 1 aktivieren

.othercon:      irqoff
                move        t_read_ptr,d0         ;Variablenfeld wechseln
                move        t_read_ptr2,t_read_ptr
                move        d0,t_read_ptr2
                move        t_write_ptr,d0
                move        t_write_ptr2,t_write_ptr
                move        d0,t_write_ptr2
                move        t_buffer,d0
                move        t_buffer2,t_buffer
                move        d0,t_buffer2
                move        t_buffer_end,d0
                move        t_buffer_end2,t_buffer_end
                move        d0,t_buffer_end2
                move        fcolor,d0
                move        fcolor2,fcolor
                move        d0,fcolor2
                move        bcolor,d0
                move        bcolor2,bcolor
                move        d0,bcolor2
                irqon
                move        cursor_adress,d0
                move        cursor_adress2,cursor_adress
                move        d0,cursor_adress2
                move        cursor_x,d0
                move        cursor_x2,cursor_x
                move        d0,cursor_x2
                move        cursor_y,d0
                move        cursor_y2,cursor_y
                move        d0,cursor_y2
                move        vt52_seq,d0
                move        vt52_seq2,vt52_seq
                move        d0,vt52_seq2
                move        save_x,d0
                move        save_x2,save_x
                move        d0,save_x2
                move        save_y,d0
                move        save_y2,save_y
                move        d0,save_y2
                move        save_adr,d0
                move        save_adr2,save_adr
                move        d0,save_adr2

.driversame:    add         -,sp,d1
                move        (d1),d0               ;Jobnummer holen
                inc         d1
                move        (d1),d1               ;Parameter holen

;Job 1 "Zeichen lesen"
;.....................................................................
                dec     sf d0                     ;Job 1?
                jne         .nojob1               ;Nein
                move       (t_read_ptr),d0                ;Zeichen lesen
                add     sf -,d0,-                 ;doch kein Zeichen?
                jne         .ok
                move        #118,d0
                ror     sf -,-,-                  ;N-Flag setzen
                jmp         .drvexit
.ok:            sub         -,-,(t_read_ptr)              ;Zeichen im Puffer löschen
                inc        t_read_ptr                     ;Lesezeiger inc
                cmp        t_read_ptr,t_buffer_end      ;Ende des Puffers?
                move eq    t_buffer,t_read_ptr            ;Ja, wieder zum Anfang
                and         #$ff,d0,d1                    ;Low Byte betrachten
                cmp         d1,#26                        ;Ctrl-Z =EOF
                jne         .ok2
                move eq    #105,d0
                ror     sf -,-,-
                jmp         .drvexit

;Job 2 "Zeichen ausgeben"
;.....................................................................
.nojob1:        dec     sf d0             ;Job 2?
                jne         .nojob2       ;Nein
                jsr         .charout
                jmp         .ok2

;Job 3 "String ausgeben"
;.....................................................................
.nojob2:        dec     sf d0             ;Job 3?
                jne         .nojob3       ;Nein
```

```
                push       d2
                move       d1,d2        ;Stringpointer
.loop:          move    sf (d2),d1      ;Zeichen holen
                jeq        .strend      ;Stringende?
.not_yet:       jsr        .charout
                inc        d2           ;Pointer++
                jmp        .loop
.strend:        cmp        #6,vt52_seq  ;Eine Grafiksequenz ist nicht unterbrechbar
                jpl        .not_yet
                pop        d2
                jmp        .ok2

;Job 4 "Hole Zeichen und lasse es im Puffer"
;..............................................................................
.nojob3:        dec     sf d0                   ;Job 4?
                jne        .drvexit             ;Nein
                move       (t_read_ptr),d0              ;Zeichen lesen
                add     sf -,d0,-               ;doch kein Zeichen?
                jne        .ok2
                ror     sf -,-,-                ;Kein Zeichen: N-Flag setzen
                jmp        .drvexit
.ok2:           clr     sf ~-                   ;N-Flag löschen
.drvexit:       clr        con_in_use
;(Hier müßte das Fairness-Flag überprüft werden!)
                rts


;Gibt das Zeichen in d1 auf dem Bildschirm aus
;..............................................................................
.charout:       and        #255,d1
                or      sf vt52_seq,-,-         ;In einer VT52-Sequenz?
                jne        .vt52_emu            ;Ja
                sub     sf #32,d1               ;Druckbares Zeichen?
                jmi        .special             ;Nein,Spezialbehandlung
                asl        d1
                asl        d1
                asl        d1
                add        #charset,d1          ;Zeichenadresse
               move       cursor_adress,d0

                move       d0,@xgraph           ;Bildschirmadresse low übergeben
                bswp       d0,-,@xgraph+1               ;und high
                bswp       (d1),-,@xgraph+7     ;8 Pixel des Zeichens ausgeben
                add        #80,d0               ;Bildschirmadresse auf nächste Zeile
                move       d0,@xgraph           ;Bildschimradresse übergeben
                bswp       d0,-,@xgraph+1               ;Bildschimradresse übergeben
                move       (d1),@xgraph+7               ;8 Pixel des Zeichens ausgeben
                inc        d1                   ;Pointer im Zeichensatz incrementieren
                add        #80,d0               ;Bildschirmadresse auf nächste Zeile
                move       d0,@xgraph           ;das ganze noch 7 mal wiederholen
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7
                inc        d1
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7
                inc        d1
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7
                inc        d1
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
```

```
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7
                inc        d1
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7
                inc        d1
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7
                inc        d1
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                bswp       (d1),-,@xgraph+7
                add        #80,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       (d1),@xgraph+7

.one_right:     inc        cursor_adress
                dec     sf cursor_x             ;CursorX läuft rückwärts (79=linker Rand,
0=rechter Rand)
               jpl         .draw_cursor         ;Kein Zeilenumbruch
                move       #79,cursor_x         ;Cursor an Anfang neue Zeile
                sub        #80,cursor_adress
.one_down:      add        #1280,cursor_adress
                dec     sf cursor_y             ;(24=oberer Rand, 0=unterer Rand)
               jpl         .draw_cursor         ;Kein Scrolling

.scrolllock:    and     sf -,umschaltflags,-     ;Scroll-lock?
                jeq        .scrollok
                jsr        SWITCHTASK            ;Ja, Taskwechsel!
                jmp        .scrolllock

.scrollok:      clr        @xgraph               ;Bildschirm um 16 Zeilen nach oben schieben
                clr        @xgraph+1
                clr        @xgraph+2
                move       #5,@xgraph+3
                move       #255,@xgraph+4                ;32000-1280 Bytes kopieren
                move       #119,@xgraph+5
.scroll:        and     sf @xgraph,#1,-          ;Warten...
                jne        .scroll
                move       #320,d0               ;Letzte Zeile löschen
.clrline:       clr        @xgraph+7
                clr        @xgraph+7
                clr        @xgraph+7
                clr        @xgraph+7
                dec     sf d0
                jne        .clrline
                clr        cursor_y              ;Cursor auf letzte Zeile
                sub        #1280,cursor_adress

;Invertiert die Cursorlinie an aktueller Cursorposition
;..............................................................................
.draw_cursor:   add        #1200,cursor_adress,d0
                move       d0,@xgraph
                bswp       d0,-,@xgraph+1
                move       d0,@xgraph+2
                bswp       d0,-,@xgraph+3
                nor        @xgraph+7,-,@xgraph+7
                rts

;Cursorbewegungen
;..............................................................................
.cursor_right:  jsr        .draw_cursor          ;Alten Cursor löschen
                jmp        .one_right

.cursor_left:   jsr        .draw_cursor          ;Alten Cursor löschen
                dec        cursor_adress
                inc        cursor_x
                cmp        #80,cursor_x
```

```
                jne         .draw_cursor          ;Noch nicht am linken Rand
                clr         cursor_x
                add         #80,cursor_adress
.one_up:        sub         #1280,cursor_adress   ;Eine Zeile hoch
                inc         cursor_y
                cmp         #25,cursor_y
                jne         .draw_cursor          ;Noch nicht oben gewesen
                add         #1280,cursor_adress   ;Doch, Cursorposition nicht ändern
                dec         cursor_y
                jmp         .draw_cursor

.cursor_up:     jsr         .draw_cursor
                jmp         .one_up

.cursor_down:   jsr         .draw_cursor
                jmp         .one_down

;Clear-Home
;.....................................................................
.clear_screen:  clr         @xgraph               ;Bildschirm löschen (lassen)
                clr         @xgraph+1
                clr         @xgraph+7
                clr         @xgraph+2
                clr         @xgraph+3
                move        #255,@xgraph+4
                move        #124,@xgraph+5
.clearwait:     and     sf  @xgraph,#1,-          ;Warten...
                jne         .clearwait
.cursor_home:   clr         cursor_adress
                move        #79,cursor_x
                move        #24,cursor_y
                jmp         .draw_cursor

;Zeile ab Cursorposition löschen
;.....................................................................
.clline_from_c: push        d2
                move        cursor_adress,d0
                move        #16,d2
.clfc_loopy:    move        cursor_x,d1
                move        d0,@xgraph
                bswp        d0,-,@xgraph+1
.clfc_loopx:    clr         @xgraph+7
                dec     sf  d1
                jpl         .clfc_loopx
                add         #80,d0
                dec     sf  d2
                jne         .clfc_loopy
                pop         d2
                jmp         .draw_cursor

;Bildschirm ab Cursorposition löschen
;.....................................................................
.clear_from_c:  push        d2
                move        cursor_adress,d0
                move        #16,d2                ;Aktuelle Zeile bis Ende löschen
.cfc_loopy:     move        cursor_x,d1
                move        d0,@xgraph
                bswp        d0,-,@xgraph+1
.cfc_loopx:     clr         @xgraph+7
                dec     sf  d1
                jpl         .cfc_loopx
                add         #80,d0
                dec     sf  d2
                jne         .cfc_loopy
                pop         d2
                sub         #80,d0
                move        d0,@xgraph+2
                bswp        d0,-,@xgraph+3
                sub         d0,#32000,d0
                move        d0,@xgraph+4
                bswp        d0,-,@xgraph+5
.cfc_wait:      and     sf  @xgraph,#1,-          ;Warten...
                jne         .cfc_wait
                jmp         .draw_cursor

;Bildschirm ab Cursorzeile nach unten scrollen
;.....................................................................
.scroll_down:   move        #29440,d0             ;Adresse der vorletzten Zeile
                move        #30720,d1             ;Adresse der letzten Zeile
.scrdwnloop:    cmp         d1,cursor_adress      ;Fertig?
                jpl         .scrdwnend
                move        d1,@xgraph
```

```
                bswp        d1,-,@xgraph+1
                move        d0,@xgraph+2
                bswp        d0,-,@xgraph+3
                move        #255,@xgraph+4
                move        #4,@xgraph+5
.scrdwnwait:    and     sf @xgraph,#1,-          ;Warten...
                jne         .scrdwnwait
                sub         #1280,d0
                sub         #1280,d1             ;Nächste Zeile
                jmp         .scrdwnloop
.scrdwnend:     rts

;Spezialbehandlung für Zeichen<=31
;.....................................................................
.special:       add         #32,d1
                cmp         #10,d1               ;Line-Feed?
                jeq         .cursor_down

                cmp         #13,d1               ;Carrige-Return?
                jne         .no_cr               ;Nein
                jsr         .draw_cursor         ;Ja, Alten Cursor aus
                sub         cursor_x,#79,d0      ;Cursor an Anfang der Zeile
                move        #79,cursor_x
                sub         d0,cursor_adress
                jmp         .draw_cursor         ;Cursor an

.no_cr:         cmp         #9,d1                ;Tabulator?
                jne         .no_hortab           ;Nein
.next_tab:      jsr         .cursor_right                ;Ja, sooft nach links, bis TAB
erreicht
                add         -,cursor_x,d0
                and     sf #7,d0,-
                jne         .next_tab
                rts

.no_hortab:     cmp         #27,d1               ;ESC?
                add eq      -,-,vt52_seq         ;Nächstes Zeichen ist Steuerzeichen
                cmp         #12,d1               ;Form-Feed?
                jeq         .clear_screen
                cmp         #8,d1                ;Backspace?
                jeq         .cursor_left
                rts

;VT-52-Emulator
;.....................................................................
.vt52_emu:      cmp         #6,vt52_seq          ;Grafik?
                jmi         .no_graph            ;Nein
                move        cursor_adress,@xgraph ;Bildschirmadresse low übergeben
                bswp        cursor_adress,-,@xgraph+1     ;und high
                move        d1,@xgraph+7                  ;8 Pixel Grafikausgabe
                add         #80,cursor_adress             ;Bildschirmadresse auf nächste Zeile
                inc         vt52_seq
                cmp         #22,vt52_seq         ;16 Bytes übertragen?
                rts ne                           ;Noch nicht
                clr         vt52_seq             ;Ja ESC-Sequenz fertig
                sub         #1280,cursor_adress  ;Cursor-Adresse restaurieren
                jmp         .one_right

.no_graph:      cmp         #2,vt52_seq          ;Vordergrundfarbe ändern?
                jne         .no_fcol
                and         #7,d1,fcolor
                clr         vt52_seq
                lsr         umschaltflags,-,d0
                and         -,d0
                cmp         d0,active_con
                rts ne
                or          #$10,fcolor,@xgraph+6
                rts

.no_fcol:       cmp         #3,vt52_seq          ;Hintergrundfarbe ändern?
                jne         .no_bcol
                and         #7,d1,bcolor
                clr         vt52_seq
                lsr         umschaltflags,-,d0
                and         -,d0
                cmp         d0,active_con
                rts ne
                move        bcolor,@xgraph+6
                rts

.no_bcol:       cmp         #4,vt52_seq          ;Cursor-X-Position?
                jne         .no_curs_x
```

```
                sub         d1,#111,cursor_x
                inc         vt52_seq              ;Als nächstes kommt die Y-Koordinate
                rts

.no_curs_x:     cmp         #5,vt52_seq           ;Cursor-Y-Position?
                jne         .no_curs_y
                sub         #32,d1
                sub         d1,#24,cursor_y
                jsr         .draw_cursor
                clr         cursor_adress
.pos_loop:      dec     sf  d1
                jmi         .pos_finish
                add         #1280,cursor_adress
                jmp         .pos_loop
.pos_finish:    sub         cursor_x,#79,d0
                add         d0,cursor_adress
                clr         vt52_seq              ;ESC-Sequenz beendet
                jmp         .draw_cursor

.no_curs_y:     cmp         #71,d1        ;ESC G :Grafikzeichen
                move eq     #6,vt52_seq   ;Nächsten 16 Zeichen sind Grafikinformation
                rts eq
                cmp         #65,d1        ;ESC A: Cursor hoch
                clr eq      vt52_seq
                jeq         .cursor_up
                cmp         #73,d1        ;ESC I: Cursor hoch (noch nicht ganz richtig!)
                clr eq      vt52_seq
                jeq         .cursor_up
                cmp         #66,d1        ;ESC B: Cursor runter
                clr eq      vt52_seq
                jeq         .cursor_down
                cmp         #67,d1        ;ESC C: Cursor rechts
                clr eq      vt52_seq
                jeq         .cursor_right
                cmp         #68,d1        ;ESC D: Cursor links
                clr eq      vt52_seq
                jeq         .cursor_left
                cmp         #69,d1        ;ESC E: Clear Home
                clr eq      vt52_seq
                jeq         .clear_screen
                cmp         #72,d1        ;ESC H: Cursor Home
                jne         .no_cur_home
                clr         vt52_seq
                jsr         .draw_cursor
                jmp         .cursor_home
.no_cur_home:   cmp         #74,d1        ;ESC J: Bildschirm ab Cursor löschen
                clr eq      vt52_seq
                jeq         .clear_from_c
                cmp         #75,d1        ;ESC K: Zeile ab Cursor löschen
                clr eq      vt52_seq
                jeq         .clline_from_c
                cmp         #106,d1       ;ESC j: Cursorposition speichern
                jne         .no_sav_cur
                clr         vt52_seq
                move        cursor_x,save_x
                move        cursor_y,save_y
                move        cursor_adress,save_adr
                rts
.no_sav_cur:    cmp         #107,d1       ;ESC k: Cursorposition restaurieren
                jne         .no_res_cur
                clr         vt52_seq
                jsr         .draw_cursor
                move        save_x,cursor_x
                move        save_y,cursor_y
                move        save_adr,cursor_adress
                jmp         .draw_cursor

.no_res_cur:    cmp         #76,d1        ;ESC L: Zeile einfügen
                jne         .no_insline
                clr         vt52_seq
                jsr         .draw_cursor
                jsr         .scroll_down
                sub         cursor_x,#79,d0       ;Cursor an Anfang der Zeile
                move        #79,cursor_x
                sub         d0,cursor_adress
                jmp         .clline_from_c
.no_insline:    cmp         #98,d1        ;ESC b? :Vordergrundfarbe wählen
                move eq     #2,vt52_seq   ;Nächstes Zeichen ist Vordergrundfarbe
                rts eq
                cmp         #99,d1        ;ESC c? :Hintergrundfarbe wählen
                move eq     #3,vt52_seq   ;Nächstes Zeichen ist Vordergrundfarbe
                rts eq
```

```
                cmp         #89,d1        ;ESC Y?? :Cursorposition setzen
                move eq     #4,vt52_seq   ;Nächstes Zeichen ist Cursor-X-Position
                rts eq
                clr         vt52_seq
                rts

;.....................................................................
charset:bload charset.obj
```

12.9. Parallelport-Treiber

```
;=====================================================================
;PARALLELPORT
;=====================================================================
equ parallel_data,$7ff8
equ parallel_status,$7ff9

;Initialisieren des Parallelporttreibers
;---------------------------------------------------------------------
parallel:       out         #8,parallel_status    ;Init auslösen
                push        #.driver
                jsr         NEWDRIVER             ;Treiber anmelden
                inc         sp
                out         #4,parallel_status    ;Init löschen
                rts

;Parallelport-Treiber
;---------------------------------------------------------------------
.driver:        dw          "par",0               ;Treibername "par"
                dw          1                     ;ist ein Zeichentreiber
                add         -,sp,d1
                move        (d1),d0               ;Jobnummer holen
                inc         d1
                move        (d1),d1               ;Parameter holen

;Job 1 "Status lesen"
;.....................................................................
                dec     sf d0                       ;Job 1?
                jne         .nojob1                 ;Nein
                and         @parallel_status,#63,d0 ;Statusreg. lesen, 5 Bits
                rts

;Job 2 "Zeichen ausgeben"
;.....................................................................
.nojob1:        dec     sf d0           ;Job 2?
                jeq         .charout    ;Ja, Zeichen ausgeben

;Job 3 "String ausgeben"
;.....................................................................
.nojob2:        dec     sf d0           ;Job 3?
                rts  ne                 ;Nein
                move        d1,d0       ;Stringpointer
.loop:          move    sf (d0),d1      ;Zeichen holen
                rts eq
                jsr         .charout
                inc         d0          ;Pointer++
                jmp         .loop

;Gibt das Zeichen in d1 auf dem Drucker aus
;.....................................................................
.charout:       and     sf @parallel_status,#1,- ;Drucker bereit?
                jeq         .do_it               ;Ja
                jsr         SWITCHTASK           ;Nein, warten
                jmp         .charout

.do_it:         out         d1,parallel_data     ;Zeichen ausgeben
                out         #2,parallel_status   ;Strobe setzen
                out         #2,parallel_status   ;Strobe setzen
                out         #2,parallel_status   ;Strobe setzen
                out         #2,parallel_status   ;Strobe setzen
                out         -,parallel_status    ;Strobe löschen
                rts
```

12.10. ROM-Disk-Treiber

```
;=====================================================================
;ROM-Disk
;=====================================================================
romdisk:        push      #.driver              ;Treiber anmelden
                jsr       NEWDRIVER
                inc       sp
                rts
```

```
;Driver
;-------------------------------------------------------------------------
.driver:        dw          "rom",0       ;Drivername "rom"
                dw          0             ;Blockdevice
                add         -,sp,d0       ;Jobnummer holen
                move        (d0),d0

;.........................................................................
;Job 1:Lesen
                dec     sf  d0                    ;Job 1?
                jne         .nojob1               ;Nein
                add         #3,sp,d0              ;Block High muß 0 sein
                move    sf  (d0),-
                jne         .notvalid
                inc         d0
                bswp        -,(d0),d1             ;Block Low *256 holen
                add         #.romddata,d1         ;d1 zeigt auf Romdiskdatenblock
                cmp         #.romddataend,d1
                jpl         .notvalid             ;Blocknummer war zu groß
                push        d2
                inc         d0
                and         #255,(d0)             ;1. Wort von 0..255 sicherstellen
                move        (d0),d2               ;d2 ist Nummer des 1. Wortes
                add         d2,d1                 ;d1 zeigt auf 1. Wort
                inc         d0
                add         (d0),d2               ;d2 ist Nummer des (letzten+1) Wortes
                cmp         d2,#256               ;Sicherstellen, daß Block nicht verlassen
wird
                move mi     #256,d2
                dec         d0
                sub         (d0),d2               ;In d2:Anzahl der Worte
                sub         #3,d0
                move        (d0),d0               ;d0 zeigt auf Userpuffer
.copyloop:      dec     sf  d2                    ;Weitere Worte kopieren
                jmi         .exit                 ;Nein
                move        (d1),(d0)             ;1 Wort kopieren
                inc         d1
                inc         d0
                jmp         .copyloop
.exit:          clr     sf  ~-                    ;Flags löschen
                pop         d2
                rts

.notvalid:      move        #$0521,d0             ;Block not Valid
                ror     sf  -,-,-                 ;N-Flag
                rts

;.........................................................................
;Job 2:Schreiben
.nojob1:        dec     sf  d0                    ;Job 2?
                jne         .nojob2               ;Nein
.wrtprot:       move        #$0727,d0             ;Doch, aber ROM-Disk ist schreibgeschützt
                ror     sf  -,-,-
                rts

;.........................................................................
;Job 3:Kapazität ermitteln
.nojob2:        dec     sf  d0                    ;Job 3?
                jne         .nojob3               ;Nein
                add         #2,sp,d0
                move        (d0),d0               ;Userpufferadresse holen
                clr         (d0)                  ;Letzte Sektor# high
                sub         #.romddata,#.romddataend,d1 ;d1:Anzahl der Worte
                bswp        d1,-,d1               ;Anzahl der Blöcke
                inc         d0
                sub         -,d1,(d0)             ;Letzte Sektor# low
                inc         d0
                clr         (d0)                  ;Sektorgröße high
                inc         d0
                move        #512,(d0)             ;und low
                clr     sf  ~-                    ;Flags löschen
                rts

;.........................................................................
;Job 4:Format
.nojob3:        dec     sf  d0                    ;Job 4?
                jeq         .wrtprot              ;Ja, Disk ist aber schreibgeschützt
                clr     sf  ~-                    ;Alle anderen Jobs ok
                rts

;.........................................................................
```

```
.romddata:
bload romddata.obj
.romddataend:
```

12.11. SCSI-Treiber

```
;=============================================================================
;SCSI-Treiber
;=============================================================================
;Fehlermeldungen:
;107 = Device existiert nicht
equ scsi_data,$7ff4
equ scsi_ctrl,$7ff5

;-----------------------------------------------------------------------
;INITIALISIERUNG UND EINHEITEN SUCHEN
;Ermittelt alle angeschlossenen Einheit. Die Suche nach weiteren Einheiten
;wird abgebrochen sobald sich eine Einheit nicht meldet, daß heißt alle
;angeschlossenen Einheiten müssen aufeinander folgende IDs haben und mit
;id=0 beginnen. Es wird mitgeteilt, ob eine wechselbares Medium vorliegt.
;Zusätzlich wird eine Herstellerspezifische Kennung ausgegeben.
scsi:           out     -,scsi_ctrl             ;Bus Reset
                push    #2                      ;kurz warten
                jsr     DELAY
                clr     @scsi_ctrl              ;Bus Reset clear
                push    #16                     ;1 sec warten
                jsr     DELAY

                move    -,accesses              ;Zugriffszähler auf 1
                clr     removable
                clr     not_ready_active        ;Drive-not-Ready-Routine nicht aktiv
                clr     media_writeprot         ;Alle Medien beschreibbar
                clr     scsi_in_use             ;Semaphor löschen
                clr     want_to_use_scsi        ;Fairness-Flag
                push    #1000           ;Speicher für MDB (Media-Deskriptor-Blöcke),...
                jsr     MALLOC          ;...Kommando- und Sektorpuffer
                add     #3,sp           ;Stack für alle Aufrufe reparieren
                add     #266,d0,media_descr     ;Pointer auf MDB
                move    d0,scsi_buffer                  ;Pointer auf Sektorpuffer
                add     #256,d0,command_buffer ;Pointer auf Kommandopuffer

;.....................................................................
;Suche nach SCSI-Geräten
                clr     d7                      ;Zähler löschen
                move    #scsidriver0,d6         ;Pointer auf Devicedriver0
.detect_loop:   push    d7                      ;Gibt es das Device?
                jsr     TEST_UNIT_READY
                inc     sp
                cmp     d0,#107                 ;Wenn nicht, fertig
                jeq     .detect_end

                push    d6                      ;SCSI-Treiber anmelden
                jsr     NEWDRIVER
                push    scsi_buffer             ;Name des Geräts holen
                push    d7
                jsr     INQUIRY                 ;Herstellerinfos erfragen

                push    #string_init            ;"SCSI-Device" ausgeben
                push    #3
                driver  #konsole.error+5

                add     #$30,d7,d0              ;Devicenummer ausgeben
                push    d0
                push    #2
                driver  #konsole.error+5

                push    #32                     ;Space ausgeben
                push    #2
                driver  #konsole.error+5
                add     #9,sp

                move    #14,d5                  ;28 Zeichen Gerätenamen ausgeben
                add     #4,scsi_buffer,d4       ;Pointer auf Name
.nameloop:      bswp    (d4),-,d0
                push    d0                      ;Zeichen High ausgeben
                push    #2
                driver  #konsole.error+5
                push    (d4)                    ;Zeichen Low ausgeben
                push    #2
                driver  #konsole.error+5
                add     #4,sp
                inc     d4
```

```
                dec     sf d5
                jne        .nameloop

                and     sf (scsi_buffer),#128,-  ;wechselbares Medium?
                jeq        .nextdevice           ;nein
                push       #string_ch            ;"wechselbares Medium" ausgeben
                push       #3
                driver     #konsole.error+5
                add        #2,sp
                add        d7,#exponents,d0
                or         (d0),removable

.nextdevice:    push       #line_feed            ;Zeilenvorschub ausgeben
                push       #3
                driver     #konsole.error+5
                add        #2,sp

                add        #scsidriver1-scsidriver0,d6   ;nächster Treiber
                inc        d7                            ;Nächstes Device
                cmp        #8,d7
                jne        .detect_loop

.detect_end:    asl        d7           ;Ungenutzte MDB wieder freigeben
                asl        d7           ;jeder MDB belegt 8 Worte
                asl        d7
                add        #266,d7      ;+256 Sektorpuffer +10 Kommandopuffer
                push       d7
                push       scsi_buffer
                jsr        MSHRINK
                add        #2,sp

                sub        -,scsi_buffer,d0
                sub        -,-,(d0)     ;Speicherblock ist Systemblock

                rts

;---------------------------------------------------------------------------
;Kommandotabelle
;6-Byte-Befehle
c_test_unit_ready:      dw 0,0,0,0,0,0
c_request_sense:        dw 3,0,0,0,27,0
c_format_unit:          dw 4,0,0,0,3,0          ;Pauschal:Interleave 3
c_inquiry:              dw 18,0,0,0,36,0        ;bis zu 36 Bytes Daten werden gelesen
;10-Byte-Befehle
c_read:                 dw 40,0,0,0,0,0,0,0,1,0 ;Blockadresse muß noch angegeben werden
c_write_verify:         dw 46,0,0,0,0,0,0,0,1,0 ;Blockadresse muß noch angegeben werden
c_read_capacity:        dw 37,0,0,0,0,0,0,0,0,0

;Ausgabestrings:
string_init:    dw "SCSI-Device ",0
string_ch:      dw " (wechselbares Medium)",0
line_feed:      dw 10,13,0

;(2^i)-Tabelle
exponents:      dw      1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768


;8 Devicedriver dür SCSI-Geräte
;---------------------------------------------------------------------------
scsidriver0:    dw      "a",0,0,0               ;Devicename "a"
                dw      0                       ;Es ist ein Blockdevice
                move    #0,d1                   ;Devicenumber
                jmp     driversame
scsidriver1:    dw      "b",0,0,0               ;Devicename "b"
                dw      0
                move    #1,d1                   ;Devicenumber
                jmp     driversame
scsidriver2:    dw      "c",0,0,0               ;Devicename "c"
                dw      0
                move    #2,d1                   ;Devicenumber
                jmp     driversame
scsidriver3:    dw      "d",0,0,0               ;Devicename "d"
                dw      0
                move    #3,d1                   ;Devicenumber
                jmp     driversame
scsidriver4:    dw      "e",0,0,0               ;Devicename "e"
                dw      0
                move    #4,d1                   ;Devicenumber
                jmp     driversame
scsidriver5:    dw      "f",0,0,0               ;Devicename "f"
                dw      0
                move    #5,d1                   ;Devicenumber
```

```
                jmp     driversame
scsidriver6:    dw      "g",0,0,0               ;Devicename "g"
                dw      0
                move    #6,d1                   ;Devicenumber
                jmp     driversame
scsidriver7:    dw      "h",0,0,0               ;Devicename "h"
                dw      0
                move    #7,d1                   ;Devicenumber

driversame:     irqoff
                cmp         scsi_in_use,-               ;SCSI belegt?
                move eq     akt_task,scsi_in_use        ;Nein, selber belegen
                irqon
                jeq         .semok
                move        akt_task,want_to_use_scsi   ;Fairness-Flag setzen
                jsr         SWITCHTASK
                jmp         driversame

.semok:         add         -,sp,d0
                pushall
                move        d1,d7               ;SCSI-ID
                move        d0,d5
                move        (d0),d6             ;Jobnummer holen

                add         #exponents,d1
                and     sf (d1),removable,-     ;Ist Medium wechselbar?

                push        d7
                jsr  ne     TEST_UNIT_READY     ;Ist Datenträger gewechselt worden?
                inc         sp

;.....................................................................
;JOB 1: LESEN
                dec     sf d6                   ;Job "Lesen"?
                jne         .writejob           ;Nein

                add         #3,d5,d0            ;Sektor low
                push        (d0)
                dec         d0
                push        (d0)                ;Sektor high
                push        d7                  ;Unitnumber
                jsr         GETBLOCK            ;Block lesen
                add         #3,sp
                jmi         .driverexit

                add         #4,d5
                and         #$ff,(d5),d2  ;d2 ist Position des 1. Worts
                inc         d5
                add         (d5),d2,d3    ;d3 ist Position des letzten Wortes
                cmp         #$101,d3      ;Sicherstellen, daß es nicht größer als $100 ist
                move pl     #$100,d3
                sub         d2,d3         ;d3 gibt Anzahl an
                sub         #4,d5
                move        (d5),d1       ;User-Pufferadresse holen
                add         #4,d0         ;d0 zeigt auf Cachebufferdaten
                add         d0,d2         ;d2 zeigt auf 1. Wort im Cachepuffer
.same:          clr         d0            ;Kein Fehler
.copyloop:      dec     sf d3             ;Weitere Worte kopieren?
                jmi         .driverexit   ;Nö
                move        (d2),(d1)     ;Daten kopieren
                inc         d1
                inc         d2
                jmp         .copyloop

;.....................................................................
;JOB 2: SCHREIBEN
.writejob:      dec     sf d6                   ;Job "Schreiben"?
                jne         .capjob             ;Nein

                add         #exponents,d7,d1    ;Medium schreibgeschützt?
                and     sf (d1),media_writeprot,-
                move ne     #$0727,d0           ;Ja: Fehlerrückgabe...
                jne         .driverexit         ;...und fertig

                add         #3,d5,d0            ;Sektor low
                push        (d0)
                dec         d0
                push        (d0)                ;Sektor high
                push        d7                  ;Unitnumber
                jsr         GETBLOCK            ;Block lesen
                add         #3,sp
                jmi         .driverexit
```

```
                or          #8,(d0)              ;Puffer ist SchreibPuffer

                add         #4,d5
                and         #$ff,(d5),d1  ;d1 ist Position des 1. Worts
                inc         d5
                add         (d5),d1,d3    ;d3 ist Position des letzten Wortes
                cmp         #$101,d3      ;Sicherstellen, daß es nicht größer als $100 ist
                move pl     #$100,d3
                sub         d1,d3         ;d3 gibt Anzahl an
                sub         #4,d5
                move        (d5),d2       ;User-Pufferadresse holen
                add         #4,d0
                add         d0,d1         ;d1 zeigt auf 1. Wort im Cachepuffer

;Schreibzugriff auf Sektor Null? Dies wird ein High-Level-Format bedeuten.
                add         #3,d5,d0
                cmp         (d0),-
                jne         .same
                dec         d0
                cmp         (d0),-
                jne         .same

;Schreibadresse war Sektor 0 -> neuen Mediadescriptor erstellen
                push        d1
                push        d7
                sub         #7,sp
                jsr         MEDIA_CHANGED
                add         #8,sp
                pop         d1
                jmp         .same

;.....................................................................
;JOB 3: KAPAZITÄT
.capjob:        dec     sf d6                    ;Job "Kapazität ermitteln"?
                jne         .formjob             ;Nein

                inc         d5                   ;Zeiger auf Pufferadresse
                push        (d5)                 ;Pufferadresse
                push        d7                   ;Devicenummer
                jsr         READ_CAPACITY
                add         #2,sp
                jmp         .driverexit

;.....................................................................
;JOB 4: FORMAT
.formjob:       dec     sf d6                    ;Job "Format"?
                jne         .decbufjob           ;Nein

                push        d7                   ;Devicenummer
                jsr         FORMAT_UNIT
                inc         sp
                jmp         .driverexit

;.....................................................................
;JOB 5: DECREMENT BUFFERS
.decbufjob:     dec     sf d6                    ;Job "Dec Buffers"?
                jne         .flushjob            ;Nein

                inc         d5
                move        (d5),d6       ;Bis zu welcher Adresse freimachen?

.decloop:       and     sf #8,(efs),-     ;Ist es ein Lesepuffer?
                jeq         .delete       ;Ja, löschen

                add         #2,efs,d0
                push        (d0)          ;Block# Low
                dec         d0
                push        (d0)          ;Block# High
                add         #3,d0
                push        d0            ;Pufferadresse
                and         #7,(efs),d0
                push        d0            ;SCSI-Unitnumber
                jsr         SCSIWRITE     ;Sektor schreiben
                add         #4,sp

.delete:        add     sf #260,efs       ;520 Bytes an den Heap zurückgeben
                cmp         d6,efs
                jmi         .decloop
                jmp         .driverexit

;.....................................................................
;JOB 6: FLUSH
```

```
.flushjob:      dec     sf d6           ;Job "Flush"?
                move ne     #$0549,d0   ;Nein, Job unbekannt...
                jne         .driverexit ;...und fertig

                or          #8,d7       ;uns interessieren nur Schreibpuffer
                move        efs,d3      ;d3 ist Pointer auf Cache-Puffer

.flushloop:     and         #15,(d3),d0
                cmp         d7,d0       ;Schreibpuffer für dieses Device?
                jne         .noflush    ;Nö

                add         #2,d3,d0
                push        (d0)        ;Block# Low
                dec         d0
                push        (d0)        ;Block# High
                add         #3,d0
                push        d0          ;Pufferadresse
                and         #7,d7,d0
                push        d0          ;SCSI-Unitnumber
                jsr         SCSIWRITE   ;Sektor schreiben
                add         #4,sp
                and         #$fff7,(d3) ;Puffer ist nur noch Lesepuffer

.noflush:       add     sf #260,d3      ;Nächster Puffer
                cmp         #bs_start,d3
                jne         .flushloop  ;Noch nicht letzter
                clr         d0

;.....................................................................
.driverexit:    clr         scsi_in_use
                cmp         want_to_use_scsi,-  ;Fairness-Flöag überprüfen
                clr         want_to_use_scsi
                jsr ne      SWITCHTASK
                move    sf d0,-             ;Fehler aufgetreten?
                ror  ne sf -,-,-            ;Wenn ja :N-Flag
                popall
                rts

;.....................................................................
;Diese Routine sorgt dafür, daß ein angegebener Block in einem Cachepuffer ist
getblock:       pushall
                add         #7,sp,d0    ;Devicenumber holen
                or          #16,(d0),d2 ;Nur gültige Puffer interessieren
                inc         d0
                move        (d0),d3     ;Sektornummer high
                inc         d0
                move        (d0),d4     ;Sektornummer low
                ror         -,-,d5      ;Zugriffszähler des Puffers, der am längsten nicht
benutzt wurde
                move        efs,d6      ;d6 pointet auf Cachepuffer

.search:        and     sf #23,(d6),d0  ;Puffer gültig für unser Device?
                cmp         d0,d2
                jne         .nohit      ;Nein
                add         -,d6,d0     ;d0 temporärer Pointer
                cmp         d3,(d0)     ;Sektornummern high gleich?
                jne         .nohit      ;Nö
                inc         d0
                cmp         d4,(d0)     ;Sektornummern low gleich?
                jne         .nohit      ;Nö
                move        d6,d7       ;Adresse steht in d7
                jmp         .cachehit

;Puffer ist uninteressant. Aber vielleicht ist es der älteste
.nohit:         add         #3,d6,d0
                cmp         d5,(d0)     ;Ist Puffer der älteste?
                jpl         .notoldest  ;Nein, nicht verwerfen
                move        (d0),d5     ;Bis hierhin ja...
                move        d6,d7       ;...Adresse merken
.notoldest:     add     sf #260,d6      ;Nächster Puffer
                cmp         #bs_start,d6
                jne         .search     ;Noch nicht letzter

;Es ist ein Cachemiss aufgetreten. Nun muß der Puffer an Adresse (d7)
;für den geforderten Sektor herhalten.
                and         #7,d2       ;Unitnumber in d2
                and     sf #8,(d7),-    ;Ist es ein Schreibpuffer?
                jeq         .getnew     ;Nein, sofort Sektor laden

;Der am längsten nicht mehr benutzte Cachepuffer war leider ein
;Schreibpuffer -> zurückschreiben
                add         #2,d7,d0
```

```
                push       (d0)          ;Block# Low
                dec        d0
                push       (d0)          ;Block# High
                add        #3,d0
                push       d0            ;Pufferadresse
                and        #7,(d7),d0    ;SCSI-Unitnumber
                push       d0
                jsr        SCSIWRITE     ;Sektor schreiben
                add        #4,sp

;Nun kann der angeforderte Sektor gelesen werden
.getnew:        add        #2,d7,d0      ;d0 ist temporärer Pointer
                push       d4            ;Block# low pushen
                move       d4,(d0)       ;und in Pufferverwaltung eintragen
                dec        d0
                push       d3            ;Block# high pushen
                move       d3,(d0)       ;und in Pufferverwaltung eintragen
                add        #3,d0
                push       d0            ;Pufferadresse pushen
                move       d2,(d7)       ;Unitnumber in Pufferverwaltung...
                push       d2            ;...und auf den Stack
                jsr        SCSIREAD      ;Block lesen
                add        #4,sp
                jmi        .errorend     ;Bei Fehler: Ende
                or         #16,(d7)      ;Puffer ist gültig

;Der Puffer enthält nun den gewünschten Sektor
.cachehit:      add        #3,d7,d0      ;Puffer-Zugriffszähler...
                cmp        accesses,(d0) ;...gleich dem Gesamtzugriffszähler?
                jeq        .end          ;Ja, Zugriffszähler nicht ändern
                inc    sf  accesses      ;Gesamtzugriffszähler erhöhen
                move       accesses,(d0) ;Wird Puffer-Zugriffszähler
                jpl        .end          ;Kein Zählerüberlauf

;Falls Gesamtzugriffszähler überläuft (=$8000 ist), werden
;alle Pufferzugriffszähler durch 256 geteilt
                add        #3,efs,d0
.divloop:       add        #$ff,(d0)     ;255 addieren, damit keine 0en entstehen
                bswp       (d0),-,(d0)   ;/256
                add        #260,d0       ;Nächster Puffer
                cmp        #bs_start,d0
                jmi        .divloop      ;Noch nicht letzter
                move       #$81,accesses

.end:           move       d7,d0
                clr     sf ~-            ;N-Flag löschen
.errorend:      popall
                rts

;Low-Level SCSI-Funktionen
;-------------------------------------------------------------------------------
;Diese Routine kopiert das SCSI-Kommando ab Adresse (d0)
;in den Kommandopuffer (weil der im RAM-Bereich ist!)
;d6 sagt aus wann das Kommando für einen Taskwechsel unterbrochen werden kann,
;da der Zugriff lange dauert.
copy_command:   move      command_buffer,d7
                move       command_buffer,d1
                move       #10,d2
.copy:          move      (d0),(d1)
                inc        d0
                inc        d1
                dec    sf d2
                jne        .copy
                rts

test_unit_ready: pushall
                move       #c_test_unit_ready,d7  ;Kommandozeiger einstellen
                sub        -,-,d6                 ;Kommando wird nie unterbrochen
                jmp        command

format_unit:    pushall
                move       #c_format_unit,d0      ;Kommando ins RAM kopieren
                jsr        copy_command
                move       #6,d6                  ;Kommando wird unterbrochen
                jmp        command

inquiry:        pushall
                move       #c_inquiry,d7          ;Kommandozeiger einstellen
                sub        -,-,d6                 ;Kommando wird nie unterbrochen
                jmp        command

read_capacity:  pushall
```

```
                move      #c_read_capacity,d7    ;Kommandozeiger einstellen
                sub       -,-,d6                 ;Kommando wird nie unterbrochen
                jmp       command

request_sense:  pushall
                move      #c_request_sense,d7    ;Kommandozeiger einstellen
                sub       -,-,d6                 ;Kommando wird nie unterbrochen
                jmp       command

scsiwrite:      pushall
                move      #c_write_verify,d0     ;Kommando ins RAM kopieren
                jsr       copy_command
                move      #522,d6                ;Kommando bei ByteNr. 522 unterbrechen
                jmp       scsiread.same          ;... um Verify-Vorgang abzuwarten

scsiread:       pushall
                move      #c_read,d0     ;Kommando ins RAM kopieren
                jsr       copy_command
                move      #10,d6         ;Kommando bei ByteNr 10 unterbrechen, (Compare)
.same:          add       #9,sp,d0       ;Blocknummer in das Kommando eintragen
                add       #2,command_buffer,d1
                bswp      (d0),-,(d1)    ;Bits 31-24
                inc       d1
                move      (d0),(d1)      ;Bits 23-16
                inc       d0
                inc       d1
                bswp      (d0),-,(d1)    ;Bits 15-8
                inc       d1
                move      (d0),(d1)      ;Bits 7-0
                jmp       command

;Befehlsausführung
;-------------------------------------------------------------------------
;Übergibt Kommando an SCSI-Device. Bei Status 2 wird die Fehlermeldung im
;Fehlerregister error zusammengefaßt, in den Bits 11-8 steht der
;Statusschlüssel und in den Bits 7-0 der Statuscode. Die ankommenden
;Bytes werden zu Worten zusammengefaßt, dabei ist das erste Byte, das
;gesendet oder empfangen wird, das High-Byte.
command:        add        #7,sp,d0
                move       (d0),d1              ;Unitnumber holen
                and        #7,d1
                move       d1,(d0)
                inc        d0
                move       (d0),d5              ;Pufferadresse holen
                clr        d3                   ;high/low-Byte-Zeiger init
                clr        d4                   ;Kontrollzähler löschen

;Target selektieren
                add        #exponents,d1,d0     ;2 hoch Unitnumber berechnen
                move       (d0),d0
                out        d0,scsi_data         ;Target selektieren

                move       #1000,d1             ;Maximal 1000
.wait_for_busy: and     sf @scsi_ctrl,#16,-             ;SCSI-Bus busy?
                jne        .comnext             ;Ja, jetzt Daten tranferieren
                dec     sf d1                   ;Timeoutzähler
                jne        .wait_for_busy

;Target nicht verfügbar, Fehler 107
                out        #0,scsi_data         ;SCSI-Bus Clear
                move       #107,d0              ;Fehler melden
                jmp        .errorend            ;und mit N-Flag beenden

;Haupt-Datenübertragungs-Schleife
.comnext:       cmp        d4,d6                ;Kommando unterbrechen?
                jsr eq     SWITCHTASK           ;ja
                and     sf @scsi_ctrl,#8,-              ;Request?
                jeq        .comnext             ;Nein, warten
                inc        d4                   ;Kontrollzähler inc
                and     sf @scsi_ctrl,#7,d0     ;Welche Phase?
                cmp        #2,d0                ;Immer noch Command-Phase?
                jne        .nocommand           ;Nein, zur nächsten Phase
                out        (d7),scsi_data       ;Kommando-Byte ausgeben
                inc        d7                   ;Zeiger erhöhen
                jmp        .comnext

.nocommand:     cmp        #4,d0                ;Welche ist die nächste Phase?
                jeq        .getdata             ;Data-In-Phase.
                jpl        .getstatus           ;Status-In
                jmp        .putdata             ;Data-Out

.dataoutnext:   cmp        d4,d6                ;Kommando unterbrechen?
```

```
                jsr eq      SWITCHTASK          ;ja
                and     sf @scsi_ctrl,#8,-              ;Request?
                jeq         .dataoutnext        ;Nein, warten
                and     sf @scsi_ctrl,#7,-              ;Immer noch Data-Out-Phase?
                jne         .getstatus          ;Nein, zur nächsten Phase
.putdata:       bswp        (d5),-,@scsi_data   ;high-Byte ausgeben
.dataoutreq:    and     sf @scsi_ctrl,#8,-              ;Request?
                jeq         .dataoutreq         ;Nein, warten
                inc         d4                  ;Kontrollzähler inc
                out         (d5),scsi_data              ;low-Byte ausgeben
                inc         d5                  ;Datapointer incrementieren
                jmp         .dataoutnext

.datainnext:    and     sf @scsi_ctrl,#8,-              ;Request?
                jeq         .datainnext         ;Nein, warten
                and     sf @scsi_ctrl,#2,-              ;Immer noch Data-In-Phase?
                jne         .getstatus          ;Nein, zur nächsten Phase
.getdata:       in          scsi_data,d0        ;Datum holen
                bswp        -,d0,(d5)           ;im high-Byte ablegen
.datainreq:     and     sf @scsi_ctrl,#8,-              ;Request?
                jeq         .datainreq          ;Nein, warten
                and     sf @scsi_ctrl,#2,-              ;Immer noch Data-In-Phase?
                jne         .getstatus          ;Nein, zur nächsten Phase
                inc         d4                  ;Kontrollzähler inc
                and         @scsi_data,#$ff,d0  ;Datum holen
                or          d0,(d5),=           ;im low-Byte ablegen
                inc         d5                  ;Datapointer incrementieren
                jmp         .datainnext

.getstatus:     and         @scsi_ctrl,#15,d0   ;Req für Status?
                cmp         #14,d0
                and         @scsi_data,#$ff,d0  ;Datum holen
                move eq     d0,status           ;evtl als Status speichern
                and     sf @scsi_ctrl,#16,-     ;Immer noch Busy?
                jne         .getstatus          ;Ja, warten

;SCSI-Datentransfer ist beendet, Fehlerüberprüfung
                and     sf @scsi_ctrl,#32,-     ;Parityerror aufgetreten?
                move ne     #$0447,d0           ;Ja, Error ausgeben
                jne         .errorend           ;...und fertig

;SCSI-Status überprüfen
                cmp         status,-
                jeq         .end                ;SCSI-Status=0:alles ok
                cmp         #2,status
                jeq         .do_req_sense               ;SCSI-Status=2:Request Sense
                move        #$0801,d0           ;Sonst: Drive-Busy-Fehler

;Routinenende
.errorend:      ror     sf -,-,-                ;N_Flag setzen
                jmp         .sameexit
.end:           clr     sf ~-                   ;N-Flag löschen
.sameexit:      popall
                rts                             ;Fertig

;Status 2 ist aufgetreten, ein Request-Sense-Kommando wird abgesetzt.
.do_req_sense:  add         #7,sp,d0
                push        scsi_buffer         ;Adresse pushen
                push        (d0)                ;Unitnumber
                jsr         REQUEST_SENSE
                add         #2,sp
                add         -,scsi_buffer,d0
                and         (d0),#$ff00,d7              ;Hi-Byte:Statusschlüssel
                add         #5,d0
                bswp        (d0),-,d0           ;Lo-Byte:Statuscode
                or          d0,d7

;Media-Changed und Drive-not-Ready behandeln
                bswp        d7,-,d6
                cmp         #6,d6               ;UNIT-Attention?
                jne         .no6
                jsr         DRIVE_NOT_READY     ;Ja->Drive-not-Ready und...
                jsr         MEDIA_CHANGED       ;..Media-Changed aufrufen
                move        d7,d0
                jmp         .errorend

.no6:           cmp         #2,d6               ;Drive not ready?
                jne         .no2
                jsr         DRIVE_NOT_READY     ;->Drive-not-Ready aufrufen
                move        d7,d0
                jmp         .errorend
```

```
.no2:           cmp         #7,d6               ;Writeprotect?
                add  eq     #7,sp,d0            ;Ja, Write-Protectbit setzen
                add  eq     #exponents,(d0),d0
                or   eq     (d0),media_writeprot  ;Writeprotect-Bit setzen
                move        d7,d0               ;Fehlernummer vom Req-Sense
                jmp         .errorend

;Media-Changed-Routine
;------------------------------------------------------------------------
media_changed:  move    sf not_ready_active,-   ;Laufe ich schon?
                rts  ne                         ;Ja, fertig
                move        -,not_ready_active  ;Wenn nein, Semaphorbit setzen
                add         #8,sp,d0            ;Unitnumber holen
                move        (d0),d0
                add         #exponents,d0,d1    ;Medium sei erst mal
                nor         (d1),-,d1
                and         d1,media_writeprot  ;nicht schreibgeschützt

;Bootsektor (Identifikation) lesen
                push       #0                   ;Low Adresse Bootsektor
                push       #0                   ;High Adresse Bootsektor
                push       scsi_buffer          ;Pufferadresse
                push       d0                   ;Unitnumber
                jsr        SCSIREAD             ;Block lesen fehlerhaft?
                add  mi    #4,sp                ;Ja, fertig
                jmi        .fertig
                jsr        SCSIWRITE            ;Nein, Testschreiben um writeprotect zu
ermitteln
                add        #4,sp

;Medium-Deskriptor aufbauen
                add        #8,sp,d0             ;Unitnumber holen
                move       (d0),d0
                asl        d0                   ;Adresse vom Medium-Deskriptor
                asl        d0
                asl        d0
                add        media_descr,d0
                add        #8,scsi_buffer,d1    ;Adresse des Mediennamens
                push       d2
                move       #7,d2                ;7 Worte (Name) kopieren
.loop:          move       (d1),(d0)
                inc        d0
                inc        d1
                dec     sf d2
                jne        .loop
                inc        d1
                move       (d1),(d0)            ;Seriennummer kopieren
                pop        d2
.fertig:clr        not_ready_active             ;Semaphor löschen
                rts

;Drive-Not-Ready-Routine
;------------------------------------------------------------------------
drive_not_ready: move   sf not_ready_active,-   ;Laufe ich schon?
                rts  ne                         ;Ja, zurück
                move       -,not_ready_active   ;Wenn nein, Semaphorbit setzen
                push       d2
                push       d3

;Weitere Schreibzugriffe verhindern
                add        #10,sp,d0            ;Devicenumber holen
                move       (d0),d0
                add        #exponents,d0,d1
                or         (d1),media_writeprot

;Gibt es für dieses Device noch Schreibpuffer?
                or         #8,d0        ;Uns interessieren nur Schreibpuffer
                move       efs,d1       ;d1 ist Pointer auf Cachepuffer
                cmp        #bs_start,d1
                jeq        .ende        ;Falls es gar keine Puffer gibt:Fertig
.loop1:         and        #15,(d1),d2
                cmp        d0,d2        ;Ist es einer?
                jeq        .found       ;Ja, große Panik
                add        #260,d1      ;Nächster Puffer
                cmp        #bs_start,d1
                jne        .loop1       ;Nein, weitersuchen

                and        #7,d0        ;Uns interessiert nur die Devicenumber
                push       d0           ;d0 sichern
                add        #$61,d0      ;Devicename
                move       d0,(command_buffer)
                add        -,command_buffer,d0
```

```
                clr         (d0)
                push        command_buffer
                jsr         OPENED_FILES  ;Gibt es noch geöffnete Files für dieses Device?
                inc         sp
                cmp         d0,-
                pop         d0            ;d0 restaurieren
                jne         .found

;Es gibt keine Schreibpuffer -> Alle Puffer für dieses Device löschen
.clearcache:    move        efs,d1        ;d1 ist Pointer auf Cachepuffer
.loop2:         and         #7,(d1),d2
                cmp         d0,d2         ;Ist es ein interessanter Puffer?
                clr eq      (d1)          ;Ja, Puffer ungültig machen
                add         #3,d1
                clr eq      (d1)          ;und Zugriffszähler auf 0 setzen
                add         #257,d1       ;Nächster Puffer
                cmp         #bs_start,d1
                jne         .loop2        ;Nein, weitersuchen

;Es gibt keine Schreibpuffer, fertig
.ende:          pop         d3
                pop         d2
                clr         not_ready_active      ;Semaphorbit löschen
                rts

;.....................................................................
;PANIK-Routine, da Schreibpuffer oder geöffnete Dateien existieren.
.warn1:         dw      13,10,"Medium ",0
.warn2:         dw      " OHNE Schreibschutz wieder in SCSI-Device ",0
.warn3:         dw      " einlegen.",13,10,"(I)gnore (O)k.",13,10,0

;Für das nicht bereite Drive gibt es noch Schreibpuffer
.found:         and         #7,d0
                asl         d0                    ;Adresse des Media-Deskriptors
                asl         d0
                asl         d0
                add         media_descr,d0,d3

.flushkey:      push        -                     ;Job 1:Zeichen lesen
                driver      #konsole.error+5      ;Tastaturpuffer leeren
                inc         sp
                jpl         .flushkey

                push        #.warn1               ;"Medium " ausgeben
                push        #3
                driver      #konsole.error+5
                add         #2,sp

                push        d4
                move        d3,d4                 ;Adresse des Media-Deskriptors
                move        #7,d2                 ;7 Worte
.nameout:       bswp        (d4),-,d0             ;Medienname ausgeben
                push        d0
                push        #2
                driver      #konsole.error+5
                add         #2,sp

                and         #$ff,(d4),d0
                push        d0
                push        #2
                driver      #konsole.error+5
                add         #2,sp

                inc         d4
                dec      sf d2
                jne         .nameout
                pop         d4

                push        #.warn2               ;" wieder in ..." ausgeben
                push        #3
                driver      #konsole.error+5
                add         #2,sp
                add         #10,sp,d0             ;Devicenumber holen
                add         #$30,(d0),d0          ;in ASCII wandeln
                push        d0
                push        #2
                driver      #konsole.error+5
                add         #2,sp
                push        #.warn3               ;" einlegen..." ausgeben
                push        #3
                driver      #konsole.error+5
                add         #2,sp
```

```
.waitforkey:    push       -
                driver     #konsole.error+5      ;Zeichen von Tastatur holen
                inc        sp
                jmi        .waitforkey

                bswp       d0,-,d0               ;Scancode ins lowbyte
                cmp        #67,d0                ;Ignore?
                jeq        .clearcache           ;...und Cachedaten verwerfen
                cmp        #68,d0                ;Ok?
                jne        .waitforkey           ;Nein, weiter auf Taste warten

;Der User hat O.K. gedrückt
                add        #10,sp,d0     ;Devicenumber holen
                move       (d0),d0
                push       #0            ;Blocknumber low
                push       #0            ;Blocknumber high
                push       scsi_buffer   ;Adresse
                push       d0            ;Device
                jsr        SCSIREAD      ;Block lesen (evtl Unit-Attention)
                jsr        SCSIREAD      ;deshalb nochmal Block lesen
                jmi        .notestwrite
                jsr        SCSIWRITE     ;Evtl. Testschreiben, für writeprotect
.notestwrite:   add        #4,sp
                jmi        .flushkey     ;Bei Schreibschutz: Meldung erneut ausgebem

;Ist es die Diskette von vorhin?
                move       d3,d0         ;Adresse des Mediadeskriptors
                add        #8,scsi_buffer,d1     ;Adresse des Bootsektors
                move       #7,d2
.comploop:      cmp        (d0),(d1)             ;Medienname gleich?
                jne        .flushkey             ;Nein
                inc        d0
                inc        d1
                dec     sf d2
                jne        .comploop
                inc        d1
                cmp        (d0),(d1)             ;Seriennummer gleich?
                jne        .flushkey             ;Nein

;Status quo ante wurde vom User tatsächlich wieder hergestellt
                add        #10,sp,d1             ;Devicenumber holen
                add        #exponents,(d1),d1    ;Medium ist wieder
                nor        (d1),-,d1
                and        d1,media_writeprot    ;beschreibbar
                jmp        .ende
```

12.12. Serieller Schnittstellen-Treiber

```
;=============================================================================
;SERIELLE SCHNITTSTELLE
;=============================================================================
;Fehler:
;1 = Paritätsfehler
;2 = Stopbitfehler
;4 = Overrunfehler
;...und Kombinationen (Addition der Fehlerwerte)!

equ empfangsregister,$7ffd
equ senderegister,$7fff
equ controllregister,$7ffe
equ statusregister,$7ffc

;Schnittstellentreiber initialisieren
;-----------------------------------------------------------------------------
;Grundeinstellung: 9600 baud, 1 Stoppbit, DTR=0, RTS=0
seriell:        push       #16                   ;16 Worte für Empfangspuffer
                jsr        MALLOC
                inc        sp
                sub        -,d0,d1
                sub        -,-,(d1)              ;Ist ein Systemblock, Besitzer $ffff
                move       d0,s_buffer           ;Pufferanfang
                add        #16,d0,s_buffer_end   ;Pufferende
.clear_loop:    sub        -,-,(d0)              ;Empfangspuffer löschen
                inc        d0
                cmp        d0,s_buffer_end
                jne        .clear_loop
                clr        @statusregister              ;Statusregister löschen
                out        #10,controllregister  ;Controllregister setzen
                move       s_buffer,s_read              ;Lesezeiger init
                move       s_buffer,s_write      ;Schreibzeiger init
                move       #seriell_int,2               ;Interruptroutine init
```

```
                push       #.driver             ;Serielltreiber anmelden
                jsr        NEWDRIVER
                inc        sp
                clr        cts                  ;0=Sendeerlaubnis
                move       -,rts
                rts

;Seriell-Schnittstellentreiber
;-------------------------------------------------------------------------
.driver:        dw         "ser",0              ;Treibername "ser"
                dw         1                    ;ist ein Zeichentreiber
                add        -,sp,d1
                move       (d1),d0              ;Jobnummer nach d0 holen
                inc        d1
                move       (d1),d1              ;Parameter nach d1 holen

;Job 1 "Zeichen aus Puffer lesen"
;.........................................................................
                dec     sf d0                   ;Job 1?
                jne        .job2                ;Nein

                move    sf (s_read),d0          ;Zeichen lesen, setzt N-Flag falls kein
Zeichen
                move mi    #118,d0              ;Kein Zeichen, fertig
                rts  mi

                sub        -,-,(s_read)         ;Zeichen im Puffer löschen
                inc        s_read                       ;Lesezeiger++
                cmp        s_read,s_buffer_end  ;Pufferende?
                move eq    s_buffer,s_read              ;Ja, wieder auf Anfang setzen

                move    sf (s_read),-           ;Zeichen lesen, setzt N-Flag falls kein
Zeichen
                jpl        .eofcheck
                cmp        rts,-                ;Wurde ein CTRL-s von uns gesendet?
                clr        rts
                move       #17,d1                       ;CTRL-q senden, da Puffer wieder
leer!
                jsr  ne    .charout             ;ja,

.eofcheck:      and        #$ff,d0,d1           ;Niederwertiges Byte bestimmen
                cmp        d1,#26               ;EOF?
                jne        .exit                ;Nein, fertig
                move       #105,d0              ;Ja, Ende
                ror     sf -,-,-                ;N-FLag setzen
                rts

.exit:          clr     sf ~-                   ;Flags löschen
                rts


;Job 2 "Zeichen senden"
;.........................................................................
.job2:          dec     sf d0           ;Job 2?
                jeq        .charout     ;Ja

;Job 3 "String senden"
;.........................................................................
                dec     sf d0           ;Job 3?
                rts  ne                 ;Nein
                move       d1,d0        ;Stringpointer
.loop:          move    sf (d0),d1      ;Zeichen holen
                rts  eq                 ;fertig wenn 0
                jsr        .charout
                inc        d0           ;Pointer++
                jmp        .loop

;Sendet das Zeichen in d1
;.........................................................................
.charout:       cmp        cts,-        ;Senden erlaubt?
                jeq        .send        ;Ja
                jsr        SWITCHTASK   ;Nein, Taskwechsel und neu versuchen
                jmp        .charout

.send:          and     sf @statusregister,#16,- ;Sendepuffer leer?
                jne        .send                ;nein, warten
                out        d1,senderegister     ;Zeichen senden
                rts

;Interroutine IRQ-Level 2
;Empfangen eines Zeichens von der seriellen Schnittstelle
;-------------------------------------------------------------------------
```

```
seriell_int:    or         ~-,#1024,flags2               ;Flags sichern
                and        @statusregister,#7,s_error    ;Fehlerüberprüfung
                and        @empfangsregister,#$ff,s_int_ar ;Zeichen ins Arbeitsregister laden
                clr        @statusregister               ;Statusregister löschen

                cmp        #19,s_int_ar          ;CTRL-s?
                move eq    -,cts                 ;Aufhören zu Senden!
                jeq        .ende
                cmp        #17,s_int_ar          ;CTRL-q?
                clr  eq    cts                   ;Senden wieder erlaubt!
                jeq        .ende

                move    sf (s_write),-           ;Puffer frei?
                jpl        .ende                 ;Nein, Interruptroutine verlassen

                bswp       -,s_error,=
                or         s_error,s_int_ar,(s_write)    ;Zeichen in Puffer schreiben
                inc        s_write                       ;Schreibpointer++
                cmp        s_write,s_buffer_end          ;Pufferende?
                move eq    s_buffer,s_write              ;ja, Zeiger auf Pufferanfang

                add        -,s_write,s_int_ar            ;3 Worte look-ahead für Puffer voll
                cmp        s_int_ar,s_buffer_end         ;Pufferende?
                move eq    s_buffer,s_int_ar             ;ja, Zeiger auf Pufferanfang
                inc        s_int_ar                      ;Schreibpointer++
                cmp        s_int_ar,s_buffer_end         ;Pufferende?
                move eq    s_buffer,s_int_ar             ;ja, Zeiger auf Pufferanfang
                inc        s_int_ar                      ;Schreibpointer++
                cmp        s_int_ar,s_buffer_end         ;Pufferende?
                move eq    s_buffer,s_int_ar             ;ja, Zeiger auf Pufferanfang
                move    sf (s_int_ar),-                  ;Frei?
                jmi        .ende

.send:          and     sf @statusregister,#16,-         ;Sendepuffer leer?
                jne        .send                 ;nein, warten
                out        #19,senderegister     ;CTRL-s senden
                move       -,rts                 ;Flag setzen

.ende:          move   sf  flags2,~dummy         ;Interrupt-Flag löschen
                jmp        seriell_int           ;PC neu einstellen
```

12.13. Dienstprogramme

12.13.1. ATTRIB

```
include lib
;------------------------------------------------------------------------------
;ATRRIB
;------------------------------------------------------------------------------
attrib:         push       #cmd
                sys        #10          ;CmdLine suchen
                inc        sp
                cmp        (d0),-       ;Dateiname mit angegeben?
                move eq    #111,d0      ;Nein, CmdLine ist leer
                jeq        .fehler

;Dateinamen in CmdLine isolieren
                move       #puffer,d1
.loop:          cmp        (d0),-
                jeq        .flags
                cmp        (d0),#32
                jeq        .flags
                move       (d0),(d1)
                inc        d0
                inc        d1
                jmp        .loop

;Flags nach Dateinamen suchen, Reihenfolg egal
.flags:         clr        (d1)
.loop2:         cmp        (d0),-
                jeq        .open
                cmp        (d0),#$57    ;W?
                or  eq     #4,flags
                cmp        (d0),#$77    ;w?
                or  eq     #4,flags
                cmp        (d0),#$58    ;X?
                or  eq     #2,flags
                cmp        (d0),#$78    ;x?
                or  eq     #2,flags
                cmp        (d0),#$44    ;D?
```

```
                or  eq      -,flags
                cmp         (d0),#$64     ;d?
                or  eq      -,flags
                inc         d0
                jmp         .loop2

;Datei öffnen
.open:          push        -
                push        #puffer
                sys         #12           ;OPEN durchführen
                add         #2,sp
                jmi         .fehler
                move        d0,fdb

;Flags setzen
                push        flags
                push        fdb
                sys         #17
                add         #2,sp
                jmi         .fehler

;DAtei schließen
.ende:          push        fdb
                sys         #15           ;CLOSE
                inc         sp
                rts

.fehler:        push        d0
                sys         #29           ;Fehlerausgabe aufrufen
                inc         sp
                jmp         .ende

cmd:            dw "cmd="
flags:          dw 0                      ;Dateiflags
fdb:            dw 0                      ;Handle
data
puffer: 256+128                           ;Puffer + Stacktiefe auf 64 Worte festgelegt
```

12.13.2. BCOLOR

```
include lib
;------------------------------------------------------------------------------
;BCOLOR
;------------------------------------------------------------------------------
;BCOLOR verändert die Hintergrundfarbe der aktuellen Konsole
bcolor:         push        #cmd
                sys         #10                   ;CmdLine suchen
                inc         sp

;Als Parameter sind nur Zahlen zwischen 0 und 7 zulässig
                cmp         #$30,(d0)             ;Zahl >= Null?
                move mi     #111,d0               ;Nein, Parameterfehler
                jmi         fehler
                cmp         (d0),#$37             ;Zahl <= 7?
                move mi     #111,d0               ;Nein, Parameterfehler
                jmi         fehler

                move        (d0),backcolor+2      ;Farbwert in String eintragen

;Vt52-Sequenz ausgeben
                pushz
                push        #backcolor
                pushz
                sys         #14           ;WRITE
                add         #3,sp
                rts pl

fehler:         push        d0
                sys         #29           ;Fehlerausgabe aufrufen
                inc         sp
                rts

cmd:            dw "cmd="
backcolor:      dw 27,"c",0,0             ;Vt52-Sequenz für Hintergrundfarbe
data
stack: 64                                 ;Stacktiefe auf 64 Worte festgelegt
```

12.13.3. CD

```
include lib
;------------------------------------------------------------------------------
;CD
```

```
;------------------------------------------------------------------------------
;C(hange) D(irectory) wechselt in das angegebene Verzeichnis
change_dir:     push       #cmd
                sys        #10          ;CmdLine suchen
                inc        sp
                cmp        (d0),-       ;Verzeichnis mit angegeben?
                move eq    #111,d0      ;Nein, CmdLine ist leer
                jeq        .fehler

;CD aufrufen
                push       d0
                sys        #24          ;CD durchführen
                inc        sp
                rts pl

.fehler:        push       d0
                sys        #29          ;Fehlerausgabe aufrufen
                inc        sp
                rts

cmd:            dw "cmd="
data
stack: 128                              ;Stacktiefe auf 128 Worte festgelegt
```

12.13.4. CLS

```
include lib
;------------------------------------------------------------------------------
;CLS
;------------------------------------------------------------------------------
;CL(ear) S(creen) löscht den aktuellen Bildschirm
cls:            pushz
                push       #clr_home    ;VT52-Sequenz ausgeben
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                rts pl

                push       d0
                sys        #29          ;Fehlerausgaben aufrufen
                inc        sp
                rts

clr_home:       dw 27,"E",0             ;VT52-Sequenz für Bildschirm löschen
data
stack: 64                               ;Stacktiefe auf 64 Worte festgelegt
```

12.13.5. COPY

```
include lib
;------------------------------------------------------------------------------
;COPY
;------------------------------------------------------------------------------
;COPY kopiert eine oder mehere Dateien. Es besteht auch die Möglich beim
;Kopiervorgang Dateien umzubenennen. Verzeichnisse können nicht kopiert werden.
;Beim Kopiervorgang wird darauf geachtet, daß niemals zwei Dateien im gleichen
;Verzeichnis den selben Namen haben.
copy:           push       #cwd                ;CWD suchen
                sys        #10
                inc        sp
                move mi    #111,d0              ;Es existiert keine CWD
                jmi        .fehler              ;... deshalb Fehler
                move       d0,d5                 ;cwd_zeiger: Zeiger auf cwd
                move       d0,d6                 ;gleiches gilt für cwd_zeiger2
                push       #cmd
                sys        #10                  ;CmdLine suchen
                inc        sp
                move       d0,CmdLine           ;CmdLine: Zeiger auf String
                cmp        (CmdLine),-          ;Ist kein Parameter angegeben?
                move eq    #111,d0              ;Nein, Parameter fehler
                jeq        .fehler

;Verschiedene Pufferbereiche einstellen
                move       #quellpuffer,quellpfad                ;Quell_pfad wird auf den
Anfang eines Directory-Pfades zeigen
                move       #zielpuffer,zielpfad          ;zielpfad wird auf den Anfang der
kopierten Datei zeigen

;Quelleverzeichnis bestimmen
                cmp        (CmdLine),#$2f               ;Ist Pfad mit angegeben?
                jeq        .explizit            ;ja,
.cwd:           cmp        (d5),-               ;Nein, Cwd in den Quellpfad kopieren
```

```
                move ne    (d5),(quellpfad)
                inc  ne    d5
                inc  ne    quellpfad
                jne        .cwd

.explizit:      cmp        (CmdLine),-          ;Angegeben Namen an Quellpfad anschließen
                move eq    #111,d0
                jeq        .fehler
                cmp        (CmdLine),#32         ;" "=Abbruch, an gegebenen Pfad kopieren
                clr  eq    (CmdLine)            ;Trenzeichen in Suchstring erzeugen
                move eq    CmdLine,CmdPos       ;Position des Trennzeichens merken für
Restauration der CmdLine
                move ne    (CmdLine),(quellpfad)
                inc        CmdLine              ;In CmdLine auf zweiten Parameter
weiterschalten
                inc  ne    quellpfad
                jne        .explizit

;Vom Pfad den Suchstring isolieren und den Pfad des entsprechendes
;Verzeichnisses isolieren
                move       CmdLine,such_strg
.cut:           cmp        (quellpfad),#$2f     ;letzten Teil löschen
                clr        (quellpfad)
                dec ne     quellpfad            ;quellpfad:letzte Position der Quelle
                dec ne     such_strg
                jne        .cut

;Ziel bestimmen
.ziel:          cmp        (CmdLine),#$2f       ;Ist Pfad mit angegeben?
                jeq        .zexplizit           ;ja...
                cmp        (CmdLine),#$2e       ;Ist aktuelles Verzeichnis gemeint?
                jne        .zcwd                ;nein...
                inc        CmdLine
                cmp        (CmdLine),-
                dec ne     CmdLine

;CWD für Zielpfad kopieren
.zcwd:          cmp        (d6),-
                move ne    (d6),(zielpfad)     ;Cwd kopieren
                inc  ne    d6
                inc  ne    zielpfad
                jne        .zcwd

;Angegebenes Pfadstück an Quellpfad anschließen
.zexplizit:     move    sf (CmdLine),(zielpfad)
                inc  ne    CmdLine
                inc  ne    zielpfad
                jne        .zexplizit

;Soll auch der Name geändert werden?
                dec        zielpfad
                cmp        (zielpfad),#$2f              ;Endet der Pfad mit /?
                inc  eq    zielpfad             ;Falls ja, soll der Name nicht geändert
werden
                move eq    #$2a,new_name
                add  eq    -,#new_name,d0
                clr  eq    (d0)
                jeq        .open

;Neuen Namen isolieren und nach new_name kopieren
.zcut:          cmp        (zielpfad),#$2f              ;letzten Teil des Zielpfades löschen
                clr ne     (zielpfad)
                dec ne     zielpfad
                dec ne     CmdLine
                jne        .zcut
                inc        zielpfad
                move       #new_name,d0
.namecopy:      move    sf (CmdLine),(d0)
                inc        CmdLine
                inc        d0
                jne        .namecopy

;Quellverzeichnis öffnen
.open:          push       -                        ;zum Lesen öffnen
                push       #quellpuffer
                sys        #12                       ;OPEN,Quellverzeichnis
                add        #2,sp
                jmi        .restauriert
                move       d0,quell_dir              ;quell_dir: FDB-Adresse

                move       #$2f,(quellpfad)          ;Ein "/" an Quellpfad anhängen
                inc        quellpfad
```

```
;Suchstring vorbereiten mit PREPARE_SEARCH
                push       quell_dir                    ;FDB-Adresse
                push       #such_strg_puffer             ;Pufferadresse
                push       such_strg                    ;Zeiger auf Namen
                sys        #26                          ;Suchstring vorbereiten
                add        #3,sp
.restauriert:   move       #32,(CmdPos)                 ;CmdLine restaurieren
                jmi        .fehler

;Zu kopierende Dateien suchen
.searchread:    add        #8,#lesepuffer,d7    ;Zeiger auf Dateinamen

;Abbruchkommando  (CTRL-C) überprüfen
                push       -                    ;EinZeichen von StIn lesen
                dec        sp
                pushz
                sys        #13                  ;READ
                add        #3,sp
                and     sf #$ff,d0              ;Nur ASCII-Code betrachten
                cmp        d0,#3                ;CTRL-C?
                jeq        .abbruch             ;Ja, Befehl abbrechen

;Nächsten passenden Eintrag suchen
                push       #such_strg_puffer     ;Zeiger auf Suchstring
                push       quell_dir             ;FDB-Adresse
                sys        #27                   ;SEARCH_NEXT
                add        #2,sp
                jpl        .ok                   ;Alles ok
                cmp        d0,#105               ;EOF?
                jeq        .ende
                jmp        .fehler

;Verzeichniseintrag lesen
.ok:            push       #16                          ;16 Wörter
                push       #lesepuffer
                push       quell_dir                    ;FDB-Adresse
                sys        #13                          ;READ
                add        #3,sp
                jmi        .fehler

;Dateinamen an Quell- und Zielpfad anhängegen, gegebenenfalls Zielnamen verändern
                move       #8,d2                  ;Zähler
                move       quellpfad,d1
                move       zielpfad,d0
                move       #new_name,d3
.copy:          bswp       (d7),-,(d1)            ;High-Byte kopieren
                bswp       (d7),-,(d0)
                cmp        (d3),#$3f              ;Bei ? Namen nicht verändern
                inc eq     d3
                jeq        .next
                cmp        (d3),#$2a              ;Bei * Namen nicht verändern
                jeq        .next
                move       (d3),(d0)
                inc        d3
.next:          inc        d1
                inc        d0
                and        #$ff,(d7),(d1)         ;Low-Byte kopieren
                and        #$ff,(d7),(d0)
                cmp        (d3),#$3f              ;Bei ? Namen nicht verändern
                inc eq     d3
                jeq        .next2
                cmp        (d3),#$2a              ;Bei * Namen nicht verändern
                jeq        .next2
                move       (d3),(d0)
                inc        d3
.next2:         inc        d1
                inc        d0
                inc        d7
                dec     sf d2                          ;Schon 16 Bytes kopiert?
                jne        .copy                       ;Nein, wiederholen
                clr        (d1)                        ;Stringendezeichen anfügen
                clr        (d0)                        ;Stringendezeichen anfügen

;Quellnamen ausgeben
                move       #quellpuffer,d0
                jsr        .stringout
                jmi        .fehler

;Quelle zum Lesen öffnen
                push       -
                push       #quellpuffer
```

```
                sys         #12                 ;Quelle öffnen
                add         #2,sp
                jmi         .fehler
                move        d0,quell_fdb

;D(irectory)-Flag überpfrüfen
                add         #15,d0              ;Directories können nicht kopiert werden
                and     sf -,(d0),-             ;Dir?
                move ne     #111,d0
                jne         .dir                ;Ja Fehlermeldung ausgeben

;Ziel zum Neuschreiben öffnen
                push        #3                  ;zum Neuschreiben öffnen
                push        #zielpuffer
                sys         #12                 ;Ziel öffnen
                add         #2,sp
                jpl         .go_on              ;Falls kein Fehler weiter bei go_on

                cmp         d0,#110             ;Existiert der Dateiname schon?
                jne         .fehler             ;Nein anderer Fehler

;Fehlermeldung, daß Datei schon existiert ausgeben
                move        #exist_schon,d0
                jsr         .stringout          ;Stringausgeben
                jmi         .fehler
                jmp         .close2             ;Quelldatei wieder schließen

;Aus Quelldatei lesen
.go_on:         move        d0,ziel_fdb
                move        -,d4                 ;Eof auf False
.copy3:         push        #2048                ;Block einlesen
                push        #kopierpuffer
                push        quell_fdb
                sys         #13                 ;READ
                add         #3,sp
                jpl         .write
                cmp         d0,#105             ;EOF-Fehler?
                jne         .spezialfehler      ;Nein, Spezialbehandlung
                clr         d4

;An Zieldatei Anzahl der gelesenen Worte schreiben
.write:         cmp         d1,-                ;Länge gleich Null?
                jeq         .test               ;ja, nichts schreiben
                push        d1                  ;Blockablegen
                cmp         -,d1                ;Länge gleich Eins?
                push ne     #kopierpuffer        ;Nein,Adresse übergeben
                push eq     kopierpuffer         ;Ja, Zeichen übergeben
                push        ziel_fdb
                sys         #14                 ;WRITE
                add         #3,sp
                jmi         .fehler
.test:          cmp         d4,-
                jne         .copy3

;Dateiflags auch kopieren
                push        quell_fdb
                sys         #16                 ;GET-FLAGS

                push        d0
                push        ziel_fdb
                sys         #17                 ;SET-FLAGS
                add         #3,sp
                jmi         .fehler

;" kopiert nach " ausgeben
                move        #nach,d0
                jsr         .stringout
                jmi         .fehler

;kompletten Zielnamen ausgeben
                move        #zielpuffer,d0
                jsr         .stringout
                jmi         .fehler

;Linefeed ausgeben
                move        #lf,d0
                jsr         .stringout
                jmi         .fehler

;die geöffneten Dateien wieder schließen
.close:         push        ziel_fdb
                sys         #15                 ;CLOSE, Ziel schließen
```

```
                inc         sp
.close2:        push        quell_fdb
                sys         #15                 ;CLOSE, Quelle schließen
                inc         sp
                jmp         .searchread         ;Nächsten zu kopierende Datei suchen

;Fehlerstring bei Directory ausgeben
.dir:           move        #dir_fehler,d0
                jsr         .stringout
                jmi         .fehler
                jmp         .close

;Bei anderem Fehler als EOF wird die zum Teil kopierte Datei wieder gelöscht,
;da sie noch unvollständig ist.
.spezialfehler: move        d0,d3
                push        ziel_fdb
                sys         #21                 ;Gefundene Datei löschen
                inc         sp
                move        d3,d0

.fehler:        push        d0
                sys         #29                 ;Fehlerstringausgabe
                inc         sp

;Quellverzeichnis und Dateien schließen
.ende:          push        quell_dir
                sys         #15                 ;Quellverzeichnis schließen
                push        quell_fdb
                sys         #15                 ;Quelldatei schließen
                push        ziel_fdb
                sys         #15                 ;Zieldatei schließen
                add         #3,sp
                rts

;String bei Befehlsunterbrechung ausgeben
.abbruch:       move        #break,d0
                jsr         .stringout
                jmp         .ende

;String ausgeben
.stringout:     pushz                           ;String ausgeben
                push        d0
                pushz                           ;an StandardOut
                sys         #14                 ;WRITE
                add         #3,sp
                rts



cmd:                    dw "cmd="
cwd:                    dw "cwd="
quell_dir:              dw 0    ;FDB-Adresse des Quellverzeichnisses
quell_fdb:              dw 0    ;FDB-Adresse der Quelldatei
ziel_fdb:               dw 0    ;FDB-Adresse der Zieldatei
such_strg:              dw 0    ;Zeiger auf den Suchstring
lf:                     dw 13,10,0      ;Linefeed
nach:                   dw " kopiert nach ",0
dir_fehler:             dw " ist Verzeichnis und kann nicht kopiert werden!",13,10,0
exist_schon:            dw " existiert schon!",13,10,0
break:                  dw 13,10,"Befehlsunterbrechung.",13,10,0

data
quellpuffer: 256         ;Puffer für Quelldateipfad
quellpfad: 1             ;Zeiger aif ein Zeichen nach dem Quellpfad, dort muß der Dateinamen
angehängt werden
zielpuffer: 256          ;Puffer für Zieldateipfad
zielpfad: 1              ;Zeiger auf den Pufferbereich, der den Zielpfad enthält (256 Worte)
kopierpuffer: 2048       ;Kopierpuffer für 4 Sektoren
lesepuffer: 16           ;Lesepuffer für Verzeichniseintrag
new_name: 16             ;Neuer Name bei Namensänderung
such_strg_puffer: 8      ;Suchstring (komprimierter Name für Search-Next)
CmdLine: 1               ;Zeiger auf Parameter in der CmdLine
CmdPos: 1                ;Zeiger für Position des gelöschten Space in der CmdLine
stack: 128               ;Stackbereich
```

12.13.6. DEL

```
include lib
;------------------------------------------------------------------------
;DEL
;------------------------------------------------------------------------
;DEL(ete) löscht Dateien, keine Verzeichnisse!
```

```
del:            push       #cwd                ;Cwd suchen
                sys        #10                 ;SEARCH_ENV
                inc        sp
                move mi    #111,d0             ;Falls keine CWD existiert Fehler
                jmi        .fehler
                move       d0,cwd_zeiger

;CmdLine suchen
                push       #cmd
                sys        #10                 ;SEARCH_ENV
                inc        sp
                move       d0,CmdLine          ;CmdLine:Zeiger auf String

;Pufferzeiger initialisieren
                move       #puffer,Pfad
                add        #256,#puffer,lesepuffer

;Pfad bestimmen
                cmp        (CmdLine),#$2f              ;Ist Pfad mit angegeben?
                jeq        .explizit
.cwd:           cmp        (cwd_zeiger),-
                move ne    (cwd_zeiger),(pfad)  ;Cwd kopieren
                inc  ne    cwd_zeiger
                inc  ne    pfad
                jne        .cwd

.explizit:      cmp        (CmdLine),-          ;" "=Abbruch, an gegebenen Pfad kopieren
                move       (CmdLine),(pfad)
                inc  ne    CmdLine
                inc  ne    pfad
                jne        .explizit

;Suchstring und Quelverzeichnis isolieren
                move       CmdLine,such_strg
.cut:           cmp        (pfad),#$2f
                clr        (pfad)
                dec ne     pfad                ;pfad:letzte position der Quelle
                dec ne     such_strg
                jne        .cut
                inc        such_strg

;Quellverzeichnis zum Lesen öffnen
                push       -                   ;zum lesen öffnen
                push       #puffer             ;Dateipfad
                sys        #12                 ;OPEN
                add        #2,sp
                jmi        .fehler
                move       d0,Quell_fdb        ;Quell_fdb: FDB-Adresse

;Ist ein Suchstring mit angegeben? Falls nein, selber einen * einsetzen
                cmp        CmdLine,such_strg   ;Ist Suchstring mit angegeben?
                add  eq    #puffer,#264,d0     ;Wenn nicht, * einsetzen
                move eq    d0,such_strg
                move eq    #$2a,(d0)
                inc  eq    d0
                clr  eq    (d0)

                cmp        (such_strg),#$2a    ;Ist * unser Suchstring?
                jne        .ok                 ;Nein, weiter machen

;Falls alle Dateien gelöscht werden sollen Warnmeldung ausgeben
                move       #warnung,d0
                jsr        .stringout
                jmi        .fehler

;Zeichen von StandardIn einlesen
.getloop:       push       -                   ;Ein Zeichen einlesen, j=ok
                dec        sp
                pushz
                sys        #13                 ;READ
                add        #3,sp
                jmi        .getloop
                move       d0,help

;Dieses Zeichen wieder ausgeben
                push       -                   ;Zeichen ausgeben
                push       d0
                pushz
                sys        #14                 ;WRITE
                add        #3,sp
                jmi        .fehler
```

```
;Falls diese Zeichen ein "j" oder "J" war weiter, sonst Abbruch
                and         #$ff,help             ;Nur Zeichencode betrachten
                cmp         help,#$4a             ;J?
                jeq         .ok                   ;ja
                cmp         help,#$6a             ;j?
                jne         .ende                 ;auch nicht!, ende

;Suchstring vorbereiten
.ok:            push        Quell_fdb
                push        lesepuffer            ;Pufferadresse
                push        such_strg             ;Zeiger auf Namen
                sys         #26                   ;PREPARE_SEARCH
                add         #3,sp
                jmi         .fehler

;Linefeed ausgeben
                move        #lf,d0
                jsr         .stringout
                jmi         .fehler

                add         #8,lesepuffer,datei_info      ;datei_info:Lesepuffer
.searchread:    add         #16,lesepuffer,Dateiname      ;Dateiname:Zeiger auf Dateinamen

;Abbruchkommando  (CTRL-C) überprüfen
                push        -             ;EinZeichen von StIn lesen
                dec         sp
                pushz
                sys         #13           ;READ
                add         #3,sp
                and     sf #$ff,d0        ;Nur ASCCI-Code interessiert
                cmp         d0,#3         ;CTRL-C =Abbruch
                jeq         .abbruch

;Nächste passende Datei suchen
                push        lesepuffer            ;Zeiger auf Suchstring
                push        Quell_fdb             ;FDB-Adresse
                sys         #27                   ;SEARCH_NEXT
                add         #2,sp
                jmi         .ende                 ;Falls keinen gefunden -> Ende


;Verzeichniseintrag lesen
                push        #16                   ;16 Wörter
                push        datei_info            ;Lesepuffer
                push        Quell_fdb             ;FDB-Adresse
                sys         #13                   ;READ
                add         #3,sp

;gefundenen Namen an Pfad anhängen
                move        #8,count                        ;Zähler
                move        pfad,help
                move        #$2f,(help)                     ;"/" anfügen
                inc         help
.copy:          bswp        (Dateiname),-,(help)            ;High-Byte kopieren
                inc         help
                and         #$ff,(Dateiname),(help)                 ;Low-Byte kopieren
                inc         help
                inc         Dateiname
                dec     sf count                            ;Schon 16 Bytes kopiert?
                jne         .copy                           ;Nein weitermachen
                clr         (help)                          ;Stringendezeichen anfügen

;kompletten Pfadnamen ausgeben
                move        #puffer,d0
                jsr         .stringout
                jmi         .fehler

;Gefundene Datei zum Schreiben öffnen um Writeprotect und Zugriffkonflikt festzustellen
                push        #2                    ;Zum Schreiben öffnen
                push        #puffer
                sys         #12                   ;OPEN
                add         #2,sp
                jpl         .dateiok              ;Kein Fehler, ->weiter
                cmp         d0,#114               ;Writeprotect?
                jne         .fehler               ;Nein anderer Fehler

;Fehlerstring ausgeben
                move        #wp_fehler,d0
                jsr         .stringout
                jmi         .fehler
                jmp         .close                ;Danach Datei wieder schließen
```

```
;Handelt es sich bei dieser Datei um ein Verzeichnis?
.dateiok:       move        d0,datei_fdb
                add         #15,d0
                and     sf -,(d0),-             ;Handelt es sich um eine Directory?
                jne         .dir                ;ja, Fehlerstring ausgeben

;Datei löschen
                push        datei_fdb
                sys         #21                 ;DELETE
                inc         sp
                jmi         .fehler

;Erfolgsmeldung ausgeben
                move        #meldung,d0
                jsr         .stringout
                jmi         .fehler

;Datei schließen, um FDB freizugeben
.close:         push        datei_fdb
                sys         #15                 ;Gefundene Datei wieder schließen
                inc         sp

                jmp         .searchread         ;Nächste Datei suchen

;Fehlerstring ausgeben
.dir:           move        #dir_fehler,d0
                jsr         .stringout
                jmi         .fehler
                jmp         .close

.fehler:push       d0
                sys         #29                 ;Fehlerausgabe aufrufen
                push        datei_fdb
                sys         #15                 ;CLOSE, Datei schließen
                add         #2,sp

.ende:          push        Quell_fdb
                sys         #15                 ;CLOSE, Verzeichnis schließen
                inc         sp

;Linefeed ausgeben
                move        #lf,d0
                jsr         .stringout
                rts

;Abruchmeldung ausgeben
.abbruch:       move        #break,d0
                jsr         .stringout
                jmp         .ende

;String an StandardOut ausgeben
.stringout:     pushz                           ;String ausgeben
                push        d0
                pushz
                sys         #14                 ;WRITE
                add         #3,sp
                rts


cmd:            dw "cmd="
cwd:            dw "cwd="
cwd_zeiger:     dw 0    ;Zeiger auf CWD
pfad:           dw 0    ;Zeiger auf Dateipfad (256 Worte Puffer)
CmdLine:        dw 0    ;Zeiger in der CmdLine
lesepuffer:     dw 0    ;Zeiger auf Suchstring
datei_info:     dw 0    ;Zeiger auf Verzeichniseintrag im Puffer (16 Worte)
Dateiname:      dw 0    ;Zeiger auf Dateinamen in Verzwichniseintrag
such_strg:      dw 0    ;Zeiger auf Suchstring
quell_fdb:      dw 0    ;FDB-Adresse des Verzeichnisses
datei_fdb:      dw 0    ;FDB-Adresse der zu löschenden Datei
count:          dw 0    ;Zähler
help:           dw 0    ;Hilfsvariable
lf:             dw 13,10,0
meldung:        dw " entfernt.",13,10,0
dir_fehler:     dw " ist Verzeichnis und kann nicht entfernt werden!",13,10,0
wp_fehler:      dw " ist schreibgeschützt!",13,10,0
warnung:        dw 13,10,"Alle Dateien löschen? (j/n) ",0
break:          dw 13,10,"Befehlsunterbrechung.",13,10,0
data
puffer: 280+128         ;Puffer und Stackbereich
```

12.13.7. DIR

```
include lib
;-------------------------------------------------------------------------------
;DIR
;-------------------------------------------------------------------------------
;DIR(ectory) zeigt den Inhalt ein angegeben Verzeichnisses an.
dir:            push       #cwd                ;Cwd suchen
                sys        #10                 ;SEARCH_ENV
                inc        sp
                move mi    #111,d0
                jmi        .fehler
                move       d0,cwd_zeiger
;CmdLine suchen
                push       #cmd
                sys        #10                 ;SEARCH_ENV
                inc        sp
                move       d0,CmdLine          ;CmdLine: Zeiger auf String

;Pufferzeiger initialisieren
                move       #puffer,Pfad
                add        #256,#puffer,lesepuffer

;Verzeichnispfad ermitteln
                cmp        (CmdLine),#$2f              ;Ist Pfad mit angegeben?
                jeq        .explizit
.cwd:           cmp        (cwd_zeiger),-
                move ne    (cwd_zeiger),(pfad)  ;Cwd kopieren
                inc  ne    cwd_zeiger
                inc  ne    pfad
                jne        .cwd

.explizit:      cmp        (CmdLine),-          ;" "=Abbruch
                move       (CmdLine),(pfad)
                inc  ne    CmdLine
                inc  ne    pfad
                jne        .explizit

;Suchstring isolieren
                move       CmdLine,such_strg
.cut:           cmp        (pfad),#$2f
                clr        (pfad)
                dec ne     pfad                ;pfad:letzte position der Quelle
                dec ne     such_strg
                jne        .cut
                inc        such_strg

;Verzeichnis zum Lesen öffnen
                push       -                   ;zum lesen öffnen
                push       #puffer             ;Dateipfad
                sys        #12                 ;OPEN
                add        #2,sp
                jmi        .fehler
                move       d0,Quell_fdb        ;Quell_fdb: FDB-Adresse

;Ist ein Suchstring mit angegeben, falls nein "*" einsetzen
                cmp        CmdLine,such_strg   ;Ist Suchstring mit angegeben
                add  eq    #puffer,#264,d0
                move eq    d0,such_strg
                move eq    #$2a,(d0)
                inc  eq    d0
                clr  eq    (d0)

;Suchstring vorbereiten für SEARCH_NEXT
                push       Quell_fdb
                push       lesepuffer          ;Pufferadresse
                push       such_strg           ;Zeiger auf Namen
                sys        #26                 ;PREPARE_SEARCH
                add        #3,sp
                jmi        .fehler
                move       lesepuffer,such_strg

;Headline ausgeben
                move       #begin,d0
                jsr        .stringout
                jmi        .fehler

                add        #264,#puffer,lesepuffer

;Nächsten Eintrag suchen
.searchread:    push       such_strg           ;Zeiger auf Suchstring
                push       Quell_fdb           ;FDB-Adresse
                sys        #27                 ;SEARCH_NEXT
```

```
                add         #2,sp
                jmi         .endestring
;Verzeichniseintrag laden
                push        #16                 ;16 Wörter
                push        lesepuffer          ;Lesepuffer
                push        Quell_fdb           ;FDB-Adresse
                sys         #13                 ;READ
                add         #3,sp
                jmi         .fehler

;Dateinamen zeichenweise ausgeben
                move        #8,count            ;Zähler
                add         #8,lesepuffer,help
                cmp         (help),-            ;Eintrag gültig?
                jeq         .searchread         ;Nein, nächsten suchen

.nameout:       bswp    sf (help),-,d0          ;High-Byte ausgeben
                move eq     #32,d0
                jsr         .out
                jmi         .fehler

                and     sf #$ff,(help),d0       ;Low Byte ausgeben
                move eq     #32,d0
                jsr         .out
                jmi         .fehler

                inc         help
                dec     sf count                ;Schon 16 Bytes ausgegeben?
                jne         .nameout

                move        #32,d0              ;Space ausgeben
                jsr         .out
                jmi         .fehler

;Dateilänge ausgeben
                add         #2,lesepuffer,high  ;Dateilänge High ermitteln
                add         -,high,low
                bswp        -,(high),help
                or          help,(low),low              ;Dateilänge LOW in Worten ermitteln
                bswp        (high),-,high

                clr         nul
                move        high,zahl           ;Dateilänge high ausgeben
                jsr         .zahl
                jmi         .fehler

                move        low,zahl            ;Dateilänge low ausgeben
                jsr         .zahl
                jmi         .fehler

                move        #space8,d0          ;8 Spaces ausgeben
                jsr         .stringout
                jmi         .fehler

;WXD Flags ausgeben
                add         #5,lesepuffer,help  ;Zeiger auf Flags
                and     sf #4,(help),-          ;Flags ausgeben
                move eq     #32,d0
                move ne     #$57,d0             ;"W"=Writeprotect
                jsr         .out
                jmi         .fehler

                move        #space3,d0          ;3 Spaces ausgeben
                jsr         .stringout
                jmi         .fehler

                and     sf #2,(help),-
                move eq     #32,d0
                move ne     #$58,d0             ;"X"=Executable
                jsr         .out
                jmi         .fehler

                move        #space3,d0          ;3 Spaces ausgeben
                jsr         .stringout
                jmi         .fehler

                and     sf -,(help),-
                move eq     #32,d0
                move ne     #$44,d0             ;"D"=Directory
                jsr         .out
                jmi         .fehler
```

```
                move       #lf,d0               ;Linefeed ausgeben
                jsr        .stringout
                jmi        .fehler
                jmp        .searchread          ;nächsten Eintrag suchen


;Zahl-Ausgabe-Routine, wortbezogen und hexadezimal
.zahl:          move       #4,count
.zahlloop:      rol        zahl                 ;4 auszugebende Bits in den unteren Teil des
Bytes
                rol        zahl
                rol        zahl
                rol        zahl
                and        #15,zahl,d0          ;Nur 4 Bits betrachten
                cmp        #10,d0               ;Buchstabe oder ZAhl ausgeben?
                add mi     #$30,d0              ;Zahl zwischen 0..9
                sub pl     #10,d0               ;Buchstabe zwischen A..F
                add pl     #$41,d0
                cmp        d0,#$30              ;Falls Null führende Nullen entfernen
                move ne    -,nul         ;Falls <>0 Führende Nullen Flag auf False
                jne        .zahlout
                cmp        nul,-         ;Falls Führende Nullen Flag True 0 durch Space
ersetzen
                move eq    #32,d0
.zahlout:       jsr        .out
                rts  mi

                dec     sf count                ;4 Ziffern ausgegeben?
                jne        .zahlloop
                rts

.out:           push       -                    ;Zeichen ausgeben
                push       d0
                pushz
                sys        #14                  ;WRITE
                add        #3,sp
                rts

;String an StandardOut ausgeben
.stringout:     pushz                           ;Stringout ausgeben
                push       d0
                pushz
                sys        #14
                add        #3,sp
                rts

;Ende String ausgeben
.endestring:    move       #endestrg,d0
                jsr        .stringout
                jmp        .ende

.fehler:push       d0
                sys        #29                  ;Fehlerausgabe aufrufen
                inc        sp

.ende:          push       Quell_fdb
                sys        #15                  ;CLOSE, Quellverzeichnis wieder schließen
                inc        sp
                rts


cmd:            dw "cmd="
cwd:            dw "cwd="
cwd_zeiger:     dw 0    ;Zeiger auf CWD
CmdLine:        dw 0    ;Zeiger auf CmdLine
pfad:           dw 0    ;Zeiger auf Verzeichnispfad (256 Worte Puffer)
such_strg:      dw 0    ;Zeiger auf Suchstring
lesepuffer:     dw 0    ;Zeiger auf Suchstring-Puffer (8 Worte)
Quell_fdb:      dw 0    ;FDB-Adresse des Verzeichnisses
count:          dw 0    ;Zähler
high:           dw 0    ;Enthält Dateilänge in Worten High
low:            dw 0    ;Enthält Dateilänge in WOrten Low
zahl:           dw 0    ;Enthält auszugebende Zahl
nul:            dw 0    ;Führende Nullen Flag (0=True, 1=False)
help:           dw 0    ;Hilfvariable
lf:             dw 13,10,0
endestrg:       dw 13,10,"Ende des Verzeichnisses.",13,10,0
space8:         dw "        ",0
space3:         dw "   ",0
begin:          dw 10,13,"Dateiname        Länge (hex)    Wrp Exe Dir",13,10
                dw "-------------------------------------------",13,10,0
data
```

```
puffer: 280+128          ;Puffer und Stackbereich
```

12.13.8. DISKINFO

```
include lib
;-------------------------------------------------------------------------------
;DISKINFO
;-------------------------------------------------------------------------------
;DISKINFO gibt den Datenträgernamen, die Speicherkapazität und die Anzahl der
;freien Worte des Datenträgers aus
diskinfo:       push       #cmd
                sys        #10                  ;CmdLine suchen
                inc        sp
                cmp        (d0),-               ;Falls CmdLine leer -> Fehler
                move eq    #111,d0
                jeq        .fehler

;Treiberadresse bestimmen
                push       d0
                sys        #8                   ;SEARCHDRIVER
                inc        sp
                jmi        .fehler
                move       d0,d5

;Handelt es sich auch um ein Blockdevice?
                dec        d0
                and     sf -,(d0),-             ;Zeichendevice?
                move ne    #111,d0
                jne        .fehler

;Name des Datenträgers ermitteln
                push       #8
                push       #8
                pushz
                pushz
                push       #puffer
                push       -
                driver     d5
                add        #6,sp
                jmi        .fehler

;String ausgeben
                move       #name,d0
                jsr        .stringout
                jmi        .fehler

;Datenträgernamen zeichenweise ausgeben
                move       #puffer,d4
                move       #8,d3
.nameloop:      bswp    sf (d4),-,d0            ;High-Byte ausgeben
                jeq        .cap
                jsr        .out
                jmi        .fehler

                and     sf #$ff,(d4),d0         ;Low-Byte ausgeben
                jeq        .cap
                jsr        .out
                jmi        .fehler

                inc        d4
                dec     sf d3                   ;Schon 16 Bytes ausgegeben
                jne        .nameloop

;Kapazität ermitteln
.cap:           push       #puffer
                push       #3                   ;JOB 3:read capacity
                driver     d5
                add        #2,sp
                jmi        .fehler

;Kapazitätsstring ausgeben
                move       #kap,d0              ;"Worte auf Datenträger insgesamt: "
                jsr        .stringout
                jmi        .fehler

;Kapazität umrechnen in Wortgröße
                clr        d6
                add        -,#puffer,d1         ;Anzahl der Worte berechen
                add     sf -,(d1)               ;Wegen SCSI Angabe muß noch ein Sektor
hinzugefügt werden
                add     cs -,puffer
                bswp       (d1),-,d0
```

```
                bswp        -,(d1)
                bswp        -,puffer
                add         d0,puffer

;Wortanzahl High ausgeben
                move        puffer,d2
                jsr         .zahlout
                jmi         .fehler

;Wortanzahl Low ausgeben
                add         -,#puffer,d0
                move        (d0),d2
                jsr         .zahlout
                jmi         .fehler

;Freispeicher ermitteln, dazu BAM durchsuchen
                push        -                       ;Länge der Bam einlesen
                push        #17
                pushz
                pushz
                push        #BAMlng
                push        -
                driver      d5
                add         #6,sp
                jmi         .fehler


;Bam Sektoren der Reihe nach einlesen und Anzahl der freien Sektoren ermitteln
.loop:          push        #256                    ;BAM-Sektor einladen
                pushz
                push        sektornr
                pushz
                push        #puffer
                push        -
                driver      d5
                add         #6,sp
                jmi         .fehler

                move        #puffer,d0
                move        #256,d7

.add:           cmp         (d0),#$ffff   ;Wort voll belegt?
                jeq         .next         ;ja, weiter
                cmp         (d0),-        ;Wort ganz leer?
                jne         .bits         ;Nein, einzelne Bits auswerten
                add  eq sf #4096,low      ;Ein freies Wort = 16 Bit *256 Worte =4096 Worte
                add  cs     -,high
                jmp         .next

.bits:          move        -,d1
                move        #16,d3
.bitloop:       and      sf d1,(d0),-
                add  eq sf #256,low       ;Pro Bit gibt es 256 freis Worte auf dem Medium
                add  cs     -,high
                asl         d1
                dec      sf d3
                jne         .bitloop

.next:          inc         d0
                dec      sf d7            ;Alle Worte des Sektors untersucht?
                jne         .add          ;Nein, weiter machen

                inc         sektornr      ;BAM-Sektornr inc.
                dec      sf BAMlng        ;Existieren noch weiter BAM-Sektoren?
                jne         .loop         ;ja,...

;Freispeicherstring ausgeben
                move        #freispeicher,d0
                jsr         .stringout
                jmi         .fehler

;Anzahl der freien Worte ausgeben
                clr         d6            ;Führende Nullen Flag auf TRUE
                move        high,d2       ;Worte high ausgeben
                jsr         .zahlout
                jmi         .fehler

                move        low,d2        ;Worte low ausgeben
                jsr         .zahlout
                jmi         .fehler

                move        #lf,d0        ;Linefeed ausgeben
```

```
                jsr         .stringout
                jmi         .fehler
                rts

;Zahl-Ausgabe-Routine
.zahlout:       move        #4,d3
.zahlloop:      rol         d2
                rol         d2
                rol         d2
                rol         d2
                and         #$f,d2,d0
                cmp         #10,d0          ;Zahl odeer Buchstabe?
                add mi      #$30,d0         ;Zahl!
                add pl      #55,d0          ;Buchstabe!

                cmp         d0,#$30         ;Führende Null?
                move ne     -,d6
                jne         .ausgeben       ;Nein ausgeben

                cmp         d6,-
                move eq     #32,d0          ;JA, Space ausgeben

.ausgeben:      jsr         .out
                rts mi

                dec      sf d3
                jne         .zahlloop
                rts

;Zeichenausgabe
.out:           push        -               ;Zeichen ausgeben
                push        d0
                pushz
                sys         #14             ;WRITE
                add         #3,sp
                rts

;String an StandardOut ausgeben
.stringout:     pushz
                push        d0
                pushz
                sys         #14             ;WRITE
                add         #3,sp
                rts

.fehler:        push        d0
                sys         #29             ;Fehlerausgabe
                inc         sp
.ende:          rts


cmd:            dw "cmd="
high:           dw 0    ;Auszugebende Länge High
low:            dw 0    ;Auszugebende Länge Low
sektornr:       dw 1    ;Sektornummer des aktuellen BAM-Sektors
lf:             dw 10,13,0
freispeicher:   dw 13,10,"Worte verfügbar: $",0
name:           dw "Medienname: ",0
kap:            dw 13,10,"Worte insgesamt: $",0

data
BAMlng: 1               ;BAM-Länge in Sektoren
puffer: 256+128         ;Puffer und Stackbereich
```

12.13.9. DRVLST

```
include lib
;-------------------------------------------------------------------------------
;DRVLST
;-------------------------------------------------------------------------------
;DR(i)V(er)L(i)ST gibt alle im System angemeldeten Treiber aus.
drvlst:         move        27,driverlist ;27:Adresse des Zeigers auf die Treiberliste

;linefeed an StandardOut
                pushz
                push        #lf
                pushz
                sys         #14                 ;WRITE
                add         #3,sp
                jmi         .fehler

;Namen eines Treibers zeichenweise ausgeben
```

```
.loop:          move       #4,count
                move       (driverlist),help
.nameloop:      move    sf (help),d0
                move eq    #32,d0
                jsr        .out
                jmi        .fehler
                inc        help
                dec     sf count
                jne        .nameloop

                add        #4,(driverlist),d0

;Block oder Zeichendevice ausgeben
                pushz
                cmp        (d0),-               ;Blockdevice?
                push eq    #bd
                push ne    #zd
                pushz
                sys        #14                  ;WRITE
                add        #3,sp
                jmi        .fehler

;Zeiger in Treiberliste weiterschalten
                inc        driverlist
                dec     sf count                ;Maximal 32 Treiber sind vorhanden
                jeq        .ende
                cmp        (driverlist),-
                jne        .loop
                rts

;Zeichen ausgeben
.out:           push       -
                push       d0
                pushz
                sys        #14                  ;WRITE
                add        #3,sp
                rts

.fehler:        push       d0
                sys        #29                  ;Fehlerausgabe
                inc        sp
.ende:          rts

driverlist:     dw 0            ;Zeiger in Treiberliste
count:          dw 32           ;Zähler für Treiber
help:           dw 0            ;Hilfsvariable
lf:             dw 10,13,0
zd:             dw ": Zeichendevice",13,10,0
bd:             dw ": Blockdevice",13,10,0
data
stack:  64                      ;Stackbereich
```

12.13.10. FCOLOR

```
include lib
;------------------------------------------------------------------------------
;FCOLOR
;------------------------------------------------------------------------------
;FCOLOR verändert die Hintergrungfarbe des aktuellen Bildschirms
fcolor:         push       #cmd
                sys        #10                  ;CmdLine suchen
                inc        sp

;Als Parameter sind nur Zahlen zwischen 0 und 7 zulässig
                cmp        #$30,(d0)            ;<0 ?
                move mi    #111,d0              ;Parameterfehler
                jmi        fehler
                cmp        (d0),#$37            ;>7 ?
                move mi    #111,d0              ;Parameterfehler
                jmi        fehler

                move       (d0),forwardcolor+2  ;Farbwert eintragen

;Vt52-Sequenz ausgeben
                pushz
                push       #forwardcolor
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                rts pl

fehler:         push       d0
```

```
                sys         #29          ;Fehlerausgabe aufrufen
                inc         sp
ende:           rts

cmd:            dw "cmd="
forwardcolor:   dw 27,"b",0,0            ;VT52-Sequenz für Farbänderung
data
stack: 64                                ;Stackbereich
```

12.13.11. FMLL

```
include lib
;------------------------------------------------------------------------------
;FMLL
;------------------------------------------------------------------------------
;F(or)M(at) L(ow) L(evel)
format_low:     move        #warnung,d0  ;Wanrstring ausgeben
                jsr         .stringout
                jmi         .fehler

;Ein Zeichen einlesen, bei "j" oder "J" weiter, sonst Abbruch
.getloop:       push        -            ;Ein Zeichen einlesen, j=ok
                dec         sp
                pushz
                sys         #13          ;READ
                add         #3,sp
                jmi         .getloop
                move        d0,d2

;Zeichen wieder ausgeben
                push        -
                push        d0
                pushz
                sys         #14          ;WRITE
                add         #3,sp
                jmi         .fehler

;ASCII-Code überprüfen
                and         #$ff,d2      ;Nur Zeichencode betrachten
                cmp         d2,#$4a      ;J?
                jeq         .ok          ;ja
                cmp         d2,#$6a      ;j?
                rts ne                   ;auch nicht!, ende

;CmdLine suchen
.ok:            push        #cmd
                sys         #10          ;SEARCH_ENV
                inc         sp
                cmp         (d0),-
                move eq     #111,d0
                jeq         .fehler

;Devicedriveradresse bestimmen
                push        d0
                sys         #8           ;SEARCHDRIVER
                inc         sp
                jmi         .fehler
                move        d0,d7

;Arbeitsmeldung ausgeben
                move        #busy,d0
                jsr         .stringout
                jmi         .fehler

;Formatkommando ausgeben
                push        #4           ;Job 4:Format
                driver      d7           ;SCSI-Format-Unit
                inc         sp
                jmi         .fehler

;Erfolgsmeldung ausgeben
                move        #meldung,d0
                jsr         .stringout
                rts pl

.fehler:push       d0
                sys         #29          ;Fehlerausgabe aufrufen
                inc         sp
                rts

;String an StandardOut ausgeben
.stringout:     pushz
```

```
                push       d0
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                rts


cmd:            dw "cmd="
busy:           dw 10,13,"Formatiere...",0
meldung:        dw 10,13,"Low-Level-Format erzeugt.",0
warnung:        dw "Alle Daten werden gelöscht! Sind Sie sicher? (j/n) ",0
data
stack: 128                      ;Stackbereich
```

12.13.12. FORMAT

```
include lib
;------------------------------------------------------------------------------
;FORMAT
;------------------------------------------------------------------------------
;FORMAT erstellt nur Rootdirectory und BAM, kein Low-Level-Format.
format:         move       #warnung,d0   ;Warnmeldung ausgeben
                jsr        stringout
                jmi        fehler

;Ein Zeichen einlesen, bei "j" oder "J" weiter, sonst Abbruch
.getloop:       push       -             ;Ein Zeichen einlesen, j=ok
                dec        sp
                pushz
                sys        #13           ;READ
                add        #3,sp
                jmi        .getloop
                move       d0,anzahl     ;Anzahl wird mißbraucht

;Zeichen wieder ausgeben
                push       -
                push       d0
                pushz
                sys        #14           ;WRITE
                add        #3,sp
                jmi        fehler

;ASCII-Code überprüfen
                and        #$ff,anzahl   ;Nur Zeichencode betrachten
                cmp        anzahl,#$4a   ;J?
                jeq        .ok           ;ja
                cmp        anzahl,#$6a   ;j?
                jne        ende          ;auch nicht!, ende

;CmdLine suchen
.ok:            push       #cmd
                sys        #10           ;SEARCH_ENV
                inc        sp

                move       d0,CmdLine
                move       d0,device     ;Zeiger auf Gerätenamen

;Treibernamen isolieren
loop1:          cmp        (CmdLine),-   ;Falls CmdLine leer -> Fehler
                move eq    #111,d0
                jeq        fehler
                cmp        (CmdLine),#32 ;Abbruch
                clr  eq    (CmdLine)     ;0 in Env-Variable erzeugen
                move eq    CmdLine,CmdPos        ;Position für Restauration der CmdLine
merken
                inc        CmdLine
                jne        loop1

;Treiberadresse bestimmen
                push       device
                sys        #8            ;SEARCHDRIVER
                inc        sp
                move       #32,(CmdPos)  ;CmdLine Restaurieren
                jmi        fehler
                move       d0,device
                dec        d0

                and     sf -,(d0),-      ;Blockdevice?
                move ne    #111,d0       ;Nein, Fehler
                jne        fehler

;Dateinamen überprüfen
```

```
                add         #8,#puffer,d0 ;Dateinamen kompremieren
                push        d0
                push        CmdLine
                sys         #28           ;SCAN, Dateiname überprüfen
                add         #2,sp
                jmi         fehler

;read cpacity in Arbeitspuffer
                push        #puffer
                push        #3            ;Job 3:read capacity
                driver      device
                add         #2,sp
                jmi         fehler

;SCSI-Angabe gibt letzte Sektornummer an, deswegen muß für die Anzahl der
;Sektoren noch eins hinzuaddiert werden
                add         -,#puffer,d1
                move        puffer,d0     ;Kapazitaet hi
                add      sf -,(d1),d1     ;Kapazitaet lo
                add cs      -,d0          ;Eins mehr wegen SCSI-Macke

;Länge der BAM in Sektoren, Wortanzahl im letzten Sektor und Bitposition
;im letzten Wort bestimmen
                asl         d0            ;Kap-hi*16
                asl         d0
                asl         d0
                asl         d0            ;d0=Anzahl Sektoren bzgl. Kap hi
                add         -,d0,sektoren
                bswp        d1,-,d0
                asr         d0
                asr         d0
                asr         d0
                asr         d0
                add         d0,sektoren   ;sektoren=(Kaphi*16)+(Kaplow div 4096)+1
                and         #15,d1,bit    ;Bit=Bitposition im letzten BAM-Wort
                asr         d1
                asr         d1
                asr         d1
                asr         d1
                and         #$ff,d1,worte ;d1=Anzahl der Worte im letzen Sektor

;Bootsektor erstellen
bootsektor:     move        #puffer,d1    ;Zeiger auf Arbeistpeicher
                move        #237,d0       ;Ende des Sektorpuffers
                clr         (d1)          ;Primärsektor der Rootdir
                inc         d1
                clr         (d1)
                inc         d1
                clr         (d1)          ;Länge der Rootdir
                inc         d1
                clr         (d1)
                inc         d1
                clr         (d1)          ;Schachtelungstiefe der Rootdir
                inc         d1
                move        -,(d1)                ;D-Flag setzen
                inc         d1
                move        28,(d1)       ;Kennung, Timerh
                inc         d1
                move        29,(d1)       ;Kennung, Timerl
                add         #10,d1        ;Dateiname ist schon eingetragen, überspringen
;               move       random,(d1)    ;Prüfwort
                move        28,(d1)       ;Hier timerl
                inc         d1
                move        sektoren,(d1) ;Laenge BAM in Sektoren
                inc         d1
                move        -,(d1)        ;Beginn der Sektorsuche
clearboot:      inc         d1            ;Rest des Verwaltungssektor...
                clr         (d1)          ;...löschen
                dec      sf d0
                jne         clearboot

;Puffer in Cache schreiben
                push        #256          ;256 Worte
                pushz                     ;Ab Position 0
                pushz                     ;Sektornummer lo
                pushz                     ;Sektornummer hi
                push        #puffer      ;Pufferadresse
                push        #2            ;Job 2: Schreiben
                driver      device
                add         #6,sp
                jmi         fehler
```

```
;1. BAM Sektor erstellen
                move       #puffer,d1    ;Anfang des Puffers
                clr        d0            ;Anzahl der Worte, die in der BAM belegt sind
                add        -,sektoren,sysbelegung        ;BAM+BOOTSEKTOR
belegte_worte:  sub     sf #16,sysbelegung
                move pl    #$ffff,(d1)           ;>16 belegt Sektoren
                inc  pl    d1
                jpl        belegte_worte

;Hinzu kommen <16 belegte Sektoren, d.h. Sektoren die kein ganzes BAM-Wort belegen
                add  mi    #16,sysbelegung
                clr        (d1)
                move       -,d0
belegt:         dec    sf  sysbelegung
                or   pl    d0,(d1)
                asl  pl    d0
                jpl        belegt

;Rest des Sekotrs löschen
                add        #puffer,#255,d0       ;Ende des Pufferbereiches
loeschen:       inc        d1
                clr        (d1)
                cmp        d1,d0
                jpe        loeschen

;Sektor speichern
                push       #256          ;256 Worte
                pushz                    ;Ab Position 0
                push       -             ;Sektornummer lo
                pushz                    ;Sektornummer hi
                push       #puffer       ;Pufferadresse
                push       #2            ;Job 2: Schreiben
                driver     device
                add        #6,sp
                jmi        fehler

;BAM-Sektoren, die ganz gegelöscht werden
                cmp        -,sektoren    ;Existiert nur ein BAM-Sektor?
                jeq        same          ;Ja
                sub        #2,sektoren,anzahl
                move       #2,sektoren   ;sektoren wird jetzt als Sektornummer benutzt, 2.ter
BAM-Sektor

loop2:          dec     sf anzahl
                jmi        last
                move       #puffer,d1
                move       #256,d0       ;Ende des Pufferbereiches

;Sektor ganz löschen
clearbam:       dec     sf d0
                clr pe     (d1)
                inc pe     d1
                jpe        clearbam

;Sektor abspeichern
                push       #256          ;256 Worte
                pushz                    ;Ab Position 0
                push       sektoren      ;Sektornummer lo
                pushz                    ;Sektornummer hi
                push       #puffer       ;Pufferadresse
                push       #2            ;Job 2: Schreiben
                driver     device
                add        #6,sp
                jmi        fehler

                inc        sektoren
                jmp        loop2

;letzten BAM-Sektor erstellen, nicht existente Sektoren müssen als
;belegt gekennzeichnet werden
last:           move       #puffer,d1
                move       worte,d0

;Bis zur Wortposition Sektor löschen
clearlast:      dec     sf d0
                clr pe     (d1)
                inc pe     d1
                jpe        clearlast

;letztes BAM-Wort erstellen
same:           add        #puffer,worte,d1      ;Position des letzten BAM-Wortes
                move       #$ffff,(d1)           ;letzten BAM-Sektor eintragen
```

```
belegt2:        dec     sf bit                  ;bit enthaelt die Anzahl der Bits im letzten
BAM-Wort
                asl pl     (d1)
                jpe        belegt2

;Rest des Sektor mit $ffff füllen, als belegt kennzeichnen
                sub        worte,#256,d0        ;Ende des Puffers
fill:           dec     sf d0
                inc  pe    d1
                move pe    #$ffff,(d1)
                jpe        fill

;Sektor abspeichern
                push       #256          ;256 Worte
                pushz                    ;Ab Position 0
                push       sektoren      ;Sektornummer lo
                pushz                    ;Sektornummer hi
                push       #puffer       ;Pufferadresse
                push       #2            ;Job 2: Schreiben
                driver     device
                add        #6,sp
                jmi        fehler

;Cache Flushen
                push       #6            ;Flush
                driver     device
                inc        sp
                jmi        fehler
                clr     sf ~-           ;Flags löschen

;Erfolgsmeldung ausgeben
                move       #meldung,d0   ;Meldung ausgeben
                jsr        stringout
                jmi        fehler

ende:           move       #lf,d0        ;Linefeed ausgeben
                jsr        stringout
                rts

fehler:         push       d0
                sys        #29
                inc        sp
                rts

;String an StandardOut ausgeben
stringout:      pushz
                push       d0
                pushz
                sys        #14           ;WRITE
                add        #3,sp
                rts


cmd:            dw "cmd="
CmdLine:        dw 0    ;Zeiger auf CmdLine
CmdPos:         dw 0    ;Enthält Position der CmdLine, an der etwas verändert wurde
sysbelegung:    dw 0    ;Enthält Anzahl der von BAM und Bootsektor belegten  Sektoren
sektoren:       dw 0    ;Anzahl der Sektoren der BAM
worte:          dw 0    ;Wortpositon des letzten BAM-Wortes im letzten BAM-Sektor
bit:            dw 0    ;Bitpositon des letzten BAM-Bits im letzten BAM-Wort
device:         dw 0    ;TReiberadresse
anzahl:         dw 0    ;Anzahl der zu löschenden BAM-Sektoren
lf:             dw 10,13,0
meldung:        dw 13,10,"High-Level-Format erzeugt.",0
warnung:        dw 10,13,"Alle Daten werden gelöscht! Sind Sie sicher? (j/n) ",0
data
puffer: 256+128         ;Puffer und Stackbereich
```

12.13.13. KILL

```
include lib
;-------------------------------------------------------------------------
;KILL
;-------------------------------------------------------------------------
;KILL entfernt eine Task aus dem System
kill:           push       #cmd
                sys        #10           ;CmdLine suchen
                inc        sp
                move       d0,CmdLine
                cmp        (d0),-        ;Falls CmdLine leer -> Fehler
                move eq    #111,d0
                jeq        .fehler
```

```
;String in Zahl konvertieren
.konvert:       cmp         (CmdLine),-
                jeq         .killtask
                move        (CmdLine),d1

                cmp         #$30,d1       ;Zahl zwischn 0..9?
                jmi         .zahlfehler
                cmp         d1,#$39
                sub pl      #$30,d1
                jpl         .zahl

                cmp         #$41,d1       ;Buchstabe zwischen A..F?
                jmi         .zahlfehler
                cmp         d1,#$46
                sub pl      #55,d1
                jpl         .zahl

                cmp         #$61,d1       ;Buchstabe zwischen a..f?
                jmi         .zahlfehler
                cmp         d1,#$66
                sub pl      #87,d1
                jpl         .zahl

.zahlfehler:    move        #111,d0       ;Parameterfehler
.fehler:        push        d0
                sys         #29           ;Fehlerausgabe aufrufen
                inc         sp
.ende:          rts

;Tasknummer zusammenbauen
.zahl:          asl         nummer
                asl         nummer
                asl         nummer
                asl         nummer
                add         d1,nummer
                inc         CmdLine
                jmp         .konvert

;Task entfernen
.killtask:      push        nummer
                sys         #6            ;KILLTASK
                inc         sp
                jmi         .fehler

;Erfolgsmeldung ausgeben
                pushz
                push        #meldung
                pushz
                sys         #14
                add         #3,sp
                jmi         .fehler
                rts

cmd:            dw "cmd="
CmdLine:        dw 0              ;Zeiger auf CmdLine
nummer:         dw 0              ;Tasknummer
meldung:        dw 13,10,"Task entfernt!",13,10,0
data
stack: 64                         ;Stackbereich
```

12.13.14. MD

```
include lib
;------------------------------------------------------------------------------
;MD
;------------------------------------------------------------------------------
;M(ake) D(irectory) erstellt ein neues Verzeichnis.
md:             push        #cmd
                sys         #10           ;CmdLine suchen
                inc         sp

;Make directory aufrufen
                push        d0
                sys         #22           ;MAKE DIRECTORY
                inc         sp
                jmi         .fehler

;Erfolgsmeldung ausgeben
                pushz
                push        #meldung
                pushz
```

```
                sys         #14           ;WRITE
                add         #3,sp
                jpl         .ende

.fehler:        push        d0
                sys         #29           ;Fehlerausgabe
                inc         sp
.ende:          rts

cmd:            dw "cmd="
meldung:        dw 10,13,"Verzeichnis erstellt.",13,10,0
data
stack: 128                                ;Stackbereich
```

12.13.15. MEM

```
include lib
;------------------------------------------------------------------------------
;MEMFREE
;------------------------------------------------------------------------------
;MEM(ory) FREE ermittelt den noch zur Verfügung stehenden Arbeitsspeicher.
memfree:        sys         #3            ;MEMINFO anfordern
                move        d0,d5
                move        d1,d4

;Zahlen in String eintragen
                move        d4,d2
                move        #freechar,d7
                jsr         .zahlout

                move        d5,d2
                move        #bigchar,d7
                jsr         .zahlout

;String an StandardOut ausgeben
                pushz
                push        #freispeicher
                pushz
                sys         #14           ;WRITE
                add         #3,sp
                rts

;Zahl-in-String-Routine (Zahl in d2 an Adresse d7 schreiben)
.zahlout:       move        #4,d3
.zahlloop:      rol         d2
                rol         d2
                rol         d2
                rol         d2
                and         #15,d2,d0
                cmp         #10,d0
                add mi      #$30,d0,(d7)
                add pl      #55,d0,(d7)
                inc         d7
                dec      sf d3
                jne         .zahlloop
                rts

freispeicher:   dw "Freier Speicher: $"
freechar:       dw "xxxx Worte",13,10,"  Größter Block: $"
bigchar:        dw "xxxx Worte",13,10,0

data
stack:  64              ;Stackbereich
```

12.13.16. RD

```
include lib
;------------------------------------------------------------------------------
;RD
;------------------------------------------------------------------------------
;R(emove) D(irectory) entfernt ein Verzeichnis. Das Verzeichnis muß dazu leer
;sein.
rd:             push        #cmd
                sys         #10           ;CmdLine suchen
                inc         sp

;Remove Directory
                push        d0
                sys         #23           ;REMOVE DIRECTORY
                inc         sp
                jmi         .fehler
```

```
;Erfolgsmeldung ausgeben
                pushz
                push        #meldung
                pushz
                sys         #14           ;WRITE
                add         #3,sp
                jpl         .ende

.fehler:push       d0
                sys         #29           ;Fehlerausgabe
                inc         sp
.ende:          rts

cmd:            dw "cmd="
meldung:        dw 13,10,"Verzeichnis entfernt!",13,10,0
data
stack: 128                                ;Stackbereich
```

12.13.17. REN

```
include lib
;------------------------------------------------------------------------------
;REN
;------------------------------------------------------------------------------
;REN(ame) ändert den Namen einer Datei oder eines Verzeichnisses.
ren:            push        #cwd          ;Cwd suchen
                sys         #10
                inc         sp
                move mi     #111,d0
                jmi         .fehler
                move        d0,cwd_zeiger

;CmdLine suchen
                push        #cmd
                sys         #10           ;SEARCH_ENV
                inc         sp
                move        d0,CmdLine
                cmp         (CmdLine),-   ;Ist kein Parameter angegeben?
                move eq     #111,d0       ;Nein, Fehler
                jeq         .fehler

;Zeiger einstellen
                move        #puffer,Pfad
                add         #256,#puffer,lesepuffer

;Pfad des Verzeichnisses bestimmen
                cmp         (CmdLine),#$2f                ;Ist Pfad mit angegeben?
                jeq         .explizit
.cwd:           cmp         (cwd_zeiger),-
                move ne     (cwd_zeiger),(pfad)   ;Cwd kopieren
                inc  ne     cwd_zeiger
                inc  ne     pfad
                jne         .cwd

.explizit:      cmp         (CmdLine),-           ;Kein neuer Name angegeben?
                move eq     #111,d0
                jeq         .fehler
                cmp         (CmdLine),#32         ;" "=Abbruch, an gegebenen Pfad kopieren
                move        (CmdLine),(pfad)
                inc  ne     CmdLine
                inc  ne     pfad
                jne         .explizit
                clr         (pfad)                ;Space löschen
                clr         (CmdLine)
                move        CmdLine,CmdPos                ;Position für restauration der
CmdLine merken

;Suchstring isolieren
                move        CmdLine,such_strg
.cut:           cmp         (pfad),#$2f           ;letzten Teil löschen
                clr         (pfad)
                dec ne      pfad                  ;pfad:letzte position der Quelle
                dec ne      such_strg
                jne         .cut
                inc         such_strg

;Verzeichnis zum Lesen öffnen
                push        -                     ;zum lesen+schreiben öffnen
                push        #puffer               ;Dateipfad
                sys         #12                   ;OPEN
                add         #2,sp,=
                jmi         .restauriert
```

```
                move        d0,Quell_fdb          ;Quell_fdb:FDB-Adresse

;Suchstring vorbereiten
                push        Quell_fdb
                push        lesepuffer            ;Pufferadresse
                push        such_strg             ;Zeiger auf Namen
                sys         #26                   ;PREPARE_SEARCH
                add         #3,sp,=
.restauriert:   move        #32,(CmdPos)          ;CmdLine Restaurieren
                jmi         .fehler

;An VerzeichnisPfad an "/" anhängen
                move        #$2f,(pfad)           ;"/" anfügen
                inc         pfad
                inc         CmdLine

                add         #8,lesepuffer,datei_info      ;datei_info:Lesepuffer
.searchread:    add         #16,lesepuffer,Dateiname      ;Dateiname:Zeiger auf Dateinamen

;Abbruchcode (CTRL-C) abfragen
                push        -             ;Ein Zeichen von StIn lesen
                dec         sp
                pushz
                sys         #13           ;READ
                add         #3,sp
                and     sf #$ff,d0        ;Nur ASCII-Code interessiert
                cmp         d0,#3;CTRL-C?
                jeq         .abbruch      ;Ja Abbruch

;Nächsten passenden Eintrag suchen
                push        lesepuffer    ;Zeiger auf Suchstring
                push        Quell_fdb     ;FDB-Adresse
                sys         #27           ;SEARCH_NEXT
                add         #2,sp,=
                jmi         .ende

;Verzeichniseintrag laden
                push        #16                   ;16 Wörter
                push        datei_info            ;Lesepuffer
                push        Quell_fdb             ;FDB-Adresse
                sys         #13                   ;READ
                add         #3,sp,=

;gefundenen Namen an Pfad anhängen und löschen
                move        #8,count              ;Zähler
                move        pfad,d0
.copy:          bswp        (Dateiname),-,(d0)
                inc         d0
                and         #$ff,(Dateiname),(d0)
                inc         d0
                inc         Dateiname
                dec     sf  count
                jne         .copy
                clr         (d0)                  ;Stringendezeichen anfügen

;alten Dateinamen ausgeben
                move        #puffer,d0
                jsr         .stringout
                jmi         .fehler

;Datei zum Lesen öffnen
                push        -
                push        #puffer
                sys         #12                   ;OPEN
                add         #2,sp,=
                jpl         .dateiok
                cmp         d0,#114               ;Writeprotect?
                jne         .fehler               ;Nein anderer Fehler
                move        #wp_fehler,d0         ;Writeprotectfehler ausgeben
                jsr         .stringout
                jmi         .fehler
                jmp         .searchread

.dateiok:       move        d0,datei_fdb

;Neuen Namen erzeugen
                move        pfad,d0
                move        CmdLine,d1
                move        -,only_one
                move        #16,count
.build:         cmp         (d1),#$2a             ;Bei * Abbruch
                clr eq      only_one              ;ES kommen Wildcards vor!, weitere
```

```
umbenennungen möglich
                jeq         .weiter
                cmp         (d1),#$3f
                clr eq      only_one
                move ne     (d1),(d0)
                cmp         (d1),-                ;Ende?
                inc ne      d1
                inc         d0
                dec      sf count
                jne         .build

;Datei zum Lesen öffnen um zu testen, ob der Dateiname schon existiert
.weiter:push       -
                push        #puffer
                sys         #12                   ;OPEN
                add         #2,sp
                jpl         .exist_schon
                cmp         d0,#109               ;Existiert-nicht-Fehler?
                jeq         .rename               ;Ja, Datei umbennenn
                jmp         .fehler

;Datei exitiert schon, Fehlermeldung ausgeben und Datei wieder schließen
.exist_schon:   push        d0
                sys         #15                   ;CLOSE
                inc         sp

;Linefeed ausgeben
                move        #lf,d0
                jsr         .stringout
                jmi         .fehler

;Neuen Dateinamen ausgeben
                move        #puffer,d0
                jsr         .stringout
                jmi         .fehler

;Fehlermeldung ausgeben
                move        #exist,d0
                jsr         .stringout
                jmi         .fehler
                jmp         .close

;Alles ok, Datei umbenennen
.rename:        push        datei_fdb
                push        pfad
                sys         #20                   ;RENAME
                add         #2,sp
                jmi         .fehler

;String ausgeben
                move        #meldung,d0
                jsr         .stringout
                jmi         .fehler

;Neuen Dateinamen ausgeben
                move        #puffer,d0
                jsr         .stringout
                jmi         .fehler

;Linefeed ausgeben
                move        #lf,d0
                jsr         .stringout
                jmi         .fehler

;Geöffnete Datei wieder schließen
.close:         push        datei_fdb
                sys         #15                   ;Gefundene Datei wieder schließen
                inc         sp
                jmi         .fehler

                cmp         only_one,-            ;Darf noch gesucht werden?
                jeq         .searchread           ;Nächste Datei suchen
                jmp         .ende

;String an StandardOut ausgeben
.stringout:     pushz
                push        d0
                pushz
                sys         #14                   ;WRITE
                add         #3,sp
                rts
```

```
;Abbruchmeldung ausgeben
.abbruch:       move       #break,d0
                jsr        .stringout
                jmp        .ende

.fehler:        push       d0
                sys        #29                  ;Fehlerausgabe
                push       datei_fdb
                sys        #15                  ;CLOSE
                add        #2,sp

.ende:          push       Quell_fdb
                sys        #15                  ;Verzeichnis schließen
                inc        sp
                rts

cmd:            dw "cmd="
cwd:            dw "cwd="
cwd_zeiger:     dw 0    ;Zeiger auf CWD
CmdLine:        dw 0    ;Zeiger auf CmdLine
pfad:           dw 0    ;Zeiger auf Dateipfad
such_strg:      dw 0    ;Zeiger auf Suchstring
CmdPos:         dw 0    ;Zeiger auf Position in der CmdLine, an der etwas verändert wurde
lesepuffer:     dw 0    ;Zeiger auf Suchstringpuffer (8 Worte)
datei_info:     dw 0    ;Zeiger auf Verzeichniseintrag (16 Worte)
Dateiname:      dw 0    ;Zeiger auf Dateinamen im Verzeichniseintrag
quell_fdb:      dw 0    ;FDB-Adresse des Verzeichnisses
datei_fdb:      dw 0    ;FDB-Adresse der umzubennenden Datei
count:          dw 0    ;Zähler
only_one:       dw 0    ;Flag, falls mehere Dateien den gleichen Namen bekommen sollen
lf:             dw 13,10,0
meldung:        dw " umbenannt in ",0
wp_fehler:      dw " ist schreibgeschützt!",13,10,0
exist:          dw " existiert schon!",13,10,0
break:          dw 13,10,"Befehlsunterbrechung.",13,10,0
data
puffer: 280+128         ;Puffer und Stackbereich
```

12.13.18. SET

```
include lib
;------------------------------------------------------------------------------
;SET
;------------------------------------------------------------------------------
;SET zeigt den Environmentbereich der aktuellen Task an, setzt und löscht
;Environmentvariablen.
set:            push       #cmd
                sys        #10          ;CmdLine suchen
                inc        sp
                move       d0,d1

;SET-Aufgabe(Anzeigen, Setzen, Löschen) auswerten
loop:           cmp        (d0),-
                jeq        show
                cmp        (d0),#$3d    ;"=" ,Abbruch
                inc        d0
                jne        loop

                cmp        (d0),-       ;Löschen oder Setzen?
                jeq        clr

;Env.-Variable zum Setzen in Puffer kopieren
                move       #puffer,d0
copy:           move    sf (d1),(d0)
                inc        d0
                inc        d1
                jne        copy

;Environment-Variable setzen
                push       #puffer      ;Setzen
                sys        #9           ;SET_ENV
                inc        sp
                jmi        fehler
                rts

;Env.-Bereich anzeigen
show:           add        #22,akt_task,d2      ;Länge des Env.-Bereiches bestimmen
                sub        #2,akt_task,d3
                add        akt_task,(d3),d3
                sub        #3,d3

loop2:          cmp        d2,d3
```

```
                rts eq
                cmp         (d2),-
                inc eq      d2
                cmp         (d2),-
                rts eq

;Env.-Var. anzeigen
                move        d2,d0
                jsr         stringout
                jmi         fehler

;Linefeed ausgeben
                move        #lf,d0
                jsr         stringout
                jmi         fehler

go_on:          inc         d2
                cmp         (d2),-
                jne         go_on
                jmp         loop2

;Environment-Variable löschen
clr:            push        d1
                sys         #11          ;CLR_ENV
                inc         sp
                rts pl

fehler:         push        d0
                sys         #29          ;Fehlerausgabe
                inc         sp
                rts

;String an StandardOut ausgeben
stringout:      pushz
                push        d0
                pushz
                sys         #14          ;WRITE
                add         #3,sp
                rts

cmd:            dw "cmd="
lf:             dw 10,13,0
data
puffer: 256+64                           ;Puffer und Stackbereich
```

12.13.19. SHELL

```
include lib
;------------------------------------------------------------------------------
;SHELL
;------------------------------------------------------------------------------
;SHELL startet eine neue Shell.
shell:          sys         #30          ;Sprung zur Shell
                rts
data
stack: 128                               ;Stack der neuen Shell
```

12.13.20. TASKLST

```
include lib
;------------------------------------------------------------------------------
;TASKLST
;------------------------------------------------------------------------------
;TASKL(i)ST zeigt alle im Tasks im System mit ihren Devices an.
tasklst:move       17,d5;Adr. 17=akt_task

;Headline an StandardOut
                move        #start,d0
                jsr         .stringout
                jmi         .fehler

;Tasks ausgeben
.loop:          add         #12,d5,d2    ;d2: Zeiger auf Taskname
                move        #8,d4        ;Acht Zeichen Taskname ausgeben

;Taskname zeichenweise ausgeben
.nameloop:      move    sf (d2),d0
                move eq     #32,d0
                jsr         .out
                jmi         .fehler
                inc         d2
                dec     sf d4
```

```
                jne         .nameloop

;3 Spaces ausgeben
                move        #space3,d0
                jsr         .stringout
                jmi         .fehler

;Tasknummer ausgeben
                sub         -,d5,d3
                move        (d3),d3
                move        #4,d4
.zahlloop:      rol         d3
                rol         d3
                rol         d3
                rol         d3
                and         #$f,d3,d0
                cmp         #10,d0
                add mi      #$30,d0
                add pl      #55,d0
                jsr         .out          ;Tasknummer ausgeben
                jmi         .fehler

                dec      sf d4
                jne         .zahlloop

;Spaces ausgeben
                move        #space,d0
                jsr         .stringout
                jmi         .fehler

;StandardIn bestimmen und ausgeben
                add         #20,d5,d2
                move        (d2),d2
                cmp         d2,-          ;Null-Device
                add         #2,d2         ;Zeiger auf Treiberadresse
                sub         #5,(d2),d2    ;Zeiger auf Namen im Treiber
                move eq     #nul,d2
                move        #4,d4

;Devicenamen ausgeben
.stinloop:      move     sf (d2),d0
                move eq     #32,d0
                jsr         .out
                jmi         .fehler
                inc         d2
                dec      sf d4
                jne         .stinloop

;Spaces ausgeben
                move        #space,d0
                jsr         .stringout
                jmi         .fehler

;StandardOut bestimmen und ausgeben
.stout:         add         #21,d5,d2
                move        (d2),d2
                cmp         d2,-
                add         #2,d2
                sub         #5,(d2),d2
                move eq     #nul,d2
                move        #4,d4

;Devicenamen ausgeben
.stoutloop:     move     sf (d2),d0
                move eq     #32,d0
                jsr         .out
                jmi         .fehler
                inc         d2
                dec      sf d4
                jne         .stoutloop

;Linefeed an StandardOut
                move        #lf,d0
                jsr         .stringout
                jmi         .fehler

;Auf nächste Task weiterschalten
                add         #11,d5
                move        (d5),d5
                cmp         d5,17
                jne         .loop
                rts
```

```
;Zeichen ausgeben
.out:           push       -
                push       d0
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                rts

;String ausgeben
.stringout:     pushz
                push       d0
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                rts

.fehler:        push       d0
                sys        #29          ;Fehlerausgabe
                inc        sp
.ende:          rts

lf:             dw 10,13,0
start:          dw 10,13,"Taskname   Handle   StdIn-Device StdOut-Device",13,10,
                dw "-----------------------------------------------",13,10,0
nul:            dw "Null",0
space:          dw "          ",0
space3:         dw "   ",0
data
stack: 64                                ;Stackbereich
```

12.13.21. TYPE

```
include lib
;------------------------------------------------------------------------------
;TYPE
;------------------------------------------------------------------------------
;TYPE zeigt den Inhalt einer Datei an
type:           push       #cmd
                sys        #10          ;CmdLine suchen
                inc        sp
                cmp        (d0),-
                move eq    #111,d0
                jeq        .fehler

;Datei zum Lesen öffnen
                push       -
                push       d0           ;Zeiger auf Dateinamen
                sys        #12          ;OPEN
                add        #2,sp
                jmi        .fehler
                move       d0,quell_fdb

;Abbruchkommando  (CTRL-C) überprüfen
.loop:          push       -            ;EinZeichen von StdIn lesen
                dec        sp
                pushz
                sys        #13          ;READ
                add        #3,sp
                and     sf #$ff,d0      ;Nur ASCII-Code interessiert
                cmp        d0,#3        ;CTRL-C =Abbruch
                jeq        .abbruch

;16 Worte einlesen
                push       #16          ;16 Zeichen lesen
                push       #puffer
                push       quell_fdb
                sys        #13          ;READ
                add        #3,sp
                jpl        .ok
                cmp        d0,#118      ;Kein Zeichen vorrätig?
                jne        .eof_check   ;Anderer Fehler!
                cmp        d1,-         ;Kein  Zeichen gelesen?
                jeq        .loop;Nein
                jmp        .ok          ;Doch!
.eof_check:     cmp        d0,#105      ;EOF
                jne        .fehler      ;Anderer Fehler!
                cmp        d1,-         ;Kein Zeichen gelesen?
                jeq        .eof         ;Nein, Ende
                clr        eof

;Diese 16 Worte ausgeben
```

```
.ok:            push       d1           ;Zeichen an StandardOut ausgeben
                cmp        -,d1         ;Eins?
                push ne    #puffer      ;Nein Adresse übergeben
                push eq    puffer       ;Ja, Zeichen übergeben
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                jmi        .fehler
                cmp        eof,-        ;Ist EOF erreicht?
                jne        .loop        ;nein, nächsten Block lesen
                jmp        .eof

.fehler:        push       d0           ;sonst...
                sys        #29          ;Fehlerausgabe aufrufen
                inc        sp

.eof:           push       quell_fdb    ;Datei wieder schließen
                sys        #15          ;CLOSE
                inc        sp
                rts

;Abruchmeldung ausgeben
.abbruch:       pushz                   ;String ausgeben
                push       #break
                pushz
                sys        #14          ;WRITE
                add        #3,sp
                jmp        .eof

cmd:            dw "cmd="
eof:            dw 1                    ;EOF-Flag
break:          dw 13,10,"Befehlsunterbrechung.",13,10,0

data
quell_fdb: 1            ;FDB-Adresse der Datei
puffer: 16+128          ;Puffer und Stackbereich
```

12.13.22. TYPEB

```
include lib
;------------------------------------------------------------------------------
;TYPEB
;Funktion entspricht dem Dienstprogramm TYPE.
;Unterschied: Es werden jeweils das höherwertige und das niederwertige Byte
;eines Wortes ausgegeben, anstatt nur des höherwertigen.
;------------------------------------------------------------------------------
;TYPE zeigt den Inhalt einer Datei an
type:           push       #cmd
                sys        #10          ;CmdLine suchen
                inc        sp
                cmp        (d0),-
                move eq    #111,d0
                jeq        .fehler

;Datei zum Lesen öffnen
                push       -
                push       d0           ;Zeiger auf Dateinamen
                sys        #12          ;OPEN
                add        #2,sp
                jmi        .fehler
                move       d0,quell_fdb

;Abbruchkommando  (CTRL-C) überprüfen
.loop:          push       -            ;EinZeichen von StdIn lesen
                dec        sp
                pushz
                sys        #13          ;READ
                add        #3,sp
                and     sf #$ff,d0      ;Nur ASCII-Code interessiert
                cmp        d0,#3;CTRL-C =Abbruch
                jeq        .abbruch

;1 Wort einlesen
                push       -            ;1 Wort lesen
                dec        sp
                push       quell_fdb
                sys        #13          ;READ
                add        #3,sp
                jpl        .ok

                cmp        d0,#118      ;Kein Zeichen vorrätig?
                jeq        .loop
```

```
                cmp         d0,#105       ;EOF
                jeq         .eof          ;Nein, Ende

.fehler:push       d0            ;sonst...
                sys         #29           ;Fehlerausgabe aufrufen
                inc         sp

.eof:           push        quell_fdb     ;Datei wieder schließen
                sys         #15           ;CLOSE
                inc         sp
                rts

;Diese 2 Zeichen ausgeben
.ok:            push        -             ;Zeichen an StandardOut ausgeben
                move        d0,d7
                bswp        d0,-,d0
                push        d0            ;Ja, Zeichen übergeben
                pushz
                sys         #14           ;WRITE
                add         #3,sp
                jmi         .fehler
                push        -             ;Zeichen an StandardOut ausgeben
                push        d7            ;Ja, Zeichen übergeben
                pushz
                sys         #14           ;WRITE
                add         #3,sp
                jmi         .fehler
                jmp         .loop;nein, nächsten Block lesen


;Abruchmeldung ausgeben
.abbruch:       pushz                     ;String ausgeben
                push        #break
                pushz
                sys         #14           ;WRITE
                add         #3,sp
                jmp         .eof

cmd:            dw "cmd="
eof:            dw 1                      ;EOF-Flag
break:          dw 13,10,"Befehlsunterbrechung.",13,10,0

data
quell_fdb:      1               ;FDB-Adresse der Datei
stack:          128             ;Stackbereich
```

12.14. LIB.ASM

```
;=============================================================================
;LIBRARY
;=============================================================================
;Vereinbarungen
;.............................................................................
equ d0,8;8 Arbeitsregister für aktuelle Task
equ d1,9
equ d2,10
equ d3,11
equ d4,12
equ d5,13
equ d6,14
equ d7,15
equ sp,16       ;Stackpointer für aktuelle Task

equ akt_task,17
equ timerh,28
equ timerl,29

;Befehlsmakros
;Sprungbefehle
;.............................................................................
macro jmp 1
  add tr hf #%1,-,=
endm

macro bra 1
  add %c hf %1,-,=
endm

macro jne 1
  add ne hf #%1,-,=
endm
```

```
macro jeq 1
  add eq hf #%1,-,=
endm

macro jpl 1
  add pl hf #%1,-,=
endm

macro jmi 1
  add mi hf #%1,-,=
endm

macro jpe 1
  add pe hf #%1,-,=
endm

macro jcs 1
  add cs hf #%1,-,=
endm

macro jcc 1
  add cc hf #%1,-,=
endm

;Nützliche Befehle
;.....................................................................
macro move 2
  or %c %f %1,-,%2
endm

macro inc 1
  add %c %f -,%1,=
endm

macro dec 1
  sub %c %f -,%1,=
endm

macro in 2
  add tr %f @%1,-,%2
endm

macro out 2
  add tr %f %1,-,@%2
endm

macro not 2
  nor %c %f %1,-,%2
endm

macro clr 1
  and %c %f -,-,%1
endm

macro cmp 2
  sub tr sf %1,%2,-
endm

;Unterprogrammaufrufe
;.....................................................................
macro jsr 1
     sub tr hf -,sp,=
     add tr hf #*+5,-,(sp)
     add %c hf #%1,-,=
     add tr hf -,sp,=
endm

macro driver 1
   sub tr hf -,sp,=
   add tr hf #*+5,-,(sp)
   add %c hf %1,-,=
   add tr hf -,sp,=
endm

macro sys 1
  sub tr hf -,sp,=
  add tr hf #*+10,-,(sp)
  sub tr hf -,sp,=
  add tr hf %1,-,(sp)
  add tr hf #$8002,-,=
  add tr hf -,sp,=
```

```
endm

macro rts 0
  add %c hf (sp),-,=
endm

;Stackoperationen
;.....................................................................
macro push 1
  sub %c hf -,sp,=
  add %c hf %1,-,(sp)
endm

macro pop 1
  or  %c hf (sp),-,%1
  add %c hf -,sp,=
endm

macro pushz 0
  sub %c hf -,sp,=
  and %c hf -,-,(sp)
endm

macro pushf 0
  sub %c hf -,sp,=
  add %c hf ~-,-,(sp)
endm

macro popf 0
  add %c sf (sp),-,~-
  add %c hf -,sp,=
endm

macro pushall 0
  sub %c hf -,sp,=
  add %c hf d2,-,(sp)
  sub %c hf -,sp,=
  add %c hf d3,-,(sp)
  sub %c hf -,sp,=
  add %c hf d4,-,(sp)
  sub %c hf -,sp,=
  add %c hf d5,-,(sp)
  sub %c hf -,sp,=
  add %c hf d6,-,(sp)
  sub %c hf -,sp,=
  add %c hf d7,-,(sp)
endm

macro popall 0
  or  %c hf (sp),-,d7
  add %c hf -,sp,=
  or  %c hf (sp),-,d6
  add %c hf -,sp,=
  or  %c hf (sp),-,d5
  add %c hf -,sp,=
  or  %c hf (sp),-,d4
  add %c hf -,sp,=
  or  %c hf (sp),-,d3
  add %c hf -,sp,=
  or  %c hf (sp),-,d2
  add %c hf -,sp,=
endm

;Interruptbefehle
;.....................................................................
macro irqoff 0
  and tr hf ~-,#$feff,~dummy
endm

macro irqon 0
  or tr hf ~-,#$0100,~dummy
endm
```

12.15. XDISK.PAS

```
PROGRAM Xdisk;

{.....................Declaration.........................}
USES Dos,Crt;

CONST anzahl=21;
      max=anzahl*256-1;
```

```
VAR data:     ARRAY[0..max] OF WORD;
    laufwerk: INTEGER;
    f:        FILE;
    s:        STRING; {Filename}
    io:       WORD;
    error:    BOOLEAN;

{......Verwaltungsinformation des Datenträgers erstellen.......}
PROCEDURE Format(vn:STRING);
VAR n:WORD;
BEGIN
    FOR n:=0 TO 18 DO data[n]:=0;
    data[1]:=2;
    data[3]:=16;
    data[4]:=0;
    data[5]:=1;
    data[6]:=$1111;
    data[7]:=$2222;
    data[16]:=$8765;
    data[17]:=1;
    data[18]:=1;
    data[19]:=0;
    WHILE Length(vn)<16 DO vn:=vn+chr(0);
    FOR n:=0 TO 7 DO data[n+8]:=(ord(vn[2*n+1]) shl 8) + ord(vn[2*n+2]);
    FOR n:=256 TO 511 DO data[n]:=$ffff;
    FOR n:=512 TO max DO data[n]:=0;
    data[513]:=3;
END;


{........Verzeichniseintrag des Programms in Directory erstellen.........}
PROCEDURE Insert;
VAR i,n:WORD;
BEGIN
          Assign(f,s+'.obj');
    WHILE Length(s)<16 DO s:=s+chr(0);
    FOR n:=0 TO 7 DO data[n+520]:=(ord(s[2*n+1]) shl 8) + ord(s[2*n+2]);
    Reset(f,2);
    io:=IOResult;
    IF io=0 THEN BEGIN
     data[514]:=Filesize(f) div 256;
     data[515]:=Filesize(f) and 255;
     data[516]:=2;
     data[517]:=2;
     FOR n:=1 TO 16 DO data[767+2*n]:=n+3;
     FOR n:=0 TO 16 DO
         FOR i:=1 TO 128 DO data[1023+(256*n)+(2*i)]:=19+n*128+i;
    END ELSE BEGIN
        writeln(' Dateifehler! ');
        writeln(' Dateinamen überprüfen und Programm XDISK neustarten!');
    END
END;


{.......Laufwerk und Dateinamen eingeben........}
PROCEDURE Entry;
VAR c:CHAR;
BEGIN
    ClrScr;
    Writeln;
    Writeln;
    Writeln;
    Write(' Welche Kennung hat das Diskettenlaufwerk (3,5")? (A/B)');
    Readln(c);
    IF ((c='a') or (c='A')) THEN laufwerk:=0 ELSE laufwerk:=1;
    Writeln;
    Writeln;
    Write(' Dateinamen ohne Endung eingeben: ');
    Readln(s);
    Writeln;
END;


{.........Verwaltungsinformation und Datei auf Diskette ablegen.........}
PROCEDURE Save;
VAR n,j,cyl,sector,head:   INTEGER;
    i:                     LONGINT;
    r:                     REGISTERS;
    tmp:                   WORD;
    errorcount:            WORD;
```

```
BEGIN
    errorcount:=0;
    error:=FALSE;
    REPEAT
     FOR i:=0 TO max DO data[i]:=(data[i] shl 8) or (data[i] shr 8);
     cyl:=0;
     head:=0;
     sector:=1;
     FOR i:=0 TO 19 DO BEGIN
        r.ah:=3;
        r.al:=1;
        r.ch:=cyl;
        r.cl:=sector;
        r.dh:=head;
        r.dl:=laufwerk;
        r.es:=Seg(data[256*i]);
        r.bx:=Ofs(data[256*i]);
        Intr($13,r);
       inc(sector);
       IF sector=19 THEN BEGIN
          sector:=1;
          Inc(Head); IF Head=2 THEN BEGIN
        Head:=0;
                Inc(Cyl);
               END;
       END;
     END;
     IF r.ah=0 THEN BEGIN
        FOR i:=0 TO ((FileSize(f) div 256)) DO BEGIN
            IF ((FileSize(f)-(i*256))< 255) THEN j:=((FileSize(f)-i*256)-1)
        ELSE j:=255;
            FOR n:=0 TO j DO BEGIN
        BlockRead(f,tmp,1);
               data[n]:=tmp;
            END;
            r.ah:=3;
              r.al:=1;
              r.ch:=cyl;
              r.cl:=sector;
              r.dh:=head;
              r.dl:=laufwerk;
              r.es:=Seg(data[0]);
              r.bx:=Ofs(data[0]);
              Intr($13,r);
            inc(sector);
            IF sector=19 THEN BEGIN
          sector:=1;
          Inc(Head); IF Head=2 THEN BEGIN
                        Head:=0;
                        Inc(Cyl);
                      END;
            END;
             END;
     END;
     close(f);
     IF r.ah<>0 then inc(errorcount,1);
    UNTIL ((r.ah=0) or (errorcount=4));
    IF r.ah<>0 THEN error:=TRUE;
END;


{.........Hauptprogramm..........}
BEGIN
    Format('XDisk!');
    Entry;
    insert;
    IF io=0 THEN Save;
    IF error=TRUE THEN BEGIN
                  writeln;
                  writeln('Diskettenfehler aufgetreten!');
    END;
END.
```

12.16. XSIM.C

```
#include <stdio.h>
#include <conio.h>
#include <mem.h>

unsigned int huge memory[65536],huge user_screen[25][80];
unsigned int status,irqlevel,adress[26],timeronoff=0;
```

```
int simstop,display_radix=0,display_user=0;
char befehl[16][6]={     "add  ","nor  ","ror  ","rol  ",
                         "sub  ","and  ","rorc ","rolc ",
                         "addc ","eor  ","asr  ","asl  ",
                         "subc ","or   ","lsr  ","bswp " };

char condition[16][3]={"me","pe","hi","ls",
                         "cc","cs","ne","eq",
                         "vc","vs","pl","mi",
                         "ge","lt","gt","le" };

void debug(unsigned int disass,unsigned int dump,int mark)
{
        unsigned int x,y,bef;
        char *format;

        switch(display_radix) {
                case 0:format="#$%04X,"; break;
                case 1:format="#%u,"; break;
        }

        irqlevel=0;
        if((status&0x0200)==0) irqlevel=1;
        if((status&0x0400)==0) irqlevel=2;
        if((status&0x0800)==0) irqlevel=3;
        if((status&0x1000)==0) irqlevel=4;
        if((status&0x2000)==0) irqlevel=5;

        for(y=1;y<=25;y++) {
                gotoxy(1,y);
                adress[y]=disass;
                if(y==mark) textattr(WHITE|BLUE<<4); else textattr(LIGHTGRAY|BLACK<<4);
                if(memory[irqlevel]==disass) cprintf("$%04X:\x10",disass);
                                        else cprintf("$%04X: ",disass);

                bef=memory[disass++];
                cprintf(befehl[bef&15]);
                if(bef&0x0800) cprintf("  "); else cprintf(condition[bef>>12]);
                if(bef&1024) cprintf(" sf "); else cprintf("    ");

                if((bef&0x8800)==0x8800) putch('&');
                if((bef&0x4800)==0x4800) putch('@');
                switch((bef>>4)&3) {
                        case 0:cprintf("-,"); break;
                        case 1:cprintf(format,memory[disass++]); break;
                        case 2:cprintf("%04X,",memory[disass++]); break;
                        case 3:cprintf("(%04X),",memory[disass++]);
                }

                switch((bef>>6)&3) {
                        case 0:cprintf("-,"); break;
                        case 1:cprintf(format,memory[disass++]); break;
                        case 2:cprintf("%04X,",memory[disass++]); break;
                        case 3:cprintf("(%04X),",memory[disass++]);
                }

                if((bef&0x2800)==0x2800) putch('&');
                if((bef&0x1800)==0x1800) putch('@');
                switch((bef>>8)&3) {
                        case 0:putch('-'); break;
                        case 1:putch('='); break;
                        case 2:cprintf("%04X",memory[disass++]); break;
                        case 3:cprintf("(%04X)",memory[disass++]);
                }

                while(wherex()<43) putch(' ');
        }

        textbackground(BLACK);
        textcolor(LIGHTRED);
        gotoxy(43,1); cprintf("     Carry:%d",status&1);
        gotoxy(43,2); cprintf("      Zero:%d",(status>>1)&1);
        gotoxy(43,3); cprintf("  Negative:%d",(status>>2)&1);
        gotoxy(43,4); cprintf("  Overflow:%d",(status>>3)&1);
        textcolor(RED);
        gotoxy(43,5); cprintf("IRQ-Enable:%d",(status>>8)&1);
        gotoxy(43,6); cprintf("     ~IRQ1:%d",(status>>9)&1);
        gotoxy(43,7); cprintf("     ~IRQ2:%d",(status>>10)&1);
        gotoxy(43,8); cprintf("     ~IRQ3:%d",(status>>11)&1);
        gotoxy(43,9); cprintf("     ~IRQ4:%d",(status>>12)&1);
        gotoxy(43,10);cprintf("     ~IRQ5:%d",(status>>13)&1);
        gotoxy(43,11);cprintf(" IRQ-Level:%d",irqlevel);
```

```
        textcolor(LIGHTBLUE);
        gotoxy(60,1); cprintf("D0:$%04X",memory[8]);
        gotoxy(60,2); cprintf("D1:$%04X",memory[9]);
        gotoxy(60,3); cprintf("D2:$%04X",memory[10]);
        gotoxy(60,4); cprintf("D3:$%04X",memory[11]);
        gotoxy(60,5); cprintf("D4:$%04X",memory[12]);
        gotoxy(60,6); cprintf("D5:$%04X",memory[13]);
        gotoxy(60,7); cprintf("D6:$%04X",memory[14]);
        gotoxy(60,8); cprintf("D7:$%04X",memory[15]);

        x=memory[16];
        textcolor(YELLOW);
        for(y=0;y<16;y++) {
                gotoxy(73,y+1);
                cprintf("$%04X",memory[x+y]);
        }

        if(display_user) { puttext(1,26,80,50,user_screen); return;}

        textcolor(CYAN);
        for(y=26;y<=50;y++) {
                gotoxy(1,y);
                cprintf("$%04X: ",dump);
                for(x=0;x<8;x++) cprintf("%04X ",memory[dump+x]);
                for(x=0;x<8;x++) {
                        bef=memory[dump+x];
                        if((bef>>8)>31) putch(bef>>8); else putch('.');
                        if((bef&255)>31) putch(bef); else putch('.');
                }
                clreol();
                dump+=x;
        }
}


void load(char *filename,unsigned int adress)
{
        FILE *f;
        unsigned int tmp;
        long c;

        for(c=0;c<65536;c++) memory[c]=0xffff;

        gotoxy(1,1); textattr(WHITE|RED<<4);
        cprintf("Loading file '%s'\n",filename);
        if((f=fopen(filename,"rb"))==NULL) {
                cprintf("Error loading file.\nPress any key.\n");
                getch();
                return;
        }

        c=adress;
        while(!feof(f)) {
                fread(&tmp,2,1,f);
                memory[c++]=(tmp<<8)|(tmp>>8);
        }
        fclose(f);

        status=0xff00;
        irqlevel=0;
        memory[irqlevel]=adress;
}

unsigned int next(unsigned int adress)
{
        int bef,xop,yop,zop;
        bef=memory[adress];
        xop=(bef>>4)&3;
        yop=(bef>>6)&3;
        zop=(bef>>8)&3;
        return adress+1+(xop>0)+(yop>0)+(zop>1);
}

unsigned int prev(unsigned int adress)
{
        int bef,xop,yop,zop;
        unsigned int old_adress=adress;

        do {
                adress--;
                bef=memory[adress];
```

```
                xop=(bef>>4)&3;
                yop=(bef>>6)&3;
                zop=(bef>>8)&3;
        } while( (old_adress!=adress+1+(xop>0)+(yop>0)+(zop>1)) && (adress>old_adress-4));
        return adress;
}

void invert(int x,int y)
{
        user_screen[y][x]^=0x7f00;
        if(display_user) puttext(x+1,y+26,x+1,y+26,&user_screen[y][x]);
}

void scroll_down(void)
{
        int x,y;
        for(y=0;y<24;y++) for(x=0;x<80;x++) user_screen[y][x]=user_screen[y+1][x];
        for(x=0;x<80;x++) user_screen[24][x]=0;
        if(display_user) puttext(1,26,80,50,user_screen);
}

void clear_screen(void)
{
        int x,y;
        for(y=0;y<25;y++) for(x=0;x<80;x++) user_screen[y][x]=0;
        if(display_user) puttext(1,26,80,50,user_screen);
}

void clr_line(int x,int y)
{
        for(;x<80;x++) user_screen[y][x]=0;
        if(display_user) puttext(1,26,80,50,user_screen);
}

void charout(unsigned char c)
{
        static int vt52seq=0,cursor_x=0,cursor_y=0,save_x=0,save_y=0,fcolor=0,bcolor=0;
        int w;

        if(vt52seq) {
                if(vt52seq==2) { fcolor=((c&4)>>2)|(c&2)|((c&1)<<2); vt52seq=0; return; }
                if(vt52seq==3) { bcolor=((c&4)>>2)|(c&2)|((c&1)<<2); vt52seq=0; return; }
                if(c=='E') {clear_screen(); cursor_x=0; cursor_y=0; invert(0,0); }
                if(c=='H') {invert(cursor_x,cursor_y); cursor_x=0; cursor_y=0; invert(0,0);
                            vt52seq=0; return; }
                if(c=='b') {vt52seq=2; return; }
                if(c=='c') {vt52seq=3; return; }
                if(c=='j') {save_x=cursor_x; save_y=cursor_y; }
                if(c=='k') {invert(cursor_x,cursor_y); cursor_x=save_x; cursor_y=save_y;
                            invert(cursor_x,cursor_y); }
                if(c=='C') {invert(cursor_x,cursor_y);
                            if(cursor_x<79) cursor_x++; else {cursor_x=0; c='B'; }
                            invert(cursor_x,cursor_y); }
                if(c=='D') {invert(cursor_x,cursor_y);
                            if(cursor_x>0) cursor_x--; else {cursor_x=79; c='A'; }
                            invert(cursor_x,cursor_y); }
                if(c=='A') {invert(cursor_x,cursor_y);
                            if(cursor_y>0) cursor_y--;
                            invert(cursor_x,cursor_y); }
                if(c=='B') {invert(cursor_x,cursor_y);
                            if(cursor_y<24) cursor_y++; else scroll_down();
                            invert(cursor_x,cursor_y); }
                if(c=='K') {clr_line(cursor_x,cursor_y); invert(cursor_x,cursor_y); }
                vt52seq=0;
                return;
        }

        if(c==27) { vt52seq=1; return; }
        if(c==13) { invert(cursor_x,cursor_y); cursor_x=0; invert(cursor_x,cursor_y); return;}
        if(c==10) { invert(cursor_x,cursor_y);
                    if(cursor_y==24) scroll_down(); else cursor_y++;
                    invert(cursor_x,cursor_y); return; }
        if(c==0) return;
        if(c<32) return;

        w=c+(fcolor<<8)+(bcolor<<12);
        if(display_user) puttext(cursor_x+1,cursor_y+26,cursor_x+1,cursor_y+26,&w);
        user_screen[cursor_y][cursor_x]=w;
        cursor_x++;
        if(cursor_x==80) { cursor_x=0; if(cursor_y==24) scroll_down(); else cursor_y++; }
                          invert(cursor_x,cursor_y);
}
```

```
void simkon(void)
{
        unsigned int d0,d1,x;

        d0=memory[8];
        d1=memory[9];

        switch(d0) {
                case 4:
                case 1: if(kbhit()) x=getch(); else x=0xffff;
                                if(x==27) simstop=1;
                                if(x==0) switch(getch()) {
                                        case 75: x=97<<8; break;
                                        case 77: x=106<<8; break;
                                        case 83: x=100<<8; break;
                                }
                                memory[8]=x;
                                break;
                case 2:charout(d1); break;
                case 3:while(memory[d1]) charout(memory[d1++]); break;
        }
}

void simdisk(void)
{
        FILE *f;
        unsigned int sp,job,addr,pos,anz,tmp;
        unsigned long sn,fp;

        sp=memory[16]+1;
        job=memory[sp++];
        addr=memory[sp++];
        sn=memory[sp++]<<16;
        sn|=memory[sp++];
        pos=memory[sp++];
        anz=memory[sp];

        f=fopen("e:\\diplom\\romddata.obj","r+b");
        fseek(f,sn*512+pos*2,SEEK_SET);
        if(job==1) for(;anz;anz--) { fread(&tmp,2,1,f); memory[addr++]=(tmp<<8)|(tmp>>8); }
        if(job==2) for(;anz;anz--) { tmp=(memory[addr]<<8)|(memory[addr]>>8); addr++;
                                     fwrite(&tmp,2,1,f); }
        fclose(f);
        memory[8]=0;
        status&=0xff00;
}

void step(void)
{
        unsigned int z,c,v,n,bef,yaddr,xaddr,zaddr,xop,yop,zop;

        irqlevel=0;
        if((status&0x0200)==0) irqlevel=1;
        if((status&0x0400)==0) irqlevel=2;
        if((status&0x0800)==0) irqlevel=3;
        if((status&0x1000)==0) irqlevel=4;
        if((status&0x2000)==0) irqlevel=5;

        bef=memory[memory[irqlevel]++];
        c=status&1;
        z=(status>>1)&1;
        n=(status>>2)&1;
        v=(status>>3)&1;

        switch((bef>>4)&3) {
                case 0: xop=1; break;
                case 1: xop=memory[memory[irqlevel]++]; break;
                case 2: xop=memory[memory[memory[irqlevel]++]]; break;
                case 3: xop=memory[memory[memory[memory[irqlevel]++]]];
        }

/*      if((bef&0x4800)==0x4800) {
                switch((bef>>4)&3) {
                        case 0: xaddr=0; break;
                        case 1: xaddr=0; break;
                        case 2: xaddr=memory[memory[irqlevel]-1]; break;
                        case 3: xaddr=memory[memory[memory[irqlevel]-1]];
                }
        }
*/
        if((bef&0x8800)==0x8800) xop=(xop&0x80f0)|(status&0x7f0f);
```

```
        switch((bef>>6)&3) {
                case 0: yaddr=irqlevel; yop=0; break;
                case 1: yaddr=memory[irqlevel]++; yop=memory[yaddr]; break;
                case 2: yaddr=memory[memory[irqlevel]++]; yop=memory[yaddr]; break;
                case 3: yaddr=memory[memory[memory[irqlevel]++]]; yop=memory[yaddr];
        }

        switch((bef>>8)&3) {
                case 1: zaddr=yaddr; break;
                case 2: zaddr=memory[memory[irqlevel]++]; break;
                case 3: zaddr=memory[memory[memory[irqlevel]++]];
        }

        if((bef&0x800)==0) switch(bef>>12) {
                case  0: if(!(z|n))      return; break;
                case  1: if(z|n) return; break;
                case  2: if(c|z) return; break;
                case  3: if(!(c|z)) return; break;
                case  4: if(c) return; break;
                case  5: if(!c) return; break;
                case  6: if(z) return; break;
                case  7: if(!z) return; break;
                case  8: if(v) return; break;
                case  9: if(!v) return; break;
                case 10: if(n) return; break;
                case 11: if(!n) return; break;
                case 12: if(n^v) return; break;
                case 13: if(n==v) return; break;
                case 14: if((n^v)|z) return; break;
                case 15: if((n==v)&!z) return;
        }

        switch(bef&15) {
                case  0: zop=xop+yop; c=(zop<(long)xop+yop); break;
                case  4: zop=yop-xop; c=(zop>(long)yop-xop); break;
                case  8: zop=xop+yop+c; c=(zop<(long)xop+yop+c); break;
                case 12: zop=yop-xop-c; c=(zop>(long)yop-xop-c); break;
                case  1: zop=(xop|yop)^0xffff; c=0; break;
                case  5: zop=xop&yop; c=0; break;
                case  9: zop=xop^yop; c=0; break;
                case 13: zop=xop|yop; c=0; break;
                case  2: zop=(xop>>1)|(xop<<15); c=xop&1; break;
                case  6: zop=(xop>>1)|(c<<15); c=xop&1; break;
                case 10: zop=(xop>>1)|(xop&0x8000); c=xop&1; break;
                case 14: zop=(xop>>1); c=xop&1; break;
                case  3: zop=(xop<<1)|(xop>>15); c=xop>>15; break;
                case  7: zop=(xop<<1)|c; c=xop>>15; break;
                case 11: zop=(xop<<1); c=xop>>15; break;
                case 15: zop=(xop>>8)|(yop<<8); c=xop>>15;
        }

        if(bef&0x400) {
                n=zop>>15;
                z=(zop==0);
                status=(status&0x7f00)|(v<<3)|(n<<2)|(z<<1)|c;
        }

        if((bef&0x2800)==0x2800) {
                if(bef&0x400) status=(status&0xff00)|(zop&0x00ff);
                if(bef&0x300) status=(status&0x00ff)|(zop&0xff00);
        }

        if((bef>>8)&3) {
                if((bef&0x1800)==0x1800) {
                        switch(zaddr) {
                                case 0x4000: simkon(); break;
                                case 0x4001: simdisk(); break;
                                case 0x7ff3: timeronoff=zop; break;
                        }
                } else if(zaddr<0x8000) memory[zaddr]=zop;
        }
}

void runto(unsigned int a)
{
        int temp=display_user;
        long c;
        char ch;

        gotoxy(1,1); textattr(WHITE|BLINK|RED<<4); cprintf("Running...\n\r");
        display_user=1;
```

```
        puttext(1,26,80,50,user_screen);
        step();

        c=0; simstop=0;
        while(!simstop && memory[irqlevel]!=a) {
                step();
                if(kbhit()) {ch=getch(); ungetch(ch); if(ch==27) simstop=1; }
                c++;
                if(timeronoff) if(c>=300) if(status&0x0100) { status&=0xdfff; c=0; }
        }

        display_user=temp;
}

void main(void)
{
        int c,marker=1;
        unsigned int disass=0x3000,dump=0x0000;
        char filename[100]="e:\\diplom\\xmos.obj";

        textmode(C4350);
        _setcursortype(_NOCURSOR);
        directvideo=1;
        clrscr();

        load(filename,0x3000);
        do {
                debug(disass,dump,marker);
                while(kbhit()) getch();
                switch(c=getch()) {
                        case 0: switch(c=getch()) {
                                                case 115:dump--; break;
                                                case 116:dump++; break;
                                                case 141:dump-=8; break;
                                                case 145:dump+=8; break;
                                                case 132:dump-=200; break;
                                                case 118:dump+=200; break;
                                                case 75:disass--; break;
                                                case 77:disass++; break;
                                                case 80:if(marker<25) marker++;
                                                        else disass=next(disass);
                                                        break;
                                                case 72:if(marker>1) marker--;
                                                        else disass=prev(disass);
                                                        break;
                                                case 81:for(c=0;c<25;c++)
                                                                        disass=next(disass);
                                                        break;
                                                case 73:for(c=0;c<25;c++)
                                                                        disass=prev(disass);
                                                        break;
                                                case 65:step();
        for(marker=1;marker<26&&memory[irqlevel]!=adress[marker];marker++);
                                                        if(marker==26) {
                                                                disass=memory[irqlevel];
                                                                marker=1;
                                                        }
                                                        break;
                                                case 67:runto(0);
        for(marker=1;marker<26&&memory[irqlevel]!=adress[marker];marker++);
                                                        if(marker==26) {
                                                                disass=memory[irqlevel];
                                                                marker=1;
                                                        }
                                                        break;
                                                case 62:runto(adress[marker]); break;
                                                case 95:load("e:\\diplom\\xmos.obj",0x3000);
                                                        break;
                                                case 108:display_user^=1; break;
                                                default:gotoxy(70,1); printf("0+%d  ",c);
                                        }
                                        break;
                        case ' ':status&=0xdfff; break;
                        case 'r':display_radix=1-display_radix; break;
                        case 14:memory[irqlevel]=adress[marker]; break;
                        case 15:
for(marker=1;marker<26&&memory[irqlevel]!=adress[marker];marker++);
                                if(marker==26) { disass=memory[irqlevel]; marker=1; }
                                break;
                        default:if(c>='0'&&c<='9') dump=(c-'0')*4096; gotoxy(70,1);
                                printf("  %d  ",c);
                }
```

```
        } while(c!=27 && c!=3);
}
```

13. Anhang E: Bild-, Tabellen- und Quellenverzeichnis

13.1. Bildverzeichnis

13.2. Tabellenverzeichnis

13.3. Quellenverzeichnis

- [CT6-88] C'T-Magazin, 6/1988, Seite 148ff: Knöpfchen, Knöpfchen, PC-Tastaturen auf den Zahn gefühlt.
- [MAX92] Maxtor Corporation, OEM Technical Manual for the 5.25-Inch Hard Disk Drive XT-8000S/SH, Revision D, März 1992, San Jose, Kalifornien, Dokumentennummer 1015586
- [MESS92] Messmer, Hans-Peter: PC-Hardwarebuch, Bonn - München - Paris, Addison-Wesley, 1992, ISBN 3-89319-357-X.
- [HINA84] Hilf, Werner und Nausch, Anton: M68000-Familie, Teil 1: Grundlagen und Architektur, München, te-wi-Verlag, 1984, ISBN 3-921803-16-0
- [HUF93], Thortsen Hufschmidt: Programmierung eines Cross-Assemblers für einen vorgegebenen 16-Bit-Prozessor, Trimesterarbeit UniBwM - IT 27/93

13.4. Werzeuge

- MS-DOS 5.0 von Microsoft Corp.
- Turbo Pascal 5.0, 5.5 und 6.0 von Borland International
- Borland C 3.1 für DOS von Borland International
- MS-Windows 3.1 von Microsoft Corp.
- Word für Windows 2.0b von Microsoft Corp.
- Ami Pro 3.0 für Windows von Lotus Development Corp.
- WORKVIEW
- XILINX

14. Anhang F: Erklärung

Wir versichern, daß die Arbeit bzw. unser Anteil daran selbständig verfaßt wurde und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt wurden.

(Uwe Reinhardt) (Oliver Henning)