TA 1600/30 T A X O

TAXO

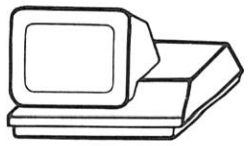
Triumph Adler EXtended Operating System

Eine einführende Beschreibung

Januar 1981

TA1630

Der dialog-orientierte Arbeitsplatz-Computer



T A X O

Triumph Adler EXtended Operating System

Eine
einführende
Beschreibung

Januar 1981

TRIUMPH ADLER

Inhaltsverzeichnis

1 EINLEITUNG

TAXO ist ein anwenderorientiertes, universelles Multitasking-Betriebssystem, das für eine große Spanne sowohl kommerzieller als auch industrieller Anwendungen geeignet ist. Es ist plattenorientiert, mit einer mächtigen Dateiverwaltung, welche indexsequentielle Dateien mit mehreren Schlüsseln (multikey indexed) einschließt. Zusätzlich werden extensive Mehrplatzanwendungen unterstützt. Daneben existieren fortschrittliche Hilfen zur Programmentwicklung sowohl im interaktiven Arbeitsmodus als auch bei Stapelverarbeitung. COBOL wird als höhere Programmiersprache angeboten. Ein umfassendes Sort/Merge Paket ist außerdem verfügbar.

Mehrplatz- und Stapelverarbeitung in COBOL in Verbindung mit einem kompletten Sort/Merge Paket weisen das TA 1630 System als ideal für viele kommerzielle Anwendungen aus.

Ein komplettes Angebot von Programmentwicklungshilfen, einschließlich interaktivem Text Editor, Makro Assembler, Link Editor und Testhilfen (Debug), verkürzen enorm die Entwicklungszeit und steigern die Effektivität der Programmierer. Programme können auf der TA 1630 entwickelt werden und auf den kleineren floppyorientierten oder speicherresidenten Mitgliedern der TA 1630 Systemfamilie ausgeführt werden.

3 PLATTENORGANISTAION

3.1 Charakteristika der physikalischen Platte

Unter dem Betriebssystem TAXO kann jede Platte Systemdateien, Systemverwaltungsbereiche und Anwenderdateien enthalten. Das Betriebssystem verwendet ein Plattenlaufwerk zum Laden des Betriebssystems selbst. Diese Platte (bezeichnet als Systemplatte) wird auch verwendet, um die internen plattenorientierten Funktionen auszuführen. Die TAXO Systemplatte oder die sekundären Platten haben die unten beschriebenen Charakteristika.

3.1.1 Systemverwaltungsbereich

Spur 0 und 1 werden sowohl auf der Systemplatte wie auch auf Sekundärplatten als Systemverwaltungsbereich verwendet.

Jede Platte wird im TAXO durch einen vom Anwender definierten Namen identifiziert. Diese Namen sind Bestandteil der Platte und werden dort aufgeschrieben. Der zugewiesene Name wird Volume ID genannt.

Das Betriebssystem führt auf jeder Platte eine Liste der benützten und unbenützten Bereiche.

Auf der Systemplatte ist der Systemlader gespeichert, welcher dazu dient, während des Urladevorgangs (Initial Program Load, IPL) das Betriebssystem in den Speicher zu bringen.

TAXO führt auf der Platte außerdem eine Liste der Bereiche, die unbrauchbar sind. Diese Liste der schlechten Plattenbereiche wird während der Platteninitialisierung angelegt. Im laufenden System wird die Liste der schlechten Plattenbereiche automatisch ergänzt.

3.1.2 Dynamische Dateibereiche

Alle Bereiche auf einer Platte, ausgenommen Spur 0 und 1, werden dynamisch den Dateien zugewiesen.

Jede Platteneinheit hat ein spezielles Dateiverzeichnis, welches VCATALOG genannt wird und in dem TAXO ein Inhaltsverzeichnis der Platteneinheit führt. Die Dateien, die im VCATALOG beschrieben sind, können Dateien sein oder selbst wieder Verzeichnisse (siehe Abbildung 3.).

Zum Zeitpunkt des Urladevorgangs kann jedes beliebige Plattenlaufwerk ausgewählt werden, welches die Systemplatte enthält. TAXO besitzt auf der Systemplatte einige Dateien, um die betriebssysteminternen plattenorientierten Funktionen zu unterstützen. Systemdateien können aber auch auf Platten gespeichert sein, die auf Sekundärlaufwerken installiert sind. Solche Platten können dazu verwendet werden, die Systemplatte zu sichern (back up) oder können beim Urladen als alternatives System verwendet werden.

TAXO verwaltet auf den Platten auch die Anwenderdateien. Anwenderdateien können Daten enthalten oder auch wieder Verzeichnisse (directory files). Anwenderverzeichnisse können Dateinamen enthalten, die zwar im Anwenderverzeichnis eindeutig sein müssen, aber im Systemverzeichnis oder in anderen Anwenderverzeichnissen ohne Schwierigkeiten wieder auftreten dürfen.

3.2 Pfadnamen

Jede beliebige Datei auf einer Platte ist eindeutig durch ihren Pfadnamen identifiziert. Der Pfadname für eine Datei ist eine Konkatenation (Hintereinanderschreibung) folgender Komponenten:

Name der Platte,

Name(n) des (der) Directories der übergeordneten Stufe(n) und des

Dateinamens.

Die einzelnen Komponenten eines Pfadnamens werden durch Punkt getrennt.

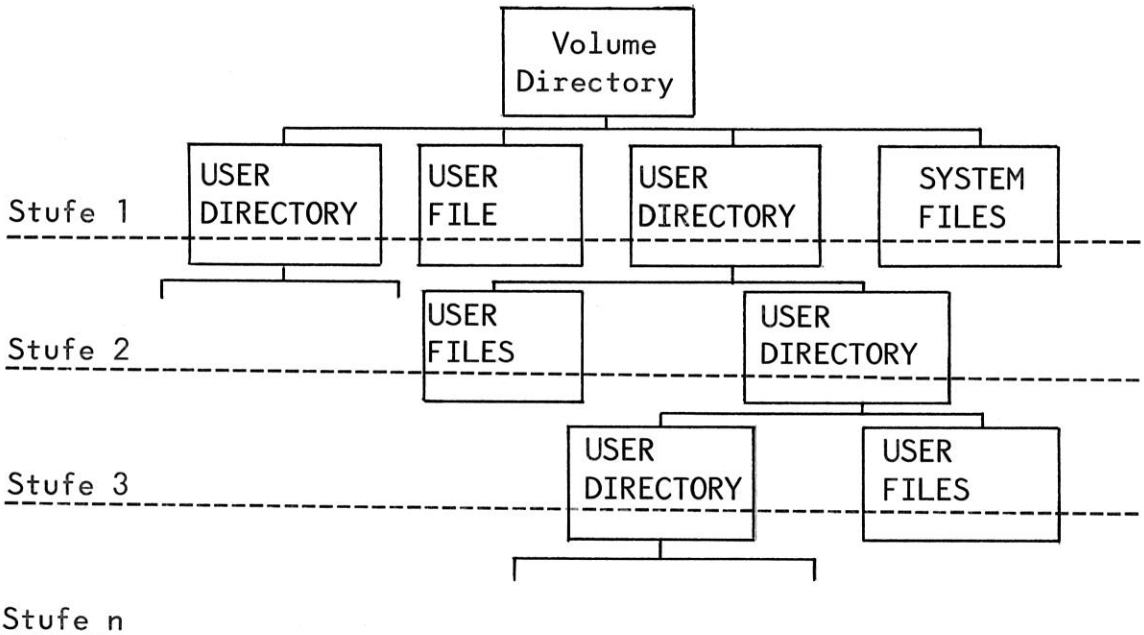


Abb. 3.1 Datei und Directory Struktur

Alle hierarchischen Zwischenstufen müssen in einem Pfadnamen vollständig angegeben werden.

Beispiele: MASTER.SOURCE.COBOL

MASTER.OBJECT.COBOL

MASTER.SOURCE.COBOLDIR.PROG1

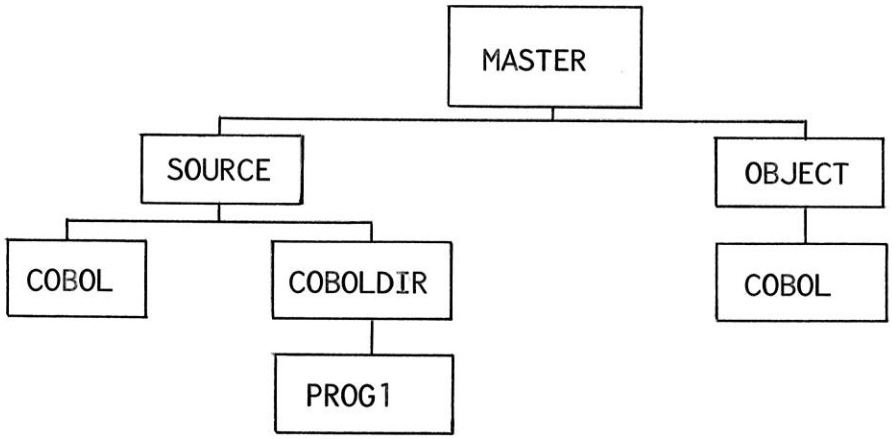


Abb. 3.2 Beispiele für Pfadnamen

4 DATEIORGANISATION

4.1 Allgemeines

Das System basiert auf einem Dateiverwaltungspaket, das ein vollständiges Angebot an Dateistrukturen und Dateieigenschaften umfaßt. TAXO kann viele eindeutig benannte Dateien auf einer Platteneinheit verarbeiten und unterstützt alle notwendigen Maßnahmen, um Plattenbereiche den Dateien zuzuordnen. Der Plattenbereich, der einer Datei zugewiesen werden kann, reicht von zwei Plattensektoren bis zum gesamten verfügbaren Speichervolumen einer Platte. Der Anwender kann den Bereich, den eine Datei auf der Platte belegen soll, selbst festlegen, oder, was häufig geschieht, durch das System automatisch zuordnen lassen.

4.2 Dateitypen

Vom TAXO werden hauptsächlich drei Dateitypen unterstützt: sequentielle, relative oder indexsequentielle Dateien mit mehreren Schlüsseln.

4.2.1 Sequentielle Dateien

Sequentielle Dateien sind nützlich, um Sätze variabler Länge in chronologischer Reihenfolge, wie sie ankommen, aufzuschreiben. Analog müssen die Daten wieder in derselben Reihenfolge gelesen werden. Für jede aktuelle Zuweisung auf eine Datei führt TAXO einen internen Satzzeiger. Bei jedem Lese- oder Schreibvorgang wird dieser Zeiger aktualisiert. Sequentielle Dateien haben die Eigenschaft, daß keine gültigen Daten jenseits des zuletzt geschriebenen Satzes existieren. Die einzige Ausnahme von dieser Regel betrifft den eingeschränkten REWRITE für einen Satz. Eine sequentielle Datei kann segmentiert sein. Außerdem wird ein mehrfaches EOF (End-of-File) innerhalb einer sequentiellen Datei unterstützt. Mehrere Programme können eine sequentielle Datei gleichzeitig an verschiedenen Stellen in der Datei lesen. Eine sequentielle Datei kann nicht von mehreren Programmen im Zugriff sein (shared), wenn ein Programm schreibend auf die Datei zugreift.

4.2.2 Relative Dateien

Relative Dateien erlauben besonders schnellen Direktzugriff. Auf die Sätze fester Länge wird durch TAXO mit Hilfe von Satznummern zugegriffen. TAXO erhöht automatisch die aktuelle Satznummer nach jedem Lese- oder Schreibvorgang, so daß ein sequentieller Zugriff möglich ist. Ein EOF (end-of-file) wird unterstützt.

4.2.3 Indexsequentielle Dateien

In indexsequentiellen Dateien (Multikey Indexed) mit mehreren Schlüsseln werden auf die Sätze variabler Länge mit einem der maximal 14 Schlüssel zugegriffen. Ein Schlüssel ist eine Zeichenkette von maximal 64 Zeichen.

Der Multikey-Zugriff ist in COBOL direkt verfügbar.

Mit Hilfe dieses Dateityps kann zum Beispiel eine Arbeitnehmerdatei erstellt werden, wobei auf jeden Arbeitnehmerstammsatz über den Namen des Arbeitnehmers, dessen Personalnummer, Versicherungsnummer oder einen anderen Schlüssel zugegriffen werden kann.

Schlüssel können so definiert werden, daß sie sich innerhalb eines Satzes überlappen. Außerdem können die Schlüssel beliebig strukturiert sein und an jeder Stelle des Satzes stehen. Ihre relative Position im Satz muß für alle Sätze einer Datei identisch sein. Einer der 14 möglichen Schlüssel muß als Hauptschlüssel (primary key) ausgezeichnet sein. Der Hauptschlüssel muß in allen Sätzen vorhanden sein. Die Sekundärschlüssel können jedoch in jedem beliebigen Satz der Datei fehlen.

Das TAXO Betriebssystem berechnet die Position eines Satzes innerhalb der Datei durch den mathematischen Prozeß des "Hashing" über dem Hauptschlüssel. Hashing bezeichnet den Algorithmus, bei dem eine Satznummer aus dem Namen berechnet wird.

Die Schlüsselwerte für den Haupt- und die Sekundärschlüssel werden innerhalb der Datei in Indextabellen verwaltet. Diese Indizes sind hierarchisch strukturiert und erlauben somit einen schnellen Direktzugriff und darüber hinaus einen sequentiellen Zugriff in sortierter Reihenfolge über jeden beliebigen Schlüssel. Bei Update-Verarbeitung für eine indexsequentielle Datei kann der Anwender in jedem beliebigen Satz die Werte von Sekundärschlüsseln ändern, Schlüssel hinzufügen oder löschen.

Das Betriebssystem TAXO schreibt während der Update-Verarbeitung einer indexsequentiellen Datei von jedem zu veränderten Satz eine Kopie des ursprünglichen Satzinhaltes. Auf diese Weise ist es dem System möglich, unvollständige Update-Vorgänge zu rekonstruieren und somit die Stabilität einer Datei zu garantieren.

4.2.4 Directory-Dateien

Die Struktur der Directory-Dateien ist nur den SCI Kommandos bekannt oder speziellen Systemprogrammen, die Directoriers verändern. Anwendungsprogramme dürfen Directories nicht verändern.

Ein Directory ist ein Verzeichnis derjenigen Dateien bzw. Directories, die der nächsten tieferen hierarchischen Stufe angehören. Das Verzeichnis enthält neben dem Namen der Datei auch Informationen über den Dateityp, Sätzlänge usw.

4.2.5 Programmdateien

Die Programmdateien sind wie relative Dateien strukturiert. Sie enthalten ladbare Programme und die zugehörigen Verwaltungsinformationen für das Betriebssystem. Programmdateien dürfen nur über SCI Kommandos verändert werden.

4.3 Dateieigenschaften

4.3.1 Lösch- und Schreibschutz

Nach jedem Anlegen einer Datei kann diese vor unbeabsichtigter Zerstörung (Überschreiben oder Löschen) geschützt werden. In manchen Fällen kann es auch wünschenswert sein, eine Datei auf lange Zeit zu schützen. Das Betriebssystem TAXO erlaubt daher, eine Datei vor dem Schreiben zu schützen (write protection). Somit können die betreffenden Daten nur noch gelesen werden. Dateien, die schreibgeschützt sind, sind automatisch vor dem Löschen geschützt.

4.3.2 Für eine Reihe von Dateianwendungen ist es zweckmäßig, gewissen Programmen bestimmte Zugriffsprivilegien zuzuordnen. Diese Privilegien sind während der gesamten Arbeit mit einer Datei zwischen Open und Close gültig.

Die Privilegien sind:

- EXCLUSIVE ACCESS (Exklusiver Zugriff) - Nur das rufende Programm kann auf einer Datei zugreifen.
- EXCLUSIVE WRITE ACCESS (Exklusiver Schreibzugriff - Nur das rufende Programm kann auf der Datei schreiben.
- SHARED ACCESS (Verteilter Zugriff) - Das rufende Programm teilt den Zugriff lesend und schreibend mit anderen Programmen.
- READ ONLY (Nur lesender Zugriff) - Das rufende Programm darf auf der Datei nicht schreiben, sondern nur lesen.

4.3.5.3 Expandierbare Dateien

Das Betriebssystem TAXO erlaubt die Festlegung der Dateigröße zum Zeitpunkt der Dateidefinition. Wenn nicht anders spezifiziert, kann eine Datei über seine ursprüngliche Größe hinauswachsen und zusätzlichen Platz belegen. Diese nachfolgenden Plattenspeicherbelegungen werden von Mal zu Mal größer, wenn die Datei die aktuelle Erweiterung überschreitet.

4.3.5.4 Geblockte Dateien

Mehrere logische Sätze können automatisch durch TAXO in größere physikalische Sätze zusammengefaßt werden. Diese größeren Sätze werden Dateiblöcke und physikalische Sätze genannt. Das Blocken spart Plattenplatz und vermindert die Anzahl der physikalischen Transfers von Daten zwischen Speicher und Platte.

4.4 IBM-kompatible Diskette

5 LOGISCHE EIN/AUSGABE PROZEDUREN

5.1 Geräte- und Zugriffsnamen

Das Betriebssystem TAXO unterstützt die Zuweisung von 4 Zeichen langen Namen für jedes periphere Gerät während der Systemgenerierung. Mit Hilfe dieser Namen wird die Peripherie zur Laufzeit des Betriebssystems identifiziert. Dateien werden durch Pfadnamen identifiziert. Somit kann eine Platte, auf der eine Datei residiert, sowohl durch den physikalischen Gerätenamen des Plattenlaufwerks als auch durch den Plattennamen (Volume ID) referenziert werden. Die Dateiidentifikatoren folgen dem Gerätenamen, dem Plattennamen bzw. der Systemplattenkennung. Falls der Plattenname fehlt und der Pfadname mit einem Punkt beginnt, wird angenommen, daß die betreffende Datei auf der Systemplatte zu finden ist. Beispiele von gültigen Namen von Geräten und Dateien sind in Tabelle 5.1 zu finden.

Allgemein werden die Namen von Geräten und Dateien Zugriffsnamen genannt. Zugriffsnamen können somit sowohl Geräte- als auch Pfadnamen sein.

Tabelle 5.1 Beispiele für Geräte- und Dateinamen

Dateibezeichner	Bedeutung
LP01	Gerätename (Drucker Nr. 1)
DS01. MYCAT. MYFILE	(Platte Nr. 1), Directory Name, Dateiname
.MYCAT. MYFILE	Systemplatte, Directory Name, Dateiname
VOLID. MYCAT. MYFILE	Plattenname, Directory Name, Dateiname

5.2 Logische Gerätenummern

TAXO verwendet für Ein/Ausgaben anstatt physikalischer Gerätebezeichnungen ausschließlich logische Gerätenummern. Dadurch werden die Programme bezüglich der Aufteilung von Ein/Ausgaben flexibler. Ein Beispiel: Ein Programm wurde geschrieben, das Eingeben von der logischen Gerätenummer (logical unit number, LUNO) 82 erwartet. Vor Start des Programms wurde der logischen Gerätenummer 82 der gewünschte Zugriffsname (ein Geräte- oder Dateiname) zugewiesen. Eine LUNO-Zuordnung kann sowohl interaktiv durch den Anwender erfolgen oder durch das Programm selbst, mit Hilfe eines entsprechenden Betriebssystemdienstes (Supervisor Call, SVC).

Die Zuweisung einer logischen Gerätenummer kann nur für das Programm, das die Zuweisung tätigte, (task Local), für alle Programme eines gegebenen Bildschirmarbeitsplatzes (terminal local) oder für alle Programme (global) gelten. Die sogenannten 'task' und 'terminal' lokalen LUNO-Zuweisungen erlauben verschiedenen Programmen oder verschiedenen Anwendern desselben Programms die Verwendung desselben LUNO unter der Voraussetzung, daß unterschiedliche Zugriffsnamen zugeordnet sind.

5.3 Dateiorientierte Geräte

Manche Geräte können zum Systemgenerierungszeitpunkt als dateiorientiert erklärt werden. Wenn ein Gerät dateiorientiert ist, wird dieses Gerät zum exklusiven Eigentum desjenigen Programms, das eine Open Operation erfolgreich auf die LUNO ausführt, die diesem Gerät zugewiesen ist. Für andere Programme wird dieses Gerät erst wieder verfügbar, wenn eine entsprechende Close Operation erfolgreich abgeschlossen ist.

5.4 Satzorientierte Geräte

Die Alternative zum dateiorientierten Gerät ist das satzorientierte Gerät. In diesem Fall können Sätze von verschiedenen Programmen freizügig auf diesem Gerät ausgegeben werden.

5.5 Ein/Ausgabe Supervisor Call

Mit Hilfe des E/A Supervisor Calls wird jeglicher Transfer von Sätzen und das Positionieren in Dateien von einem Programm ausgeführt. Zusätzlich erlaubt der E/A Supervisor Call gewisse Dienste, wie das Erzeugen einer Datei und das Zuweisen einer LUNO. Eine Liste der verfügbaren Operationen ist in Tabelle 5.2 zu finden.

Darüber hinaus gibt es im TAXO Supervisor Calls, die das Angebot an E/A Operationen ergänzen und unterstützen:

- Warten auf die Beendigung einer bestimmten, vorher angestoßenen Ein/Ausgabe Operation
- Warten auf die Beendigung einer beliebigen, vorher angestoßenen Ein/Ausgabe Operation
- Abbrechen einer vorher angestoßenen Ein/Ausgabe Operation
- Erkennen eines speziellen Tastaturzeichens (Funktionstasten)

Tabelle 5.2 Geräte- und Dateioperationen mittels E/A SVC

Assign LUNO
Release LUNO
Fetch Characteristics of Device or File
Verify Legality of Access Name
Open LUNO
Close LUNO
Close LUNO and Write End-Of-File
Open LUNO and Rewind
Forward Space Record
Backward Space Record
Read Record
Write Record
Read Direct
Write Direct
Write End-of-File
Rewind
Unload
Rewrite the Previous Record
Create a File
Delete a File
Establish Immediate/Deferred Write Mode
Change a File Name
Write Protect/Delete Protect/Unprotect a File
Add an Alias Name for a File
Delete an Alias Name for a File
Unlock a Record
Key Index File Operations
Open Extend
Modify Access Privileges

6 PROGRAMMIERSYSTEM

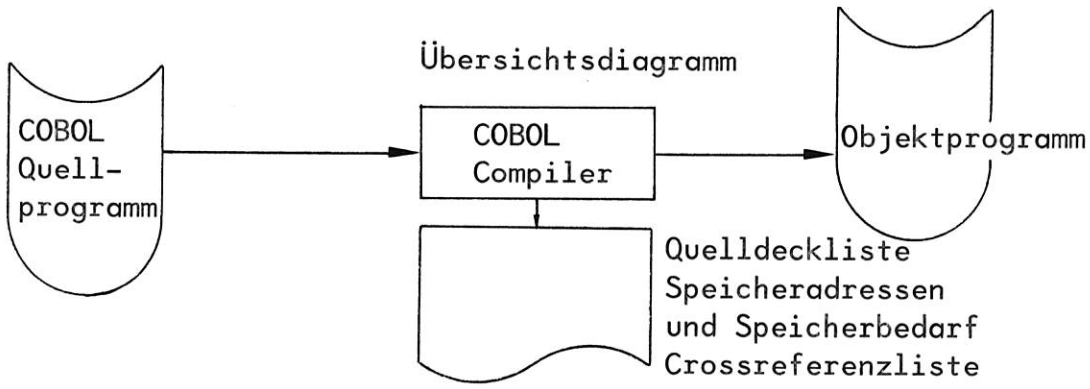
6.1 COBOL

COBOL ist eine höhere Programmiersprache, die die Problemlösung in Worten und Syntax, ähnlich der englischen Sprache , erlaubt. Sie besteht aus einer Menge von englischen Wörtern und Symbolen, mit deren Hilfe ein Programmierer seine Probleme formulieren kann und ein lauffähiges Programm erhält. Da diese Sprache dem Englischen ähnlich ist, sind Programme, die in COBOL geschrieben sind, nahezu selbstdokumentierend, und die Zeit, um einen Programmieranfänger diese Sprache zu lehren, wird sehr verkürzt.

COBOL Programme können direkt mit Hilfe eines speziellen Ausführungskommandos gestartet werden oder sie können mit Hilfe des Link Editors gebunden und in einer TAXO-Programmdatei installiert werden.

6.1.1 COBOL-Compiler

Der COBOL-Compiler ist ein 1-Pass-Compiler, der als Eingabe ein COBOL-Programm erwartet und als Ausgabe ein binäres Objektprogramm und die Liste des Quellprogramms produziert. Die Liste des COBOL-Quellprogramms enthält die relativen Speicheradressen der COBOL-Datenelemente, eine Zusammenfassung über den Speicherbedarf des Programms und -falls gewünscht- eine Crossreferenzliste.



Der Compiler verarbeitet den Eingabetext sequentiell, wobei immer dann ein neuer Satz abgerufen wird, wenn er benötigt wird. Während der Quelltext verarbeitet wird, werden globale Daten wie auch Diagnosemeldungen -falls erforderlich- in Tabellen aufgezeichnet. Wenn die Untersuchung eines "Abschnitts" des Quelldecks abgeschlossen ist, werden die Original-Quellprogrammzeilen mit den dazugehörigen Fehlermeldungen ausgegeben. Während der Verarbeitung des Quelltextes wird Objektcode generiert und ausgegeben.

Zwei Eigenschaften unterscheiden den vorliegenden Compiler von den meisten Sprachenprozessoren:

- Die Speicherverwaltung und
- die Sprache, in der der Compiler geschrieben ist.

Um bestmögliche Speicherausnutzung zu gewährleisten, verwaltet der Compiler seine Tabellen dynamisch. Die dynamischen Tabellen werden ROLLs genannt, der Speicherbereich, in dem sie sich befinden, heißt ROLL MEMORY. ROLLs besitzen variable Länge und können als "Last in, First out"-Keller betrachtet werden; sie können innerhalb des ROLL MEMORY beliebig verschoben werden.

Der Compiler ist zum größten Teil in einer interpretierten Sprache, der POP-Sprache geschrieben. POP-Instruktionen sind als Einadreß-Befehle konzipiert. Ein POP wird interpretativ über ein Programm, das in Maschinensprache geschrieben ist, ausgeführt, indem innerhalb dieses Programms zu einem Unterprogramm gesprungen wird, das die gewünschte POP Operation ausführt.

Das binäre Objektprogramm, das der Compiler generiert, besteht aus Folgen von Objektstruktionen, die vom COBOL-Laufzeitsystem ausgeführt werden.

6.1.2 Sprache

Die implementierte COBOL-Sprache basiert auf dem ANSI X3.23-1974 COBOL Standard. Die folgende Tabelle zeigt den Umfang der Implementierung des COBOL Standards.

	Level 1	Level 2
NUCLEUS	▨	▨
TABLE HANDLING	▨	
SEQUENTIAL I-O	▨	▨
RELATIVE I-O	▨	▨
INDEXED I-O	▨	▨
SORT-MERGE		
REPORT WRITER		
SEGMENTATION	▨	
LIBRARY	▨	
DEBUG*	▨	
INTER-PROGRAM COMMUNICATION	▨	
COMMUNICATION		

* Implementiert ist die Funktion der "Debug"-Zeile und ein systemspezifischer interaktiver Laufzeitdebugger.

Das implementierte COBOL weist nur wenige Ausnahmen bezüglich des ANSI X3.23-1974 COBOL auf, die ebenso wie die Erweiterungen gemäß COBOL-Standard im COBOL-Programmierhandbuch detailliert beschrieben sind.

Als besondere Erweiterung, die nicht im Standard COBOL enthalten ist, ist die Einführung eines SCREEN-I/O-Moduls zu bezeichnen, die zur Unterstützung der Bildschirmprogrammierung dienen soll.

6.1.3 COBOL-Laufzeitsystem

Das vom Compiler erzeugte Objektprogramm hat folgendes Aussehen:

Aus jedem COBOL-Befehl wird eine Folge von Objektcodeanweisungen erzeugt. Es handelt sich dabei um Einadreßbefehle, die in der sogenannten PSECT stehen.

Darüber hinaus werden Daten in verschiedenen Sections abgelegt:

ESECT:
Es werden explizite Daten gespeichert (i.a. Konstante).

FSECT:
Für jede SELECT-Anweisung wird ein COBOL FILE DEFINER (CFD) und ein FILE DEFINITION BLOCK (FDB) generiert. Beide sind in der PSECT enthalten, miteinander verzeigert und beschreiben die definierte Datei.

LSECT:
Hier werden alle Literale abgelegt.

TSECT:
Die hier gespeicherten temporären Daten sind eine sich ändernde Menge von Daten, die durch Referenzen aus den Anweisungen benötigt werden.

DSECT:
Es handelt sich hier um implizite Daten, z.B. zur Beschreibung von Datenelementen usw. (Typ, Länge, num. Zeichenpositionen usw.).

Das COBOL-Laufzeitsystem liest die einzelnen Objektcodeanweisungen in der PSECT und führt in Abhängigkeit vom Objectcode entsprechende Assemblerunterprogrammroutinen aus. Diese verarbeiten die zugehörige Datenadresse, indem sie sie entweder als Direktoperand in einer der beschriebenen Sections interpretieren oder sich über Daten- bzw. File-Beschreibungen die in mehreren Sections stehenden Parameter holen. Auf diese Art und Weise kann ein COBOL-Befehl durch Ausführung mehrerer Objektanweisungen unter Zugriff auf entsprechende Parameter in den Sections ausgeführt werden.

6.1.4 Bildschirmeigenschaften

Um die durch die Hardware zur Verfügung gestellten umfangreichen Möglichkeiten der Bildschirm-Ein/Ausgabe voll nutzen und möglichst einfach programmieren zu können, wurde die Sprache COBOL um einen Screen-I/O-Modul erweitert.

Der Screen-I/O-Modul unterstützt die vorhandenen Bildschirm-Hardwarefunktionen und stellt dem Anwender Sprachelemente zur Definition und Verarbeitung einer Bildschirm-Dateiorganisation zur Verfügung.

6.1.4.1 Dateiorganisation für den Bildschirm

Der physikalische Bildschirm kann per COBOL in mehrere logische Bildschirmbereiche unterteilt werden. Für jeden Bildschirmbereich wird in COBOL eine Datei angelegt. Jedem Bildschirmbereich kann eine Maske zugeordnet werden, d.h. es werden in einer neu eingeführten SCREEN SECTION geschützte Felder mit Texten definiert und auf den Bildschirm ausgegeben. Bei der Eingabe von Benutzerdateien dienen diese Maskenfelder als Bedienerführung und können nicht verändert werden.

Zwischen diesen Maskenfeldern stehen i.a. Eingabefehler, die vom Benutzer ausgefüllt und nach vollständiger Eingabe eines logischen Bildschirmbereiches in den Satzbereich der zugehörigen Datei übertragen werden. Aus diesem Grund enthält die Satzbeschreibung für einen logischen Bildschirmbereich nur die Eingabefelder.

Sollen in einem Programm mehrere Masken verarbeitet werden, so ist es sinnvoll, diese über ein getrenntes Programm zu definieren und auf Platte zu speichern. Es besteht dann die Möglichkeit, vordefinierte Masken einzulesen und direkt auf den Bildschirm auszugeben.
Zur Definition und Abspeicherung von Masken steht ein Dienstprogramm MASGEN zur Verfügung, das die Möglichkeit bietet, interaktiv am Bildschirm Masken zu erstellen.

6.1.4.2 Unterstützung der Bildschirm-Hardwarefunktionen

Als erstes kann der Verarbeitungsmodus festgelegt werden:

a) Page Modus
Ein über eine Datei-Definition festgelegter Bildschirmbereich wird vom Benutzer ausgefüllt und die Eingabe wird durch eine Auslösetaste beendet. Dadurch wird der gesamte Bildschirmbereich (ungeschützte Felder) in den Satzbereich übertragen. Die Summe der Längen der ungeschützten Felder des Bildschirmbereiches ist also gleich der Satzlänge.

b) Scroll-Modus
Die Eingabe in den logischen Bildschirmbereich erfolgt immer in der untersten Zeile. Nach Beendigung einer Zeileneingabe wird der gesamte Bildschirmbereich um eine Zeile nach oben geschoben und der Inhalt der oben aus dem Bildschirmbereich herausfallenden Zeile (ungeschützte Felder) wird in den Satzbereich der zugehörigen Datei übertragen. Ein Satz besteht also aus den ungeschützten Feldern einer Zeile des Bildschirmbereiches.

Zusätzlich zum Verarbeitungsmodus können jedem Feld sog. Feldattribute zugeordnet werden.

Folgende Feldattribute sind möglich:

- NOT OPTIONAL: Zwangseingabe: wenigstens ein Zeichen muß eingegeben werden
- EXACT: Vollaustastung: alle Zeichen des Feldes müssen eingegeben werden
- CHECK Prüfroutine: die eingegebenen Daten werden durch eine frei in COBOL programmierbare Prüfroutine auf Richtigkeit untersucht
- NO ERASE: Konstantenfeld: das Feld wird durch einen DELETE-Befehl nicht gelöscht
- PROTECTED: geschütztes Feld: das Feld kann bei der Eingabe nicht verändert werden, der Cursor wird automatisch auf das nächste Eingabefeld positioniert.
- INVERSE: die Darstellung des Feldes erfolgt dunkel auf hellem Hintergrund.
- BLINKING: Das Feld blinkt
- BRIGHT: Das Feld wird mit erhöhter Helligkeit dargestellt.
- UNDERLINED: Das Feld wird unterstrichen.
- NOT VISIBLE: Das eingegebene Feld wird am Bildschirm nicht abgebildet.

Alle diese Feldattribute können in der SCREEN SECTION bei jeder Datenbeschreibung mit Hilfe der FIELD ATTRIBUTE-Klausel angegeben werden (Syntaxbeschreibung siehe nächste Seite); darüber hinaus werden aus den üblichen Klauseln der Datenbeschreibung weitere Feldattribute abgeleitet:

- JUSTIFIED RIGHT: Die Eingabe in ein alphanumerisches Feld erfolgt rechtsbündig.

- numerisch, alphanumerisch Unterdrückung führender Nullen
- Dezimalpunktausrichtung

Diese Informationen werden aus der PICTURE-Zeichenkette abgeleitet.

Für Anwendungsfälle, bei denen keine Datei-Verarbeitung erforderlich ist, besteht natürlich auch die Möglichkeit einer feldweisen Ein/Ausgabe auf den Bildschirm, wobei mit und ohne Feldattribute gearbeitet werden kann.

6.2.4.3 Bildschirmbeschreibung (SCREEN SECTION)

Die Beschreibung des Bildschirmes (vor allem der Bildschirmmasken) erfolgt in der SCREEN SECTION. Dort besteht die Möglichkeit, Bildschirmsätze, aber auch unabhängige Feldbeschreibungen inklusive Feldattribute zu definieren. Die SCREEN SECTION muß die letzte Section innerhalb der DATA DIVISION sein:

Bildschirmdatenbeschreibung

```
Stufennummer    { Datenname-1 }
                { FILLER      }
                { ATTRIBUTE   }

[ ; { PICTURE } IS Zeichenkette ]
    { PIC     }

[; USAGE IS DISPLAY

[;[SIGN IS] TRAILING [SEPARATE CHARACTER]]

[ ; { JUST      } RIGHT ]
    { JUSTIFIED }

[; BLANK WHEN ZERO]

[; VALUE IS Literal]

[; LINE IS [PLUS] Ganzzahl-1]

[; POSITION IS [PLUS] Ganzzahl-2]

[;FIELD ATTRIBUTE IS

    [ { [NOT OPTIONAL] [EXACT] [WITH CHECK] [NO ERASE] } ]
      {                PROTECTED                       }

    [ { [INVERSE] [BLINKING] [BRIGHT] [UNDERLINED] } ]
      {             NOT VISIBLE                    }

     [ Merkname [,Merkname] ...] ] .
```

6.2 Hilfsmittel für die Programmentwicklung

Zusätzlich zu dem umfassenden Vorrat an Dienstprogrammen, die in Verbindung mit dem Betriebssystem TAXO arbeiten, gibt es vier besonders wichtige Hilfsmittel für die Programmentwicklung.

- o einen interaktiven Texteditor,
- o einen Makro Assembler,
- o einen Link Editor und
- o ein Testpaket Debug.

6.2.1 Interaktiver Texteditor

TAXO unterstützt einen interaktiven Texteditor, der von jedem satzorientierten interaktiven Bildschirmgerät aus bedient werden kann. Die Editierungsoperationen können sowohl über spezielle Editierungstasten als auch über Kommandos ausgelöst werden. Falls die Operationen über ein Kommando angestoßen werden, werden durch das Betriebssystem die notwendigen Parameter auf dem Terminal angefordert. Die Editierungsoperationen erlauben die Modifikation, die Einfügung und das Löschen ganzer Sätze oder Zeichenketten innerhalb eines Satzes.

6.2.2 Makro Assembler

Der TAXO System Assembler ist ein sehr mächtiges Hilfsmittel zur Programmierung der Maschinensprache. Darüber hinaus besitzt der Makro Assembler Erweiterung hinsichtlich der Makroprogrammierung und Möglichkeiten zum bedingten assemblieren. Die Makroprogrammierung unterstützt die Manipulation von Zeichenketten, den Zugriff auf Binärwerte in der Symboltabelle und den Zugriff auf Makrodefinitionsbibliotheken.

Der Objektcode, der durch den Assembler erzeugt wird, ist verschieblich (relocatable) und kann in Segmente eingeteilt werden. Jedes Segment besitzt seinen eigenen Programmzähler. Die Segmente können sein: Common Bereiche, Programmsegmente und Datensegmente. Der Link Editor des Systems sammelt die Segmente gleichen Typs in einem zusammenhängenden Speicherbereich. Eine Folge von Assemblerbefehlen kann in Abhänigkeit von einem Wert der Assemblerarithmetik oder eines logischen Wertes bedingt übersetzt werden. Zusätzlich gibt es Anweisungen für den Assembler, eine gewisse Anzahl von Informationen im Assembler Listing mit auszugeben.

6.2.3 Link Editor

Der TAXO System Link Editor kann verschieblichen Objektcode des Assemblers oder des COBOL Compilers verarbeiten und die einzelnen Module in zusammenhängende Lademodule umwandeln. Verweise zwischen den Modulen werden dabei befriedigt. Commonbereiche, Programmsegmente und Datensegmente werden gesammelt und jedem Segmenttyp wird sein eigener zusammenhängender Speicherbereich zugewiesen.

Der Link Editor verarbeitet Kontrollanweisungen, mit deren Hilfe die Verwendung sogenannter 'shared' Prozeduren und die Overlay Struktur definiert werden kann. Zusätzliche Möglichkeiten sind bezüglich der Generierung einer Ladeliste gegeben oder des automatischen Suchens nach Objektbibliotheken, um automatisch offene Referenzen zu lösen. Außerdem kann auch ein teilweises Linken durchgeführt werden, so daß externe Referenzen in einem späteren Zeitpunkt aufgelöst werden können. Die Ausgabe des Link Editors kann auf eine Objektdatei erfolgen oder direkt als Speicherabbild in eine Programmdatei oder in eine TAXO Imagedatei.

6.2.4 Testpaket Debug

Das Testpaket DEBUG ist ein interaktives, symbolisches Testprogramm für Assemblerprogramme. DEBUG erlaubt das Anzeigen und Modifizieren der Register der arithmetischen Einheit, der sogenannten 'Workspace' Register und des Speichers. Mit DEBUG kann die Ausführung einer Task überwacht werden. Während der Laufzeit einer Task kann diese angehalten und wieder gestartet werden. Mit Hilfe von Haltepunkten kann die Ausführung einer Task gezielt unterbrochen werden. Die Ausführung einer Task kann auch simuliert werden. In diesem Falle wird zwischen jeder Anweisung eine Analyse durchgeführt. Die Art und Weise, wie der Programmzähler oder der Speicherinhalt abgefragt wird, kann spezifiziert werden.

6.2.5 Sort/Merge

TAXO verfügt über ein umfassendes Sort/Merge-Paket, auf das in vielfältigster Weise zugegriffen werden kann. Über den SCI kann das Sort/Merge-Paket sowohl im interaktiven als auch im Batch Modus bedient werden. COBOL-Programme können über die CALL Anweisung an das Sort/Merge-Paket angeschlossen werden. Sowohl der Sortier- als auch der Mischprozeß unterstützen die Satzauswahl, das Reformatieren der Eingabe und das Aussummieren in der Ausgabe. Die Abbildung 6.1 zeigt ein Beispiel für einen Sortier- und Mischvorgang. Aufsteigende Schlüsselordnung oder eine andere beliebige Sortierfolge kann definiert werden. Jede beliebige Anzahl von Schlüsseln kann spezifiziert werden, solange die Gesamtlänge aller Schlüssel kleiner als 256 Zeichen ist. Der Mischprozeß erlaubt maximal 5 Eingabedateien. Die unterschiedlichen Arten von Sortierprozessen sind:

- voller Sortierprozeß,
- Sortieren nach Adressen,
- Sortieren nach Schlüsseln und
- Summensort.

Abteilung A Verkäufe des Monats
Datei: Monat. Abtlg. A

1 67 89 11 12 40

Artikel-Nr.	Menge	Gesamtbetrag	Name
1000-1	10	100	BAKER
3000-1	10	300	CHARLES
2000-1	20	400	ABLE
5000-1	20	1000	CHARLES
4000-1	20	200	BAKER

1 67 89 11 12 40

Artikel-Nr.	Menge	Gesamtbetrag	Name
7000-1	5	350	ABLE
8000-1	40	400	DOGGER
2000-1	10	200	BAKER

Schritte 1 und 2: Sortierung beider Dateien in alphabetischer Reihenfolge. Umstellen der Daten, so daß der Name das erste Feld ist.

ABLE	2000-1	20	200
BAKER	1000-1	10	100
BAKER	4000-1	20	200
CHARLES	3000-1	10	300
CHARLES	5000-1	20	1000

ABLE	7000-1	5	350
BAKER	2000-1	10	200
DOGGER	8000-1	40	400

Schritt 3: Mischen beider Dateien zu einer Datei.

ABLE	2000-1	20	200
ABLE	7000-1	5	350
BAKER	1000-1	10	100
BAKER	2000-1	10	350
BAKER	4000-1	20	200
CHARLES	3000-1	10	300
CHARLES	5000-1	20	1000
DOGGER	8000-1	40	400

Anmerkung: die hier gezeigten Schritte 3 und 4 sind Beispiele; beide können durch ein Summenmischen ersetzt werden.

Schritt 4: Summieren der monatlichen Gesamtverkäufe an jeden Kunden. Ein fügen eines Dollarzeichens vor dem verkauften Betrag.

ABLE	$	550
BAKER	$	650
CHARLES	$	1300
DOGGER	$	400

Abbildung 6.1 Ein Sort/Merge-Prozeß mit der Beschreibung der Druck-listen und der Ergebnisse aller Einzelschritte

7 KOMMANDOSPRACHE

7.1 Allgemeines

Das Betriebssystem besitzt sowohl eine interaktive Schnittstelle zum Anwender als auch einen Mechanismus für Stapelverarbeitung. ('batch'-Modus). Alle Kommandos werden durch Aktivierung des Systems Kommando Interpreters (SCI) sowohl im interaktiven Modus als auch im Batchmodus interpretiert.

7.2 Interaktive Operationen

Durch den SCI besitzt der Anwender ein exzellentes Hilfsmittel, das Gesamtsystem zu kontrollieren. Alle Eingaben, die von der Bedienkraft einzugeben sind, werden durch erläuternde Texte angefordert. Eingabefelder können einfach editiert werden und werden vom System geprüft. Durch einen erfahrenen Bediener können gewisse Eingaben gespart werden (und damit Zeit), da alle Parameter für ein Kommando eingegeben werden können, bevor sie das System anfordert. Wenn eine Teilliste von Parametern eingegeben wird, werden alle noch nicht vom Bediener spezifizierten Argumente bzw. deren Standardwerte vom System automatisch an den Bediener ausgegeben. Das SCI Paket wird vom Betriebssystem an allen interaktiven Geräten unterstützt, einschließlich der satzorientierten Datenterminals. Für alle Typen von Systemterminals ist eine Hierarchie von Kommando Menues verfügbar. Die Kommando Menues erlauben jedem Terminal, die Kommandos in einer logischen Gruppierung abzurufen.

An jedem Terminal kann ein Anwender ein Vordergrund- und Hintergrundprogramm gleichzeitig aktiv haben. Vordergrundprogramme können mit dem Anwender über das Terminal direkt korrespondieren, während Hintergrundprogramme diese Möglichkeit nicht besitzen.

7.3 Stapelverarbeitung (Batch processing)

Das eine Hintergrundprogramm, das an einem Terminal aktiv sein kann, kann auch eine Kopie des SCI selbst sein. In diesem Fall interpretiert der SCI die Kommandos im Hintergrund und verhält sich wie bei einer Stapelverarbeitung. Die Eingabe erfolgt im Batch Modus von einer beliebigen sequentiellen Datei, aber nicht vom Terminal selbst. Ein Anwender kann die Stapelverarbeitung initialisieren, ihren Status abfragen und Informationen über deren normale oder abnormale Beendigung erhalten.

7.4 Synonyme

Lange Zeichenketten, die gewöhnlich von einer Bedienkraft einzugeben sind, können mit Hilfe von Synonymen abgekürzt werden. TAXO führt eine Liste der Synonyme, die ein Anwender verwendet. Wenn ein Synonym durch den Bediener verwendet wird, wird es intern durch das Betriebssystem durch die aktuelle lange Zeichenkette ersetzt.

7.5 Flexible Kommando-Prozeduren

Alle in der Anlage vorhandenen Bedienerkommandos sind in Form von Dateien, die in einem speziellen Verzeichnis (Directory) katalogisiert sind, auf der Systemplatte gespeichert. Die Kommandospezifikationen sind in einer speziell dafür geschaffenen höheren Programmiersprache geschrieben. Dadurch, daß ein Anwender ein Kommando eingibt, interpretiert der SCI diese höheren Programmiersprachelemente und führt somit das Anzeigen von Parameteranfragen, das Einlesen von Daten und das Verifizieren von Funktionen aus. Instruktionen innerhalb der Sprachstruktur teilen dem System mit, wie gewisse Daten zu sammeln sind und an welche Ausführungsprogramme diese Daten weiterzuleiten sind. Diese Vorgehensweise vereinfacht das Hinzufügen oder Ändern von anwendungsspezifischen Bedienerkommandos.

7.6 Verfügbare Kommandos

TAXO bietet eine umfangreiche Liste von Kommandos an, die verschiedene Dienstfunktionen ausführen. Viele Kommandos sind zusätzlich für den Programmierer geschaffen worden, um ihm die Entwicklung von Anwendungssoftware unter TAXO zu erleichtern. In Tabelle 7.1 wird eine abgekürzte Liste der Funktionsbereiche vorgestellt, für die ein oder mehrere Kommandos existieren

Tabelle 7.1 Funktionsbereiche der TAXO-SCI-Kommandos

- An- und Abmelden des Bedieners beim System (Log in/Log out)
- Zeit und Datum initialisieren und abfragen
- Initialisieren von Platten
- An- und Abmelden von Platten beim System
- Sichern, Wiederherstellen und Kopieren von Platteninhalten
- Anlegen und Löschen von Verzeichnissen und Dateien
- Unterstützung von Synonymen
- Zuweisung von Alias Namen für Dateien
- Ändern von Dateinamen und des Dateischutzes
- Anzeigen und Auflisten von Verzeichnissen und Dateien
- Kopieren von Verzeichnissen und Dateien
- Zuweisen, Positionieren und Freigeben von logischen Geräten
- Anzeigen des Systems E/A Status
- Anzeigen des System Taskstatus
- Aktivieren und Kontrollieren von Programmen
- Eingeben, Aktivieren von Batch Kommandos
- Kontrolle der Bedienstationen (Anwenderidentifikation, Terminal, Status usw.)
- Installieren und Entfernen von Programmen
- Aktivierung des System Log
- Testen von Programmen einschließlich solcher Funktionen wie
  Setzen von Stopadressen
  Ausgeben von Speicher/Platteninhalten
  Dezimal/Hexadezimal Arithmetikhilfen
  Interaktives Überwachen von Programmen
- Texteditieren
- Aktivierung von Übersetzern wie
  Assembler
  COBOL
- Aktivierung des Link Editors
- Aktivierung des Sort/Merge Paketes

8 SYSTEMGENERIERUNG, SYSTEMLOGBUCH

8.1 Systemgenerierung

8.2 Systemlogbuch

Das Betriebssystem TAXO bietet die Möglichkeit, ein Logbuch zu führen. TAXO notiert in diesem Logbuch Ein/Ausgabefehler und Taskfehler. Bei den Ein/Ausgabefehlern werden sowohl logische Fehler als auch Hardwarefehler aufgezeichnet. Programme, die abnormal beendet werden, bewirken automatisch eine entsprechende Fehlermitteilung im Logbuch. Anwendungsprogramme können zusätzlich Meldung in das Logbuch eintragen, indem sie sich einer entsprechenden CALL-Schnittstelle bedienen oder eines speziellen Super visor Calls. Durch ein Systemkommando kann der Beginn der Logbuchführung ausgelöst werden. Zu diesem Zeitpunkt wird auch die betreffende Plattendatei oder das Ausgabegerät festgelegt. Im allgemeinen werden zwei getrennte Dateien angelegt. Während in die eine Logdatei geschrieben wird, kann die andere gesichert werden. Alle Logbucheintragungen werden mit Datum und Zeit versehen.

9 BETRIEBSSYSTEMEIGENSCHAFTEN

9.1 Programmverwaltung

Anwenderprogramme, die unter der Kontrolle des Betriebssystems TAXO arbeiten, können aus Daten, Prozeduren und Overlays zusammengesetzt sein. Programme werden installiert und in Programmdateien in der Form des Speicherabbildes gespeichert. Wenn ein Programm aktiviert wird, wird dieses Speicherabbild in einen beliebigen freien Speicherbereich geladen. Die TA 1630 Hardware besitzt eine sogenannte 'Mapping'-Einrichtung für den Speicher, die von der Notwendigkeit entbindet, Programme relokatibel zu schreiben. Ein aktives Programm kann während seiner Lebenszeit vom Betriebssystem mehrere Male auf die Systemplatte ausgelagert werden und in unterschiedliche Speicherbereiche eingeladen werden (Roll in, Roll out). Dieser Vorgang dient dazu, die verfügbare Rechenzeit der Zentraleinheit (CPU) und den verfügbaren Speicher effizient unter den konkurrierenden Programmen aufzuteilen. Wenn ein Programm im Speicher ist und aktiv, bewirbt es sich mit anderen Programmen um CPU-Rechenzeit auf der Basis eines Prioritätenschemas. Wenn ein Programm beendet wird, werden vom Betriebssystem TAXO alle von diesem Programm belegten Betriebsmittel wie Dateien, Geräte und Speicher freigegeben. Die außergewöhnliche TAXO-Programmstruktur wird durch die TA 1630 Hardwareeinrichtung, nämlich dem 'Mapping' für den Speicher, erreicht, die es ermöglicht, drei getrennt geladene Programmsegmente in einem einzigen zusammenhängenden Programmadressraum abzubilden.

9.1.1 Tasks

Eine spezielle Aktivierungsmöglichkeit weist ein Programm als Task aus. Das Betriebssystem TAXO kann für mehrere Tasks gleichzeitig den Speicher, die Rechenzeit und die peripheren Betriebsmittel verteilen. Definitionsgemäß kann jede Task in nur genau einem Ausführungszustand sein. Während eine Task tatsächlich aktiv ist, d.h. ausgeführt wird, sind andere suspendiert oder warten auf Reaktivierung. In einer typischen Mischung von Tasks, warten die meisten Tasks auf Ausführung, abhängig von der Fertigstellung einer Ein/Ausgabe Operation. Während diese Tasks auf die Beendigung einer Ein/Ausgabe Operation warten, haben andere (jedoch immer genau eine zu einem Zeitpunkt) Zugriff auf die Betriebsmittel des Systems und führen Befehle aus.

9.1.2 Gemeinsame Prozeduren und mehrfach vorhandene Tasks

Häufig ist es wünschenswert, daß ein bestimmtes Programm gleichzeitig mehrfach ausgeführt wird. Dies ist der Fall bei Mehrplatzanwendungen oder in industriellen Anwendungen, wo mehrere gleich gebaute Geräte kontrolliert werden müssen. Ein Beispiel könnte ein Programm sein, das mehrere Bankenterminals gleichzeitig bedient.

In vielen Fällen ist der prozedurale Teil eines Programms für jede der gleichzeitigen Ausführungen identisch, während die Daten eindeutig einer jeden einzelnen Programmausführung zuzuordnen sind. Im Betriebssystem TAXO werden gemeinsame prozedurale Teile PROCEDURE genannt, während der eindeutige Teil TASK genannt wird. Die drei möglichen Programmsegmente sind daher auf Prozeduren und eine Task zu verteilen.

Ein Programm, das unter TAXO arbeitet, darf aus einer Task und keiner, einer oder zwei Prozeduren bestehen.
Die Prozeduren können gemeinsam mit anderen ausführbaren Tasks verwendet werden. Das Konzept der gemeinsamen Prozeduren spart Speicher und befreit von der Notwendigkeit den prozeduralen Teil eines Programmes mehrfach im Speicher zu halten. Andererseits ist der Taskanteil tatsächlich für jede einzelne Ausführung eines Programms eindeutig. TAXO unterstützt deshalb bequeme Mechanismen Kopien von Tasks für Mehrfachausführung zu aktivieren.

In den Fällen, wo jede gleichzeitige Programmaktivierung denselben Initialzustand (Daten) besitzt, muß das Speicherabbild eines Programms nur einmal auf der Platte gespeichert werden. Für jeden Bildschirmarbeitsplatz kann dann von einem einzigen Speicherabbild, das in einer Programmdatei auf der Platte gespeichert ist, eine eindeutige Kopie aktiviert werden. Somit wird ebenfalls Speicherplatz auf der Platte gespart und Zeit, indem man davon befreit ist, für jede mögliche gleichzeitige Aktivierung eines Programms mit demselben Initialzustand eine Kopie davon zu installieren.

9.1.3 Gemeinsame Daten zwischen Tasks

Gelegentlich ist es wünschenswert, zwischen zwei oder mehreren Tasks einen Datenblock gemeinsam zu haben. Im TAXO gibt es eine bequeme Methode dies zu erreichen, indem man eine gemeinsame Prozedur definiert, die die Daten enthält, auf die mehrere Tasks gemeinsam zugreifen sollen.

9.1.4 Overlays (Überlagerungen)

Wenn ein Programm groß wird, ist es oft wünschenswert, nur einen bestimmten Teil dieses Programms zu einem Zeitpunkt im Speicher zu halten. TAXO unterstützt die Overlaytechnik, die es gestattet, auf nicht speicherresidente Programmmodule zuzugreifen. Das sogenannte Wurzelsegment (Root) eines Programms ist der permanent speicherresidente Teil. Der Rest des Programms wird in plattenresidente Oberlays aufgeteilt. Genau ein Overlay ist dann speicherresident, wenn es benötigt wird.

9.1.5 Aktivierung einer Task durch ein Programm

Jede Task kann die Aktivierung einer anderen Task bewirken. Das Ergebnis eines solchen Vorgangs ist, daß beide Tasks gleichzeitig aktiv sind. TAXO erlaubt auch die Identifikation eines Arbeitsplatzes, der die neue Task zugeordnet sein soll. Auf diese Weise sind alle terminallokalen LUNO Zuweisungen der neuen Task zugänglich. Weiterhin kann angegeben werden, das die aktivierende Task solange su endiert wird, bis die aufgerufene Task beendet ist. Damit wird für ein Hauptanwendungsprogramm, das an einem Arbeitsplatz aktiv ist, ein bequemer Mechanismus angeboten, Unterprozesse zu aktivieren, die entweder parallel zum Hauptprogramm laufen oder an Stelle des Hauptprogramms. Es ist zu beachten, daß im letzteren Fall, wenn der Unterprozeß beendet ist, das Hauptprogramm wieder fortgeführt wird.

9.1.6 Prioritätensteuerung in TAXO

Das TAXO-Betriebssystem fordert, daß jede Task eine definierte Prioritätsstufe besitzt.
Es sind 4 Prioritätsstufen verfügbar:

Höchste Stufe	0	TAXO interne Verwendung
	1	interaktiver Modus
	2	
Niedrigste Stufe	3	Batch Modus
Gleitende Stufe	4	

Die Prioritätsstufe 0 wird für besonders kritische Systemfunktionen verwendet und ist ausschließlich für das Betriebssystem selbst reserviert.
Prioritätsstufe 1 erlaubt besonders schnelles Antwortverhalten für Programme, die interaktiv mit dem Terminal arbeiten.
Prioritätsstufe 2 ist für Programme vorgesehen, die intensiv Plattenzugriffe ausführen.

Prioritätsstufe 3 ist für Batch-Programme vorgesehen, die keine interaktiven Benutzereingriffe voraussetzen. Auf dieser Stufe werden Tasks nur dann ausgeführt, wenn keine höherpriore Task (interaktive oder Systemtask) auf Ausführung wartet.
Prioritätsstufe 4 schaltet die Prioritätsstufe automatisch zwischen 1 und 2 während der Programmausführung um.

TAXO bringt diejenigen Task zur Ausführung, die die höchste Priorität besitzt. Der TAXO-Scheduler wird dann aktiv, wenn eine der folgenden Bedingungen eintritt:

- Ein externer Interrupt bewirkt den Aufruf einer Task.
- Eine ausführende Task wird suspendiert.
- Task Sentry erniedrigt die Priorität der ausführenden Task.
- Das Time Slice der ausführenden Task wird überschritten.
- Eine Task, die im Time Delay war, wird aktiv.

Falls der Task Sentry die Priorität einer ausführenden Task erniedrigt hat, wird anschließend eine der Tasks ausgewählt, die auf Ausführung warten, falls eine Task höherer Priorität im System existiert.

Aktive Programme der höchsten Priorität werden im "Round-Robin"-Verfahren ("einer nach dem anderen") ausgeführt. Nachdem eine Task ein festes Zeitintervall lang aktiv war, werden alle Tasks im System vom Scheduler erneut überprüft, um einer anderen Task Ausführungszeit zuzuteilen.

9.1.7 Tasks mit variabler Priorität

Wenn eine Task installiert wird, sollte sie mit gleitender Priorität (Stufe 4) versehen werden. Zum Zeitpunkt der ersten Ausführung einer Task wird die Priorität mit Stufe 1 initialisiert. Die Prioritätenstufe der Task wird automatisch nach einer gewissen Anzahl von Zeitscheiben (die Anzahl der Time slices ist ein Generierungsparameter von TAXO) auf Zwei erniedrigt, wenn Berechnungen ausgeführt werden.

Die Priorität wird auf Eins gesetzt, wenn Interaktionen mit dem Terminal ausgeführt werden, und auf Zwei, wenn andere E/A-Geräte angesprochen werden. Die gleitende Priorität (verwaltet durch TAXO) bietet schnelle Antwort auf E/A-Ereignisse und verzögert das Programm während Perioden starker CPU-Belastung. Zum Beispiel sollten Anwendungsprogramme, die Interaktiv arbeiten, normalerweise mit gleitender Priorität installiert werden.

9.1.8 Programmdateien

Alle Tasks, Prozeduren und Overlays werden in Programmdateien installiert, Diese Dateien basieren auf dem relativen Dateityp und sind erweiterbar. Sie enthalten das Programmspeicherabbild in Blöcken, die der Satzlänge der Datei entsprechen. Die Programmdatei enthält innerhalb der Datei ein internes Verzeichnis, das sowohl Zeiger auf ein jedes Speicherabbild in der Datei enthält, wie auch andere relevante Informationen über ein jedes Speicherabbild. Üblicherweise benötigt TAXO zwei Plattenzugriffe, um eine plattenresidente Task, Prozedur oder ein Overlay zu laden - einen Zugriff für das Verzeichnis und einen für das Programmspeicherabbild selbst.

Manchmal kann es vorkommen, daß ein Programmspeicherabbild nicht kontinuierlich auf der Platte gespeichert ist und daher zusätzliche Zugriffe notwendig sind. Eine Programmdatei ist als Systemprogrammdatei ausgezeichnet. Die Systemprogrammdatei enthält ursprünglich nur Programme, die wesentlicher Teil des Betriebssystems TAXO sind. Zur Aufnahme von Anwenderprogrammen sollten andere Programmdateien angelegt werden.

9.1.9 Programmidentifikation

Alle Programmteile, die in einer Programmdatei gespeichert sind (Tasks, Prozeduren oder Overlays), werden zur Installationszeit eindeutig durch eine Nummer spezifiziert. Durch das interne Verzeichnis in der Programmdatei besteht die Möglichkeit, über den Programmnamen zuzugreifen. Ein Programm kann mit oder ohne das replicatible Attribut installiert werden.

9.2 Speicherverwaltung

Das TAXO Betriebssystem bedient sich des Vorteils der TA 1630 'Mapping'-Einrichtung, die es erlaubt, Speicher dynamisch an plattenresidente Tasks, Prozedursegmente und ebenso an Blockpuffer für Dateien zu vergeben. Die zugewiesenen Speicherbereiche können je nach Bedarf auf die Platte ausgelagert werden (Roll out). Die Technik des Ein- und Auslagern (Roll in, Roll out) garantiert eine effektive Ausnützung sowohl des Hauptspeichers wie auf der CPU Zeit.

9.2.1 Roll in, Roll out (Ein- und Auslagern)

Das Betriebssystem TAXO besitzt einen Algorithmus, der es erlaubt, Programmen hoher Priorität vorrangig Speicherplatz vor Programmen niedriger Priorität einzuräumen. Jedes beliebige Programm kann den Platz eines suspendierten Programms beanspruchen. Immer wenn ungenügend Speicherplatz verfügbar ist, um ein Programm auszuführen, sucht TAXO nach passenden Programmen niedriger Priorität oder nach suspendierten Tasksegmenten, um sie auf die Platte auszulagern.

Diesen Prozeß nennt man "Roll out". Analog wird entsprechend der aktuellen Prioritätenverteilung das ausgelagerte Programm wieder von der Platte eingelesen und die Ausführung fortgesetzt. Durch die 'Mapping'-Einrichtung kann ein Programmsegment in einen anderen Speicherbereich zurückgespeichert werden als es zum Zeitpunkt des Auslagerns belegt hat. Der prioritätsorientierte Roll-in/Roll-out Mechanismus bewirkt, das hochpriorisierte Tasks einen garantierten und unmittelbaren Zugriff auf den Speicher besitzen, um auf den Anwender oder andere externe Ereignisse reagieren zu können.

Gemeinsam von mehreren Tasks verwendete Prozeduren können erst dann ausgelagert werden, wenn alle zugeordneten Tasks ebenfalls ausgelagert sind. Ähnlich können Tasks, für die eine Ein- oder Ausgabeoperation durchgeführt wird, nicht ausgelagert werden, bis der entsprechende Transfer beendet ist.
Dateiblöcke sind zusammen mit den Tasks und Prozeduren im dynamischen Speicherbetrieb angelegt. Jeder Datenblock, den das Betriebssystem beim letzten Plattentransfer erhalten hat, kann von TAXO neu belegt werden, wenn der Speicherplatz benötigt wird. Diese Blöcke werden daher auf die zugehörige Dateistelle auf der Platte zurückgeschrieben, um Speicherplatz zu gewinnen (falls die entsprechenden Dateiblöcke das Zurückschreiben erfordern).

9.2.2 Speicherresidente Tasks

Programme, die normalerweise in Programmdateien plattenresident sind, werden vom TAXO jedes Mal geladen, wenn sie aktiviert werden. Das Betriebssystem unterstützt aber auch die Festlegung, ein Programm als speicherresident zu definieren, Speicherresidente Programme werden dann geladen, wenn das Betriebssystem von der Platte geladen wird. Dies geschieht zum sogenannten Urladezeitpunkt (IPL, Initial Program Load). Die speicherresidenten Programme bleiben auch nach Beendigung der betreffenden Programme im Speicher.

9.3 Supervisor Calls

Programme fordern Betriebssystemdienste vom TAXO mit Hilfe sogenannter 'Supervisor Calls' an. Ein Supervisor Call wird durch eine Instruktion eingeleitet , die die Ausführungskontrolle an das Betriebssystem übergibt. Jeder Supervisor Call (SVC) umfaßt einen Informationsblock, der detaillierte Parameter über den gewünschten Systemdient enthält. Die Tabelle 9.1 zeigt alle SVCs, die durch das TAXO Betriebssystem unterstützt werden und einem Anwenderprogramm zur Verfügung stehen. Darüber hinaus gibt es SVCs, die nicht in der Tabelle aufgeführt sind und nur privilegierten Tasks vorbehalten sind.

Tabelle 9.1 Allgemeines Betriebssystem Supervisor Calls

File and I/O Calls	Datei- und E/A-Aufrufe
Program Control Calls	Aufrufe zur Programmkontrolle
Activate a task	Aktiviere eine Task
Activate a task at a specified future time	Aktiviere eine Task zu einem späteren Zeitpunkt
Reactivate a sudpended task	Reaktiviere eine suspendierte Task
Load an overlay	Lade ein Overlay
End of task	Beende eine Task
Momentarily suspend a task	Suspendiere vorübergehend eine Task
Suspend a task for time period	Suspendiere eine Task für eine bestimmte Zeit
Change task priority level	Ändere die Prioritätsstufe einer Task
Determine status of task	Bestimme den Status einer Task
Retrieve input parameters	Wiederhole Eingabeparameter
Task identification services	Dienste zur Identifizierung einer Task
Memory control	Speicherkontrolle
Expand the task's memory segment	Erweitere das Speichersegment einer Task
Contract the task's memory segment	Vermindere das Speichersegment einer Task
Other calls	Andere Aufrufe
ASCII/binary conversion services	ASCII/binär Konvertierungsdienste
Intertask communications	Datenaustausch zwischen Tasks
Log a message	Meldungen in eine Datei für Systemmeldungen eintragen
Fetch time and Julian date	Julianische Datum

9.4 Fehlerkontrolle

Das Betriebssystem TAXO beinhaltet eine Fülle von Fehlerkontrollen. Wenn während der Ein/Ausgabe von Daten ein Gerätefehler festgestellt wird, wird versucht, den Datentransfer zu wiederholen. Nach mehreren Versuchen erst wird der Gerätefehler an das verursachende Programm gemeldet. Zusätzlich besteht die Möglichkeit, den Fehler im Systemlogbuch einzutragen.

Fehlermeldungen werden an die Programme gemeldet, die illegale Supervisor Calls verwenden. In manchen Fällen werden zusätzliche Informationen zurückgemeldet, die den Fehler näher beschreiben.

Die TA 1630 'Mapping'-Einrichtung schützt das Betriebssystem vor Zerstörung durch fehlerhafte Anwenderprogramme. Ähnlich sind Anwenderprogramme gegeneinander vor Zerstörung durch falsche Programme geschützt, es sei denn, die Programme verwenden gemeinsame Prozeduren. Wenn ein Anwendungsprogramm nicht mehr weiter aufgrund einer illegalen Funktion ausgeführt werden kann, wird es vom Betriebssystem abnormal beendet. Eine solche illegale Funktion kann die Adressierung eines Speichers außerhalb des Programmbereiches sein, eine illegale Anweisung oder ähnliches. In diesen Fällen wird eine entsprechende Meldung in das Systemlogbuch eingetragen, die einen Fehlercode enthält, der den Grund für die abnormale Beendigung beschreibt.

Jede Task kann zusätzlich eine Befehlsfolge enthalten, die im Falle der abnormalen Beendigung auszuführen ist. Diese Befehlsfolge nennt man 'end-action'-Routine. Das Betriebssystem TAXO kehrt im Falle des Programmabbruches ein letztes Mal zum beendeten Programm zurück, indem es dessen 'end-action'-Routine aktiviert. Der Anwendungsprogrammierer kann nun in dieser Routine die Abbruchbedingung analysieren und entsprechende Rettaktionen einleiten.

In manchen schwerwiegenden Umständen kann ein Systemfehler vorkommen. Solche Umstände sind Hardwarefehler der Systemplatte während einer kritischen Operation. Das Betriebssystem selbst wird dann abnormal beendet. In diesen seltenen Fällen wird ein sogenannter Crashcode am Frontpanel an ezeigt. Das Betriebssystem TAXO ermöglicht es dem Bediener, mit Hilfe eines einfachen Operating einen Speicherabzug (post-mortem image) auf die Platte zu schreiben, wenn ein Systemzusammenbruch vorausgegangen war.