



**Das Dokumentations -**

**und**

**Source - Verwaltungssystem**

**COBOL - Analyser Bedienungsanleitung**

**Version: A.03.00**

**Stand: 7. Mai 2001**

## **Hinweis**

DieSSD IT Consulting GmbH haftet nicht für etwaige Fehler in dieser Dokumentation. Eine Haftung für mittelbare und unmittelbare Schäden, die im Zusammenhang mit der Lieferung oder dem Gebrauch dieser Dokumentation entstehen, ist ausgeschlossen, soweit dies gesetzlich zulässig ist.

Diese Dokumentation enthält urheberrechtlich geschützte Informationen.

Alle Rechte, insbesondere das Recht auf Vervielfältigung und Verbreitung sowie der Übersetzung, bleiben vorbehalten. Kein Teil der Dokumentation darf in irgendeiner Form (durch Fotokopie, Mikrofilm oder ein anderes Verfahren) ohne vorherige schriftliche Zustimmung der SSD IT Consulting GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Copyright:

# **SSD IT Consulting GmbH**

**Gütschstrasse 11  
CH 6404 Greppen**

Telefon: +41 (0)41 390 31 92  
FAX: +41 (0)41 390 31 93  
Natel: +41 (0)79 286 54 13  
EMail: Support@ssd-it.ch

### Drucklegende

Version A.01.00	September 1989
Version A.01.01	September 1991
Version A.01.02	Februar 1993
Version A.02.07	Juni 1994
Version A.03.00	Mai 2001

## 1. Aufgaben des SD/3000 Cobol Analysers

Der COBOL Analyser erstellt für bestehende COBOL-Programme die Modulköpfe durch automatische Analyse der Programme. Der COBOL Analyser arbeitet analog dem Compiler. Er kann daher auch nur syntaktisch richtige Programme analysieren.

Der COBOL Analyser ist ein Zusatzprodukt zu SD/3000. Es erfordert mindestens die SD/3000-Version A.02.03. Der Grund dafür liegt darin, daß mit dieser Version der Aufbau der Köpfe erweitert wurde. Für alle Dateireferenzen sind nicht nur der Referenztyp I oder O möglich, sondern die Art des Zugriffs (z.B. P für PUT oder D für DELETE). Außerdem können ab der Version A.02.03 Referenzen auf Masken und COPYLIB-Member dokumentiert werden.

Bei der Analyse werden die folgenden Programmelemente erkannt:

- COPYLIB Referenzierungen
- Standard COBOL I/O Aufrufe
- COBOL SORT/MERGE Aufrufe
- COBOL KSAM Intrinsic Aufrufe (CK-Routinen)
- TurboIMAGE Intrinsic Aufrufe
- FileSystem Intrinsic Aufrufe
- SQL Befehle
- V/3000 Intrinsic Aufrufe
- bestimmte System-Intrinsics (z.B. CREATEPROCESS)
- eigene Datei Zugriffsroutinen
- usw.

## 2. Aufruf des COBOL Analysers

### 2.1 Analyse von Moduln über SD-COMPILE Eintrag

Mit dieser Analysefunktion wird ein bereits in SD/3000 eingetragenes Modul analysiert. Über den vorhandenen Eintrag in der Übersetzungsmaske wird der Name der Sourcedatei sowie der Name der zugehörigen Standard-Copylib ermittelt. Das Ergebnis der Analyse (der Modulkopf) ist eine temporäre Datei mit dem Namen SD. Diese Datei kann anschließend vom Programmierer ergänzt / korrigiert werden und mit einem beliebigen Editor in das Programm kopiert werden.

Der Analyser wird mit dem folgenden UDC aufgerufen:

```
:SDCOBANM {modul} [ , {proj} [ , {opt1} , {opt2}, . . . ] ]
```

- |                |  |
|----------------|--|
| <b>{modul}</b> | Das zu analysierende Modul.  |
| <b>{proj}</b>  | Das Projekt, zu dem das Modul gehört. Voreinstellung ist *.<br>Es wird die Voreinstellung aus der Datenbank benutzt.                         |
| <b>{optn}</b>  | In beliebiger Reihenfolge bis zu 4 Optionen, die durch einen Buchstaben gekennzeichnet werden. Die Optionen sind im Kapitel 2.4 beschrieben. |

## 2.2 Analyse von Programmen über Dateinamen

Mit dieser Analysefunktion kann ein bestehendes COBOL-Programm analysiert werden ohne daß ein Eintrag in SD/3000 vorhanden ist. Das Ergebnis der Analyse (der Modulkopf) ist eine temporäre Datei mit dem Namen SD. Diese Datei kann anschließend vom Programmierer ergänzt / korrigiert werden und mit einem beliebigen Editor in das Programm kopiert werden.

Diese Version des Analysers wird mit dem folgenden UDC aufgerufen:

**:SDCOBANF {datei} [ , {opt1} , {opt2} , ... ]**

**{datei}** MPE-Dateiname der zu analysierenden COBOL-Source.

**{optn}** In beliebiger Reihenfolge bis zu 4 Optionen, die durch einen Buchstaben gekennzeichnet werden. Die Optionen sind im Kapitel 2.4 beschrieben.

Da bei diesem Analyser-Aufruf kein Zugriff auf SD/3000 erfolgen kann, ist es erforderlich, evtl. benutzte COPYLIB's, soweit sie nicht im Programm explizit angesprochen werden, vor dem Aufruf mit dem folgenden FILE-Befehl zuzuweisen:

**:FILE COPYLIB={name der standard copylib}**

## 2.3 Analyse von Programmen über Dateinamen + Update

Mit dieser Analysefunktion kann ein bestehendes COBOL-Programm analysiert werden ohne daß ein Eintrag in SD/3000 vorhanden ist. Gleichzeitig wird das Ergebnis der Analyse automatisch mit Hilfe des EDITOR's in die vorhandene Programmdatei kopiert.

Diese Version des Analysers wird mit dem folgenden UDC aufgerufen:

**:SDCOBANX {datei} [ , {opt1} , {opt2} , ... ]**

**{datei}** MPE-Dateiname der zu analysierenden COBOL-Source.  
Wenn die Datei comprimiert/squished/qedified ist, wird sie vorher in eine normale ASCII-Datei verwandelt und nach der Analyse wieder entsprechend zurückverwandelt.

**{optn}** In beliebiger Reihenfolge bis zu 4 Optionen, die durch einen Buchstaben gekennzeichnet werden. Die Optionen sind im Kapitel 2.4 beschrieben.

Da bei diesem Analyser-Aufruf kein Zugriff auf SD/3000 erfolgen kann, ist es erforderlich, evtl. benutzte COPYLIB's, soweit sie nicht im Programm explizit angesprochen werden, vor dem Aufruf mit dem folgenden FILE-Befehl zuzuweisen:

**:FILE COPYLIB={name der standard copylib}**

## 2.4 Optionen beim Aufruf des Analysers

Die verschiedenen Optionen des Analysers beeinflussen nicht die eigentliche Analyse sondern geben nur zusätzliche Informationen für den Anwender bzw. zur Analyse von Fehlern.

### **Get Trace Option**

Interner GETTOKEN Trace

### **Kopf Option**

Der erzeugte Modulkopf wird über den File SDLISTE ausgegeben.

### **List Option**

Das Sourceprogramm wird während der Analyse über den File SDLISTE ausgegeben.

### **Routinen Trace Option**

Interner ROUTINEN Trace

### **Table Option**

Die internen Tabellen des Analysers werden nach der ersten und zweiten Phase auf den File SDTABLE ausgegeben.

### 3. Analysierbare COBOL Sprachelemente

#### 3.1 Allgemeine HP-COBOL und Preprocessor-Befehle

##### **COPY**

Ist im COPY-Befehl eine COPYLIB spezifiziert, so wird diese COPYLIB eröffnet. Wurde die COPYLIB nicht im COPY-Befehl angegeben, so wird die Standard-Copylib gemäß SD/3000 Eintrag bzw. bei Aufruf über SDCOBANF der File COPYLIB eröffnet. Alle verwendeten COPYLIB's werden vom Analyser nur einmal eröffnet und nach Abschluß der ersten Phase wieder geschlossen. Das Member wird für die 2. Phase des Analysers zwischengespeichert und es wird eine COPYLIB-MEMBER-Referenz für den Modulkopf generiert.

##### **AUTHOR**

Es werden die nachfolgenden Zeichen (maximal 20) bis zum Auftreten des Punktes gespeichert und später als Autor in den Modulkopf generiert.

##### **\$TITLE**

Es werden bis zu 2 Titelzeilen mit jeweils bis zu 80 Zeichen erkannt und später als Inhaltsangabe in den Modulkopf generiert.

##### **PROGRAM-ID**

Der Programmname wird als Modulname in den Modulkopf übernommen.

##### **PROCEDURE DIVISION**

Folgt in diesem Befehl ein **USING**-Teil, so wird das Programm als Unterprogramm erkannt. Entsprechend wird im Modulkopf als Aufruf **CALL** generiert, im anderen Fall **RUN**.

##### **MOVE**

Der COBOL Analyser führt intern während der gesamten Analysephase 1 einen sogenannten MOVESTACK. In diesem Stack werden bestimmte COBOL-MOVE-Befehle zwischengespeichert und können z.B. für die dynamische Zuweisung von Dateinamen erkannt werden. Der MOVESTACK wird für alle Dateinamen (FileSystem, TurboIMAGE, sonstige Intrinsic) sowie Maskennamen verwendet.

## 3.2 Standard COBOL I/O

### **SELECT**

Über den SELECT-Befehl wird die Verbindung zwischen dem COBOL internen Dateinamen und dem echten MPE-Dateinamen/Dateinamen hergestellt. Der Analyser merkt sich daher beide Namen für die nachfolgende Verarbeitung. Im ASSIGN-Teil sind MPE-Dateinamen und Systemdateien (z.B. \$STDLIST) zulässig. Eventuell eingetragene Datei-Kennworte werden extrahiert.

### **FD/SD Section**

In diesen Sections können zu jeder Datei ein oder mehrere Datensätze definiert werden. Der Analyser merkt sich dazu alle Satzdefinitionen auf der **01**-Stufe. Dies gilt sowohl für die FD als auch die SD Section.

### **READ**

Es wird für die Datei eine GET-Referenz erzeugt.

### **OPEN**

Es wird für die Datei eine OPEN-Referenz erzeugt.

### **DELETE**

Es wird für die Datei eine DELETE-Referenz erzeugt.

### **REWRITE**

Es wird für die Datei eine UPDATE-Referenz erzeugt.

### **EXCLUSIVE**

Es wird für die Datei eine LOCK-Referenz erzeugt.

### **ACCEPT**

Wenn im Befehl ein FROM CONSOLE folgt, wird eine GET-Referenz für die Datei \$CONSOLE generiert. Im anderen Fall eine GET-Referenz für \$STDIN.

### **DISPLAY**

Wenn im Befehl ein UPON CONSOLE folgt, wird eine PUT-Referenz für die Datei \$CONSOLE generiert. Im anderen Fall eine PUT-Referenz für \$STDLIST.

## 3.2 COBOL KSAM I/O Routinen (CK-Routinen)

Bei Nutzung dieser KSAM I/O Routinen wird im Programm ein spezieller Kontrollblock definiert. Dieser Kontrollblock enthält den echten Dateinamen der MPE-Datei. Der Analyser ermittelt diesen Namen in der 2. Phase. Wird der Kontrollblock während des Programmlaufs modifiziert, kann der Analyser den Namen der Datei nicht ermitteln.

### **CKDELETE**

Es wird eine DELETE-Referenz generiert.

### **CKLOCK**

Es wird eine LOCK-Referenz generiert.

### **CKOPEN/CKOPENSHR**

Für beide Routinen wird eine OPEN-Referenz generiert.

### **CKREAD/CKREADBYKEY**

Für beide Routinen wird eine GET-Referenz generiert.

### **CKREWRITE**

Es wird eine UPDATE-Referenz generiert.

### **CKSTART**

Es wird eine CONTROL-Referenz generiert.

### **CKWRITE**

Es wird eine PUT-Referenz generiert.

## 3.3 TurboIMAGE Intrinsic

Referenzen für IMAGE werden teilweise für die gesamte Datenbank als auch für Datasets gebildet. Dynamische Zuweisungen über den MOVESTACK sind möglich.

### **DBCNTROL**

Für die Datenbank wird eine CONTROL-Referenz generiert.

### **DBDELETE**

Für das Dataset wird eine DELETE-Referenz generiert.

### **DBFIND**

Für das Dataset wird eine CONTROL-Referenz generiert.

### **DBGET**

Für das Dataset wird eine GET-Referenz generiert.

### **DBINFO**

Für die Datenbank wird eine INFO-Referenz generiert.

### **DBLOCK**

Für die Datenbank wird eine LOCK-Referenz generiert.

### **DBOPEN**

Für die Datenbank wird eine OPEN-Referenz generiert.

### **DBUPDATE**

Für das Dataset wird eine UPDATE-Referenz generiert.

### 3.4 FileSystem Intrinsic

Beim Arbeiten mit FileSystem-Intrinsic wird in der Regel nur eine Filenummer verwendet. Diese Nummer wird beim FOPEN-Aufruf ermittelt. Wenn diese Nummer in verschiedenen Variablen verwendet wird, so kann der Analyser keine Referenzen auf den Dateinamen finden. Wird jedoch immer eine Filenummer-Variable für eine Datei verwendet, so kann der Analyser alle Referenzen sauber zuordnen. Bei den Intrinsic, in denen ein Dateiname auftritt, wird der MOVESTACK unterstützt.

#### **FCONTROL / FDEVICECONTROL / FPOINT / FSETMODE / FSPACE**

Über die Filenummer wird eine CONTROL-Referenz generiert.

#### **FDELETE / FREMOVE**

Über die Filenummer wird eine DELETE-Referenz generiert.

#### **FFILEINFO / FGETINFO / FGETKEYINFO**

Über die Filenummer wird eine INFO-Referenz generiert.

#### **FFINDBYKEY / FFINDN**

Über die Filenummer wird eine CONTROL-Referenz generiert.

#### **FLABELINFO**

Für die angegebene Datei wird eine INFO-Referenz generiert.

#### **FLOCK**

Über die Filenummer wird eine LOCK-Referenz generiert.

#### **FOPEN**

Für die angegebene Datei wird eine OPEN-Referenz generiert. Gleichzeitig wird die Filenummer zwischengespeichert um spätere Referenzen hierüber auf den Dateinamen zurückzuführen zu können.

#### **FREAD / FREADBACKWARD / FREADBYKEY / FREADC / FREADDIR / FREADLABEL / FREADSEEK**

Über die Filenummer wird eine GET-Referenz generiert.

#### **FRENAME**

Über die Filenummer wird eine CONTROL-Referenz generiert. Zusätzlich wird auch für den neuen Dateinamen eine CONTROL-Referenz generiert.

#### **FUPDATE**

Über die Filenummer wird eine UPDATE-Referenz generiert.

#### **FWRITE / FWRITEDIR / FWRITELABEL**

Über die Filenummer wird eine PUT-Referenz generiert.

### 3.5 HP-SQL

#### **CONNECT**

Für das DBEnvironment wird eine OPEN-Referenz generiert.

#### **DECLARE CURSOR / SELECT**

Für die angesprochene(n) TABLE(s) wird eine GET-Referenz generiert.

#### **DELETE**

Für die TABLE wird eine DELETE-Referenz generiert.

#### **INSERT**

Für die TABLE wird eine PUT-Referenz generiert.

#### **LOCK**

Für die TABLE wird eine LOCK-Referenz generiert.

#### **UPDATE**

Für die TABLE wird eine UPDATE-Referenz generiert.

### 3.6 V/3000

#### **VOPENFORMF**

Es wird der Dateiname der Maskendatei ermittelt und für die spätere Verwendung zwischengespeichert.

#### **VGETNEXTFORM**

Es wird zunächst versucht den Maskennamen zu ermitteln. Dazu wird im MOVESTACK nach einer Variablen mit einer der folgenden Buchstabenfolgen im Namen gesucht:

NFNAM  
NEXTF  
NEXT-F

Wird ein derartiger Name gefunden und handelt es sich um einen gültigen Maskennamen, so wird eine Maskenreferenz gebildet.

### 3.7 sonstige Intrinsic

#### **CATOPEN**

Für den verwendeten CATALOG wird eine GET-Dateireferenz generiert.

#### **CREATE / CREATEPROCESS / LOADPROC**

Für die angesprochenen Moduln wird eine externe Referenz generiert.

#### **PRINT**

Es wird eine PUT-Referenz für \$STDLIST generiert.

#### **PRINTOP**

Es wird eine PUT-Referenz für \$CONSOLE generiert.

#### **PRINTOPREPLY**

Es wird eine PUT- und GET-Referenz für \$CONSOLE generiert.

#### **READ**

Es wird eine GET-Referenz für \$STDIN generiert.

#### **READX**

Es wird eine GET-Referenz für \$STDINX generiert.

### 3.8 Eigene Datei Zugriffsroutinen

Der Analyser erlaubt bis zu 10 Benutzer-Zugriffsroutinen zu definieren, die mit CALL aufgerufen werden. Diese Routinen werden über das Standard-Verwaltungsprogramm SDMAINT (Maske 22) definiert. Eine detaillierte Beschreibung dazu befindet sich in der allgemeinen Bedienungsanleitung SD/3000 im Kapitel 2.22.

Bei Verwendung dieser Zugriffsroutinen wird analog zu den anderen Dateireferenzierungen eine Ermittlung des Dateinamens (evtl. über MOVESTACK) durchgeführt. Die Referenzierung erfolgt mit dem in der Maske definierten Kennbuchstaben.

## 4 Limitationen und Fehler

Damit der COBOL Analyser performant läuft, sind eine Reihe von Limitationen die in der Praxis jedoch nicht zu Problemen führen vorhanden.

Anzahl externer Referenzen	100
Anzahl externer Dateien gesamt	300
Anzahl FS-Dateien	100
Anzahl COBOL-Dateien	100
Anzahl KSAM-Dateien	100
Anzahl COPYLIB's	100
Anzahl COPYLIB Member	300
Anzahl V/3000 Maskendateien	50
Anzahl V/3000 Masken	100
Länge Variablenamen	60

Geschachtelte COPY-Befehle sind nicht erlaubt.

CALL's mit Variablenamen sind nicht erlaubt.

Wenn Variablenreferenzen innerhalb des Programms nicht aufgelöst werden können, so wird in den Modulkopf der Variablenname mit einem vorgestellten \* generiert. Dies ist z.B. der Fall, wenn an Unterprogramme Dateinummern oder Variablen mit Dateinamen übergeben werden. Damit hat der Programmierer die Möglichkeit, diese Dinge leicht zu korrigieren.

## Inhaltsverzeichnis

1	Aufgaben des SD/3000 Cobol Analysers .....	3
2	Aufruf des COBOL Analysers .....	3
2.1	Analyse von Moduln über SD-COMPILE Eintrag .....	3
2.2	Analyse von Programmen über Dateinamen .....	4
2.3	Analyse von Programmen über Dateinamen + Update .....	4
2.4	Optionen beim Aufruf des Analysers .....	5
3	Analysierbare COBOL Sprachelemente .....	6
3.1	Allgemeine HP-COBOL und Preprocessor-Befehle .....	6
3.2	Standard COBOL I/O .....	7
3.2	COBOL KSAM I/O Routinen (CK-Routinen) .....	8
3.3	TurboIMAGE Intrinsic .....	8
3.4	FileSystem Intrinsic .....	9
3.5	HP-SQL .....	11
3.6	V/3000 .....	11
3.7	sonstige Intrinsic .....	12
3.8	Eigene Datei Zugriffsroutinen .....	12
4	Limitationen und Fehler .....	13

**Index**

\$	
\$CONSOLE.....	5
\$STDLIST.....	5
\$TITLE.....	4
<	
<datei>.....	1
<modul>.....	1
<optn>.....	1; 2
<proj>.....	1
<b>A</b>	
ACCEPT.....	5
ASSIGN.....	5
AUTHOR.....	4
<b>C</b>	
CATOPEN.....	10
CKDELETE.....	6
CKLOCK.....	6
CKOPEN.....	6
CKOPENSHR.....	6
CKREAD.....	6
CKREADBYKEY.....	6
CKREWRITE.....	6
CKSTART.....	6
CKWRITE.....	6
CONNECT.....	9
COPY.....	4
COPYLIB.....	2
CREATE.....	10
CREATEPROCESS.....	10
<b>D</b>	
DBCNTROL.....	6
DBDELETE.....	6
DBFIND.....	6
DBGGET.....	6
DBINFO.....	6
DBLOCK.....	6
DBOPEN.....	6
DBUPDATE.....	7
DECLARE.....	9
DELETE.....	5; 9
DISPLAY.....	5
<b>E</b>	
EXCLUSIVE.....	5

**Index**

<b>F</b>	
FCONTROL .....	7
FD .....	5
FDELETE .....	7
FDEVICECONTROL.....	7
FFILEINFO .....	7
FFINDBYKEY .....	7
FFINDN.....	7
FGETINFO.....	7
FGETKEYINFO.....	7
FILE COPYLIB.....	2
FLABELINFO .....	7
FLOCK .....	7
FOPEN .....	7
FPOINT .....	7
FREAD .....	7
FREADBACKWARD .....	7
FREADBYKEY .....	7
FREADC .....	7
FREADDIR .....	7
FREADLABEL .....	7
FREADSEEK .....	7
FREMOVE .....	7
FRENAME .....	8
FSETMODE .....	7
FSPACE.....	7
FUPDATE .....	8
FWRITE .....	8
FWRITEDIR .....	8
FWRITELABEL.....	8
<b>G</b>	
Get Trace Option .....	3
<b>I</b>	
INSERT .....	9
<b>K</b>	
Kopf Option.....	3
<b>L</b>	
List Option .....	3
LOADPROC.....	10
LOCK .....	9
<b>M</b>	
MOVE .....	4
<b>O</b>	
OPEN.....	5

**Index****P**

PRINT .....	10
PRINTOP .....	10
PRINTOPREPLY .....	10
PROCEDURE DIVISION .....	4
PROGRAM-ID .....	4

**R**

READ .....	5; 10
READX .....	10
REWRITE .....	5
Routinen Trace Option .....	3

**S**

SD .....	5
SDCOBANF .....	2
SDCOBANM .....	1
SDCOBANX .....	2
SDLISTE .....	3
SDTABLE .....	3
SELECT .....	5; 9

**T**

Table Option .....	3
--------------------	---

**U**

UPDATE .....	9
--------------	---

**V**

VGETNEXTFORM .....	9
VOPENFORMF .....	9