



SOFiSTiK

Basisfunktionalitäten

SOFiSTiK | 2018

SOFiSTiK
Basisfunktionalitäten

SOFiSTiK Manual, Version 2018-2
Software Version SOFiSTiK 2018

Copyright © 2017 by SOFiSTiK AG, Oberschleissheim, Germany.

SOFiSTiK AG

Hauptsitz Oberschleissheim

Bruckmannring 38

85764 Oberschleissheim

Deutschland

T +49 (0)89 315878-0

F +49 (0)89 315878-23

Niederlassung Nürnberg

Burgschmietstr. 40

90419 Nürnberg

Deutschland

T +49 (0)911 39901-0

F +49(0)911 397904

info@sofistik.de
www.sofistik.de

Dieses Handbuch ist urheberrechtlich geschützt. Kein Teil darf ohne schriftliche Genehmigung der SOFiSTiK AG in irgendeiner Weise vervielfältigt, übersetzt oder umgeschrieben werden. SOFiSTiK behält sich das Recht vor, diese Veröffentlichung jederzeit zu überarbeiten oder inhaltlich zu ändern.

SOFiSTiK versichert, dass Handbuch und Programm nach bestem Wissen und Gewissen erstellt wurden, übernimmt jedoch keine Gewähr dafür, dass Handbuch oder Programm fehlerfrei sind. Fehler oder Unzulänglichkeiten werden nach Bekanntwerden in der Regel aber beseitigt.

Der Benutzer bleibt für seine Anwendungen selber verantwortlich. Er hat sich durch Stichproben von der Richtigkeit seiner Berechnungen zu überzeugen.

Titelseite

Projekt: Neubau SOFiSTiK Bürogebäude, Nürnberg | Generalübernehmer: WOLFF & MÜLLER, Stuttgart | Architektur: WABE-PLAN ARCHITEKTUR, Stuttgart | Tragwerksplanung: Boll und Partner. Beratende Ingenieure VBI, Stuttgart | Haustechnik: GM Planen + Beraten, Griesheim | Entwurf: Gerhard P. Wirth gpwirtharchitekten, Nürnberg | Visualisierung: Armin Dariz, BiMOTION GmbH

Inhaltsverzeichnis

Inhaltsverzeichnis	i
1 Einführung	1-1
1.1 SOFiSTiK	1-1
1.2 Beispiele für die Anwendung der Software	1-1
2 Wie funktioniert SOFiSTiK ?	2-1
2.1 Überblick	2-1
2.2 SOFiSTiK FEA Definitionen und Grenzen	2-3
2.2.1 Datenbasis	2-3
2.2.2 Querschnitte und Materialien	2-3
2.2.3 Vernetzung und Modellierung	2-3
2.2.4 Berechnung und Bemessung	2-4
2.2.5 Standard-Lastfallnummerierungen	2-4
2.3 Programme	2-5
2.3.1 Eingabeprogramme	2-5
2.3.2 Rechenprogramme	2-5
2.3.3 Bemessungsprogramme	2-5
2.3.4 Auswertungsprogramme	2-5
2.4 User Interface	2-6
2.5 Eingabedateien	2-7
2.6 Datenbasis	2-7
2.7 Interaktive Module	2-9
2.8 Neuigkeiten	2-9
2.9 Namenskonventionen der SOFiSTiK Dateien	2-10
2.10 Datensicherung	2-11
2.11 SOFiSTiK Optionen	2-12
2.11.1 Einstellungen der Sprache	2-12
2.11.2 Einheiten	2-12
3 Schnittstellen	3-1
3.1 CDB Export nach Excel	3-1
3.2 Export nach DAT	3-1
3.3 IFC Import und Export	3-1
3.4 Extensions für Autodesk® Revit®	3-2
3.5 SOFiSTiK Rhinoceros Interface	3-2
4 Konvertierung von Projekten (Statik)	4-1
4.1 Vorbemerkung	4-1
4.2 Konvertierung von Projektdateien	4-1
4.2.1 SSD Projektdateien (.sofistik)	4-1

4.2.2	SOFIPLUS(-X) Projektdateien (.dwg)	4-1
4.2.3	Datenbank (.cdb)	4-2
4.2.4	CADiNP Eingaben	4-2
4.3	Änderungen	4-3
4.3.1	Entfallene Normen	4-3
5	SSD - SOFiSTiK Structural Desktop	5-1
5.1	Benutzeroberfläche des SSD	5-1
5.2	Arbeitsweise	5-1
5.2.1	Gruppen	5-1
5.2.2	Tasks	5-1
5.2.3	Vorlagendateien <i>name.sofistik</i>	5-2
5.3	Aufbau und Funktionsweise	5-4
5.3.1	Berechnungsstatus	5-4
5.4	Zusätzliche Aufrufmöglichkeiten	5-4
6	Grafische Eingabe SOFiPLUS(-X)	6-1
6.1	Allgemeine Hinweise	6-1
6.2	Arbeitsweise	6-1
6.2.1	Start aus dem Structural Desktop (SSD)	6-1
6.3	Aufbau	6-2
7	Eingabe mit TEDDY	7-1
7.1	Allgemeines	7-1
7.2	Starten von TEDDY	7-1
7.3	Wichtige Befehle	7-2
7.3.1	Kombinationen mit der Alt-Taste (Auswahl)	7-2
7.3.2	Kombinationen mit der Strg-Taste (Auswahl)	7-3
7.3.3	Übersicht über alle Strg- / Alt- Befehle	7-3
7.3.4	Beispiel mit Spaltenblock	7-3
7.4	Eingabe von Daten	7-3
7.5	Hilfe und interaktives Handbuch	7-4
7.5.1	Aktivierung der Hilfe	7-4
7.6	TEDDY als Kommandozentrale	7-5
7.7	TEDDY- Kapitel und TEDDY-Label	7-6
8	CADiNP - Eingabesprache	8-1
8.1	Definitionen und Bezeichnungen	8-1
8.1.1	Zeilen (Physikalische Sätze)	8-1
8.1.2	Sätze (Logische Sätze)	8-1
8.1.3	Trennzeichen	8-2
8.1.4	Datenwerte	8-2
8.2	Eingabesyntax	8-3
8.2.1	Grundsätzliche Eingabeform	8-3
8.2.2	Standardwert	8-3
8.2.3	Wiederholung	8-3
8.2.4	Inkrementierung/Dekrementierung	8-3
8.2.5	Kommentar	8-4
8.2.6	Satzfortsetzung	8-4
8.2.7	Satztrennung	8-4

8.2.8	Positionierung	8-4
8.2.9	Tabellendefinition	8-4
8.2.10	Help-Satz	8-5
8.2.11	Generierung	8-5
8.2.12	Wertreihe	8-6
8.2.13	Einheiten Konvertierung	8-7
8.2.14	LET - und STO - Variable	8-7
8.2.15	Arithmetische Ausdrücke	8-10
8.2.16	FUN - Definition von Funktionen	8-12
8.2.17	LOOP, ENDLOOP - Schleifen und Sprünge	8-13
8.2.18	IF - Logische Abfragen	8-14
8.2.19	@CDB - Auswahl einer CDBASE	8-16
8.2.20	@KEY - Zugriff auf CDBASE	8-16
8.2.21	@() - Zugriff auf CDBASE	8-16
8.3	Generelle Satznamen	8-18
8.3.1	KOPF – Überschriftenzeilen	8-20
8.3.2	ENDE – Ende Eingabeblock	8-20
8.3.3	TXA – Vorbemerkungen	8-20
8.3.4	TXE – Nachbemerkungen	8-21
8.3.5	<TEXT> – Textblock	8-21
8.3.6	</TEXT> – Ende eines Textblocks	8-22
8.3.7	ECHO – Ausgabeumfang	8-23
8.3.8	UNIT – Einheiten für Ein- und Ausgabe	8-24
8.3.9	SEIT – Ein- und Ausgabeformat	8-24
8.3.10	SIZE – Grafisches Format	8-27
8.4	Erzeugen von Bildern	8-30
8.4.1	<PICT> – Start eines Bildes	8-30
8.4.2	GNT – Skalierung des Bildes	8-30
8.4.3	GPL – Polygonzug	8-31
8.4.4	GPM – Polymarker	8-31
8.4.5	GFA – Füllfläche	8-31
8.4.6	GGDP – Allgemeines Grafikelement	8-32
8.4.7	GTXI – Beschriftung	8-33
8.4.8	GSCA – Vermaßung	8-33
8.5	Attribute grafischer Darstellungen	8-34
8.5.1	GCOL – Farbauswahl	8-34
8.5.2	GPLI – Polyline Attribute	8-35
8.5.3	GPMI – Polymark Attribute	8-36
8.5.4	GTXI – Text Attribute	8-37
8.5.5	GFAI – Fill Area Attribute	8-37
8.6	Einfügen von Bildern	8-39
8.6.1	<LINK> – Einfügen eines Bildes	8-39
8.7	Parametrisierte Eingaben	8-39
8.8	Kompatibilität von Datensätzen	8-41
9	DEF - Umgebungsvariablen	9-1
10	Start der Berechnung	10-1
10.1	Allgemeines	10-1
10.2	#DEFINE - Parametersubstitution	10-2

10.3	#INCLUDE - Blockbildung	10-3
10.4	APPLY - Einbinden von Daten während der Berechnung	10-4
10.5	#IF - Bedingte Eingaben	10-4
10.6	Templates	10-5
10.7	Iteration über mehrere Module	10-6
10.8	Betriebssystem Kommandos	10-7
10.9	Job-Geschichte	10-8
10.10	Start eines Einzelprogramms	10-8
10.11	Zusätzliche Aufrufmöglichkeiten	10-9
11	Ausgabe	11-1
11.1	Ergebnis- Ausgabe mit dem Report Browser	11-1
11.1.1	Allgemeines	11-1
11.1.2	Aktivierung von Report Browser	11-1
11.1.3	Features	11-1
11.1.4	Tabellenausgabe	11-3
11.1.5	Individuelle Einstellungen des Report Browsers	11-3
11.1.6	Drucken- Dialog	11-3
11.1.7	Erweiterte Funktionen	11-4
11.1.8	Funktionen im Navigationsbaum	11-4
11.1.9	Zusätzliche Aufrufmöglichkeiten	11-5
11.2	Protokoll-Datei (.prt)	11-7
11.3	Grafische Ausgabe Result Viewer	11-7
11.3.1	Allgemeine Hinweise	11-7
12	Problembehandlung	12-1
12.1	Allgemeines	12-1
12.2	Ordentliche Warnungen bzw. Fehlermeldungen	12-1
12.3	Fehlerstrategien	12-2
12.3.1	Probleme mit der Datenbasis (*.cdb)	12-2
12.3.2	Eingabefehler im TEDDY Datensatz	12-2
12.3.3	Fehler bei der Systemgenerierung in SOFiPLUS	12-2
12.3.4	Fehler bei der Berechnung	12-3
12.3.5	Probleme bei den Bemessungsergebnissen	12-3
12.4	Support	12-4
12.4.1	Erreichbarkeit SOFiSTiK Support	12-4
12.4.2	Mithilfe des Kunden	12-5
12.4.3	Supportanfrage via SOFiSTiK Online Portal	12-5
12.4.4	Supportanfrage aus SSD / TEDDY	12-6
12.4.5	Erstellung der Diagnose.xml Datei	12-6
13	Weiterführende Hilfen	13-1
13.1	Administration Handbuch	13-1
13.2	VERiFiCATiON Manual	13-1
13.3	Tutorials	13-1
13.4	YouTube - Schulungs- und Präsentations Videos	13-1
13.5	Infoportal	13-1
13.6	CADINP Beispielsammlung	13-2
13.7	Forum	13-2

1 Einführung

1.1 SOFiSTiK

Die SOFiSTiK AG entwickelt und vertreibt Software für den Bereich des Ingenieurbaus mit besonderem Schwerpunkt für den konstruktiven Ingenieurbau.

Die Software ist unter einem einheitlichen Konzept entwickelt worden, die es dem normalen Anwender erlaubt, schnell zu einer Lösung gelangen zu können, ohne dass dem Spezialisten mit großer Erfahrung in numerischen Methoden unnötige Schranken auferlegt werden.

Ein besonderes Kennzeichen der Software ist ihr modularer Aufbau und offene Schnittstellen. Dies wird vor allem dadurch erreicht, dass die Rechenkerne klassische Batch-Programme sind, die mit einer kleinen Eingabedatei angesteuert werden und dabei vollen Zugriff auf eine Datenbasis haben.

Für jedes Programmmodul steht ein eigenes Handbuch zur Verfügung. Die Handbücher enthalten Angaben über die theoretischen Hintergründe und Beschreibungen zur Ein- und Ausgabe der Programme. Wir empfehlen, sich vor Anwendung des jeweiligen Programmmoduls mit dem entsprechenden Handbuch vertraut zu machen.

1.2 Beispiele für die Anwendung der Software

Für jedes Programm stehen einführende Beispiele zur Verfügung. Die Eingabedateien zu diesen Beispielen sind im Installationsverzeichnis SOFiSTiK in den jeweiligen Unterordnern *programmname* ⇒ deutsch zu finden.

Auf der Homepage der SOFiSTiK AG unter <http://www.sofistik.de/Infoportal>

haben Anwender und Interessenten die Möglichkeit, sich über die Anwendungsmöglichkeiten der SOFiSTiK-Software anhand von ausführlichen Beispielen zu informieren. Die Datensätze können als Muster für eigene Aufgabenstellungen verwendet werden.

Mehr Informationen über Beispiele und Hilfen zu den Programmen finden Sie in diesem Handbuch im Kapitel: 13 [Weiterführende Hilfen](#).

2 Wie funktioniert SOFiSTiK ?

2.1 Überblick

Im Zentrum der SOFiSTiK-Statikprogramme steht eine Datenbasis (CDB). Eine Vielzahl von Programmen, die über eine Eingabedatei oder direkt mit einem grafischen User-Interface angesprochen werden, tauschen ihre Daten ausschließlich über die Datenbasis.

Die SOFiSTiK-Software setzt sich aus zahlreichen Modulen zusammen. Für den Einstieg ist der "SOFiSTiK-Structural-Desktop" SSD geeignet, da er viele Abhängigkeiten der Module komfortabel verwaltet. Dafür existiert ein eigenes Tutorial, eine kurze Einführung finden Sie im Kapitel 5 : [SSD - SOFiSTiK Structural Desktop](#).

Um die volle Leistungsfähigkeit der Software ausnutzen zu können, ist es sinnvoll, die modulare Struktur und die CADiNP-Texteingabe als die grundlegendste Zugangsmethode zu verstehen, auf der alle anderen Eingabevarianten prinzipiell aufbauen. Die Flexibilität der CADiNP-Eingabe stellt für den eingearbeiteten Benutzer eine attraktive, unverzichtbare Möglichkeit dar, insbesondere weil damit alle Nebeninformationen in Form von Kommentaren und Formeln mit erfasst werden können.

Es ergibt sich das folgende Organisations- und Bedienungsschema:

- Erzeugen von Eingabedateien oder Datenbasen manuell oder mit Generierungsprogrammen
- Rechnen von Eingabedateien
- Ausgabe von Ergebnislisten und Grafiken

Das Verfahren ist nicht an irgendwelche starren Masken oder Abläufe gebunden. Dies macht die Software sehr flexibel. Der Benutzer kann jederzeit die ihm effektivste Form der Eingabe wählen und ist in der Reihenfolge der Berechnungsschritte nur geringen Zwängen unterworfen. Außerdem kann man ohne Probleme Datensätze zwischen verschiedenen Rechnern und Betriebssystemen austauschen.



Abbildung 2.1: Modulübersicht

2.2 SOFiSTiK FEA Definitionen und Grenzen

In den folgenden Unterkapiteln werden die Grenzen und besondere Definitionen der Software beschrieben. Zu beachten ist, dass es sich an dieser Stelle lediglich um einen Überblick handelt. Detailliertere oder alternative Spezifikationen können den Handbüchern der entsprechenden Programmmodule entnommen werden.

2.2.1 Datenbasis

Beschreibung	Max. Größe
Max. Größe der Datenbasis CDBASEVER=501 (siehe Kapitel 9) (Default)	256 GB
Max. Größe der Datenbasis CDBASEVER=503 (siehe Kapitel 9)	1024 GB

Tabelle 2.1: Grenzen der Datenbasisgröße

2.2.2 Querschnitte und Materialien

Beschreibung	Max. Anzahl	ID-Bereich
Materialien	999	1 - 999
Materialien je Querschnitt	31	
Querschnitte	9 999	1 - 9 999
Längsbewehrungsränge je Querschnitt	9	
Schubschnitte je Querschnitt	255	
Polygonpunkte je Polygon	255	
Bügelbewehrungsränge je Querschnitt	15	
Bauabschnitte je Querschnitt	10	1 - 9 999
Bauabschnitte je Querschnitt inklusive Vorspann-Abschnitte	99	
Bohrprofile	999	1 - 999

Tabelle 2.2: Grenzen Querschnitte und Materialien

2.2.3 Vernetzung und Modellierung

Beschreibung	Max. Anzahl	ID-Bereich
Strukturpunkte	99 999	1 - 99 999
Strukturlinien	99 999	1 - 99 999
Strukturflächen	99 999	1 - 99 999
Strukturvolumen	99 999	1 - 99 999
Primärgruppen	1 000	0 - 999

Tabelle 2.3: Grenzen Finite Elemente und geometrische Modellierung

2.2.4 Berechnung und Bemessung

Beschreibung	Max. Anzahl	ID-Bereich
Anzahl der möglichen Lastfälle	999 999	1 - 999 999
Anzahl der Bauabschnitte	9 999	1 - 9 999
Anzahl der Kombinationen	999	1 - 999

Tabelle 2.4: Grenzen Berechnung und Bemessung

Hinweis

Grundsätzlich gibt es keine strikte Obergrenze für z.B. Elementnummern. In den meisten Fällen können in der Ausgabe jedoch nur *maximal sieben Stellen* angezeigt werden.

2.2.5 Standard-Lastfallnummerierungen

Lastfälle	Beschreibung
LC 1 - 999	Einzellastfälle
LC 1 000 - 1099	Lastkombinationen (Theorie 2. Ordnung, nichtlineare Lastfälle,...)
LC 1 100 - 1 899	Ergebnislastfälle für Bemessung Grenzzustand der Gebrauchstauglichkeit
LC 1 900 - 1 999	Ergebnislastfälle für Bemessung Grenzzustand der Gebrauchstauglichkeit
LC 2 100 - 2 399	Ergebnislastfälle für Bemessung Grenzzustand der Tragfähigkeit
LC 3 970 - 3 999	CSM: Eingusszustandslastfälle (STEU guss)
LC 4 000 - 4 999	CSM: Summenlastfälle mit Gesamtverschiebungen und -schnittgrößen
LC 5 000 - 5 999	CSM: Differenzverschiebungen und -schnittgrößen
LC 6 000 - 6 999	CSM: AQB-Eigenspannungen aus Kriechen und Schwinden
LC 7 000 - 7 999	CSM: Ergebnisspannungen der AQB-LFSP-Auswertung
LC 10 001 - 10 999	Eigenwerte

Tabelle 2.5: Lastfallnummerierung für Bauzustände des Programms CSM

2.3 Programme

2.3.1 Eingabeprogramme

Interaktive Programme:

SOFIPLUS(-X)	Grafische Eingabe auf AutoCAD Basis (-X: Mit AutoCAD OEM Kern)
Querschnittseditor	Grafische Eingabe von Querschnitten unter AutoCAD® (Bestandteil von SOFIPLUS(-X))

Batch-Programme:

AQUA	Materialien und Querschnitte
SOFIMSHA	Import und Export Finiter Elemente und Stabwerke
SOFIMSHC	Geometrische Modellierung
SOFILOAD	Lasten und Lastfunktionen
TENDON	Geometrie der Spanngliedführung
CSM	Construction Stage Manager

2.3.2 Rechenprogramme

ASE	Allgemeine Statik Finiter Element Strukturen
HASE	Halbraumanalyse für statische Boden-Struktur-Interaktion
TALPA	2D Finite Elemente in der Geotechnik
DYNA	Dynamische Berechnungen
ELLA	Erweiterte Verkehrslast Auswertung
HYDRA	Grundwasser- und Wärmemodelle
STAR2	Statik der Stabtragwerke Theorie II. Ordnung
RELY	Structural Reliability powered by Strurel
DOLFYN	Fluid Dynamics powered by Cyclone Fluid Dynamics BV

2.3.3 Bemessungsprogramme

MAXIMA	Überlagerung
AQB	Bemessung von Querschnitten
BEMESS	Bemessung von Flächentragwerken
BDK	Stabilitätsnachweis für Stahlbauteile

2.3.4 Auswertungsprogramme

interaktive Programme:

Result Viewer	Grafisches und tabellarisches Postprocessing für Finite Elemente
WINGRAF	Grafische Darstellung Finiter Elemente und Stabwerke

Batch-Programme:

WING	Grafische Darstellung Finiter Elemente und Stabwerke
RESULTS	Grafisches und tabellarisches Postprocessing für Finite Elemente
DBPRIN	Drucken von in der Datenbasis gespeicherten Ergebnissen
SIR	Schnitte im Raum
DYNR	Grafische Ausgabe instationärer Berechnungen und Antwortspektren
TEXTILE	Zuschnitt von Membrantragwerken

2.4 User Interface

Nach der Installation der Software sind verschiedene Dateitypen mit SOFiSTiK-Programmen verknüpft worden. Deshalb ist unmittelbar über den Explorer ein Einstieg möglich.

Die im Explorer enthaltenen SOFiSTiK-Dateitypen haben folgende Bedeutungen:



Die Endungen **.sofistik** sind die Projektdateien des SSD (SOFiSTiK Structural Desktop).



Die Endungen **.dat** sind Eingabedateien. Ein Doppelklick öffnet das Programm TEDDY. Über die rechte Maus-taste kann außerdem das Programm WPS gestartet werden (run).



Die Endungen **.cdb** sind Datenbanken. Ein Doppelklick öffnet das Programm Animator. Über die rechte Maustaste sind außerdem verfügbar:

Datenbank Locks bereinigen	Entfernen von Locks der Datenbasis
Datenbank Informationen	Diagnose Tool für Datenbasis
Export nach DAT	SOFiSTiK Export
Grafische Ausgabe (WinGRAF)	Grafische Ausgaben WinGRAF
Post-Processing (Result Viewer)	Numerische Ausgaben Result Viewer



Die Endungen **.plb** sind Ausgabedateien, die auch Grafiken enthalten. Ein Doppelklick öffnet Report Browser.



Die Endungen **.prt** sind Protokolldateien der Gesamtberechnungen. Sie enthalten Informationen über den gesamten Berechnungsverlauf, wie Fehlermeldungen, Warnungen, Parameter des Gleichungssystems, Iterationsverlauf, Speicherbelegung und Rechenzeiten.



Die Endungen **.erg** sind klassische Ausgabedateien im Textformat. Die Ergebnisse der Gesamtberechnung sind darin wie in der PLB, jedoch ohne Bilder enthalten.




Die Endungen **.lst** sind die Ausgabedateien der Einzelprogramme. Sie enthalten nur die Ergebnisse der letzten Einzelberechnung und sind mit dem entsprechenden Abschnitt der ERG-Datei identisch. Sie können vor allem bei Fehlersituationen eine schnelle Diagnose ermöglichen.



GRA Die Endungen **.gra** sind CADINP-Eingabedateien des Programms WINGRAF und

können mit dem TEDDY geöffnet und bearbeitet werden.

 Die Endungen **.results** sind CADINP-Eingabedateien des Programms Result Viewer und können mit dem TEDDY geöffnet und bearbeitet werden.

2.5 Eingabedateien

Die Daten sind in einer freien Eingabeform in Sätzen im sogenannten CADINP-Format. (Vgl. Kapitel 8: [CADINP - Eingabesprache](#)). Dies ist eine programmierbare Makro-Sprache, die in vielen Fällen sehr effiziente Eingabeformen ermöglicht. Zu einer statischen Position können beliebig viele auch ineinander verschachtelte Dateien gehören.

Die Endungen **.sir** sind Eingabedateien, die vom Programm SIR erzeugt werden. Diese können mit TEDDY geöffnet und bearbeitet werden.

Die Endungen **.sofistix** sind spezielle Vorlagedateien vom SSD (SOFiSTiK Structural Desktop). Mit diesen Dateien erhält man für ausgewählte Systeme (z.B. eine Platte) voreingestellte Eingaben im SSD zur weiteren Bearbeitung.

2.6 Datenbasis

Normalerweise existiert für jede Position eines statischen Systems mindestens eine Datenbasis mit einem Projektnamen und einer oder mehreren zugehörigen Eingabedateien. Die Datenbasis kann bei größeren Projekten sehr wertvoll werden und sollte dann regelmäßig gesichert werden.

Unter einem System wird hier die Obermenge aller Teile eines Bauwerks oder Teilbauwerks verstanden, die im Verlaufe der Lebensdauer statisch zusammenwirken. Die Programme SOFiMSHA/SOFiMSHC oder SOFiPLUS erzeugen z.B. für ebene oder räumliche Tragwerke das statische Grundsystem. Die Anzahl und Art der Elemente wird dabei endgültig festgelegt. Bei der Berechnung kann aber jeweils ein Teilsystem herangezogen werden. Dies geschieht über die Gruppennummer. Der Benutzer sollte diesen Umstand bereits beim Entwurf seines statischen Systems berücksichtigen.

Es können Randbedingungen und Materialkonstanten beliebig verändert werden, es können mit weiteren Modulen Lastfälle berechnet oder überlagert werden, es können Bemessungen durchgeführt und Ergebnisse grafisch dargestellt werden. Alle Ergebnisse werden jedoch immer nur einfach gespeichert. Eine Berechnung eines Lastfalls mit veränderten Lasten überschreibt somit die alten Ergebnisse. Werden Querschnitte geändert, so werden auch alle abhängigen Ergebnisse (Bewehrung, Spannungen etc.) gelöscht, falls dies nicht explizit anderweitig definiert wird.


Die Datenbasis wird mit dem Software-System "CDBASE" verwaltet. Die indexsequentielle Struktur erlaubt einen effektiven Zugriff. Die zugehörige Dokumentation ist in der Datei CDBASE.CHM abgelegt, die Unterprogramme sind für programmierende Benutzer erhältlich.

Eine Datenbank kann in das CAD-System SOFiPLUS vollständig importiert oder daraus exportiert werden.

Für die Behandlung der Datenbasis ist ein vollständig interaktives Programm DBINFO vorhanden, das nicht nur statistische Informationen sondern auch Ausgabe- und Editiermöglichkeiten

ten sowie eine Kopierfunktion bereitstellt.

Das Programm DBINFO kann mit eine der folgenden Möglichkeiten aufgerufen werden:

- aus dem SSD oder TEDDY
 - SOFiSTiK → Datenbank Tools → Datenbank Informationen
- aus dem Explorer
 - Markieren der Datenbank (name.cdb)
 - rechte Maustaste → Datenbasis Information
- aus der Command Shell
 - Aufruf der Command Shell im TEDDY über Icon , Aufruf DBINFO, Angabe des Namens der Datenbasis

Nach Aufruf des Programmes DBINFO erscheint das folgende Menü:

- | | |
|--|---------------------------|
| a - Aufrufgeschichte | o - Ausgabemedium |
| s - Struktur (Elemente) | |
| l - Lastfälle | |
| m - Mischen 2. Datei | b - Backup anlegen |
| d - Liste der Sätze | g - Ausgabe 8 Stellen EIN |
| e - Editieren der Sätze | ** nur auf eigene Gefahr |
| z - Löschen des Errorflags
(Loeschen aller Locks mit Aufruf DBINFO project Z) | |
| q - Ausgang | |

Bitte entsprechende Taste drücken

Option a druckt die Historie der Zugriffe aller Programme auf die Datenbasis aus. Option s und l geben Auskunft über die gespeicherten Lastfälle und Elemente in der Datenbasis.

Option m erlaubt die Übernahme von Lastfällen aus einer anderen Datenbasis. Dies ist natürlich nur sinnvoll, wenn beide Basen das gleiche statische System beinhalten. Einsatzgebiete dafür gibt es vor allem bei sehr großen Projekten, wenn mehrere Mitarbeiter unterschiedlichste Untersuchungen am System bearbeiten.

Option d erlaubt die Anzeige von Dateninhalten der Datenbasis. Mit Option e können auch Daten verändert werden. Dies sollte nur auf ausdrückliche Anweisung von SOFiSTiK Mitarbeitern erfolgen. Eine Beschreibung der Dateninhalte ist dabei von Vorteil.

Beim Aufruf von DBINFO projekt,Z können Fehler-Flags und Record-Locks entfernt werden, die z.B. beim Programmabsturz bestehen bleiben könnten. Das Entfernen der Locks ist ebenfalls im SSD, TEDDY oder im Explorer möglich:

- aus dem SSD oder TEDDY
 - SOFiSTiK Datenbank Tools Locks bereinigen
- aus dem Explorer
 - Markieren der Datenbank (name.cdb)
 - RMB (rechte Maustaste) → Datenbasis Locks entfernen

Hinweis

Im SOFiSTiK-Environment (Vgl. Kapitel 9: [DEF - Umgebungsvariablen](#)) kann der Parameter CDACCESS=SINGLE gesetzt werden, der die Multitasking-Eigenschaften deaktiviert. Damit werden auch keine Record-Locks eingetragen.

2.7 Interaktive Module

Die modulare Struktur spiegelt sich auch in der Konzeption der interaktiven Module wieder. Statt eines einzigen Fensters in dem abwechselnd Eingabe oder Ausgabe zu sehen sind, kann der Benutzer viele Module verwenden, die auch alle gleichzeitig auf die Datenbasis zugreifen und sich gegenseitig Nachrichten zukommen lassen. Der Wechsel zwischen den Modulen erfolgt über die Windows-Task-Leiste oder über Funktionstasten oder Tool-Buttons, die innerhalb von TEDDY, WPS, Report Browser aber auch anderen Programmen erreichbar sind:



Das Programm WPS (wps.exe) zum Rechnen der Datensätze



Das Programm Report Browser zum Betrachten der Ausgaben inklusive der Grafiken.



Mit TASKS ist es möglich, Tasks für verschiedene Berechnungs- oder Bemessungsaufgaben einzufügen (analog den Tasks im SSD - SOFiSTiK Structural Desktop).



Mit TASKS ist es möglich, Tasks für verschiedene Berechnungs- oder Bemessungsaufgaben einzufügen (analog den Tasks im SSD - SOFiSTiK Structural Desktop).



Das Programm Result Viewer zur Ausgabe von maßstabsgerechten Querschnittsgrafiken und zur Ausgabe von selektiven Tabellen mit Werten aus der Datenbasis



Das Programm WINGRAF zur Ausgabe von maßstabsgerechten System- und Ergebnisgrafiken

2.8 Neuigkeiten

Die Programme der SOFiSTiK unterliegen einer ständigen Weiterentwicklung. Die Handbücher werden deshalb ständig aktualisiert und den Programmen als PDF-Dateien beigegeben. Die aktuellen Änderungen der Programme werden in sogenannten LOG-Dateien im HTML-Format gesammelt und dem Benutzer auf verschiedene Weise zur Verfügung gestellt.

- SOFiSTiK stellt die Dateien im Internet unmittelbar zur Verfügung.
- Der SSD und der TEDDY haben unter → Hilfe einen Unterpunkt für die LOG-Datei.
- Die Dateien können als RSS-Feeds auf der SOFiSTiK-Homepage abonniert werden.

2.9 Namenskonventionen der SOFiSTiK Dateien

Die wichtigsten Dateien die bei der Benutzung von SOFiSTiK entstehen sind die bereits benannten Eingabedateien (**.sofistik** und **.dat**) und die Datenbasis (**.cdb**). Die übrigen Dateitypen haben folgende Bedeutung:

Die Installationsdateien der SOFiSTiK Programme haben folgende Bedeutung:

.exe	Ausführbares Programm
.dll	Systemdatei die dynamisch zugeladen wird
.cmd	Kommando-Datei
.err	Fehler-Datei zum Programmmodul enthält alle Fehlermeldungen, Eingabesätze und Ausgabertexte in Deutsch und Englisch
.tax	Enthält benutzerspezifische Einstellungen
.tab	Enthält lesbare Parameter für ein Programm
.tbb	Enthält Menüs der grafischen Programme
.htm	Log-Datei Diese Datei enthält zu jedem Programm die Änderungen zum letzten Handbuch, sowie die beseitigten Fehler.
.pdf	Handbücher im Adobe portable document format
.chm	Komprimierte HTM-Hilfe-Datei
.wtm	Windows-TEDDY-Makrodatei
.def	Definitionsdatei für Parameter
.ini	Parameterdatei für Normen

Bei der Bearbeitung von Projekten entstehen folgende Dateien:

.dat	Eingabedatei, jedoch nicht zwingend
.dwg	Datei der Zeichnung von SOFiPLUS, in der das statische System und die Belastung eingegeben werden
.sofistik	Projektdatei des SSD (SOFiSTiK Structural Desktop)
.sofistix	Vorlagedatei des SSD (SOFiSTiK Structural Desktop)
.gra	Eingabedatei von WINGRAF erzeugt
.results	Eingabedatei von Result Viewer erzeugt
.lst	Ausgabedatei der Einzelprogramme
.erg	Ausgabedatei der Gesamtberechnung
.prt	Protokolldatei der Gesamtberechnung
.plb	Ergebnisdatei mit Grafiken
.cdb	Datenbasis, sollte normalerweise nicht gelöscht werden.
.cde	Eigenformen dynamischer Berechnungen

- .\$dn** Sowie andere Dateien mit \$ (Windows) oder z (Unix)
.zdn gefolgt von einem Buchstaben und einer Zahl
Dies sind Restartdateien (z.B. Steifigkeitsmatrix). Sie können gelöscht werden, jedoch müssen die Dateien eventuell später wieder neu erstellt werden, was gewisse Rechenzeit erfordern kann.
- .\$0n** Sowie andere Dateien mit \$ (Windows) oder z (Unix)
.z0n gefolgt von zwei Zahlen.
Arbeitsdateien können immer gelöscht werden. Normalerweise werden diese Dateien vom Programm bei normalem Ende selber gelöscht. Dateien mit wilden Zahlenkombinationen als Namen sind vom Betriebssystem angelegte Hilfsdateien, die infolge eines Warmstarts nicht mehr gelöscht werden konnten.
- .\$\$\$** Dies sind Hilfsdateien unter Windows, die gelöscht werden können.
.zzz Dies sind Hilfsdateien unter Linux, die gelöscht werden können.
- .#nn** Dies sind Hilfsdateien unter Windows, die gelöscht werden können.
.ynn Dies sind Hilfsdateien unter Unix, die gelöscht werden können.
- .###** Dies sind Error-Log Dateien unter Windows, die gelöscht werden können.
.yyy Dies sind Error-Log Dateien unter Unix, die gelöscht werden können.
- .sdb** Shadow-Datenbasis: Die Datei wird beim Beenden aller beteiligten Programme normalerweise gelöscht.

2.10 Datensicherung

Last but not least, muss darauf hingewiesen werden, dass alle wertvollen Daten unter Umständen zerstört werden können. Abgesehen von Ungeschicklichkeiten des Anwenders selbst, kann durch Fehler der Hardware, des Betriebssystems oder der Programme Informationen auf der Festplatte zerstört werden.

Erste Regel ist deshalb, die regelmäßige Anfertigung von Sicherheitskopien. Der Grundsatz dabei sollte sein, dass wertvolle Information immer auf drei unabhängigen Medien gespeichert wird, denn beim Anfertigen der Sicherung kann ein Fehler eventuell Kopie und Original zerstören. Insbesondere die Eingabedaten sollten täglich gesichert werden. Bei größeren Projekten muss auch die Datenbasis, die oft die Ergebnisse wochenlangender Arbeit und vieler Stunden Rechenzeit umfasst, gelegentlich gesichert werden.

Wichtige Dateien sind auf jeden Fall alle selbst erzeugten DAT-Dateien, sowie die Dateien **.sofistik**, die grafische Projektdatei **.dwg** und die Eingabedateien für das grafische Postprocessing **.gra** und **.results**. Im Programm TEDDY gibt es auch eine Funktion mit der man überflüssige Daten löschen kann. Falls eine CDB wichtige Daten enthält und gesichert werden soll, so kann man mit dem Programm DBINFO eine bereinigte bzw. verkleinerte backup-Version erzeugen.

2.11 SOFiSTiK Optionen

Bei den SOFiSTiK - Programmen stehen für die spezifischen SOFiSTiK-Einstellungen drei Einstellungsmenüs zur Verfügung.

Menü	Speicherort
SOFiSTiK > Anwender Optionen	Optionen für den einzelnen PC Diese Einstellungen werden in der Registrierungs-Datenbank abgelegt.
SOFiSTiK > Projekt Optionen	Optionen, die zum Projekt gehören Diese Einstellungen werden in einer Datei SOFiSTiK.DEF im aktuellen Projekt-Verzeichnis abgelegt.*

- * Die Einstellungen der Datei SOFiSTiK.DEF im Projektverzeichnis überschreiben die Voreinstellungen im übergeordneten Ordner (Globale Optionen).

2.11.1 Einstellungen der Sprache

Es wird unterschieden zwischen der Sprache der Dialoge und der Eingabe- bzw. Ausgabesprache der Dateien.

Die Sprache der Dialoge wird auf dem lokalen Rechner in die Registrierungs-Datenbank gespeichert und ist unter **SOFiSTiK > Anwender Optionen > SOFiSTiK Allgemein > Allgemein** einstellbar. Das Programm muss neu gestartet werden, damit diese Änderung aktiviert wird.

Die Eingabe- bzw. Ausgabesprache wird in der Datei **sofistik.def** abgespeichert.

2.11.2 Einheiten

SOFiSTiK-Programme erlauben die Ein- und Ausgabe bestmöglich in ingenieurmäßigen Einheiten darzustellen. Die Datenbank ist dimensionsrein in den SI-Einheiten kN, m, sec angelegt. Für Ein- und Ausgabe können jedoch auch andere Einheiten (z.B. N, mm oder cm²) oder eine unterschiedliche Anzahl von Nachkommastellen verwendet werden.

Drei Kategorien von Einheiten werden unterschieden:

- mm* **Feste Einheit.** Die Eingabe erfolgt immer in der angegebenen Einheit.
- [*mm*] **Explizite Einheit.** Die Eingabe erfolgt in der angegebenen Einheit, kann aber auch mit einer expliziten Vorgabe (z.B. 2.5[*m*]) erfolgen.
- [*mm*]₁₀₁₁ **Implizite Einheit.** Implizite Einheiten sind semantisch kategorisiert und durch einen entsprechenden Index (in grün) gekennzeichnet (Beispiel: Kategorien der Einheit "Länge" sind u.a. geodätische Höhe, Querschnittabmessung, Bauteildicke). Die Default-Eingabeeinheiten für die Kategorien werden durch das aktuell aktive (normenabhängige) Einheitenset festgelegt. Die Voreinstellung kann, wie oben dargestellt, durch explizite Zuweisung überschrieben werden. Angegeben ist die Voreinstellung für das Einheitenset 5 (Euro-Normen, NORM UNIT 5).

Die impliziten Einheiten sind semantisch differenziert, also z.B. "Systemabmessung",

"Querschnittsabmessung" oder "geodätische Höhe". Sie sind gekoppelt an ein (normenabhängiges) Einheitenset, welches in der jeweiligen INI-Datei hinterlegt ist. Alternativ kann das Einheitenset über den System-Dialog oder den Satz NORM explizit vorgegeben werden. Das Einheitenset weist dann die expliziten (=tatsächlich verwendeten) Einheiten zu, beispielsweise [m] für "Systemabmessung", [mm] für "Querschnittsabmessung" und [m] für "geodätische Höhe". Die hier getroffene Einstellung gilt gleichermaßen für Ein- und Ausgabe (dies hat sich in der Version 2012 gegenüber den Vorgängerversionen geändert) Man kann jedoch in jedem Eingabeblock über den Eingabesatz UNIT temporär für diesen Modul-Lauf ein anderes Einheitenset oder einzelne Einheiten zuweisen. Die aktuell geschaltete explizite Eingabe-Einheit kann dem Dialogkontext entnommen werden, bzw. - für Skript basierte CADINP Eingabe - der dynamischen Hilfe in der TEDDY-Statuszeile (Alternativ: Abfrage der aktuellen Einheiten mit dem CADINP-Kommando HELP). Eine explizite Vorgabe der Einheit wird hier ebenfalls unterstützt (siehe Abschnitt 8.2.13 [Einheiten Konvertierung](#)).

Explizite Einheiten dagegen sind nicht an ein Einheitenset gekoppelt, wie für implizite Einheiten besteht aber die Möglichkeit der Einheiten-Konvertierung bei der Eingabe (siehe Abschnitt 8.2.13 [Einheiten Konvertierung](#)).

Feste Einheiten sind eine aussterbende Spezies und werden nur noch an einigen wenigen Stellen erwartet.

Folgende Einheitensets sind vorgesehen:

- 0 = Standardeinheiten (m, kN, sec mit historischen Abweichungen)
- 1 = deutscher Hochbau (Querschnitte in cm, System in m)
- 2 = deutscher Stahlbau (Querschnitte in mm, cm², dm⁴, System in m)
- 3 = Brückenbau (wie 0, aber Schnittgrößen in MN statt kN)
- 4 = Grundbau (m, kN, sec)
- 5 = Ingenieurbau (Querschnitte in mm, System in m)
- 6 = metrisches System (Alle Abmessungen in mm, Lasten in kN)
- 7 = Mechanik (Alle Abmessungen in mm, Lasten in N)
- 8 = US customary (Imperial) Einheiten (AASHTO: foot, lbs, kip)
- 9 = US customary (Imperial) Einheiten (ACI/AISC: inch, lbs, kip)

Die Umrechnung der Imperial-Einheiten erfolgt nach einem Gesetz vom 1.Juli 1959 das amerikanische und britische Definitionen zusammenfasste zu:

- 1 inch = 0.0254 m (exakt!)
- 1 yard = 0.9144 m
- 1 lb = 0.45359237 kg
(avoirdupois)
- 1 lbf = 4.4482216 N

Eine ton ist immer die short ton zu 2000 lb.

3 Schnittstellen

3.1 CDB Export nach Excel

Mit der CDB Schnittstelle können Daten aus der SOFiSTiK Datenbank CDB in Microsoft Excel (VBA), Python, C++, C#, Fortran oder VB.NET angezeigt werden. Die gewünschten Daten können selbst ausgewählt und anschließend weiterbearbeitet werden.

Hinweis

Weitere Informationen über "SOFiSTiK Schnittstelle für VBA, Visual Basic .NET, C#, C++, Python, Fortran und CADINP @KEY" sind im Handbuch [CDB_INTERFACES](#) enthalten.

Weiter unten finden Sie die Interface Beispiele:

📁 C: ▶ Program Files ▶ SOFiSTiK ▶ 2018 ▶ ANALYSIS_50_X64 ▶ interfaces

3.2 Export nach DAT

Mit dem Export nach DAT können Informationen aus der SOFiSTiK Datenbank CDB in CADINP-Eingaben zur weiteren Verwendung mit dem Texteditor TEDDY umgewandelt werden. Damit ist es möglich, aus grafischen und interaktiven Eingaben über die Datenbank in CADINP-Eingaben zu erstellen. Z.B. kann aus einer Systemeingabe mit SOFiPLUS eine Eingabe des Programms SOFiMSHC erstellt werden. Die Materialien, Querschnitte und Lasten können jeweils auch nur material-, querschnitts- und lastfallweise selektiert werden.

Der Aufruf erfolgt aus dem SSD oder dem TEDDY über den Button  .

3.3 IFC Import und Export

SOFiSTiK bietet die Möglichkeit Daten im Format der Industry Foundation Classes (IFC) zu lesen und zu schreiben. Mit Hilfe des Befehls Datei IFC Import/Export können Materialien, Querschnitte, Strukturelemente und Lasten in eine IFC-Datei geschrieben werden. SOFiSTiK unterstützt dabei den sogenannten "Structural Analysis View" der IFC im Format `2x3 \textcolor{blue}{www.buildingsmart.de/bim-know-how/ifc}`. Auf die gleiche Weise kann eine IFC-Datei in dem Format gelesen und in eine SOFiSTiK Datenbank gewandelt werden. Der IFC-Import erzeugt dabei eine SOFiSTiK CADINP Eingabedatei, die in den SSD integriert oder per Teddy weiterbearbeitet werden kann.

Leider unterstützen die meisten IFC-kompatiblen Softwareprodukte ausschließlich den "Coordination View", der primär Rohbau- oder Architekturdaten überträgt. Dateien mit diesen Inhalten müssen über den Umweg eines Imports in *Autodesk Revit* oder *Autodesk Architectural Desktop* importiert werden. Eine Generierung des "Coordination View" aus SOFiSTiK heraus ist nicht möglich, weil die erforderlichen (üblicherweise volumetrischen) Informationen im SOFiSTiK-Analyse-Modell nicht vorliegen.

Einen guten Überblick über die Unterschiede der verschiedenen IFC-Varianten bietet folgende Seite: [\textcolor{blue}{www.buildingsmart.de/bim-know-how/ifc}](http://www.buildingsmart.de/bim-know-how/ifc).

Die Bedienung des Dialogs sollte selbsterklärend sein, nach Wahl der Übertragungsrichtung ist die zu übertragende Datei zu wählen, mit dem Befehl Anwenden wird die Konvertierung gestartet.

3.4 Extensions für Autodesk® Revit®

Die SOFiSTiK Schnittstelle Autodesk® Revit® bietet eine nahtlose Integration der FE Berechnung mit allen Möglichkeiten der SOFiSTiK Software. Die vollautomatische FE Netzgenerierung mit einem der leistungsfähigsten 3D Netzgeneratoren wird direkt in Revit Structure gestartet, um Systemänderungen schnell in statische Rechenmodelle umzusetzen.

Die aktuelle Version der Schnittstelle bietet zudem viele Funktionen an, welche sowohl die Kommunikation zwischen Revit Structure und der SOFiSTiK Software verbessern als auch Eingabe im Revit Structure erleichtern.

Folgende Funktionen erleichtern den Workflow:

- Materialmapping
- Querschnittsmapping
- SOFiSTiK Gruppennummern direkt im Revit Structure zuweisen
- SOFiSTiK Lastverteilungsfläche für Flächenlasten auf Trägerrosten
- Flächenlast teilen für schachbrettartige Lastanordnung bei Flachdecken
- Subsysteme rechnen: Export eines Subsystems (z.B. Platte mit Unterzügen), wobei angeschlossene Bauteile wie Wände, Stützen, etc. als elastische oder feste Auflager angesetzt werden.

Eine sofortige Kontrolle der Struktur ist über den ANIMATOR und mit WinGRAF direkt aus Revit möglich. Das System kann dann komfortabel in ein SOFiSTiK Structural Desktop (SSD) Projekt integriert werden, um die weiteren Berechnungsschritte dialoggestützt durchzuführen. Die weitere Modifikation der Struktur ist natürlich auch mit dem auf AutoCAD basierenden SOFiPLUS möglich.

3.5 SOFiSTiK Rhinoceros Interface

Das SOFiSTiK Rhinoceros Interface erweitert die Funktionalität von Rhino um die Definition und Generierung eines Finite-Elemente-Modells für eine Strukturanalyse mit SOFiSTiK. Geometrieobjekte in Rhino wie Punkte, Kurven oder Flächen können um Strukturinformationen wie Querschnitte oder Materialien erweitert werden. Die Vernetzung des Modells mit Stäben und Flächenelementen erfolgt aus dem Programm heraus.

Eine Reihe zusätzlicher Hilfsmittel, die in einer SOFiSTiK Toolbox zusammengefasst sind, bieten die Möglichkeit zur Visualisierung von Querschnitten (z.B. für Renderings), zur Darstellung der Attribute im Modell oder zur Selektion von Objekten nach bestimmten Eigenschaften.

Mit Installation des SOFiSTiK Rhinoceros Interfaces wird Rhino in die SOFiSTiK Berechnungsumgebung (SOFiSTiK Structural Desktop) integriert, so dass sich ein reibungsloser Übergang zwischen Modellieren, Vernetzen und allen Schritten der Strukturanalyse ergibt, ohne Daten zwischen verschiedenen Programmen manuell austauschen zu müssen.

In ähnlicher Weise wie alle Daten in Rhino selbst, können auch die Strukturinformationen mit Hilfe von Rhino Script oder benutzerdefinierten Programmerweiterungen bearbeitet und verändert werden. Damit lässt sich die Funktionalität des SOFiSTiK Rhino Interfaces erweitern und an unternehmens- oder projektspezifische Anforderungen anpassen.

Weitere Informationen sind im Handbuch [RHINO_INTERFACE](#) enthalten.

4 Konvertierung von Projekten (Statik)

4.1 Vorbemerkung

Dieses Dokument stellt Informationen und Hinweise zusammen, die eine Konvertierung bestehender SOFiSTiK Statik Projekte (Version 2016) zur Bearbeitung mit der Version 2018 erleichtern sollen.

Hinweis

Wir empfehlen in laufenden Projekten die Hauptversion nicht zu wechseln, falls dies doch notwendig wird kontaktieren Sie bitte support@sofistik.de für individuelle Beratung.

4.2 Konvertierung von Projektdateien

4.2.1 SSD Projektdateien (.sofistik)

Beim Öffnen der Projektdatei im SSD SOFiSTiK 2018 wird eine — vom Anwender zu bestätigende — automatische Konvertierung der SSD-relevanten Datenformate angestoßen.

Hinweis

Nach dieser Konvertierung ist die Projektdatei für die Bearbeitung mit früheren SSD Versionen < SOFiSTiK 2018 gesperrt.

Zusätzlich wird ein automatisches Upgrade der Datenbankinhalte (.cdb) auf das Datenformat der Version 2018 angestoßen (siehe Unterabschnitt 4.2.3).

Eventuell nicht mehr vorhandene Tasks werden mit Ihrer letzten Texteingabe in einen reinen Text-Task umgewandelt.

4.2.2 SOFiPLUS(-X) Projektdateien (.dwg)

Beim Öffnen der Projektdatei in SOFiPLUS(-X) SOFiSTiK 2018 wird eine — vom Anwender zu bestätigende — automatische Konvertierung der SOFiPLUS-relevanten Datenformate angestoßen.

Hinweis

Nach dieser Konvertierung ist die Projektdatei für die Bearbeitung mit früheren SOFiPLUS(-X) Versionen < SOFiSTiK 2018 gesperrt.

Zusätzlich wird ein automatisches Upgrade der Datenbankinhalte (.cdb) auf das Datenformat der Version 2018 angestoßen (siehe Unterabschnitt 4.2.3).

4.2.3 Datenbank (.cdb)

Das Format der Daten in Datenbanken der Version 2016 und Version 2018 ist weitgehend kompatibel, aber nicht identisch.

Hinweis

Ein Mischen von Daten unterschiedlicher Datenbank-Versionen ist nicht qualitätsgesichert und entspricht nicht der empfohlenen Vorgehensweise.

Es steht ein Upgrade-Mechanismus zur Verfügung, der eine Konvertierung der Daten der Version 2016 auf das Datenformat der Version 2018 leistet. Diese Konvertierung wird implizit angestoßen durch

- eine Konvertierung von SSD Projektdateien (.sofistik) oder
- eine Konvertierung von SOFiPLUS Projektdateien (.dwg).

Eine Konvertierung kann ebenfalls ausgelöst werden durch explizites Öffnen der Datenbank im SSD 2018.

Hinweis

Nach dieser Konvertierung ist die Datenbank für die Bearbeitung mit früheren SOFiSTiK Versionen < SOFiSTiK 2018 gesperrt.

4.2.4 CADiNP Eingaben

Für CADiNP Eingaben ist eine automatische Konvertierung aufgrund des generischen Charakters der Skriptsprache im allgemeinen Fall nicht möglich und wird daher nicht unterstützt (siehe Abschnitt 8.8).

SOFiSTiK verfolgt in diesem Zusammenhang nachstehende Strategie.

In Zukunft entfallende Eingaben werden als *überholt* (deprecated) markiert. Für überholte Eingaben gilt:

- Sie sind (und bleiben) voll funktional für das aktuelle Major Release (Version 2018).
- Für diese Eingaben ist eine Unterstützung ab einem definierten Zeitpunkt in der Zukunft (in der Regel für das nächste Major Release) nicht mehr vorgesehen.
- Eine Dokumentation oder Eingabebeschreibung für das aktuelle Major Release (Version 2018) im Regelfall bereits nicht mehr verfügbar.
- Überholte Eingaben werden mit einer Warnung gemeldet. Diese Warnung enthält in der Regel bereits einen konkreten Hinweis zur vorgeschlagenen aktuellen Ersetzung.

Hinweis

Für neue oder langfristige Projekte wird empfohlen, überholte Eingaben zu vermeiden und stattdessen die aktuelle Ersetzung zu verwenden.

4.3 Änderungen

Zur Erleichterung des Überblicks enthält dieser Abschnitt – ohne Anspruch auf Vollständigkeit – eine Zusammenstellung wesentlicher Änderungen im Vergleich zur Version SOFiSTiK 2016.

4.3.1 Entfallene Normen

Folgende Normen sind in der Version 2018 ersatzlos entfallen: SIA 162, JS JRA, MSZ UT414, ET RC-2001, IS IRC18, IS IRC21, US ACI-318-99, US ACI-318-02, US AASHTO-2002, SNIP RK50333

Für folgende entfallene Normen werden bestehende Datensätze aus Version 2016 automatisch gemappt:

DIN 1052	gemappt nach DIN 1052-1988
DIN 1052	gemappt nach DIN 1052-1988
EN 1999-2007	gemappt nach EN 1993-2005
EN 1999-2007 mit Ländercode DE/49	gemappt nach DIN EN1993-2005
I DM-2005	gemappt nach UNI DM-2008
US AISC	gemappt nach US AISC-2005

Für Normen der Länder Italien, Schweden und Spanien wurde die Länderbezeichnungen umbenannt:

Italien	neu UNI
Schweden	neu SS
Spanien	neu UNE

Die Normenbezeichnungen werden für bestehende Projekte aus Version 2016 automatisch umgestellt.

5 SSD - SOFiSTiK Structural Desktop

5.1 Benutzeroberfläche des SSD

Der SOFiSTiK Structural Desktop (SSD) stellt eine einheitliche Benutzeroberfläche für das Gesamtpaket der SOFiSTiK-Software dar. Das Modul steuert Pre-Processing, Processing und Post-Processing.



Ein Doppelklick auf das Programmsymbol oder eine beliebige Datei **.sofistik** öffnet den SSD.

Das statische System kann grafisch mit SOFiPLUS(-X) oder als parametrisierte Texteingabe über TEDDY eingegeben werden. Die Steuerung der Berechnung und der Bemessung erfolgt über Dialoge, die im Taskbaum projektbezogen verwaltet werden.

Der Bildschirm ist unterteilt in die drei Bereiche: Taskbaum, Tabellenbereich und Arbeitsbereich

5.2 Arbeitsweise

Der SSD arbeitet aufgaben-(= task)orientiert. Die Tasks sind in Gruppen angeordnet (z. B. die Gruppe "System" enthält die Tasks für Materialien, Querschnitte, Geometrie, Lasten und Kombinationsvorschriften). Beim Öffnen eines Projektes werden abhängig von der Aufgabenstellung die erforderlichen Gruppen und Tasks voreingestellt.

5.2.1 Gruppen

Die einzelnen Berechnungsgruppen sorgen für die Übersicht im Baum und strukturieren die Gesamtberechnung. Diese Struktur kann vom Anwender jederzeit geändert werden, indem er Namen der Tasks modifiziert und/oder die einzelnen Tasks mit der Maus an die gewünschte Stelle zieht. Der Anwender kann weitere Aufgabengruppen mit zugehörigen Tasks ergänzen oder auch entfernen (Rechtsklickmenü im Baum).

5.2.2 Tasks

Die möglichen Tasks stehen über das Rechtsklickmenü im Taskbaum zur Verfügung. Sie können i. d. R. an jeder beliebigen Stelle im Baum eingefügt werden. Sobald mit der rechten Maustaste **RMB** den Befehl **Task einfügen** ausgewählt wird, erscheint nachfolgende Dialogbox mit allen verfügbaren Tasks.


Taskbaum

Im Taskbaum stehen die einzelnen Tasks zur Verfügung und werden über ein sich anpassendes Rechtsklickmenü bearbeitet.

Arbeitsbereich

In den Arbeitsbereich hängt sich standardmäßig der ANIMATOR zur grafischen Überprüfung des Systems ein. Je nach Bearbeitungsstand werden in diesem Bereich auch WPS zur Kontrolle der Berechnung oder TEDDY zur Texteingabe dargestellt. Die grafische Bearbeitung mit SOFiPLUS(-X) wird in einem eigenen Fenster durchgeführt.

5.2.3 Vorlagendateien *name.sofistix*

Für die Bearbeitung von häufig wiederkehrenden Standardaufgaben sind die Vorlagendateien mit der Endung **.sofistix** vorgesehen. Die "allgemeinen" Vorlagen sind in einem Unterverzeichnis des Statikverzeichnisses gespeichert, standardmäßig z.B. in  C: ▶ Programme ▶ SOFiSTiK ▶ 2016 ▶ ANALYSIS_33_X64 ▶ SSD-Templates.

Anlegen benutzerdefinierter Vorlagenverzeichnisse

Für eigene Vorlagen kann der Anwender sich weitere Vorlagenverzeichnisse definieren.

SOFISTiK > Anwenderoptionen > SSD-Vorlagen Verzeichnis > File > Open > Hinzufügen

In diesem Verzeichnis können weitere Unterverzeichnisse angelegt werden. Diese Unterverzeichnisse erscheinen beim Aufruf als Registerkarte. Es ist maximal eine Ebene von Unterverzeichnissen vorgesehen.

Erzeugen von eigenen "benutzerdefinierten" Vorlagendateien

Eine beliebige Datei **name.sofistik** wird in das gewünschte Vorlagenverzeichnis als Vorlage **name.sofistix** gespeichert.

Alle aktuellen Projekteinstellungen lassen sich als Vorlage abspeichern. Insbesondere die Gliederung und Reihenfolge der Tasks bleiben erhalten. Das Material und die Querschnitte sind von der gewählten Norm abhängig. Eine einmal festgelegte Norm kann innerhalb des Projektes nicht geändert werden. Datei > Projekt als Vorlage speichern ...

Eine spätere **Änderung der Norm ist möglich**, wenn die Vorlage "Projekt ohne normenabhängige Tasks und Informationen speichern" gespeichert wird.

Die vorhandenen Vorlagenverzeichnisse werden unter Verzeichnisse angezeigt. Die gespeicherte Datei name.sofistix steht nun als weitere Vorlage zur Verfügung.

Arbeiten mit Vorlagendateien *name.sofistix*

Datei > Neues Projekt von Vorlage...

Die vorhandenen Vorlagen aus dem Vorlagenpfad werden angeboten.

Stammverzeichnis:

- "Allgemein"

Die gewünschte Vorlagendatei **name.sofistik** wird selektiert und mit dem Knopf Speichern unter ... unter einem (neuen) Dateinamen *name1.sofistik* in ein Projektverzeichnis

gespeichert.






Die neue Datei enthält alle Tasks der Vorlage. Ebenso werden die Daten (z.B. Querschnitte, Geometrie,... usw.) von der Vorlage in die neue Datei übertragen. Die Daten stehen sofort für eine Berechnung bereit.

Bei "Vorlagen ohne Norm" kann die Norm neu festgelegt werden. Die Materialien, Querschnitte, Einwirkungen und ggf. weitere normenabhängige Angaben müssen ergänzt werden.

5.3 Aufbau und Funktionsweise

5.3.1 Berechnungsstatus

Die einzelnen Tasks erhalten ein Symbol für ihren Berechnungsstatus.

	Without calculation	Input is written directly into the database
	grüner Haken	keine Berechnung erforderlich
	blauer Pfeil	neuere Eingaben im Dialog → Berechnung erforderlich
	blaues Kreuz	Daten nicht aktuell → Berechnung erforderlich
	rotes Kreuz	Fehler enthalten → Berechnung erforderlich
	grünes Kreuz	Warnungen enthalten → Berechnung eventuell erforderlich

5.4 Zusätzliche Aufrufmöglichkeiten

Für besondere Zwecke stehen dem erfahrenen Anwender zusätzliche Aufrufmöglichkeiten aus der Eingabeaufforderung (Command-Shell) zur Verfügung:

Parameter	Beschreibung
+ bzw. ++	Öffnet die letzte bzw. vorletzte Datei. Wird ohne vorgestellten Schalter - oder / verwendet.
-nosingle	Startet die Applikation als eigenständige Instanz (no single application).
-test	Setzt einen generellen Test-Flag. Dieser ist nur für die Entwickler von Interesse. Je nach Entwicklungsstand werden einzelne Meldungsfenster aktiviert.
-noani	Beim Start vom SSD wird das automatische Öffnen eines AnimatorViews unterdrückt.
-dat:name	Beim Start vom SSD wird direkt eine Gesamt-DAT-Datei erstellt und das SSD-Fenster wieder unmittelbar geschlossen.

6 Grafische Eingabe SOFiPLUS(-X)

6.1 Allgemeine Hinweise

SOFiPLUS(-X) ist eine CAD - orientierte Eingabeoberfläche zur Erzeugung von Strukturen und Belastungen. Diese werden für die Berechnung mit den Programmen der SOFiSTiK aufbereitet. Das Programm steht als Vollversion SOFiPLUS (Aufsatz für AutoCAD) oder als stand alone Produkt SOFiPLUS-X (incl. AutoCAD OEM - Kern) zur Verfügung.

Zur anschließenden Weiterbearbeitung werden der SSD und die Berechnungsprogramme der SOFiSTiK benötigt.

Hinweis

Wir empfehlen das Arbeiten mit Strukturelementen und der automatischen Vernetzung. Der Workflow mit direkter Bearbeitung von FE Elementen wird in einer der nächsten Versionen nicht mehr unterstützt werden.

6.2 Arbeitsweise

Die allgemeine Arbeitsweise ist im Handbuch SOFiPLUS beschrieben. Beispiele und Lehrfilme zu den einzelnen Features finden Sie unter dem Link Infoportal auf der Homepage der SOFiSTiK AG

<http://www.sofistik.de/Infoportal>

und auf unserem YouTube Kanal


<http://www.youtube.com/user/SOFiSTiKAG/videos>

6.2.1 Start aus dem Structural Desktop (SSD)

Das Programm SSD wird aus dem Programmmanager, oder über die Verknüpfung auf dem Desktop gestartet. Bei einem neuen Projekt muss sofort das statische System festgelegt werden, da die Arbeit mit dem SSD ohne ein statisches System nicht möglich ist. Als Art der Systemeingabe wird "SOFiPLUS(-X) - grafische Systemeingabe" eingestellt. Dadurch wird der untere Teil der Dialogbox gefüllt und es können zusätzliche Einstellungen für die Arbeit mit dem Programm SOFiPLUS gemacht werden.

Nach dem Verlassen der Dialogbox "SOFiSTiK Systeminformation" mit werden auf der linken Seite des SSD - Bildschirms einige Aufgaben (Tasks) eingefügt. In der Gruppe "System" befindet sich auch die Aufgabe "Grafische System- und Lasteingabe (SOFiPLUS(-X))". Über diesen Task kann das Programm SOFiPLUS mit einem Doppelklick gestartet werden. Es folgt die Eingabe des statischen Systems und der zugehörigen Belastung. Mit dem Befehle werden die Elemente für die Systemeingabe und die Lasten in die SOFiSTiK Datenbasis (.cdb) geschrieben und stehen so den weiteren Berechnungsprogrammen zur Verfügung.

Soll die Einstellung "Art der Systemeingabe" voreingestellt auf den Wert "SOFiPLUS(-X) -

grafische Systemeingabe" geändert werden, so geschieht dies in der Dialogbox "SOFiSTiK: Anwenderoptionen". Diese findet sich im Abrollmenü .

6.3 Aufbau

Nach dem Start von SOFiPLUS(-X) öffnet sich folgender Bildschirm:

Zur Eingabe des statischen Systems wird vor allem die Sidebar verwendet. Warnungen und/oder Fehler beim Schreiben in die Datenbasis werden ebenfalls in der Sidebar protokolliert.

Zur Einarbeitung in das Programm SOFiPLUS(-X) wird empfohlen, ein Beispiel aus dem Infoportal durchzuarbeiten oder ein online Training aus unserem YouTube Kanal durchzuführen. Schulungen - zusammen mit dem Programm SSD - werden regelmäßig angeboten.

7 Eingabe mit TEDDY

7.1 Allgemeines

Mit TEDDY steht ein Befehlszentrum und leistungsfähiger Texteditor zur Verfügung, der die Eingabesprache CADINP (Kapitel 8 : [CADINP- Eingabesprache](#)) optimal unterstützt.

TEDDY stellt beim Bearbeiten der Eingabedateien eine integrierte Hilfe zur Verfügung. Es werden automatisch die Kennworte des aktuellen Eingabedatensatzes in der Status-Zeile eingeblendet und auf Anforderung **F1**-Taste wird die zugehörige Seite des Handbuches (PDF-Datei) am Bildschirm simultan eingeblendet, ohne den Editiervorgang zu unterbrechen. Mit der **ESC**-Taste wird die Hilfe wieder verlassen.

Ausführliche Erläuterungen zum TEDDY sind unter **Hilfe** **Teddy Hilfe** zu finden.

7.2 Starten von TEDDY

TEDDY ist eine Windows-Anwendung und lässt sich mit den dafür vorgesehen Möglichkeiten starten.



Ein Doppelklick auf das Programmsymbol oder eine beliebige DAT-Datei öffnet das Programm TEDDY.

Für besondere Zwecke stehen dem erfahrenen Anwender zusätzliche Aufrufmöglichkeiten aus der Eingabeaufforderung (Command-Shell) zur Verfügung:

```
ted [ /optionen] [ Dateiname] [ Zeile Spalte]
```

Dateiname:

Anstelle eines Dateinamens kann auch + oder ++ eingegeben werden. Damit wird die letzte bzw. die vorletzte Datei geladen. Mit abc*.dat wird der Datei-Öffnen-Dialog mit der übergebenen Suchmaske gestartet. Ein Punkt steht für die Suchmaske *.* des ganzen Verzeichnisses.

Optionen:

/sSofistik-Pfad Pfadangabe zu SOFiSTiK-Programm-Dateien (Umbiegen bzw. Setzen der Umgebungsvariablen SOFiSTiK=...)

/1 bis /4 Mit diesen Flags können mehrere Instanzen von TEDDY gestartet werden. Die Zahlen 1 bis 4 beeinflussen die Farbe des TEDDY-Icons.

Übersicht der möglichen Eingabeaufforderungen:

Parameter	Beschreibung
+ bzw. ++	Öffnet die letzte bzw. vorletzte Datei. Wird ohne vorgestellten Schalter - oder / verwendet.

Parameter	Beschreibung
-0 -1 -2 -3 -4	Ermöglicht den Start von bis zu 5 Instanzen (Voreinstellung: -0).
-nosingle	Startet die Applikation als eigenständige Instanz (no single application).
-test	Setzt einen generellen Test-Flag. Dieser ist nur für die Entwickler von Interesse. Je nach Entwicklungsstand werden einzelne Meldungsfenster aktiviert.
Zeilen-Nr	Der Cursor wird direkt in der angegebenen Zeile platziert. Die Zeilen-Nummer muss der letzte übergebene Parameter sein.

7.3 Wichtige Befehle

Die wichtigsten Befehle sind auf Funktionstasten gelegt:

Taste	Beschreibung
F1	Hilfe zum aktuellen Eingabe-SATZ (verlassen mit Esc -Taste)
F2	Suchen
F3	Weitersuchen
F4	Suchen und Ersetzen
F5	Block Anfang/Ende markieren
F6	Markierten Block kopieren
F7	Markierten Block verschieben
F8	Aktuelle Zeile doppeln
F9	Fenster wechseln
F12	Berechnung starten

Weiterhin gibt es spezielle Tasten-Codes, die durch Drücken der **Alt**-Taste bzw. der **Strg**-Taste in Verbindung mit einer normalen Taste oder Funktionstaste erzeugt werden. Diese Tastenkombinationen sind ein wichtiges Bedienungsinstrument, das für den geübten Schreiber eine sehr schnelle Handhabung bietet. (10 Finger System!). Einige Tastenkombinationen sind hier zusammengestellt:

7.3.1 Kombinationen mit der Alt-Taste (Auswahl)

Taste	Beschreibung
Alt + L	markieren mehrere Zeilen (Zeilenblock)
Alt + B	markieren mehrere Spalten (Spaltenblock)
Alt + H	Blockmarkierungen löschen
Alt + R	Datei als Block einlesen
Alt + W	(markierten) Block als Datei schreiben

Taste	Beschreibung
Alt + Z	Zahlenspalten-Operationen (bei markiertem Spaltenblock)

7.3.2 Kombinationen mit der Strg-Taste (Auswahl)

Taste	Beschreibung
Strg + A	Alles markieren
Strg + C	Kopieren von markiertem Block
Strg + V	Einfügen von markiertem Block
Strg + X	Ausschneiden von markiertem Block
Strg + S	Sichern
Strg + Y	Zeile löschen
Strg + Z	Befehl rückgängig

7.3.3 Übersicht über alle Strg- / Alt- Befehle

Eine vollständige Zusammenstellung aller **Strg**- Kommandos finden Sie in der integrierten TEDDY-Hilfe bei **Hilfe** > **Strg- Kommandos**.

7.3.4 Beispiel mit Spaltenblock

Es wird ein (leerer) Spaltenblock aufgezo-

- **Alt** + **B** -Taste für Blockanfang
- Block aufziehen
- **Alt** + **B** - Taste für Blockende

Der Blockbereich ist nun farbig markiert.

- **Alt** + **Z** - Taste

Nun ist der Zahlenspalten-Dialog geöffnet. Damit kann man Reihen erstellen oder Werte addieren und vieles mehr.

7.4 Eingabe von Daten

Jedes SOFiSTiK-Programm erwartet Eingabedaten. Die Eingabedaten werden in Text-Dateien zur Verfügung gestellt. Damit die Daten korrekt interpretiert werden, müssen sie in einer definierten Reihenfolge vorliegen. Ein Datensatz beginnt mit dem reservierten Satznamen PROG gefolgt von dem Programmnamen. Innerhalb einer Datendatei können so mehrere Programmmodule angesteuert werden:

```
+PROG AQUA
//Daten für Programm AQUA
```

```
+PROG TENDON  
//Daten für Programm TENDON
```

```
+PROG SOFIMSHA  
//Daten für Programm SOFIMSHA
```

Damit TEDDY in der Lage ist, eine Datei als Eingabedatei zu erkennen, muss die erste Zeile der Eingabedatei eine bestimmte Form haben. Dies kann sein:

```
PROG AQUA
```

```
+PROG AQUA
```

```
-PROG AQUA
```

Darüber hinaus geht TEDDY bei jeder Datei mit der Endung **.dat** davon aus, dass es sich um eine SOFiSTiK-Eingabedatei handelt. Zusätzlich besteht die Möglichkeit, über den Menüpunkt **Extras** > **Dateityp** den Typ SOFiSTiK-Eingabe explizit einzustellen.

Sofern die entsprechenden Dateien **progname.err** existieren, kann TEDDY bei allen Zeilen die Satznamen und Kennworte erkennen. In der Statuszeile werden die Kennworte zu dem aktuellen Satznamen, das aktuelle Modul und die Eingabesprache (Deutsch oder Englisch) angezeigt.

7.5 Hilfe und interaktives Handbuch

7.5.1 Aktivierung der Hilfe

Mit der **F1**-Taste wird die Hilfe aktiviert. Zu dem in der zugehörigen PROG-Zeile aufgerufenen Programm wird die passende Handbuchseite aufgeschlagen. Die Namen der zugehörigen aktuellen Handbücher können als PDF mit einer zusätzlichen Sprachkennung (**_0** Deutsch oder **_1** Englisch) aus progname abgeleitet werden und müssen natürlich vorhanden sein.

Im Navigationsfenster wird das Inhaltsverzeichnis des zugehörigen Handbuches aufgelistet, so dass nun auch alle Erläuterungen und Beispiele interaktiv nachgeschlagen werden können. Sobald man aber wieder dem TEDDY den Fokus gibt, springt die Hilfe immer auf die aktuelle Handbuchseite.

Die Hilfe wird mit der **Esc**-Taste wieder verlassen. Voraussetzung ist, dass das Programm TEDDY den Fokus hat. Alternativ lässt sich das Hilfe-Fenster mit dem Schließ - **X** beenden.

Die Handbuchseiten können einzeln (oder komplett) ausgedruckt werden. Das Layout ist für einen zweiseitigen Druck optimiert.

















 Hilfe **F1**

 Hilfe schließen **Esc**

 Satznamen beibehalten

7.6 TEDDY als Kommandozentrale

Neben der Datenaufbereitung als Texteditor (Eingabe erstellen, ergänzen, ändern) stellt TEDDY eine geschlossene Benutzerumgebung zur Verfügung. Ohne Verlassen des Editors lässt sich die Berechnung starten, die Ergebnisse anzeigen, die interaktive grafische Ausgabe ansteuern und vieles mehr. Hierzu steht ein Werkzeugkasten (Toolbar) zur Verfügung:

	Programmstart WPS F12
	Schnellstart WPS (ohne Nachfrage)
	Schnellstart WPS nur aktuelles Modul der Eingabe
	+/-PROG (Wechsel Modulsteuerung)
	+PROG (alle setzen)
	-PROG (alle deaktivieren)
	+/-PROG ab/bis Cursor
	SOFiSTiK-Dateien
	Eingabedatei *.dat
	Ergebnisdatei *.erg
	Listdatei *.lst
	Protokolldatei *.prt
	Animator (Ergebnisanimation)
	WinGRAF (grafische Darstellung)
	Result Viewer (grafische Darstellung und Listenausgabe)
	Tasks (analog SSD)
	Export nach DAT
	Archivieren
	Aufräumen
	Datenbank aufräumen
	Explorer
	Command-Shell

7.7 TEDDY- Kapitel und TEDDY-Label

Eine TEDDY- Eingabedatei kann durch Kapitel- und durch Label-Einträge strukturiert werden.

Die Kapitel und Label erscheinen in der Modulübersicht im TEDDY-Modulbaum und können dort angesprungen werden.

!#! Kapitel

!*! Kapitel

In der PLB-Ergebnisdatei können einzelne Kapitel mit Report Browser per Mausklick geöffnet und geschlossen werden. Label dienen jedoch ausschließlich als Sprungziele.

Kapiteleinträge können nur außerhalb der Datenblöcke PROG/ENDE gesetzt werden

```
PROG
...
ENDE

!#!Kapitel User-Text

PROG
...
ENDE
```

Mit **!+!Kapitel** wird ein Kapitel in TEDDY als geöffnet und mit **!-!Kapitel** wird das Kapitel als geschlossen markiert (**!#!Kapitel** ist identisch mit **!+!Kapitel**).

Label-Einträge können auch innerhalb eines Datenblockes PROG/ENDE gesetzt werden und sind als reine Sprungziele konzipiert.

```
PROG
KOPF
!*!Label User-Text
...
ENDE
```

8 CADINP - Eingabesprache

Die Eingabesprache CADINP zur formatfreien Eingabe von Daten ist eine Weiterentwicklung der vom Bundesministerium für Forschung und Technologie im Rahmen des CAD-Programmes geförderten Studie CADINT (M.Ahn, K.H.Böckeler, W.Haas / Eingabekonventionen für CAD-Programme, CAD-Bericht Kfk-CAD 39, Kernforschungszentrum Karlsruhe, 1976).

Die von SOFiSTiK entwickelte Variante enthält umfangreiche Erweiterungen, die in einigen wenigen Punkten eine Abweichung von diesem Standard notwendig machten.

Die vollen Fähigkeiten der Sprache, die jede Art von geführter Eingabe wohl niemals erreichen kann, werden im Zusammenwirken mit SPS bzw. WPS möglich. CADINP und SPS stellen zusammen eine Programmiersprache zur Berechnung von Problemen des konstruktiven Ingenieurbaus dar. Die Verarbeitung erfolgt in zwei Stufen.

- SPS bzw. WPS erzeugen aus einer Datei und beliebigen eingeschobenen Dateien für jedes Programm eine Eingabedatei. Diesen Vorgang nennt man in der Informatik "parsen" was auf deutsch "grammatisch analysieren" heißt. Bei diesem Vorgang werden üblicherweise global definierte Texte eingesetzt und ganze Blöcke übersprungen oder eingefügt. Strings werden mit \$(name) angesprochen. Sie können in der Eingabedatei oder im Environment definiert oder undefiniert sein. Gerechnet wird zu diesem Zeitpunkt noch gar nichts, weshalb es einen Unterschied macht, ob der String $A = "2+2"$ an der Stelle $$(A)*3$ oder $$(A))^3$ eingefügt wird.
- Die Rechenprogramme selbst starten jedes für sich ihren eigenen CADINP-Prozessor, der jetzt wirkliche Berechnungen durchführt. Auch hier gibt es wieder Variablen, die aber jetzt mit einem # markiert werden, keine Klammern brauchen und als Zahlen gespeichert sind. Sie können als Zahlen oder Texte innerhalb des jeweiligen Moduls weiterverarbeitet werden.

Mit der Kombination dieser beider Stufen lassen sich komplette Statiken mit Handrechnungen und Dokumentation der Rechenannahmen beschreiben, die eine vollständige Statik aus dem Rechner erzeugen können.

8.1 Definitionen und Bezeichnungen

8.1.1 Zeilen (Physikalische Sätze)

Die Eingabe erfolgt in Zeilen. Eine Zeile hat maximal 255 Zeichen und entspricht einer Zeile am Bildschirm.

8.1.2 Sätze (Logische Sätze)

Gleichartige oder logisch zusammengehörende Daten werden zu Sätzen zusammengefasst. Jeder Satz hat einen Satznamen. Die Sätze brauchen nicht mit der Einteilung in Zeilen übereinzustimmen.

8.1.3 Trennzeichen

Innerhalb eines Satzes werden die einzelnen Werte durch Trennzeichen getrennt. Als Trennzeichen werden ein oder mehrere Leerzeichen (Blank) verwendet.

8.1.4 Datenwerte

Daten können Zahlen oder Texte (=Literals) sein. Zahlen können mit Vorzeichen und einem Exponenten behaftet sein. Wird an einer Stelle eine ganzzahlige Eingabe erwartet, so wird die eingegebene Zahl auf den nächsten ganzzahligen Wert gerundet, z.B.

```
2      0.04   -.5   3.7E8  -0.5E-3
```

Texte sind beliebige Folgen von Zeichen. Nur die ersten vier Zeichen sind signifikant. Sofern Verwechslungen mit Zahlen oder Satznamen bzw. Kennworten möglich sind, oder der Text Leerstellen, oder die Zeichen `'`, `;` oder `'$'` enthält, ist der Text in einfache oder doppelte Hochkomma (Apostrophe) einzuschließen.

```
ALFA  KNOT  1S45  'A B'  '1.45'  'A;B'  "Ab"
```

Mit Version 2010 sind alle internationalen Zeichen (UTF8) verwendbar, frühere Versionen waren auf die lokale code-page beschränkt. Dafür sind aber bei den kurzen Schlüsseln mit genau 4 Zeichen nur die ANSI-Zeichen (also keine Umlaute oder Akzente) erlaubt.

Kleinbuchstaben werden automatisch in Großbuchstaben umgesetzt, sofern sie nicht innerhalb zweier Apostrophe stehen. Es gibt einige wenige Elemente bei denen diese Umsetzung in jedem Falle erfolgt (z.B. Namen von Einwirkungen) um Mehrdeutigkeiten zu vermeiden, diese werden in der Spalte Dimension mit einem **LITnn** anstelle eines **Litnn** markiert.

Wenn ein Literal eine Variable enthält (siehe), so sind drei Fälle zu unterscheiden:

- Das Literal wurde mit Hochkommata definiert und beginnt mit einem Gleichheitszeichen (=). In diesem Falle wird das Literal z.B. für Formeln wie angegeben aber ohne das führende = weitergegeben. Dies ist nur sinnvoll, wenn das Programm an dieser Stelle das Literal auch in dieser Form verarbeiten kann.
- Die Variable wurde mit einem Literal belegt, dann wird das Literal der Variablen an dieser Stelle eingesetzt.
- Die Variable wurde mit einem Zahlenwert besetzt, dann wird dieser Zahlenwert formatiert in das Literal eingesetzt.

Bei einem optionalen Literal anstelle eines Zahlenwertes wie auch bei einer Text-Variablen ist per default nur die Form `#name` ohne jede Arithmetik oder Indizierung erlaubt (z.B. `A#X` mit `#X=25` ergibt `"A25"`). Wenn man eine Zahl mit Arithmetik verwenden möchte, muss man der Formel ein Gleichheitszeichen voranstellen. Aus `#X/2` wird also das Literal `„25/2“`, aus `=#X/5` wird die Zahl 12.5.

Innerhalb von echten Literalen steht hingegen die Funktion `#()` zur Verfügung. Dabei kann man das Format als zweiten Parameter angeben. z.B. `##(ALPHA,8.3)`. Der ganzzahlige Teil des Formats definiert die Spaltenbreite, die Zehntel definieren die Nachkommastellen, hier also für 8 Spalten mit 3 Nachkommastellen. Das erste Argument darf ein beliebig komplexer arithmetischer Ausdruck sein, deshalb muss das `#` Zeichen quasi wiederholt werden.

8.2 Eingabesyntax

8.2.1 Grundsätzliche Eingabeform

- Die Eingabe besteht aus Sätzen.
- Der Eingabesatz besteht aus dem Satznamen und ihm folgenden Datenwerten in der Reihenfolge der Eingabebeschreibung.
- Jeder Eingabesatz erstreckt sich über eine Eingabezeile.
- Die Angabe des Satznamen kann entfallen, wenn ein Satz derselben Satzart vorausgeht.
- Der Satz kann an beliebiger Stelle der Zeile beginnen.
- Die Daten werden durch ein oder mehrere Trennzeichen voneinander getrennt.

8.2.2 Standardwert

Wird anstelle eines Datenwertes das Minuszeichen (-) angegeben, so wird vom Programm für die betreffende Eingabe der in der Programmbeschreibung festgelegte Standardwert eingesetzt. Diese Eingabe wurde mitunter zum Überspringen von Kennworten verwendet. Da dies aber stark von der Reihenfolge der Kennworte abhängig ist, die sich mitunter ändern kann und sollte diese Form nicht mehr verwendet werden.

Werden für einen Satz weniger Daten angegeben als in der Beschreibung vorgesehen, so wird für die fehlenden Daten der jeweilige Standardwert angenommen.

Das Minuszeichen muss von Trennzeichen eingeschlossen werden.

Standardwerte können innerhalb einer Tabellenüberschrift umdefiniert werden.

8.2.3 Wiederholung

Wird anstelle eines Datenwertes das Gleichheitszeichen (=) angegeben, so wird vom Programm für die betreffende Eingabe der Wert des direkt vorausgegangenen Satzes wiederholt. Dies ist jedoch nur möglich, wenn der Satzname sich nicht geändert hat.

Werden zwei Gleichheitszeichen (==) (ohne Leerstelle dazwischen!) angegeben, so werden dieser und alle nach diesem noch folgenden Werte wiederholt. Eine Positionierung im vorausgegangenen Satz wird nicht übernommen.

War der Wert des vorausgegangenen Satzes eine Generierungsanweisung oder Wertreihe, so wird die Generierungsanweisung bzw. Wertreihe wiederholt.

Die Zeichen für Wiederholung müssen in Trennzeichen eingeschlossen werden.

8.2.4 Inkrementierung/Dekrementierung

Wird anstelle eines Datenwertes ein doppeltes Plus- oder Minuszeichen (++ oder --) angegeben, so wird vom Programm für die betreffende Eingabe der Wert des direkt vorausgegangenen Satzes um 1 erhöht bzw. erniedrigt. Dies gilt auch für Literale sowie Wertreihen und Generierungen. Dies ist jedoch nur möglich, wenn der Satzname sich nicht geändert hat.

Die Zeichen für Inkrementierung/Dekrementierung müssen in Trennzeichen eingeschlossen

werden.

8.2.5 Kommentar

Die Kommentarzeichen \$, ! oder // bewirken, dass der Rest der Zeile als Kommentar angesehen und nicht weiter interpretiert wird. Das Kommentarzeichen sollte sowohl vom Kommentar als auch von der Zeile davor mit einem Trennzeichen abgesetzt werden.

Innerhalb einer +PROG-Zeile (siehe SPS) ist nur das \$-Zeichen als Kommentar erlaubt, dort muss es mit Trennzeichen abgesetzt werden. Innerhalb einer +SYS-Zeile sind gar keine Kommentare erlaubt.

8.2.6 Satzfortsetzung

Die beiden Sonderzeichen \$\$ bedeuten, dass der Eingabesatz in der nächsten Zeile fortgesetzt wird.

Die nach \$\$ stehenden Zeichen werden als Kommentar interpretiert.

Vor dem \$\$ Zeichen muss ein Trennzeichen angeordnet werden. Somit können keine Ausdrücke oder Wertreihen über mehrere Zeilen gebildet werden.

8.2.7 Satztrennung

Mehrere Sätze können in einer Zeile angeordnet werden.

Die Sätze sind durch den Strichpunkt (;) voneinander zu trennen.

Der Strichpunkt muss nicht von Trennzeichen eingeschlossen werden.

Dies gilt nicht für die Eingabe von Textzeilen (z.B. KOPF).

8.2.8 Positionierung

Durch die Angabe eines Kennwortes kann innerhalb eines Satzes auf die entsprechende Position gesprungen werden.

Beispiel:

Ein Satz sei mit den Kennworten K1 K2 K3 K4 K5 definiert. Dann sind folgende Eingaben gleichwertig:

```
SATZ 1.00 - - 2.00 3.00
SATZ 1.00 K4 2.00 3.00
SATZ K1 1.00 K4 2.00 K5 3.00
SATZ K4 2.00 K1 1.00 K5 3.00
```

8.2.9 Tabellendefinition

Durch die Angabe eines Satzes, der nur aus Kennworten besteht, kann die voreingestellte Reihenfolge der Eingabewerte verändert werden. Die nachfolgenden Sätze dürfen dann keine Satznamen enthalten, da diese die alte Reihenfolge wieder herstellen.

Das letzte Beispiel könnte also auch wie folgt eingegeben werden:

SATZ	K1	K4	K5
1.00	2.00	3.00	

Die Tabelle bleibt erhalten, bis ein neuer Satzname angegeben oder der gleiche Satzname wiederholt wird. Innerhalb der Tabelle kann positioniert werden, auch auf Kennworte, die in der Überschrift nicht angegeben waren.

Innerhalb des Tabellenkopfs können die Standardwerte umdefiniert werden, indem hinter dem Kennwort ohne weitere Trennzeichen ein Gleichheitszeichen (=) und der neue Standardwert folgen.

SATZ	K1	K2	K5=4
1	2	6	
2	5		
3	6	-	

ist gleichwertig zu

SATZ	K1	K2	K5
1	2	6	
2	5	4	
3	6	4	

Eine solche Zuweisung kann auch eine elegante Umgehung von mehrdeutigen Literalen sein. Wenn es einen Literalwert (z.B. GRP) an der ersten Eingabeposition gibt, der auch einen Satznamen darstellt, so kann man durch Umordnung und eventuell mit einer Zuweisung dieses Problem ohne die Verwendung von Apostrophen umgehen:

STAB	BIS	TYP	PA	VON=GRP
1		TEMP	30	
2		TEMP	25	

8.2.10 Help-Satz

Durch die Eingabe von HELP wird die Liste der in diesem Modul möglichen Satznamen ausgegeben. Durch HELP gefolgt von einem Satznamen wird die Liste der Kennworte des Satzes ausgegeben. Durch HELP gefolgt von einem Satznamen und einem Kennwort werden die zulässigen Eingabewerte einer Position ausgegeben.

Diese Option ist vor allem beim Fehlersuchen im interaktivem Betrieb sinnvoll.

8.2.11 Generierung

An Stelle eines Datenwertes kann eine Generierungsvorschrift eingegeben werden. Dadurch werden eine Reihe von Daten automatisch erzeugt.

Die Generierungsvorschrift muss in Klammern eingeschlossen werden. Ihre Elemente müssen durch Trennzeichen oder Unterstreichungsstriche voneinander getrennt werden.

Es gibt zwei Arten von Generierungsvorschriften:

führende Generierungsvorschrift

(Anfangswert Endwert Inkrement)

zugehörige Generierungsvorschrift

(Anfangswert Inkrement)

Die führende Generierungsvorschrift bestimmt die Anzahl der zu generierenden Daten und darf pro Satz nur einmal, aber an beliebiger Stelle vorkommen. Die zugehörige Generierungsvorschrift darf bei jedem Wert definiert werden.

Eine Generierung von Literalen ist ebenfalls möglich. Dabei ist das Inkrement ebenfalls als Literal interpretiert und darf nur Zahlen enthalten.

Das Inkrement darf auch negativ sein.

Bei einer langen Generierung können Rundungsfehler entstehen, die nur sehr schwer zu identifizieren sind. Wird daher der angegebene Endwert einer führenden Generierung nicht mit einer Genauigkeit von 0.0001 der Schrittweite erreicht, oder ergeben sich mehr als 9999 Iterationen so erfolgt eine Fehlermeldung und die Generierung wird nicht ausgeführt.

Beispiel:

KNOT (1 5 1) 0 (0.5 -0.1)

erzeugt die Eingabesätze:

KNOT	1	0	0.5
KNOT	2	0	0.4
KNOT	3	0	0.3
KNOT	4	0	0.2
KNOT	5	0	0.1

Beispiel:

BEW (A0.1 D0.4 10.1)

erzeugt die Eingabesätze:

BEW	A0.1
BEW	B0.2
BEW	C0.3
BEW	D0.4

8.2.12 Wertreihe

Anstelle einer Generierung kann auch eine Wertreihe angegeben werden. Das ist eine Aufzählung von Werten, durch Komma oder Unterstreichung getrennt.

Wert, Wert, Wert, Wert z.B. 1, 2, 7, 9

Die Wertreihe darf keine Trennzeichen enthalten und kann nicht in einer Fortsetzungszeile fortgeführt werden.

Sind mehrere Wertreihen in einem Satz definiert, so müssen sie in der Anzahl der Werte sowohl untereinander als auch mit einer eventuell definierten führenden Generierung übereinstimmen.

Die maximale Anzahl von Werten pro Reihe ist durch einen internen Speicher begrenzt. In der Regel sind jedoch mindestens 25 Werte möglich.

Eine Wertreihe kann auch mit festen Literalen (LIT) gebildet werden. Eine Mischung von Zahlen und Literalen ist jedoch ebenso wenig zulässig wie eine Liste von freidefinierten Literalen.

8.2.13 Einheiten Konvertierung

An jeden Zahlenwert, Wertreihe oder Generierung ist es grundsätzlich möglich die verwendete Einheit explizit in eckigen Klammern anzuhängen. Wenn also eine Eingabe einer Länge in m erwartet wird, so kann man statt 0.3048 auch 304.8[mm] oder 12[in] angeben. Diese Option wird unterstützt für explizite und implizite Einheiten (siehe Abschnitt Einheiten). CADINP überprüft, ob die angegebene Einheit zulässig ist, d.h. zur Familie verwandter Einheiten gehört.

Wird eine Einheit in einem Tabellenkopf definiert, so gilt sie als Voreinstellung für die ganze Tabelle, kann aber jederzeit explizit überschrieben werden. Soll nur die Einheit vorgegeben werden, so kann der Zahlenwert entfallen, also z.B.

```
KNOT X=[m] Y=5.0[m] Z=[mm]
```

8.2.14 LET - und STO - Variable

Es können Variable gesetzt und in folgenden Sätzen verwendet werden. Nach Angabe von LET# (gilt nur lokal in diesem Programm) bzw. STO# (Wert der Variablen wird in der Datenbank gespeichert) folgt unmittelbar der Name der Variablen (bis zu 16 Zeichen, 1. Zeichen muss ein Buchstabe sein) und einem optionalen Feldindex. Für temporäre Variablen kann statt des Namens auch eine positive Zahl verwendet werden. Diese Variablen können dann beliebig anstelle von Zahlen verwendet werden. Nach einem Trennzeichen steht der Wert, der der Variablen zugewiesen werden soll. Dieser kann numerisch sein oder ein Text, der in Apostrophen eingeschlossen wird. LET# bzw. STO# verändern nicht den aktuell definierten Satznamen und müssen daher bei jeder Zuweisung angegeben werden. Variablennamen dürfen keine Sonderzeichen enthalten oder mit reservierten Namen der arithmetischen Funktionen (z.B. SIN oder SIN30) kollidieren. Anstelle des Zahlenwertes darf auch eine Wertreihe oder eine führende Generierung oder ein Literal stehen. In diesem Fall werden die Variablen, die der angegebenen Variablen folgen, mit den weiteren Werten belegt. Dies entspricht einem Feld einer benannten Variablen.

Die Einheit der Variablen ist abhängig vom Kontext wo sie verwendet wird. So kann eine Variable normalerweise nicht gleichzeitig für eine Querschnittsdefinition in [mm] und eine Systemdefinition in [m] verwendet werden. Eine Variable kann aber eine Einheit erhalten, dann wird der angegebene Wert unmittelbar konvertiert und kann an beliebiger Stelle weiterverwendet werden. Wenn eine Variable eine Einheit erhalten hat, so wird diese an alle daraus abgeleiteten Variablen und Einheiten vererbt. Dabei wird natürlich geprüft ob die resultierende Einheit an dieser Stelle zulässig ist. Wenn z.B. an einer Stelle eine Spannung erwartet und #P/#A defi-

niert wird, so können beide Variablen keine Einheit haben, #P als Kraft und #A als Fläche oder #P als Kraft pro Länge und #A als Breite definiert sein. Wird zu einer Variablen mit Einheit ein Skalar addiert oder subtrahiert, wird der Skalar in der bei der Variablen hinterlegten Einheit interpretiert.

Ein Array von Variablen hat insgesamt immer die Einheit, die dem Element mit dem Index 0 zugewiesen wurde.

Eine Variable wird dadurch verwendet, dass man sie mit einem explizit vorangestellten # Zeichen eindeutig als Variable kenntlich macht. Wenn man dem Ausdruck zusätzlich ein = Zeichen voranstellt, so werden auch alle unbekanntenen Zeichenfolgen als Variablennamen interpretiert.

```

LET#PI 3.1415      ! Zuweisung an Variable #PI
LET#TXT 'Mein Text' ! Zuweisung eines Literals

LET#11 4,5,6      ! Zuweisung an Variablen 11,12,13
LET#A 4,5,6       ! Zuweisung 4,5,6 an das Feld A[0:2]
LET#A(2) 5.1      ! Zuweisung an einzelnes Feldelement
LET#A(3) 7,8      ! Erweiterung um 2 Feldelemente
LET#1 =A(1)       ! Zugriff auf das zweite Feldelement
LET#1 #(A+1)      ! Zugriff auf das zweite Feldelement

LET# #10 12.50    ! Zuweisung 12.50 an Variable mit
                  ! der Nummer, die derzeit in #10
                  ! abgelegt ist (Indizierung)

LET#4 ##10        ! Zuweisung der Variablen mit Nummer
                  ! die in Variable 10 definiert
                  ! ist an die Variable 4

! Deklaration Feld, Füllen mit Werten über Fortsetzungszeilen
LET#f(100) 0
LET#F 1,2,3,4,5,6,7,11,12,13,14,15,16,17,21,22,23,24,25,26,27

LET#B 20[mm]      ! Zuweisung mit Einheit
LET#D 1.5[m]      ! Zuweisung mit Einheit
LET#A #B*#D       ! Zuweisung mit Einheit: #A is 0.3[m2]
LET#X #B*#D[-]    ! Zuweisung ohne Einheit: #A is 0.3

```

Taucht eine Variable, die zugewiesen wird, auch im Ausdruck auf, so wird erst der Ausdruck berechnet und dann der neue Wert zugewiesen.

Variablen können mit dem Kommando PRT# für Testzwecke in den Echoprint der Eingabe ausgegeben werden. Der Name der Variablen wird ohne Trennzeichen unmittelbar hinter dem # erwartet. Komfortabler ist jedoch die Ausgabe mit TXA / TXE oder <TEXT> innerhalb eines Literaltextes.

Benannte Variablen können auch dauerhaft in der Datenbasis gespeichert werden. Mit dem

Befehl `STO#name` wird die Variable `name` mit ihrem aktuellen oder dem danach wie bei `LET#` angegebenem Wert gespeichert und für alle anderen Programme danach verfügbar. Z.B.

`STO#C 30 ! Zuweisung und Abspeichern der Variablen in der Datenbasis`

Möchte man hingegen eine gespeicherte Variable wieder löschen, so erfolgt dies mit dem Befehl `DEL#name`. In diesem Falle sind auch sogenannte Platzhalter erlaubt. Also `DEL#OPT*` oder `DEL#A?00` löschen alle Variablen die diesem Muster entsprechen. Der Gebrauch von `DEL#` innerhalb von Kontrollstrukturen wie Schleifen oder IF-Bedingungen kann zu unerwarteten Effekten führen.

Für Sonderfälle kann man eine benannte Variable mit ihrem zuletzt gespeicherten Wert reinitialisieren:

<code>RCL#ALL</code>	Einlesen aller benannten Variablen
<code>RCL#name</code>	Einlesen einer benannten Variablen/Feldes
<code>RCL#name(3)</code>	Einlesen eines Eintrags eines Feldes
<code>RCL#name cdbfile</code>	Einlesen Variable aus einer anderen CDB

Reservierte Variablen-Namen

Es gibt einige reservierte Namen, die von den Programmen selbständig gesetzt werden. Der Benutzer kann diese Variablen innerhalb einer CADINP-Eingabe natürlich mit eigenen Werten wieder überschreiben.

- Die Variablen `VERSION(0)` und `VERSION(1)` werden vom Programm mit der Version der DLL und des eigentlichen Programms vorbesetzt, so dass man Eingaben abhängig von der Versionsnummer beschreiben kann.
- Die Variable `PI` wird mit 3.141593 vorbesetzt.
- Die Variablen-Arrays `GRP_MASS`, `SCT_MASS` und `MAT_MASS` sowie `GRP_REIN` und `SCT_REIN` sind nach einer Ausgabe der Systemstatistik ebenfalls definiert. Sie enthalten die Massen der Elemente (`MASS`) und der Bewehrungen (`REIN`) aufgliedert nach Gruppen (`GRP_`) und Querschnittsnummern (`SCT_`) oder Materialien (`MAT_`), wobei der Index 0 jeweils für die Gesamtwerte steht.

Alle anderen Variablen beginnen mit den ersten drei Buchstaben des entsprechenden Programms gefolgt von einem Unterstrich:

- ASE speichert in `ASE_ITER` die Werte:
 - `ASE_ITER(0)` = erste Lastfallnummer
 - `ASE_ITER(1)` = letzte erreichte Lastfallnummer
 - `ASE_ITER(2)` = letzter Lastfaktor
- Die Variablen des Arrays `AQB_USAGE` werden von `AQB` mit den Ausnutzungsgraden der Aufgaben des letzten Eingabeblocks besetzt.
- Alle Variablen beginnend mit `OPT_` sind für `OPTIMA` reserviert.

Für eine Verfolgung der Variablen gibt es ein Kommando `DBG#`. Damit können Testausgaben und ein interaktiver Debug-Modus gesteuert werden. `DBG#` verwendet die Variable `#0`, die

deshalb nicht anderweitig verwendet werden kann.

DBG#0	Keine Ausgabe von Zwischenwerten
DBG#1	Ausgabe der erzeugten Eingabesätze
DBG#2	Zusätzliche Ausgabe aller Zuweisungen
DBG#3	Zusätzliche Ausgabe der ausgewählten Strukturen (CDB Zugriff)
DBG#4	Ausgabe auf Konsole umleiten
DBG#8	Eingabe von Konsole (interaktiver Modus)
DBG#	Umschalten zwischen Option 15 und Option 0 (=halt und continue)

sowie

DBG# -2	Sofortiges Beenden des gesamten Programmlaufs wobei anstehende TXE-Texte nach der Fehlermeldung noch gedruckt werden
---------	--

8.2.15 Arithmetische Ausdrücke

Anstelle eines Zahlenwertes kann ein beliebiger arithmetischer Ausdruck stehen. Der Ausdruck darf Klammern, aber keine Trennzeichen enthalten.

Als Operatoren sind zugelassen:

+	-	Addition, Subtraktion
*	/	Multiplikation, Division
**	oder ^ (Dächle)	Exponentiation
==	<>	Abfrage auf Gleich/Ungleich
>=	<=	Abfrage auf Relationen
>	<	Abfrage auf Relationen Ergebnis: Wahr (1.) oder Falsch (0.)
&		Bitweise logische Operation auf den ganzzahligen Anteil mit UND bzw. ODER

Sofern kein Operator angegeben ist, wird Multiplikation angenommen. Die Rangfolge der Operationen entspricht der mathematischen Konvention. Die logischen Operatoren sind alle gleichwertig, es wird nachdrücklich empfohlen, bei kombinierten Ausdrücken Klammern zu verwenden.

Als Funktionen innerhalb eines Ausdrucks sind zugelassen:

SIN(x), COS(x), TAN(x)	Trigonometrische Funktionen
ATN(x), ATN(y, x)	Arcus-Tangens x oder y/x
ARC(x)	Winkelargumente im Bogenmaß
SQR(x)	Quadratwurzel
ABS(x)	Absolutwert
EXP(x)	Exponentiation zu e
LOG(x)	Natürlicher Logarithmus

LGT(x)	Logarithmus zur Basis 10
DIV(x,y), xDIVy, DIV(x/y)	Ganzzahliger Anteil x/y
MOD(x,y), xMODy	Divisionsrest x/y
MOD(x)	Nachkommastellen von x
MIN(x,y,...), MAX(x,y,...)	Minimum oder Maximum
RANDOM(x)	Zufallszahl zwischen 0 und 1 (x=0 reinitialisiert Sequ.)
IIF(expr, val1, val2)	Setzt den Wert val1 if expr ungleich Null ist, und val2 wenn expr==0 ist.

Arithmetische Ausdrücke können auch innerhalb von Wertreihen oder Generierungen erscheinen. Die Kennworte DEG, GON und RAD können jederzeit durch Komma getrennt vor das Argument gestellt werden. Ihre Definition bleibt erhalten wenn man in einer Zuweisung nur diesen Text angibt (z.B. LET#0 RAD).

Beispiele:

```
SIN(30.)+3*COS(45.)  oder SIN30+3COS45
SIN(RAD, 2.435)
```

```
100.+MOD(354, 32)    oder 100+354MOD32
```

```
120.+12.
3(5.0+4.0)
```

```
COS(#1) SIN(#1)
345*#11+##12
```

Interpolation und Tabellen:

Ein besonderes Feature steht beim Zugriff auf Variablen-Felder zur Verfügung. Wenn der Index gebrochen angegeben wird, so wird zwischen den Werten des Feldes interpoliert:

```
LET#A(0) 10.0
LET#A(1) 14.0
LET#A(2) 16.0
LET#A(3) 17.0
```

```
LET#B =A(1.3) => #B = 14.6
```

Analog kann man auch komplexere Interpolationen definieren. Dazu benötigt man zwei gleichlange Felder von X und Y-Werten. Diese werden dann mit einer speziellen Zuweisung über ein Literal zu einer Tabellenfunktion zusammengefasst:

```
LET#X 0.0, 2.0, 3.5
LET#Y 0.0, 100.0, 100.0
LET#SIG 'TAB(X, Y)'
```

Der Ausdruck #SIG(1.73) interpoliert dann für diesen X-Wert in der Tabelle der y-Werte linear. Möchte man höhere Funktionen bei der Interpolation benutzen, so kann man ein drittes Feld mit den Ableitungen angeben:

```
LET#DY -,0,-
LET#SIG 'TAB(X,Y,DY)'
```

In obigem Beispiel wurde nur die Ableitung für den mittleren Punkt definiert, somit ergibt sich ein parabelförmiger Verlauf der Funktion. Falls die Ableitung an beiden Enden eines Intervalls definiert ist, ergibt sich ein kubischer Spline als Funktionsverlauf.

Variablen und Literale:

Für den Fall, dass man einen Text in einer Variablen abspeichern möchte, kann man dies ebenfalls mit einer LET/STO Anweisung erledigen (Die Verwendung der Apostrophe ist zwingend):

```
LET#TEXT 'ABCDEFGHIJK'
```

Der Text wird mit jeweils 8 Buchstaben in einem Speicherort der Variablen abgelegt, #TEXT(1) wäre im obigen Beispiel also "IJK" und kann auch einzeln abgeändert werden. Die Speicherung einzelner Buchstaben ist jedoch nicht möglich. Bei weiteren Zuweisungen kann man jedoch auch Teilstrings in der Form ansprechen, dass man #TEXT(3:7) das dritte bis siebte Zeichen anspricht. (Statt der Zahlen sind natürlich auch arithmetische Ausdrücke erlaubt). Eine Textvariable kann aus der CDB gelesen werden und man kann bei einer LET/STO Anweisung auch einen Text in eine oder mehrere Zahlen umwandeln, im folgenden werden die beiden Zahlen den Variablen #VALT(0) und #VALT(1) zugewiesen:

```
LET#TEXT '1.23,1.48'
LET#VALT VAL(#TEXT)
```

8.2.16 FUN - Definition von Funktionen

Falls man einen arithmetischen Ausdruck häufiger verwenden will oder die Bedeutung einer Variablen umdefinieren möchte, kann man auch Funktionen definieren. Eine Funktion wird als Literal definiert:

```
LET#F '=FUN(var,formelAusdruck)'
! z.B.
LET#F '=FUN(x,3*#x**3-2*#x**2+5*#x)''
LET#1 #F(1.234)
```

Das Literal muss mit der Zeichenfolge "=FUN(" beginnen, danach folgt der Name des Parameters, danach ein FormelAusdruck der alle definierten Variablen enthalten darf. Sofern die verwendete Variable schon definiert ist, wird diese durch den Aufruf nicht verändert. Rekursive Aufrufe sind möglich.

8.2.17 LOOP, ENDLOOP - Schleifen und Sprünge

Die mächtigste Generierungsart wird durch Schleifen zur Verfügung gestellt. Diese entsprechen den DO-Loops aus FORTRAN bzw. FOR NEXT Schleifen aus Basic. Die Schleife beginnt mit einem Satz LOOP und endet mit einer ENDLOOP Anweisung. Die Schleife wird so oft durchlaufen, bis entweder die nach LOOP angegebene Anzahl von Durchläufen (Default 9999) erreicht ist, oder der nach ENDLOOP stehende Ausdruck Null oder negativ wird. Wird für die Anzahl der Schleifen nach LOOP der Name einer Variablen angegeben, so wird die Anzahl der Werte in dieser Variablen verwendet.

Die Schleifen können bis zu einer Tiefe von 32 geschachtelt werden und dürfen beliebige Eingabeelemente enthalten. Sofern nicht bei LOOP ein Wert definiert wurde, wird eine Schleife maximal 9999 mal durchlaufen.

Jede Schleifenkonstruktion darf nicht mehr als 256 Zeilen enthalten. Innerhalb der Zeilen können jedoch auch mehrfache Sätze definiert werden (getrennt mit ;). Möchte man mehr als 256 Zeilen verwenden, so muss man vor der ersten Verwendung eines Loops definieren:

```
LET#LOOPSIZE anzahl_der_Zeilen
```

Es ist möglich, den Index der Schleife auf eine Variable zu speichern, indem man unmittelbar an LOOP anschließend den Namen der Variablen angibt. Der Index beginnt bei Null. Die Variable darf innerhalb des Loop beliebig verändert werden, sie wird nach Auswertung der Ende-Bedingung jeweils wieder restauriert. Generierung von Knoten und Federn auf einem Halb-Kreis mit Abstand von 30 Grad:

```
LET#1 1 , LET#2 0.
LOOP 7
KNOT #1 COS(#2) SIN(#2)
FEDE #1 #1 DX COS(#2) DY SIN(#2) CP 1.E5
LET#1 #1+1
LET#2 #2+30.
ENDLOOP
```

Statt LOOP 7 / ENDLOOP kann auch LOOP / ENDLOOP #2 < =180. verwendet werden. Mit einer Endabfrage kann man eine Schleife auch vorzeitig verlassen.

Beispiel für zweifach verschachtelte Generierung

```
LOOP#1 3
  TXB ADEF #1+1
  LOOP 2
    TXB BDIV 0.5 #1+1
    TXB      0.2 1
  ENDLOOP
ENDLOOP
```

erzeugt:

```
ADEF 1
```

```

BDIV 0.5 1
    0.2 1
    0.5 1
    0.2 1
ADEF 2
  BDIV 0.5 2
    0.2 1
    0.5 2
    0.2 1
ADEF 3
  BDIV 0.5 3
    0.2 1
    0.5 3
    0.2 1

```

Wenn man eine Schleife über alle Elemente eines Feldes machen möchte so kann man nur den Namen des Feldes (ohne #) angeben:

```

LET#A 10,22,34,55,76,83
LOOP#1 A ! Nur der Name, #A wäre der Wert 10!
  KNOT #1+1 X #A(#1)
ENDLOOP

```

Mit einer kleinen Erweiterung kann man die Anzahl der definierten Werte in einer Variable abspeichern, dabei erhält man 0 wenn die Variable nicht definiert ist:

```

LOOP#ANZ DEF(A)
ENDLOOP

```

8.2.18 IF - Logische Abfragen

Wichtiges Element einer Programmiersprache sind bedingte Teile (IF THEN ELSE oder CASE). Die Realisierung von Sprüngen ist in CADINP nicht möglich da diese mathematisch bewiesen nicht zwingend erforderlich sind. Ein IF-Block ist aktiv, wenn der dahinter stehende Ausdruck ungleich Null ist. Hilfreich sind hierzu die logischen Operatoren. Texte können nur auf == oder != abgefragt werden, dabei wird Groß und Kleinschreibung berücksichtigt und da die gesamten Strings rechts und links vom Vergleichsoperator verwendet werden, sind Klammern Teile des Textes und können daher nicht als operatoren verwendet werden.

```

! #1 sei die Steuervariable
IF #1
  Diese Zeilen werden Eingabe, wenn #1 > 0 ist
  ...
ELSE
  Diese Zeilen werden Eingabe, wenn #1 = oder < 0 ist
  ...
ENDIF

IF #1==12
  Diese Zeilen werden Eingabe, wenn #1 gleich 12 ist

```

```

...
ELSE
    Diese Zeilen werden Eingabe, wenn #1 ungleich 12 ist
...
ENDIF
    
```

Die Erzeugung von CASE Konstruktionen erfolgt über eine Reihe von zusätzlichen ELSEIF-Statements:

```

if (Bedingung_1)
    ....
elseif (Bedingung_2)
    ....
elseif (Bedingung_3)
    ....
else
    ....
endif
    
```

Beispiel:

```

IF (#A < 0.3) ! Bedingung 1
    LET#WERT 0.50
ELSEIF (#A>1.0) ! oder Bedingung 2
    LET#WERT 0.70
ELSE
    LET#WERT 0.50+0.20*(#A-0.3) ! sonst dieser Wert
ENDIF
    
```

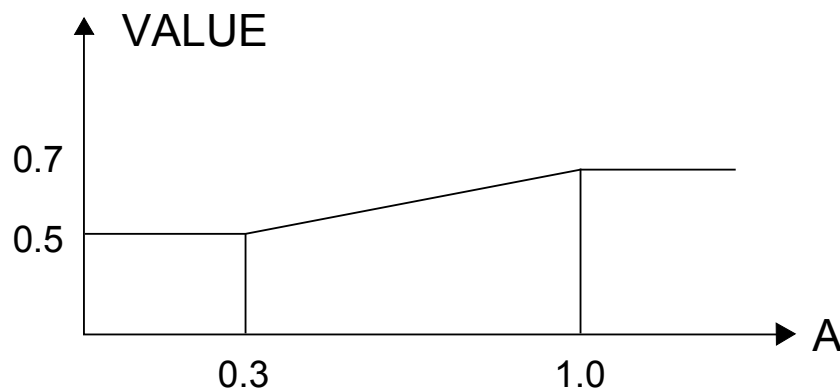


Abbildung 8.1: WERT in Abhängigkeit von A

D.h. es wird nur eine der Möglichkeiten für WERT in Abhängigkeit von A gewählt.

Für Spezialfälle lassen sich über ein entsprechendes Verzweigungs-konstrukt Eingabe erstellen, die versionsübergreifend in verschiedenen SOFiSTiK-Umgebungen lauffähig sind:

```

IF (#VERSION>=2012)
    !.....
    
```

```
ELSEIF (#VERSION==2010)
!....
ELSE // ältere SOFiSTiK Releases
!....
ENDIF
```

8.2.19 @CDB - Auswahl einer CDBASE

CADINP erlaubt auch den Zugriff auf Inhalte der Datenbasis. Dazu ist jedoch die Beschreibung der Datenbankinhalte (**cdbase.chm**) erforderlich. Die Syntax gliedert sich in drei Teile. Mit @CDB wird die Datei der Datenbasis ausgewählt:

```
@CDB dateiname
```

wählt eine beliebige Datenbank **dateiname.cdb** aus. Voreingestellt ist die aktuelle Projektdatenbank. Eine explizite Eingabe dieser Datenbank ist hingegen nicht erlaubt.

Die Daten in einer CDB bestehen aus einzelnen Sätzen mit gleichem Aufbau, die unter verschiedenen 64-Bit Indices KWH/KWL abgelegt sind. Grundsätzlich besteht ein solcher Datensatz aus einem ganzzahligen Bereich, der Nummern enthält und einem Gleitkommabereich, der sich daran anschließt. Die Beschreibung dieser Strukturen ist für den Benutzer in der Datei **cdbase.chm** bzw. für Programme in der Datei **cdbase.cdb** abgelegt.

8.2.20 @KEY - Zugriff auf CDBASE

```
@KEY KWH KWL [ SEL1 SEL2 SEL3 SEL4 SEL5 SEL6 IERR ]
```

wählt einen Zugriffsschlüssel (Kennwort) KWH/KWL entsprechend der CDBASE-Beschreibung aus. KWH ist eine bis zu 8-buchstabile Typbezeichnung und KWL ist eine optionale Nummer (z.B. Lastfall- oder Querschnittsnummer). Die Werte KWL, SEL1 bis SEL6 werden entsprechend vorbesetzt.

Für besondere Zwecke kann man auch einen optionalen Filter mit bis zu sechs ganzzahligen oder 4-buchstabigen Schlüsselwerten SEL1 bis SEL6 auswählen, dabei müssen Schlüssel, die nicht signifikant sind oder erst beim Zugriff definiert werden, mit -1 eingegeben werden. Die Schlüssel entsprechen den Integer-Positionen der Datenbank.

Das Kennwort muss in der Datenbank vorhanden sein, anderenfalls erfolgt eine Fehlermeldung. Der Lesezeiger wird an den Anfang des Kennworts gesetzt. Wird bei IERR jedoch die Nummer einer Variablen (z.B. 999) angegeben, so wird in dieser ein Wert >0 gesetzt, wenn der Schlüssel vorhanden ist. Alternativ kann man die Variable #CDB_IER setzen und abfragen.

8.2.21 @() - Zugriff auf CDBASE

Der eigentliche Zugriff innerhalb eines Satzes erfolgt dann mit einer Funktion @() im Rahmen eines arithmetischen Ausdrucks.

```
@(position+offset) or @(no,position+offset)
```

Liest aus der Datenbasis den nächsten Record, der durch den @KEY-Filter und im zweiten Fall ergänzt um "nr" auf dem letzten definierten SELi selektiert wurde. Der Wert des Ausdrucks ist der gespeicherte Wert mit dem Namen "position", oder falls position eine Zahl ist, an der Stelle die um den ganzzahligen Wert "position" relativ zu dem letzten definierten Selektionswert steht. Der Wert 0 bzw. negative Positionswerte liefern folglich Integerwerte aus dem Selektionsbereich der Schlüssel. Mit "offset" kann auf den Index eines Feldes zugegriffen werden.

Beim Zugriff wird immer an der aktuellen Position begonnen. Wird also ein Wert an einer gleichen oder kleineren Position gesucht, so erhält man automatisch den nächsten Record. Wenn das Ende eines Schlüssels dabei erreicht wird, erhält man eine Fehlermeldung, wenn der Schlüssel über @KEY bereits vollständig definiert wurde @(position), wohingegen bei @(nr, position) mit dem Lesen einmal wieder von vorne begonnen wird.

Man kann jedoch auch zwei Status-Variablen definieren. Wenn diese Variablen nicht negativ definiert sind, werden die angesprochenen Werte dort abgelegt und die entsprechende Fehlermeldungen entfallen:

- CDB_LEN für die tatsächliche Länge des eingelesenen Satzes
- CDB_IER für die Fehlerbedingungen (0=ok, 1=record_zu_kurz, 2=Dateiende, 3=Key_nicht_definiert).

Dies ist erforderlich um zu verhindern, dass das Programm in eine unendliche Schleife geraten kann.

Um den mitwirkenden I_y-Wert eines Querschnitts 5 aus der aktuellen Projektdatenbank zu verwenden, ist einzugeben:

```
@KEY SECT_EFF 5
LET#5 @(IY)
```

Um Schnittgrößen der Knoten 101 und 102 der Gruppe 2 des Lastfalls 12 zu verwenden, ist einzugeben:

```
@CDB PROJEKT1
@KEY QUAD_NFO 12 2 ! Selector Gruppe 2
LET#101 @(101,MXX) ! Moment m-xx
LET#102 @(102,MYX) ! Moment m-yy
```

Um alle Auflagerkräfte der Knoten mit Endziffer 0 als Lasten zu erhalten, ist einzugeben:

```
LET#CDB_IER 0
@CDB PROJEKT1
@KEY N_DISP 12
LOOP ! Alle Saetze
  LET#10 @NR,@PX,@PY,@PZ
  IF (#CDB_IER < 2)&(#10 <> 0) ! Ende oder Header
    IF 0==MOD(#10,10) ! Endziffer 0
      LAST #10 PX #11 #12 #13
    ENDF
```

```

ENDIF
ENDLOOP #CDB_IER < 2           ! Sonst kein Ende

```

Um die Querschnittsnummern eines Stabes 101 zu erhalten (Da hier zwei verschiedene Schlüssel im Wechsel gelesen werden, kann die Literaleingabe hier nicht verwendet werden):

```

LET#CDB_IER 0
@CDB PROJEKT1
@KEY 100 0 -1 -1 -1 -1       ! 4 Integer-Keys
LET#NR 0
LOOP                          ! Alle Sätze
  LET#10 @(-3),@(-2)         ! 1. und 2. Integer
  IF #10 ; LET#NR #10        ! Stabsatz
  ELSE                         ! Abschnittssatz
    IF #NR==101
      LET#Q #11
    ENDIF
  ENDIF
ENDLOOP #CDB_IER < 2       ! Bedingung - Ende

```

Für Texte in der CDB muss beachtet werden, dass eine richtige Verarbeitung nur mit Kenntnis der Datenstruktur bzw. der in der in der **cdbase.chm** definierten Namen erfolgen kann:

```

! Einwirkung und Bezeichnung des Lastfalls 101
@KEY LC_CTRL 101
LET#ACT LIT(@ITYP)
LET#TXT LIT(@RTEX)

! Bezeichnung des Materials 1
@KEY MAT 1
LET#MATTXT LIT(@TITLE)

```

8.3 Generelle Satznamen

Die Eingabe erfolgt in Blöcken. Jeder Eingabeblock beginnt mit den eventuellen KOPF-Zeilen und endet mit einem ENDE Satz. Einige Programme erlauben die Angabe mehrerer Blöcke hintereinander. Die Berechnung wird dann beendet, wenn ein leerer Block (ENDE/ENDE) erkannt wird.

In der Eingabebeschreibung werden für alle Satznamen die gleichen Tabellen verwendet. Neben dem Satznamen in einem extra Kästchen oben links gibt es vier Spalten.

- Die erste Spalte (Wert) gibt die Kennworte des Satzes an.
- In der zweiten Spalte stehen die Erläuterungen und eventuell zulässigen Literale.
- In der dritten Spalte ist die Dimension des Eingabewertes. Ein "-" Zeichen steht für dimensionslose Werte, ein "*" steht für Werte mit unterschiedlichen Möglichkeiten, die z.B. über SEIT UNIE ausgewählt werden. **LIT** steht hier für Werte, bei denen nur feste Literale

erlaubt sind. **Lit***nn* bezeichnet freie Eingabewerte, die bis zu *nn* Stellen beliebigen Text haben können. **LIT***nn* steht für freie Literale, die zwangsweise in Großbuchstaben konvertiert werden.

- Die letzte Spalte schließlich gibt die Voreinstellungen an. Hier bedeutet das "-" Zeichen, dass keine Voreinstellung existiert, der Wert braucht auch nicht definiert zu werden. Hingegen bedeutet ein "!", dass der Wert unbedingt angegeben werden muss. Ein "*" schließlich steht immer dort, wo die Voreinstellung sich nach weiteren Umständen richtet, die im Text erläutert werden.

Einige Satznamen haben in allen Programmen die gleichen Bedeutungen. Diese sind:

8.3.1 KOPF – Überschriftszeilen

KOPF

Wert	Bedeutung	Unit	Voreinst.
	Kopfzeile	LIT72	-

Es können bis zu 10 Überschriftszeilen definiert werden. Die erste wird für die Bauwerksbezeichnung reserviert und ist innerhalb der SOFiSTiK-Kette nur bei AQUA, SOFiMSHA/B/C und Templates definierbar. In allen anderen Modulen wird diese dann aus der Datenbasis übernommen und kann nicht mehr geändert werden. Die anderen sind frei und werden durch Angabe von KOPF gefolgt von einem Trennzeichen und beliebigem Text bis zu 72 Zeichen definiert. Werden innerhalb eines Eingabeblockes keine Angaben gemacht, so bleiben die Kopfzeilen erhalten. Über die Ausbildung des Seitenkopfes vgl. [SEIT](#). Die Kopfzeile wird ohne Literalzeichen (Apostroph) eingegeben. Voreinstellungen der Kopfzeilen können im SOFiSTiK.DEF mit den Variablen KOPF1, KOPF2 und KOPF3 definiert werden.

8.3.2 ENDE – Ende Eingabeblock

ENDE

Wert	Bedeutung	Unit	Voreinst.
	Ende der Eingabe oder des Eingabeblocks	–	-

Die Satzart ENDE beschließt jeden Eingabeblock. Innerhalb eines Eingabeblocks werden beispielsweise je ein Berechnungsfall definiert. Das Ende der gesamten Eingabe wird durch einen doppelten Satz ENDE definiert. Wenn die Eingabedatei physikalisch zu Ende ist, wird dieser Satz von CADINP erzeugt, falls er fehlen sollte.

8.3.3 TXA – Vorbemerkungen

TXA

Tabelle wird auf der Folgeseite fortgesetzt.

Wert	Bedeutung	Unit	Voreinst.
	Einführender Text	LIT72	-

8.3.4 TXE – Nachbemerkungen

TXE

Wert	Bedeutung	Unit	Voreinst.
	Abschließender Text	LIT72	-

Mit TXA und TXE lassen sich pro Eingabeblock beliebige erläuternde Texte vor die Berechnung (TXA) oder hinter die Berechnung (TXE) anordnen. Die Anzahl der Zeilen ist nicht begrenzt. Jeder dieser zwei Textbereiche kann mit größeren Textblöcken **<TEXT> ... </TEXT>** oder mit Bildern zwischen den Kennworten **<PICT>** und **</PICT>** ergänzt werden. Ein Bild darf jedoch nicht innerhalb eines **<TEXT> ... </TEXT>** Blocks auftreten.

Die TXA/TXE - Zeilen werden ohne Literalzeichen (Apostroph) eingegeben. Innerhalb der Textzeilen werden Ausdrücke der Form **#(Ausdruck,dd)** durch den aktuellen Wert der Variablen im Format **dd** ersetzt.

Innerhalb der Texte werden einige HTML-Komponenten ausgewertet. Eine HTML-Komponente ist dadurch gekennzeichnet, dass sie mit einem **<** beginnt, auf das weder ein Leerzeichen noch ein weiteres **<** folgt. SOFiSTiK interpretiert die folgenden Komponenten (andere werden ignoriert):

<FF>	Neue Seite
<LF>	Leerzeile
 ... 	Fettdruck
<i> ... </i>	Kursivdruck
<u> ... </u>	Unterstreichen

8.3.5 <TEXT> – Textblock

<TEXT>

Wert	Bedeutung	Unit	Voreinst.
	Überschrift eines Textblocks	LIT72	-

8.3.6 </TEXT> – Ende eines Textblocks

</TEXT>

Wert	Bedeutung	Unit	Voreinst.
	Ende eines Textblocks	LIT72	-

Größere Text-Blöcke können ohne Satznamen zwischen den besonderen Satznamen **<TEXT>** und **</TEXT>** definiert werden. Je nachdem ob vorher ein **TXA** oder **TXE** definiert wurde, werden sie diesem Bereich zugeordnet. Der Text hinter **<TEXT>** wird in das Inhaltsverzeichnis von Report Browser übernommen, sofern er definiert wurde.

Der Textblock kann in eine Datei geschrieben werden, dazu definiert man:

```
<TEXT,FILE=foobar.txt>
```

Soll ein weiterer Textblock der Datei hinzugefügt werden, so erfolgt dies über ein vorangestelltes +, z.B.

```
<TEXT,FILE=+foobar.txt>
```

Für die Erstellung von Templates kann man editierbare Felder mit speziellen HTML-Tags definieren:

```
<EDIT:name>$(name) </EDIT>
```

Editierbare Referenzen auf Parameter

```
<EDIT:name,Format=6.2>$(name) </EDIT>
```

Editierbare Referenzen auf Parameter mit einer Formatangabe für "###.##" (Gesamtstellenzahl.Nachkommastellen)
 Format=6 reserviert sechs Spalten,
 Format=-6 plaziert linksbündig

```
<EDIT:name,List=str1,str2,str3>$(name) </EDIT>
```

Editierbare Referenzen auf Parameter

mit einer Listbox möglicher Werte

`<EDIT:name,List=str1,str2,str3,Update=Yes>$(name) </EDIT>`

Erzwingt unmittelbar nach einer Eingabe ein Update des Templates

8.3.7 ECHO – Ausgabeumfang

ECHO

Wert	Bedeutung	Unit	Voreinst.
OPT	ECHO Option Dieser Wert beschreibt einen Bereich der Ausgabe oder Rechenoptionen, auf den sich dieser ECHO-Satz bezieht. Die aktuelle Liste der möglichen Optionen ist dem jeweiligen Handbuch zu entnehmen. VOLL alle Ausgabeoptionen	LIT	VOLL
WERT	Wert der ECHO-Option -1 oder AUS keine Berechnung 0 oder NEIN keine Ausgabe 1 oder JA normale Ausgabe 2 oder VOLL erweiterte Ausgabe 3 oder EXTR extreme Ausgabe	LIT	VOLL

Zu Beginn des Programms sind alle Optionen für neue Ergebnisse auf den Wert 1 voreingestellt. Die Ausgabe bereits gerechneter Werte (z.B. Knotenkoordinaten in Rechenprogrammen) ist mit 0 vorbesetzt. Mit Angabe von ECHO KNOT wird dann z.B. die Option KNOT auf den Wert 2 gesetzt. Die Voreinstellung für WERT wird also erst aktiv, wenn ein Satz ECHO angegeben wird. Die genaue Wirkung der Optionen ist der Ausgabebeschreibung der einzelnen Handbücher zu entnehmen. Generell ist zu bemerken, dass mit JA eine möglichst kurz gefasste Ausgabe erreicht wird, mit VOLL eine umfangreichere Ausgabe, und bei EXTR in der Regel Werte ausgegeben werden, die entweder weitere Berechnungsschritte nötig machen oder sehr viel Papier ergeben können. Die Angabe dieses Wertes sollte deshalb mit Vorsicht erfolgen.

Wünscht der Benutzer keine Ausgaben bestimmter Ergebnisse, so müssen diese explizit mit NEIN deaktiviert werden. Soll z. B. nur die Ausgabeoption REAK gesetzt sein, so muss eingegeben werden:

```
ECHO OPT VOLL WERT NEIN
ECHO REAK
```

8.3.8 UNIT – Einheiten für Ein- und Ausgabe

UNIT

Wert	Bedeutung	Unit	Voreinst.
TYPE	Nr eines Einheitensets (0-9) oder Nr oder Bezeichnung des impliciten items	<i>Lit16</i>	!
USE	Die Einheit die verwendet werden soll	<i>Lit16</i>	*
DIG	Anzahl der Nachkommastellen oder E1 bis E7 für Exponentialdarstellung	–	*
SET	Wirkungsbereich OUT Ausgabe IN Eingabe INOUEin- und Ausgabe	<i>LIT</i>	INOUE

SOFISTIK-Programme erlauben die Ein- und Ausgabe bestmöglich in ingenieurmäßigen Einheiten darzustellen. Die Datenbank ist dimensionsrein in den SI-Einheiten kN, m, sec angelegt. Der Einheiten für Ein- und Ausgabe wird bei der Auswahl einer Norm über einen globalen Set ausgewählt. Die Satzart UNIT definiert für den aktuellen Programmlauf temporär andere Einheiten für Ein- oder Ausgabe. Man hat die Möglichkeit, ,it TYPE komplett auf ein anderes Einheitenset umzustellen:

- 0 = Standardeinheiten (m, kN, sec mit historischen Abweichungen)
- 1 = deutscher Hochbau (Querschnitte in cm, System in m)
- 2 = deutscher Stahlbau (Querschnitte in mm, cm², dm⁴, System in m)
- 3 = Brückenbau (wie 0, aber Schnittgrößen in MN statt kN)
- 4 = Grundbau (m, kN, sec)
- 5 = Ingenieurbau (Querschnitte in mm, System in m)
- 6 = metrisches System (Alle Abmessungen in mm, Lasten in kN)
- 7 = Mechanik (Alle Abmessungen in mm, Lasten in N)
- 8 = US customary (Imperial) Einheiten (AASHTO: foot, lbs, kip)
- 9 = US customary (Imperial) Einheiten (ACI/AISC: inch, lbs, kip)

Man hat aber auch die Möglichkeit ein einzelnes Objekt aus der Liste der impliziten Einheiten in der CDBASE.CHM siehe 2.6 [Datenbank](#) auszuwählen (z.B. 1001 oder GEO_LENGTH, siehe cdbase.chm) und nur für dieses die Einheiten umzustellen, dann muss mit einer Angaben zu USE eine passende Einheit (z.B. N, mm oder cm²) und optional eine unterschiedliche Anzahl von Nachkommastellen mit DIG gewählt werden.

8.3.9 SEIT – Ein- und Ausgabeformat

SEIT

Wert	Bedeutung	Unit	Voreinst.
NRST	Nummer der ersten Seite in der Ausgabe Eine negative Eingabe schaltet die Seitennummerierung ab.	—	*
NZEI	Anzahl der Zeilen pro Seite	—	*
RAND	Anzahl der Spalten Heftrand links	—	*
SPRA	Ausgabesprache, sofern implementiert 0 deutsch 1 englisch 2 französisch 3 spanisch	—	*
SPRE	Eingabesprache, sofern implementiert 0 deutsch 1 englisch	—	*
UNIA	obsolet: Einheitensystem Ausgabe	—	*
UNIE	obsolet: Einheitensystem Eingabe	—	*
FORM	Format des Seitenkopfes 0 einzeliger Kopf 1 mehrzeiliger Kopf 2 ZTVK mit grafischen Zeichen 3 ZTVK ohne grafische Zeichen 4 Kurzkopf	—	*
PRIL	Printlevel in Ausgabedatei -2 nur Fehlermeldungen -1 zusätzlich Warnungen 0 zusätzlich Informationen 1 zusätzlich Rechenzeiten	—	0
BEZ	Bezeichnung für Seite (z.B. "Seite III/")	LIT12	*

Die Satzart SEIT ist im Großen und Ganzen nur noch historisch, denn sie beschreibt Parameter, die vom Betriebssystem abgeleitet werden oder in der SOFISTIK.DEF vordefiniert werden sollten. Außer den Sprachen und den Einheiten können alle Werte noch nachträglich in Report Browser gesetzt werden.

Standard

SOFISTIK AG * Bruckmannring 38 * 85764 Oberschleißheim SOFISTIK 2018 AQUA - ALLGEMEINE QUERSCHNITTE	Seite 1 04.12.2017
--	-----------------------

#KOPF1
#KOPF2
#KOPF3
#KOPF4
#KOPF5
#KOPF6

Materials and Cross-Section

Kein Rahmen

SOFISTIK AG * Bruckmannring 38 * 85764 Oberschleißheim
SOFISTIK 2018 AQUA - ALLGEMEINE QUERSCHNITTE

Seite 1
04.12.2017

Model and geometry

Materials and Cross-Section

Standardnorm ist EuroNorm EN 1993-1-1:2005 Steel Structures (Europe) V 2018

Structure: A (Buildings)

Schneelastzone : 1

Mit Rahmen

SOFISTIK AG * Bruckmannring 38 * 85764 Oberschleißheim SOFISTIK 2018 AQUA - ALLGEMEINE QUERSCHNITTE	Seite 1 04.12.2017
--	-----------------------

#KOPF1
#KOPF2
#KOPF3
#KOPF4
#KOPF5
#KOPF6

Materials and Cross-Section

Standardnorm ist EuroNorm EN 1993-1-1:2005 Steel Structures (Europe) V 2018

ZTF-K Rahmen

Bei der ZTVK Version werden von den Kopfzeilen die ersten 48 Zeichen verwertet. Der Firmentext kann nur von SOFiSTiK dauerhaft geändert werden. Jedoch kann man in den SOFiSTiK-Einstellungen einen anderen Bearbeiternamen für ein Projekt festlegen.

Verfasser : SOFiSTiK AG * Bruckmannring 38 * 85764 Oberschleißheim Programm : SOFiSTiK 2018 AQUA - ALLGEMEINE QUERSCHNITTE	
Bauwerk : #KOPF1 #KOPF2	ASB Nr: Datum: 04.12.2017

Materials and Cross-Section

Standardnorm ist EuroNorm EN 1993-1-1:2005 Steel Structures (Europe) V 2018

Structure: A (Buildings)

Schneelastzone : 1

Bauteil : #KOPF3 Block : #KOPF4	Seite 1	Archiv Nr.:
Vorgang : #KOPF5 #KOPF6		

ZTV-ING Rahmen

In **sofistik.def** werden die festen Anteile angegeben (in der Regel Baumassnahme, Strassenbauverwaltung Aufsteller). Jeder KOPF in der **.dat** Datei füllt dann die fehlenden Kopf-Zeilen auf.

- KOPF1 Baumassnahme
 KOPF2 Strassenbauverwaltung
 KOPF3 Aufsteller
 KOPF4 Bauteil
 KOPF5 Kapitel 1
 KOPF6 Kapitel 2

Baumaßnahme: #KOPF1	Bauwerksnummer (ASB):						
Straßenbauverwaltung: #KOPF2							
Aufsteller: #KOPF3	Datum: 04.12.2017						
Materials and Cross-Section Standardnorm ist EuroNorm EN 1993-1-1:2005 Steel Structures (Europe) V 2018 Structure: A (Buildings) Schneelastzone : 1							
Materialien							
<table border="1"> <thead> <tr> <th>Mat</th> <th>Materialbezeichnung</th> <th>γ-M</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>S 275 (EN 1993)</td> <td>1.00</td> </tr> </tbody> </table>	Mat	Materialbezeichnung	γ-M	1	S 275 (EN 1993)	1.00	
Mat	Materialbezeichnung	γ-M					
1	S 275 (EN 1993)	1.00					

Bauteil: #KOPF4	Seite: 1
Kapitel / Vorgang: #KOPF5 #KOPF6	Archiv Nr.:

8.3.10 SIZE – Grafisches Format

SIZE

Wert	Bedeutung	Unit	Voreinst.
DINA	Größe des Zeichenbogens Positiver Wert Querformat Negativer Wert Hochformat	—	-URS
M	Maßstab 0 Formatfüllend * Ingenieurmäßiger Maßstab, Zeichnung möglichst blattfüllend	—	*
W	Breite des Blattes/Bereichs	cm	*
H	Höhe des Blattes/Bereich	cm	*
RAND	Zeichnen eines Randes NEIN zeichnet inneren Rand + Maßketten	LIT	*

Tabelle wird auf der Folgeseite fortgesetzt.

Wert	Bedeutung	Unit	Voreinst.
	<p>W und H definieren in diesem Falle die Größe des bezeichnenbaren Bereichs (für Plotter mit Einzelblatt und Laserdrucker, dann Voreinst.)</p> <p>JA</p> <p>Um die Zeichnung wird ein Rand freigelassen. W und H sind die Abmessungen des Blatts (2 cm Heftrand, sonst 0.5 cm)</p> <p>VOLL</p> <p>Zusätzlich wird ein Schnittrand um das Zeichenblatt gezogen (für Trommelplotter oder Drucker) (Voreinst. für volle Formate)</p>		
FORM	<p>Form des Randes</p> <p>STAN Standard-Rahmen</p> <p>ZTVK ZTVK88-Rahmen</p> <p>URS Report Browser-Einstellung</p>	LIT	*

Dieser Satz erscheint bei den meisten Programmen mit grafischer Ausgabe. Hier wird das Ausgabeformat definiert. Generell gibt es zwei Varianten.

- Der Benutzer gibt das Papierformat in Kurzform über DIN A und/oder explizit mit Breite W und Höhe H an. Das Programm kann den Maßstab selber festlegen indem es die größtmögliche Darstellung unter Einhaltung der gängigen Werte 1:1, 1:2 und 1:5 oder deren Vielfachen wählt. Eine explizite Vorgabe des Maßstabes M wird beachtet, sofern technisch möglich. Eine Angabe M 0 erzeugt eine größtmögliche Darstellung mit krummem Maßstab.
- Der Benutzer gibt den Maßstab vor und erreicht durch die Angabe von W und/oder H zu Null, dass das Papierformat entsprechend groß gewählt wird. Wird nur ein Wert zu Null definiert, so wird das Bild in der zweiten Richtung eingepaßt, der Maßstab braucht in diesem Fall nicht definiert zu werden.

Statt der DIN Größen kann bei DIN A auch ein amerikanisches Format oder die Spezialformate der Hewlett-Packard Plotter angegeben werden. In diesem Fall wird ein Bild mit Rahmen, entsprechend der Eingabe zu FORM und RAND, angefordert, das genau so vom Programm Report Browser gedruckt wird. Mit URS bzw. FORM URS werden Bilder ohne Rahmen angefordert, der Rahmen wird dann vor dem Drucken vom Programm Report Browser hinzugefügt. DIN A URS entspricht einem A4-Papier, mit W und H kann die Bildgröße direkt vorgegeben werden. FORM URS errechnet aus der vorgegebenen Papiergröße die passende Bildgröße, z.B.

“SIZE 3 FORM URS”; W und H bestimmen dann die Papiergröße. Ein vorgestelltes Minus-Zeichen beschreibt die Hochformate:

URS,-URS	Die maximale Bildgröße für Report Browser
A,-A	ANSI Paper A (11 x 8.5 inch)
B,-B	ANSI Paper B (17 x 11 inch)
C,-C	ANSI Paper C (22 x 17 inch)
D,-D	ANSI Paper D (34 x 22 inch)
HPA,-HPA	Hewlett Packard Paper A (259 x 199 mm)
HPB,-HPB	Hewlett Packard Paper B (416 x 259 mm)
HPC,-HPC	Hewlett Packard Paper C (529 x 378 mm)
HPD,-HPD	Hewlett Packard Paper D (809 x 528 mm)
HP4,-HP4	Hewlett Packard Papier 4 (276 x 193 mm)
HP3,-HP3	Hewlett Packard Papier 3 (404 x 276 mm)
HP2,-HP2	Hewlett Packard Papier 2 (564 x 366 mm)
HP1,-HP1	Hewlett Packard Papier 1 (787 x 564 mm)
WIN,-WIN	Papierformat des aktuellen WINDOWS-Druckers
MP,-MP	Matrix-Drucker mit 8 x 12 inch (200 x 287 mm)
LP,-LP	Laserdrucker mit A4 Papier (198.4 x 280 mm)
MPW	breiter Matrix-Drucker 14 x 12 inch Papier Bei diesen Formaten ist RAND NEIN die Voreinstellung.

8.4 Erzeugen von Bildern

Mit den folgenden Sätzen können maßstäbliche Skizzen in die Vor- oder Nachbemerkenungen eingefügt werden. Dabei stehen die GKS-Befehle Polylinie, Polymarker, Füllgebiet und Text sowie das allgemeine Primitive und ein spezieller Bemaßungsbefehl zur Verfügung.

8.4.1 <PICT> – Start eines Bildes

<PICT>

Wert	Bedeutung	Unit	Voreinst.
XCM	Abmessung horizontal	cm	*
YCM	Abmessung vertikal	cm	*
TEXT	Bildüberschrift	Lit64	-

Mit dem Satz <PICT> beginnt jedes Bild. Es umfasst alle Elemente bis zu dem nächsten Satz vom Typ </PICT>. Es wird automatisch eine Transformation Nr. 1 initialisiert, die eine Zeichnung in cm erlaubt.

8.4.2 GNT – Skalierung des Bildes

GNT

Wert	Bedeutung	Unit	Voreinst.
NR	Nummer der Transformation	—	*
M	Maßstab	—	-
XMIN	Fenster der User-Koordinaten	*	-
YMIN		*	-
XMAX		*	-
YMAX		*	-
WXMI	Fenster im Papierbereich	cm	0
WYMI		cm	*
WXMA		cm	0
WYMA		cm	*

Damit wird eine Transformation ausgewählt (nur NR angeben) oder definiert. Es wird auf eine gleichmäßige Skalierung in beiden Richtungen errechnet. Wenn man unterschiedliche Skalierungen in X und Y-Richtung haben will, so muss M < 0 und alle vier Koordinaten des Fensters angegeben werden. GKS unterstützt 3 unterschiedliche Transformationen.

8.4.3 GPL – Polygonzug
GPL

Wert	Bedeutung	Unit	Voreinst.
X1	(Alternativ können Polygonzug in separaten Sätzen mit jeweils immer nur einem Punkt X1,Y1 in entsprechend vielen Sätzen definiert werden. Damit sind dann auch bis zu 255 Punkte möglich)	*	!
Y1		*	!
X2		*	X1
Y2		*	Y1
....	
X16		*	X15
Y16		*	Y15

8.4.4 GPM – Polymarker
GPM

Wert	Bedeutung	Unit	Voreinst.
X1	Koordinaten der Markersymbole	*	!
Y1		*	!
X2		*	X1
Y2		*	Y1
....	
X16		*	X15
Y16		*	Y15

8.4.5 GFA – Füllfläche
GFA

Wert	Bedeutung	Unit	Voreinst.
X1	Koordinaten der Füllfläche	*	!
Y1		*	!

Tabelle wird auf der Folgeseite fortgesetzt.

Wert	Bedeutung	Unit	Voreinst.
X2	(Alternativ können Füllflächen in separaten Sätzen mit jeweils immer nur einem Punkt X1,Y1 in entsprechend vielen Sätzen definiert werden. Damit sind dann auch bis zu 255 Punkte möglich)	*	X1
Y2		*	Y1
....	
X16		*	X15
Y16		*	Y15

8.4.6 GGDP – Allgemeines Grafikelement

GGDP

Wert	Bedeutung	Unit	Voreinst.
TYPE	Typ des Elements CIRC Vollkreis BUTT Kreisfläche ARC Kreisbogen VECT zentrierte Pfeilchen VEC1 Pfeilchen beginnend VEC2 Pfeilchen endend	<i>LIT</i>	CIRC
X1	Koordinaten oder	*	!
Y1	Koordinatenrichtungen des Elements	*	!
X2		*	!
Y2		*	!
....	
X15		*	-
Y15		*	-

Das GGDP ist die Methode um im GKS komplexere Element anzusteuern. Diese sind bei SOFiSTiK:

- CIRC** Ein Vollkreis mit Mittelpunkt (X1,Y1) und einem beliebigen Peripheriepunkt (X2,Y2). OPT wird nicht ausgewertet.
- BUTT** Eine Vollkreisfläche mit Mittelpunkt (X1,Y1) und einem beliebigen Peripheriepunkt (X2,Y2). OPT wird nicht ausgewertet.
- ARC** Ein Kreisbogen mit Mittelpunkt (X1,Y1) der vom Peripherie-punkt (X2,Y2) zum Punkt

(X3,Y3) geht.

VECT Ein einzelner Vektor im Punkt (X1,Y1) mit den Richtungskomponenten (X2,Y2) oder ein Bereich mit Vektoren, die entlang einer Basislinie von P1 zu P3 zu P5 etc. abgetragen werden in den Richtungen P2, P4 P6 etc. Bei VECT ist der Mittelpunkt der Vektoren auf der Basislinie, bei VEC1 der Startpunkt und bei VEC2 der Endpunkt mit der Pfeilspitze. Die Größe und der Abstand der Pfeilspitzen wird über die aktuelle Schriftgröße gewählt.

8.4.7 GTXT – Beschriftung

GTXT

Wert	Bedeutung	Unit	Voreinst.
X	Koordinaten des Einfügepunkts	*	!
Y		*	!
TEXT	Text	—	-
VAL	Zahlenwert	*	-
DIM	Dimension	—	1
ND	Anzahl der Nachkommastellen	—	*

Ein Text kann höchst unterschiedlich angeordnet werden. Der darzustellende Text an sich kann aus einer Textkomponente und/oder einem Zahlenwert zusammengesetzt werden. Der Zahlenwert kann in einer Dimension wie in der cdbase.chm siehe 2.6 [Datenbank](#) angegeben konvertiert und formatiert werden.

8.4.8 GSCA – Vermaßung

GSCA

Wert	Bedeutung	Unit	Voreinst.
X1	Koordinaten des ersten Punkts	*	!
Y1		*	!
X2	Koordinaten des zweiten Punkts	*	X1
Y2		*	Y1
TEXT	Text	—	-
VAL	Zahlenwert	*	-
DIM	Dimension	—	1
ND	Anzahl der Nachkommastellen	—	*

GSCA ist eine Sonderform der Beschriftung. Es wird eine Maßlinie gezeichnet und ein Text an diese geschrieben. Dieser Text ist in der Voreinstellung durch den Abstand der Maßpunkte gegeben, er kann jedoch explizit über eine Zahl und einen Text erweitert werden. Die Kombination von beliebigem Text und dem voreingestellten Abstand wird erreicht, wenn der Text mit einem Gleichheitszeichen "=" endet. Der Zahlenwert kann in einer Dimension wie in der cdbase.chm siehe 2.6 [Datenbank](#) angegeben konvertiert und formatiert werden.

8.5 Attribute grafischer Darstellungen

Die grundlegenden Zeichenelemente Linie, Markierung, Text und Fläche können Attribute haben. Der Benutzer gibt in der Regel alle Attribute in den Programmen mit grafischer Ausgabe durch Angabe von gebündelten Nummern an. Diese sind nach folgendem Muster aufgebaut:

$$ind = 1000 \cdot col + 100 \cdot ibr + ityp$$

Für Windows-Programme sind unterschiedliche Paletten für Bildschirme mit hellem oder dunklem Hintergrund sowie monochrome oder farbige Drucker vorgesehen. Sie können diese Definitionen in Report Browser bei den Optionen in der Registry für alle Programme verändern.

Die folgenden CADINP-Satznamen stehen nur innerhalb der Bilddefinitionen <PICT> ... </PICT> zur Verfügung. Jedoch gelten die Erläuterungen der gebündelten und individuellen Attribute auch für alle anderen Programme.

8.5.1 GCOL – Farbauswahl

GCOL

Wert	Bedeutung	Unit	Voreinst.
COL	Stiftnummer oder Farbe	—	1
R	Rot-Anteil	—	-
G	Grün-Anteil	—	-
B	Blau-Anteil	—	-

Mit GCOL wird eine Farbe für alle Elemente ausgewählt. Bei COL kann eine Nummer zwischen 1 und 15 oder eine der englischen Farbbezeichnungen der nachfolgenden Tabelle angegeben werden. Sofern das Ausgabegerät dieses unterstützt kann man auch die Farbe (empfohlen für die Nummern 9 bis 15) mit den RGB-Anteilen explizit vorgeben.

col = Colour Index (Farbe)	0 = Hintergrund
STAN	1 = schwarz bzw. weiß (Stift 1)
RED	2 = rot (Stift 2)
GREE	3 = grün (Stift 3)
BLUE	4 = blau (Stift 4)
YELL	5 = gelb (Stift 5)

MAGE	6 = magenta	(Stift 6)
CYAN	7 = cyan	(Stift 7)
BROW	8 = braun	(Stift 8)

8.5.2 GPLI – Polyline Attribute

GPLI

Wert	Bedeutung	Unit	Voreinst.
IND	Gebündelter SOFiSTiK-Index	–	1
COL	Farbe (wie bei GCOL)	–/LIT	-
TYPE	Linientyp	LIT	-
	SOLI durchgezogen		
	DASH strichliert		
	DOT punktiert		
	DDOT strichpunktiert		
	NDAS eng strichliert		
	NDOT eng punktiert		
	NDDO eng strichpunktiert		
	WDAS weit strichliert		
	WDOT weit punktiert		
	WDDO weit strichpunktiert		
WIDT	Linienbreite	–	-
SCAT	Art der Vermassung GSCA	–	2
	1 Striche schräg		
	2 kleine Kreise		
	3 Pfeilchen		

Für Linien gilt für den gebündelten Typ:

$$ind = 1000 \cdot col + 100 \cdot widt + type$$

widt = Linienbreitenfaktor	0 = normale Breite
	1 = 1.4fache Breite
	2 = 2.0fache Breite
	3 = 2.8fache Breite
	4 = 4.0fache Breite
	5 = 5.6fache Breite

etc.

type = Strichart

- 1 = durchgezogen
- 2 = strichliert mittel
- 3 = punktiert mittel
- 4 = strichpunktiert mittel
- 5 = strichliert eng
- 6 = punktiert eng
- 7 = strichpunktiert eng
- 8 = strichliert weit
- 9 = punktiert weit
- 10 = strichpunktiert weit

8.5.3 GPMI – Polymark Attribute

GPMI

Wert	Bedeutung	Unit	Voreinst.
IND	Gebündelter SOFiSTiK-Index	—	1
COL	Farbe (wie bei GCOL)	—/LIT	-
TYPE	Markertyp (Nummer oder Literal) · + * o X	LIT	-
SIZE	Größenfaktor	—	-

Für Markierungen gilt:

$$ind = 1000 \cdot col + 100 \cdot size + type$$

size = Markergröße

- 0 = normale Größe
- 1 = 1.4fache Größe
- 2 = 2.0fache Größe
- 3 = 2.8fache Größe
- 4 = 4.0fache Größe
- 5 = 5.6fache Größe
- etc.

type = Markerart

- 1 = . Punkt
- 2 = + Plus
- 3 = * Stern
- 4 = o Kringel
- 5 = x Kreuz

8.5.4 GTXI – Text Attribute

GTXI

Wert	Bedeutung	Unit	Voreinst.
IND	Gebündelter SOFiSTiK-Index	–	1
COL	Farbe (wie bei GCOL)	–/LIT	-
H	Texthöhe	cm	-
BX	Schreibrichtung	–	1
BY		–	0
HALI	Horizontale Ausrichtung	LIT	NORM
	NORM Voreinstellung (PATH)		
	LEFT linksbündig		
	CENT zentriert		
	RIGH rechtsbündig		
VALI	Vertikale Ausrichtung	LIT	NORM
	NORM Voreinstellung (PATH)		
	TOP oberster Zellenrand		
	CAP oberster Buchstabenteil		
	HALF Mittellinie		
	BASE Schreiblinie		
	BOTT Unterlängen		
PATH	Schreibrichtung	LIT	RIGH
	RIGH nach rechts		
	LEFT nach links		
	UP nach oben		
	DOWN nach unten		
EXPA	Expansionfaktor	–	1.0
SPAC	zusätzliche Zwischenräume	–	0.0
FONT	Fontnummer	–	-

Bei Text gilt für den gebündelten Index:

$$ind = 1000 \cdot col + font$$

font = Schriftart

installationsabhängig

8.5.5 GFAI – Fill Area Attribute

GFAI

Wert	Bedeutung	Unit	Voreinst.
IND	Gebündelter SOFiSTiK-Index	—	1
COL	Farbe (wie bei GCOL)	—/LIT	-
STYL	Fill area Style	LIT	-
	HOLL Hollow (nur Umrandung)		
	SOLI Solid (Farbfüllung)		
	PATT Pattern (Graustufen)		
	HATC Hatch (Schraffur)		
	BPAT Pattern mit Rand		
	BHAT Hatch mit Rand		
TYPE	Style index	—	-

Bei Flächen (Fill Area) gilt für den gebündelten index:

$$ind = 1000 \cdot col + 100 \cdot styl + type$$

- styl = Fill Area Style
- 0 = hollow (nur Berandung gezeichnet)
 - 1 = solid (vollflächig anlegen)
 - 2 = pattern (Muster füllen)
 - 3 = hatch (Schraffieren)

type = Fill Area Style Index (nur bei styl =2/3)

Die Auswirkungen des Wertes type sind geräteabhängig. Klassische Plotter können zum Beispiel meistens keine Muster darstellen. Muster 1 ist vollkommen leer, bei Muster 2 ist in der Regel eine gleichmäßige leichte Schummerung zu erwarten, die mit steigender Nummer dichter wird. Ab type=11 sind spezielle Musterungen vorgesehen. Wird auf den Index der Wert 32 addiert, so sind die Muster durchscheinend (nicht deckend).

Bei Schraffuren, die immer durchscheinend sind, sind verschiedene Varianten in ein oder zwei Richtungen vorgesehen.

- | | | |
|------|-----------|---|
| type | 1 / 2 / 3 | = vertikal / horizontal / vertikal+horizontal |
| | 4 / 5 / 6 | = diagonal +45 Grad / -45 Grad / +45 und -45 |
| | 7-12 | = wie 1 bis 6 strichliert |
| | 13-24 | = wie 1 bis 12 weiter Abstand |
| | 25,26 | = Stahlbeton |
| | 27 | = Erde |

8.6 Einfügen von Bildern

8.6.1 <LINK> – Einfügen eines Bildes

<LINK>

Wert	Bedeutung	Unit	Voreinst.
	Dateiname.bmp	LIT72	-

Mit **<Link>** können Bilder im BMP-Format in Eingabedateien und Templates eingefügt werden.

In einem Block für Texte mit **<Text> ... </Text>** bzw. für zu erstellende Bilder mit **<Pict> ... </Pict>** ist das Einfügen von Bilddateien nicht möglich.

8.7 Parametrisierte Eingaben

Die Eingabesprache CADINP erlaubt die Definition von Standardeingaben mit freien Parametern.

Zum Beispiel könnte ein einfacher Fachwerkträger des folgenden Typus definiert werden:

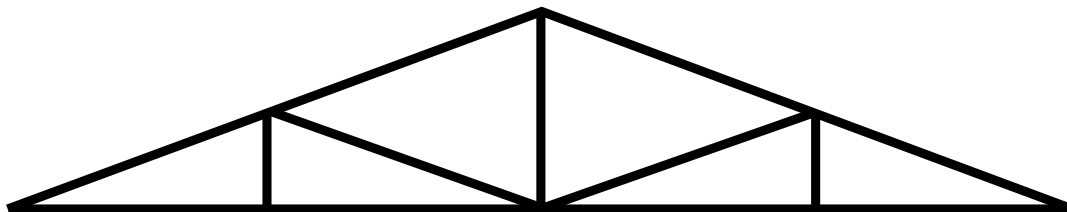


Abbildung 8.2: Fachwerkträger

Parameter seien hier die Spannweite und die Höhe. Eine Eingabe zu SOFIMSHA könnte dann wie folgt definiert werden:

+PROG SOFIMSHA

KOPF TRUSS

LET#1 12.40 ! STÜTZWEITE

LET#2 2.30 ! HÖHE

KNOT 1 0 0 PP

(2 4 1) (#1/4 #1/4) 0

5 #1 0 PP

KNOT 6 #1/4 -#2/2

```

7 #1/2 -#2
8 3*#1/4 -#2/2
FACH (1 4 1) (1 1) (2 1) QNR 1
FACH 5 1 6 QNR 1
6 6 7 ==
7 7 8 ==
8 8 5 ==
FACH (11 13 1) (2 1) (6 1) QNR 2
FACH (14 15 1) (6 2) 3 QNR 2
ENDE

```

Umfangreichere Möglichkeiten werden durch den Einsatz von Schleifen geboten. Soll zum Beispiel eine Knotenreihe auf einem Hyperbelbogen generiert werden, so erlaubt die folgende Eingabe dieses:

```

PROG SOFIMSHA
KOPF KÜHLTURM
! MERIDIAN HYPERBOLIC EQUATION (R/R0)**2-((Z-Z0)/ZZ)**2 = 1
! (R1/R0)**2-(-Z0/ZZ)**2 = 1 => (1/ZZ)**2
! (R2/R0)**2-((H-Z0)/ZZ)**2 = 1
! PARAMETER:
STO#H 160.0 ! TOTAL HEIGHT
STO#R0 30.0 ! SMALLEST RADIUS
STO#R1 55.0 ! RADIUS AT BOTTOM
STO#R2 35.0 ! RADIUS AT TOP
LET#1 SQR(#R1**2-#R0**2)
LET#Z0 #H*#1*(SQR(#R2**2-#R0**2)-#1)/(#R2**2-#R1**2)
LET#ZZ #Z0/SQR((#R1/#R0)**2-1.0)
LET#M 16 ! SUBDIVISION IN HEIGHT
TXA GENERATION OF A HYPERBOLIC COOLING TOWER
TXA HEIGHT RADIUS
TXA 0.0 #(R1,9.1)
TXA #(Z0,9.1) #(R0,9.1)
TXA #(H,8.1) #(R2,9.1)
TXA
TXA PARAMETER ZZ OF HYPERBOLIC EQUATION = #(ZZ,9.3)

LET#2 #H/#M ! DELTA-Z
LOOP#1 #M+1
LET#Z #1*#2
KNOT #1*50+1 #R0*SQR(1.+((#Z-#Z0)/#ZZ)**2) 0.0 #Z
ENDLOOP
ENDE

```

Für den Anfänger mögen diese Eingaben verwirrend sein, sie sind aber genau die Möglichkeit, die es dem Ingenieur erlaubt, häufig vorkommende Berechnungen völlig zu automatisieren. Viele Statik-Makros haben sich im Laufe der Zeit aus einfachen Anfängen zu komplexen Pro-

grammen entwickelt.

8.8 Kompatibilität von Datensätzen

Eine textbasierte Eingabe eignet sich recht gut für eine langfristige Speicherung von Eingabedaten, sicher auch besser als ein proprietäres Binärfomat. Leider ergeben sich jedoch auch hier immer wieder Umstellungen und Änderungen. Damit ein Organismus leben kann, müssen auch mal veraltete Eingaben sterben.

SOFiSTiK muss hier immer zwischen der Verständlichkeit des Handbuchs, dem Abfangen von möglichen Fehleingaben und der Kompatibilität der Datensätze abwägen.

Daher haben wir folgende Grundregeln aufgestellt:

- Eingabedatensätze werden bevorzugt hinten ergänzt, wenn sich aber ein Eintrag logisch weiter vorne besser darstellt (z.B. 3. Koordinate hinzugekommen), wird diese vorne eingefügt. Auch Bezeichnungen bleiben grundsätzlich immer hinten. Der Benutzer sollte also vorangestellte Kennworte bei allen Werten verwenden, die ergänzende Daten beschreiben, z.B:

```
KNOT 100 10.25 20.00 24.00 FIX PZ
```

Bei größeren Datenmengen ist immer eine Tabelle sinnvoll, der Gebrauch von Standardwerten '-' sollte grundsätzlich vermieden werden. Bei exportierten Daten kann eine Entfernung "überflüssiger" Daten der sinnvollste Weg sein um alte Datensätze weiter verwenden zu können.

- Es kann leider passieren, dass sich auch mal Satznamen oder Kennworte ändern. Dies ist in der Regel ein Hinweis auf eine andere Eingabesyntax oder Bedeutung der Werte. Der normale Weg ist der, dass in einer Version diese Eingaben als obsolet markiert sind und eine Warnung erzeugt wird, dass sich diese Eingabe in einer der nächsten Versionen ändern wird. Wir empfehlen Eingaben, die in späteren Versionen weiterverwendet werden sollen, rechtzeitig umzustellen. Obwohl wir uns bemühen, die alten Eingaben so lange als möglich noch zu verarbeiten, wird die Beschreibung im nächsten Major-Release definitiv entfernt.
- Eine andere Eingabeform auf einem bestehenden Kennwort ist ein besonders heikler Fall. Wir bemühen uns dann alte Datenwerte abzufangen oder zumindest mit einer Warnung zu erkennen (z.B. mm statt m), aber das ist leider nicht immer möglich. In vielen Fällen kann die explizite Eingabe einer Einheit die Verhältnisse klarstellen. Der Benutzer sollte daher nach einem Release-Wechsel seine Ergebnisse besonders sorgfältig prüfen.
- Häufiger dagegen ist die Änderung von Voreinstellungen. Dies kann durch Änderung der Normen, durch verbesserte Programmlogik oder Wünsche der Kunden verursacht sein. Hier gibt es keinen Königsweg, wenn man wenig Werte eingibt profitiert man von diesen Änderungen automatisch, aber es kann auch passieren, dass der Datensatz nach einem Update andere Ergebnisse liefert. Dass man z.B. bei einer Berechnung nach Eurocode automatisch die landestypischen Besonderheiten aus den Einstellungen des Betriebssystems setzen kann, erleichtert zwar die Eingabe ein bisschen, kann aber zu anderen Ergebnissen auf einem anderen Rechner führen und sollte deshalb vermieden werden.

9 DEF - Umgebungsvariablen

Einige Parameter der SOFiSTiK-Programme lassen sich über definierte Umgebungsvariablen (Environment) beeinflussen. Eine Zusammenstellung dieser Variablen findet sich am Ende dieses Abschnitts. Die beeinflussten Bereiche sind:

- Voreinstellungen für Layout (Papiergröße, Ausgabesprache etc.)
- Parameter zur Datenbank CDBASE (Buffer, Multitasking etc.)
- Standard-Eingabesätze

Environment Variablen können mit den klassischen Befehlen des Betriebssystems gesetzt werden, z.B.

Windows Desktop	Arbeitsplatz→Eigenschaften→Erweitert
Windows CMD	<i>SET parameter=value</i>
Linux Bash	<i>export parameter=value</i>

Empfohlen wird jedoch, diese Parameter in einer eigenen Konfigurationsdatei mit Namen *sofistik.def* zu bündeln, die z.B. auch projektweise vorgehalten werden kann. In dieser Datei können alle Variablen des SOFiSTiK-Environments sowie beliebige Definitionen für Eingabeblocke (Vgl. Kapitel 9: [DEF - Umgebungsvariablen](#)) abgelegt werden:

z.B.:

```
SOFISTIK_C=49
GRAFSIZE=SIZE -HP 0
KOPF1=Büroneubau der Xyz-Versicherung
STDMAT=BETO 1 B 35 ; STA 2 BST 500
```

Vorrangig ist eine Definition in der Eingabe-Datei. Dann folgt die Definition in der Umgebung des Betriebssystems, nachrangig eine Definition in SOFISTIK.DEF. Wenn für verschiedene Projekte unterschiedliche Einstellungen gewünscht werden, empfiehlt sich die Abspeicherung der Datei SOFISTIK.DEF im Projektverzeichnis. Dieses erfolgt am besten beim Editieren mit TEDDY unter SOFISTIK → Optionen:

Die folgende Tabelle umfasst alle derzeit benutzen Werte (Einträge in Englisch, damit in allen Versionen gleich!):

Variable	Parameters (separated by comma)	possible values
SOFISTIK_NAME	alternate user name	
SOFISTIK_C	nation (International Telefon code)	00 = Generic Europe 01 = United states 49 = Germany 43 = Austria 41 = Switzerland

Variable	Parameters (separated by comma)	possible values
	ch_set (character set of computer) Input language	44 = Great Britain 81 = Nippon 86 = China 91 = India 30 = Greece 31 = Netherlands 32 = Belgium 33 = France 34 = Spain 45 = Danmark 46 = Sweden 47 = Norway 39 = Italy 351 = Portugal 352 = Luxembourg 353 = Ireland 358 = Suomi/Finland -2 = utf8 0 = German 1 = English
SOFISTIK_A	Header type in printout language of output	0 = one line 1 = multiple lines 2 = ZTVK grafical 3 = ZTVK standard chars 0 = deutsch 1 = englisch 2 = französisch 3 = spanisch
	level of messages	-3 = nothing at all -2 = only error messages -1 = errors and warnings 0 = statistics

Variable	Parameters (separated by comma)	possible values
	Units of output	1 = computing times see cdbase.chm 2.6
SOFISTIK_P	Parameters for LST/ERG files	
	number of lines per page	68
	No of columns left margin	6
	No of lines bottom margin	0
	No of lines top margin	0
	Type of formfeed	0 = blank lines only 1 = form-feed character
	Type of linefeed	0 = default 1 = convert UNIX to WIN
	Total of printable columns	82
	Pagenumber	0 = default -1 = omit numbers
KOPF1 KOPF2 KOPF3	First Header Line Second Header Line Third Header Line	
SOFISTIK_PRODIR	alternate directory for the database and project files	Saves all project files without explicit pathname
SOFISTIK_TMPDIR	alternate directory for all temporary project files	Defaults to SOFISTIK_PRODIR
PSJOBPAR	Default Values for WPS/SPS	see Chapter 10.11.
PSJOBINFO	Time & File stamping for WPS/SPS	see Chapter 10.11.
CDBASEMEM	Size of Memory for CDBASE in Bytes or MBytes	default: 1MB
CDACCESS	CDBASE multitasking feature	SINGLE = deactivate NOWAIT = do not wait if locked (useful for remote analysis)

Variable	Parameters (separated by comma)	possible values
CDBASETEMP	Directoryname for temporary scratch files	Using TEMP or TMP if not defined
CDBASEVER	Version of CDBASE format	CDBASEVER=501 maximum 256 GB (Default) CDBASEVER=503 maximum 1024 GB
SOF_NUM_THREADS	Default number of threads to be used	NUMBER_OF_PROCESSORS or OMP_NUM_THREADS

Die in der Tabelle aufgeführten Variablen müssen in der SOFISTIK.DEF am Beginn vor dem ersten Klammerausdruck stehen, z.B.

```

SOFISTIK_C=...
SOFISTIK_A=...
SOFISTIK_P=...
[Layout-1]
....

```

10 Start der Berechnung

10.1 Allgemeines

Jedes Programm erwartet seine Eingabedaten auf einer Datei. Die Eingabe besteht aus Sätzen, die einen Namen haben und nach den CADINP-Regeln im freien Format aufgebaut sind.

Im Rahmen einer Projektbearbeitung ergibt sich in der Regel ein Ablauf von Einzelmodulen, die zum Zusammenwirken einer kompletten statischen Berechnung erforderlich sind. Dabei ist die normale Vorgehensweise die, alle Daten in einer einzigen oder zumindest wenigen Dateien zu sammeln und durch vorgestellte Anweisungen im Datensatz das zugehörige Programm anzugeben.

Ein Eingabedatensatz ist damit prinzipiell nach folgendem Schema aufgebaut:

```
PROG AQUA
HEAD
! Material and cross-section definition
END

PROG SOFIMSHA
HEAD
! Input data for FE-system
END

PROG ASE
HEAD
! Input data for analysis of 1st load case
END

PROG ASE
HEAD
! Input data for analysis of 2nd load case
END

! Copy the report file somewhere else ...
+SYS COPY "*.plb" d:

PROG WING
HEAD
! System and result plots
END
```

Die erste Zeile sollte eine PROG-Zeile sein. Jeder Modul kann beliebig oft angesprochen wer-

den. Die Reihenfolge der Programmstarts entspricht der Reihenfolge der PROG-Zeilen in der Datei. Steht statt *PROG* oder *+PROG* ein *-PROG*, so wird das entsprechende Modul übersprungen. Es können auch einzelne Module direkt ausgewählt werden.

Die Eingabe erfolgt mit dem System TEDDY oder einem Generierungsprogramm. Diese Dateien werden durch das Programm SPS einer Berechnung zugeführt. SPS kann innerhalb von TEDDY oder SOFIPLUS auch direkt gestartet werden.

Hinweis

Die vollständige Verarbeitung und Berechnung eines SOFIStiK Eingabe-Datensatzes erfolgt entweder interaktiv via *WPS* oder als Batch-Job via *SPS*.

10.2 #DEFINE - Parametersubstitution

SPS ist weiter in der Lage, globale Ersetzungen in der Eingabedatei vorzunehmen (parsen). Die Vereinbarung des Textblocks kann an beliebiger Stelle mit einer entsprechenden Zeile `#define name=text` erfolgen. Der Name des Textblocks besteht aus einem Text mit bis zu 10 Zeichen ohne \$. Das erste Zeichen eines Textblocknamen muss ein Buchstaben sein. Der Wert des Textblocks kann beliebig lang sein. Parameter innerhalb einer Zuweisung werden erst bei der Ersetzung aufgelöst. Die Umdefinition eines definierten Textblocks ist möglich.

Im weiteren Text der Eingabe, auch in Blöcken, kann ein Parameter ersetzt werden, indem er mit der Syntax `$(name)` angesprochen wird. Leerzeichen sind zwischen \$ und (nicht erlaubt. Groß- und Kleinschreibung hat keine Auswirkungen. Die Ersetzung erfolgt gegebenenfalls rekursiv, Definitionen wie `$(A$(INDEX))` sind erlaubt.

Beispiel:

```

$PROG                                (Kennung für TEDDY)
#define LAENGE=3.70
#define BREITE=30
#define HOEHE =50
#define PLATTE=50 20 10 80

PROG AQUA
BETO 1 B 25
STAH 1 BST 500
QB 1 $(HOEHE) $(BREITE)
QB 2 $(PLATTE) ASU 2.3
ENDE

PROG SOFIMSHA
SYST ROST
KNOT 1 0.0 0.0 FIX PP
KNOT 2 $(LAENGE)/2 0.0
KNOT 3 $(LAENGE)
STAB 1 1 2 1
STAB 2 2 3 1
ENDE

```

Im Gegensatz zu den CADINP-Variablen #() werden mit \$() Strings ersetzt, man kann also an einer Stelle auch eine Generierungsanweisung oder Literale ersetzen.

Voreingestellt sind zwei Parameter \$(NAME) mit dem Hauptnamen der Ausgabedatei und \$(PROJEKT) mit dem Namen des Projekts. Diese können vor allem bei System-Kommandos hilfreich sein.

Textblöcke, die nicht in der Eingabedatei definiert sind, können über einen SET-befehl gesetzt werden. Wenn also (z. B. in der AUTOEXEC.BAT) das Kommando eingegeben wird:

```
SET SIZE=LP 0 FORM ZTVK
```

so kann in allen Dateien über SPS das entsprechende Format definiert werden:

```
SIZE $(SIZE)
```

Außerdem ist es möglich Parameter für SPS global in einer Datei SOFiSTiK.DEF zu beschreiben. Damit ergibt sich eine dreistufige Hierarchie:

- Vorrangig ist eine Definition in der Eingabedatei
- Zweitrangig ist die Definition mit SET (vermeiden!)
- Nachrangig ist eine Definition in SOFiSTiK.DEF

10.3 #INCLUDE - Blockbildung

Mit der Blockbildung können mehrere Datenzeilen an beliebiger Stelle im Datensatz mehrfach gesetzt werden. Es sind max. 256 Blöcke und beliebig viele Dateiblöcke in beliebiger Reihenfolge zugelassen. Blöcke dürfen bis zu einer Tiefe von 32 verschachtelt werden.

#DEFINE name	Anfang des Blockes name (bis 8 Zeichen)
#ENDDF	Ende des aktuellen Blockes
#UNDEF name	Löschen eines definierten Blocks
#INCLUDE name	Setzen des Blockes name aus dem Speicher oder aus der Datei name

Damit lassen sich zum Beispiel nicht nur häufig vorkommende Eingabezeilen einmalig definieren und dann beliebig oft innerhalb der ganzen Eingabe wiederverwenden, sondern auch Unterprogramme realisieren.

Beispiel:

```
#define SECT
$PROG AQUA
$ Trapezquerschnitt PARAMETER B0,BU,H
QPOL UPZ
QP 1 #1/2 -#3/2
    2 #2/2 #3/2
UBEW 3
#endif
```

```

PROG AQUA
STAH 1 ST 37
QNR 1 ; LET#1 0.60,0.20,0.60
#include SECT
QNR 2 ; LET#1 0.60,0.30,0.60
#include SECT
QNR 3 ; LET#1 0.60,0.30,0.70
#include SECT
ENDE

```

Bei UNIX ist die Groß/Kleinschreibung beim Einfügen von Dateien genau zu beachten! Die alten Formate \$BLOCK BEG/END/SET werden auch noch unterstützt, sollten aber langfristig nicht mehr verwendet werden.

10.4 APPLY - Einbinden von Daten während der Berechnung

Mit dem Befehl APPLY kann eine Datei an beliebiger Stelle zwischen zwei Eingabeblocks im Datensatz eingebunden werden, jedoch nicht innerhalb eines Modules. Diese Datei sollte ein oder mehrere Modulaufrufe +PROG enthalten.

Während bei #INCLUDE (vgl. Kap. 10.3: [#INCLUDE - Blockbildung](#)) die Daten vor der Berechnung in den Datensatz eingefügt werden, wird der Befehl APPLY während der Berechnung ausgeführt, das heißt, die Daten können während einer Berechnung generiert und im Anschluss an der richtigen Stelle eingefügt werden.

Zusätzlich lässt sich APPLY mit einem Vorzeichen +/- steuern, das bedeutet, mit +APPLY wird der Befehl ausgeführt, mit -APPLY wird er übersprungen.

Ein typisches Beispiel für die Anwendung von APPLY ist das Programm CSM (Construction Stage Manager). Die mit dem CSM erstellte Datei \$(NAME)_csm.dat wird im Anschluss mit APPLY in die Berechnung eingefügt.

```

+PROG CSM
...
ENDE
+APPLY "$(NAME)_csm.dat"  $ enthält mindestens ein +PROG
+PROG ASE
...
ENDE

```

10.5 #IF - Bedingte Eingaben

Man kann IF THEN ELSE Konstruktionen definieren. Damit kann man sowohl größere Eingabeblocks ein- und ausschalten, was bei IF-Konstruktionen in CADINP schwieriger ist, als auch mehrere Programmaufrufe in einen Block zusammenfassen.

Die entsprechenden Steuerzeilen sind allen C-Programmierern gut vertraut. In Spalte 1 beginnend gibt es folgende Möglichkeiten:

```
#if ausdruck
```

```

    beliebige Zeilen, auch PROG und SYS
#else
    beliebige Zeilen, auch PROG und SYS
#endif

```

„ausdruck“ kann der Namen eines Blocks oder einer Variablen sein, der als wahr angesehen wird, wenn dieser definiert und nicht leer oder 0 ist, oder ein Vergleich \$(MODUS)==EC bzw. \$(MODUS) < >EC oder \$(ANZAHL)>3 sein. Die Vergleiche > oder < sind allerdings keine Zahlenvergleiche sondern lexikalische Vergleiche, wobei Zahlen rechtsbündig und Texte linksbündig verarbeitet werden, also ist $A < B$, $10 < 18$ aber leider auch $10.0 > 12$.

Falls die Bedingung falsch ist, wird der erste Block überlesen, der mit #else eingeleitete Block wird hingegen verarbeitet. Selbstverständlich kann der zweite Block auch entfallen wenn er nicht benötigt wird.

Diese Konstruktionen dürfen bis zur Tiefe 32 verschachtelt werden. Unmotivierte #else oder #endif führen ebenso zu einem Fehler wie nicht abgeschlossene Konstruktionen.

Beispiel:

```

#define DOAQB=0

#if DOAQB
    PROG AQB
    KOPF .....
    LF .....
    BEME .....
    ENDE
#endif

    PROG STAR2
    KOPF .....
    #if DOAQB
        $ Bemessung mit AQB
    #else
        BEME .....
    #endif
    .....
    ENDE

```

Das Einrücken dient nur der Schönheit der Darstellung. Im obigen Beispiel wird also STAR2 zur Bemessung gerechnet. Wenn DOAQB=1 eingesetzt wird, so wird statt dessen AQB mit der Bemessung beauftragt.

10.6 Templates

Eine besondere Form der Eingabedateien sind die sogenannten Templates. Diese sind in der folgenden Form strukturiert:

```

#DEFINE L1=10.0
#DEFINE L2=20.0
#DEFINE L3=30.0
#DEFINE P=12.0

PROG TEMPLATE (oder ein beliebiges anderes Programm)
LET#L1 $(L1)
LET#L2 $(L2)
LET#L3 $(L3)
LET#L #L1+#L2+#L3
TXA Vorbemerkung
<TEXT>
Dies ist die Berechnung eines 3-feld-Trägers mit den
Stützweiten:
    L1 = <EDIT:L1>$(L1)</EDIT>
    L2 = <EDIT:L2>$(L2)</EDIT>
    L3 = <EDIT:L3>$(L3)</EDIT>
    SUM = #(L,10.2)
</TEXT>
<PICT>
    ....
</PICT>

PROG AQUA
    normaler Datensatz

```

Häufig wird man nur die Vorbemerkungen mit der Erläuterung der wichtigsten Eingangsparameter in einem eigenen Programm TEMPLATE zusammenfassen. Die Eingabe zu diesem Programm besteht dann nur aus Arithmetik, Texten sowie entsprechenden Skizzen. Der Mechanismus ist aber für jedes Programm möglich.

Man kann nun in Report Browser in der Ausgabe die wichtigen Parameter verändern und den Datensatz für dieses eine Modul neu berechnen lassen. Damit kann man komplexe Makros für andere Benutzer erstellen, die problemspezifisch nur noch wenige Eingaben vorgeben müssen..

10.7 Iteration über mehrere Module

Die Theorie 2. Ordnung ist unter Berücksichtigung nichtlinearer Werkstoffe in STAR2/AQB enthalten. In STAR2 kann jedoch nur ein Bemessungsmodus berücksichtigt werden. Eine Bemessung mit verschiedenen Bemessungsmodi kann durch einen iterativen Ablauf zwischen AQB und STAR2 realisiert werden. Die Iteration wird durch Angabe des Steuerwortes ITER in den PROG-Sätzen gesteuert.

```

PROG STAR2      1. Schritt für Theorie 1.0rdnung
STEU I          Lastfälle definieren
LF .....
ENDE
PROG AQB ITER parm      parm mit Leerzeichen abtrennen
STAB .... ; LF ....    Auswahl und Bemessung definieren

```



```

BEME .... ; DEHN ....
ENDE
PROG STAR2 ITER parm
KOPF ....
STEU II 1
ENDE
    
```

Die maximale Anzahl der Iterationen kann an ITER angehängt werden (z.B. ITER 30), Voreinstellung ist 20 Iterationen.

Der Mechanismus wurde ab Release 23 auf andere Kombinationen von Modulen erweitert. Der Abbruch der Iteration muss dann jedoch im Zuge einer CADINP-Eingabe explizit durch einen Befehl EXIT_ITERATION erfolgen.

```

PROG TEMPLATE
STO#TARGET 0 ; STO#PARAM 1.0
PROG AQUA ITER
RCL#PARAM
QNR ....          Definition Querschnitt mit #PARAM
PROG AQB ITER
BEME ....          Bemessen und Speichern Ergebnisse
ENDE
@KEY / LET# TARGET ...      Ermittle eine Zielfunktion
IF ABS(#TARGET) < 0.001
  EXIT_ITERATION
ELSE
  STO#PARAM new_target_value
ENDIF
    
```

10.8 Betriebssystem Kommandos

Es können beliebige shell Kommandos in den Ablauf der Berechnung integriert werden. Parametersubstitution ist auch für diese Zeilen aktiv, z.B. werden die Zeichenfolge \$(NAME) durch den Hauptnamen der Eingabe bzw. Ausgabedatei, \$(PROJEKT) durch den Namen des Projekts ersetzt.

*SYS kommando	kommando	wird als kommando in Batchfile geschrieben, wenn die letzte PROG Zeile aktiv war.
+SYS kommando	kommando	wird als kommando in Batchfile geschrieben.
-SYS kommando	kommando	wird überlesen, deaktiviert alle folgenden *SYS Zeilen.

z.B:

+SYS del "\$(PROJEKT).\$D1" löscht die Steifigkeitsmatrix

+SYS wait "name.exe"

WPS wird veranlasst, mit der Abarbeitung der weiteren Module solange zu warten, bis der mit "+SYS -wait name.exe" gestartete Prozess wieder beendet ist.

10.9 Job-Geschichte

Bei der Berechnung mit vielen Einzelprogrammen, kann es wichtig werden, die zeitliche Abfolge zu rekonstruieren. SPS schreibt deshalb in die erzeugten Eingabedateien eine Hilfsinformation, die mit einer Variablen des Environments gesetzt werden kann:

```
SET PSJOBINFO=n
```

n = 0 keine Informationen

n = 1 Filename und Datum(nur im Datenecho)

n = 2 wie 1, einschl. Jobnummer(nur im Datenecho)

n = 3 wie 2, Jobnummer+Uhrzeit auch auf jeder Ausgabeseite

Die Voreinstellung ist 2. Die Jobnummer besteht aus einer Bezeichnung des Rechners mit 8 Buchstaben und einer fortlaufenden Nummer, die in einer Datei PSJOBNR gespeichert sind. Bei Mehrbenutzerbetrieb ist darauf zu achten, dass PSJOBNR eventuell im aktuellen Verzeichnis anzulegen ist, wenn Zugriffskonflikte auftreten sollten.

Beispiele zu PSJOBINFO:

```
PROG AQUA
$ Datenfile: D:\STATIK\P00\DAT0.DAT (.#01) 14:48:48 25/05/95
KOPF .....
```

```
PROG AQUA
$ Datenfile: D:\STATIK\P00\DAT0.DAT (.#01)          25/05/95
$ Jobnummer: PC-Nr:17/8700063                       14:48:48
KOPF .....
```

Die Datei PSJOBNR wird ab PSJOBINFO=2 benötigt. Die ersten 8 Spalten enthalten beliebigen Text. Sie werden nicht gedruckt, wenn sie leer sind. Wenn SPS keine Datei PSJOBNR findet, wird eine im lokalen Directory angelegt.

10.10 Start eines Einzelprogramms

Gelegentlich kann es vorkommen, dass man ein Programm direkt starten möchte. Dies ist generell möglich, wenn die Eingabedatei alle Parameter-Ersetzungen schon hinter sich hat. Möchte man aus einer normalen Eingabedatei die geparste Eingabedatei erstellen, so erfolgt dies aus WPS mit

```
Datei > Geparste Datei speichern ...
```

wenn man die geparste Gesamtdatei speichern möchte, oder alternativ mit

Datei > Geparstes Modul name speichern ...

wenn man nur ein einziges geparstes Modul benötigt. Die Modulauswahl erfolgt hierbei im WPS-Modul-Baum

Der Start der Einzelprogramme erfolgt dann mit

programm dateiname [-parm] [projekt]

"dateiname" ist der Name der Eingabedatei; er gibt auch den Grund-Namen der Ausgabe-dateien an. Wenn der Name Leerzeichen enthält, muss er mit " eingeschlossen werden. *projekt* ist der Name der Datenbank. Als *parameter* sind im Wesentlichen die gleichen Parameter wie bei SPS vorgesehen. Es kommen jedoch noch weitere hinzu, die z.B. steuern, welche Art von Metafile erzeugt werden soll und ob ein bestehendes Metafile ergänzt oder gelöscht werden soll.

Wenn man ein Rechen-Programm ganz ohne Parameter aufruft, so meldet es sich mit seinem Namen, seiner Versionsnummer und der Liste der möglichen Parameter. Der Benutzer wird nach dem Namen der Eingabedatei gefragt. Eine leere Eingabe (Return-Taste) beendet das Programm. Diese Art des Aufrufs ist sinnvoll, um die Integrität des Programms zu testen. Es werden die Lizenzierung und die Version der ERR-Datei abgefragt.

10.11 Zusätzliche Aufrufmöglichkeiten

Für besondere Zwecke stehen dem erfahrenen Anwender zusätzliche Aufrufmöglichkeiten aus der Eingabeaufforderung (Command-Shell) zur Verfügung:

Parameter	Beschreibung
+ bzw. ++	Öffnet die letzte bzw. vorletzte Datei. Wird ohne vorgestellten Schalter - oder / verwendet.
-0 -1 -2 -3 -4	Ermöglicht den Start von bis zu 5 Instanzen (Voreinstellung: -0).
-nosingle	Startet die Applikation als eigenständige Instanz (no single application).
-test	Setzt einen generellen Test-Flag. Dieser ist nur für die Entwickler von Interesse. Je nach Entwicklungsstand werden einzelne Meldungsfenster aktiviert.
-b	Startet WPS im Batch-Modus. Die Berechnung wird sofort gestartet und am Ende wird das Fenster wieder automatisch geschlossen.
-e	Erzeugt bei Auftreten eines Fehlers am Ende der Berechnung ein Meldungsfenster. Ist im Zusammenhang mit -b oder -run sinnvoll. Falls der Parameter nicht gesetzt ist und die Berechnung mit einem Fehler endet, wird kein Meldungsfenster angezeigt. WPS endet jedoch nicht, so dass der Fehler im Fehlerprotokoll nachgelesen werden kann.
-close	Schließt nach einer Berechnung das Fenster unabhängig davon ob ein Fehler aufgetreten ist oder nicht. Ist im Zusammenhang mit -b oder -run und ohne -e sinnvoll.

Parameter	Beschreibung
-noclose	Schließt nach einer Berechnung das Fenster nicht. Ist im Zusammenhang mit -b oder -run sinnvoll.
-z	Hängt den Ergebnisausdruck (name.plb bzw. name.erg) an einen evtl. vorhandenen Ergebnisausdruck an.
-run	Startet unmittelbar die Berechnung (z.B. beim Aufruf über Teddy mit "Schnellstart").
-run:xx	Startet unmittelbar die Berechnung (z.B. beim Aufruf über Teddy mit "Schnellstart"). Mit xx kann die Nummer der PROG-Zeile angegeben werden, die als einziges aktives Modul gerechnet werden soll.
-urs:xx	Setzt das Modul "PROG modul urs:xx" aktiv und alle anderen inaktiv.
-cdb:name	Name der cdb (falls dieser vom DAT-Namen abweicht).

11 Ausgabe

Bei der Berechnung mit **WPS / SPS** werden normalerweise drei Ausgabedateien erzeugt:

- die Report Browser - Ausgaben und graphische Darstellungen (**.plb**)
- die Protokoll-Datei (**.prt**)
- die Ergebnis-Datei (**.lst** und **.erg**)

Hinweis

Die von uns empfohlenen und qualitätsgesicherten Druckerschriften sind Consolas 9pt und Courier New 8pt.

11.1 Ergebnis- Ausgabe mit dem Report Browser

11.1.1 Allgemeines

Mit dem Report Browser lassen sich die Berechnungsergebnisse aller SOFiSTiK-Programm bequem und einfach zusammenstellen und ausdrucken.

Der Report Browser ist ähnlich aufgebaut wie der Windows-Explorer und bietet dem Anwender vielfältige Möglichkeiten zur individuellen Gestaltung von Ergebnisausdrucken, z.B. bei der selektiven Auswahl des Ausgabeumfangs (Text/Grafik) oder bei der Layoutgestaltung (Firmenlogo, Fußzeile, etc.).

Generellen Einstellungen, wie die Schriftart oder die maximale Dateigröße die komplett geladen werden kann, werden im **Menü SOFiSTiK >> Anwenderoptionen** eingestellt. Projekt bezogene Einstellungen können global eingestellt werden mit **SOFiSTiK >> Globale Optionen**, wenn sie für mehrere Projekte gelten, oder mit **SOFiSTiK >> Projekt Optionen**, wenn sie nur für ein Projekt gelten (weitere Erläuterungen finden Sie im Kapitel 2.11 (siehe [SOFiSTiK Optionen](#)).

11.1.2 Aktivierung von Report Browser



Nach einer SOFiSTiK- Berechnung kann über das Report Browser-Icon der Ergebnis-Ausdruck (mit der Erweiterung **.plb**).

Die vom Anwender spezifizierten Einstellungen werden in einer zugehörigen **.urs** Datei gespeichert und stehen nach einer Neuberechnung in der Regel wieder zur Verfügung.

11.1.3 Features

Selektive Festlegung des Ausgabeumfangs: Der Ausgabeumfang der Ergebnisausgabe kann vom Anwender auf seine Anforderungen angepasst werden.

Über das Glühbirnen-Symbol im Gliederungsbaum werden die Modulergebnisse aktiviert und deaktiviert. Durch Auf- und Zuklappen der Buch-Icons, können Berechnungsabschnitte ein- und ausgeschaltet werden. Die Einstellungen bezüglich des Ausgabeumfangs werden in der zugehörigen **.urs** Datei gespeichert und stehen nach einer Neuberechnung in der Regel wieder zur Verfügung.


Ganze Abschnitte können temporär im Übersichtsbaum verschoben werden und werden nun beim Ausdrucken berücksichtigt. Die geänderte Reihenfolge steht nach einer Neuberechnung jedoch nicht mehr zur Verfügung. Die gewünschte Modulreihenfolge sollte somit bereits in der Dateneingabe festgelegt werden.

Einfügen eines Inhaltsverzeichnisses

Mit **Einfügen** > **"Inhaltsverzeichnis einfügen"** wird vollautomatisch ein Inhaltsverzeichnis des Gesamtausdrucks erzeugt. Mit **Einfügen** > **"Inhaltsverzeichnis löschen"** wird ein vorhandenes Inhaltsverzeichnis gelöscht. Bei den Modul-Überschriften im Inhaltsverzeichnis wird der Text aus dem Report Browser - Modulbaum eingefügt, bzw. der Modulname falls kein Text definiert ist. Deaktivierte Abschnitte werden hierbei nicht mit aufgenommen. Weiterhin wird die Seitenzahl im Ausdruck ergänzt. Die Einstellung **Inhaltsverzeichnis einfügen** wird in der **.urs** Datei gespeichert.

11.1.4 Tabellenausgabe

Ab Version SOFiSTiK 2018 werden mit dem Report Browser die Ergebnisse zunehmend in Tabellenform ausgegeben. Die Lesbarkeit des Ergebnisausdruckes wird somit deutlich verbessert.

Legende: Bei Bedarf wird eine Legende zur Erläuterung der Tabellenausgabe ergänzt. Diese kann wahlweise mit einer verkleinerten Schriftgröße und ebenfalls wahlweise in farbiger Schrift ausgegeben werden. Der erfahrene Anwender kann die Legende auch ausblenden, was jedoch nicht generell empfohlen wird  Menü SOFiSTiK >> Anwenderoptionen... >> Report Browser.






Fußnoten: Eine weitere Verbesserung der Lesbarkeit der Ergebnisausgabe wird durch erläuternde Fußnoten erreicht. Im Gegensatz zu der Legende kann eine Fußnote nicht ausgeblendet werden, da sie den Ergebnisausdruck vervollständigt.

11.1.5 Individuelle Einstellungen des Report Browsers

Im Menü SOFiSTiK Anwenderoptionen Reportbrowser kann der gewünschte Ausgabeumfang des Berechnungsergebnisses voreingestellt werden. Grundsätzlich wird empfohlen, Fehler und Warnungen immer anzuzeigen. Diese Voreinstellung gilt für alle Projekte.

Hinweis

Wenn das Echo der Eingabe nicht angezeigt wird, werden Kapitel, die Fehler und Warnungen enthalten, zwangsweise aktiviert. Falls dies nicht gewünscht wird, die Echo-Ausgabe also trotz Warnung bzw. Fehlermeldung geschlossen bleiben soll, kann dies durch Setzen von Kapitel Einstellungen haben Vorrang im Optionendialog unterdrückt werden.

Die bei den Anwenderoptionen vordefinierten Einstellungen können innerhalb des Report Browsers über die Icons , , , ,  nochmals umgestellt werden, sodass Warnungen und Fehler usw. für die aktuelle Datei "temporär" angepasst werden können.

11.1.6 Drucken- Dialog

Preview- Fenster: Im Drucken- Dialog ist ein Vorschau-Fenster (Preview) implementiert. Verschiedene Angaben des ausgewählten Druckers, wie Typ und Standort, werden angezeigt.

Ausgabe in Datei: Eine Druck- Ausgabe in eine Datei ist ebenfalls möglich. Bei der Auswahl "Datei erzeugen" wird eine PDF- Datei im aktuellen Verzeichnis mit dem zugehörigen Dateinamen "name.pdf" erstellt. Es ist hierbei keine zusätzliche Software von Adobe erforderlich.

Besondere Druckoptionen: Es können mehrere Seiten verkleinert auf ein Blatt gedruckt werden. Temporär kann für das Drucken die Gestaltung der Seitennummer bzw. des Datums umgestellt werden.

11.1.7 Erweiterte Funktionen

Zoom-Handling: Neben den üblichen Zoom-Funktionen steht ein Schieberegler zur Verfügung, mit dem ein schnelles, stufenloses Zoomen im aktuellen Fenster ermöglicht wird. Mit **Strg** + **Mausrad** wird systematisch die Ansicht vergrößern bzw. verkleinert und mit **Strg** + **Mausziehen** kann gezielt ein Zoom-Fenster aufgezogen werden.

Selektionsmöglichkeiten: Mit Report Browser werden folgende Selektionsmöglichkeiten unterstützt:

- Zeilenblock
- Spaltenblock
- Tabellenspalten
- Einzelne Wörter

Der selektierte Bereich kann mit **Bearbeiten** > **kopieren** in der Zwischenablage gespeichert werden. Ein Speichern im EXCEL-Format ist ebenfalls möglich **Bearbeiten** > **Kopieren nach EXCEL** und in EXCEL **Bearbeiten** > **Inhalt Einfügen**). Das Aufziehen eines Zeilenblocks mit der Maus erfolgt **außerhalb des Randes**. Das Aufziehen eines Spaltenblocks mit der Maus erfolgt innerhalb des Randes. Alternativ kann zunächst ein Bereich in der Überschriftzeile selektiert werden und im Anschluss mit **Rechtsklick** > **Selektieren** > **Spalte selektieren** eine vollständige Spalte ausgewählt werden.

11.1.8 Funktionen im Navigationsbaum

Selektion im Navigationsbaum: Ähnlich wie im Explorer lassen sich mit der **Strg**- bzw. **↑**-Taste in Verbindung mit der Maus mehrere Berechnungsabschnitte auswählen und im Rechtsklickmenü gleichzeitig bearbeiten.

Unterstützung der TEDDY-Kapitel: Wenn eine Ergebnis-Datei durch Kapiteleinträge strukturiert ist (→ Eingabe der Kapitel in TEDDY mit `!#!Kapitel`), werden diese Kapitel von Report Browser unterstützt. Die einzelnen Kapitel können aktiviert bzw. deaktiviert werden (bei deaktivierten Kapiteln ist das Symbol um 90° gedreht) und komplett zusammengesoben werden (im Rechtsklickmenü).

Unterstützung bei der Fehlersuche: Beim Auftreten von Warnungen werden die betroffenen Module und die entsprechenden Berechnungsabschnitte im Navigationsbaum mit einem grünen Kreuz markiert.

Erweiterte Bearbeitungsmöglichkeiten des .plb- Ergebnisses: Benutzerdefinierte Änderungen der Ergebnisdatei (Seitenumbrüche, Leerzeilen, ...) werden in der zugehörigen **.urs**-Datei gespeichert. Sie stehen somit beim wiederholten Öffnen der **.plb** weiterhin zur Verfügung. Nach einer Neuberechnung werden sie nur in Ausnahmefällen übernommen, da in der Regel eine eindeutige Zuordnung nicht mehr möglich ist.

11.1.9 Zusätzliche Aufrufmöglichkeiten

Für besondere Zwecke stehen dem erfahrenen Anwender zusätzliche Aufrufmöglichkeiten aus der Eingabeaufforderung (Command-Shell) zur Verfügung:

• Aufruf mit Dateiname

Aufruf: **ursula** [/optionen] [Dateiname]

Dateiname: Anstelle eines Dateinamens kann auch **+** or **++** übergeben werden. Damit wird die letzte bzw. die vorletzte Datei geladen. Mit **abc*.plb** wird der Datei-Öffnen-Dialog mit der übergebenen Suchmaske gestartet. Die Eingabe von **."** wird als Abkürzung für die Suchmaske **"*.*** interpretiert.

• Spezielle Aufrufparameter

Optionen	Erläuterung
-t	Erzwingt eine Text-Ansicht.
-g	Erzwingt eine Grafik-Ansicht.
-r	Erzwingt ein automatisches Refresh (ohne Nachfrage), falls die PLB sich geändert hat.
-stli:AutoCad-Version	Unterstützt Stahlliste.
-txt:name.txt	Erzeugt eine Textdatei.
-gkx	Nur sinnvoll im Zusammenhang mit <i>-txt:name.txt</i> , erzeugt eine Textdatei einschließlich Grafik-Dump.
-urs:name.urs	Explizite Vorgabe einer URS-Datei.
-ssd:_xxx	<i>_xxx</i> = Liste der Pseudoextensions für eine Gesamtdatei. Beispiel: <i>ursula name.plb -ssd:_002;_003;_006</i> . Macht aus <i>name_002.plb</i> , <i>name_003.plb</i> und <i>name_006.plb</i> eine Gesamt-PLB.
-plbs:name1.plb;...	Fügt mehrere PLBs zu einer Gesamtdatei zusammen. Beispiel: <i>ursula name.plb -plbs:name1.plb;name2.plb</i> Macht aus <i>name.plb</i> , <i>name1.plb</i> und <i>name2.plb</i> eine Gesamt-PLB.
-hidden:urs1;urs2;...	Zeigt die mit der Urs-Id spezifizierten Module nicht an. Beispiel: <i>ursula name.plb -hidden:3.2;3.3</i>
-m	Eine Möglichkeit beim Arbeiten mit großen Dateien. Es werden zunächst lediglich die Daten zum Aufbau des Modulbaumes geladen.
-close	Ursula wird nach einem Aufruf unmittelbar wieder geschlossen. z.B.: <i>ursula -stli:191 -close /mdb=mdb name stl name</i>

• Aufruf für direktes Drucken

Optionen	Erläuterung
-print	Datei wird direkt gedruckt.
-printto:"Druckername"	Datei wird direkt auf dem angegebenen Drucker gedruckt. Die Anweisung "PDF" als Druckername erzeugt eine PDF-Ausgabe in " <i>name.pdf</i> ".
-page:all	Es werden alle Seiten gedruckt. Alternativ kann eine einzelne Seite oder ein Seitenbereich angegeben werden. Beispiel: <i>ursula name.plb -print -page:3-5</i>
-picture:all	Es werden alle Bilder gedruckt. Alternativ kann ein einzelnes Bild oder ein Bereich von Bildern angegeben werden. Beispiel: <i>ursula name.plb -print -picture:3-5</i>
-size:Ax	Stellt am Drucker die Papiergröße ein (möglich ist: A4, A3, A2, A1 oder A0).

• Generelle Aufrufparameter

Optionen	Erläuterung
-s:Sofistik-Pfad	Pfadangabe zu den Sofistik-EXE-Dateien (Umbiegen bzw. Setzen der Umgebungsvariablen SOFiSTiK=...)
-1 bis -4	Mit diesen Flags können mehrere Instanzen von Ursula gestartet werden.
-nosingle	Startet Ursula als eigenständige Instanz.
-test:nr	Setzt einen generellen Test-Flag. Nur für die Entwickler von Interesse. Je nach Entwicklungsstand werden einzelne Meldungsfenster aktiviert.

• Spezielle Tastaturbefehle

Einige spezielle Befehle lassen sich über die Tastatur aktivieren:

Befehl	Kurzbeschreibung
Strg + E	Markierten Text im Excel-Format in die Zwischenablage kopieren
Strg + G	Gehe zur Seite
Strg + L	Weiter suchen
Strg + D	Erzeugt einen PLB-Dump (nur für Entwickler)
Strg + U	Zeigt die URS-Datei (nur für Entwickler)
Strg + N	Leerzeile einfügen
Strg + Y	Zeile löschen

• Belegung der Strg-Tasten (Windows Standard)

Folgende Standard-Windows-Befehle werden unterstützt:

Befehl	Kurzbeschreibung
Strg + O	Datei öffnen
Strg + S	Datei speichern
Strg + P	Drucken
Strg + A	Gesamten Text markieren
Strg + C	Markierten Text/Grafik in die Zwischenablage kopieren
Strg + V	Grafik aus der Zwischenablage in den Modulbaum einfügen
Strg + F	Suchen

11.2 Protokoll-Datei (.prt)

Die **.prt** Datei wird als Protokoll-Datei bezeichnet und enthält wichtige Meldungen zur allgemeinen Betriebssituation. Sie wird normalerweise nur bei unvorhergesehenen Berechnungsabläufen benötigt. Sie enthält z.B. neben den benötigten Rechenzeiten noch Informationen über Fehler, Konvergenzbedingungen, Dateien etc.

Unter Windows erfolgt die Ausgabe der Dateien über die Windows-Druckroutinen. Diese werden durch Aufruf "Drucken" oder durch Drag and Drop auf ein Drucker-Symbol angestoßen.

11.3 Grafische Ausgabe Result Viewer

11.3.1 Allgemeine Hinweise

Der Result Viewer ist ein Programm für das Postprocessing von Berechnungen mit Finiten Elementen und Stabwerken. Es erlaubt die tabellarische Darstellung von fast allen in der Datenbasis gespeicherten Informationen, wie z.B. Informationen zu den Strukturen, Berechnungs- und Bemessungsergebnisse. Ausserdem können alle Querschnittsinformationen sowie Querschnittsergebnisse dargestellt werden. [RESULTVIEWER](#)

12 Problembehandlung

Nachfolgend werden nur Probleme im Zusammenhang mit der Anwendung der Software behandelt. Weitere Hinweise zur Behandlung von Installationsproblemen und Problemen mit dem Programmschutz finden Sie im Administrator Handbuch.

12.1 Allgemeines

Sollten Sie als Benutzer auf eine Ihnen vollkommen unerklärliche Situation stoßen, so rufen Sie sich folgende Punkte ins Gedächtnis.

- Die Wahrscheinlichkeit, dass der Fehler durch eine falsche Eingabe erzeugt wird, ist ziemlich groß. Auch die Beachtung von Warnungen im Zuge der vorigen Berechnungen oder Installation kann wertvolle Hinweise liefern.
- Alle Programme, auch die von SOFiSTiK, haben solange Fehler, wie sie benutzt werden. Diese treten bevorzugt immer dann auf, wenn Sie unter Zeitdruck ein völlig neues Gebiet in der Programmanwendung betreten. Deshalb sollten Sie möglichst erst an kleinen Beispielen das prinzipielle Verhalten untersuchen. Eine der häufigsten Ursachen für Fehler solcher Art ist eine Fehlinterpretation des Handbuchs oder der implementierten Theorie.
- Wenn etwas eben noch gelaufen ist, dann denken Sie über alles nach, was sich seitdem geändert hat (neuer Rechner, neues Betriebssystem, andere Eingaben etc.)
- Auch wenn es relativ unwahrscheinlich ist, kann eine Fehlermeldung auch einmal nicht die wahre Ursache des Fehlers beschreiben. Dies wird vor allem dann passieren, wenn ein Fehler nicht korrekt abgefangen wurde. Die Programme brechen nämlich nicht beim ersten Fehler ab, sondern versuchen so lange wie möglich weitere Fehler zu prüfen.
- Bevor Sie stundenlang suchen, wenden Sie sich an den Support oder Ihren Betreuer. Auch wenn Ihnen eventuell eine Rechnung gestellt werden sollte, haben Sie wirtschaftlich gehandelt, denn die Fehler werden in der Regel schneller erkannt.

12.2 Ordentliche Warnungen bzw. Fehlermeldungen

Der Ablauf der Berechnung sowie die für die Berechnung verwendete Programmversion wird in einer Protokoll - Datei vom Filetyp (*.PRT) festgehalten. Diese Datei sollte immer bei außergewöhnlichen Situationen (Programmfehler oder Bedienungsfehler) kontrolliert werden. Bei einer Support Anfrage ist diese Datei wesentlicher Bestandteil der Anhänge. Der Ablauf der Berechnungen wird automatisch gestoppt, wenn ein Fehler aufgetreten ist. Als reguläre Fehler bzw. Warnungen kommen in Betracht:

- Warnungen, das Programm gibt die Meldung aus: ++++ WARNUNG NR. nnnn IN PROGRAMM xxxx sowie eine oder mehrere Folgezeilen mit erläuterndem Text. Die Berechnung wird fortgesetzt.
- Benutzer- oder Datenfehler, das Programm gibt die Meldung aus: ++++ FEHLER NR. nnnn IN PROGRAMM xxxx sowie eine oder mehrere Folgezeilen mit erläuterndem Text. Die Berechnung wird je nach Schwere des Fehlers fortgesetzt.

Jedes Programm schreibt zum Schluss die Anzahl der Warnungen und Fehler in das PRT-File und in die Ausgabedatei. Sofern das Programm abgebrochen wurde, steht auch dieses in den Dateien.

Warnungen können modulweise mit der Eingabe `STEU WARN Nummer der Warnung` ausgeschaltet werden.

12.3 Fehlerstrategien

Nachfolgend wollen wir Ihnen ein paar Fehlersuchstrategien an die Hand geben, mit denen Sie die Fehler einkreisen und in vielen Fällen sogar selbst finden und lösen können.

Reduktion Datensatz:

Die wichtigste Strategie besteht darin den Eingabedatensatz, das SSD Projekt und auch die SOFiPLUS-Zeichnung zu reduzieren. Löschen Sie daher bitte alle Eingaben, Elemente, Programmblöcke, etc. die den Fehler nicht betreffen. Damit haben Sie einen kleinen übersichtlichen Datensatz, der schnell zu prüfen und zu verbessern ist. Mit dieser Strategie können Sie fast alle Eingabeprobleme eingrenzen und beheben.

Diese Strategie wenden Sie bitte auch an, wenn Sie sich zur weiteren Unterstützung an unseren Support wenden.

12.3.1 Probleme mit der Datenbasis (*.cdb)

Neben Eingabefehlern kann es auch zu Problemen mit der Datenbasis CDBASE kommen. Diese können ihren Grund darin haben, dass

- die Datei durch Abbruch eines Programms nicht mehr richtig organisiert ist (Record in Unordnung oder vergessene Record-Locks). Die Locks können Sie mit `DBINFO projekt,Z` oder mit einem entsprechenden Befehl in WPS entfernen. In hartnäckigeren Fällen bringt das Löschen bzw. Restaurieren der Datenbasis und neue Berechnungen Abhilfe.
- Eine Sicherung der Datenbasis immer mal zwischendurch ist bei größeren Projekten absolut sinnvoll.

12.3.2 Eingabefehler im TEDDY Datensatz

Bei Eingabefehlern wird Ihnen dies über eine Fehlermeldung mitgeteilt. Wenn Sie das Ergebnisprotokoll in der Report Browser öffnen, dann wird Ihnen die Fehlermeldung im ECHO Ausdruck der Ausgabe direkt nach der fehlerhaften Eingabezeile dargestellt. Auch wird Ihnen mitgeteilt, welche Eingabe das Programm von Ihnen erwartet. Damit haben sie einen direkten Hinweis auf die Fehlerursache und können diese einfach ändern und damit beheben.

12.3.3 Fehler bei der Systemgenerierung in SOFiPLUS

Hier ist die wichtigste Strategie die Reduktion der Eingabe. Gehen Sie einfach so vor, dass Sie die Hälfte Ihres Systems markieren und löschen. Danach exportieren Sie das System. Tritt nun der Fehler immer noch auf, wiederholen Sie das Ganze. Tritt der Fehler nicht auf, dann

gehen Sie eine Schritt zurück und löschen die andere Hälfte des Systems. Dies wiederholen Sie so oft, bis Sie ein kleines und überschaubares System haben. Dabei haben Sie die möglichen Eingabefehler bereits selbst gefunden und behoben, oder ein kleines System erzeugt, an dem Support und Entwicklung das Problem einfach und schnell nachvollziehen können. Damit können mögliche Programmfehler sehr schnell erkannt und behoben werden.

Wenn die Ursache gefunden und behoben ist, dann können Sie dies auch direkt an Ihrem endgültigen System durchführen.

12.3.4 Fehler bei der Berechnung

Im Wesentlichen tauchen immer zwei typische Probleme bei der Berechnung auf. Entweder ist das System instabil oder bei einer nichtlinearen Berechnung wird keine Konvergenz gefunden.

Systeminstabilität:

Fehler bei der Berechnung sind größtenteils Systeminstabilitäten. Bei einer Berechnung mit dem PROG ASE werden über einen numerischen Trick Eigenformen berechnet, die Sie sich im ANIMATOR ansehen können. Mit diesen Verformungsbildern werden Sie direkt auf die möglichen Ursachen hingewiesen.

Gerade bei 3d Systemen mit vielen Gelenken und Kopplungen sind solche Probleme besonders häufig. Hier sollten Sie so vorgehen, dass Sie am Anfang Ihres Systems keine Gelenke und Kopplungen definieren. Ist das System im Eigengewichtlastfall stabil, so können Sie Schritt für Schritt weitere Gelenke und Kopplungen definieren. Bitte nach jedem Schritt das System exportieren und den Eigengewichtlastfall berechnen und im ANIMATOR prüfen. Damit gehen Sie schrittweise vom Groben ins Feine und haben eine schnelle und effiziente Kontrolle über Ihre Arbeit. Auftretende Probleme sind so einfach erkennbar und sofort lösbar.

Sollte das Problem weiterhin bestehen, dann bitte wieder die Eingabe reduzieren.

Nichtlineare Berechnung:

Bei nichtlinearen Berechnungen wird eine iterative Berechnung durchgeführt. Im Wesentlichen gibt es zwei Gründe, warum die Berechnung nicht konvergiert. Entweder ist das System nicht tragfähig. Dann könnten Sie durch eine Veränderung der Steifigkeiten oder der Querschnitte das System "ertüchtigen" und eine Konvergenz erreichen. Es kann aber auch daran liegen, dass das gewählte iterative Verfahren für das betrachtete System numerische Problem hat. In diesem Fall wird Ihnen im Ergebnisprotokoll eine Reihe von Änderungsmöglichkeiten vorgeschlagen. Mit diesen Hilfen lässt sich in aller Regel das Problem lösen. Wenn nicht, wenden Sie sich bitte an den Support.

12.3.5 Probleme bei den Bemessungsergebnissen

Die meisten Fragen tauchen bei der Interpretation der Bemessungsergebnisse auf. Oftmals hilft es im Kapitel "Theoretische Grundlagen" des jeweiligen Handbuches nachzulesen. Dort sind die Grundlagen der Berechnungsalgorithmen kurz erläutert und ermöglichen eine Handrechnung zur Überprüfung der Ergebnisse.

Sofern Sie nicht weiterkommen wenden Sie sich gerne per E-Mail an unseren technischen Support unter support@sofistik.de. Damit wir Ihnen schnellstmöglich weiterhelfen können

benötigen wir nachfolgende Informationen und Daten von Ihnen:

1. Ihre Eingabedaten (*.sofistik und *.dwg , oder *.dat), die auf das Wesentliche reduziert sind. Bitte alle überflüssigen und nicht notwendigen Tasks / Module löschen.
2. Die Datei *.prt der letzten Berechnung, welche den Wortlaut der Warnung/Fehlermeldung enthält.
3. Eine Beschreibung welche Tasks / Module in welcher Reihenfolge zu starten sind.
4. Eine Diagose .xml Datei mit der Information über die von Ihnen verwendeten Programmversionen. Diese können Sie über das SSD / TEDDY Menü **Hilfe** > **Diagnose...** erstellen und als *.xml Datei speichern.
5. Beschränken Sie sich bei der Erzeugung der Bemessungsergebnisse auf ein Element und einen Lastfall. Fordern Sie eine umfangreiche Ausgabe des Ausdrucks an (ECHO VOLL EXTR). Schicken Sie uns Ihre Report Browser Ausgabe, indem Sie die wenigen Seiten als PDF drucken. Im Report Browser finden Sie im Menü **Datei** > **PDF erstellen** den dazu notwendigen Befehl.
6. Markieren Sie in der PDF, die von Ihnen angefragten Ergebnisse und fügen im PDF einen Kommentar hinzu, indem Sie Ihre Erwartungshaltung und Handrechnung dokumentieren.

Zippen Sie alle notwendigen Daten und fügen diese gepackte Datei der E-Mail mit Ihrer Problembeschreibung an.

12.4 Support

Sofern die oben beschriebenen Strategien nicht zum Ziel führen, wenden Sie sich bitte an unseren Support. Die Supportleistungen sind in den Supportbedingungen geregelt. Diese finden Sie im Internet unter

http://www.sofistik.de/fileadmin/FILES/support/Support_Service_Erlaeuterungen_06_2012.pdf

12.4.1 Erreichbarkeit SOFiSTiK Support

Unser Support ist über das SOFiSTiK Online Portal, per Email, per Fax bzw. telefonisch erreichbar. Um möglichst effizient zu arbeiten und um Ihnen schnellstmöglich eine Antwort auf Ihre Anfrage zu geben, sind unsere Supporter nicht direkt telefonisch erreichbar.

Unser oberstes Ziel ist es, Ihnen einen zielgerichteten und effizienten Support zu bieten. Um dies zu gewährleisten, ist es notwendig, dass wir uns Ihren anspruchsvollen Problemstellungen in Ruhe widmen können. Wir wollen auch vermeiden, dass Sie Ihre wertvolle Zeit am Telefon wartend verbringen, bzw. mehrfach verbunden werden.

Bei Ihren teilweise sehr komplexen Anfragen ist es in den meisten Fällen nicht möglich, direkt eine Antwort zu geben. Wir beschäftigen uns mit Ihren Fragen ohne Störungen von außen und melden uns anschließend per E-Mail oder telefonisch. Entweder können wir Ihnen gleich eine Lösung nennen, oder aber wir haben uns mit dem Problem soweit beschäftigt, dass wir telefonisch direkt in die Diskussion einsteigen und gemeinsam eine Lösung oder Umgehung finden können.

12.4.2 Mithilfe des Kunden

Für eine schnelle Bearbeitung Ihrer Anfragen sind wir auf Ihre Unterstützung angewiesen. Bitte beachten Sie bei Ihren Anfragen nachfolgende Punkte:

- Bitte geben Sie immer Ihre Kundennummer an. Diese finden Sie z.B. auf dem letzten Software Service Invoice im html-Dokument.
- Bitte geben Sie immer die verwendete Versionsnummer aller Programme an, z.B. SOFiCAD-Konstruktion 2016 mit AutoCAD 2016, SOFiPLUS 2016 mit AutoCAD 2016. Bei den Statikprogrammen wird bei jeder Berechnung ein Protokoll erzeugt. Diese Datei mit Endung *.prt enthält alle verwendeten Programmversionen, alle Warnungen und Fehlermeldungen.
- Informationen zum Betriebssystem (z.B. Windows 7 (32bit / 64bit), Windows 8, Linux) benötigen wir ebenfalls.
- Aufbereitung der Probleme, so dass wir diese direkt nachvollziehen können. Bitte denken Sie daran, dass wir nur dann Probleme lösen und beheben können, wenn wir diese auch nachvollziehen können.
- Minimierung der Zeichnung, bzw. des Projektdatensatzes auf die für das Problem wesentlichen Programmteile. Löschen aller nicht notwendigen Daten.
- Möglichst genaue Beschreibung wann, wo und unter welchen Bedingungen das Problem auftritt. Beispiel Statik: "Im Stab 2037 an der Stelle $x = 0.00$ kann ich im Bemessungslastfall 2031 die ermittelte Bewehrung (SOFiSTiK 2016-1 AQB) in Höhe von 12.35 cm^2 nicht nachvollziehen, Gemäß meiner Handrechnung erhalte ich einen Wert von 8.5 cm^2 . Woher kommt der Unterschied? Anbei meine Handrechnung."
- Informationen über die bereits selbst durchgeführten Untersuchungen, eventuell Scans der Handrechnungen
- Übersendung aller notwendigen Daten, die wir für die Nachvollziehbarkeit benötigen. Dies sind im Regelfall die Dateien: *.sofistik, *.dwg, *.dat, *.prt, Diagnose.xml sowie *.gra und *.plb, sofern notwendig. Bitte schicken Sie uns KEINE Ergebnisdateien *.cdb, *.erg, die meistens sehr groß sind. In besonderen Fällen fordern wir diese explizit an. Bitte fassen Sie alle Anlagen zu einer gezippten Datei zusammen.

Wir bitten um Ihr Verständnis, dass Support-Anfragen ohne die vollständigen Unterlagen im Regelfall solange nicht bearbeitet werden, bis wir alle erforderlichen Unterlagen zur Verfügung haben.

12.4.3 Supportanfrage via SOFiSTiK Online Portal

Als Wartungskunde haben Sie Zugang zu unserem SOFiSTiK Online Portal. Je Firma gibt es mindestens einen Administrator und beliebig viele User. Die Userverwaltung liegt damit in den Händen der Kunden.

Durch die Nutzung des Portals haben Sie nachfolgende Vorteile:

- Sie können Ihre Supportanfragen direkt in unser System einstellen.
- Anfragen können auch außerhalb der Geschäftszeiten eingestellt werden.
- Support-Anfragen über das Online Portal werden bevorzugt bearbeitet.

- Sie können jederzeit in einer FAQ Datenbank nach Lösungen suchen.
- Sie können in Ihren Supportvorgängen recherchieren.
- Sie sehen den Status Ihrer laufenden Anfragen.
- Sie können Ihre Firmendaten (Adresse, Ansprechpartner und SOFiSTiK Online User) selbst administrieren. In der Regel gibt es einen Administrator pro Firma)

Mit Hilfe dieses Portals sind Sie von unserer zentralen Supportannahme unabhängig, Ihre Anfrage ist sofort in unserem System verfügbar und Sie bekommen in der Regel auch schneller eine Antwort. Daher empfehlen wir Ihnen dieses Portal zu nutzen.

Das Online Portal finden Sie auf www.sofistik.de/support unter Links SOFiSTiK Online (Portal). Eine Hilfe zur Anwendung dieses Portals ist dort ebenfalls vorhanden.

Bitte legen Sie für jede Anfrage einen neuen Vorgang an. Pflegen Sie nur eindeutige Rückfragen in einen vorhandenen Supportvorgang ein. Diese Trennung der Anfragen ist für die Übersichtlichkeit bei der Bearbeitung und für spätere Recherchen unbedingt notwendig.

12.4.4 Supportanfrage aus SSD / TEDDY

Im SSD und im TEDDY gibt es die Möglichkeit über das Menü "Hilfe > SOFiSTiK Support Wizard ..." ein Problembeschreibung inklusive aller notwendigen Informationen und Daten zu erstellen.

Diese gezippte Datei und den erzeugten Text der Problembeschreibung können Sie am schnellsten über das SOFiSTiK Online Portal in unser System einstellen.

12.4.5 Erstellung der Diagnose.xml Datei

Die Diagnose.xml ist eine sehr wichtige Datei für uns Supporter. Informationen über alle für die Installation relevanten Daten und die Lizenzdaten sind darin enthalten.

Sie können diese Datei einfach aus TEDDY und SSD über das Menü "Hilfe > Diagnostik" erstellen. Im geöffneten Dialog bitte die Datei als Diagnose abspeichern und an den Support schicken.

In Sonderfällen starten Sie das Programm direkt aus dem Installationsverzeichnis:
z.B.: C:\Program Files (x86)\SOFiSTiK\2016\ANALYSIS_33_X64\diagnose.exe


13 Weiterführende Hilfen

Zum besseren Verständnis der Software haben wir neben den Handbüchern weiterführende Hilfen für Sie entwickelt.

13.1 Administration Handbuch

Für alle Fragen zur Installation, Lizenzierung und für den Software Updates finden Sie Antworten in unserem Online Administrations Handbuch (link missing). Diese Administrations Handbuch steht nur in englischer Sprache zur Verfügung (siehe <http://www.sofistik.com>).



13.2 VERiFiCATiON Manual

Zum besseren Verständnis der technischen Hintergründe der Software stehen Ihnen über das SSD / TEDDY Menü  zwei Verification Manuals ([VERIFICATION_MANUAL_MECHANICAL](#) und [VERIFICATION_MANUAL_DESIGN](#)) zur Verfügung. Darin werden die Möglichkeiten der SOFiSTiK gezeigt, um die Möglichkeiten der SOFiSTiK Software für nichttriviale und auch unkomplizierte Aufgabenstellungen anhand von Referenzlösung zeigen, um den Anwender Sicherheit bei den SOFiSTiK-Lösungen zu bieten. Ein weiteres Ziel ist es, die breite Palette von Berechnungsoptionen für die meisten Elemente und wichtigen Lösungsmöglichkeiten darzulegen und somit eine Anleitung für ähnliche Problemstellungen zu geben.

Diese Handbuch enthält eine Zusammenstellung von ausgewählten Benchmarks, die sich jeweils auf ein bestimmtes mechanisches Thema oder eine Bemessungsnorm beziehen. Die mit der SOFiSTiK-Software berechneten Ergebnisse werden den Referenzlösungen gegenübergestellt.

Das VERiFiCATiON Manual steht nur in englischer Sprache zur Verfügung.

13.3 Tutorials

Ergänzend zu den Handbüchern können Sie weitere Erläuterungen in unseren Online Tutorials (link einbauen) finden. Diese können direkt aus dem SSD / TEDDY Menü   aufgerufen werden.

Die Online Tutorials stehen nur in englischer Sprache zur Verfügung.

13.4 YouTube - Schulungs- und Präsentations Videos


Sie finden eine umfangreiche Sammlung von Videos auf unserem YouTube Kanal: www.youtube.com/user/SOFiSTiKAG.

13.5 Infoportal

Über <http://www.sofistik.de/infoportal> gelangen Sie zu unserem Web basierten Infoportal. Darüber können Sie nach allen Arten von Informationen suchen. Die Suche erfolgt über

die Kategorien (Dokumententyp, Produktgruppe, Themenbereich, Anwendung), oder über eine Volltextsuche. Unter anderem sind über das Infoportal einige Lehrfilme verfügbar. Diese zeigen alle wesentlichen Programm Features z.B. für die grafische Eingabe mit SOFiPLUS.

13.6 CADINP Beispielsammlung

Über den TEDDY gelangen Sie direkt zu einer umfassenden Beispielsammlung sortiert nach Programmen in der CADINP Eingabesprache. Sie finden diese Beispiel über den TEDDY Menü .

13.7 Forum

Das Forum dient als Diskussionsplattform für SOFiSTiK Anwender, Entwickler und Supporter. Ein Anspruch auf Beantwortung von Support Anfragen im Forum besteht nicht. Wenden Sie sich dazu via Email an support@sofistik.de.

Sie erreichen das Forum über <http://www.sofistik.de/forum>.