Anhang A
Compiler-Befehle und -Anweisungen

Compiler-Befehle und -Anweisungen teilen dem Compiler mit, wie er einen Maschinencode zu erzeugen hat. Meist handelt es sich hierbei um die Frage, ob Prüfroutinen in ein Programm eingebaut werden sollen (und somit aus Sicherheitsgründen auf maximale Geschwindigkeit zu verzichten ist) oder nicht.

Compiler-Befehle werden wie Kommentare geschrieben: Auf (* folgen ein Dollarzeichen ($) und ein Großbuchstabe, der einen der Compiler-Befehle bezeichnet. Das letzte Zeichen, ein Plus (+), Minus (-) oder Gleich (=), zeigt an, ob die Eigenschaft eines Compiler-Befehls aktiviert (+) oder ausgeschaltet (-) werden soll; das Gleichheitszeichen (=) bringt den Compilerbefehl in den vorletzten Zustand.

```
(*$B-*)  (* B ausschalten                          *)
(*$R+*)  (* R aktivieren                          *)
(*$Q=*)  (* Q in ursprünglichen Zustand bringen *)
```

Es ist möglich, eine ganze Liste von Compiler-Befehlen anzugeben:

```
(*$S-,R+,I-,V-,O+*)
```

Bitte beachten Sie, daß vor und nach dem Dollarzeichen kein Leerzeichen geschrieben werden darf, da sonst der Compiler die gesetzten Befehle nicht mehr richtig erkennen würden.

Compiler-Anweisungen werden ebenfalls wie Kommentare geschrieben, jedoch nicht mit einem der Zeichen +, - oder = ein- oder ausgeschaltet. An ihre Stelle treten Zahlenwerte oder Namen, die gewisse Größe festlegen. Zu den Compiler-Anweisungen gehören die folgenden Buchstaben: C, D, M und S (für S existiert auch ein Compiler-Befehl).

Bitte beachten Sie, daß gewisse Compiler-Befehle auch mit Hilfe des Untermenüs *Options, Compiler* verändert werden können.

A - Alias-Bezeichnung

Voreinstellung *(*$A+*)*; mit der Zeichenfolge ::= kann einer Variablen ein neuer Name zugeordnet werden (es wird eine "Alias-Variable" erzeugt; siehe *CONST*); falls der Compiler-Befehl A bei einer Variablen-Deklaration desaktiviert ist, kann eine solche Variable nicht zugleich direkt und indirekt angesprochen werden.

B - Ctrl-Break erlauben

Voreinstellung *(*$B+*)*; falls aktiviert, kann ein Programm mit *Ctrl-Break* abgebrochen werden. Dieser Compiler-Schalter muß im Haupt-Programm (-Modul) stehen. Mit den beiden Prozeduren *EnableBreakCheck* und *DisableBreakCheck* kann die Ctrl-Break-Prüfung an beliebigen Stellen ein- und ausgeschaltet werden.

C - CPU-Register sichern

Voreinstellung *(*$C F0*)*; diese Compiler-Anweisung legt fest, welche Register bei einem Prozedur-Aufruf gesichert werden müssen, um sie beim Prozedur-Ende wieder restaurieren zu können. Die einzelnen CPU-Register werden wie folgt codiert (hexadezimale Werte):

```
AX=01   CX=02   DX=04   BX=08
DS=10   ES=20   SI=40   DI=80
```

Standardmäßig werden die Register DS, ES, SI und DI gesichert (daraus ergibt sich der Wert F0 (Hex-Wert; =10 +20+40+80)). Das BP-Register wird immer auf den Stack gebracht.

D - Datensegment-Name

Diese Compiler-Anweisung definiert den Namen des Segmentes neu, in das die globalen Variablen gelegt werden sollen (standardmäßig verwendet der Compiler den Namen des Haupt-Moduls).
Bei einer Änderung des Datensegment-Namens muß die Compiler-Anweisung **D** im Haupt-Modul oder im Definitions- und Implementations-Modul vor dem reservierten Wort *MODULE* stehen; außerdem muß der neue Name mit der Buchstabenfolge **D_** eingeleitet werden.

E - Variante Records

Voreinstellung *(*$E-*)*; falls aktiviert, werden die varianten Teile eines Records solange wie möglich in den CPU-Registern behalten, andernfalls sofort wieder in den Arbeits-Speicher kopiert (siehe auch Compiler-Befehl W).

F - Far-Aufruf

Voreinstellung *(*F*)*; eine mit diesem Compiler-Befehl markierte Prozedur wird als *far* definiert (weiter Sprung).

G - Modulangabe in externen Namen

Voreinstellung *(*$G+*)*; falls desaktiviert, werden den externen Namen keine Modul-Namen vorangestellt (kann zu Namenskonflikten führen).

H - Konstante Aggregate

Voreinstellung *(*$H-*)*; falls aktiviert, können konstante Aggregate (siehe *CONST*) wie initialisierte Variablen verwendet werden.

I - Index-Prüfung

Voreinstellung *(*$I-*)*; im aktivierten Zustand werden alle Indizes von Arrays geprüft und das Programm durch einen Laufzeit-Fehler abgebrochen, wenn ein Index einen unzulässigen Wert aufweist.

J - Interrupt-Prozeduren

Voreinstellung *(*$J-*)*; wenn für eine Routine *(*$J+*)* gilt, wird sie mit dem Befehl IRET (Assembler-Befehl) verlassen; somit läßt sie sich als Interrupt-Routine verwenden (viele Anwendungen lassen sich jedoch mit *IOTRANSFER* verwirklichen, so daß dieser Compiler-Befehl kaum zur Anwendung kommt).

K - Aufruf-Konvention von C

Voreinstellung *(*$K-*)*; die Programmiersprache C geht mit den Parametern von Prozeduren und Funktionen anders um als Modula-2 (oder Pascal). Mit dem Compiler-Befehl K kann jedoch diese Umgangsform angenommen werden (nötig, wenn C- und Modula-Programme zusammengelinkt werden).

M - Codesegment-Name

Diese Compiler-Anweisung definiert den Namen des Segmentes neu, in das der Programm-Code geschrieben wird (standardmäßig verwendet der Compiler den Namen des aktuellen Moduls; jedes Modul erhält sein eigenes Code-Segment, das eine Größe von 64 KBytes nicht überschreiten darf).
Bei einer Änderung des Codesegment-Namens muß die Compiler-Anweisung M vor dem reservierten Wort *MODULE* stehen; außerdem muß der neue Name mit der Zeichenfolge C_ eingeleitet werden.

N - Near-Aufruf

Voreinstellung *(*F*)*, siehe Compiler-Befehl F; eine mit diesem Compiler-Befehl markierte Prozedur wird als *near* definiert, d.h. sie kann nur innerhalb eines einzigen Segmentes (64 KBytes) aufgerufen werden.

O - Überlauf-Prüfung (ganze Zahlen)

Voreinstellung *(*$O-*)*; wenn aktiviert, werden arithmetische Überläufe bei ganzen Zahlen durch einen Laufzeit-Fehler signalisiert (für reelle Zahlen siehe *FloatExc*).

P - Generieren externer Namen

Voreinstellung *(*$P-*)*; falls dieser Compiler-Befehl aktiviert wird, werden für lokale Prozeduren (sie sind beispielsweise innerhalb von Prozeduren definiert) externe Namen erzeugt. Diese Namenserweiterung kann einerseits das Arbeiten mit einem Debugger erleichtern, andererseits aber auch zu Namenskonflikten führen (mehrere identische Prozedur-Namen).

Q - Tracing von Prozeduren

Voreinstellung *(*$Q-*)*; alle Routinen, für die der Compiler-Befehl *(*$Q+*)* gilt, werden speziell behandelt, sobald das Modul *ProcTrace* importiert wird (siehe *Install*).

R - Teilbereichs-Prüfung

Voreinstellung *(*$R-*)*; bei aktivierter Schalterstellung wird ein Programm durch einem Laufzeit-Fehler unterbrochen, wenn versucht wird, einer Variablen einen ganzzahligen Wert zuzuordnen, der nicht im definierten Bereich liegt.

S - Stackgröße

Voreinstellung *($S 4000*)*; ermöglicht die Festlegung der Stackgröße (in Bytes; 16 KBytes entspricht Standard) und muß im Hauptmodul stehen. Folgende Compiler-Anweisung definiert einen Stack von 32 KBytes:

```
(*$S 8000*)  (* 32 KBytes, denn 8000 ist Hex-Wert *)
```

Der Stack kann eine Größe von FFFF Bytes (65'535 Bytes) nicht übersteigen. Bitte beachten Sie, daß TopSpeed Modula-2 auch den Compiler-Befehl mit dem Buchstaben S kennt (siehe folgende Beschreibung).

S - Stacküberlauf-Prüfung

Voreinstellung *(*$S-*)*; bei aktiver Schalterstellung wird ein Laufzeitfehler gemeldet, wenn versucht wird, auf dem vollen Stack weitere Werte abzulegen. Der Stack enthält Rücksprung-Adressen von Prozeduren, lokale Variablen und Prozedur-Parameter (siehe auch Compiler-Anweisung S (Stackgröße)). Die Standard-Größe des Stacks beträgt 16 KBytes.

V - Kopieren von offenen Arrays

Voreinstellung *(*$V+*)*; offene Arrays werden normalerweise auf den Stack kopiert, wenn es sich nicht um VAR-Parameter handelt. Bei der Desaktivierung des Compiler-Befehls V bleibt dieser einleitende Kopiervorgang beim Prozedur-Eintritt aus, so daß innerhalb der Prozedur direkt auf die Originaldaten zugegriffen wird (keine guter Programmier-Stil, da der gleiche Effekt mit dem Voranstellen des reservierten Wortes *VAR* erreicht werden kann).

```
MODULE test;
IMPORT IO;
(*$V-*)  (* Kopieren von offenen Array ausschalten *)

  PROCEDURE upper(s:ARRAY OF CHAR);
  VAR
    i:CARDINAL;
  BEGIN
    FOR i:=0 TO HIGH(s) DO
      s[i]:=CAP(s[i])  (* Original-Daten verändern *)
    END
  END upper;
```

```
VAR
  txt:ARRAY [0..255] OF CHAR;
BEGIN
  txt:="Grüß Gott, wie geht's denn immer?";
  upper(txt);    (* wandelt in Groß-Buchstaben um *)
  IO.WrStr(txt)  (* <txt> nun in Groß-Schreibung *)
END test.
```

W - Volatile Variablen

Voreinstellung *(*$W-*)*; TopSpeed Modula-2 versucht, eine Variable solange wie möglich in einem CPU-Register zu halten. Ist dies nicht erwünscht, kann mit dem Compiler-Befehl *(*$W+*)* der Compiler gezwungen werden, eine Variable sofort wieder in den Arbeits-Speicher zu schreiben (Variablen werden volatil (flüchtig) gemacht). Alle globalen Variablen, die zur Kommunikation zwischen parallelen Prozessen eingesetzt werden, sollten volatil sein; ebenso ist es beim Einsatz eines Sourcecode-Debuggers empfehlenswert, mit volantilen Variablen zu arbeiten (da auf diese Weise jederzeit die aktuellen Variablen-Inhalte angezeigt werden).

X - Stack-Erweiterung für 8087

Voreinstellung *(*$X-*)*; bei einer tiefen Verschachtelung von mathematischen Routinen kann der Stack des Coprozessors 8087 überfordert sein. Bei aktiviertem Compiler-Befehl X wird in diesem Fall auf den Haupt-Stack (siehe auch Compiler-Anweisung S) ausgewichen.

Y - Identische variante Teile

Voreinstellung *(*$Y-*)*; falls aktiviert, können variante Teile eines Records dieselben Namen haben. Zu beachten ist jedoch, daß diese Namen identische Daten-Typen und gleiche Offset-Adressen aufweisen (siehe Funktion *Ofs*).

Z - Zeiger-Prüfung

Voreinstellung *(*$Z-*)*; im aktivierten Zustand wird ein Laufzeit-Fehler erzeugt, wenn auf den Inhalt einer Zeigervariablen zugegriffen werden soll, obwohl diese den Wert *NIL* enthält (und somit auf "nichts" zeigt).

Anhang B
Übersicht der Begriffe

An dieser Stelle werden alle zusammengehörigen Routinen und Bezeichner zusammengefaßt. Jeden hier aufgeführte Begriff finden Sie in diesem Buch beschrieben.

Reservierte Wörter

ARRAY, BEGIN, CASE, CONST, DEFINITION, END, EXIT, EXPORT, FOR, FORWARD, FROM . IMPORT, GOTO, IF . THEN . ELSIF . ELSE, IMPLEMENTATION, IMPORT, LABEL, LOOP . END, MODULE, POINTER, PROCEDURE, QUALIFIED, RECORD, REPEAT . UNTIL, RETURN, SET OF, TYPE, VAR, WHILE, WITH

Datentypen

ADDRESS, BITSET, BOOLEAN, BYTE, CARDINAL, CHAR, CHARSET, INTEGER, LONGCARD, LONGINT, LONGREAL, LONGWORD, PROC, REAL, SHORTADDR, SHORTCARD, SHORTINT, WORD

Operatoren

AND, DIV, IN, MOD, NOT, OR (XOR siehe OR)

Standard-Routinen

ABS, ADR, CAP, CHR, DEC, DISPOSE, EXCL, FLOAT, HALT, HIGH, INC, INCL, MAX, MIN, NEW, ODD, ORD, SIZE, TRUNC, VAL, VSIZE

Standard-Konstanten

FALSE, NIL, NULLPROC, TRUE

Moduln

AsmLib, FIO, FloatExc, Graph, IO, Lib, MATHLIB, Process, ProcTrace, Storage, Str, SYSTEM, Window

Modul AsmLib

ActivePage, BufferToScreen, BufferWrite, CompareStr, DosExec, GetInProgramFlag, InitScreenType, PalXlat, ScreenToBuffer, SetInProgramFlag, SetVideoPage, Speaker

Modul FIO

Append, AssignBuffer, ChDir, Close, Create, Erase, Exists, GetDir, GetPos, IOresult, MkDir, Open, Rd<typ>, RdBin, RdItem, RdStr, ReadFirstEntry, ReadNextEntry, Rename, RmDir, Seek, Size, Truncate, Wr<typ>, WrBin, WrCharRep, WrLn, WrStr, WrStrAdj

Modul FloatExc

DisableExceptionHandling, EnableExceptionHandling

Modul Graph

Circle, Disc, Ellipse, GraphMode, HLine, Init<karte>, Line, Plot, Point, Polygon, TextMode

Modul IO

EndOfRd, KeyPressed, Rd<typ>, RdItem, RdKey, RdLn, RdStr, RedirectInput, RedirectOutput, Wr<typ>, WrCharRep, WrLn, WrStr, WrStrAdj

Modul Lib

AddAddr, Compare, DecAddr, Delay, DisableBreakCheck, Dos, EnableBreakCheck, Environment, Execute, FatalError, Fill, HashString, HSort, IncAddr, Intr, LongJmp, MathError, MathError2, Move, NoSound, ParamCount, ParamStr, QSort, RAND, RANDOM, RANDOMIZE, ScanL, ScanNeL, ScanNeR, ScanR, SetJmp, SetReturnCode, Sound, SubAddr, Terminate, UserBreak, WordFill, WordMove

Modul MATHLIB

ACos, ASin, ATan, ATan2, BcdToLong, ClearExceptions, Cos, CosH, Exp, LoadControlWord, Log, Log10, LongToBcd, Mod, Pow, Rexp, Sin, SinH, Sqrt, StoreControlWord, StoreEnvironment, Tan, TanH

Modul Process

Awaited, Delay, Init, Lock, Notify, SEND, StartProcess, StartScheduler, StopScheduler, Unlock, WAIT

Modul ProcTrace

Check, Get_Name, GetCsIp, Install, Monitor

Modul Storage

ALLOCATE, Available, DEALLOCATE, HeapAllocate, HeapAvail, HeapChangeAlloc, HeapChangeSize, HeapDeallocate, HeapTotalAvail, MakeHeap

Modul Str

Append, Caps, CardToStr, Compare, Concat, Copy, Delete, FixRealToStr, Insert, IntToStr, Item, ItemS, Length, Match, Pos, RealToStr, Slice, StrToCard, StrToInt, StrToReal

Modul SYSTEM

CurrentPriority, CurrentProcess, DI, EI, GetFlags, In, InterruptRegisters, IOTRANSFER, Listen, NewPriority, NEWPROCESS, Ofs, Out, Seg, SetFlags, TRANSFER

Modul Window

At, Change, Clear, Close, ClrEol, ConvertCoords, CursorOff, CursorOn, DelLine, DirectWrite, GotoXY, Hide, Info, InsLine, ObscuredAt, Open, PaletteColor, PaletteColorUsed, PaletteOpen, PutBeneath, PutOnTop, RdBufferLn, SetFrame, SetPalette, SetPaletteColor, SetProcessLocks, SetTitle, SetWrap, SnapShot, TextBackground, TextColor, Top, Use, Used, WhereX, WhereY, WrBufferLn