Quellcode
der insertierten ELAN – Pakete

Version:	1.8.0
Teil:	Single – User
Stand:	10.11.86

Herausgeber:

Gesellschaft für Mathematik und Datenverarbeitung mbH

Postfach 1240, Schloß Birlinghoven
D-5205 Sankt Augustin 1
Telefon(02241) 14-1, Telex 8 89 469 gmd d
Telefax(02241) 14 28 89, BTX *43900#
Teletex 2627-224135=GMDVV

Texterstellung:

Dieser Text wurde mit der EUMEL-Textverarbeitung erstellt und aufbereitet und mit dem Agfa Laserdrucksystem P400 gedruckt.

Umschlaggestaltung:

Hannelotte Wecken

Hinweis:

Diese Dokumentation wurde mit größtmöglicher Sorgfalt erstellt. Dennoch wird für die Korrektheit und Vollständigkeit der gemachten Angaben keine Gewähr übernommen. Bei vermuteten Fehlern der Software oder der Dokumentation bitten wir um baldige Meldung, damit eine Korrektur möglichst rasch erfolgen kann. Anregungen und Kritik sind jederzeit willkommen.

Inhaltsverzeichnis

1. Übersicht über die insertierten Pakete

'(M)'	vor der Paketnummer heißt, daß dies Objekt nur im Multi-User vorhanden ist.
'(S)'	vor der Paketnummer heißt, daß dies Objekt nur im Single-User vorhanden ist.
'(T)'	vor der Paketnummer heißt, daß dies Objekt nur in einem System mit Textverarbeitung vorhanden ist.

Die Paketnummer ergibt sich aus der Reihenfolge, in der die Pakete im Multi-User mit Textverarbeitung insertiert wurden. Bitte beachten Sie, daß diese Reihenfolge nicht der Insertierungsreihenfolge im Single-User entspricht. Der Quellcode der insertierten Pakete ist in Teil 4 nach Paketnummern sortiert.

1. a
2. bits
3. text
4. pcb and init control
5. dataspace
6. basic transput
7. bool
8. integer
9. error handling
10. real
11. date handling
12. command dialogue
13. thesaurus handling
14. local manager
15. pattern match
16. file handling
17. elan do interface
18. scanner
19. screen description
20. tasten verwaltung
21. editor paket
22. editor functions
23. std transput
24. local manager part 2
25. eumel coder part 1
26. mathlib
27. command handler

S29. advertising
S30. tasks single
S31. font store
48. basic archive
S49. archive single
39. nameset

2. Übersicht über die exportierten Objekte nach Paketen geordnet:

'(M)'	vor der Paketnummer heißt, daß dies Objekt nur im Multi-User vorhanden ist.
'(S)'	vor der Paketnummer heißt, daß dies Objekt nur im Single-User vorhanden ist.
'(T)'	vor der Paketnummer heißt, daß dies Objekt nur in einem System mit Textverarbeitung vorhanden ist.

Die Paketnummer ergibt sich aus der Reihenfolge, in der die Pakete im Multi-User mit Textverarbeitung insertiert wurden. Bitte beachten Sie, daß diese Reihenfolge nicht der Insertierungsreihenfolge im Single-User entspricht. Der Quellcode der insertierten Pakete ist in Teil 4 nach Paketnummern sortiert.

PACKET a :

PACKET bits :

PROC rotate (INT VAR bits, INT CONST number of bits)	2-19
INT OP AND (INT CONST left, right)	2-23
INT OP OR (INT CONST left, right)	2-27
INT OP XOR (INT CONST left, right)	2-31
BOOL PROC bit (INT CONST bits, bit no)	2-35
PROC set bit (INT VAR bits, INT CONST bit no)	2-41
PROC reset bit (INT VAR bits, INT CONST bit no)	2-47
INT PROC lowest set (INT CONST bits)	2-53
INT PROC lowest reset (INT CONST bits)	2-65

PACKET text :

INT CONST max text length	3-35
TEXT OP SUB (TEXT CONST text, INT CONST pos)	3-37
TEXT PROC subtext (TEXT CONST source, INT CONST from, to)	3-41
TEXT PROC subtext (TEXT CONST source, INT CONST from)	3-45
INT PROC code (TEXT CONST text)	3-49
TEXT PROC code (INT CONST code)	3-53
INT OP ISUB (TEXT CONST text, INT CONST index)	3-57
PROC replace (TEXT VAR text, INT CONST index, value)	3-61
REAL OP RSUB (TEXT CONST text, INT CONST index)	3-65
PROC replace (TEXT VAR text, INT CONST index, REAL CONST code)	3-69
PROC replace (TEXT VAR dest, INT CONST pos, TEXT CONST source)	3-74
TEXT PROC text (TEXT CONST source, INT CONST length)	3-78
TEXT PROC text (TEXT CONST source, INT CONST length, from)	3-95
OP CAT (TEXT VAR right, TEXT CONST left)	3-99
TEXT OP + (TEXT CONST left, right)	3-103
TEXT OP * (INT CONST times, TEXT CONST source)	3-109
INT PROC length (TEXT CONST text)	3-120
INT OP LENGTH (TEXT CONST text)	3-124
INT PROC pos (TEXT CONST source, pattern)	3-128
INT PROC pos (TEXT CONST source, pattern, INT CONST from)	3-132
INT PROC pos (TEXT CONST source, pattern, INT CONST from, to)	3-136
INT PROC pos (TEXT CONST source, low, high, INT CONST from)	3-140

TEXT PROC incharety (INT CONST time limit)	6-95
PROC pause (INT CONST time limit)	6-100
PROC pause	6-105
PROC cat input (TEXT VAR t, esc char)	6-113
PROC cursor (INT CONST x, y)	6-118
PROC get cursor (INT VAR x, y)	6-124
PROC cout (INT CONST number)	6-128
INT PROC channel	6-133
BOOL PROC online	6-137
PROC control (INT CONST code1, code2, code3, INT VAR return code)	6-142
PROC blockout (ROW 256 INT CONST block, INT CONST code1, code2, INT VAR return code)	6-146
PROC blockin (ROW 256 INT VAR block, INT CONST code1, code2, INT VAR return code)	6-161

PACKET bool :

BOOL CONST true	7-4
BOOL CONST false	7-4
BOOL OP XOR (BOOL CONST left, right)	7-7

PACKET integer :

INT PROC minint	8-7
INT PROC maxint	8-9
TEXT PROC text (INT CONST number)	8-12
TEXT PROC text (INT CONST number, length)	8-25
INT PROC int (TEXT CONST number)	8-38
INT OP MOD (INT CONST left, right)	8-95
INT PROC sign (INT CONST argument)	8-101
INT OP SIGN (INT CONST argument)	8-110
INT PROC abs (INT CONST argument)	8-114
INT OP ABS (INT CONST argument)	8-122
INT OP ** (INT CONST arg, exp)	8-126
INT PROC min (INT CONST first, second)	8-154
INT PROC max (INT CONST first, second)	8-160
BOOL PROC last conversion ok	8-170
PROC set conversion (BOOL CONST success)	8-174
PROC initialize random (INT CONST start)	8-197
INT PROC random (INT CONST lower bound, upper bound)	8-208

PACKET error handling :

PROC enable stop	9-28
PROC disable stop	9-32
BOOL PROC is error	9-40
PROC clear error	9-44
TEXT PROC error message	9-72
INT PROC error code	9-79
INT PROC error line	9-85
PROC errorstop (TEXT CONST message)	9-101
PROC errorstop (INT CONST code, TEXT CONST message)	9-107
PROC put error	9-116
PROC stop	9-135

PACKET real :

REAL PROC max real	10-40
REAL PROC small real	10-42
INT PROC decimal exponent (REAL CONST mantissa)	10-48
PROC set exp (INT CONST exponent, REAL VAR number)	10-52
REAL PROC floor (REAL CONST real)	10-62
REAL PROC round (REAL CONST real, INT CONST digits)	10-66
TEXT PROC text (REAL CONST real)	10-88
TEXT PROC text (REAL CONST real, INT CONST length)	10-155
TEXT PROC text (REAL CONST real, INT CONST length, fracs)	10-199
REAL PROC real (TEXT CONST text)	10-243
REAL PROC abs (REAL CONST value)	10-329
REAL OP ABS (REAL CONST value)	10-338
INT PROC sign (REAL CONST value)	10-344
INT OP SIGN (REAL CONST value)	10-353
REAL OP MOD (REAL CONST left, right)	10-359
REAL PROC frac (REAL CONST value)	10-369
REAL PROC max (REAL CONST a, b)	10-375
REAL PROC min (REAL CONST a, b)	10-381
OP INCR (REAL VAR dest, REAL CONST increment)	10-387
OP DECR (REAL VAR dest, REAL CONST decrement)	10-393
INT PROC int (REAL CONST value)	10-399
REAL PROC real (INT CONST value)	10-424

PACKET date handling :

REAL PROC day	11-28
REAL PROC hour	11-29
REAL PROC minute	11-30
REAL PROC second	11-31
TEXT PROC date	11-33
TEXT PROC date (REAL CONST datum)	11-44
TEXT PROC day (REAL CONST datum)	11-128
TEXT PROC month (REAL CONST datum)	11-139
TEXT PROC year (REAL CONST datum)	11-156
TEXT PROC time of day	11-166
TEXT PROC time of day (REAL CONST value)	11-170
TEXT PROC time (REAL CONST value)	11-174
TEXT PROC time (REAL CONST value, INT CONST length)	11-178
REAL PROC date (TEXT CONST datum)	11-201
REAL PROC time (TEXT CONST time)	11-260
REAL CONST hour	11-265

PACKET command dialogue :

TYPE QUIET	12-29
QUIET PROC quiet	12-31
BOOL PROC command dialogue	12-36
PROC command dialogue (BOOL CONST status)	12-40
BOOL PROC yes (TEXT CONST question)	12-45
BOOL PROC no (TEXT CONST question)	12-81
PROC say (TEXT CONST message)	12-87
PROC param position (INT CONST x)	12-95
TEXT PROC last param	12-101
PROC last param (TEXT CONST new)	12-114
TEXT PROC std	12-118

PACKET thesaurus handling :

TYPE THESAURUS	13-17
THESAURUS PROC empty thesaurus	13-96
OP := (THESAURUS VAR dest, THESAURUS CONST source)	13-103
PROC insert (THESAURUS VAR thesaurus, TEXT CONST name, INT VAR index)	13-111
PROC insert (THESAURUS VAR thesaurus, TEXT CONST name)	13-170
PROC delete (THESAURUS VAR thesaurus, TEXT CONST name, INT VAR index)	13-180
PROC delete (THESAURUS VAR thesaurus, INT CONST index)	13-188
BOOL OP CONTAINS (THESAURUS CONST thesaurus, TEXT CONST name)	13-231
PROC rename (THESAURUS VAR thesaurus, TEXT CONST old, new)	13-242
PROC rename (THESAURUS VAR thesaurus, INT CONST index, TEXT CONST new)	13-248
INT PROC link (THESAURUS CONST thesaurus, TEXT CONST name)	13-267
TEXT PROC name (THESAURUS CONST thesaurus, INT CONST index)	13-274
PROC get (THESAURUS CONST thesaurus, TEXT VAR name, INT VAR index)	13-281
INT PROC highest entry (THESAURUS CONST thesaurus)	13-323

PACKET local manager :

PROC create (TEXT CONST name)	14-61
DATASPACE PROC new (TEXT CONST name)	14-86
DATASPACE PROC old (TEXT CONST name)	14-96
DATASPACE PROC old (TEXT CONST name, INT CONST expected type)	14-110
BOOL PROC exists (TEXT CONST name)	14-127
PROC forget (TEXT CONST name)	14-134
PROC forget (TEXT CONST name, QUIET CONST q)	14-145
PROC forget	14-157
PROC status (TEXT CONST name, status text)	14-166
TEXT PROC status (TEXT CONST name)	14-176
PROC status (INT CONST pos, TEXT CONST status pattern)	14-187
PROC copy (DATASPACE CONST source, TEXT CONST dest name)	14-200
PROC copy (TEXT CONST source name, dest name)	14-218
PROC rename (TEXT CONST old name, new name)	14-224
PROC begin list	14-237
PROC get list entry (TEXT VAR entry, status text)	14-244
TEXT PROC write password	14-257
TEXT PROC read password	14-263
PROC enter password (TEXT CONST password)	14-270
PROC enter password (TEXT CONST file name, write pass, read pass)	14-284
BOOL PROC read permission (TEXT CONST name, supply password)	14-312
BOOL PROC write permission (TEXT CONST name, supply password)	14-335
THESAURUS PROC all	14-358

PACKET pattern match :

TEXT OP - (TEXT CONST alphabet)	15-43
TEXT OP OR (TEXT CONST a, b)	15-54
TEXT OP ** (TEXT CONST p, INT CONST x)	15-58
TEXT CONST any	15-62
TEXT PROC any (INT CONST n)	15-64
TEXT PROC any (TEXT CONST a)	15-70
TEXT PROC any (INT CONST n, TEXT CONST a)	15-72
TEXT PROC notion (TEXT CONST t)	15-78
TEXT CONST bound	15-87
TEXT PROC match (INT CONST x)	15-91
INT PROC matchpos (INT CONST x)	15-95
INT PROC matchend (INT CONST x)	15-97

TEXT PROC somefix (TEXT CONST pattern)	15-123
BOOL OP UNLIKE (TEXT CONST t, p)	15-191
BOOL OP LIKE (TEXT CONST t, pattern)	15-193
TEXT PROC notion (TEXT CONST t, INT CONST r)	15-765

PACKET file handling :

TYPE FILE	16-150
TYPE FRANGE	16-152
OP := (FILE VAR left, FILE CONST right)	16-160
TEXT PROC prefix (TEXT CONST pattern)	16-546
BOOL PROC pattern found	16-674
TRANSPUTDIRECTION PROC input	16-851
TRANSPUTDIRECTION PROC output	16-856
TRANSPUTDIRECTION PROC modify	16-861
FILE PROC sequential file (TRANSPUTDIRECTION CONST mode, DATASPACE CONST ds)	16-869
FILE PROC sequential file (TRANSPUTDIRECTION CONST mode, TEXT CONST name)	16-885
PROC reset (FILE VAR f)	16-926
PROC reset (FILE VAR f, TRANSPUTDIRECTION CONST mode)	16-935
PROC input (FILE VAR f)	16-962
PROC output (FILE VAR f)	16-969
PROC modify (FILE VAR f)	16-976
PROC close (FILE VAR f)	16-983
PROC to line (FILE VAR f, INT CONST destination line)	16-1023
PROC to first record (FILE VAR f)	16-1031
PROC to eof (FILE VAR f)	16-1038
PROC putline (FILE VAR f, TEXT CONST word)	16-1045
PROC delete record (FILE VAR f)	16-1055
PROC insert record (FILE VAR f)	16-1063
PROC down (FILE VAR f)	16-1071
PROC up (FILE VAR f)	16-1078
PROC down (FILE VAR f, INT CONST n)	16-1085
PROC up (FILE VAR f, INT CONST n)	16-1091
PROC write record (FILE VAR f, TEXT CONST record)	16-1098
PROC read record (FILE CONST f, TEXT VAR record)	16-1111
PROC line (FILE VAR f)	16-1119
PROC line (FILE VAR f, INT CONST lines)	16-1145
PROC getline (FILE VAR f, TEXT VAR text)	16-1152
BOOL PROC is first record (FILE CONST f)	16-1174
BOOL PROC eof (FILE CONST f)	16-1182
INT PROC line no (FILE CONST f)	16-1197
PROC line type (FILE VAR f, INT CONST t)	16-1204
INT PROC line type (FILE CONST f)	16-1210
PROC put (FILE VAR f, TEXT CONST word)	16-1217
PROC put (FILE VAR f, INT CONST value)	16-1236
PROC put (FILE VAR f, REAL CONST real)	16-1243
PROC write (FILE VAR f, TEXT CONST word)	16-1250
PROC get (FILE VAR f, TEXT VAR word, TEXT CONST separator)	16-1268
PROC get (FILE VAR f, TEXT VAR word, INT CONST max length)	16-1312
PROC get (FILE VAR f, TEXT VAR word)	16-1336
PROC get (FILE VAR f, INT VAR number)	16-1346
PROC get (FILE VAR f, REAL VAR number)	16-1354
PROC split line (FILE VAR f, INT CONST split col)	16-1365
PROC split line (FILE VAR f, INT CONST split col, BOOL CONST note indentation)	16-1371
PROC concatenate line (FILE VAR f, BOOL CONST delete blanks)	16-1407
PROC set range (FILE VAR f, INT CONST start line, start col, FRANGE VAR old range)	16-1550

PROC set range (FILE VAR f, FRANGE VAR new range)	16-1620
PROC reset range (FILE VAR f)	16-1668
PROC remove (FILE VAR f, INT CONST size)	16-1679
PROC clear removed (FILE VAR f)	16-1687
PROC reinsert (FILE VAR f)	16-1695
PROC copy attributes (FILE CONST source file, FILE VAR dest file)	16-1703
INT PROC max line length (FILE CONST f)	16-1716
PROC max line length (FILE VAR f, INT CONST new limit)	16-1723
TEXT PROC headline (FILE CONST f)	16-1732
PROC headline (FILE VAR f, TEXT CONST head)	16-1739
PROC get tabs (FILE CONST f, TEXT VAR tabs)	16-1746
PROC put tabs (FILE VAR f, TEXT CONST tabs)	16-1753
INT PROC edit info (FILE CONST f)	16-1760
PROC edit info (FILE VAR f, INT CONST info)	16-1767
INT PROC lines (FILE CONST f)	16-1774
INT PROC removed lines (FILE CONST f)	16-1781
INT PROC segments (FILE CONST f)	16-1788
INT PROC col (FILE CONST f)	16-1795
PROC col (FILE VAR f, INT CONST new column)	16-1801
TEXT PROC word (FILE CONST f)	16-1809
TEXT PROC word (FILE CONST f, TEXT CONST delimiter)	16-1815
TEXT PROC word (FILE CONST f, INT CONST max length)	16-1825
BOOL PROC at (FILE CONST f, TEXT CONST word)	16-1831
PROC exec (PROC (TEXT VAR, TEXT CONST) proc, FILE VAR f, TEXT CONST t)	16-1844
PROC exec (PROC (TEXT VAR, INT CONST) proc, FILE VAR f, INT CONST i)	16-1853
INT PROC pos (FILE CONST f, TEXT CONST pattern, INT CONST i)	16-1861
PROC down (FILE VAR f, TEXT CONST pattern)	16-1869
PROC down (FILE VAR f, TEXT CONST pattern, INT CONST max line)	16-1875
PROC downety (FILE VAR f, TEXT CONST pattern)	16-1884
PROC downety (FILE VAR f, TEXT CONST pattern, INT CONST max line)	16-1890
PROC up (FILE VAR f, TEXT CONST pattern)	16-1899
PROC up (FILE VAR f, TEXT CONST pattern, INT CONST max line)	16-1905
PROC uppety (FILE VAR f, TEXT CONST pattern)	16-1914
PROC uppety (FILE VAR f, TEXT CONST pattern, INT CONST max line)	16-1920
INT PROC len (FILE CONST f)	16-1930
TEXT PROC subtext (FILE CONST f, INT CONST from, to)	16-1938
PROC change (FILE VAR f, INT CONST from, to, TEXT CONST new)	16-1946
BOOL PROC mark (FILE CONST f)	16-1956
PROC mark (FILE VAR f, INT CONST line no, col)	16-1962
INT PROC mark line no (FILE CONST f)	16-1973
INT PROC mark col (FILE CONST f)	16-1982
PROC set marked range (FILE VAR f, FRANGE VAR old range)	16-1993
PROC sort (TEXT CONST dateiname)	16-2016
PROC sort (TEXT CONST dateiname, INT CONST sortieranfang)	16-2020
PROC lex sort (TEXT CONST dateiname)	16-2025
PROC lex sort (TEXT CONST dateiname, INT CONST sortieranfang)	16-2029

PACKET elan do interface :

PROC do (TEXT CONST command)	17-23
PROC no do again	17-44

PACKET scanner :

PROC scan (TEXT CONST scan text)	18-36
PROC continue scan (TEXT CONST scan text)	18-44
PROC next symbol (TEXT VAR symbol)	18-52
PROC next symbol (TEXT VAR symbol, INT VAR type)	18-59

PROC scan (FILE VAR f)	18-296
PROC next symbol (FILE VAR f, TEXT VAR symbol)	18-303
PROC next symbol (FILE VAR f, TEXT VAR symbol, INT VAR type)	18-312

PACKET screen description :

INT PROC xsize	19-9
INT PROC ysize	19-11
INT PROC marksize	19-13
PROC xsize (INT CONST i)	19-15
PROC ysize (INT CONST i)	19-17
PROC marksize (INT CONST i)	19-19
BOOL PROC mark refresh line mode	19-24
PROC mark refresh line mode (BOOL CONST b)	19-28

PACKET tasten verwaltung :

PROC lernsequenz auf taste legen (TEXT CONST taste, lernsequenz)	20-27
TEXT PROC lernsequenz auf taste (TEXT CONST taste)	20-63
PROC kommando auf taste legen (TEXT CONST taste, kommando)	20-71
TEXT PROC kommando auf taste (TEXT CONST taste)	20-79
BOOL PROC taste enthaelt kommando (TEXT CONST taste)	20-87
PROC std tastenbelegung	20-92

PACKET editor paket :

PROC editget command (BOOL CONST schalter)	21-81
PROC editget (TEXT VAR editsatz, INT CONST editlimit, editlaenge, TEXT CONST sep, res, TEXT VAR exit char)	21-87
PROC editget (TEXT VAR editsatz, INT CONST editlimit, TEXT VAR exit char)	21-192
PROC editget (TEXT VAR editsatz, TEXT CONST sep, res, TEXT VAR exit char)	21-196
PROC editget (TEXT VAR editsatz)	21-200
PROC editget (TEXT VAR editsatz, INT CONST editlimit, editlaenge)	21-205
BOOL PROC is editget	21-913
PROC get editline (TEXT VAR editline, INT VAR editpos, editmarke)	21-917
PROC put editline (TEXT CONST editline, INT CONST editpos, editmarke)	21-925
BOOL PROC within kanji (TEXT CONST satz, INT CONST stelle)	21-933
BOOL PROC is kanji esc (TEXT CONST char)	21-947
BOOL PROC two bytes	21-952
PROC two bytes (BOOL CONST new mode)	21-954
BOOL PROC write permission	21-989
PROC push (TEXT CONST ausfuehrkommando)	21-991
PROC type (TEXT CONST ausfuehrkommando)	21-1009
PROC getchar (TEXT VAR zeichen)	21-1173
BOOL PROC is incharety (TEXT CONST muster)	21-1194
TEXT PROC getcharety	21-1208
PROC get editcursor (INT VAR x, y)	21-1219
INT PROC aktueller editor	21-1279
INT PROC groesster editor	21-1281
PROC quit last	21-2144
PROC quit	21-2153
INT CONST aktueller editor	21-2158
PROC nichts neu	21-2217
PROC satznr neu	21-2219
PROC ueberschrift neu	21-2221
PROC zeile neu	21-2223
PROC abschnitt neu (INT CONST von satznr, bis satznr)	21-2228

PROC bildabschnitt neu (INT CONST von zeile, bis zeile)	21-2237
PROC bild neu	21-2249
PROC bild neu (FILE VAR f)	21-2251
PROC alles neu	21-2261
PROC satznr zeigen	21-2270
PROC ueberschrift zeigen	21-2274
PROC bild zeigen	21-2417
PROC set busy indicator	21-2508
PROC word wrap (BOOL CONST b)	21-2607
BOOL PROC word wrap	21-2623
INT PROC margin	21-2630
PROC margin (INT CONST i)	21-2632
BOOL PROC rubin mode	21-2653
BOOL PROC rubin mode (INT CONST editor nr)	21-2655
PROC edit (INT CONST i, TEXT CONST res, PROC (TEXT CONST) kommando interpreter)	21-2665
PROC edit (INT CONST von, bis, start, TEXT CONST res, PROC (TEXT CONST) kommando interpreter)	21-2670
PROC open editor (FILE CONST new file, BOOL CONST access)	21-2753
PROC open editor (INT CONST editor nr, FILE CONST new file, BOOL CONST access, INT CONST x start, y, x len start, y len)	21-2792
PROC open editor (INT CONST i)	21-2881
FILE PROC editfile	21-2929
PROC get window (INT VAR x, y, x size, y size)	21-2935

PACKET editor functions :

PROC std kommando interpreter (TEXT CONST taste)	22-58
PROC edit (FILE VAR f)	22-176
PROC edit (FILE VAR f, INT CONST x, y, x size, y size)	22-186
PROC edit (FILE VAR f, TEXT CONST res, PROC (TEXT CONST) kdo interpreter)	22-193
PROC edit	22-200
PROC edit (TEXT CONST filename)	22-233
PROC edit (TEXT CONST filename, INT CONST x, y, x size, y size)	22-250
PROC edit (INT CONST i)	22-261
PROC show (FILE VAR f)	22-266
PROC show (TEXT CONST filename)	22-273
PROC show	22-283
OP PUT (TEXT CONST filename)	22-293
OP P (TEXT CONST filename)	22-362
OP GET (TEXT CONST filename)	22-367
OP G (TEXT CONST filename)	22-438
INT PROC len	22-443
PROC col (INT CONST stelle)	22-448
INT PROC col	22-453
PROC limit (INT CONST limit)	22-458
INT PROC limit	22-463
INT PROC lines	22-468
INT PROC line no	22-473
PROC to line (INT CONST satz nr)	22-478
OP T (INT CONST satz nr)	22-488
PROC down (INT CONST anz)	22-493
OP D (INT CONST anz)	22-498
PROC up (INT CONST anz)	22-503
OP U (INT CONST anz)	22-508
PROC down (TEXT CONST muster)	22-513
OP D (TEXT CONST muster)	22-525
PROC down (TEXT CONST muster, INT CONST anz)	22-530

PROC up (TEXT CONST muster)	22-535
OP U (TEXT CONST muster)	22-547
PROC up (TEXT CONST muster, INT CONST anz)	22-552
PROC downety (TEXT CONST muster)	22-557
PROC downety (TEXT CONST muster, INT CONST anz)	22-565
PROC uppety (TEXT CONST muster)	22-570
PROC uppety (TEXT CONST muster, INT CONST anz)	22-578
OP C (TEXT CONST old, new)	22-583
OP C (TEXT CONST replacement)	22-587
PROC change to (TEXT CONST old, new)	22-596
OP CA (TEXT CONST old, new)	22-613
PROC change all (TEXT CONST old, new)	22-618
BOOL PROC eof	22-623
BOOL PROC mark	22-628
PROC mark (BOOL CONST mark on)	22-633
BOOL PROC at (TEXT CONST pattern)	22-642
TEXT PROC word	22-646
TEXT PROC word (TEXT CONST sep)	22-651
TEXT PROC word (INT CONST len)	22-656
PROC note (TEXT CONST text)	22-671
PROC note (INT CONST number)	22-677
PROC note line	22-683
BOOL PROC anything noted	22-689
FILE PROC note file	22-695
PROC note edit (FILE VAR context)	22-701
PROC note edit	22-731

PACKET std transput :

PROC sysout (TEXT CONST file name)	23-33
TEXT PROC sysout	23-44
PROC sysin (TEXT CONST file name)	23-48
TEXT PROC sysin	23-59
PROC put (TEXT CONST word)	23-64
PROC put (INT CONST number)	23-73
PROC put (REAL CONST number)	23-79
PROC putline (TEXT CONST textline)	23-85
PROC line	23-94
PROC line (INT CONST times)	23-103
PROC page	23-112
PROC write (TEXT CONST word)	23-120
PROC get (TEXT VAR word)	23-130
PROC get (TEXT VAR word, TEXT CONST separator)	23-152
PROC get (INT VAR number)	23-175
PROC get (REAL VAR number)	23-182
PROC get (TEXT VAR word, INT CONST length)	23-189
PROC getline (TEXT VAR textline)	23-203
PROC get secret line (TEXT VAR textline)	23-217

PACKET local manager part 2 :

PROC list	24-10
PROC list (FILE VAR f)	24-22

PACKET eumel coder part 1 :

PROC help (TEXT CONST proc name)	25-328
PROC bulletin (TEXT CONST packet name)	25-559
PROC bulletin	25-679
PROC packets	25-701
PROC run (TEXT CONST file name)	25-735
PROC run	25-744
PROC run again	25-748
PROC insert (TEXT CONST file name)	25-756
PROC insert	25-765
PROC prot (TEXT CONST file name)	25-826
PROC prot off	25-832
BOOL PROC prot	25-837
PROC check on	25-841
PROC check off	25-845
BOOL PROC check	25-849
PROC warnings on	25-853
PROC warnings off	25-857
BOOL PROC warnings	25-861

PACKET mathlib :

REAL PROC pi	26-25
REAL PROC e	26-26
REAL PROC ln (REAL CONST x)	26-28
REAL PROC log10 (REAL CONST x)	26-32
REAL PROC log2 (REAL CONST z)	26-36
REAL PROC sqrt (REAL CONST z)	26-64
REAL PROC exp (REAL CONST z)	26-83
REAL PROC tan (REAL CONST x)	26-111
REAL PROC tand (REAL CONST x)	26-116
REAL PROC sin (REAL CONST x)	26-146
REAL PROC sind (REAL CONST x)	26-154
REAL PROC cos (REAL CONST x)	26-162
REAL PROC cosd (REAL CONST x)	26-170
REAL PROC arctan (REAL CONST y)	26-204
REAL PROC arctand (REAL CONST x)	26-218
REAL OP ** (REAL CONST b, e)	26-222
REAL OP ** (REAL CONST a, INT CONST b)	26-231
REAL PROC random	26-259
PROC initializerandom (REAL CONST z)	26-263

PACKET command handler :

PROC get command (TEXT CONST command text)	27-33
PROC get command (TEXT CONST command text, TEXT VAR command line)	27-39
PROC analyze command (TEXT CONST command list, INT CONST permitted type, INT VAR command index, number of params, TEXT VAR param 1, param 2)	27-106
PROC analyze command (TEXT CONST command list, command line, INT CONST permitted type, INT VAR command index, number of params, TEXT VAR param 1, param 2)	27-117
PROC do command	27-233
PROC command error	27-247
PROC cover tracks	27-266
PROC cover tracks (TEXT VAR secret)	27-277

PACKET advertising :

SOME PROC eumel must advertise	S29-4

PACKET tasks single :

TYPE TASK	S30-38
TASK PROC myself	S30-44
OP := (TASK VAR dest, TASK CONST source)	S30-51
BOOL OP = (TASK CONST left, right)	S30-57
BOOL PROC is niltask (TASK CONST t)	S30-63
INT PROC pcb (TASK CONST id, INT CONST field)	S30-69
INT PROC status (TASK CONST id)	S30-75
INT PROC channel (TASK CONST id)	S30-81
REAL PROC clock (TASK CONST id)	S30-87
INT PROC storage (TASK CONST id)	S30-93
PROC continue (INT CONST channel no)	S30-112
INT PROC dataspaces	S30-121

PACKET font store :

PROC font table (TEXT CONST new font table)	S31-88
TEXT PROC font table	S31-128
PROC list font tables	S31-135
PROC list fonts (TEXT CONST name)	S31-164
PROC list fonts	S31-176
INT PROC x step conversion (REAL CONST cm)	S31-218
REAL PROC x step conversion (INT CONST steps)	S31-229
INT PROC y step conversion (REAL CONST cm)	S31-237
REAL PROC y step conversion (INT CONST steps)	S31-248
TEXT PROC on string (INT CONST modification)	S31-256
TEXT PROC off string (INT CONST modification)	S31-270
INT PROC font (TEXT CONST font name)	S31-284
TEXT PROC font (INT CONST font number)	S31-298
BOOL PROC font exists (TEXT CONST font name)	S31-311
BOOL PROC next larger font exists (INT CONST font number, INT VAR next larger font)	S31-318
BOOL PROC next smaller font exists (INT CONST font number, INT VAR next smaller font)	S31-338
INT PROC font lead (INT CONST font number)	S31-358
INT PROC font height (INT CONST font number)	S31-371
INT PROC font depth (INT CONST font number)	S31-384
INT PROC indentation pitch (INT CONST font number)	S31-397
INT PROC char pitch (INT CONST font number, TEXT CONST char)	S31-410
INT PROC extended char pitch (INT CONST font number, TEXT CONST esc char, char)	S31-430
TEXT PROC replacement (INT CONST font number, TEXT CONST char)	S31-452
TEXT PROC extended replacement (INT CONST font number, TEXT CONST esc char, char)	S31-480
TEXT PROC font string (INT CONST font number)	S31-537
TEXT PROC y offsets (INT CONST font number)	S31-550
INT PROC bold offset (INT CONST font number)	S31-563
PROC get font (INT CONST font number, INT VAR indentation pitch, font lead, font height, font depth, ROW 256 INT VAR pitch table)	S31-576
PROC get replacements (INT CONST font number, TEXT VAR replacements, ROW 256 INT VAR replacements table)	S31-595

PACKET basic archive :

INT PROC block number	48-43
PROC seek (INT CONST block)	48-47
PROC rewind	48-51
PROC skip dataspace	48-59
PROC read (DATASPACE VAR ds)	48-70
PROC read (DATASPACE VAR ds, INT CONST max pages, BOOL CONST error accept)	48-74
PROC check read	48-118
PROC write (DATASPACE CONST ds)	48-134
PROC read block (DATASPACE VAR ds, INT CONST ds page no, INT CONST block no, INT VAR return code)	48-268
PROC write block (DATASPACE CONST ds, INT CONST ds page no, INT CONST mode, INT CONST block no, INT VAR return code)	48-292
INT PROC size (INT CONST key)	48-321
INT PROC archive blocks	48-329
PROC search dataspace (INT VAR ds pages)	48-333
PROC format archive (INT CONST format code)	48-378

PACKET archive single :

PROC archive (TEXT CONST name)	S49-72
PROC release (TASK CONST t)	S49-81
PROC fetch (TEXT CONST file name)	S49-215
PROC fetch (TEXT CONST file name, TASK CONST from)	S49-221
PROC erase	S49-308
PROC erase (TEXT CONST file name)	S49-314
PROC erase (TEXT CONST file name, TASK CONST dest)	S49-320
PROC save	S49-411
PROC save (TEXT CONST file name)	S49-417
PROC save (TEXT CONST file name, TASK CONST to)	S49-423
PROC check (TEXT CONST name, TASK CONST from)	S49-534
BOOL PROC exists (TEXT CONST name, TASK CONST from)	S49-563
PROC list (TASK CONST from)	S49-576
PROC list (FILE VAR list file, TASK CONST from)	S49-588
THESAURUS OP ALL (TASK CONST from)	S49-649
PROC clear (TASK CONST dest)	S49-678
PROC format (INT CONST format code, TASK CONST dest)	S49-722
PROC format (TASK CONST dest)	S49-740

PACKET name set :

THESAURUS OP + (THESAURUS CONST left, right)	39-32
THESAURUS OP + (THESAURUS CONST left, TEXT CONST right)	39-47
THESAURUS OP - (THESAURUS CONST left, right)	39-57
THESAURUS OP - (THESAURUS CONST left, TEXT CONST right)	39-72
THESAURUS OP / (THESAURUS CONST left, right)	39-81
THESAURUS OP ALL (TEXT CONST file name)	39-96
THESAURUS OP SOME (THESAURUS CONST thesaurus)	39-105
THESAURUS OP SOME (TASK CONST task)	39-130
THESAURUS OP SOME (TEXT CONST file name)	39-136
THESAURUS OP LIKE (THESAURUS CONST thesaurus, TEXT CONST pattern)	39-142
THESAURUS PROC remainder	39-157
PROC do (PROC (TEXT CONST) operate, THESAURUS CONST thesaurus)	39-163
PROC do (PROC (TEXT CONST, TASK CONST) operate, THESAURUS CONST thesaurus, TASK CONST task)	39-199

OP FILLBY (THESAURUS VAR thesaurus, FILE VAR file)	39-237
OP FILLBY (FILE VAR file, THESAURUS CONST thesaurus)	39-254
OP FILLBY (TEXT CONST file name, THESAURUS CONST thesaurus)	39-267
PROC fetch (THESAURUS CONST nameset)	39-276
PROC fetch (THESAURUS CONST nameset, TASK CONST task)	39-282
PROC save (THESAURUS CONST nameset)	39-288
PROC save (THESAURUS CONST nameset, TASK CONST task)	39-294
PROC fetch all	39-300
PROC fetch all (TASK CONST manager)	39-306
PROC save all	39-312
PROC save all (TASK CONST manager)	39-318
PROC forget (THESAURUS CONST nameset)	39-324
PROC erase (THESAURUS CONST nameset)	39-330
PROC erase (THESAURUS CONST nameset, TASK CONST task)	39-336
PROC insert (THESAURUS CONST nameset)	39-342
PROC edit (THESAURUS CONST nameset)	39-348

PACKET system info :

PROC task status	S40-34
PROC storage info	S40-45
PROC help	S40-61
PROC help (FILE VAR help file)	S40-71

PACKET konfigurieren :

PROC new configuration	52-39
PROC flow (INT CONST nr, INT CONST dtype)	52-68
PROC ysize (INT CONST channel, new size, INT VAR old size)	52-72
PROC input buffer size (INT CONST nr, size)	52-76
PROC baudrate (INT CONST nr, rate)	52-81
PROC bits (INT CONST channel, number, parity)	52-85
PROC bits (INT CONST channel, key)	52-89
PROC new type (TEXT CONST dtype)	52-105
PROC link (INT CONST nr, TEXT CONST dtype)	52-142
PROC enter outcode (INT CONST eumel code, ziel code)	52-156
PROC enter outcode (INT CONST eumel code, wartezeit, TEXT CONST sequenz)	52-178
PROC enter outcode (INT CONST eumelcode, TEXT CONST wert)	52-194
PROC enter incode (INT CONST elan code, TEXT CONST sequenz)	52-211
PROC cursor logic (INT CONST dist, TEXT CONST pre, mid, post)	52-225
PROC ansi cursor (TEXT CONST pre, mid, post)	52-231
PROC cursor logic (INT CONST dist, modus, TEXT CONST pre, mid, post)	52-237
PROC elbit cursor	52-247

PACKET configurator single :

PROC configurate	S53-220
PROC exec configuration	S53-447
PROC setup	S53-453

PACKET single user monitor :

PROC monitor	S43-34
PROC monitor (PROC init system)	S43-40
PROC set date	S43-121

PROC shutup	S43-182
PROC save system	S43-192
PROC collect garbage blocks	S43-202
PROC fixpoint	S43-208
PROC set clock (REAL CONST time)	S43-235

3. Übersicht über die exportierten Objekte alphabetisch geordnet:

'(M)' vor der Paketnummer heißt, daß dies Objekt nur im Multi-User vorhanden ist.
'(S)' vor der Paketnummer heißt, daß dies Objekt nur im Single-User vorhanden ist.
'(T)' vor der Paketnummer heißt, daß dies Objekt nur in einem System mit Textverarbeitung vorhanden ist.

Die Paketnummer ergibt sich aus der Reihenfolge, in der die Pakete im Multi-User mit Textverarbeitung insertiert wurden. Bitte beachten Sie, daß diese Reihenfolge nicht der Insertierungsreihenfolge im Single-User entspricht. Der Quellcode der insertierten Pakete ist in Teil 4 nach Paketnummern sortiert.

* (INT CONST times, TEXT CONST source) --> TEXT	3-109
** (INT CONST arg, exp) --> INT	8-126
** (REAL CONST a, INT CONST b) --> REAL	26-231
** (REAL CONST b, e) --> REAL	26-222
** (TEXT CONST p, INT CONST x) --> TEXT	15-58
+ (TEXT CONST left, right) --> TEXT	3-103
+ (THESAURUS CONST left, TEXT CONST right) --> THESAURUS	39-47
+ (THESAURUS CONST left, right) --> THESAURUS	39-32
- (TEXT CONST alphabet) --> TEXT	15-43
- (THESAURUS CONST left, TEXT CONST right) --> THESAURUS	39-72
- (THESAURUS CONST left, right) --> THESAURUS	39-57
/ (THESAURUS CONST left, right) --> THESAURUS	39-81
:= (DATASPACE VAR dest, DATASPACE CONST source)	5-23
:= (FILE VAR left, FILE CONST right)	16-160
:= (INITFLAG VAR flag, BOOL CONST flagtrue)	4-39
:= (TASK VAR dest, TASK CONST source)	S30-51
:= (THESAURUS VAR dest, THESAURUS CONST source)	13-103
= (TASK CONST left, right) --> BOOL	S30-57

A

abschnitt neu (INT CONST von satznr, bis satznr)	21-2228
ABS (INT CONST argument) --> INT	8-122
abs (INT CONST argument) --> INT	8-114
abs (REAL CONST value) --> REAL	10-329
ABS (REAL CONST value) --> REAL	10-338
aktueller editor --> INT	21-1279
aktueller editor --> INT	21-2158
ALIGN	5-21
alles neu	21-2261
ALL (TASK CONST from) --> THESAURUS	S49-649
ALL (TEXT CONST file name) --> THESAURUS	39-96
all --> THESAURUS	14-358
analyze command (TEXT CONST command list, command line, INT CONST permitted type, INT VAR command index, number of params, TEXT VAR param 1, param 2)	27-117
analyze command (TEXT CONST command list, INT CONST permitted type, INT VAR command index, number of params, TEXT VAR param 1, param 2)	27-106
AND (INT CONST left, right) --> INT	2-23
ansi cursor (TEXT CONST pre, mid, post)	52-231
any (INT CONST n) --> TEXT	15-64

4. Quellecode der insertierten Pakete

'(M)'	vor der Paketnummer heißt, daß dies Objekt nur im Multi – User vorhanden ist.
'(S)'	vor der Paketnummer heißt, daß dies Objekt nur im Single – User vorhanden ist.
'(T)'	vor der Paketnummer heißt, daß dies Objekt nur in einem System mit Textverarbeitung vorhanden ist.

Die Paketnummer ergibt sich aus der Reihenfolge, in der die Pakete im Multi – User mit Textverarbeitung insertiert wurden. Bitte beachten Sie, daß diese Reihenfolge nicht der Insertierungsreihenfolge im Single – User entspricht. Der Quellcode der insertierten Pakete ist in Teil 4 nach Paketnummern sortiert.

```
  1                                 | "run again impossible"
  2                                 | "recursive run"
  3                                 | "                             "
  4                                 | " Compiler Error : "
  5                                 |"             "
  6                                 |"                                   |"
  7                                 |" Fehler entdeckt  "
  8                                 |"Keine Fehler gefunden,  "
  9                                 |" "
 10                                 |"       *******  ENDE DER UEBERSETZUNG  *******"
 11                                 |"FEHLER bei >> "
 12                                 |" <<  "
 13                                 |"weiter bei "
 14                                 |"TEXTende (Anfuehrungszeichen) fehlt irgendwo"
 15                                 |"Kommentarende fehlt irgendwo"
 16                                 |"nach dem Hauptprogramm darf kein Paket folgen"
 17                                 |"ungueltiger Name fuer ein DEFINES-Objekt"
 18                                 |"':' fehlt"
 19                                 |"nach ENDPACKET folgt nicht der Paketname"
 20                                 |"ENDPACKET fehlt"
 21                                 |"CONST oder VAR fehlt"
 22                                 |"ungueltiger Name"
 23                                 |" ',' in Deklarationsliste fehlt"
 24                                 |"ist nicht der PROC Name"
 25                                 |"fehlerhaftes Ende des Hauptprogramms"
 26                                 |"ENDPROC fehlt"
 27                                 |"PROC/OP Schachtelung unzulaessig"
 28                                 |"OP darf kein Parameter sein"
 29                                 |"steht mehrfach im PACKET Interface"
 30                                 |" ist mehrfach deklariert"
 31                                 |"ist schon als Datenobjekt deklariert"
 32                                 |"ist schon als PROC/OP deklariert"
 33                                 |"')' nach Parameterliste erwartet"
 34                                 |"Standard-Schluesselwort kann nicht redefiniert werden"
 35                                 |"ungueltig als BOLD"
 36                                 |"'(' fehlt"
 37                                 |"CONST bzw VAR nicht bei Strukturfeldern"
 38                                 |"'=' fehlt"
 39                                 |"Schluesselwort wird im Paket schon andersartig verwandt"
 40                                 |"Datentyp fehlt"
 41                                 |"ungueltiger OP Name"
 42                                 |"OP muss monadisch oder dyadisch sein"
 43                                 |"ist nicht der OP Name"
 44                                 |"ENDOP fehlt"
 45                                 |"Name nach ENDPROC fehlt"
 46                                 |"Name nach ENDOP fehlt"
 47                                 |"';' fehlt"
 48                                 |"END END ist Unsinn"
 49                                 |"Dieses END... kenne ich nicht"
 50                                 |"ROW Groesse ist kein INT"
 51                                 |"ROW Groesse ist kein Denoter"
 52                                 |"Ein ROW muss mindestens ein Element haben"
 53                                 |"ROW Groesse fehlt"
 54                                 |"Parameter kann man nicht initialisieren"
 55                                 |"Konstanten muessen initialisiert werden"
 56                                 |"'::' verwenden"
 57                                 |"')' fehlt"
 58                                 |"Exponent fehlt"
 59                                 |"Undefinierter Typ"
 60                                 |"Rekursiv definierter Typ"
 61                                 |"Mehrfach definierter Selektor"
 62                                 |"Variable bzw. Abkuerzung in der Paket-Schnittstelle"
```

```
  63                         |"undefinierte ROW Groesse"
  64                         |"Typ Deklarationen nur im Paketrumpf"
  65                         |"CONST bzw. VAR ohne Zusammenhang"
  66                         |"ist nicht deklariert, steht aber in der Paket-Schnittstelle"
  67                         |"ist nicht deklariert"
  68                         |"unbekanntes Kommando"
  69                         |"THIS IS NO CORRECT EXTERNAL NUMBER."
  70                         |"Schluesselwort unzulaessig"
  71                         |"Name erwartet"
  72                         |"Denoter erwartet"
  73                         |"ENDPROC ohne Zusammenhang"
  74                         |"ENDOP ohne Zusammenhang"
  75                         |"Refinement ohne Zusammenhang"
  76                         |"Delimiter zwischen Paket-Refinement und Deklaration fehlt"
  77                         |"unzulaessiges Selektor-Symbol (kein Name)"
  78                         |"BOUND Schachtelungen unzulaessig"
  79                         |"BOUND-Objekte unzulaessig als Parameter"
  80                         |"Textende fehlt"
  81                         |"TEXT-Denoter zu lang"
  82                         |
  83                         |"Denoter-Wert wird fuer diese Maschine zu gross"
  84                         |"Compiler-Fehler, wenden Sie sich an Ihren Systemberater!"
  85                         |"ist ein zusammenhangloses Schluesselwort"
  86                         |"'::' nur fuer Initialisierungen, sonst ':='"
  87                         |"welches Objekt soll verlassen werden?"
  88                         |"du bist gar nicht innerhalb dieses Refinements"
  89                         |"nur die eigene PROC / OP kann verlassen werden"
  90                         |"THEN fehlt"
  91                         |"FI fehlt"
  92                         |"BOOL-Ausdruck erwartet"
  93                         |"ELSE-Teil ist notwendig, da ein Wert geliefert wird"
  94                         |"INT-Ausdruck erwartet"
  95                         |"OF fehlt"
  96                         |"Keine Typanpassung moeglich"
  97                         |"CASE-Label fehlt"
  98                         |"mindestens eine CASE-Anweisung geben"
  99                         |"CASE-Label ist zu gross (skipped)"
 100                         |"mehrfach definiertes CASE-Label"
 101                         |"ungueltiges Zeichen nach CASE-Label"
 102                         |"OTHERWISE-Teil fehlt"
 103                         |"END SELECT fehlt"
 104                         |"rekursiver Aufruf eines Refinements"
 105                         |" wird nicht benutzt"
 106                         |"';' oder Operator ('+','-',...) fehlt"
 107                         |"undefinierter monadischer Operator"
 108                         |"undefinierter dyadischer Operator"
 109                         |"Auf die Feinstruktur des Typs kann man nicht mehr zugreifen"
 110                         |"fuer diesen Typ nicht definierter Selektor"
 111                         |"INT,REAL,BOOL,TEXT koennen nicht selektiert werden"
 112                         |"bei ROWs nur Subscription"
 113                         |"nicht selektierbar"
 114                         |"unzulaessiger Index fuer Subscription"
 115                         |"'[' ohne Zusammenhang"
 116                         |"']' ohne Zusammenhang"
 117                         |"']' nach Subscription fehlt"
 118                         |"ungueltig zwischen Anweisungen"
 119                         |"nur die letzte Anweisung eines Abschnitts darf einen Wert liefern"
 120                         |"Der Paketrumpf kann keinen Wert liefern"
 121                         |"anstelle des letzten Symbols wurde ein Operand erwartet"
 122                         |"Der Schleifenrumpf darf keinen Wert liefern"
 123                         |"die Laufvariable muss eine INT VAR sein"
 124                         |"wird schon in einer aeusseren Schleife als Laufvariable benutzt"
```

```
|"FROM erwartet"
|"UPTO bzw DOWNTO fehlt"
|"REPEAT fehlt"
|"END REP fehlt"
|"die Konstante darf nicht veraendert werden"
|"in einer FOR-Schleife darf die Laufvariable nicht veraendert werden"
|"falscher Typ des Resultats"
|"ist CONST, es wird aber ein VAR Parameter verlangt"
|"unbekannte Prozedur"
|"Parameter-Prozedur liefert falsches Resultat"
|"Anzahl bzw. Typen der Parameter sind falsch"
|"unbekannte Parameter-Prozedur"
|"aktuelle Parameter-Prozedur hat CONST-, formale hat VAR-Parameter"
|"Kein Konstruktor moeglich, da die Feinstruktur hier unbekannt ist"
|"zu wenig Felder angegeben"
|"zu viele Felder angegeben"
|"unzulaessiger Trenner zwischen Feldern"
|"Feld hat falschen Typ"
|"falsche Element-Anzahl im ROW-Konstruktor"
|"Dieser Typ kann nicht noch mehr konkretisiert werden"
|"BOUND-Objekt zu gross"
|
|"Warnung in Zeile "
|"   Zeile "
|"in Zeile "
|" <----+---> "
|" TYPE undefiniert "
|" MODE undefiniert "
|"Parameter spezifiziert: "
|"Parameter Typ(en) sind: "
|" B Code, "
|" B Paketdaten generiert"
|"Operand: "
|"Operanden: "
|", "
|"erwartet "
|"gefunden "
|" "
|
|(* 001 *)    END
|(* 002 *)    ENDPACKET
|(* 003 *)    ENDOP
|(* 004 *)    ENDOPERATOR
|(* 005 *)    ENDPROC
|(* 006 *)    ENDPROCEDURE
```

```
*************************|(* 007 *)    PACKET
                         |(* 008 *)    OP
                         |(* 009 *)    OPERATOR
                         |(* 010 *)    PROC
                         |(* 011 *)    PROCEDURE
                         |(* 012 *)    FI
                         |(* 013 *)    ENDIF
                         |(* 014 *)    ENDREP
                         |(* 015 *)    ENDREPEAT
                         |(* 016 *)    PER
                         |(* 017 *)    ELIF
                         |(* 018 *)    ELSE
                         |(* 019 *)    UNTIL
                         |(* 020 *)    CASE
                         |(* 021 *)    OTHERWISE
                         |(* 022 *)    ENDSELECT
                         |(* 023 *)    INTERNAL
                         |(* 024 *)    DEFINES
                         |(* 025 *)    LET
                         |(* 026 *)    TYPE
                         |(* 027 *)    INT
                         |(* 028 *)    REAL
                         |(* 029 *)    DATASPACE
                         |(* 030 *)    TEXT
                         |(* 031 *)    BOOL
                         |(* 032 *)    BOUND
                         |(* 033 *)    ROW
                         |(* 034 *)    STRUCT
                         |(* 035 *)    CONST
                         |(* 036 *)    VAR
                         |(* 037 INIT CONTROL *)    INTERNAL
                         |(* 038 *)     CONCR
                         |(* 039 *)     REP
                         |(* 040 *)     REPEAT
                         |(* 041 *)     SELECT
                         |(* 042 *)     EXTERNAL
                         |(* 043 *)     IF
                         |(* 044 *)     THEN
                         |(* 045 *)     OF
                         |(* 046 *)     FOR
                         |(* 047 *)     FROM
                         |(* 048 *)     UPTO
                         |(* 049 *)     DOWNTO
                         |(* 050 *)     WHILE
                         |(* 051 *)     LEAVE
                         |(* 052 *)     WITH
                         |(* 053 *)     TRUE
                         |(* 054 *)     FALSE
                         |(* 055 *)     ::  SBL  :=  INCR  DECR
                         |(* 056 *)     + - * / DIV MOD
                         |              **
                         |              AND
                         |              CAND
                         |              OR
                         |              COR
                         |              NOT
                         |              =  <>  >  >= < <=
                         |(*040  *)     MAIN
                         |(*043*)  ENDOFFILE
                         |
a **************************|PACKET a :
                            |


out ........................|PROC out (TEXT CONST t) :
                            |  EXTERNAL 60
                            |ENDPROC out ;
                            |


outtext ....................|PROC out text (TEXT CONST t, INT CONST typ) :
                            |  INTERNAL 257 ;
                            |  IF typ = typ
                            |    THEN out (t)
                            |  FI
                            |ENDPROC out text ;
                            |


outline ....................|PROC out line (INT CONST typ) :
                            |  INTERNAL 258 ;
                            |  IF typ = typ
                            |    THEN out (""13""10"")
                            |  FI
                            |ENDPROC out line ;
                            |
                            |ENDPACKET a ;
                            |
                            |
                            |
                            |
                            |
                            |
                            |
                            |
                            |
                            |
                            |
                            |
                           |
bits **********************|PACKET bits DEFINES
                           |
                           |    AND ,
                           |    OR ,
                           |    XOR ,
                           |    bit ,
                           |    lowest reset ,
                           |    lowest set ,
                           |    reset bit ,
                           |    rotate ,
                           |    set bit :
                           |
                           |LET bits per int = 16 ;
                           |
                           |ROW bits per int INT VAR bit mask := ROW bits per int INT:
                           |
                           |         (1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,-327
                           |         67-1) ;
                           |


rotate ....................|PROC rotate (INT VAR bits, INT CONST number of bits) :
                           |  EXTERNAL 83
                           |ENDPROC rotate ;
                           |


AND .......................|INT OP AND (INT CONST left, right) :
                           |  EXTERNAL 124
                           |ENDOP AND ;
                           |


OR ........................|INT OP OR (INT CONST left, right) :
                           |  EXTERNAL 125
                           |ENDOP OR ;
                           |


XOR .......................|INT OP XOR (INT CONST left, right) :
                           |  EXTERNAL 121
                           |ENDOP XOR ;
                           |


bit .......................|BOOL PROC bit (INT CONST bits, bit no) :
                           |
                           |  (bits AND bit mask (bit no+1)) <> 0
                           |
                           |ENDPROC bit ;
                           |


setbit ....................|PROC set bit (INT VAR bits, INT CONST bit no) :
                           |
                           |  bits := bits OR bit mask (bit no+1)
                           |
                           |ENDPROC set bit ;
                           |
resetbit .................|PROC reset bit (INT VAR bits,INT CONST bit no) :
                          |
                          |  bits := bits XOR (bits AND bit mask (bit no+1))
                          |
                          |ENDPROC reset bit ;
                          |


lowestset ................|INT PROC lowest set (INT CONST bits) :
                          |
                          |  INT VAR mask index ;
                          |  FOR mask index FROM 1 UPTO 16 REP
                          |    IF (bits AND bit mask (mask index)) <> 0
                          |      THEN LEAVE lowest set WITH mask index - 1
                          |    FI
                          |  PER ;
                          |  -1
                          |
                          |ENDPROC lowest set ;
                          |


lowestreset ..............|INT PROC lowest reset (INT CONST bits) :
                          |
                          |  INT VAR mask index ;
                          |  FOR mask index FROM 1 UPTO bits per int REP
                          |    IF (bits AND bit mask (mask index)) = 0
                          |      THEN LEAVE lowest reset WITH mask index - 1
                          |    FI
                          |  PER ;
                          |  -1
                          |
                          |ENDPROC lowest reset ;
                          |
                          |ENDPACKET bits ;
```

```
                             |(* ------------------- VERSION 3     06.03.86 ------------------- *)
text ************************|PACKET text DEFINES
                             |
                             |            max text length ,
                             |            SUB ,
                             |            subtext ,
                             |            text ,
                             |            length , LENGTH ,
                             |            CAT ,
                             |            + ,
                             |            * ,
                             |            replace ,
                             |            change ,
                             |            change all ,
                             |            compress ,
                             |            pos ,
                             |            code ,
                             |            ISUB ,
                             |            RSUB ,
                             |            delete char ,
                             |            insert char ,
                             |            delete int ,
                             |            insert int ,
                             |            heap size ,
                             |            collect heap garbage ,
                             |            stranalyze ,
                             |            LEXEQUAL ,
                             |            LEXGREATER ,
                             |            LEXGREATEREQUAL :
                             |
                             |
                             |
                             |TEXT VAR text buffer , tail buffer ;
                             |
                             |INT CONST max text length := 32000 ;
                             |

SUB .........................|TEXT OP SUB (TEXT CONST text, INT CONST pos ) :
                             | EXTERNAL 48
                             |END OP SUB ;
                             |

subtext .....................|TEXT PROC subtext (TEXT CONST source, INT CONST from, to ):
                             | EXTERNAL 49
                             |ENDPROC subtext ;
                             |

subtext .....................|TEXT PROC subtext (TEXT CONST source, INT CONST from ) :
                             | EXTERNAL 50
                             |ENDPROC subtext ;
                             |

code ........................|INT PROC code (TEXT CONST text) :
                             | EXTERNAL 46
                             |END PROC code ;
                             |
code ......................|TEXT PROC code (INT CONST code) :
                           | EXTERNAL 47
                           |ENDPROC code ;
                           |


ISUB ......................|INT OP ISUB (TEXT CONST text, INT CONST index) :
                           | EXTERNAL 44
                           |ENDOP ISUB ;
                           |


replace ...................|PROC replace (TEXT VAR text, INT CONST index, value) :
                           | EXTERNAL 45
                           |ENDPROC replace ;
                           |


RSUB ......................|REAL OP RSUB (TEXT CONST text, INT CONST index) :
                           | EXTERNAL 100
                           |ENDOP RSUB ;
                           |


replace ...................|PROC replace (TEXT VAR text, INT CONST index, REAL CONST code) :
                           | EXTERNAL 101
                           |ENDPROC replace ;
                           |
                           |


replace ...................|PROC replace (TEXT VAR dest, INT CONST pos, TEXT CONST source) :
                           | EXTERNAL 51
                           |ENDPROC replace ;
                           |


text ......................|TEXT PROC text (TEXT CONST source, INT CONST length ) :
                           |
                           |  IF length < LENGTH source
                           |     THEN text buffer := subtext (source,1,length)
                           |     ELSE text buffer := source ;
                           |          mit blanks auffuellen
                           |  FI ;
                           |  text buffer .
                           |
                           |
   mitblanksauffuellen     |mit blanks auffuellen :
                           |  INT VAR i ;
                           |  FOR i FROM 1 UPTO length - LENGTH source REP
                           |    text buffer CAT " "
                           |  PER .
                           |
                           |ENDPROC text ;
                           |


text ......................|TEXT PROC text (TEXT CONST source, INT CONST length, from) :
                           |  text ( subtext (source, from) , length )
                           |ENDPROC text ;
                           |
CAT .......................|OP CAT (TEXT VAR right, TEXT CONST left ) :
                           | EXTERNAL 52
                           |ENDOP CAT ;
                           |

+ .........................|TEXT OP + (TEXT CONST left, right) :
                           | text buffer := left ;
                           | text buffer CAT right ;
                           | text buffer
                           |ENDOP + ;
                           |

* .........................|TEXT OP * (INT CONST times, TEXT CONST source ) :
                           |
                           |  text buffer := "" ;
                           |  INT VAR i ;
                           |  FOR i FROM 1 UPTO times REP
                           |    text buffer CAT source
                           |  PER ;
                           |  text buffer
                           |
                           |ENDOP * ;
                           |

length ....................|INT PROC length (TEXT CONST text ) :
                           | EXTERNAL 53
                           |ENDPROC length ;
                           |

LENGTH ....................|INT OP LENGTH (TEXT CONST text ) :
                           | EXTERNAL 53
                           |ENDOP LENGTH ;
                           |

pos .......................|INT PROC pos (TEXT CONST source, pattern) :
                           | EXTERNAL 54
                           |ENDPROC pos ;
                           |

pos .......................|INT PROC pos (TEXT CONST source, pattern, INT CONST from) :
                           | EXTERNAL 55
                           |ENDPROC pos ;
                           |

pos .......................|INT PROC pos (TEXT CONST source, pattern, INT CONST from, to) :
                           | EXTERNAL 56
                           |ENDPROC pos ;
                           |

pos .......................|INT PROC pos (TEXT CONST source, low, high, INT CONST from) :
                           | EXTERNAL 58
                           |ENDPROC pos ;
                           |
compress ................|TEXT PROC compress (TEXT CONST text) :
                         |
                         |  INT VAR begin, end ;
                         |
                         |  search first non blank ;
                         |  search last non blank ;
                         |  text buffer := subtext (text, begin, end) ;
                         |  text buffer .
                         |
                         |
   searchfirstnonblank   |search first non blank :
                         |  begin := 1 ;
                         |  WHILE (text SUB begin) = " " REP
                         |    begin INCR 1
                         |  PER .
                         |
                         |
   searchlastnonblank    |search last non blank :
                         |  end := LENGTH text ;
                         |  WHILE (text SUB end) = " " REP
                         |    end DECR 1
                         |  PER .
                         |
                         |ENDPROC compress ;
                         |


change ..................|PROC change (TEXT VAR destination, INT CONST from, to, TEXT CONST
                         |     new) :
                         |
                         |  IF LENGTH new = to - from + 1 AND to <= LENGTH destination
                         |    THEN replace (destination, from, new)
                         |    ELSE change via buffer
                         |  FI .
                         |
                         |
   changeviabuffer       |change via buffer :
                         |  text buffer := subtext (destination, 1, from-1) ;
                         |  text buffer CAT new ;
                         |  tail buffer := subtext (destination, to + 1) ;
                         |  text buffer CAT tail buffer ;
                         |  destination := text buffer
                         |
                         |ENDPROC change ;
                         |


change ..................|PROC change (TEXT VAR destination, TEXT CONST old, new) :
                         |
                         |  INT CONST position := pos (destination, old) ;
                         |  IF position > 0
                         |    THEN change (destination, position, position + LENGTH old -1,
                         |         new)
                         |  FI
                         |
                         |ENDPROC change ;
                         |


changeall ...............|PROC change all (TEXT VAR destination, TEXT CONST old, new) :
                         |
                         |  INT VAR position := pos (destination, old) ;
```

```
                            |  IF LENGTH old = LENGTH new
                            |    THEN change by replace
                            |    ELSE change by change
                            |  FI .
                            |
                            |
      changebyreplace       |change by replace :
                            |  WHILE position > 0 REP
                            |    replace (destination, position, new) ;
                            |    position := pos (destination, old, position + LENGTH new)
                            |  PER .
                            |
                            |
      changebychange        |change by change :
                            |  WHILE position > 0 REP
                            |    change (destination, position, position + LENGTH old - 1 , new)
                            |    position := pos (destination, old, position + LENGTH new)
                            |  PER .
                            |
                            |ENDPROC change all ;
                            |


   deletechar ..............|PROC delete char (TEXT VAR string, INT CONST delete pos) :
                            |
                            |  IF delete pos > 0
                            |    THEN tail buffer := subtext (string, delete pos + 1) ;
                            |         string := subtext (string, 1, delete pos - 1) ;
                            |         string CAT tail buffer
                            |  FI
                            |
                            |END PROC delete char ;
                            |


   insertchar ..............|PROC insert char (TEXT VAR string, TEXT CONST char,
                            |                  INT CONST insert pos) :
                            |
                            |  IF insert pos > 0 AND insert pos <= LENGTH string + 1
                            |    THEN tail buffer := subtext (string, insert pos) ;
                            |         string := subtext (string, 1, insert pos - 1) ;
                            |         string CAT char ;
                            |         string CAT tail buffer
                            |  FI
                            |
                            |END PROC insert char ;
                            |


   heapsize ................|INT PROC heap size :
                            |  EXTERNAL 93
                            |ENDPROC heap size ;
                            |


   collectheapgarbage ......|PROC collect heap garbage :
                            |  EXTERNAL 94
                            |ENDPROC collect heap garbage ;
                            |
stranalyze ..............|PROC stranalyze (ROW 256 INT CONST table, INT VAR sum, INT CONST max
  +                      |     sum,
                         |               TEXT CONST string, INT VAR index, INT CONST to,
                         |               INT VAR exit code) :
                         |  EXTERNAL 57
                         |ENDPROC stranalyze ;
                         |
                         |(********************************************************************)
                         |(* lexikographische Vergleiche                                      *)
                         |(* Nach DIN 5007, Abschnitt 1 und Abschnitt 3.2 (Bindestrich)      *)
                         |(* Autor: Rainer Hahn, Jochen Liedtke                              *)
                         |(* Stand: 1.7.4 (Jan. 1985)                                        *)
                         |(********************************************************************)
                         |LET first umlaut = ""214"" ,
                         |    umlauts      = ""214""215""216""217""218""219""251"" ;
                         |
                         |
                         |TEXT VAR left letter, right letter;
                         |


LEXEQUAL ................|BOOL OP LEXEQUAL (TEXT CONST left, right) :
                         |
                         |  compare (left, right) ;
                         |  left letter = right letter
                         |
                         |ENDOP LEXEQUAL ;
                         |


LEXGREATER ..............|BOOL OP LEXGREATER (TEXT CONST left, right) :
                         |
                         |  compare (left, right) ;
                         |  left letter > right letter
                         |
                         |ENDOP LEXGREATER ;
                         |


LEXGREATEREQUAL .........|BOOL OP LEXGREATEREQUAL (TEXT CONST left, right) :
                         |
                         |  compare (left, right) ;
                         |  left letter >= right letter
                         |
                         |ENDOP LEXGREATEREQUAL ;
                         |


compare .................|PROC compare (TEXT CONST left, right) :
                         |
                         |  to begin of lex relevant text ;
                         |  REP
                         |    get left letter ;
                         |    get right letter
                         |  UNTIL NOT letter match OR both ended PER .
                         |
                         |
   tobeginoflexrelevantte |to begin of lex relevant text :
                         |  INT VAR
                         |  left pos  := pos (left, ""65"",""254"", 1) ,
                         |  right pos := pos (right,""65"",""254"", 1) ;
                         |  IF left pos = 0
                         |    THEN left pos := LENGTH left + 1
                         |  FI ;
                         |  IF right pos = 0
                         |    THEN right pos := LENGTH right + 1
                         |  FI .
                         |
                         |
getleftletter            |get left letter :
                         |  left letter := left SUB left pos ;
                         |  left pos INCR 1 .
                         |
                         |
getrightletter           |get right letter :
                         |  right letter := right SUB right pos ;
                         |  right pos INCR 1 .
                         |
                         |
lettermatch              |letter match :
                         |  IF left letter = right letter
                         |    THEN TRUE
                         |    ELSE dine (left, left letter, left pos) ;
                         |         dine (right, right letter, right pos) ;
                         |         IF exactly one letter is double letter
                         |           THEN expand other letter
                         |         FI ;
                         |         left letter = right letter
                         |  FI .
                         |
                         |
exactlyoneletterisdoub   |exactly one letter is double letter :
                         |  LENGTH left letter <> LENGTH right letter.
                         |
                         |
expandotherletter        |expand other letter :
                         |  IF LENGTH left letter = 1
                         |    THEN left letter CAT (left SUB left pos) ;
                         |         left pos INCR 1
                         |    ELSE right letter CAT (right SUB right pos) ;
                         |         right pos INCR 1
                         |  FI .
                         |
                         |
bothended                |both ended : left letter = "" .
                         |
                         |ENDPROC compare ;
                         |


dine ....................|PROC dine (TEXT CONST string, TEXT VAR char, INT VAR string pos)
                         |
                         |  skip non letter chars ;
                         |  IF is capital letter
                         |    THEN translate to small letter
                         |  ELIF char >= first umlaut
                         |    THEN translate umlaut
                         |  FI .
                         |
                         |
skipnonletterchars       |skip non letter chars :
                         |  WHILE NOT (is letter OR end of string) REP
                         |    char := string SUB string pos ;
                         |    string pos INCR 1
```

```
                              |  PER .
                              |
                              |
    translatetosmallletter    |translate to small letter :
                              |  char := code (code (char) + 32) .
                              |
                              |
    translateumlaut           |translate umlaut :
                              |  SELECT pos (umlauts, char) OF
                              |    CASE 1,4 : char := "ae"
                              |    CASE 2,5 : char := "oe"
                              |    CASE 3,6 : char := "ue"
                              |    CASE 7   : char := "ss"
                              |  ENDSELECT .
                              |
                              |
    iscapitalletter           |is capital letter :
                              |  INT VAR char code := code (char) ;
                              |  65 <= char code AND char code <= 90 .
                              |
                              |
    isletter                  |is letter :
                              |  char code := code (char) OR 32 ;
                              |  (97 <= char code AND char code <= 122)  OR  char code >= 128 .
                              |
                              |
    endofstring               |end of string : char = "" .
                              |
                              |ENDPROC dine ;
                              |


CAT ..........................|OP CAT (TEXT VAR result, INT CONST number) :
                              |    result CAT "  ";
                              |    replace (result, LENGTH result DIV 2, number);
                              |END OP CAT;
                              |


insertint ....................|PROC insert int (TEXT VAR result, INT CONST insert pos, number)
                              |    INT VAR pos := insert pos * 2 - 1;
                              |    change  (result, pos, pos - 1, "  ");
                              |    replace (result, insert pos, number);
                              |END PROC insert int;
                              |


deleteint ....................|PROC delete int (TEXT VAR result, INT CONST delete pos) :
                              |    INT VAR pos := delete pos * 2;
                              |    change  (result, pos - 1, pos, "")
                              |END PROC delete int;
                              |
                              |ENDPACKET text ;
```

```
                            |
pcbandinitcontrol ********* |PACKET pcb and init control DEFINES          (* Autor: J.Liedtke *)
                            |                                             (* Stand: 25.08.84  *)
                            |  session ,
                            |  pcb ,
                            |  set line nr ,
                            |  clock ,
                            |  INITFLAG ,
                            |  := ,
                            |  initialized ,
                            |  storage ,
                            |  id ,
                            |  ke :
                            |
                            |
                            |LET line number field     = 1 ,
                            |    myself id field       = 9 ;
                            |
                            |TYPE INITFLAG = INT ;
                            |
                            |


session ....................|INT PROC session :
                            |  EXTERNAL 126
                            |ENDPROC session ;
                            |


pcb ........................|INT PROC pcb (INT CONST field) :
                            |  EXTERNAL 80
                            |ENDPROC pcb ;
                            |


writepcb ...................|PROC write pcb (INT CONST task nr, field, value) :
                            |  EXTERNAL 105
                            |ENDPROC write pcb ;
                            |


setlinenr ..................|PROC set line nr (INT CONST value) :
                            |  write pcb (pcb (myself id field), line number field, value)
                            |ENDPROC set line nr ;
                            |
                            |


:= .........................|OP := (INITFLAG VAR flag, BOOL CONST flagtrue) :
                            |
                            |  IF flagtrue
                            |    THEN CONCR (flag) := myself no
                            |    ELSE CONCR (flag) := 0
                            |  FI .
                            |
                            |
   myselfno                 |myself no :   pcb (myself id field) AND 255 .
                            |
                            |ENDOP := ;
                            |
initialized .............|BOOL PROC initialized (INITFLAG VAR flag) :
                         |
                         |  IF CONCR (flag) = myself no
                         |    THEN TRUE
                         |    ELSE CONCR (flag) := myself no ;
                         |         FALSE
                         |  FI .
                         |
                         |
   myselfno              |myself no :   pcb (myself id field) AND 255 .
                         |
                         |ENDPROC initialized ;
                         |


clock ...................|REAL PROC clock (INT CONST nr) :
                         |  EXTERNAL 102
                         |ENDPROC clock ;
                         |


storage .................|PROC storage (INT VAR size, used) :
                         |  EXTERNAL 89
                         |ENDPROC storage ;
                         |


id ......................|INT PROC id (INT CONST no) :
                         |  EXTERNAL 129
                         |ENDPROC id ;
                         |


ke ......................|PROC ke :
                         |  EXTERNAL 6
                         |ENDPROC ke ;
                         |
                         |ENDPACKET pcb and init control ;
```

```
                                |(* ------------------- VERSION 3     22.04.86 -------------------
dataspace ******************    |PACKET dataspace DEFINES
                                |
                                |    := ,
                                |    nilspace ,
                                |    forget ,
                                |    type ,
                                |    heap size ,
                                |    storage ,
                                |    ds pages ,
                                |    next ds page ,
                                |    blockout ,
                                |    blockin ,
                                |    ALIGN :
                                |
                                |
                                |LET myself id field   = 9 ,
                                |    lowest ds number  = 4 ,
                                |    highest ds number = 255 ;
                                |
                                |TYPE ALIGN = ROW 252 INT ;
                                |

:= .............................|OP := (DATASPACE VAR dest, DATASPACE CONST source ) :
                                |  EXTERNAL 70
                                |ENDOP := ;
                                |

nilspace .......................|DATASPACE PROC nilspace :
                                |  EXTERNAL 69
                                |ENDPROC nilspace ;
                                |

forget .........................|PROC forget (DATASPACE CONST dataspace ) :
                                |  EXTERNAL 71
                                |ENDPROC forget ;
                                |

type ...........................|PROC type (DATASPACE CONST ds, INT CONST type) :
                                |  EXTERNAL 72
                                |ENDPROC type ;
                                |

type ...........................|INT PROC type (DATASPACE CONST ds) :
                                |  EXTERNAL 73
                                |ENDPROC type ;
                                |

heapsize .......................|INT PROC heap size (DATASPACE CONST ds) :
                                |  EXTERNAL 74
                                |ENDPROC heap size ;
                                |
 47  storage ..................|INT PROC storage (DATASPACE CONST ds) :
 48                            |  (ds pages (ds) + 1) DIV 2
 49                            |ENDPROC storage ;
 50                            |


 51  dspages ..................|INT PROC ds pages (DATASPACE CONST ds) :
 52                            |  pages (ds, pcb (myself id field))
 53                            |ENDPROC ds pages ;
 54                            |


 55  pages ....................|INT PROC pages (DATASPACE CONST ds, INT CONST task nr) :
 56                            |  EXTERNAL 88
 57                            |ENDPROC pages ;
 58                            |


 59  nextdspage ...............|INT PROC next ds page (DATASPACE CONST ds, INT CONST page nr) :
 60                            |  EXTERNAL 87
 61                            |ENDPROC next ds page ;
 62                            |


 63  blockout .................|PROC blockout (DATASPACE CONST ds, INT CONST page nr, code1, code2,
 64                            |                   INT VAR return code) :
 65                            |  EXTERNAL 85
 66                            |ENDPROC blockout ;
 67                            |


 68  blockin ..................|PROC blockin (DATASPACE VAR ds, INT CONST page nr, code1, code2,
 69                            |                  INT VAR return code) :
 70                            |  EXTERNAL 86
 71                            |ENDPROC blockin ;
 72                            |
 73                            |ENDPACKET dataspace ;
```

```
                                     |
basictransput ************|PACKET basic transput DEFINES
                                     |                                  out ,
                                     |                                  outsubtext ,
                                     |                                  outtext ,
                                     |                                  TIMESOUT ,
                                     |                                  cout ,
                                     |                                  display ,
                                     |                                  inchar ,
                                     |                                  incharety ,
                                     |                                  cat input ,
                                     |                                  pause ,
                                     |                                  cursor ,
                                     |                                  get cursor ,
                                     |                                  channel ,
                                     |                                  online ,
                                     |                                  control ,
                                     |                                  blockout ,
                                     |                                  blockin :
                                     |
                                     |
                                     |
                                     |LET channel field  = 4 ,
                                     |    blank times 64 =
                                     |    "
                                     |         " ;
                                     |
                                     |LET BLOCKIO = STRUCT (ALIGN page align, ROW 256 INT buffer) ,
                                     |              buffer page = 2 ;
                                     |
                                     |BOUND BLOCKIO VAR block io ;
                                     |DATASPACE VAR block io ds ;
                                     |INITFLAG VAR this packet := FALSE ;
                                     |
                                     |


out ......................|PROC out (TEXT CONST text ) :
                                     | EXTERNAL 60
                                     |ENDPROC out ;
                                     |


outsubtext ..............|PROC outsubtext ( TEXT CONST source, INT CONST from ) :
                                     | EXTERNAL 62
                                     |END PROC outsubtext;
                                     |


outsubtext ..............|PROC outsubtext (TEXT CONST source, INT CONST from, to) :
                                     | EXTERNAL 63
                                     |END PROC outsubtext;
                                     |


outtext ..................|PROC outtext ( TEXT CONST source, INT CONST from, to ) :
                                     |  out subtext (source, from, to) ;
                                     |  INT VAR trailing ;
                                     |  IF from <= LENGTH source
                                     |    THEN trailing := to - LENGTH source
                                     |    ELSE trailing := to + 1 - from
                                     |  FI ;
```

```
                          |  IF trailing > 0
                          |    THEN trailing TIMESOUT " "
                          |  FI
                          |ENDPROC outtext ;
                          |

TIMESOUT .................|OP TIMESOUT (INT CONST times, TEXT CONST text) :
                          |
                          |  IF text = " "
                          |    THEN fast timesout blank
                          |    ELSE timesout
                          |  FI .
                          |
                          |
   fasttimesoutblank      |fast timesout blank :
                          |  INT VAR i := 0 ;
                          |  WHILE i + 64 < times REP
                          |    out (blank times 64) ;
                          |    i INCR 64
                          |  PER ;
                          |  outsubtext (blank times 64, 1, times - i) .
                          |
                          |
   timesout               |timesout :
                          |  FOR i FROM 1 UPTO times REP
                          |    out(text)
                          |  ENDREP .
                          |
                          |ENDOP TIMESOUT ;
                          |

display ..................|PROC display (TEXT CONST text) :
                          |  IF online
                          |    THEN out (text)
                          |  FI
                          |ENDPROC display ;
                          |

inchar ...................|PROC inchar (TEXT VAR character ) :
                          | EXTERNAL 64
                          |ENDPROC inchar ;
                          |

incharety ................|TEXT PROC incharety :
                          | EXTERNAL 65
                          |END PROC incharety ;
                          |

incharety ................|TEXT PROC incharety (INT CONST time limit) :
                          | internal pause (time limit) ;
                          | incharety
                          |ENDPROC incharety ;
                          |
pause ....................|PROC pause (INT CONST time limit) :
                          | internal pause (time limit) ;
                          | TEXT CONST dummy := incharety
                          |ENDPROC pause ;
                          |


pause ....................|PROC pause :
                          | TEXT VAR dummy; inchar (dummy)
                          |ENDPROC pause ;
                          |


internalpause ............|PROC internal pause (INT CONST time limit) :
                          | EXTERNAL 66
                          |ENDPROC internal pause ;
                          |


catinput .................|PROC cat input (TEXT VAR t, esc char) :
                          | EXTERNAL 68
                          |ENDPROC cat input ;
                          |
                          |


cursor ...................|PROC cursor (INT CONST x, y) :
                          | out (""6"") ;
                          | out (code(y-1)) ;
                          | out (code(x-1)) ;
                          |ENDPROC cursor ;
                          |


getcursor ................|PROC get cursor (INT VAR x, y) :
                          |  EXTERNAL 67
                          |ENDPROC get cursor ;
                          |


cout .....................|PROC cout (INT CONST number) :
                          |  EXTERNAL 61
                          |ENDPROC cout ;
                          |
                          |


channel ..................|INT PROC channel :
                          |  pcb (channel field)
                          |ENDPROC channel ;
                          |


online ...................|BOOL PROC online :
                          |  pcb (channel field) <> 0
                          |ENDPROC online ;
                          |
                          |
control ..................|PROC control (INT CONST code1, code2, code3, INT VAR return code)
                          |  EXTERNAL 84
                          |ENDPROC control ;
                          |


blockout .................|PROC blockout (ROW 256 INT CONST block, INT CONST code1, code2,
                          |                INT VAR return code) :
                          |
                          |  access block io ds ;
                          |  block io.buffer := block ;
                          |  blockout (block io ds, buffer page, code1, code2, return code)
                          |
                          |
   accessblockiods        |access block io ds :
                          |  IF NOT initialized (this packet)
                          |    THEN block io ds := nilspace
                          |  FI ;
                          |  block io := block io ds .
                          |
                          |ENDPROC blockout ;
                          |


blockin ..................|PROC blockin (ROW 256 INT VAR block, INT CONST code1, code2,
                          |               INT VAR return code) :
                          |
                          |  access block io ds ;
                          |  blockin (block io ds, buffer page, code1, code2, return code) ;
                          |  block := block io.buffer .
                          |
                          |
   accessblockiods        |access block io ds :
                          |  IF NOT initialized (this packet)
                          |    THEN block io ds := nilspace
                          |  FI ;
                          |  block io := block io ds .
                          |
                          |ENDPROC blockin ;
                          |
                          |ENDPACKET basic transput ;
                          |
                          |
                          |
bool *********************|PACKET bool DEFINES  XOR, true, false :
                          |
                          |BOOL CONST true := TRUE ,
                          |           false:= FALSE ;
                          |

XOR ......................|BOOL OP XOR (BOOL CONST left, right) :
                          |
                          |  IF left THEN NOT right
                          |          ELSE right
                          |  FI
                          |
                          |ENDOP XOR ;
                          |
                          |ENDPACKET bool ;
                          |(* -------------------  STAND :       23.10.85
                          |     -------------------*)
integer ******************|PACKET integer DEFINES text, int, MOD,
                          |                      sign, SIGN, abs, ABS, **, min, max, minint,
                          |                           maxint,
                          |                      random, initialize random ,
                          |                      last conversion ok, set conversion :
                          |


minint ...................|INT PROC minint : -32767 - 1 ENDPROC minint ;
                          |


maxint ...................|INT PROC maxint : 32767 ENDPROC maxint ;
                          |
                          |


text .....................|TEXT PROC text (INT CONST number) :
                          |
                          |  IF   number =  minint THEN "-32768"
                          |  ELIF number <  0      THEN "-" + text(-number)
                          |  ELIF number <= 9      THEN code (number + 48)
                          |                        ELSE text (number DIV 10) + digit
                          |  FI .
                          |
                          |
   digit                  |digit :
                          |  code ( number MOD 10 + 48 ) .
                          |
                          |ENDPROC text ;
                          |


text .....................|TEXT PROC text (INT CONST number, length) :
                          |
                          |  TEXT VAR result := text (number) ;
                          |  INT CONST number length := LENGTH result ;
                          |  IF number length < length
                          |    THEN (length - number length) * " " + result
                          |  ELIF number length > length
                          |    THEN length * "*"
                          |  ELSE   result
                          |  FI
                          |
                          |ENDPROC text ;
                          |


int ......................|INT PROC int (TEXT CONST number) :
                          |
                          |  skip blanks and sign ;
                          |  get value ;
                          |  result .
                          |
                          |
   skipblanksandsign      |skip blanks and sign :
                          |  BOOL VAR number is positive ;
                          |  INT VAR pos := 1 ;
                          |  skip blanks ;
                          |  IF (number SUB pos) = "-"
```

```
                              |   THEN number is positive := FALSE ;
                              |        pos INCR 1
                              | ELIF (number SUB pos) = "+"
                              |   THEN number is positive := TRUE ;
                              |        pos INCR 1
                              | ELSE   number is positive := TRUE
                              | FI .
                              |
                              |
   getvalue                   |get value :
                              | INT VAR value ;
                              | get first digit ;
                              | WHILE is digit REP
                              |   value := value * 10 + digit ;
                              |   pos INCR 1
                              | PER ;
                              | set conversion ok result .
                              |
                              |
   getfirstdigit              |get first digit :
                              | IF is digit
                              |   THEN value := digit ;
                              |        pos INCR 1
                              |   ELSE set conversion (FALSE) ;
                              |        LEAVE int WITH 0
                              | FI .
                              |
                              |
   isdigit                    |is digit : 0 <= digit AND digit <= 9 .
                              |
                              |
   digit                      |digit : code (number SUB pos) - 48 .
                              |
                              |
   result                     |result :
                              | IF number is positive
                              |   THEN  value
                              |   ELSE - value
                              | FI .
                              |
                              |
   setconversionokresult      |set conversion ok result :
                              | skip blanks ;
                              | conversion ok := (pos > LENGTH number) .
                              |
                              |
   skipblanks                 |skip blanks :
                              | WHILE (number SUB pos) = " " REP
                              |   pos INCR 1
                              | PER .
                              |
                              |ENDPROC int ;
                              |


MOD ..........................|INT OP MOD (INT CONST left, right) :
                              |
                              | EXTERNAL 43
                              |
                              |ENDOP MOD ;
                              |
sign ......................|INT PROC sign (INT CONST argument) :
                           |
                           |  IF argument < 0 THEN -1
                           |  ELIF argument > 0 THEN 1
                           |  ELSE 0
                           |  FI
                           |
                           |ENDPROC sign ;
                           |


SIGN ......................|INT OP SIGN (INT CONST argument) :
                           |  sign (argument)
                           |ENDOP SIGN ;
                           |


abs .......................|INT PROC abs (INT CONST argument) :
                           |
                           |  IF argument > 0 THEN argument
                           |  ELSE - argument
                           |  FI
                           |
                           |ENDPROC abs ;
                           |


ABS .......................|INT OP ABS (INT CONST argument) :
                           |  abs (argument)
                           |ENDOP ABS ;
                           |


** ........................|INT OP ** (INT CONST arg, exp) :
                           |
                           |  INT VAR x := arg , z := 1 ,
                           |          counter := exp ;
                           |
                           |  IF exp = 0
                           |    THEN LEAVE ** WITH 1
                           |  ELIF exp < 0
                           |    THEN LEAVE ** WITH 1 DIV arg
                           |  FI ;
                           |
                           |  WHILE counter >= 2 REP
                           |    calculate new x and z ;
                           |    counter := counter DIV 2 ;
                           |  ENDREP ;
                           |  z * x .
                           |
                           |
   calculatenewxandz       |calculate new x and z :
                           |  IF counter is not even
                           |    THEN z := z * x
                           |  FI ;
                           |  x := x * x .
                           |
                           |
   counterisnoteven        |counter is not even :
                           |  counter MOD 2 = 1 .
                           |
                           |ENDOP ** ;
```

```
                          |

min ......................|INT PROC min (INT CONST first, second) :
                          |
                          |  IF first < second THEN first ELSE second FI
                          |
                          |ENDPROC min ;
                          |

max ......................|INT PROC max (INT CONST first, second) :
                          |
                          |  IF first > second THEN first ELSE second FI
                          |
                          |ENDPROC max ;
                          |
                          |
                          |
                          |BOOL VAR conversion ok := TRUE ;
                          |

lastconversionok .........|BOOL PROC last conversion ok :
                          |  conversion ok
                          |ENDPROC last conversion ok ;
                          |

setconversion ............|PROC set conversion (BOOL CONST success) :
                          |  conversion ok := success
                          |ENDPROC set conversion ;
                          |
                          |
                          |
                          |(*******************************************************************)
                          |(*                                                                 *)
                          |(*                                          Autor: A. Flammenkamp  *)
                          |(*      RANDOM GENERATOR                                           *)
                          |(*                                                                 *)
                          |(*                      x    :=  4095 * x  MOD (4095*4096+4093) *)
                          |(*                       n+1              n                       *)
                          |(*                                                                 *)
                          |(*                      Periode: 2**24-4  >  16.0e6               *)
                          |(*                                                                 *)
                          |(*      Beachte:  x = 4096 * x1 + x0,  0 <= x0,x1 < 4096          *)
                          |(*                                                                 *)
                          |(*******************************************************************)
                          |
                          |
                          |INT VAR high := 1, low := 0 ;
                          |

initializerandom .........|PROC initialize random (INT CONST start) :
                          |
                          |  low := start MOD 4096 ;
                          |  IF start < 0
                          |    THEN high := 256 + 16 + start DIV 4096 ;
                          |         IF low <> 0 THEN high DECR 1 FI
                          |    ELSE high := 256 + start DIV 4096
                          |  FI
                          |
                          |ENDPROC initialize random ;
                          |

random ...................|INT PROC random (INT CONST lower bound, upper bound) :
                          |
                          |  compute new random value ;
                          |  normalize high ;
                          |  normalize low ;
                          |  map into interval .
                          |
                          |
   computenewrandomvalue  |compute new random value :
                          |  (*  (high,low) := (low-high , 3*high-low) *)
                          |  high := low - high ;
                          |  low INCR low - 3 * high ,
                          |
                          |
   normalizehigh          |normalize high :
                          |  IF high < 0
                          |    THEN high INCR 4096 ; low DECR 3
                          |  FI .
                          |
                          |
   normalizelow           |normalize low :
                          |  (*  high INCR low DIV 4096 ;
                          |      low := low MOD 4096
                          |  *)
                          |  IF low >= 4096 THEN low overflow
                          |  ELIF low < 0   THEN low underflow
                          |  FI .
                          |
                          |
   lowoverflow            |low overflow :
                          |  IF low >= 8192
                          |    THEN low DECR 8192 ; high INCR 2
                          |    ELSE low DECR 4096 ; high INCR 1 ; post normalization
                          |  FI .
                          |
                          |
   postnormalization      |post normalization :
                          |  (*  IF (high,low) >= (4095,4093)
                          |        THEN (high,low) DECR (4095,4093)
                          |      FI
                          |  *)
                          |  IF high >= 4095
                          |    THEN IF    low >= 4093 THEN high DECR 4095  ; low DECR 4093
                          |         ELIF high  = 4096 THEN high := 0       ; low INCR 3
                          |         FI
                          |  FI .
                          |
                          |
   lowunderflow           |low underflow :
                          |  low INCR 4096 ; high DECR 1 .
                          |
                          |
   mapintointerval        |map into interval :
                          |  INT VAR number := high MOD 16 - 8 ;
                          |  number INCR 4095 * number + low ;
                          |  IF lower bound <= upper bound
                          |    THEN lower bound + number MOD (upper bound - lower bound + 1)
```

```
    ELSE upper bound + number MOD (lower bound - upper bound + 1)
  FI .

ENDPROC random ;


ENDPACKET integer ;
```

```
                            |
errorhandling ************* |PACKET error handling DEFINES
                            |
                            |    enable stop ,
                            |    disable stop ,
                            |    is error ,
                            |    clear error ,
                            |    errormessage ,
                            |    error code ,
                            |    error line ,
                            |    put error ,
                            |    errorstop ,
                            |    stop :
                            |
                            |
                            |LET cr lf            = ""13""10"" ,
                            |    line nr field    = 1 ,
                            |    error line field = 2 ,
                            |    error code field = 3 ,
                            |    syntax error code= 100 ,
                            |
                            |    error pre        = ""7""13""10""5"FEHLER : " ;
                            |
                            |
                            |TEXT VAR errortext := "" ;
                            |
                            |


enablestop .................|PROC enable stop :
                            |  EXTERNAL 75
                            |ENDPROC enable stop ;
                            |


disablestop ................|PROC disable stop :
                            |  EXTERNAL 76
                            |ENDPROC disable stop ;
                            |


seterrorstop ...............|PROC set error stop (INT CONST code) :
                            |  EXTERNAL 77
                            |ENDPROC set error stop ;
                            |


iserror ....................|BOOL PROC is error :
                            |  EXTERNAL 78
                            |ENDPROC is error ;
                            |


clearerror .................|PROC clear error :
                            |  EXTERNAL 79
                            |ENDPROC clear error ;
                            |


selecterrormessage .........|PROC select error message :
                            |
                            |  SELECT error code OF
                            |    CASE 1 : error text := "'halt' vom Terminal"
                            |    CASE 2 : error text := "Stack-Ueberlauf"
                            |    CASE 3 : error text := "Heap-Ueberlauf"
                            |    CASE 4 : error text := "INT-Ueberlauf"
                            |    CASE 5 : error text := "DIV durch 0"
                            |    CASE 6 : error text := "REAL-Ueberlauf"
                            |    CASE 7 : error text := "TEXT-Ueberlauf"
                            |    CASE 8 : error text := "zu viele DATASPACEs"
                            |    CASE 9 : error text := "Ueberlauf bei Subskription"
                            |    CASE 10: error text := "Unterlauf bei Subskription"
                            |    CASE 11: error text := "falscher DATASPACE-Zugriff"
                            |    CASE 12: error text := "INT nicht initialisiert"
                            |    CASE 13: error text := "REAL nicht initialisiert"
                            |    CASE 14: error text := "TEXT nicht initialisiert"
                            |    CASE 15: error text := "nicht implementiert"
                            |    CASE 16: error text := "Block unlesbar"
                            |    CASE 17: error text := "Codefehler"
                            |  END SELECT
                            |
                            |ENDPROC select error message ;
                            |


errormessage ...............|TEXT PROC error message :
                            |
                            |  select error message ;
                            |  error text
                            |
                            |ENDPROC error message ;
                            |


errorcode ..................|INT PROC error code :
                            |
                            |  pcb (error code field)
                            |
                            |ENDPROC error code ;
                            |


errorline ..................|INT PROC error line :
                            |
                            |  IF is error
                            |    THEN pcb (error line field)
                            |    ELSE 0
                            |  FI
                            |
                            |ENDPROC error line ;
                            |


syntaxerror ................|PROC syntax error (TEXT CONST message) :
                            |
                            |  INTERNAL 259 ;
                            |  errorstop (syntax error code, message) .
                            |
                            |ENDPROC syntax error ;
                            |
errorstop ................|PROC errorstop (TEXT CONST message) :
                          |
                          |  errorstop (0, message) ;
                          |
                          |ENDPROC errorstop ;
                          |


errorstop ................|PROC errorstop (INT CONST code, TEXT CONST message) :
                          |
                          |  IF NOT is error
                          |    THEN error text := message ;
                          |         set error stop (code)
                          |  FI
                          |
                          |ENDPROC errorstop ;
                          |


puterror .................|PROC put error :
                          |
                          |  IF is error
                          |    THEN select error message ;
                          |         IF error text <> ""
                          |           THEN put error message
                          |         FI
                          |  FI .
                          |
                          |
   puterrormessage        |put error message :
                          |  out (error pre) ;
                          |  out (error text) ;
                          |  IF error line > 0
                          |    THEN out (" bei Zeile "); out (text (error line)) ;
                          |  FI ;
                          |  out (cr lf) .
                          |
                          |ENDPROC put error ;
                          |


stop .....................|PROC stop :
                          |
                          |  errorstop ("stop")
                          |
                          |ENDPROC stop ;
                          |
                          |ENDPACKET error handling ;
```

```
                            |(* ------------------- VERSION 6     05.05.86 ------------------- *
real **********************|PACKET real DEFINES                               (* Autor: J.Liedtke *
                            |
                            |   text ,
                            |   int ,
                            |   real ,
                            |   round ,
                            |   floor ,
                            |   frac ,
                            |   decimal exponent ,
                            |   set exp ,
                            |   INCR ,
                            |   DECR ,
                            |   abs ,
                            |   ABS ,
                            |   sign ,
                            |   SIGN ,
                            |   MOD ,
                            |   min ,
                            |   max ,
                            |   max real ,
                            |   small real :
                            |
                            |LET mantissa length = 13 ,
                            |    digit zero index = 1 ,
                            |    digit nine index = 10 ;
                            |INT CONST
                            |    decimal point index := -1 ;
                            |
                            |TEXT VAR mantissa ;
                            |
                            |ROW 10 REAL VAR real digit ;
                            |
                            |INT VAR i ; REAL VAR d := 0.0 ;
                            |FOR i FROM 1 UPTO 10 REP
                            |  real digit (i) := d ;
                            |  d := d + 1.0
                            |PER ;
                            |


maxreal ...................|REAL PROC max real : 9.999999999999e126 ENDPROC max real ;
                            |


smallreal .................|REAL PROC small real : 1.0e-12 ENDPROC small real ;
                            |


sld .......................|PROC sld (INT CONST in, REAL VAR real, INT VAR out) :
                            |  EXTERNAL 96
                            |ENDPROC sld ;
                            |


decimalexponent ...........|INT PROC decimal exponent (REAL CONST mantissa) :
                            |  EXTERNAL 97
                            |ENDPROC decimal exponent ;
                            |
setexp ...................|PROC set exp (INT CONST exponent, REAL VAR number) :
                          |  EXTERNAL 98
                          |ENDPROC set exp ;
                          |

tenpower .................|REAL PROC tenpower (INT CONST exponent) :
                          |  REAL VAR result := 1.0 ;
                          |  set exp (exponent, result) ;
                          |  result
                          |ENDPROC tenpower ;
                          |

floor ....................|REAL PROC floor (REAL CONST real) :
                          |  EXTERNAL 99
                          |ENDPROC floor ;
                          |

round ....................|REAL PROC round (REAL CONST real, INT CONST digits) :
                          |
                          |  REAL VAR result := real ;
                          |  IF (real <> 0.0) CAND (decimal exponent (real) + digits < mantissa
                          |       length)
                          |    THEN round result ;
                          |  FI ;
                          |  result .
                          |
                          |
   roundresult            |round result :
                          |  set exp (decimal exponent (result) + digits, result) ;
                          |  IF result >= 0.0
                          |    THEN result := floor (result + 0.5)
                          |    ELSE result := floor (result - 0.5)
                          |  FI ;
                          |  IF result <> 0.0
                          |    THEN set exp (decimal exponent (result) - digits, result)
                          |  FI .
                          |
                          |ENDPROC round ;
                          |
                          |TEXT VAR result ;
                          |

text .....................|TEXT PROC text (REAL CONST real) :
                          |
                          |  REAL VAR value := rounded to seven digits ;
                          |  IF value = 0.0
                          |    THEN "0.0"
                          |    ELSE
                          |      process sign ;
                          |      get mantissa (value) ;
                          |      INT CONST exponent := decimal exponent (value) ;
                          |      get short mantissa ;
                          |      IF exponent > 7 OR exponent < LENGTH short mantissa - 7
                          |        THEN scientific notation
                          |        ELSE short notation
                          |      FI
                          |    FI .
                          |
     roundedtosevendigits    |rounded to seven digits :
                             |  round ( real * tenpower( -decimal exponent(real) ) , 6 )
                             |        * tenpower ( decimal exponent(real) ) .
                             |
                             |
     processsign             |process sign :
                             |  IF value < 0.0
                             |    THEN result := "-" ;
                             |         value := - value
                             |    ELSE result := ""
                             |  FI .
                             |
                             |
     getshortmantissa        |get short mantissa :
                             |  INT VAR i := 7 ;
                             |  WHILE (mantissa SUB i) = "0" REP
                             |    i DECR 1
                             |  UNTIL i=1 END REP ;
                             |  TEXT CONST short mantissa := subtext (mantissa, 1, i) .
                             |
                             |
     scientificnotation      |scientific notation :
                             |  result CAT (mantissa SUB 1) ;
                             |  result CAT "." ;
                             |  result CAT subtext (mantissa, 2, 7) ;
                             |  result + "e" + text (exponent) .
                             |
                             |
     shortnotation           |short notation :
                             |  IF exponent < 0
                             |    THEN result + "0." + (-exponent - 1) * "0" + short mantissa
                             |    ELSE result CAT subtext (short mantissa, 1, exponent+1) ;
                             |         result CAT (exponent+1 - LENGTH short mantissa) * "0" ;
                             |         result CAT "." ;
                             |         result CAT subtext (short mantissa, exponent+2) ;
                             |         IF LENGTH short mantissa < exponent + 2
                             |           THEN result + "0"
                             |           ELSE result
                             |         FI
                             |  FI .
                             |
                             |ENDPROC text ;
                             |


   getmantissa ..............|PROC get mantissa (REAL CONST number) :
                             |
                             |  REAL VAR real mantissa := number ;
                             |  mantissa := "" ;
                             |  INT VAR i , digit ;
                             |  FOR i FROM 1 UPTO mantissa length REP
                             |    sld (0, real mantissa, digit) ;
                             |    mantissa CAT code (digit + 48)
                             |  PER ;
                             |
                             |ENDPROC get mantissa ;
                             |


   text .....................|TEXT PROC text (REAL CONST real, INT CONST length) :
                             |
                             |  INT CONST mantissa length := min (length - 7, 13) ;
```

```
                              |  IF mantissa length > 0
                              |    THEN construct scientific notation
                              |    ELSE result := length * "*"
                              |  FI ;
                              |  result .
                              |
                              |
       constructscientificnot |construct scientific notation :
                              |  REAL VAR value := rounded real ;
                              |  IF value = 0.0
                              |    THEN result := subtext (" 0.0                ", 1, length)
                              |    ELSE process sign ;
                              |         process mantissa ;
                              |         process exponent
                              |  FI .
                              |
                              |
       roundedreal            |rounded real :
                              |  round (real * tenpower ( -decimal exponent (real)) , mantissa
                              |       length - 1)
                              |  * tenpower (decimal exponent (real)) .
                              |
                              |
       processsign            |process sign :
                              |  IF value < 0.0
                              |    THEN result := "-"
                              |    ELSE result := "+"
                              |  FI .
                              |
                              |
       processmantissa        |process mantissa :
                              |  get mantissa (value) ;
                              |  result CAT (mantissa SUB 1) ;
                              |  result CAT "." ;
                              |  result CAT subtext (mantissa, 2, mantissa length) .
                              |
                              |
       processexponent        |process exponent :
                              |  IF decimal exponent (value) >= 0
                              |    THEN result CAT "e+"
                              |    ELSE result CAT "e-"
                              |  FI ;
                              |  result CAT text (ABS decimal exponent (value), 3) ;
                              |  change all (result, " ", "0") .
                              |
                              |ENDPROC text ;
                              |


    text .....................|TEXT PROC text (REAL CONST real, INT CONST length, fracs) :
                              |
                              |  REAL VAR value := round (real, fracs) ;
                              |  INT VAR  exponent  := decimal exponent (value) ;
                              |           IF value = 0.0 THEN exponent := 0 FI ;
                              |  INT VAR  floors    := exponent + 1 ,
                              |           floor length := length - fracs - 1 ;
                              |           IF value < 0.0 THEN floor length DECR 1 FI ;
                              |
                              |  IF value too big
                              |    THEN length * "*"
                              |    ELSE transformed value
                              |  FI .
                              |
                              |
    transformedvalue          |transformed value :
                              |  process leading blanks and sign ;
                              |  get mantissa (value) ;
                              |  result CAT subtext (mantissa, 1, floors) ;
                              |  IF LENGTH mantissa < floors
                              |    THEN result CAT (floors - LENGTH mantissa) * "0"
                              |  FI ;
                              |  result CAT "." ;
                              |  IF exponent < 0
                              |    THEN result CAT (-floors) * "0" ;
                              |         result CAT subtext (mantissa, 1, length - LENGTH result)
                              |    ELSE result CAT subtext (mantissa, floors+1, floors + fracs)
                              |  FI ;
                              |  IF LENGTH result < length
                              |    THEN result CAT (length - LENGTH result) * "0"
                              |  FI ;
                              |  result .
                              |
                              |
    processleadingblanksan    |process leading blanks and sign :
                              |  result := (floor length - max(floors,0)) * " " ;
                              |  IF value < 0.0
                              |    THEN result CAT "-" ;
                              |         value := - value
                              |  FI .
                              |
                              |
    valuetoobig               |value too big :
                              |  floors > floor length .
                              |
                              |ENDPROC text ;
                              |


real .........................|REAL PROC real (TEXT CONST text) :
                              |
                              |  skip leading blanks ;
                              |  sign ;
                              |  mantissa part ;
                              |  exponent ;
                              |  result .
                              |
                              |
    skipleadingblanks         |skip leading blanks :
                              |  INT VAR pos := 1 ;
                              |  skip blanks .
                              |
                              |
    skipblanks                |skip blanks :
                              |  WHILE (text SUB pos) = " " REP
                              |    pos INCR 1
                              |  PER .
                              |
                              |
    sign                      |sign :
                              |  BOOL VAR negative ;
                              |  IF (text SUB pos) = "-"
                              |    THEN negative := TRUE ;
                              |         pos INCR 1
                              |  ELIF (text SUB pos) = "+"
```

```
                         |    THEN negative := FALSE ;
                         |         pos INCR 1
                         |  ELSE   negative := FALSE
                         |  FI .
                         |
                         |
mantissapart             |mantissa part:
                         |  REAL VAR value ;
                         |  INT VAR exponent pos := 0 ;
                         |  get first digit ;
                         |  WHILE pos <= LENGTH text REP
                         |    digit := code (text SUB pos) - 47 ;
                         |    IF digit >= digit zero index AND digit <= digit nine index
                         |      THEN value := value * 10.0 + real digit (digit) ;
                         |           pos INCR 1
                         |    ELIF digit = decimal point index AND exponent pos = 0
                         |      THEN pos INCR 1 ;
                         |           exponent pos := pos
                         |    ELSE LEAVE mantissa part
                         |    FI
                         |  END REP .
                         |
                         |
getfirstdigit            |get first digit :
                         |  INT VAR digit := code (text SUB pos) - 47 ;
                         |  IF digit = decimal point index
                         |    THEN pos INCR 1 ;
                         |         exponent pos := pos ;
                         |         digit := code (text SUB pos) - 47
                         |  FI ;
                         |  IF digit >= digit zero index AND digit <= digit nine index
                         |    THEN value := real digit (digit) ;
                         |         pos INCR 1
                         |    ELSE set conversion (FALSE) ;
                         |         LEAVE real WITH 0.0
                         |  FI .
                         |
                         |
exponent                 |exponent :
                         |  INT VAR exp ;
                         |  IF exponent pos > 0
                         |    THEN exp := exponent pos - pos
                         |    ELSE exp := 0
                         |  FI ;
                         |  IF (text SUB pos) = "e"
                         |    THEN exp INCR int (subtext(text,pos+1))
                         |    ELSE no more nonblank chars permitted
                         |  FI .
                         |
                         |
nomorenonblankcharsper   |no more nonblank chars permitted :
                         |  skip blanks ;
                         |  IF pos > LENGTH text
                         |    THEN set conversion (TRUE)
                         |    ELSE set conversion (FALSE)
                         |  FI .
                         |
                         |
result                   |result :
                         |  value := value * tenpower (exp) ;
                         |  IF negative
                         |    THEN - value
```

```
                              |    ELSE value
                              |  FI .
                              |
                              |ENDPROC real ;
                              |
                              |


abs ..........................|REAL PROC abs (REAL CONST value) :
                              |
                              |  IF value >= 0.0
                              |    THEN  value
                              |    ELSE -value
                              |  FI
                              |
                              |ENDPROC abs ;
                              |


ABS ..........................|REAL OP ABS (REAL CONST value) :
                              |
                              |  abs (value)
                              |
                              |ENDOP ABS ;
                              |


sign .........................|INT PROC sign (REAL CONST value) :
                              |
                              |  IF   value < 0.0 THEN -1
                              |  ELIF value = 0.0 THEN  0
                              |  ELSE                   1
                              |  FI
                              |
                              |ENDPROC sign ;
                              |


SIGN .........................|INT OP SIGN (REAL CONST value) :
                              |
                              |  sign (value)
                              |
                              |ENDOP SIGN ;
                              |


MOD ..........................|REAL OP MOD (REAL CONST left, right) :
                              |
                              |  REAL VAR result := left - floor (left/right) * right ;
                              |  IF result < 0.0
                              |    THEN result + abs (right)
                              |    ELSE result
                              |  FI
                              |
                              |ENDOP MOD ;
                              |


frac .........................|REAL PROC frac (REAL CONST value) :
                              |
                              |  value - floor (value)
                              |
                              |ENDPROC frac ;
                              |


max ..........................|REAL PROC max (REAL CONST a, b) :
                              |
                              |  IF a > b THEN a ELSE b FI
                              |
                              |ENDPROC max ;
                              |


min ..........................|REAL PROC min (REAL CONST a, b) :
                              |
                              |  IF a < b THEN a ELSE b FI
                              |
                              |ENDPROC min ;
                              |


INCR .........................|OP INCR (REAL VAR dest, REAL CONST increment) :
                              |
                              |  dest := dest + increment
                              |
                              |ENDOP INCR ;
                              |


DECR .........................|OP DECR (REAL VAR dest, REAL CONST decrement) :
                              |
                              |  dest := dest - decrement
                              |
                              |ENDOP DECR ;
                              |


int ..........................|INT PROC int (REAL CONST value) :
                              |
                              |  IF value = minint value
                              |    THEN minint
                              |    ELSE compute int result ;
                              |         IF value < 0.0
                              |           THEN - result
                              |           ELSE   result
                              |         FI
                              |  FI .
                              |
                              |
   computeintresult           |compute int result :
                              |  INT VAR result := 0, digit ,i ;
                              |  REAL VAR mantissa := value ;
                              |
                              |  FOR i FROM 0 UPTO decimal exponent (value) REP
                              |    sld (0, mantissa, digit) ;
                              |    result := result * 10 + digit
                              |  PER .
                              |
                              |
   minintvalue                |minint value :  - 32768.0 .
```

```
    minint                |minint      :  - 32767 - 1 .
                          |
                          |ENDPROC int ;
                          |


 real ....................|REAL PROC real (INT CONST value) :
                          |
                          |  IF value < 0
                          |    THEN - real (-value)
                          |  ELIF value < 10
                          |    THEN real digit (value+1)
                          |    ELSE split value into head and last digit ;
                          |         real (head) * 10.0 + real digit (last digit+1)
                          |  FI .
                          |
                          |
    splitvalueintoheadandl |split value into head and last digit :
                          |  INT CONST
                          |  head := value DIV 10 ,
                          |  last digit := value - head * 10 .
                          |
                          |ENDPROC real ;
                          |
                          |ENDPACKET real ;
```

```
datehandling *************|PACKET date handling DEFINES date, time,        (*  Autor: H.
                          |     Indenbirken  *)
                          |                            time of day,       (*  Stand:
                          |                                 02.06.1986 (wk)*)
                          |                            month, day , year ,
                          |                            hour ,
                          |                            minute,
                          |                            second :
                          |
                          |LET middle yearlength = 31557380.0,
                          |    weeklength =    604800.0,
                          |    daylength  =     86400.0,
                          |    hours      =      3600.0,
                          |    minutes    =        60.0,
                          |    seconds    =         1.0;
                          |
                          |(* Tage bis zum Jahr 01.01.1900: 693970.25   5.995903e10   Sekunden
                          |          *)
                          |(* Dieser Tag ist ein Montag
                          |          *)
                          |
                          |REAL VAR begin of today := 0.0 , end of today := 0.0 ;
                          |
                          |TEXT VAR today , result ;
                          |
                          |
                          |ROW 12 REAL CONST previous days :: ROW 12 REAL : (0.0, 2678400.0,
                          |     5097600.0,
                          |                                             7776000.0, 10368000.0,
                          |                                                  13046400.0,
                          |                                            15638400.0, 18316800.0,
                          |                                                  20995200.0,
                          |                                            23587200.0, 26265600.0,
                          |                                                  28857600.0);
                          |


day ......................|REAL PROC day: day length END PROC day;


hour .....................|REAL PROC hour: hours     END PROC hour;


minute ...................|REAL PROC minute: minutes END PROC minute;


second ...................|REAL PROC second: seconds END PROC second;
                          |


date .....................|TEXT PROC date :
                          |
                          |  IF clock (1) < begin of today OR end of today <= clock (1)
                          |  THEN begin of today := clock (1) ;
                          |       end of today := floor (begin of
                          |            today/daylength)*daylength+daylength;
                          |       today := date (begin of today)
                          |  FI ;
                          |  today
                          |
                          |ENDPROC date ;
                          |
date ....................|TEXT PROC date (REAL CONST datum):
                         |  INT VAR year :: int (datum/middle yearlength),
                         |          day  :: int (((datum - datum MOD daylength) MOD middle
                         |                yearlength) / daylength) + 1;
                         |
                         |correct kalendary day;
                         |
                         |  calculate month and correct day;
                         |  result := daytext;
                         |  result CAT monthtext;
                         |  result CAT yeartext;
                         |  change all (result, " ", "0") ;
                         |  result .
                         |
                         |
   correctkalendaryday   |correct kalendary day:
                         |  IF day >= 60 AND NOT leapyear
                         |  THEN day INCR 1 FI  .
                         |
                         |
   leapyear              |leapyear:
                         |  IF year MOD 100 = 0
                         |    THEN year MOD 400 = 0
                         |    ELSE year MOD 4 = 0
                         |  FI.
                         |
                         |
   calculatemonthandcorre |calculate month and correct day:
                         |  INT VAR month;
                         |  IF day > 182
                         |  THEN IF day > 274
                         |       THEN IF day > 305
                         |            THEN IF day > 335
                         |                 THEN month := 12;
                         |                      day DECR 335
                         |                 ELSE month := 11;
                         |                      day DECR 305
                         |                 FI
                         |            ELSE month := 10;
                         |                 day DECR 274
                         |            FI
                         |       ELSE IF day > 213
                         |            THEN IF day > 244
                         |                 THEN month := 9;
                         |                      day DECR 244
                         |                 ELSE month := 8;
                         |                      day DECR 213
                         |                 FI
                         |            ELSE month := 7;
                         |                 day DECR 182
                         |            FI
                         |       FI
                         |  ELSE IF day > 91
                         |       THEN IF day > 121
                         |            THEN IF day > 152
                         |                 THEN month := 6;
                         |                      day DECR 152
                         |                 ELSE month := 5;
                         |                      day DECR 121
                         |                 FI
                         |            ELSE month := 4;
                         |                 day DECR 91
```

```
                          |            FI
                          |       ELSE IF day > 31
                          |            THEN IF day > 60
                          |                 THEN month := 3;
                          |                      day DECR 60
                          |                 ELSE month := 2;
                          |                      day DECR 31
                          |                 FI
                          |            ELSE month := 1 FI
                          |       FI
                          |  FI  .
                          |
                          |
   daytext                |daytext :
                          |  text (day, 2) + "."
                          |
                          |
   monthtext              |monthtext :
                          |  text (month,2) + "."
                          |
                          |
   yeartext               |yeartext:
                          |  IF 1900 <= year AND year < 2000
                          |    THEN text (year - 1900, 2)
                          |    ELSE text (year, 4)
                          |  FI   .
                          |
                          |END PROC date;
                          |


day ......................|TEXT PROC day (REAL CONST datum):
                          |  SELECT int ((datum MOD weeklength)/daylength) OF
                          |  CASE 1: "Donnerstag"
                          |  CASE 2: "Freitag"
                          |  CASE 3: "Samstag"
                          |  CASE 4: "Sonntag"
                          |  CASE 5: "Montag"
                          |  CASE 6: "Dienstag"
                          |  OTHERWISE "Mittwoch" ENDSELECT  .
                          |END PROC day;
                          |


month ....................|TEXT PROC month (REAL CONST datum):
                          |  SELECT int (subtext (date (datum), 4, 5)) OF
                          |  CASE  1: "Januar"
                          |  CASE  2: "Februar"
                          |  CASE  3: "März"
                          |  CASE  4: "April"
                          |  CASE  5: "Mai"
                          |  CASE  6: "Juni"
                          |  CASE  7: "Juli"
                          |  CASE  8: "August"
                          |  CASE  9: "September"
                          |  CASE 10: "Oktober"
                          |  CASE 11: "November"
                          |  OTHERWISE "Dezember" ENDSELECT  .
                          |
                          |END PROC month;
                          |
year ....................|TEXT PROC year (REAL CONST datum) :
                         |
                         |  TEXT VAR buffer := subtext (date (datum), 7) ;
                         |  IF LENGTH buffer = 2
                         |    THEN "19" + buffer
                         |    ELSE buffer
                         |  FI .
                         |
                         |ENDPROC year ;
                         |


timeofday ...............|TEXT PROC time of day :
                         |  time of day (clock (1))
                         |ENDPROC time of day ;
                         |


timeofday ...............|TEXT PROC time of day (REAL CONST value) :
                         |  subtext (time (value MOD daylength), 1, 5)
                         |ENDPROC time of day ;
                         |


time ....................|TEXT PROC time (REAL CONST value) :
                         |  time (value,10)
                         |ENDPROC time ;
                         |


time ....................|TEXT PROC time (REAL CONST value, INT CONST length) :
                         |  result := "" ;
                         |  IF length > 7
                         |    THEN result CAT hour ;
                         |         result CAT ":"
                         |  FI ;
                         |  result CAT minute ;
                         |  result CAT ":" ;
                         |  result CAT rest ;
                         |  change all (result, " ", "0") ;
                         |  result .
                         |
                         |
   hour                  |hour :
                         |  text (int (value/hours), length-8) .
                         |
                         |
   minute                |minute :
                         |  text (int (value/minutes MOD 60.0), 2)  .
                         |
                         |
   rest                  |rest :
                         |  text (value MOD minutes, 4, 1) .
                         |
                         |END PROC time ;
                         |


date ....................|REAL PROC date (TEXT CONST datum) :
                         |  split and check datum;
                         |  real (day no)*daylength +
                         |  previous days [month no] + calendary day +
                        |  floor (real (year no)*middleyearlength / daylength)*daylength  .
                        |
                        |
splitandcheckdatum      |split and check datum:
                        |  INT CONST day no :: first no;
                        |  IF NOT last conversion ok
                        |  THEN errorstop ("inkorrekte Datumsangabe (Tag) : " + datum) FI;
                        |
                        |  INT CONST month no :: second no;
                        |  IF NOT last conversion ok OR month no < 1 OR month no > 12
                        |  THEN errorstop ("inkorrekte Datumsangabe (Monat) : " + datum) FI
                        |
                        |  INT CONST year no :: third no + century;
                        |  IF NOT last conversion ok
                        |  THEN errorstop ("inkorrekte Datumsangabe (Jahr) : " + datum) FI;
                        |
                        |  IF day no < 1 OR day no > size of month
                        |  THEN errorstop ("inkorrekte Datumsangabe (Tag) : " + datum) FI
                        |
                        |
century                 |century:
                        |  IF (length (datum) - second pos) <= 2
                        |  THEN 1900
                        |  ELSE 0 FI  .
                        |
                        |
sizeofmonth             |size of month:
                        |  SELECT month no OF
                        |  CASE 1, 3, 5, 7, 8, 10, 12: 31
                        |  CASE 4, 6, 9, 11: 30
                        |  OTHERWISE february size ENDSELECT  .
                        |
                        |
februarysize            |february size:
                        |  IF leapyear
                        |  THEN 29
                        |  ELSE 28 FI  .
                        |
                        |
calendaryday            |calendary day:
                        |  IF month no > 2 AND leapyear
                        |  THEN daylength
                        |  ELSE 0.0 FI  .
                        |
                        |
leapyear                |leapyear:
                        |  year no MOD 4 = 0  AND year no MOD 400 <> 0  .
                        |
                        |
firstno                 |first no:
                        |  INT CONST first pos :: pos (datum, ".");
                        |  int (subtext (datum, 1, first pos-1))  .
                        |
                        |
secondno                |second no:
                        |  INT CONST second pos :: pos (datum, ".", first pos+1);
                        |  int (subtext (datum, first pos + 1, second pos-1))  .
                        |
                        |
thirdno                 |third no:
                        |  int (subtext (datum, second pos + 1))  .
                        |
                            |END PROC date;
                            |

  time .....................|REAL PROC time (TEXT CONST time) :
                            |  split and check time;
                            |  hour + min + sec  .
                            |
                            |
     splitandchecktime      |split and check time:
                            |  REAL CONST hour :: hour no * hours;
                            |  IF NOT last conversion ok
                            |  THEN errorstop ("inkorrekte Datumsangabe (Stunde) : " + time) FI;
                            |
                            |  REAL CONST min  :: min no * minutes;
                            |  IF NOT last conversion ok
                            |  THEN errorstop ("inkorrekte Datumsangabe (Minute) : " + time) FI;
                            |
                            |  REAL CONST sec :: sec no;
                            |  IF NOT last conversion ok
                            |  THEN errorstop ("inkorrekte Datumsangabe (Sekunde) : " + time) FI;
                            |
                            |  set conversion (hour ok AND min ok AND sec ok)  .
                            |
                            |
     hourno                 |hour no:
                            |  INT CONST hour pos :: pos (time, ":");
                            |  real (subtext (time, 1, hour pos-1))  .
                            |
                            |
     minno                  |min no:
                            |  INT VAR min pos :: pos (time, ":", hour pos+1);
                            |  IF min pos = 0
                            |    THEN real (subtext (time, hour pos + 1, LENGTH time))
                            |    ELSE real (subtext (time, hour pos + 1, min pos-1))
                            |  FI .
                            |
                            |
     secno                  |sec no:
                            |  IF min pos = 0
                            |    THEN 0.0
                            |    ELSE real (subtext (time, min pos + 1))
                            |  FI .
                            |
                            |
     hourok                 |hour ok: 0.0 <= hour AND hour < daylength  .
                            |
     minok                  |min ok:  0.0 <= min  AND min  < hours  .
                            |
     secok                  |sec ok:  0.0 <= sec  AND sec  < minutes  .
                            |END PROC time;
                            |
                            |END PACKET datehandling
```

```
                           |
commanddialogue ***********|PACKET command dialogue DEFINES                  (* Autor: J.Liedtke *)
                           |                                                  (* Stand:  25.11.83 *)
                           |      command dialogue ,
                           |      say ,
                           |      yes ,
                           |      no ,
                           |      param position ,
                           |      last param ,
                           |      std ,
                           |      QUIET ,
                           |      quiet :
                           |
                           |
                           |LET up      = ""3"" ,
                           |    right   = ""2"" ,
                           |    cr lf   = ""13""10"" ,
                           |    param pre  = " (""" ,
                           |    param post = """)"13""10"" ;
                           |
                           |
                           |TEXT VAR std param := "" ;
                           |
                           |BOOL VAR dialogue flag := TRUE ;
                           |
                           |INT VAR param x := 0 ;
                           |
                           |
                           |TYPE QUIET = INT ;
                           |


quiet .....................|QUIET PROC quiet :
                           |  QUIET:(0)
                           |ENDPROC quiet ;
                           |
                           |


commanddialogue ...........|BOOL PROC command dialogue :
                           |  dialogue flag
                           |ENDPROC command dialogue ;
                           |


commanddialogue ...........|PROC command dialogue (BOOL CONST status) :
                           |  dialogue flag := status
                           |ENDPROC command dialogue ;
                           |
                           |


yes .......................|BOOL PROC yes (TEXT CONST question) :
                           |
                           |  IF dialogue flag
                           |    THEN ask question
                           |    ELSE TRUE
                           |  FI .
                           |
    askquestion            |ask question :
                           |  out (question) ;
                           |  skip previous input chars ;
                           |  out (" (j/n) ? ") ;
                           |  get answer ;
                           |  IF correct answer
                           |    THEN out (answer) ;
                           |         out (cr lf) ;
                           |         positive answer
                           |    ELSE out (""7"") ;
                           |         LENGTH question + 9 TIMESOUT ""8"" ;
                           |         yes (question)
                           |  FI .
                           |
                           |
    getanswer              |get answer :
                           |  TEXT VAR answer ;
                           |  inchar (answer) .
                           |
                           |
    correctanswer          |correct answer :
                           |  pos ("jnyJNY", answer) > 0 .
                           |
                           |
    positiveanswer         |positive answer :
                           |  pos ("jyJY", answer) > 0 .
                           |
                           |
    skippreviousinputchars |skip previous input chars :
                           |  REP UNTIL incharety = "" PER .
                           |
                           |ENDPROC yes ;
                           |


no ........................|BOOL PROC no (TEXT CONST question) :
                           |
                           |  NOT yes (question)
                           |
                           |ENDPROC no ;
                           |


say .......................|PROC say (TEXT CONST message) :
                           |
                           |  IF dialogue flag
                           |    THEN out (message)
                           |  FI
                           |
                           |ENDPROC say ;
                           |


paramposition .............|PROC param position (INT CONST x) :
                           |
                           |  param x := x
                           |
                           |ENDPROC param position ;
                           |
lastparam ................|TEXT PROC last param :
                          |
                          |  IF param x > 0 AND online
                          |    THEN out (up) ;
                          |         param x TIMESOUT right ;
                          |         out (param pre) ;
                          |         out (std param) ;
                          |         out (param post)
                          |  FI ;
                          |  std param .
                          |
                          |ENDPROC last param ;
                          |


lastparam ................|PROC last param (TEXT CONST new) :
                          |  std param := new
                          |ENDPROC last param ;
                          |


std ......................|TEXT PROC std :
                          |  std param
                          |ENDPROC std ;
                          |
                          |ENDPACKET command dialogue ;
```

```
                               |(* ------------------- VERSION 2     06.03.86 ------------------- *)
thesaurushandling ********|PACKET thesaurus handling              (* Autor: J.Liedtke       *)
                               |
                               |       DEFINES     THESAURUS ,
                               |                   := ,
                               |                   empty thesaurus ,
                               |                   insert,          (* fuegt ein Element ein *)
                               |                   delete,          (* loescht ein Element falls vorhanden*)
                               |                   rename,          (* aendert ein Element falls vorhanden*)
                               |                   CONTAINS ,       (* stellt fest, ob enthalten *)
                               |                   link ,           (* index in thesaurus       *)
                               |                   name ,           (* name of entry            *)
                               |                   get ,            (* get next entry ("" is eof)*)
                               |                   highest entry :  (* highest valid index of thes*)
                               |
                               |
                               |TYPE THESAURUS = TEXT ;
                               |
                               |LET thesaurus size = 200 ,
                               |    nil = 0 ,
                               |    niltext = "" ,
                               |    max name length = 80 ,
                               |
                               |    begin entry char = ""0"" ,
                               |    end entry char   = ""1"" ,
                               |
                               |    nil entry        = ""0""1"" ,
                               |    nil name         = "" ,
                               |
                               |    quote            = """" ;
                               |
                               |TEXT VAR entry ;
                               |INT VAR  cache index := 0 ,
                               |         cache pos ;
                               |
                               |


access ...................|PROC access (THESAURUS CONST thesaurus, TEXT CONST name) :
                               |
                               |  construct entry ;
                               |  IF NOT cache identifies entry
                               |    THEN search through thesaurus list
                               |  FI ;
                               |  IF entry found
                               |    THEN cache index := code (list SUB (cache pos - 1))
                               |    ELSE cache index := 0
                               |  FI .
                               |
                               |
   constructentry              |construct entry :
                               |  entry := begin entry char ;
                               |  entry CAT name ;
                               |  decode invalid chars (entry, 2) ;
                               |  entry CAT end entry char .
                               |
                               |
   searchthroughthesaurus      |search through thesaurus list :
                               |  cache pos := pos (list, entry) .
                               |
    cacheidentifiesentry   |cache identifies entry :
                           |  cache pos <> 0 AND
                           |  pos (list, entry, cache pos, cache pos + LENGTH entry) = cache pos
                           |        .
                           |
                           |
    entryfound             |entry found :  cache pos > 0 .
                           |
                           |
    list                   |list :  CONCR (thesaurus) .
                           |
                           |ENDPROC access ;
                           |


  access ..................|PROC access (THESAURUS CONST thesaurus, INT CONST index) :
                           |
                           |  IF cache identifies index
                           |    THEN cache index := index ;
                           |         construct entry
                           |    ELSE cache pos := pos (list, code (index) + begin entry char) ;
                           |         IF entry found
                           |           THEN cache pos INCR 1 ;
                           |                cache index := index ;
                           |                construct entry
                           |           ELSE cache index := 0 ;
                           |                entry := niltext
                           |         FI
                           |  FI .
                           |
                           |
    constructentry         |construct entry :
                           |  entry := subtext (list, cache pos, pos (list, end entry char,
                           |       cache pos)) .
                           |
                           |
    cacheidentifiesindex   |cache identifies index :
                           |  subtext (list, cache pos-1, cache pos) = code (index) + begin
                           |       entry char .
                           |
                           |
    entryfound             |entry found :  cache pos > 0 .
                           |
                           |
    list                   |list :    CONCR (thesaurus) .
                           |
                           |ENDPROC access ;
                           |
                           |
                           |


  emptythesaurus ..........|THESAURUS PROC empty thesaurus :
                           |
                           |  THESAURUS : (""1"")
                           |
                           |ENDPROC empty thesaurus ;
                           |
                           |
:= .......................|OP := (THESAURUS VAR dest, THESAURUS CONST source ) :
                          |
                          |  CONCR (dest) := CONCR (source) .
                          |
                          |ENDOP := ;
                          |
                          |TEXT VAR insert name ;
                          |


insert ...................|PROC insert (THESAURUS VAR thesaurus, TEXT CONST name, INT VAR
                          |     index) :
                          |
                          |  insert name := name ;
                          |  decode invalid chars (insert name, 1) ;
                          |  IF insert name = "" OR LENGTH insert name > max name length
                          |    THEN index := nil ; errorstop ("Name unzulaessig")
                          |    ELSE insert element
                          |  FI .
                          |
                          |
   insertelement          |insert element :
                          |  search free entry ;
                          |  IF   entry found
                          |    THEN insert into directory
                          |    ELSE add entry to directory if possible
                          |  FI .
                          |
                          |
   searchfreeentry        |search free entry :
                          |  access (thesaurus, nil name) .
                          |
                          |
   insertintodirectory    |insert into directory :
                          |  change (list, cache pos + 1, cache pos, insert name) ;
                          |  index := cache index .
                          |
                          |
   addentrytodirectoryifp |add entry to directory if possible :
                          |  INT CONST next free index := code (list SUB LENGTH list) ;
                          |  IF next free index <= thesaurus size
                          |    THEN add entry to directory
                          |    ELSE directory overflow
                          |  FI .
                          |
                          |
   addentrytodirectory    |add entry to directory :
                          |  list CAT begin entry char ;
                          |  cache pos := LENGTH list ;
                          |  cache index := next free index ;
                          |  list CAT insert name ;
                          |  list CAT end entry char + code (next free index + 1) ;
                          |  index := cache index .
                          |
                          |
   directoryoverflow      |directory overflow :
                          |  index := nil .
                          |
                          |
   entryfound             |entry found :   cache index > 0 .
                          |
   list                  |list :          CONCR (thesaurus) .
                         |
                         |ENDPROC insert ;
                         |

decodeinvalidchars .......|PROC decode invalid chars (TEXT VAR name, INT CONST start pos) :
                         |
                         |  INT VAR invalid char pos := pos (name, ""0"", ""31"", start pos) ;
                         |  WHILE invalid char pos > 0 REP
                         |    change (name, invalid char pos, invalid char pos, decoded char) ;
                         |    invalid char pos := pos (name, ""0"", ""31"", invalid char pos)
                         |  PER .
                         |
                         |
   decodedchar           |decoded char :  quote + text(code(name SUB invalid char pos)) +
                         |     quote.
                         |
                         |ENDPROC decode invalid chars ;
                         |

insert ...................|PROC insert (THESAURUS VAR thesaurus, TEXT CONST name) :
                         |
                         |  INT VAR index ;
                         |  insert (thesaurus, name, index) ;
                         |  IF index = nil AND NOT is error
                         |    THEN errorstop ("THESAURUS-Ueberlauf")
                         |  FI .
                         |
                         |ENDPROC insert ;
                         |

delete ...................|PROC delete (THESAURUS VAR thesaurus, TEXT CONST name, INT VAR
                         |     index) :
                         |
                         |  access (thesaurus, name) ;
                         |  index := cache index ;
                         |  delete (thesaurus, index) .
                         |
                         |ENDPROC delete ;
                         |

delete ...................|PROC delete (THESAURUS VAR thesaurus, INT CONST index) :
                         |
                         |  access (thesaurus, index) ;
                         |  IF entry found
                         |    THEN delete entry
                         |  FI .
                         |
                         |
   deleteentry           |delete entry :
                         |  IF is last entry of thesaurus
                         |    THEN cut off as much as possible
                         |    ELSE set to nil entry
                         |  FI .
                         |
    settonilentry         |set to nil entry :
                          |  change (list, cache pos, cache pos + LENGTH entry - 1, nil entry)
                          |      .
                          |
                          |
    cutoffasmuchaspossible |cut off as much as possible :
                          |  WHILE predecessor is also nil entry REP
                          |    set cache to this entry
                          |  PER ;
                          |  list := subtext (list, 1, cache pos - 1) ;
                          |  erase cache .
                          |
                          |
    predecessorisalsonilen |predecessor is also nil entry :
                          |  subtext (list, cache pos - 3, cache pos - 2) = nil entry .
                          |
                          |
    setcachetothisentry   |set cache to this entry :
                          |  cache pos DECR 3 .
                          |
                          |
    erasecache            |erase cache :
                          |  cache pos := 0 ;
                          |  cache index := 0 .
                          |
                          |
    islastentryofthesaurus |is last entry of thesaurus :
                          |  pos (list, end entry char, cache pos) = LENGTH list - 1 .
                          |
                          |
    list                  |list :     CONCR (thesaurus) .
                          |
                          |
    entryfound            |entry found :   cache index > nil .
                          |
                          |ENDPROC delete ;
                          |
                          |


CONTAINS .................|BOOL OP CONTAINS (THESAURUS CONST thesaurus, TEXT CONST name ) :
                          |
                          |  IF   name = niltext OR LENGTH name > max name length
                          |    THEN FALSE
                          |    ELSE access (thesaurus, name) ; entry found
                          |  FI .
                          |
                          |
    entryfound            |entry found :   cache index > nil .
                          |
                          |ENDOP CONTAINS ;
                          |


rename ...................|PROC rename (THESAURUS VAR thesaurus, TEXT CONST old, new) :
                          |
                          |  rename (thesaurus, link (thesaurus, old), new)
                          |
                          |ENDPROC rename ;
                          |
rename ....................|PROC rename (THESAURUS VAR thesaurus, INT CONST index, TEXT CONST
                           |     new) :
                           |
                           |  insert name := new ;
                           |  decode invalid chars (insert name, 1) ;
                           |  IF insert name = "" OR LENGTH insert name > max name length
                           |    THEN errorstop ("Name unzulaessig")
                           |    ELSE change to new name
                           |  FI .
                           |
                           |
   changetonewname         |change to new name :
                           |  access (thesaurus, index) ;
                           |  IF cache index <> 0 AND entry <> ""
                           |    THEN change (list, cache pos + 1, cache pos + LENGTH entry - 2,
                           |         insert name)
                           |  FI .
                           |
                           |
   list                    |list :    CONCR (thesaurus) .
                           |
                           |ENDPROC rename ;
                           |


link ......................|INT PROC link (THESAURUS CONST thesaurus, TEXT CONST name) :
                           |
                           |  access (thesaurus, name) ;
                           |  cache index .
                           |
                           |ENDPROC link ;
                           |


name ......................|TEXT PROC name (THESAURUS CONST thesaurus, INT CONST index) :
                           |
                           |  access (thesaurus, index) ;
                           |  subtext (entry, 2, LENGTH entry - 1) .
                           |
                           |ENDPROC name ;
                           |


get .......................|PROC get (THESAURUS CONST thesaurus, TEXT VAR name, INT VAR index) :
                           |
                           |  identify index ;
                           |  REP
                           |    to next entry
                           |  UNTIL end of list COR valid entry found PER .
                           |
                           |
   identifyindex           |identify index :
                           |  IF index = 0
                           |    THEN cache index := 0 ;
                           |         cache pos   := 1
                           |    ELSE access (thesaurus, index)
                           |  FI .
                           |
                           |
   tonextentry             |to next entry :
                           |  cache pos := pos (list, begin entry char, cache pos + 1) ;
                           |  IF cache pos > 0
```

```
                          |    THEN get entry
                          |    ELSE get nil entry
                          |  FI .
                          |
                          |
    getentry              |get entry :
                          |  cache index INCR 1 ;
                          |  index := cache index ;
                          |  name := subtext (list, cache pos + 1, end entry pos - 1) .
                          |
                          |
    getnilentry           |get nil entry :
                          |  cache index := 0 ;
                          |  cache pos := 0 ;
                          |  index := 0 ;
                          |  name := "" .
                          |
                          |
    endentrypos           |end entry pos :     pos (list, end entry char, cache pos) .
                          |
                          |
    endoflist             |end of list :       index = 0 .
                          |
                          |
    validentryfound       |valid entry found : name <> "" .
                          |
                          |
    list                  |list :              CONCR (thesaurus) .
                          |
                          |ENDPROC get ;
                          |


 highestentry ............|INT PROC highest entry (THESAURUS CONST thesaurus) :
                          |     (*840813*)
                          |
                          |  code (list SUB LENGTH list) - 1 .
                          |
                          |
    list                  |list :   CONCR (thesaurus) .
                          |
                          |ENDPROC highest entry ;
                          |
                          |ENDPACKET thesaurus handling ;
```

```
                                |(* ------------------- VERSION 2     24.02.86 ------------------*)
localmanager *************|PACKET local manager                              (* Autor: J.Liedtke
                                |
                                |  DEFINES
                                |        create,        (* neue lokale Datei einrichten *)
                                |        new,           (* 'create' und Datei liefern   *)
                                |        old,           (* bestehende Datei liefern     *)
                                |        forget,        (* lokale Datei loeschen        *)
                                |        exists,        (* existiert Datei (lokal) ?    *)
                                |        status,        (* setzt und liefert Status     *)
                                |        rename,        (* Umbenennung                  *)
                                |        copy ,         (* Datenraum in Datei kopieren  *)
                                |        enter password,(* Passwort einfuehren          *)
                                |        write password ,
                                |        read password ,
                                |        write permission ,
                                |        read permission ,
                                |        begin list ,
                                |        get list entry ,
                                |        all :
                                |
                                |
                                |
                                |LET size = 200 ,
                                |    nil  = 0 ;
                                |
                                |INT VAR index ;
                                |
                                |TEXT VAR system write password := "" ,
                                |         system read password := "" ,
                                |         actual password ;
                                |
                                |INITFLAG VAR this packet := FALSE ;
                                |
                                |DATASPACE VAR password space ;
                                |
                                |BOUND ROW size STRUCT (TEXT write, read) VAR passwords ;
                                |
                                |
                                |THESAURUS VAR dir := empty thesaurus ;
                                |
                                |ROW size STRUCT (DATASPACE ds,
                                |                 BOOL protected,
                                |                 TEXT status) VAR crowd ;
                                |
                                |


initializeifnecessary ....|PROC initialize if necessary :
                                |
                                |  IF NOT initialized (this packet)
                                |    THEN system write password := "" ;
                                |         system read password := "" ;
                                |         dir := empty thesaurus ;
                                |         password space := nilspace ;
                                |         passwords := password space
                                |  FI
                                |
                                |ENDPROC initialize if necessary ;
                                |
                                |
                                |
create ...................|PROC create (TEXT CONST name) :
                          |
                          |IF exists (name )
                          |  THEN error (name, "existiert bereits") ;
                          |       index := nil
                          |  ELSE insert and initialize entry
                          |FI .
                          |
                          |
   insertandinitializeent |insert and initialize entry :
                          |  disable stop ;
                          |  insert (dir, name, index) ;
                          |  IF index <> nil
                          |    THEN crowd (index).ds := nilspace ;
                          |         IF is error
                          |           THEN delete (dir, name, index) ;
                          |                LEAVE create
                          |         FI ;
                          |         status (name, "") ;
                          |         crowd (index).protected := FALSE
                          |  ELIF NOT is error
                          |    THEN errorstop ("zu viele Dateien")
                          |  FI .
                          |
                          |ENDPROC create ;
                          |

new ......................|DATASPACE PROC new (TEXT CONST name) :
                          |
                          |  create (name) ;
                          |  IF index <> nil
                          |    THEN crowd (index).ds
                          |    ELSE nilspace
                          |  FI
                          |
                          |ENDPROC new ;
                          |

old ......................|DATASPACE PROC old (TEXT CONST name) :
                          |
                          |  initialize if necessary ;
                          |  index := link (dir, name) ;
                          |  IF index = 0
                          |    THEN error (name, "gibt es nicht") ;
                          |         nilspace
                          |    ELSE space
                          |  FI .
                          |
                          |
   space                  |space : crowd (index).ds .
                          |
                          |ENDPROC old ;
                          |

old ......................|DATASPACE PROC old (TEXT CONST name, INT CONST expected type) :
                          |
                          |  initialize if necessary ;
                          |  index := link (dir, name) ;
                          |  IF index = 0
                          |    THEN error (name, "gibt es nicht") ;
                          |         nilspace
                          |  ELIF type (space) <> expected type
                          |    THEN errorstop ("Datenraum hat falschen Typ") ;
                          |         nilspace
                          |    ELSE space
                          |  FI .
                          |
                          |
     space                |space : crowd (index).ds .
                          |
                          |ENDPROC old ;
                          |


   exists ................|BOOL PROC exists (TEXT CONST name) :
                          |
                          |  initialize if necessary ;
                          |  dir CONTAINS name
                          |
                          |ENDPROC exists ;
                          |


   forget ................|PROC forget (TEXT CONST name ) :
                          |
                          |  initialize if necessary ;
                          |  say ("""") ;
                          |  say (name) ;
                          |  IF   NOT exists (name)   THEN say (""" existiert nicht")
                          |  ELIF yes (""" loeschen") THEN forget (name, quiet)
                          |  FI .
                          |
                          |ENDPROC forget ;
                          |


   forget ................|PROC forget (TEXT CONST name, QUIET CONST q) :
                          |
                          |  initialize if necessary ;
                          |  disable stop ;
                          |  delete (dir, name, index) ;
                          |  IF index <> nil
                          |    THEN forget ( crowd (index).ds ) ;
                          |         crowd (index).status := ""
                          |  FI .
                          |
                          |ENDPROC forget ;
                          |


   forget ................|PROC forget :
                          |
                          |  BOOL VAR status := command dialogue ;
                          |  command dialogue (TRUE) ;
                          |  forget (last param) ;
                          |  command dialogue (status)
                          |
                          |ENDPROC forget ;
                          |
status ..................|PROC status (TEXT CONST name, status text) :
                         |
                         |  initialize if necessary ;
                         |  INT VAR index := link (dir, name) ;
                         |  IF index > 0
                         |    THEN crowd (index).status := date + " " + text (status text, 4)
                         |  FI
                         |
                         |ENDPROC status ;
                         |


status ..................|TEXT PROC status (TEXT CONST name) :
                         |
                         |  initialize if necessary ;
                         |  INT VAR index := link (dir, name) ;
                         |  IF index > 0
                         |    THEN crowd (index).status
                         |    ELSE ""
                         |  FI
                         |
                         |ENDPROC status ;
                         |


status ..................|PROC status (INT CONST pos, TEXT CONST status pattern) :
                         |
                         |  initialize if necessary ;
                         |  INT VAR index := 0 ;
                         |  WHILE index < highest entry (dir) REP
                         |    index INCR 1 ;
                         |    replace (actual status, pos , status pattern)
                         |  PER .
                         |
                         |
   actualstatus          |actual status : crowd (index).status .
                         |
                         |ENDPROC status ;
                         |


copy ....................|PROC copy (DATASPACE CONST source, TEXT CONST dest name) :
                         |
                         |  IF exists (dest name)
                         |    THEN error (dest name, "existiert bereits")
                         |    ELSE copy file
                         |  FI .
                         |
                         |
   copyfile              |copy file :
                         |  disable stop ;
                         |  create ( dest name ) ;
                         |  INT VAR index := link (dir, dest name) ;
                         |  IF index > nil
                         |    THEN forget (crowd (index).ds) ;
                         |         crowd (index).ds := source
                         |  FI
                         |
                         |ENDPROC copy ;
                         |
copy ....................|PROC copy (TEXT CONST source name, dest name) :
                         |
                         |  copy (old (source name), dest name)
                         |
                         |ENDPROC copy ;
                         |


rename ..................|PROC rename (TEXT CONST old name, new name) :
                         |
                         |  IF exists (new name)
                         |    THEN error (new name, "existiert bereits")
                         |  ELIF exists (old name)
                         |    THEN rename (dir, old name, new name) ;
                         |         last param (new name)
                         |  ELSE   error (old name, "gibt es nicht")
                         |  FI .
                         |
                         |ENDPROC rename ;
                         |
                         |


beginlist ...............|PROC begin list :
                         |
                         |  initialize if necessary ;
                         |  index := 0
                         |
                         |ENDPROC begin list ;
                         |


getlistentry ............|PROC get list entry (TEXT VAR entry, status text) :
                         |
                         |  get (dir, entry, index) ;
                         |  IF found
                         |    THEN status text := crowd (index).status ;
                         |    ELSE status text := "" ;
                         |  FI .
                         |
                         |
   found                 |found : index > 0 .
                         |
                         |ENDPROC get list entry ;
                         |
                         |


writepassword ...........|TEXT PROC write password :
                         |
                         |  system write password
                         |
                         |ENDPROC write password ;
                         |


readpassword ............|TEXT PROC read password :
                         |
                         |  system read password
                         |
                         |ENDPROC read password ;
                         |
                             |

enterpassword ...............|PROC enter password (TEXT CONST password) :
                             |
                             |  initialize if necessary ;
                             |  say (""3""5"") ;
                             |  INT CONST slash pos := pos (password, "/") ;
                             |  IF slash pos = 0
                             |    THEN system write password := password ;
                             |         system read password  := password
                             |    ELSE system write password := subtext (password, 1, slash pos-1)
                             |         ;
                             |         system read password  := subtext (password, slash pos+1)
                             |  FI .
                             |
                             |ENDPROC enter password ;
                             |

enterpassword ...............|PROC enter password (TEXT CONST file name, write pass, read pass) :
                             |
                             |  INT CONST index := link (dir, file name) ;
                             |  IF index > 0
                             |    THEN set protect password
                             |  FI .
                             |
                             |
   setprotectpassword        |set protect password :
                             |  IF write pass = "" AND read pass = ""
                             |    THEN crowd (index).protected := FALSE
                             |    ELSE crowd (index).protected := TRUE ;
                             |         passwords (index).write := write pass ;
                             |         passwords (index).read  := read pass
                             |  FI .
                             |
                             |ENDPROC enter password ;
                             |

passwordindex ...............|INT PROC password index (TEXT CONST file name) :
                             |
                             |  initialize if necessary ;
                             |  INT CONST index := link (dir, file name) ;
                             |  IF index > 0 CAND crowd (index).protected
                             |    THEN index
                             |    ELSE 0
                             |  FI
                             |
                             |ENDPROC password index ;
                             |

readpermission ..............|BOOL PROC read permission (TEXT CONST name, supply password) :
                             |
                             |  (****************************************************************)
                             |  (*  for reasons of data security the password check algorithm  *)
                             |  (*  must not copy parts of the file password into variables     *)
                             |  (*  located in the standard dataspace!                          *)
                             |  (****************************************************************)
                             |
                             |  access file password ;
```

```
                                 | file has no password COR (supply password <> "-" AND read password
                                 |      match) .
                                 |
                                 |
      readpasswordmatch          |read password match :
                                 | file password.read = supply password OR file password.read = "" .
                                 |
                                 |
      accessfilepassword         |access file password :
                                 | INT CONST pw index := password index (name) .
                                 |
                                 |
      filepassword               |file password :  passwords (pw index) .
                                 |
                                 |
      filehasnopassword          |file has no password :  pw index = 0 .
                                 |
                                 |ENDPROC read permission ;
                                 |


   writepermission ..........|BOOL PROC write permission (TEXT CONST name, supply password) :
                                 |
                                 | (****************************************************************)
                                 | (*  for reasons of data security the password check algorithm   *)
                                 | (*  must not copy parts of the file password into variables     *)
                                 | (*  located in the standard dataspace!                          *)
                                 | (****************************************************************)
                                 |
                                 | access file password ;
                                 | file has no password COR (supply password <> "-" AND write
                                 |      password match).
                                 |
                                 |
      writepasswordmatch         |write password match :
                                 | file password.write = supply password OR file password.write = ""
                                 |
                                 |
      accessfilepassword         |access file password :
                                 | INT CONST pw index := password index (name) .
                                 |
                                 |
      filepassword               |file password :  passwords (pw index) .
                                 |
                                 |
      filehasnopassword          |file has no password :  pw index = 0 .
                                 |
                                 |ENDPROC write permission ;
                                 |


   all ......................|THESAURUS PROC all :
                                 |
                                 | initialize if necessary ;
                                 | THESAURUS VAR result := dir ; (*ueberfluessig ab naechstem
                                 |      Compiler *)
                                 | result
                                 |
                                 |ENDPROC all ;
                                 |
error ....................|PROC error (TEXT CONST file name, error text) :
                          |
                          |  errorstop ("""" + file name + """ " + error text)
                          |
                          |ENDPROC error ;
                          |
                          |ENDPACKET local manager ;
```

```
patternmatch **************|PACKET pattern match DEFINES                    (* Author:
                           |     P.Heyderhoff *)
                           |                                                 (* Date:
                           |                                                      09.06.1986
                           |                                                      *)
                           |    -,
                           |    OR,
                           |    **,
                           |    any,
                           |    notion,
                           |    bound,
                           |    match,
                           |    matchpos,
                           |    matchend,
                           |    somefix,
                           |    UNLIKE,
                           |    LIKE :
                           |
                           |(*------- Operation codes of the internal intermeadiate language:
                           |     --------*)
                           |
                           |LET
                           |    z        = ""0"",
                           |    stopz    = ""1""0"",
                           |    closez   = ""2""0"",
                           |    closor   = ""2""0""3""0"",
                           |    or       = ""3"",
                           |    oralpha  = ""3""5"",
                           |    open2    = ""4""0""4""0"",
                           |    alpha    = ""5"",
                           |    alphaz   = ""5""0"",
                           |    lenz     = ""6""0"",
                           |    nilz     = ""6""0""0""0""7""0"",       (* =    any (0)      *)
                           |    starz    = ""7""0"",
                           |    star     = ""8""0""2""7""0""1""0"",    (* =    any ** 1     *)
                           |    powerz   = ""8""0"",
                           |    powerz0  = ""8""0""1"",
                           |    notionz  = ""9""0"",
                           |    fullz    = ""10""0"",
                           |    boundz   = ""11""0"";
                           |(*---------------------------------------------------------------
                           |     -----*)
                           |
                           |LET undefined = 0,                                   (* fixleft
                           |     value      *)
                           |    forcer    = 0,                                   (* value
                           |         parameter   *)
                           |    delimiter = " !""#$%&'()*+,-./:;<=>?@^_`-";      (* for 'PROC
                           |         notion' *)
                           |


- .........................|TEXT OP - (TEXT CONST alphabet ):
                           |     p:= "";
                           |     INT VAR j;
                           |     FOR j FROM 0 UPTO 255
                           |     REP IF   pos(alphabet,code(j)) = 0
                           |         THEN p CAT code(j)
                           |         FI
                           |     PER;
                           |     p
                           |  ENDOP -;
                           |

OR ........................|TEXT OP OR (TEXT CONST a, b):
                           |     open2 + notnil (a) + closor + notnil (b) + closez
                           |  ENDOP OR;
                           |

** ........................|TEXT OP ** (TEXT CONST p, INT CONST x):
                           |     powerz + code (1+x) + notnil (p) + stopz
                           |  ENDOP **;
                           |
                           |TEXT CONST any:= starz;
                           |

any .......................|TEXT PROC any (INT CONST n):
                           |     TEXT VAR t:= "  ";
                           |     replace (t, 1, ABSn);
                           |     lenz + t + starz
                           |  ENDPROC any;
                           |

any .......................|TEXT PROC any (TEXT CONST a): alphaz + a + starz ENDPROC any;
                           |

any .......................|TEXT PROC any (INT CONST n, TEXT CONST a):
                           |     TEXT VAR t:= "  ";
                           |     replace (t, 1, ABSn);
                           |     lenz + t + alphaz + a + starz
                           |  ENDPROC any;
                           |

notion ....................|TEXT PROC notion (TEXT CONST t): notionz + notnil(t) + stopz ENDPROC
                           |     notion;
                           |

notnil ....................|TEXT PROC notnil (TEXT CONST t):
                           |     IF   t = ""
                           |     THEN nilz
                           |     ELSE t
                           |     FI
                           |  ENDPROC notnil;
                           |
                           |TEXT CONST bound := boundz;
                           |

full ......................|TEXT PROC full (TEXT CONST t): fullz + t + stopz ENDPROC full;
                           |

match .....................|TEXT PROC match (INT CONST x):
                           |     subtext (p, matchpos(x), matchend(x))
                           |  ENDPROC match;
                           |
matchpos ................|INT PROC matchpos (INT CONST x): mapos (1 + x MOD 256) ENDPROC
                         |     matchpos;
                         |


matchend ................|INT PROC matchend (INT CONST x): maend (1 + x MOD 256) - 1
                         | ENDPROC matchend;
                         |
                         |(*----------------- GLOBAL VARIABLES:
                         |     ----------------------------------*)
                         |
                         |ROW  256 INT VAR
                         |                (* Table of match registers. Each entry consists of
                         |                     two *)
                         |                (* pointers, which points to the TEXT object 't'
                         |                      *)
                         |    mapos,      (* points to the beginning of the match
                         |            *)
                         |    maend;      (* points to the position after the end of match
                         |            *)
                         |
                         |INT  VAR ppos, tpos,   (* workpositions in pattern 'p' and text 't'
                         |         *)
                         |         floatpos,     (* accumulation of all pending floatlengths
                         |                *)
                         |         failpos,      (* result of 'PROC in alpha'
                         |                *)
                         |         plen, tlen,   (* length of pattern 'p' and length of text
                         |              't'   *)
                         |         skipcount,    (* for track forward skipping
                         |                *)
                         |         multi, vari;  (* for handling of nonexclusive alternatives
                         |                *)
                         |
                         |TEXT VAR p,            (* the pattern to be find  or some result
                         |         *)
                         |         stack,        (* stack of pending assignments
                         |                *)
                         |         alphabet:=""; (* result of 'PROC find alpha', reset to nil
                         |                *)
                         |                       (* after its usage by 'find any'
                         |                              *)
                         |
                         |BOOL VAR fix,          (* text position is fixed and not floating
                         |         *)
                         |         no vari;      (* not variing the order of alternatives
                         |                *)
                         |


somefix .................|TEXT PROC somefix (TEXT CONST pattern):
                         |
                         |    (* delivers the first text occuring unconditionally in the
                         |         pattern *)
                         |
                         |    p:= pattern;
                         |    INT VAR j:= 1, n:= 0, k, len:= LENGTH p;
                         |    REP
                         |        SELECT text( subtext (p, j, j+1), 2) ISUB 1 OF
                         |        CASE 1,3,7,9,10,11: j INCR 2
                         |        CASE 2:             j INCR 2; n DECR 1 (* condition closed
                         |            *)
                         |        CASE 4:            j INCR 2; n INCR 1 (* condition opened
                         |            *)
                         |        CASE 5:            j := pos (p, starz, j+2) + 2
                         |        CASE 6:            j INCR 4
                         |        CASE 8:            j INCR 3
                         |        OTHERWISE  k:= pos(p, z, j+1) - 1;     ,
                         |                   IF k <= 0 THEN k:= 1+len FI;
                         |                   IF   star found
                         |                   THEN change (p, starpos, starpos, star);
                         |                        len:= LENGTH p;
                         |                        k:= starpos
                         |                   FI;
                         |                   IF n =  0 CAND ( p SUB k ) <> or CAND k > j
                         |                   THEN LEAVE somefix WITH subtext(p,j,k-1)
                         |                   ELSE j:=k
                         |                   FI
                         |        ENDSELECT
                         |    UNTIL j > len
                         |    PER;
                         |    "" .
                         |
                         |
     starfound           | star found:
                         |    INT VAR starpos:= pos (p, "*", j);
                         |    starpos > 0 CAND starpos <= k .
                         |
                         | ENDPROC somefix;
                         |

  skip ..................|PROC skip (TEXT CONST p, BOOL CONST upto or):
                         |
                         |    (* skips 'ppos' upto the end of the opened nest, n = nesting
                         |         level *)
                         |
                         |    INT VAR n:= 0;
                         |    REP
                         |        SELECT text (subtext (p, ppos, ppos+1), 2) ISUB 1 OF
                         |        CASE 1,2:    IF   n = 0
                         |                     THEN LEAVE skip
                         |                     FI;
                         |                     ppos INCR 2;
                         |                     nDECR1
                         |        CASE 3:      IF   n = 0 CAND upto or
                         |                     THEN LEAVE skip
                         |                     FI;
                         |                     ppos INCR 2
                         |        CASE 7:      ppos INCR 2
                         |        CASE 4,9,10,11: ppos INCR 2;
                         |                     n INCR 1
                         |        CASE 5:      ppos:= pos (p, starz, ppos+2) + 2
                         |        CASE 6:      ppos INCR 4
                         |        CASE 8:      ppos INCR 3;
                         |                     n INCR 1
                         |        OTHERWISE    ppos:= pos(p, z, ppos+1) - 1;
                         |                     IF   ppos < 0
                         |                     THEN ppos:= plen;
                         |                          LEAVE skip
                         |                     FI
                         |        ENDSELECT
                         |    PER
                         | ENDPROC skip;
```

```
                             |

UNLIKE ......................|BOOL OP UNLIKE (TEXT CONST t, p): NOT ( t LIKE p ) ENDOP UNLIKE;
                             |

LIKE ........................|BOOL OP LIKE (TEXT CONST t, pattern):
                             |     init;
                             |     BOOL CONST found:= find (t,1,1, fixresult, floatresult);
                             |     save;
                             |     found.
                             |
                             |
   init                      |  init:     no vari:= TRUE;
                             |            vari:= 0;
                             |            tlen:= 1 + LENGTH t;
                             |            p:= full (pattern);
                             |            IF   pos (p, bound) > 0
                             |            THEN
                             |                 IF   subtext (p, 14, 15) = bound
                             |                 THEN p:= subtext (p, 1, 8) + powerz0 + subtext (p,
                             |                      16)
                             |                 FI;
                             |                 plen:= LENGTH p - 7;
                             |                 IF   subtext (p, plen, plen+1) = bound
                             |                 THEN p:= subtext (p, 1, plen - 1) + stopz + stopz
                             |                 FI;
                             |            FI;
                             |            plen:= LENGTH p + 1;
                             |            INT VAR fixresult, floatresult;
                             |            tpos:= 1;
                             |            floatpos:= 0;
                             |            stack:= "";
                             |            alphabet:= "";
                             |            fix:= TRUE;
                             |            skipcount:= 0;
                             |            multi:= 0.
                             |
                             |
   save                      |  save:     p:= t
                             |
                             |  ENDOP LIKE;
                             |
                             |(*-------- Realisation of the pattern matching algorithms 'find'
                             |     --------*)
                             |

find ........................|BOOL PROC find
                             |     (TEXT CONST t, INT CONST unit, from, INT VAR fixleft,
                             |          floatlen):
                             |
                             |     initialize;
                             |     BOOL CONST found:= pattern unit;
                             |     SELECT next command * unit OF
                             |            CASE 0,1,2:         found
                             |            CASE 3:             next;
                             |                                find alternative
                             |            OTHERWISE           find concatenation
                             |     ENDSELECT .
                             |
findalternative       | find alternative:
                      |    IF   found
                      |    THEN save left position;
                      |         backtrack;
                      |         IF   find pattern CAND better
                      |         THEN note multiplicity
                      |         ELSE back to first one
                      |         FI
                      |    ELSE backtrack multi
                      |    FI.
                      |
                      |
better                | better:                permutation XOR more left.
                      |
                      |
permutation           | permutation:           vari MOD 2 = 1.
                      |
                      |
saveleftposition      | save left position:    j:= fixleft.
                      |
                      |
moreleft              | more left:             j > fixleft.
                      |
                      |
backtrackmulti        | backtrack multi:       multi:= 2 * backmulti + 1;
                      |                        vari:= backvari DIV 2;
                      |                        find pattern.
                      |
                      |
notemultiplicity      | note multiplicity:     multi:= 2 * multi + 1;
                      |                        vari:= vari DIV 2;
                      |                        TRUE.
                      |
                      |
backtofirstone        | back to first one:     backtrack;
                      |                        IF   find first subpattern
                      |                        THEN skip (p, FALSE);
                      |                             note multiplicity
                      |                        ELSE errorstop ("pattern");
                      |                             FALSE
                      |                        FI.
                      |
                      |
findconcatenation     | find concatenation:
                      |    IF   found
                      |    THEN IF ppos=plen COR find pattern COR track forward
                      |         COR ( multi > backmulti CAND vari = 0 CAND find variation
                      |              )
                      |         THEN TRUE
                      |         ELSE backtrack; FALSE
                      |         FI
                      |    ELSE skip (p, TRUE); FALSE
                      |    FI.
                      |
                      |
trackforward          | track forward:                    (* must be performed before
                      |      variation *)
                      |    j:=0;
                      |    last multi:= multi;
                      |    last vari:= vari;
                      |    WHILE skipcount = 0
                      |    REP   IF tlen = tpos
```

```
                       |           THEN LEAVE track forward WITH FALSE
                       |           FI;
                       |           backtrack;
                       |           j INCR 1;
                       |           skipcount:= j
                       |     UNTIL find first subpattern CAND find pattern
                       |     PER;
                       |     j:= skipcount;
                       |     skipcount:=0;
                       |     j=0.
                       |
                       |
findvariation          |  find variation:
                       |     multi:= last multi;
                       |     vari:= last vari;
                       |     FOR k FROM 1 UPTO (multi+1) DIV (backmulti+1) - 1
                       |     REP backtrack with variation;
                       |         IF   find first subpattern CAND find pattern
                       |         THEN vari:=0;
                       |              LEAVE find variation WITH TRUE
                       |         FI
                       |     PER;
                       |     FALSE.
                       |
                       |
backtrackwithvariation |  backtrack with variation:
                       |     backtrack;
                       |     vari:= k.
                       |
                       |
findpattern            |  find pattern:
                       |     find (t, 1, ppos+forcer, fixresult, floatresult) CAND keep
                       |          result.
                       |
                       |
findfirstsubpattern    |  find first subpattern:
                       |     find (t, 0, from, fixresult, floatresult) CAND keep result
                       |
                       |
initialize             |  initialize:
                       |     INT  VAR   j,
                       |                k,
                       |                fixresult,
                       |                floatresult,
                       |                last multi,
                       |                last vari;
                       |     BOOL CONST backfix:= fix;
                       |     TEXT CONST backstack:= stack;
                       |     floatlen:= 0;
                       |     INT  CONST back:= tpos,
                       |                backfloat:= floatpos,
                       |                backskip:= skipcount,
                       |                backmulti:= multi,
                       |                backvari:= vari;
                       |     fixleft:= fixleft0.
                       |
                       |
fixleft0               |  fixleft0: IF fix THEN back ELSE undefined FI.
                       |
backtrack             | backtrack:
                      |    fix:= backfix;
                      |    tpos:= back;
                      |    fixleft:= fixleft0;
                      |    floatlen:= 0;
                      |    floatpos:= backfloat;
                      |    stack:= backstack;
                      |    skipcount:= backskip;
                      |    multi:= backmulti;
                      |    vari:= backvari.
                      |
                      |
keepresult            | keep result:
                      |    IF   fixleft = undefined
                      |    THEN IF   fixresult = undefined
                      |         THEN floatlen INCR floatresult
                      |         ELSE fixleft  := fixresult - floatlen;
                      |              floatpos DECR floatlen;
                      |              floatlen:= 0
                      |         FI
                      |    FI;
                      |    TRUE.
                      |
                      |
patternunit           | pattern unit:
                      |    init ppos;
                      |    SELECT command OF
                      |           CASE 1,2:   find end
                      |           CASE 3:     find nil
                      |           CASE 4:     find choice
                      |           CASE 5:     find alphabet
                      |           CASE 6:     find fixlength any
                      |           CASE 7:     find varlength any
                      |           CASE 8:     find and store match
                      |           CASE 9:     find notion
                      |           CASE 10:    find full
                      |           CASE 11:    next; find nil
                      |           OTHERWISE   find plain text                    END
                      |                SELECT.
                      |
                      |
initppos              | init ppos:    ppos:= from + 2.
                      |
                      |
command               | command:      text (subtext (p, from, from+1), 2) ISUB 1.
                      |
                      |
nextcommand           | next command: text (subtext (p, ppos, ppos+1), 2) ISUB 1.
                      |
                      |
next                  | next:         ppos INCR 2.
                      |
                      |
findend               | find end:     ppos DECR 2;
                      |               fixleft:= tpos;
                      |               LEAVE find WITH TRUE;
                      |               TRUE.
                      |
                      |
findnil               | find nil:     ppos DECR 2;
                      |               fixleft:= tpos;
                      |               TRUE.
```

```
                          |
                          |
findchoice                | find choice:  IF   find pattern
                          |               THEN next; TRUE
                          |               ELSE next; FALSE
                          |               FI.
                          |
                          |
findplaintext             | find plain text: find text upto next command;
                          |                  IF   fix        THEN allow fix position only
                          |                  ELIF text found THEN allow variable position
                          |                                  ELSE allow backtrack
                          |                  FI.
                          |
                          |
findtextuptonextcomman    | find text upto next command:
                          |    ppos:= pos (p, z, from + 1);
                          |    IF   ppos = 0
                          |    THEN ppos:= plen
                          |    ELSE ppos DECR 1
                          |    FI;
                          |    IF   star found
                          |    THEN change (p, starpos, starpos, star);
                          |         plen:= 1 + LENGTH p;
                          |         ppos:= starpos
                          |    FI;
                          |    tpos:= pos (t, subtext (p, from, ppos - 1), tpos).
                          |
                          |
starfound                 | star found:
                          |    INT VAR starpos:= pos (p, "*", from);
                          |    starpos > 0 CAND starpos <= ppos .
                          |
                          |
textfound                 | text found:
                          |    WHILE skipcount > 0 CAND tpos > 0
                          |    REP   skipcount DECR 1;
                          |          tpos:= pos (t, subtext(p,from,ppos-1), tpos+1)
                          |    PER;
                          |    tpos > 0 .
                          |
                          |
allowfixpositiononly      | allow fix position only:
                          |    IF   tpos = back
                          |    THEN tpos INCR (ppos-from); TRUE
                          |    ELSE tpos:= back;
                          |         from = ppos
                          |    FI.
                          |
                          |
allowvariableposition     | allow variable position:
                          |    IF   alphabet = "" COR in alpha (t, back, tpos)
                          |    THEN fix it;
                          |         tpos INCR (ppos-from);
                          |         TRUE
                          |    ELSE tpos:= back;
                          |         FALSE
                          |    FI.
                          |
allowbacktrack          | allow backtrack:
                        |    tpos:= back;
                        |    IF   from = ppos
                        |    THEN fix it;
                        |         TRUE
                        |    ELSE FALSE
                        |    FI .
                        |
                        |
findalphabet            | find alphabet:
                        |    j:= pos (p, starz, ppos);
                        |    alphabet:= subtext (p, ppos, j-1);
                        |    ppos := j;
                        |    TRUE.
                        |
                        |
findfixlengthany        | find fixlength any:
                        |    get length value;
                        |    find alpha attribut;
                        |    IF   alphabet = ""
                        |    THEN find any with fix length
                        |    ELSE find any in alphabet with fix length
                        |    FI.
                        |
                        |
getlengthvalue          | get length value:
                        |    floatlen:= subtext(p, ppos, ppos+1) ISUB 1;
                        |    ppos INCR 4.
                        |
                        |
findalphaattribut       | find alpha attribut:
                        |    IF (p SUB (ppos-2)) = alpha CAND find alphabet
                        |    THEN next
                        |    FI.
                        |
                        |
findanywithfixlength    | find any with fix length:
                        |    tpos INCR floatlen;
                        |    IF   tpos > tlen
                        |    THEN tpos:= back;
                        |         floatlen:=0;
                        |         FALSE
                        |    ELSE IF   fix THEN floatlen:= 0
                        |         ELIF floatlen = 0
                        |         THEN fix it                        (* unlike niltext 6.6.
  +                     |              *)
                        |         ELSE floatpos INCR floatlen
                        |         FI;
                        |         TRUE
                        |    FI.
                        |
                        |
findanyinalphabetwithf  | find any in alphabet with fix length:
                        |    IF   first character in alpha
                        |    THEN IF NOT fix THEN fix it FI;
                        |         set fix found
                        |    ELSE set fix not found
                        |    FI.
                        |
firstcharacterinalpha   | first character in alpha:
                        |   (fix COR advance) CAND in alpha (t, tpos, tpos+floatlen).
                        |
                        |
advance                 | advance:
                        |    FOR tpos FROM back UPTO tlen
                        |    REP IF   pos (alphabet, t SUB tpos) > 0
                        |        THEN LEAVE advance WITH TRUE
                        |        FI
                        |    PER;
                        |    FALSE.
                        |
                        |
fixit                   | fix it:
                        |    fixleft:= back-floatpos;
                        |    make fix (back);
                        |    fixleft:= tpos.
                        |
                        |
setfixfound             | set fix found:
                        |    tpos INCR floatlen;
                        |    floatlen:= 0;
                        |    alphabet:= "";
                        |    TRUE.
                        |
                        |
setfixnotfound          | set fix not found:      tpos:= back;
                        |                         alphabet:= "";
                        |                         floatlen:= 0;
                        |                         FALSE.
                        |
                        |
findvarlengthany        | find varlength any:     IF   alphabet = ""
                        |                         THEN really any
                        |                         ELSE find varlength any in alphabet
                        |                         FI.
                        |
                        |
reallyany               | really any:             IF   fix
                        |                         THEN fix:= FALSE;
                        |                              fixleft:= tpos
                        |                         ELIF floatpos = 0
                        |                         THEN fixleft:= tpos                (* 6.6.
    +                   |                              *)
                        |                         FI;
                        |                         TRUE .
                        |
                        |
findvarlengthanyinalph  | find varlength any in alphabet:
                        |                         IF fix THEN fixleft := tpos FI;
                        |                         IF   fix CAND pos (alphabet, t SUB tpos) > 0
                        |                         COR  NOT fix CAND advance
                        |                         THEN IF NOT fix THEN fix it FI;
                        |                              set var found
                        |                         ELSE set var not found
                        |                         FI.
                        |
                        |
setvarfound             | set var found:          tpos:= end of varlength any;
                        |                         alphabet:= "";
                        |                         TRUE.
```

```
setvarnotfound         |  set var not found:    tpos:= back;
                       |                        alphabet:= "";
                       |                        FALSE.
                       |
endofvarlengthany      |  end of varlength any: IF   NOT in alpha(t,tpos,tlen)
                       |                        THEN failpos
                       |                        ELSE tlen
                       |                        FI.
                       |
                       |
findandstorematch      |  find and store match: get register name;
                       |                        IF  find pattern
                       |                        THEN next;
                       |                             store;
                       |                             TRUE
                       |                        ELSE next;
                       |                             FALSE
                       |                        FI.
                       |
                       |
store                  |  store:                IF   fix
                       |                        THEN mapos (reg):= fixleft;
                       |                             maend (reg):= tpos
                       |                        ELSE stack CAT code(floatlen) +
                       |                                       code(floatpos) +
                       |                                            code(fixleft) + c
                       |                        FI.
                       |
                       |
getregistername        |  get register name:    TEXT CONST c:= p SUB (ppos);
                       |                        INT  VAR reg:= code (c);
                       |                        ppos INCR 1.
                       |
                       |
findnotion             |  find notion:    float notion;
                       |                  exhaust notion .
                       |
                       |
floatnotion            |  float notion:   j:= back;
                       |                  REP IF   find pattern
                       |                      THEN IF   is notion (t, fixleft)
                       |                           THEN LEAVE find notion WITH TRUE
                       |                           ELIF backfix
                       |                           THEN LEAVE float notion
                       |                           ELSE go ahead FI
                       |                      ELIF j=back
                       |                      THEN next;
                       |                           LEAVE find notion WITH FALSE
                       |                      ELSE LEAVE float notion
                       |                      FI
                       |                  PER.
                       |
                       |
goahead                |  go ahead:       j INCR 1;
                       |                  IF simple THEN j:= max (tpos, j) FI;
                       |                  notion backtrack.
                       |
                       |
simple                 |  simple:         k:= from;
                       |                  REP k := pos (p, z, k+2);
                       |                      IF   k > ppos-3
                       |                      THEN LEAVE simple WITH TRUE
```

```
                          |                        ELIF pos (oralpha, p SUB k-1) > 0
                          |                        THEN LEAVE simple WITH FALSE
                          |                        FI
                          |                    PER;
                          |                    FALSE.
                          |
                          |
notionbacktrack           | notion backtrack: tpos:= j;
                          |                    fix:= backfix;
                          |                    fixleft:= fixleft0;
                          |                    floatlen:= 0;
                          |                    floatpos:= backfloat + tpos - back;
                          |                    stack:= backstack;
                          |                    ppos:= from + 2 .
                          |
                          |
exhaustnotion             | exhaust notion: IF   notion expansion
                          |                    COR  multi > backmulti
                          |                    CAND no vari
                          |                    CAND notion variation
                          |                    THEN TRUE
                          |                    ELSE backtrack; FALSE
                          |                    FI.
                          |
                          |
notionexpansion           | notion expansion: j:= 0;
                          |                    multi:= last multi;
                          |                    vari:= last vari;
                          |                    WHILE skipcount = 0
                          |                    REP skip and try PER;
                          |                    j:= skipcount;
                          |                    skipcount:= 0;
                          |                    j = 0.
                          |
                          |
skipandtry                | skip and try:      backtrack;
                          |                    j INCR 1;
                          |                    skipcount:=j;
                          |                    ppos:= from + 2;
                          |                    IF   find pattern
                          |                    THEN IF    is notion (t, fixleft)
                          |                         THEN  LEAVE find notion WITH TRUE
                          |                         FI
                          |                    ELSE next; LEAVE find notion WITH FALSE
                          |                    FI .
                          |
                          |
notionvariation           | notion variation: no vari:= FALSE;
                          |                    last multi:= multi;
                          |                    last vari:= vari;
                          |                    FOR k FROM 1 UPTO (multi+1) DIV (backmulti+1) - 1
                          |                    REP backtrack with variation;
                          |                        IF   find first subpattern
                          |                        THEN no vari:= TRUE;
                          |                             LEAVE find notion WITH TRUE
                          |                        FI
                          |                    PER;
                          |                    no vari:= TRUE;
                          |                    FALSE.
                          |
findfull                 | find full:
                         |   find pattern CAND (end of line COR exhaust line).
                         |
                         |
endofline                | end of line:
                         |   next;
                         |   IF   fix
                         |   THEN tpos = tlen
                         |   ELSE tpos:= tlen;
                         |        make fix (1);
                         |        TRUE
                         |   FI.
                         |
                         |
exhaustline              | exhaust line:
                         |   IF   full expansion COR multi > 0 CAND no vari CAND full
                         |        variation
                         |   THEN TRUE ELSE backtrack;
                         |        FALSE
                         |   FI.
                         |
                         |
fullexpansion            | full expansion:
                         |    j:=0;
                         |    last multi:= multi;
                         |    last vari:= vari;
                         |    WHILE skipcount = 0
                         |    REP   IF tlen = tpos
                         |          THEN LEAVE full expansion WITH FALSE
                         |          FI;
                         |          backtrack;
                         |          j INCR 1;
                         |          skipcount:= j;
                         |          ppos:=from + 2
                         |    UNTIL find pattern CAND tpos=tlen
                         |    PER;
                         |    j:= skipcount;
                         |    skipcount:=0;
                         |    j=0.
                         |
                         |
fullvariation            | full variation:
                         |   no vari:= FALSE;
                         |   multi:= last multi;
                         |   vari:= last vari;
                         |   FOR k FROM 1 UPTO multi
                         |   REP backtrack with variation;
                         |       IF   find first subpattern
                         |       THEN no vari:= TRUE;
                         |            LEAVE find WITH TRUE
                         |       FI
                         |   PER;
                         |   no vari:= TRUE;
                         |   FALSE.
                         |
                         | ENDPROC find;
                         |


isnotion ................|BOOL PROC is notion (TEXT CONST t, INT CONST fixleft):
                         |    ppos INCR 2;
                         |    (    NOT  fix
                             |          COR  tpos = tlen
                             |          COR  pos (delimiter, t SUB tpos)   > 0
                             |          COR  pos (delimiter, t SUB tpos-1) > 0
                             |          COR  (t SUB tpos)  <= "Z"
                             |          CAND (t SUB tpos-1) > "Z"   )
                             |     CAND (    fixleft <= 1
                             |          COR  pos (delimiter, t SUB fixleft-1) > 0
                             |          COR  pos (delimiter, t SUB fixleft)   > 0
                             |          COR  (t SUB fixleft)    > "Z"
                             |          CAND (t SUB fixleft-1) <= "Z"   )
                             |
                             | END PROC is notion;
                             |


makefix .....................|PROC make fix (INT CONST back):
                             |     WHILE stack not empty
                             |     REP  INT VAR reg:= code (stack SUB top),
                             |                  pos:= code (stack SUB top-1),
                             |                  len:= code (stack SUB top-3),
                             |                  dis:= code (stack SUB top-2) - floatpos;
                             |          maend(reg):= min (tpos + dis, tlen);              (* 6.6.
                             |               *)
                             |          mapos(reg):= pos or fix or float;
                             |          stack:= subtext (stack,1,top-4)
                             |     PER;
                             |     fix:= TRUE;
                             |     floatpos:= 0 .
                             |
                             |
   stacknotempty             |     stack not empty: INT VAR top:= LENGTH stack;
                             |                      top > 0.
                             |
                             |
   posorfixorfloat           |     pos or fix or float:
                             |          IF   pos = undefined
                             |          THEN IF len = 0
                             |               THEN min (back + dis, tlen)
                             |               ELSE maend(reg) - len
                             |               FI
                             |          ELSE pos
                             |          FI.
                             |
                             | ENDPROC make fix;
                             |


inalpha .....................|BOOL PROC in alpha (TEXT CONST t, INT CONST from, to):
                             |     FOR failpos FROM from UPTO to - 1
                             |     REP IF   pos (alphabet, t SUB failpos) = 0
                             |         THEN LEAVE in alpha WITH FALSE
                             |         FI
                             |     PER;
                             |     TRUE
                             |  ENDPROC in alpha;
                             |


notion ......................|TEXT PROC notion (TEXT CONST t, INT CONST r): notion (t) ** r
                             |     ENDPROC notion;
                             |
                             |ENDPACKET pattern match;
```

```
                            |(* -------------------- VERSION 35     02.06.86 -------------------- *)
filehandling ***************|PACKET file handling DEFINES      (* Autoren: J.Liedtke, D.Martinek *)
                            |       (***********)
                            |
                            |       FILE,
                            |       :=,
                            |       sequential file,
                            |       reorganize,
                            |       input,
                            |       output,
                            |       modify,
                            |       close,
                            |       putline,
                            |       getline,
                            |       put,
                            |       get,
                            |       write ,
                            |       line,
                            |       reset,
                            |       down,
                            |       up,
                            |       downety,
                            |       uppety,
                            |       pattern found,
                            |       to first record,
                            |       to line,
                            |       to eof,
                            |       insert record,
                            |       delete record,
                            |       read record,
                            |       write record,
                            |       is first record,
                            |       eof,
                            |       line no,
                            |       FRANGE,
                            |       set range,
                            |       reset range ,
                            |       remove,
                            |       clear removed,
                            |       reinsert,
                            |       max line length,
                            |       edit info,
                            |       line type ,
                            |       copy attributes ,
                            |       headline,
                            |       put tabs,
                            |       get tabs,
                            |       col,
                            |       word,
                            |       at,
                            |       removed lines,
                            |       exec,
                            |       pos ,
                            |       len ,
                            |       subtext ,
                            |       change ,
                            |       lines ,
                            |       segments ,
                            |       mark ,
                            |       mark line no ,
                            |       mark col ,
                            |       set marked range ,
|       split line ,
|       concatenate line ,
|       prefix ,
|       sort ,
|       lexsort :
|
|
|(**********************************************************************
|     **)
|(*
|      *)
|(*  Terminologie:
|      *)
|(*
|      *)
|(*
|      *)
|(*    ATOMROW        Menge aller Atome eines FILEs.
|      *)
|(*                   Die einzelnen Atome haben zwar eine Position
|      *)
|(*                   im Row, aber in dieser Betrachtung keine
|      *)
|(*                   logische Reihenfolge.
|      *)
|(*
|      *)
|(*    ATOM           Basiselement, kann eine Zeile der Datei und die
|      *)
|(*                   zugehoerige Verwaltungsinformation aufnehmen
|      *)
|(*
|      *)
|(*    CHAIN          Zyklisch geschlossene Kette von Segmenten.
|      *)
|(*
|      *)
|(*    SEGMENT        Teilbereich des Atomrows, enthaelt 1 oder mehr
|      *)
|(*                   zusammenhaengende Atoms.
|      *)
|(*                   Jedes Segment hat ein Vorgaenger- und ein
|      *)
|(*                   Nachfolgersegment.
|      *)
|(*                   Jedes Segment enthaelt einen logisch zumsammen-
|      *)
|(*                   haengenden Teile einer Sequence.
|      *)
|(*
|      *)
|(*    SEQUENCE       Logische Folge von Lines.
|      *)
|(*                   Jede Sequence ist Teil einer Chain oder besteht
|      *)
|(*                   vollstaendig daraus:
|      *)
|(*
|      *)
|(*                        SEG1--SEG2--SEG3--SEG4--SEG5
|      *)
|(*                          :----sequence----:
|      *)
```

```
|(*
|      *)
|(*                   Die 'Reihenfolge' ebenso wie die 'Anzahl' der
|      *)
|(*                   Lines ist eine wesentliche Eigenschaft einer
|      *)
|(*                   Sequence.
|      *)
|(*
|      *)
|(*   LINE            Ein Atom als Element ein Sequence betrachtet.
|      *)
|(*
|      *)
|(*
|      *)
|(*********************************************************************
|     **)
|(*
|      *)
|(* Eigenschaften:
|      *)
|(*
|      *)
|(*   Folgende Mengen bilden eine Zerlegung (im math. Sinn) einer
|      *)
|(*   gesamten Datei:
|      *)
|(*                        used segment chain
|      *)
|(*                        scratch segment chain
|      *)
|(*                        free segment chain
|      *)
|(*                        unused tail
|      *)
|(*
|      *)
|(*   Fuer jedes X aus (used, scratch, free) gelten:
|      *)
|(*
|      *)
|(*   'X sequence' ist echte Teilmenge von 'X segment chain'.
|      *)
|(*
|      *)
|(*                       (Daraus folgt, es gibt keine leere
|     'chain'.) *)
|(*
|      *)
|(*   'X segment chain' ist zyklisch gekettet.
|      *)
|(*
|      *)
|(*   Alle Atome von 'X segment chain' haben definierten Inhalt.
|      *)
|(*
|      *)
|(*********************************************************************
|     **)
|
|
                              |LET file size    = 4075 ,
                              |    nil          = 0 ,
                              |
                              |    free root    = 1 ,
                              |    scratch root = 2 ,
                              |    used root    = 3 ,
                              |    first unused = 4 ;
                              |
                              |
                              |LET SEQUENCE = STRUCT (INT index, segment begin, segment end,
                              |                       INT line no, lines),
                              |    SEGMENT  = STRUCT (INT succ, pred, end),
                              |    ATOM     = STRUCT (SEGMENT seg, INT type, TEXT line),
                              |    ATOMROW  = ROW filesize ATOM,
                              |
                              |    LIST     = STRUCT (SEQUENCE used, INT prefix lines, postfix
                              |         lines,
                              |                      SEQUENCE scratch, free, INT unused tail,
                              |                      INT mode, col, limit, edit info, mark line,
                              |                           mark col,
                              |                      ATOMROW atoms);
                              |
                              |TYPE FILE = BOUND LIST ;
                              |
                              |TYPE FRANGE = STRUCT (INT pre, post, BOOL pre was split, post was
                              |     split);
                              |
                              |


:= ...........................|OP := (FRANGE VAR left, FRANGE CONST right):
                              |  CONCR (left) := CONCR (right)
                              |ENDOP := ;
                              |
                              |


:= ...........................|OP := (FILE VAR left, FILE CONST right):
                              |  EXTERNAL 260
                              |END OP :=;
                              |
                              |


becomes ......................|PROC becomes (INT VAR a, b) :
                              |  INTERNAL 260  ;
                              |  a := b
                              |END PROC becomes;
                              |
                              |


initialize ...................|PROC initialize (FILE VAR f) :
                              |
                              |  f.used       := SEQUENCE : (used root, used root, used root, 1, 0);
                              |  f.prefix lines  := 0;
                              |  f.postfix lines := 0;
                              |  f.free       := SEQUENCE : (free root, free root, free root, 1, 0);
                              |  f.scratch    := SEQUENCE : (scratch root, scratch root, scratch
                              |       root, 1, 0);
                              |  f.unused tail := first unused;
                              |
                          |   f.limit := 77;
                          |   f.edit info := 0;
                          |   f.col := 1 ;
                          |   f.mark line := 0 ;
                          |   f.mark col := 0 ;
                          |
                          |   INT VAR i;
                          |   FOR i FROM 1 UPTO 3 REP
                          |     root (i).seg  := SEGMENT : (i, i, i);
                          |     root (i).line := ""
                          |   PER;
                          |   put tabs (f, "") .
                          |
                          |
     root                 |root : f.atoms .
                          |
                          |END PROC initialize;
                          |
                          |
                          |(*********************************************************************
                          |     **)
                          |(*
                          |      *)
                          |(*       Segment Handler        (SEGMENTs & CHAINs)
                          |      *)
                          |(*
                          |      *)
                          |(*********************************************************************
                          |     **)
                          |


   segs ..................|INT PROC segs (SEQUENCE CONST s, ATOMROW CONST atom) :
                          |
                          |  INT VAR number of segments := 0 ,
                          |  actual segment := s.segment begin ;
                          |  REP
                          |    number of segments INCR 1 ;
                          |    actual segment := atom (actual segment).seg.succ
                          |  UNTIL actual segment = s.segment begin PER ;
                          |  number of segments .
                          |
                          |ENDPROC segs ;
                          |
                          |


   nextsegment ...........|PROC next segment (SEQUENCE VAR s, ATOMROW CONST atom) :
                          |
                          |  disable stop;
                          |  s.line no INCR (s.segment end - s.index + 1);
                          |  INT CONST new segment index := actual segment.succ;
                          |  s.segment begin := new segment index;
                          |  s.segment end   := new segment.end;
                          |  s.index         := new segment index .
                          |
                          |
     actualsegment        |actual segment :  atom (s.segment begin).seg .
                          |
     newsegment           |new segment :     atom (new segment index).seg .
                          |
                          |END PROC next segment;
```

```
                             |
                             |

previoussegment ..........|PROC previous segment (SEQUENCE VAR s, ATOMROW CONST atom) :
                             |
                             |  disable stop;
                             |  s.line no DECR (s.index - s.segment begin + 1);
                             |  INT CONST new segment index := actual segment.pred;
                             |  s.segment begin := new segment index;
                             |  s.segment end   := new segment.end;
                             |  s.index         := s.segment end .
                             |
                             |
   actualsegment             |actual segment :  atom (s.segment begin).seg .
                             |
   newsegment                |new segment :     atom (new segment index).seg .
                             |
                             |END PROC previous segment;
                             |
                             |

splitsegment .............|PROC split segment (SEQUENCE VAR s, ATOMROW VAR atom) :
                             |
                             |  disable stop;
                             |  IF not at segment top
                             |    THEN split segment at actual position
                             |  FI .
                             |
                             |
   splitsegmentatactualpo    |split segment at actual position :
                             |  INT CONST pred index   := s.segment begin,
                             |            actual index := s.index,
                             |            succ index   := pred.succ;
                             |
                             |  actual.pred  := pred index;
                             |  actual.succ  := succ index;
                             |  actual.end   := s.segment end;
                             |
                             |  pred.succ    := actual index;
                             |  pred.end     := actual index - 1;
                             |
                             |  succ.pred    := actual index;
                             |
                             |  s.segment begin := actual index .
                             |
                             |
   notatsegmenttop           |not at segment top : s.index > s.segment begin .
                             |
                             |
   pred                      |pred   : atom (pred index).seg .
                             |
                             |
   actual                    |actual : atom (actual index).seg .
                             |
                             |
   succ                      |succ   : atom (succ index).seg .
                             |
                             |END PROC split segment;
                             |
                             |
joinsegments ............|PROC join segments (ATOMROW VAR atom,
                         |                      INT CONST first index, INT VAR second index) :
                         |
                         |  disable stop;
                         |  IF first seg.end + 1 = second index
                         |    THEN attach second to first segment
                         |    ELSE link first to second segment
                         |  FI .
                         |
                         |
   attachsecondtofirstseg |attach second to first segment :
                         |  first seg.end  := second seg.end;
                         |  INT VAR successor of second := second seg.succ;
                         |  IF successor of second = second index
                         |    THEN first seg.succ := first index
                         |    ELSE join segments (atom, first index, successor of second)
                         |  FI;
                         |  second index := first index .
                         |
                         |
   linkfirsttosecondsegme |link first to second segment :
                         |  first seg.succ  := second index;
                         |  second seg.pred := first index .
                         |
                         |
   firstseg              |first seg :  atom (first index).seg .
                         |
   secondseg             |second seg : atom (second index).seg .
                         |
                         |END PROC join segments;
                         |
                         |


deletesegments ..........|PROC delete segments (SEQUENCE VAR from, ATOMROW VAR atom,
                         |                        INT CONST first index, last index, lines) :
                         |
                         |  determine surrounding segments and new atom index;
                         |  join surrounding segments;
                         |  update sequence descriptor .
                         |
                         |
   determinesurroundingse |determine surrounding segments and new atom index :
                         |  INT VAR pred index   := first seg.pred,
                         |          actual index := last seg.succ;
                         |  from.index := actual index .
                         |
                         |
   joinsurroundingsegment |join surrounding segments :
                         |  join segments (atom, pred index, actual index) .
                         |
                         |
   updatesequencedescript |update sequence descriptor :
                         |  from.segment begin := actual index;
                         |  from.segment end := actual seg.end;
                         |  from.lines DECR lines .
                         |
                         |
   actualseg             |actual seg :  atom (actual index).seg .
```

```
    firstseg               |first seg :   atom (first index).seg .
                           |
    lastseg                |last seg :    atom (last index).seg .
                           |
                           |END PROC delete segments;
                           |
                           |


insertsegments ...........|PROC insert segments (SEQUENCE VAR into, ATOMROW VAR atom,
                           |                          INT CONST first index, last index, lines) :
                           |
                           |  join into sequence and new segments;
                           |  update sequence descriptor .
                           |
                           |
    joinintosequenceandnew |join into sequence and new segments :
                           |  INT VAR actual index := into.index,
                           |          pred index   := actual seg.pred;
                           |  join segments (atom, last index, actual index);
                           |  actual index := first index;
                           |  join segments (atom, pred index, actual index) .
                           |
                           |
    updatesequencedescript |update sequence descriptor :
                           |  into.index := first index;
                           |  into.segment begin := actual index;
                           |  into.segment end := actual seg.end;
                           |  into.lines INCR lines .
                           |
                           |
    actualseg              |actual seg :  atom (actual index).seg .
                           |
                           |END PROC insert segments;
                           |
                           |


nextatom .................|PROC next atom (SEQUENCE VAR s, ATOMROW CONST atom) :
                           |
                           |  IF s.line no <= s.lines
                           |    THEN to next atom
                           |    ELSE errorstop ("'down' nach Dateiende")
                           |  FI .
                           |
                           |
    tonextatom             |to next atom :
                           |  disable stop;
                           |  IF s.index = s.segment end
                           |    THEN next segment (s, atom)
                           |    ELSE s.index INCR 1;
                           |         s.line no INCR 1
                           |  FI
                           |
                           |END PROC next atom;
                           |
                           |


nextatoms ................|PROC next atoms (SEQUENCE VAR s, ATOMROW CONST atom, INT CONST
                           |     times) :
                           |
                             |  INT CONST destination line := min (s.line no + times, s.lines + 1)
                             |  jump upto destination segment;
                             |  position within destination segment .
                             |
                             |
    jumpuptodestinationseg   |jump upto destination segment :
                             |  WHILE s.line no + length of actual segments tail < destination
                             |       line REP
                             |    next segment (s, atom);
                             |  PER .
                             |
                             |
    positionwithindestinat   |position within destination segment :
                             |  disable stop;
                             |  s.index INCR (destination line - s.line no);
                             |  s.line no := destination line .
                             |
                             |
    lengthofactualsegments   |length of actual segments tail :   s.segment end - s.index .
                             |
                             |END PROC next atoms;
                             |
                             |


previousatom ............|PROC previous atom (SEQUENCE VAR s, ATOMROW CONST atom) :
                             |
                             |  IF s.line no > 1
                             |    THEN to previous atom
                             |    ELSE errorstop ("'up' am Dateianfang")
                             |  FI .
                             |
                             |
    topreviousatom           |to previous atom :
                             |  disable stop;
                             |  IF s.index = s.segment begin
                             |    THEN previous segment (s, atom)
                             |    ELSE s.index DECR 1;
                             |         s.line no DECR 1
                             |  FI
                             |
                             |END PROC previous atom;
                             |
                             |


previousatoms ...........|PROC previous atoms (SEQUENCE VAR s, ATOMROW CONST atom, INT CONST
                             |     times) :
                             |
                             |  INT CONST destination line := max (1, s.line no - times);
                             |  jump back to destination segment;
                             |  position within destination segment .
                             |
                             |
    jumpbacktodestinations   |jump back to destination segment :
                             |  WHILE s.line no - length of actual segments head > destination
                             |       line REP
                             |    previous segment (s, atom);
                             |  PER .
                             |
   positionwithindestinat |position within destination segment :
                          |  disable stop;
                          |  s.index DECR (s.line no - destination line);
                          |  s.line no := destination line .
                          |
                          |
   lengthofactualsegments |length of actual segments head :   s.index - s.segment begin .
                          |
                          |END PROC previous atoms;
                          |
                          |
                          |TEXT VAR pre, pat, pattern0;
                          |INT VAR last search line ;
                          |


searchdown ...............|PROC search down (SEQUENCE VAR s, ATOMROW CONST atom, TEXT CONST
                          |     pattern,
                          |                    INT CONST max lines, INT VAR column) :
                          |
                          |  INT CONST start col := column ,
                          |            start line := s.lineno ;
                          |  last search line := min (s.lines, s.lineno + max lines) ;
                          |  pre:= somefix (pattern) ;
                          |  pattern0 := pattern ** 0 ;
                          |  down in atoms (s, atom, pre, column);
                          |  IF NOT (last search succeeded CAND like pattern)
                          |    THEN try again
                          |  FI;
                          |  last search succeeded := TRUE ;
                          |  column := matchpos (0) .
                          |
                          |
   tryagain               |try again:
                          |  WHILE s.line no < last search line
                          |    REP next atom (s, atom) ;
                          |        column := 1 ;
                          |        down in atoms (s, atom, pre, column);
                          |        IF last search succeeded CAND like pattern
                          |          THEN LEAVE try again
                          |        FI
                          |    PER;
                          |    column := 1 + LENGTH record;
                          |    last search succeeded := FALSE ;
                          |    LEAVE search down.
                          |
                          |
   likepattern            |like pattern :
                          |  correct position ;
                          |  pat := any (column-1) ;
                          |  pat CAT any ;
                          |  pat CAT pattern0 ;
                          |  pat CAT any ;
                          |  record LIKE pat .
                          |
                          |
   correctposition        |correct position :
                          |  IF s.lineno = start line
                          |    THEN column := start col
                          |    ELSE column := 1
                          |  FI .
                          |
    record                |record : atom (s.index).line .
                          |
                          |ENDPROC search down ;
                          |

 downinatoms .............|PROC down in atoms (SEQUENCE VAR s, ATOMROW CONST atom, TEXT CONST
                          |     pattern,
                          |                    INT VAR column) :
                          |
                          |  last search succeeded := FALSE ;
                          |  search forwards in actual line ;
                          |  IF NOT found AND s.line no < last search line
                          |    THEN search in following lines
                          |  FI ;
                          |  IF found
                          |    THEN last search succeeded := TRUE
                          |    ELSE set column behind last char
                          |  FI .
                          |
                          |
     setcolumnbehindlastcha |set column behind last char :
                          |  column := LENGTH atom (s.index).line + 1 .
                          |
                          |
     searchforwardsinactual |search forwards in actual line :
                          |  IF pattern <> ""
                          |    THEN column := pos (atom (s.index).line, pattern, column)
                          |  ELIF column > LENGTH atom (s.index).line
                          |    THEN column := 0
                          |  FI .
                          |
                          |
     searchinfollowinglines |search in following lines :
                          |  next atom (s, atom) ;
                          |  IF pattern = ""
                          |    THEN column := 1 ;
                          |         LEAVE search in following lines
                          |  FI ;
                          |  REP
                          |    search forwards through segment ;
                          |    update file position forwards ;
                          |    IF found OR s.line no = last search line
                          |      THEN LEAVE search in following lines
                          |      ELSE next segment (s, atom)
                          |    FI
                          |  PER .
                          |
                          |
     searchforwardsthroughs |search forwards through segment :
                          |  INT VAR search index := s.index ,
                          |  last index := min (s.segment end, s.index+(last search line-s.line
                          |     no));
                          |  REP
                          |    column := pos (atom (search index).line, pattern) ;
                          |    IF found OR search index = last index
                          |      THEN LEAVE search forwards through segment
                          |    FI ;
                          |    search index INCR 1
                          |  PER .
                          |
    updatefilepositionforw |update file position forwards :
                           |  disable stop ;
                           |  s.line no INCR (search index - s.index) ;
                           |  s.index := search index ;
                           |  enable stop .
                           |
                           |
    found                  |found :  column > 0 .
                           |
                           |ENDPROC down in atoms ;
                           |


 prefix ...................|TEXT PROC prefix (TEXT CONST pattern) :
                           |
                           |  INT VAR invalid char pos := pos (pattern, ""0"", ""31"", 1) ;
                           |  SELECT invalid char pos OF
                           |    CASE 0     : pattern
                           |    CASE 1     : ""
                           |    OTHERWISE : subtext (pattern, 1, invalid char pos - 1)
                           |  ENDSELECT .
                           |
                           |ENDPROC prefix ;
                           |


 searchup .................|PROC search up (SEQUENCE VAR s, ATOMROW CONST atom, TEXT CONST
                           |     pattern,
                           |                 INT CONST max lines, INT VAR column) :
                           |
                           |     last search line := max (1, s.lineno - max lines) ;
                           |     pre:= prefix (pattern);
                           |     pattern0 := pattern ** 0;
                           |     remember start point ;
                           |     up in atoms (s, atom, pre, column);
                           |     IF   NOT (last search succeeded CAND last pattern in line found)
                           |     THEN try again
                           |     FI;
                           |     last search succeeded := TRUE ;
                           |     column := matchpos (0) .
                           |
                           |
    tryagain               |  try again:
                           |     WHILE s.lineno > last search line OR column > 1
                           |     REP   previous atom (s, atom);
                           |           column := LENGTH record ;
                           |           up in atoms (s, atom, pre, column);
                           |           IF last search succeeded CAND last pattern in line found
                           |             THEN LEAVE try again
                           |           FI
                           |     PER;
                           |     column := 1;
                           |     last search succeeded := FALSE ;
                           |     LEAVE search up.
                           |
                           |
    rememberstartpoint     |  remember start point :
                           |     INT VAR c:= column, r:= s.lineno;.
                           |
     lastpatterninlinefound |  last pattern in line found :
                            |    column := 2 ;
                            |    WHILE like pattern CAND right of start REP
                            |      column := matchpos (0) +1
                            |    PER ;
                            |    column DECR 1 ;
                            |    like pattern CAND right of start .
                            |
                            |
     likepattern            |  like pattern :
                            |     pat := any (column-1) ;
                            |     pat CAT any ;
                            |     pat CAT pattern0 ;
                            |     pat CAT any ;
                            |     record LIKE pat .
                            |
                            |
     rightofstart           |  right of start : (r > s.lineno COR c >= matchpos(0)) .
                            |
     record                 |  record         : atom (s.index).line .
                            |
                            |ENDPROC search up ;
                            |


  upinatoms ................|PROC up in atoms (SEQUENCE VAR s, ATOMROW CONST atom, TEXT CONST
                            |     pattern,
                            |                    INT VAR column) :
                            |
                            |  last search succeeded := FALSE ;
                            |  search backwards in actual line ;
                            |  IF NOT found AND s.line no > last search line
                            |    THEN search in preceeding lines
                            |  FI ;
                            |  IF found
                            |    THEN last search succeeded := TRUE
                            |    ELSE column := 1
                            |  FI .
                            |
                            |
     searchbackwardsinactua |search backwards in actual line :
                            |  IF pattern = ""
                            |    THEN LEAVE search backwards in actual line
                            |  FI ;
                            |  INT VAR last pos , new pos := 0 ;
                            |  REP
                            |    last pos := new pos ;
                            |    new pos := pos (atom (s.index).line, pattern, last pos+1) ;
                            |  UNTIL new pos = 0 OR new pos > column PER ;
                            |  column := last pos .
                            |
                            |
     searchinpreceedingline |search in preceeding lines :
                            |  previous atom (s, atom) ;
                            |  IF pattern = ""
                            |    THEN column := LENGTH atom (s.index).line + 1 ;
                            |         last search succeeded := TRUE ;
                            |         LEAVE search in preceeding lines
                            |  FI ;
                            |  REP
                            |    search backwards through segment ;
                            |    update file position backwards ;
```

```
                              |    IF found OR s.line no = last search line
                              |      THEN LEAVE search in preceeding lines
                              |      ELSE previous segment (s, atom)
                              |    FI
                              |  PER .
                              |
                              |
    searchbackwardsthrough    |search backwards through segment :
                              |  INT VAR search index := s.index ,
                              |  last index := max (s.segment begin, s.index-(s.line no-last search
                              |       line));
                              |  REP
                              |    new pos := 0 ;
                              |    REP
                              |      column := new pos ;
                              |      new pos := pos (atom (search index).line, pattern, column+1) ;
                              |    UNTIL new pos = 0 PER ;
                              |    IF found OR search index = last index
                              |      THEN LEAVE search backwards through segment
                              |    FI ;
                              |    search index DECR 1
                              |  PER .
                              |
                              |
    updatefilepositionback    |update file position backwards :
                              |  disable stop ;
                              |  s.line no DECR (s.index - search index) ;
                              |  s.index := search index ;
                              |  enable stop .
                              |
                              |
    found                     |found :  column > 0 .
                              |
                              |ENDPROC up in atoms ;
                              |
                              |BOOL VAR last search succeeded ;
                              |


 patternfound ............... |BOOL PROC pattern found :
                              |  last search succeeded
                              |ENDPROC pattern found ;
                              |
                              |
                              |


 deleteatom ................. |PROC delete atom (SEQUENCE VAR used, free, ATOMROW VAR atom) :
                              |
                              |  disable stop;
                              |  IF used.line no <= used.lines
                              |    THEN delete actual atom
                              |    ELSE errorstop ("'delete' am Dateiende")
                              |  FI .
                              |
                              |
    deleteactualatom          |delete actual atom :
                              |  position behind actual free segment;
                              |  split segment (used, atom);
                              |  INT VAR actual index := used.index;
                              |  cut off tail of actual used segment;
                              |  delete segments (used, atom, actual index, actual index, 1);
```

```
                         | insert segments (free, atom, actual index, actual index, 1) .
                         |
                         |
   positionbehindactualfr |position behind actual free segment :
                         | IF free.line no <= free.lines
                         |   THEN next segment (free, atom)
                         | FI .
                         |
                         |
   cutofftailofactualused |cut off tail of actual used segment :
                         | IF actual index <> used.segment end
                         |   THEN used.index INCR 1;
                         |        split segment (used, atom);
                         |        used.index DECR 1
                         | FI .
                         |
                         |END PROC delete atom;
                         |
                         |


insertatom ..............|PROC insert atom (SEQUENCE VAR used, free,INT VAR unused, ATOMROW
                         |     VAR atom) :
                         |
                         | disable stop;
                         | split segment (used, atom);
                         | IF free.lines > 0
                         |   THEN insert new atom from free sequence
                         | ELIF unused <= file size
                         |   THEN insert new atom from unused tail
                         | ELSE   errorstop ("FILE-Ueberlauf")
                         | FI .
                         |
                         |
   insertnewatomfromfrees |insert new atom from free sequence :
                         | get a free segments head;
                         | make this atom to actual segment;
                         | transfer from free to used chain .
                         |
                         |
   getafreesegmentshead  |get a free segments head :
                         | IF actual free segment is root segment
                         |   THEN previous segment (free, atom)
                         | FI;
                         | position to actual segments head .
                         |
                         |
   positiontoactualsegmen |position to actual segments head :
                         | INT VAR actual index := free.segment begin;
                         | free.line no DECR (free.index - actual index);
                         | free.index := actual index .
                         |
                         |
   makethisatomtoactualse |make this atom to actual segment :
                         | IF free.segment end > actual index
                         |   THEN free.index INCR 1;
                         |        split segment (free, atom);
                         |        free.index DECR 1
                         | FI .
                         |
    transferfromfreetoused |transfer from free to used chain :
                           |  delete segments (free, atom, actual index, actual index, 1);
                           |  insert segments (used, atom, actual index, actual index, 1);
                           |  atom (actual index).line := "" .
                           |
                           |
    insertnewatomfromunuse |insert new atom from unused tail :
                           |  actual index := unused;
                           |  atom (actual index).seg :=
                           |                       SEGMENT:(actual index, actual index, actual
                           |                            index);
                           |  atom (actual index).line := "";
                           |  insert segments (used, atom, actual index, actual index, 1);
                           |  unused INCR 1 .
                           |
                           |
    actualfreesegmentisroo |actual free segment is root segment :    free.segment begin = free
                           |     root .
                           |
                           |END PROC insert atom;
                           |
                           |


  insertnext ..............|PROC insert next (SEQUENCE VAR used, free, INT VAR unused, ATOMROW
                           |     VAR atom,
                           |                   TEXT CONST record) :
                           |
                           |  IF used.line no > used.lines
                           |    THEN insert atom (used, free, unused, atom)
                           |  ELIF actual position before unused nonempty atomrow part
                           |    THEN forward and insert atom by simple extension of used atomrow
                           |         part
                           |    ELSE next atom (used, atom);
                           |         insert atom (used, free, unused, atom)
                           |  FI;
                           |  atom (used.index).line := record .
                           |
                           |
    forwardandinsertatomby |forward and insert atom by simple extension of used atomrow part :
                           |  used.line no INCR 1;
                           |  used.lines INCR 1;
                           |  used.index INCR 1;
                           |  used.segment end INCR 1;
                           |  atom (used.segment begin).seg.end INCR 1;
                           |  unused INCR 1 .
                           |
                           |
    actualpositionbeforeun |actual position before unused nonempty atomrow part :
                           |  used.index = unused - 1 AND unused part not empty .
                           |
                           |
    unusedpartnotempty     |unused part not empty :  unused <= file size .
                           |
                           |END PROC insert next;
                           |
                           |


  transfersubsequence .....|PROC transfer subsequence (SEQUENCE VAR source, dest,
                           |                              ATOMROW VAR atom, INT CONST size) :
                           |
                       |  IF size > 0
                       |    THEN INT VAR subsequence size := min (size, source.line no);
                       |         mark begin of source part;
                       |         mark end of source part;
                       |         split destination sequence;
                       |         transfer part
                       |  FI .
                       |
                       |
markbeginofsourcepart  |mark begin of source part :
                       |  previous atoms (source, atom, subsequence size - 1);
                       |  split segment (source, atom);
                       |  INT CONST first := source.segment begin .
                       |
                       |
markendofsourcepart    |mark end of source part :
                       |  next atoms (source, atom, subsequence size - 1);
                       |  INT CONST last := source.segment begin;
                       |  next atom (source, atom);
                       |  split segment (source, atom) .
                       |
                       |
splitdestinationsequen |split destination sequence :
                       |  split segment (dest, atom) .
                       |
                       |
transferpart           |transfer part :
                       |  disable stop;
                       |  delete segments (source, atom, first, last, subsequence size);
                       |  source.line no DECR subsequence size;
                       |  insert segments (dest, atom, first, last, subsequence size);
                       |  next atoms (dest, atom, subsequence size - 1) .
                       |
                       |END PROC transfer subsequence;
                       |
                       |
                       |
                       |(*********************************************************************
                       |     )
                       |(*****
                       |     *****)
                       |(*****                    FILE handler
                       |     *****)
                       |(*****
                       |     *****)
                       |(*********************************************************************
                       |     )
                       |
                       |
                       |
                       |LET file type    = 1003 ,
                       |    file type 16 = 1002 ,
                       |
                       |    closed       = 0,
                       |    inp          = 1,
                       |    outp         = 2,
                       |    mod          = 3,
                       |    end          = 4,
                       |
                       |    max limit    = 16000,
                       |    super limit  = 16001;
                       |
                               |
                               |TYPE TRANSPUTDIRECTION = INT;
                               |
                               |

input .........................|TRANSPUTDIRECTION PROC input :
                               |  TRANSPUTDIRECTION : (inp)
                               |END PROC input;
                               |
                               |

output ........................|TRANSPUTDIRECTION PROC output :
                               |  TRANSPUTDIRECTION : (outp)
                               |END PROC output;
                               |
                               |

modify ........................|TRANSPUTDIRECTION PROC modify :
                               |  TRANSPUTDIRECTION : (mod)
                               |END PROC modify;
                               |
                               |
                               |FILE VAR result file;
                               |
                               |

sequentialfile ................|FILE PROC sequential file (TRANSPUTDIRECTION CONST mode,
                               |                           DATASPACE CONST ds) :
                               |  IF   type (ds) = file type
                               |  THEN result := ds
                               |  ELIF type (ds) < 0
                               |  THEN result := ds; type (ds, file type); initialize (result file)
                               |  ELSE enable stop; errorstop ("Datenraum hat falschen Typ")
                               |  FI;
                               |  reset (result file, mode);
                               |  result file .
                               |
                               |
   result                      |result : CONCR (result file) .
                               |
                               |END PROC sequential file;
                               |
                               |

sequentialfile ................|FILE PROC sequential file (TRANSPUTDIRECTION CONST mode, TEXT CONST
   +                           |     name) :
                               |
                               |  IF   exists (name)
                               |  THEN get dataspace if file
                               |  ELIF CONCR (mode) <> inp
                               |  THEN get new file space
                               |  ELSE errorstop (""""+name+""" gibt es nicht") ; enable stop
                               |  FI;
                               |  update status if necessary;
                               |  reset (result file, mode);
                               |  result file .
                               |
     getdataspaceiffile     |get dataspace if file :
                            |  IF type (old (name)) = file type 16
                            |    THEN reorganize (name)
                            |  FI ;
                            |  result := old (name, file type) ;
                            |  IF is 170 file
                            |    THEN result.col := 1 ;
                            |         result.mark line := 0 ;
                            |         result.mark col  := 0
                            |  FI .
                            |
                            |
     is170file              |is 170 file : result.mark col < 0 .
                            |
                            |
     getnewfilespace        |get new file space :
                            |  result := new (name);
                            |  IF   NOT is error
                            |  THEN type (old (name), file type); initialize (result file)
                            |  FI .
                            |
                            |
     updatestatusifnecessar |update status if necessary :
                            |  IF   CONCR (mode) <> inp
                            |  THEN status (name, ""); headline (result file, name)
                            |  FI .
                            |
                            |
     result                 |result : CONCR (result file) .
                            |
                            |END PROC sequential file;
                            |
                            |


  reset ....................|PROC reset (FILE VAR f) :
                            |
                            |  IF f.mode = end
                            |    THEN reset (f, input)
                            |    ELSE reset (f, TRANSPUTDIRECTION:(f.mode))
                            |  FI .
                            |
                            |ENDPROC reset ;
                            |


  reset ....................|PROC reset (FILE VAR f, TRANSPUTDIRECTION CONST mode) :
                            |
                            |  IF f.mode <> mod OR new mode <> mod
                            |    THEN f.mode := new mode ;
                            |         initialize file index
                            |  FI .
                            |
                            |
     initializefileindex    |initialize file index :
                            |  IF new mode = outp
                            |    THEN to line without check (f, f.used.lines);
                            |         col := super limit
                            |    ELSE to line without check (f, 1);
                            |         col := 1 ;
                            |         IF new mode = inp AND file is empty
                            |           THEN f.mode := end
```

```
                               |         FI
                               |  FI .
                               |
                               |
   fileisempty                 |file is empty : f.used.lines = 0 .
                               |
                               |
   newmode                     |new mode  : CONCR (mode) .
                               |
                               |
   col                         |col :   CONCR (CONCR (f)).col .
                               |
                               |END PROC reset;
                               |
                               |


input .........................|PROC input (FILE VAR f) :
                               |
                               |  reset (f, input) .
                               |
                               |END PROC input;
                               |
                               |


output ........................|PROC output (FILE VAR f) :
                               |
                               |  reset (f, output)
                               |
                               |END PROC output;
                               |
                               |


modify ........................|PROC modify (FILE VAR f) :
                               |
                               |  reset (f, modify)
                               |
                               |END PROC modify;
                               |
                               |


close .........................|PROC close (FILE VAR f) :
                               |
                               |  f.mode := closed .
                               |
                               |END PROC close;
                               |
                               |


checkmode .....................|PROC check mode (FILE CONST f, INT CONST mode) :
                               |
                               |  IF f.mode = mode
                               |    THEN LEAVE check mode
                               |    ELIF f.mode = closed
                               |    THEN errorstop ("Datei zu!")
                               |    ELIF f.mode = mod
                               |    THEN errorstop ("unzulaessiger Zugriff auf modify-FILE")
                               |    ELIF mode = mod
```

```
                                  |    THEN errorstop ("Zugriff nur auf modify-FILE zulaessig")
                                  |    ELIF f.mode = end
                                  |    THEN errorstop ("Leseversuch nach Dateiende")
                                  |    ELIF mode = inp
                                  |    THEN errorstop ("Leseversuch auf output-FILE")
                                  |    ELIF mode = outp
                                  |    THEN errorstop ("Schreibversuch auf input-FILE")
                                  |  FI .
                                  |
                                  |END PROC check mode;
                                  |
                                  |


tolinewithoutcheck .......|PROC to line without check (FILE VAR f, INT CONST destination line)
                                  |
                                  |  INT CONST distance := destination line - f.used.line no;
                                  |  IF   distance > 0
                                  |    THEN next atoms (f.used, f.atoms, distance)
                                  |  ELIF distance < 0
                                  |    THEN previous atoms (f.used, f.atoms, - distance)
                                  |  FI .
                                  |
                                  |END PROC to line without check;
                                  |
                                  |


toline ...................|PROC to line (FILE VAR f, INT CONST destination line) :
                                  |
                                  |  check mode (f, mod);
                                  |  to line without check (f, destination line)
                                  |
                                  |END PROC to line;
                                  |
                                  |


tofirstrecord ............|PROC to first record (FILE VAR f) :
                                  |
                                  |  to line (f, 1)
                                  |
                                  |END PROC to first record;
                                  |
                                  |


toeof ....................|PROC to eof (FILE VAR f) :
                                  |
                                  |  to line (f, f.used.lines + 1) .
                                  |
                                  |END PROC to eof;
                                  |
                                  |


putline ..................|PROC putline (FILE VAR f, TEXT CONST word) :
                                  |
                                  |  write (f, word);
                                  |  col := super limit .
                                  |
   col                      |col : CONCR (CONCR (f)).col .
                            |
                            |END PROC putline;
                            |
                            |

deleterecord ...............|PROC delete record (FILE VAR f) :
                            |
                            |  check mode (f, mod);
                            |  delete atom (f.used, f.free, f.atoms) .
                            |
                            |END PROC delete record;
                            |
                            |

insertrecord ...............|PROC insert record (FILE VAR f) :
                            |
                            |  check mode (f, mod);
                            |  insert atom (f.used, f.free, f.unused tail, f.atoms) .
                            |
                            |END PROC insert record;
                            |
                            |

down .......................|PROC down (FILE VAR f) :
                            |
                            |  check mode (f, mod);
                            |  next atom (f.used, f.atoms) .
                            |
                            |END PROC down ;
                            |

up .........................|PROC up (FILE VAR f) :
                            |
                            |  check mode (f, mod);
                            |  previous atom (f.used, f.atoms) .
                            |
                            |END PROC up ;
                            |

down .......................|PROC down (FILE VAR f, INT CONST n) :
                            |
                            |  to line (f, lineno (f) + n)
                            |
                            |ENDPROC down ;
                            |

up .........................|PROC up (FILE VAR f, INT CONST n) :
                            |
                            |  to line (f, lineno (f) - n)
                            |
                            |ENDPROC up ;
                            |
                            |
writerecord ..............|PROC write record (FILE VAR f, TEXT CONST record) :
                          |
                          |  check mode (f, mod);
                          |  IF   not at eof
                          |    THEN f.atoms (f.used.index).line := record
                          |    ELSE errorstop ("'write' nach Dateiende")
                          |  FI .
                          |
                          |
   notateof               |not at eof :  f.used.line no <= f.used.lines .
                          |
                          |END PROC write record;
                          |
                          |


readrecord ...............|PROC read record (FILE CONST f, TEXT VAR record) :
                          |
                          |  check mode (f, mod);
                          |  record := f.atoms (f.used.index).line .
                          |
                          |END PROC read record;
                          |
                          |


line .....................|PROC line (FILE VAR f) :
                          |
                          |  IF   mode = end
                          |    THEN errorstop ("Leseversuch nach Dateiende")
                          |    ELIF mode = inp
                          |    THEN next atom (f.used, f.atoms); col := 1; check eof
                          |    ELIF mode = outp
                          |    THEN IF col <= max limit
                          |           THEN col := super limit
                          |           ELSE append empty line
                          |         FI
                          |  FI .
                          |
                          |
   appendemptyline        |append empty line :
                          |  insert next (f.used, f.free, f.unused tail, f.atoms, "") .
                          |
                          |
   col                    |col : CONCR (CONCR (f)).col .
                          |
                          |
   mode                   |mode : CONCR (CONCR (f)).mode .
                          |
                          |
   checkeof               |check eof :
                          |  IF eof (f) THEN mode := end FI .
                          |
                          |END PROC line;
                          |
                          |


line .....................|PROC line (FILE VAR f, INT CONST lines) :
                          |
                          |  INT VAR i; FOR i FROM 1 UPTO lines REP line (f) PER
                          |
                             |END PROC line;
                             |
                             |


getline .....................|PROC getline (FILE VAR f, TEXT VAR text) :
                             |
                             |  check mode (f, inp);
                             |  text := subtext (record, f.col);
                             |  IF f.used.line no >= f.used.lines
                             |    THEN f.mode := end ;
                             |         set end of file
                             |    ELSE to next line ;
                             |         f.col := 1
                             |  FI .
                             |
                             |
   tonextline                |to next line :
                             |  next atom (f.used, f.atoms) .
                             |
                             |
   setendoffile              |set end of file :
                             |  f.col := LENGTH record + 1 .
                             |
                             |
   record                    |record : f.atoms (f.used.index).line .
                             |
                             |END PROC getline;
                             |
                             |


isfirstrecord ...............|BOOL PROC is first record (FILE CONST f) :
                             |
                             |  check mode (f, mod);
                             |  f.used.line no = 1 .
                             |
                             |END PROC is first record;
                             |
                             |


eof .........................|BOOL PROC eof (FILE CONST f) :
                             |
                             |  IF   line no < lines THEN FALSE
                             |  ELIF line no = lines THEN col > LENGTH record
                             |  ELSE                      TRUE
                             |  FI .
                             |
                             |
   lineno                    |line no :  f.used.line no .
                             |
   lines                     |lines   :  f.used.lines .
                             |
   col                       |col     :  f.col .
                             |
   record                    |record  :  f.atoms (f.used.index).line .
                             |
                             |END PROC eof;
                             |
                             |
lineno ..................|INT PROC line no (FILE CONST f) :
                         |
                         |  f.used.line no .
                         |
                         |END PROC line no;
                         |
                         |


linetype ................|PROC line type (FILE VAR f, INT CONST t) :
                         |
                         |  f.atoms (f.used.index).type := t .
                         |
                         |ENDPROC line type ;
                         |


linetype ................|INT PROC line type (FILE CONST f) :
                         |
                         |  f.atoms (f.used.index).type .
                         |
                         |ENDPROC line type ;
                         |
                         |


put .....................|PROC put (FILE VAR f, TEXT CONST word) :
                         |
                         |  check mode (f, outp);
                         |  IF col + LENGTH word > f.limit
                         |    THEN append new line
                         |    ELSE record CAT word
                         |  FI;
                         |  record CAT " ";
                         |  col := LENGTH record + 1 .
                         |
                         |
   appendnewline         |append new line :
                         |  insert next (f.used, f.free, f.unused tail, f.atoms, word) .
                         |
                         |
   record                |record : f.atoms (f.used.index).line .
                         |
   col                   |col :    f.col .
                         |
                         |END PROC put;
                         |
                         |


put .....................|PROC put (FILE VAR f, INT CONST value) :
                         |
                         |  put (f, text (value))
                         |
                         |END PROC put;
                         |
                         |


put .....................|PROC put (FILE VAR f, REAL CONST real) :
                         |
                         |  put (f, text (real))
```

```
                          |
                          |END PROC put;
                          |
                          |


write ....................|PROC write (FILE VAR f, TEXT CONST word) :
                          |
                          |  check mode (f, outp);
                          |  IF col + LENGTH word - 1 > f.limit
                          |    THEN append new line
                          |    ELSE record CAT word
                          |  FI;
                          |  col := LENGTH record + 1 .
                          |
                          |
   appendnewline          |append new line :
                          |  insert next (f.used, f.free, f.unused tail, f.atoms, word) .
                          |
                          |
   record                 |record : f.atoms (f.used.index).line .
                          |
   col                    |col    : f.col .
                          |
                          |END PROC write;
                          |
                          |


get ......................|PROC get (FILE VAR f, TEXT VAR word, TEXT CONST separator) :
                          |
                          |  check mode (f, inp);
                          |  skip separators;
                          |  IF word found
                          |    THEN get word
                          |    ELSE try to find word in next line
                          |  FI .
                          |
                          |
   skipseparators         |skip separators :
                          |  INT CONST separator length := LENGTH separator;
                          |  WHILE is separator REP col INCR separator length PER .
                          |
                          |
   isseparator            |is separator :
                          |  subtext (record, col, col + separator length - 1) = separator .
                          |
                          |
   wordfound              |word found : col <= LENGTH record .
                          |
                          |
   getword                |get word :
                          |  INT VAR end of word := pos (record, separator, col) - 1;
                          |  IF separator found
                          |    THEN get text upto separator
                          |    ELSE get rest of record
                          |  FI .
                          |
                          |
   separatorfound         |separator found : end of word >= 0 .
                          |
    gettextuptoseparator  |get text upto separator :
                          |  word := subtext (record, col, end of word);
                          |  col := end of word + separator length + 1;
                          |  IF col > LENGTH record THEN line (f) FI .
                          |
                          |
    getrestofrecord       |get rest of record :
                          |  word := subtext (record, col); line (f) .
                          |
                          |
    record                |record : f.atoms (f.used.index).line .
                          |
    col                   |col    : f.col .
                          |
                          |
    trytofindwordinnextlin |try to find word in next line :
                          |  line (f); IF eof (f) THEN word := "" ELSE get (f, word, separator)
                          |       FI .
                          |
                          |END PROC get;
                          |
                          |


get ......................|PROC get (FILE VAR f, TEXT VAR word, INT CONST max length) :
                          |
                          |  check mode (f, inp);
                          |  IF word is only a part of record
                          |    THEN get text of certain length
                          |    ELSE get rest of record
                          |  FI .
                          |
                          |
    wordisonlyapartofrecor |word is only a part of record :
                          |  col <= LENGTH record - max length .
                          |
                          |
    gettextofcertainlength |get text of certain length :
                          |  word := text (record, max length, col);
                          |  col INCR max length .
                          |
                          |
    getrestofrecord       |get rest of record :
                          |  word := subtext (record, col); line (f) .
                          |
                          |
    record                |record : f.atoms (f.used.index).line .
                          |
    col                   |col    : f.col .
                          |
                          |END PROC get;
                          |
                          |


get ......................|PROC get (FILE VAR f, TEXT VAR word) :
                          |
                          |  get (f, word, " ")
                          |
                          |END PROC get;
                          |
                          |
                        |TEXT VAR number word;
                        |
                        |


get ....................|PROC get (FILE VAR f, INT VAR number) :
                        |
                        |  get (f, number word);
                        |  number := int (number word)
                        |
                        |END PROC get;
                        |
                        |


get ....................|PROC get (FILE VAR f, REAL VAR number) :
                        |
                        |  get (f, number word);
                        |  number := real (number word)
                        |
                        |END PROC get;
                        |
                        |
                        |TEXT VAR split record ;
                        |INT VAR indentation ;
                        |


splitline ..............|PROC split line (FILE VAR f, INT CONST split col) :
                        |
                        |  split line (f, split col, TRUE)
                        |
                        |ENDPROC split line ;
                        |


splitline ..............|PROC split line (FILE VAR f, INT CONST split col, BOOL CONST note
                        |     indentation ) :
                        |
                        |  IF note indentation
                        |    THEN get indentation
                        |    ELSE indentation := 0
                        |  FI ;
                        |  get split record ;
                        |  insert split record and indentation ;
                        |  cut off old record .
                        |
                        |
   getindentation       |get indentation :
                        |  indentation := pos (actual record,""33"",""254"",1) - 1 ;
                        |  IF indentation < 0 OR indentation >= split col
                        |    THEN indentation := split col - 1
                        |  FI .
                        |
                        |
   getsplitrecord       |get split record :
                        |  split record := subtext (actual record, split col, max limit)
                        |
                        |
   insertsplitrecordandin |insert split record and indentation :
                        |  down (f) ;
                        |  insert record (f) ;
```

```
                              | INT VAR i ;
                              | FOR i FROM 1 UPTO indentation REP
                              |   actual record CAT " "
                              | PER ;
                              | actual record CAT split record ;
                              | up (f) .
                              |
                              |
    cutoffoldrecord           |cut off old record :
                              | actual record := subtext (actual record, 1, split col-1) .
                              |
                              |
    actualrecord              |actual record : f.atoms (f.used.index).line .
                              |
                              |ENDPROC split line ;
                              |


concatenateline ..........|PROC concatenate line (FILE VAR f, BOOL CONST delete blanks) :
                              |
                              | down (f) ;
                              | split record := actual record ;
                              | IF delete blanks
                              |   THEN delete leading blanks
                              | FI ;
                              | delete record (f) ;
                              | up (f) ;
                              | actual record CAT split record .
                              |
                              |
    deleteleadingblanks       |delete leading blanks :
                              | INT CONST non blank col := pos (split record, ""33"", ""254"", 1)
                              | IF non blank col > 0
                              |   THEN split record := subtext (split record, non blank col)
                              | FI .
                              |
                              |
    actualrecord              |actual record :  f.atoms (f.used.index).line .
                              |
                              |ENDPROC concatenate line ;
                              |


concatenateline ..........|PROC concatenate line (FILE VAR f) :
                              | concatenate line (f, TRUE)
                              |ENDPROC concatenate line ;
                              |


reorganize ...............|PROC reorganize :
                              |
                              | reorganize (last param)
                              |
                              |END PROC reorganize;
                              |
                              |
                              |TEXT VAR file record ;
                              |
reorganize ..............|PROC reorganize (TEXT CONST file name) :
                         |
                         |  enable stop ;
                         |  FILE VAR input file, output file;
                         |  DATASPACE VAR scratch space;
                         |  INT CONST type of dataspace := type (old (file name)) ;
                         |  INT VAR counter;
                         |
                         |  last param (file name);
                         |  IF type of dataspace = file type
                         |    THEN reorganize new to new
                         |  ELIF type of dataspace = file type 16
                         |    THEN reorganize old to new
                         |    ELSE errorstop ("Datenraum hat falschen Typ")
                         |  FI;
                         |  replace file space by scratch space .
                         |
                         |
   reorganizenewtonew    |reorganize new to new :
                         |  input  file := sequential file (input, file name);
                         |  disable stop ;
                         |  scratch space := nilspace ;
                         |  output file := sequential file (output, scratch space);
                         |  copy attributes (input file, output file) ;
                         |
                         |  FOR counter FROM 1 UPTO 9999
                         |  WHILE NOT eof (input file) REP
                         |    cout (counter);
                         |    getline (input file, file record);
                         |    putline (output file, file record);
                         |    check for interrupt
                         |  PER .
                         |
                         |
   reorganizeoldtonew    |reorganize old to new :
                         |  LET OLDRECORD = STRUCT (INT succ, pred, x, y, TEXT record);
                         |  LET OLDFILE = BOUND ROW 4075 OLDRECORD;
                         |  LET dateianker = 2, freianker = 1;
                         |  INT VAR index := dateianker;
                         |
                         |  OLDFILE VAR old file := old (file name);
                         |  disable stop;
                         |  scratch space := nilspace;
                         |  output file := sequential file (output, scratch space);
                         |  get old attributes ;
                         |
                         |  say ("Datei wird in 1.7-Format gewandelt: ") ;
                         |
                         |  FOR counter FROM 1 UPTO 9999
                         |  WHILE NOT end of old file REP
                         |    cout (counter);
                         |    index := next record;
                         |    file record := record of old file ;
                         |    IF pos (file record, ""128"", ""250"", 1) > 0
                         |      THEN change special chars
                         |    FI ;
                         |    putline (output file, file record);
                         |    check for interrupt
                         |  PER .
                         |
getoldattributes      |get old attributes :
                      |  get old headline ;
                      |  get old limit and tabs .
                      |
                      |
getoldheadline        |get old headline :
                      |  headline (output file, old file (dateianker).record) .
                      |
                      |
getoldlimitandtabs    |get old limit and tabs :
                      |  file record := old file (freianker).record ;
                      |  max line length (output file, int (subtext (file record, 11, 15))
                 +    |      ;
                      |  put tabs (output file, subtext (file record, 16)) .
                      |
                      |
changespecialchars    |change special chars :
                      |  change all (file record, ""193"", ""214"") (* Ae *) ;
                      |  change all (file record, ""207"", ""215"") (* Oe *) ;
                      |  change all (file record, ""213"", ""216"") (* Ue *) ;
                      |  change all (file record, ""225"", ""217"") (* ae *) ;
                      |  change all (file record, ""239"", ""218"") (* oe *) ;
                      |  change all (file record, ""245"", ""219"") (* ue *) ;
                      |  change all (file record, ""235"", ""220"") (* k  *) ;
                      |  change all (file record, ""173"", ""221"") (* -  *) ;
                      |  change all (file record, ""163"", ""222"") (* fis  *) ;
                      |  change all (file record, ""160"", ""223"") (* blank *) ;
                      |  change all (file record, ""194"", ""251"") (* eszet *) .
                      |
                      |
endofoldfile          |end of old file :      next record = dateianker .
                      |
                      |
nextrecord            |next record :          old file (index).succ .
                      |
                      |
recordofoldfile       |record of old file :   old file (index).record .
                      |
                      |
checkforinterrupt     |check for interrupt :
                      |  INT VAR size, used ;
                      |  storage (size, used) ;
                      |  IF used > size
                      |    THEN errorstop ("Speicherengpass")
                      |  FI ;
                      |  IF is error
                      |    THEN forget (scratch space) ; LEAVE reorganize
                      |  FI .
                      |
                      |
replacefilespacebyscra |replace file space by scratch space :
                      |  headline (output file, file name);
                      |  forget (file name, quiet) ;
                      |  type (scratch space, file type);
                      |  copy (scratch space, file name);
                      |  forget (scratch space) .
                      |
                      |END PROC reorganize;
                      |
                      |
setrange ................|PROC set range (FILE VAR f, INT CONST start line, start col,
                         |                FRANGE VAR old range) :
                         |
                         |  check mode (f, mod);
                         |  IF valid restriction parameters
                         |    THEN prepare last line ;
                         |         prepare first line ;
                         |         save old range ;
                         |         set new range
                         |    ELSE errorstop ("FRANGE ungueltig")
                         |  FI .
                         |
                         |
   validrestrictionparame |valid restriction parameters :
                         |  start line > 0 AND start col > 0 AND start before or at actual
                         |       point .
                         |
                         |
   startbeforeoratactualp |start before or at actual point :
                         |  start line < line no (f)  OR
                         |  start line = line no (f) AND start col <= col (f) .
                         |
                         |
   preparelastline       |prepare last line :
                         |  INT VAR last line ;
                         |  IF col (f) > 1
                         |    THEN split line (f, col(f), FALSE)
                         |  FI .
                         |
                         |
   preparefirstline      |prepare first line :
                         |  IF start col > 1
                         |    THEN split start line ;
                         |  FI .
                         |
                         |
   splitstartline        |split start line :
                         |  INT VAR old line no := line no (f) ;
                         |  to line (f, start line) ;
                         |  split line (f, start col, FALSE) ;
                         |  to line (f, old line no + 1) .
                         |
                         |
   saveoldrange          |save old range :
                         |  old range.pre := f.prefix lines ;
                         |  old range.post:= f.postfix lines .
                         |
                         |
   setnewrange           |set new range :
                         |  get pre lines ;
                         |  get post lines ;
                         |  disable stop ;
                         |  f.prefix lines INCR pre lines ;
                         |  f.postfix lines INCR post lines ;
                         |  f.used.lines DECR (post lines + pre lines) ;
                         |  f.used.line no DECR pre lines .
                         |
                         |
   getprelines           |get pre lines :
                         |  INT VAR pre lines ;
                         |  IF start col = 1
                         |    THEN old range.pre was split := FALSE ;
```

```
                              |         pre lines := start line - 1
                              |    ELSE old range.pre was split := TRUE ;
                              |         pre lines := start line
                              |  FI .
                              |
                              |
   getpostlines               |get post lines :
                              |  INT VAR post lines ;
                              |  IF col (f) = 1
                              |    THEN old range.post was split := FALSE ;
                              |         post lines := lines (f) - line no (f) + 1
                              |    ELSE old range.post was split := TRUE ;
                              |         post lines := lines (f) - line no (f)
                              |  FI .
                              |
                              |END PROC set range;
                              |
                              |


setrange .....................|PROC set range (FILE VAR f, FRANGE VAR new range) :
                              |
                              |  check mode (f, mod);
                              |  INT CONST pre add  := prefix - new range.pre,
                              |            post add := postfix - new range.post;
                              |  IF pre add < 0 OR post add < 0
                              |    THEN errorstop ("FRANGE ungueltig")
                              |    ELSE set new range;
                              |         undo splitting if necessary ;
                              |         make range var invalid
                              |  FI .
                              |
                              |
   setnewrange                |set new range :
                              |  disable stop;
                              |  prefix DECR pre add;
                              |  postfix DECR post add;
                              |  used.line no INCR pre add;
                              |  used.lines INCR (pre add + post add) .
                              |
                              |
   undosplittingifnecessa     |undo splitting if necessary :
                              |  IF new range.pre was split
                              |    THEN concatenate first line
                              |  FI ;
                              |  IF new range.post was split
                              |    THEN concatenate last line
                              |  FI .
                              |
                              |
   concatenatefirstline       |concatenate first line :
                              |  INT VAR old line := line no (f) ;
                              |  to line (f, pre add) ;
                              |  concatenate line (f, FALSE) ;
                              |  to line (f, old line - 1) .
                              |
                              |
   concatenatelastline        |concatenate last line :
                              |  old line := line no (f) ;
                              |  to line (f, lines (f) - post add) ;
                              |  concatenate line (f, FALSE) ;
                              |  to line (f, old line) .
```

```
                          |
                          |
   makerangevarinvalid    |make range var invalid :
                          |  new range.pre := maxint .
                          |
                          |
   used                   |used :    f.used .
                          |
   prefix                 |prefix :  f.prefix lines .
                          |
   postfix                |postfix : f.postfix lines .
                          |
                          |END PROC set range;
                          |


resetrange ...............|PROC reset range (FILE VAR f) :
                          |
                          |  FRANGE VAR complete ;
                          |  complete.pre := 0 ;
                          |  complete.post:= 0 ;
                          |  complete.pre was split := FALSE ;
                          |  complete.post was split:= FALSE ;
                          |  set range (f, complete)
                          |
                          |ENDPROC reset range ;
                          |


remove ...................|PROC remove (FILE VAR f, INT CONST size) :
                          |
                          |  check mode (f, mod);
                          |  transfer subsequence (f.used, f.scratch, f.atoms, size) .
                          |
                          |END PROC remove;
                          |
                          |


clearremoved .............|PROC clear removed (FILE VAR f) :
                          |
                          |  check mode (f, mod);
                          |  transfer subsequence (f.scratch, f.free, f.atoms, f.scratch.lines)
                          |       .
                          |
                          |END PROC clear removed;
                          |
                          |


reinsert .................|PROC reinsert (FILE VAR f) :
                          |
                          |  check mode (f, mod);
                          |  transfer subsequence (f.scratch, f.used, f.atoms, f.scratch.lines)
                          |       .
                          |
                          |END PROC reinsert;
                          |
                          |
copyattributes ...........|PROC copy attributes (FILE CONST source file, FILE VAR dest file) :
                          |
                          |  dest.limit                       := source.limit ;
                          |  dest.atoms (free root).line      := source.atoms (free root).line ;
                          |  dest.atoms (scratch root).line := source.atoms (scratch root).line
                          |      ;
                          |  dest.edit info                   := source.edit info .
                          |
                          |
   dest                   |dest   : CONCR (CONCR (dest file)) .
                          |
   source                 |source : CONCR (CONCR (source file)) .
                          |
                          |ENDPROC copy attributes ;
                          |
                          |


maxlinelength ............|INT PROC max line length (FILE CONST f) :
                          |
                          |  f.limit .
                          |
                          |END PROC max line length;
                          |
                          |


maxlinelength ............|PROC max line length (FILE VAR f, INT CONST new limit) :
                          |
                          |  IF new limit > 0 AND new limit <= max limit
                          |    THEN f.limit := new limit
                          |  FI .
                          |
                          |END PROC max line length;
                          |
                          |


headline .................|TEXT PROC headline (FILE CONST f) :
                          |
                          |  f.atoms (free root).line .
                          |
                          |END PROC headline;
                          |
                          |


headline .................|PROC headline (FILE VAR f, TEXT CONST head) :
                          |
                          |  f.atoms (free root).line := head .
                          |
                          |END PROC headline;
                          |
                          |


gettabs ..................|PROC get tabs (FILE CONST f, TEXT VAR tabs) :
                          |
                          |  tabs := f.atoms (scratch root).line .
                          |
                          |END PROC get tabs;
                          |
                             |
puttabs .................. |PROC put tabs (FILE VAR f, TEXT CONST tabs) :
                             |
                             |  f.atoms (scratch root).line := tabs .
                             |
                             |END PROC put tabs;
                             |
                             |
editinfo ................. |INT PROC edit info (FILE CONST f) :
                             |
                             |  f.edit info .
                             |
                             |END PROC edit info;
                             |
                             |
editinfo ................. |PROC edit info (FILE VAR f, INT CONST info) :
                             |
                             |  f.edit info := info .
                             |
                             |END PROC edit info;
                             |
                             |
lines .................... |INT PROC lines (FILE CONST f) :
                             |
                             |  f.used.lines .
                             |
                             |END PROC lines;
                             |
                             |
removedlines ............. |INT PROC removed lines (FILE CONST f) :
                             |
                             |  f.scratch.lines .
                             |
                             |END PROC removed lines;
                             |
                             |
segments ................. |INT PROC segments (FILE CONST f) :
                             |
                             |  segs(f.used,f.atoms) + segs(f.scratch,f.atoms) +
                             |       segs(f.free,f.atoms) - 2 .
                             |
                             |ENDPROC segments ;
                             |
                             |
col ...................... |INT PROC col (FILE CONST f) :
                             |
                             |  f.col
                             |
                                |ENDPROC col ;
                                |


col ............................|PROC col (FILE VAR f, INT CONST new column) :
                                |
                                |  IF new column > 0
                                |    THEN f.col := new column
                                |  FI
                                |
                                |ENDPROC col ;
                                |


word ...........................|TEXT PROC word (FILE CONST f) :
                                |
                                |  word (f, " ")
                                |
                                |ENDPROC word ;
                                |


word ...........................|TEXT PROC word (FILE CONST f, TEXT CONST delimiter) :
                                |
                                |  INT VAR del pos := pos (f, delimiter, col (f)) ;
                                |  IF del pos = 0
                                |    THEN del pos := len (f) + 1
                                |  FI ;
                                |  subtext (f, col (f), del pos - 1)
                                |
                                |ENDPROC word ;
                                |


word ...........................|TEXT PROC word (FILE CONST f, INT CONST max length) :
                                |
                                |  subtext (f, col (f), col (f) + max length - 1)
                                |
                                |ENDPROC word ;
                                |


at .............................|BOOL PROC at (FILE CONST f, TEXT CONST word) :
                                |
                                |  pat := any (column-1) ;
                                |  pat CAT word ;
                                |  pat CAT any ;
                                |  record LIKE pat .
                                |
                                |
   column                       |column : f.col .
                                |
   record                       |record : f.atoms (f.used.index).line .
                                |
                                |ENDPROC at ;
                                |
                                |


exec ...........................|PROC exec (PROC (TEXT VAR, TEXT CONST) proc, FILE VAR f, TEXT CONST
  +                             |     t) :
                                |
                         |  proc (record, t) .
                         |
                         |
   record                |record : f.atoms (f.used.index).line .
                         |
                         |END PROC exec;
                         |
                         |


exec ....................|PROC exec (PROC (TEXT VAR, INT CONST) proc, FILE VAR f, INT CONST i)
                         |     :
                         |
                         |  proc (record, i) .
                         |
                         |
   record                |record : f.atoms (f.used.index).line .
                         |
                         |END PROC exec;
                         |


pos .....................|INT PROC pos (FILE CONST f, TEXT CONST pattern, INT CONST i) :
                         |
                         |  pos (record, pattern, i) .
                         |
                         |
   record                |record : f.atoms (f.used.index).line .
                         |
                         |END PROC pos ;
                         |


down ....................|PROC down (FILE VAR f, TEXT CONST pattern) :
                         |
                         |  down (f, pattern, file size)
                         |
                         |ENDPROC down ;
                         |


down ....................|PROC down (FILE VAR f, TEXT CONST pattern, INT CONST max line) :
                         |
                         |  check mode (f,mod) ;
                         |  INT VAR pattern pos := f.col + 1 ;
                         |  search down (f.used, f.atoms, pattern, max line, pattern pos) ;
                         |  f.col := pattern pos
                         |
                         |ENDPROC down ;
                         |


downety .................|PROC downety (FILE VAR f, TEXT CONST pattern) :
                         |
                         |  downety (f, pattern, file size)
                         |
                         |ENDPROC downety ;
                         |
downety ..................|PROC downety (FILE VAR f, TEXT CONST pattern, INT CONST max line)
                          |
                          |  check mode (f,mod) ;
                          |  INT VAR pattern pos := f.col ;
                          |  search down (f.used, f.atoms, pattern, max line, pattern pos) ;
                          |  f.col := pattern pos
                          |
                          |ENDPROC downety ;
                          |


up .......................|PROC up (FILE VAR f, TEXT CONST pattern) :
                          |
                          |  up (f, pattern, file size)
                          |
                          |ENDPROC up ;
                          |


up .......................|PROC up (FILE VAR f, TEXT CONST pattern, INT CONST max line) :
                          |
                          |  check mode (f,mod) ;
                          |  INT VAR pattern pos := f.col - 1 ;
                          |  search up (f.used, f.atoms, pattern, max line, pattern pos) ;
                          |  f.col := pattern pos
                          |
                          |ENDPROC up ;
                          |


uppety ...................|PROC uppety (FILE VAR f, TEXT CONST pattern) :
                          |
                          |  uppety (f, pattern, file size)
                          |
                          |ENDPROC uppety ;
                          |


uppety ...................|PROC uppety (FILE VAR f, TEXT CONST pattern, INT CONST max line)
                          |
                          |  check mode (f,mod) ;
                          |  INT VAR pattern pos := f.col ;
                          |  search up (f.used, f.atoms, pattern, max line, pattern pos) ;
                          |  f.col := pattern pos
                          |
                          |ENDPROC uppety ;
                          |
                          |


len ......................|INT PROC len (FILE CONST f) :
                          |
                          |  length (record) .
                          |
                          |
   record                 |record : f.atoms (f.used.index).line .
                          |
                          |ENDPROC len ;
                          |
subtext ..................|TEXT PROC subtext (FILE CONST f, INT CONST from, to) :
                          |
                          |  subtext (record, from, to) .
                          |
                          |
   record                 |record : f.atoms (f.used.index).line .
                          |
                          |ENDPROC subtext ;
                          |


change ...................|PROC change (FILE VAR f, INT CONST from, to, TEXT CONST new) :
                          |
                          |  check mode (f, mod) ;
                          |  change (record, from, to, new) .
                          |
                          |
   record                 |record : f.atoms (f.used.index).line .
                          |
                          |ENDPROC change ;
                          |
                          |


mark .....................|BOOL PROC mark (FILE CONST f) :
                          |
                          |  f.mark line > 0
                          |
                          |ENDPROC mark ;
                          |


mark .....................|PROC mark (FILE VAR f, INT CONST line no, col) :
                          |
                          |  IF line no > 0
                          |    THEN f.mark line := line no + f.prefix lines ;
                          |         f.mark col  := col
                          |    ELSE f.mark line := 0 ;
                          |         f.mark col  := 0
                          |  FI
                          |
                          |ENDPROC mark ;
                          |


marklineno ...............|INT PROC mark line no (FILE CONST f) :
                          |
                          |  IF f.mark line = 0
                          |    THEN 0
                          |    ELSE max (1, f.mark line - f.prefix lines)
                          |  FI
                          |
                          |ENDPROC mark line no ;
                          |


markcol ..................|INT PROC mark col (FILE CONST f) :
                          |
                          |  IF f.mark line = 0
                          |    THEN 0
                          |  ELIF f.mark line <= f.prefix lines
                          |    THEN 1
```

```
                            |  ELSE f.mark col
                            |  FI
                            |
                            |ENDPROC mark col ;
                            |


setmarkedrange .............|PROC set marked range (FILE VAR f, FRANGE VAR old range) :
                            |
                            |  IF mark (f)
                            |    THEN set range (f, mark line no (f), mark col (f), old range)
                            |    ELSE old range := previous range of file
                            |  FI .
                            |
                            |
   previousrangeoffile      |previous range of file :
                            |  FRANGE : (f.prefix lines, f.postfix lines, FALSE, FALSE) .
                            |
                            |ENDPROC set marked range ;
                            |
                            |
                            |(*****************************************************************)
                            |
                            |                                                   (* Autor:
                            |     P.Heyderhoff *)
                            |                                                   (* Stand: 11.10.83
                            |         *)
                            |
                            |BOUND LIST VAR datei;
                            |INT  VAR sortierstelle, sortanker;
                            |BOOL VAR ascii sort;
                            |TEXT VAR median, tausch , links, rechts;
                            |


sort .......................|PROC sort (TEXT CONST dateiname) :
                            |     sort (dateiname, 1)
                            |END PROC sort;
                            |


sort .......................|PROC sort (TEXT CONST dateiname, INT CONST sortieranfang) :
                            |     ascii sort := TRUE ;
                            |     sortierstelle := sortieranfang; sortiere (dateiname)
                            |END PROC sort;
                            |


lexsort ....................|PROC lex sort (TEXT CONST dateiname) :
                            |     lex sort (dateiname, 1)
                            |ENDPROC lex sort ;
                            |


lexsort ....................|PROC lex sort (TEXT CONST dateiname, INT CONST sortieranfang) :
                            |     ascii sort := FALSE ;
                            |     sortierstelle := sortieranfang; sortiere (dateiname)
                            |ENDPROC lex sort ;
                            |
sortiere ................|PROC sortiere (TEXT CONST dateiname) :
                         |
                         |  reorganize file if necessary ;
                         |  sort file .
                         |
                         |
   reorganizefileifnecess |reorganize file if necessary :
                         |  FILE VAR f := sequential file (modify, dateiname) ;
                         |  IF segments (f) > 1
                         |    THEN reorganize (dateiname)
                         |  FI .
                         |
                         |
   sortfile              |sort file :
                         |  f := sequential file (modify, dateiname) ;
                         |  INT CONST sortende := lines (f) + 3 ;
                         |  sortanker := 1 + 3 ;
                         |  datei := old (dateiname) ;
                         |  quicksort(sortanker, sortende) .
                         |
                         |END PROC sortiere;
                         |


quicksort ...............|PROC quicksort ( INT CONST anfang, ende ) :
                         |     IF   anfang < ende
                         |     THEN INT VAR p,q;
                         |          spalte    (anfang, ende, p, q);
                         |          quicksort (anfang, q);
                         |          quicksort (p, ende)    FI
                         |END PROC quicksort;
                         |


spalte ..................|PROC spalte (INT CONST anfang, ende, INT VAR p, q):
                         |     fange an der seite an und waehle den median;
                         |     ruecke p und q so dicht wie moeglich zusammen;
                         |     hole ggf median in die mitte .
                         |
                         |
   fangeanderseiteanundwa |   fange an der seite an und waehle den median :
                         |     p := anfang; q := ende ;
                         |     INT CONST m :: (p + q) DIV 2 ;
                         |     median := subtext(datei m, sortierstelle) .
                         |
                         |
   rueckepundqsodichtwiem |   ruecke p und q so dicht wie moeglich zusammen :
                         |     REP schiebe p und q so weit wie moeglich auf bzw ab;
                         |         IF p < q THEN vertausche die beiden FI
                         |     UNTIL p > q END REP .
                         |
                         |
   vertauschediebeiden    |   vertausche die beiden :
                         |     tausch := datei p; datei p := datei q; datei q := tausch;
                         |     p INCR 1; q DECR 1 .
                         |
                         |
   schiebepundqsoweitwiem |   schiebe p und q so weit wie moeglich auf bzw ab :
                         |     WHILE p kann groesser werden REP p INCR 1 END REP;
                         |     WHILE q kann kleiner  werden REP q DECR 1 END REP .
                         |
pkanngroesserwerden      |  p kann groesser werden :
                         |    IF p <= ende
                         |      THEN links := subtext (datei p, sortierstelle) ;
                         |           IF ascii sort
                         |             THEN median >= links
                         |             ELSE median LEXGREATEREQUAL links
                         |           FI
                         |      ELSE FALSE
                         |    FI .
                         |
                         |
qkannkleinerwerden       |  q kann kleiner werden :
                         |    IF q >= anfang
                         |      THEN rechts := subtext(datei q, sortierstelle) ;
                         |           IF ascii sort
                         |             THEN rechts >= median
                         |             ELSE rechts LEXGREATEREQUAL median
                         |           FI
                         |      ELSE FALSE
                         |    FI .
                         |
                         |
holeggfmedianindiemitt   |  hole ggf median in die mitte :
                         |    IF   m < q THEN vertausche m und q
                         |    ELIF m > p THEN vertausche m und p FI .
                         |
                         |
vertauschemundq          |  vertausche m und q :
                         |    tausch := datei m; datei m := datei q; datei q := tausch; q
   +                     |         DECR 1 .
                         |
                         |
vertauschemundp          |  vertausche m und p :
                         |    tausch := datei m; datei m := datei p; datei p := tausch; p
   +                     |         INCR 1 .
                         |
                         |
dateim                   |  datei m :  datei.atoms (m).line .
                         |
dateip                   |  datei p :  datei.atoms (p).line .
                         |
dateiq                   |  datei q :  datei.atoms (q).line .
                         |
                         |END PROC spalte;
                         |
                         |END PACKET file handling;
```

```
                          |
elandointerface **********|PACKET elan do interface DEFINES                   (*Autor: J.Lied
                          |     *)
                          |                                                    (*Stand: 08.11.85
                          |     *)
                          |    do ,
                          |    no do again :
                          |
                          |
                          |LET no ins = FALSE ,
                          |    no lst = FALSE ,
                          |    no check = FALSE ,
                          |    no sermon = FALSE ,
                          |    compile line mode = 2 ,
                          |    do again mode = 4 ,
                          |    max command length = 2000 ;
                          |
                          |
                          |INT VAR do again mod nr := 0 ;
                          |TEXT VAR previous command := "" ;
                          |
                          |DATASPACE VAR ds ;
                          |
                          |


do .......................|PROC do (TEXT CONST command) :
                          |
                          |  enable stop ;
                          |  IF LENGTH command > max command length
                          |    THEN errorstop ("Kommando zu lang")
                          |  ELIF do again mod nr <> 0 AND command = previous command
                          |    THEN do again
                          |    ELSE previous command := command ;
                          |         compile and execute
                          |  FI .
                          |
                          |
   doagain                |do again :
                          |  elan (do again mode, ds, "", do again mod nr,
                          |        no ins, no lst, no check, no sermon) .
                          |
                          |
   compileandexecute      |compile and execute :
                          |  elan (compile line mode, ds, command, do again mod nr,
                          |        no ins, no lst, no check, no sermon) .
                          |
                          |ENDPROC do ;
                          |


nodoagain ................|PROC no do again :
                          |
                          |  do again mod nr := 0
                          |
                          |ENDPROC no do again ;
                          |


elan .....................|PROC elan (INT CONST mode, DATASPACE CONST source, TEXT CONST line,
                          |          INT VAR start module number,
                          |          BOOL CONST ins, lst, rt check, ser) :
```

```
|  EXTERNAL 256
|ENDPROC elan ;
|
|ENDPACKET elan do interface ;
```

```
                            |(* ------------------- VERSION 4     14.05.86 ------------------- *)
scanner ********************|PACKET scanner DEFINES                       (* Autor: J.Liedtke      *)
                            |
                            |      scan ,
                            |      continue scan ,
                            |      next symbol :
                            |
                            |
                            |LET tag     = 1 ,
                            |    bold    = 2 ,
                            |    number  = 3 ,
                            |    text    = 4 ,
                            |    operator= 5 ,
                            |    delimiter = 6 ,
                            |    end of file = 7 ,
                            |    within comment = 8 ,
                            |    within text    = 9 ;
                            |
                            |LET digit 0        = 48 ,
                            |    digit 9        = 57 ,
                            |    upper case a   = 65 ,
                            |    upper case z   = 90 ,
                            |    lower case a   = 97 ,
                            |    lower case z   = 122;
                            |
                            |
                            |TEXT VAR line := "" ,
                            |         char := "" ,
                            |         chars:= "" ;
                            |
                            |INT VAR  position  := 0 ,
                            |         comment depth ;
                            |BOOL VAR continue text ;
                            |
                            |


scan .......................|PROC scan (TEXT CONST scan text) :
                            |
                            |  comment depth := 0 ;
                            |  continue text := FALSE ;
                            |  continue scan (scan text)
                            |
                            |ENDPROC scan ;
                            |


continuescan ...............|PROC continue scan (TEXT CONST scan text) :
                            |
                            |  line := scan text ;
                            |  position  := 0 ;
                            |  nextchar
                            |
                            |ENDPROC continue scan ;
                            |


nextsymbol .................|PROC next symbol (TEXT VAR symbol) :
                            |
                            |  INT VAR type ;
                            |  next symbol (symbol, type)
                            |
                               |ENDPROC next symbol ;
                               |

nextsymbol ....................|PROC next symbol (TEXT VAR symbol, INT VAR type) :
                               |
                               |  skip blanks ;
                               |  IF   is begin comment          THEN process comment
                               |  ELIF comment depth > 0         THEN comment depth DECR 1 ;
                               |                                      process comment
                               |  ELIF is quote OR continue text THEN process text
                               |  ELIF is lower case letter      THEN process tag
                               |  ELIF is upper case letter      THEN process bold
                               |  ELIF is digit                  THEN process number
                               |  ELIF is delimiter              THEN process delimiter
                               |  ELIF is niltext                THEN eof
                               |  ELSE process operator
                               |  FI .
                               |
                               |
                               |
   processcomment              |process comment :
                               |  read comment ;
                               |  IF comment depth = 0
                               |    THEN next symbol (symbol, type)
                               |    ELSE type := within comment ;
                               |         symbol := ""
                               |  FI .
                               |
                               |
   processtag                  |process tag :
                               |  type := tag ;
                               |  assemble chars (lower case a, lower case z) ;
                               |  symbol := chars ;
                               |  REP
                               |    skip blanks ;
                               |    IF is lower case letter
                               |      THEN assemble chars (lower case a, lower case z)
                               |    ELIF is digit
                               |      THEN assemble chars (digit 0, digit 9)
                               |    ELSE   LEAVE process tag
                               |    FI ;
                               |    symbol CAT chars
                               |  PER ;
                               |  nextchar .
                               |
                               |
   processbold                 |process bold :
                               |  type := bold ;
                               |  assemble chars (upper case a, upper case z) ;
                               |  symbol := chars .
                               |
                               |
   processnumber               |process number :
                               |  type := number ;
                               |  assemble chars (digit 0, digit 9) ;
                               |  symbol := chars ;
                               |  IF char = "." AND ahead char is digit
                               |    THEN process fraction ;
                               |         IF char = "e"
                               |           THEN process exponent
                               |         FI
```

```
                        |  FI .
                        |
                        |
aheadcharisdigit        |ahead char is digit :
                        |  digit 0 <= code (ahead char) AND code (ahead char) <= digit 9 .
                        |
                        |
processfraction         |process fraction :
                        |  symbol CAT char ;
                        |  nextchar ;
                        |  assemble chars (digit 0, digit 9) ;
                        |  symbol CAT chars .
                        |
                        |
processexponent         |process exponent :
                        |  symbol CAT char ;
                        |  nextchar ;
                        |  IF char = "+" OR char = "-"
                        |    THEN symbol CAT char ;
                        |         nextchar
                        |  FI ;
                        |  assemble chars (digit 0, digit 9) ;
                        |  symbol CAT chars .
                        |
                        |
processtext             |process text :
                        |  type := text ;
                        |  symbol := "" ;
                        |  IF continue text
                        |    THEN continue text := FALSE
                        |    ELSE next char
                        |  FI ;
                        |  WHILE not end of text REP
                        |    assemble chars (35, 254) ;
                        |    symbol CAT chars ;
                        |    IF NOT is quote
                        |      THEN symbol CAT char ;
                        |           nextchar
                        |    FI
                        |  ENDREP .
                        |
                        |
notendoftext            |not end of text :
                        |  IF is niltext
                        |    THEN continue text := TRUE ; type := within text ; FALSE
                        |  ELIF is quote
                        |    THEN end of text or exception
                        |  ELSE TRUE
                        |  FI .
                        |
                        |
endoftextorexception    |end of text or exception :
                        |  next char ;
                        |  IF is quote
                        |    THEN get quote ;        TRUE
                        |  ELIF is digit
                        |    THEN get special char ; TRUE
                        |  ELSE   FALSE
                        |  FI .
                        |
getquote               |get quote :
                       |  symbol CAT char ;
                       |  nextchar .
                       |
                       |
getspecialchar         |get special char :
                       |  assemble chars (digit 0, digit 9) ;
                       |  symbol CAT code (int (chars) ) ;
                       |  nextchar .
                       |
                       |
processdelimiter       |process delimiter :
                       |  type := delimiter ;
                       |  symbol := char ;
                       |  nextchar .
                       |
                       |
processoperator        |process operator :
                       |  type := operator ;
                       |  symbol := char ;
                       |  nextchar ;
                       |  IF symbol = ":"
                       |    THEN IF char = "=" OR char = ":"
                       |           THEN symbol := ":=" ;
                       |                nextchar
                       |           ELSE type := delimiter
                       |         FI
                       |  ELIF is relational double char
                       |    THEN symbol CAT char ;
                       |         nextchar
                       |  ELIF symbol = "*" AND char = "*"
                       |    THEN symbol := "**" ;
                       |         next char
                       |  FI .
                       |
                       |
eof                    |eof :
                       |  type := end of file ;
                       |  symbol := "" .
                       |
                       |
islowercaseletter      |is lower case letter :
                       |     lower case a <= code (char) AND code (char) <= lower case z .
                       |
                       |
isuppercaseletter      |is upper case letter :
                       |     upper case a <= code (char) AND code (char) <= upper case z .
                       |
                       |
isdigit                |is digit :
                       |     digit 0 <= code (char) AND code (char) <= digit 9 .
                       |
                       |
isdelimiter            |is delimiter : pos ( "()[].,;" , char ) > 0 .
                       |
                       |
isrelationaldoublechar |is relational double char :
                       |  TEXT VAR double := symbol + char ;
                       |  double = "<>" OR double = "<=" OR double = ">=" .
                       |
   isquote               |is quote : char = """" .
                         |
                         |
   isniltext             |is niltext : char = "" .
                         |
                         |
   isbegincomment        |is begin comment : char = "{" OR char = "(" AND ahead char = "*" .
                         |
                         |ENDPROC next symbol ;
                         |


nextchar ................|PROC next char :
                         |
                         |  position INCR 1 ;
                         |  char := line SUB position
                         |
                         |ENDPROC next char ;
                         |


skipblanks ..............|PROC skip blanks :
                         |
                         |  position := pos (line, ""33"", ""254"", position) ;
                         |  IF position = 0
                         |    THEN position := LENGTH line + 1
                         |  FI ;
                         |  char := line SUB position .
                         |
                         |ENDPROC skip blanks ;
                         |


aheadchar ...............|TEXT PROC ahead char :
                         |
                         |  line SUB position+1
                         |
                         |ENDPROC ahead char ;
                         |


assemblechars ...........|PROC assemble chars (INT CONST low, high) :
                         |
                         |  INT CONST begin := position ;
                         |  position behind valid text ;
                         |  chars := subtext (line, begin, position-1) ;
                         |  char := line SUB position .
                         |
                         |
   positionbehindvalidtex |position behind valid text :
                         |  position := pos (line, ""32"", code (low-1), begin) ;
                         |  IF position = 0
                         |    THEN position := LENGTH line + 1
                         |  FI ;
                         |  INT CONST higher pos := pos (line, code (high+1), ""254"", begin) ;
                         |  IF higher pos <> 0 AND higher pos < position
                         |    THEN position := higher pos
                         |  FI .
                         |
                         |ENDPROC assemble chars ;
                         |
                         |
readcomment .............|PROC read comment :
                         |
                         |  TEXT VAR last char ;
                         |  comment depth INCR 1 ;
                         |  REP
                         |    last char := char ;
                         |    nextchar ;
                         |    IF is begin comment
                         |      THEN read comment
                         |    FI ;
                         |    IF char = ""
                         |      THEN LEAVE read comment
                         |    FI
                         |  UNTIL is end comment PER ;
                         |  comment depth DECR 1 ;
                         |  next char ;
                         |  skip blanks .
                         |
                         |
   isendcomment          |is end comment :
                         |  char = "}" OR char = ")" AND last char = "*" .
                         |
                         |
   isbegincomment        |is begin comment :
                         |  char = "{" OR char = "(" AND ahead char = "*" .
                         |
                         |ENDPROC read comment ;
                         |
                         |


scan ....................|PROC scan (FILE VAR f) :
                         |
                         |  getline (f, line) ;
                         |  scan (line)
                         |
                         |ENDPROC scan ;
                         |


nextsymbol ..............|PROC next symbol (FILE VAR f, TEXT VAR symbol) :
                         |
                         |  INT VAR type ;
                         |  next symbol (f, symbol, type)
                         |
                         |ENDPROC next symbol ;
                         |
                         |TEXT VAR scanned ;
                         |


nextsymbol ..............|PROC next symbol (FILE VAR f, TEXT VAR symbol, INT VAR type) :
                         |
                         |  next symbol (symbol, type) ;
                         |  WHILE type >= 7 AND NOT eof (f) REP
                         |    getline (f, line) ;
                         |    continue scan (line) ;
                         |    next symbol (scanned, type) ;
                         |    symbol CAT scanned
                         |  PER .
                         |
                         |ENDPROC next symbol ;
```

```
|
|ENDPACKET scanner ;
```

```
                             |
screendescription ********   |PACKET screen description DEFINES
                             |
                             |     xsize, ysize, marksize, mark refresh line mode :
                             |
                             |
                             |INT VAR xs := 80, ys := 24, ms := 1;
                             |

xsize .......................|INT PROC xsize: xs END PROC xsize;
                             |

ysize .......................|INT PROC ysize: ys END PROC ysize;
                             |

marksize ....................|INT PROC marksize: ms END PROC marksize;
                             |

xsize .......................|PROC xsize (INT CONST i): xs := i END PROC xsize;
                             |

ysize .......................|PROC ysize (INT CONST i): ys := i END PROC ysize;
                             |

marksize ....................|PROC marksize (INT CONST i): ms := i END PROC marksize;
                             |
                             |
                             |BOOL VAR line mode := FALSE;
                             |

markrefreshlinemode .........|BOOL PROC mark refresh line mode:
                             |  line mode
                             |END PROC mark refresh line mode;
                             |

markrefreshlinemode .........|PROC mark refresh line mode (BOOL CONST b):
                             |  line mode := b
                             |END PROC mark refresh line mode;
                             |
                             |END PACKET screen description ;
```

```
                           |
tastenverwaltung **********|PACKET  tasten verwaltung  DEFINES
                           |     #009 *)
                           |        (***************)
                           |
                           |        lernsequenz auf taste legen,
                           |        lernsequenz auf taste,
                           |        kommando auf taste legen,
                           |        kommando auf taste,
                           |        taste enthaelt kommando,
                           |        std tastenbelegung :
                           |
                           |
                           |
                           |LET kommandoidentifikation = ""0"" ,
                           |    esc = ""27"" ,
                           |    niltext = "" ,
                           |    hop right left up down cr tab rubin rubout mark esc
                           |           = ""1""2""8""3""10""13""9""11""12""16""27"" ;
                           |
                           |
                           |ROW 256 TEXT VAR belegung;
                           |INT VAR i; FOR i FROM 1 UPTO 256 REP belegung (i) := "" PER;
                           |
                           |std tastenbelegung;
                           |
                           |


lernsequenzauftasteleg ...|PROC lernsequenz auf taste legen (TEXT CONST taste, lernsequenz) :
                           |
                           |  belege (belegung (code (taste) + 1), taste, lernsequenz)
                           |
                           |ENDPROC lernsequenz auf taste legen ;
                           |


belege ....................|PROC belege (TEXT VAR tastenpuffer, TEXT CONST taste, lernsequenz) :
                           |  tastenpuffer := lernsequenz ;
                           |  verhindere rekursives lernen .
                           |
                           |
   verhindererekursivesle  |verhindere rekursives lernen :
                           |  loesche alle folgen esc taste aber nicht esc esc taste ;
                           |  IF taste ist freies sonderzeichen
                           |    THEN change all (tastenpuffer, taste, niltext)
                           |  FI .
                           |
                           |
   loeschealIefolgenescta  |loesche alle folgen esc taste aber nicht esc esc taste :
                           |  INT VAR i := pos (tastenpuffer, esc + taste) ;
                           |  WHILE i > 0 REP
                           |    IF ist esc esc taste
                           |      THEN i INCR 1
                           |      ELSE change (tastenpuffer, i, i+1, niltext)
                           |    FI ;
                           |    i := pos (tastenpuffer, esc + taste, i)
                           |  PER .
                           |
   istescesctaste          |ist esc esc taste :
                           |  (tastenpuffer SUB i-1) = esc AND (tastenpuffer SUB i-2) <> esc .
                           |
                           |
   tasteistfreiessonderze  |taste ist freies sonderzeichen :
                           |  taste < ""32"" AND
                           |  pos (hop right left up down cr tab rubin rubout mark esc, taste)
                           |      0 .
                           |
                           |END PROC belege ;
                           |
                           |


lernsequenzauftaste ......|TEXT PROC lernsequenz auf taste (TEXT CONST taste) :
                          |  IF   taste enthaelt kommando (taste)
                          |  THEN ""
                          |  ELSE belegung (code (taste) + 1)
                          |  FI
                          |END PROC lernsequenz auf taste;
                          |
                          |


kommandoauftastelegen ....|PROC kommando auf taste legen (TEXT CONST taste, kommando) :
                          |
                          |  belegung (code (taste) + 1) := kommandoidentifikation;
                          |  belegung (code (taste) + 1) CAT kommando
                          |
                          |END PROC kommando auf taste legen;
                          |
                          |


kommandoauftaste .........|TEXT PROC kommando auf taste (TEXT CONST taste) :
                          |  IF   taste enthaelt kommando (taste)
                          |  THEN subtext (belegung (code (taste) + 1), 2)
                          |  ELSE ""
                          |  FI
                          |END PROC kommando auf taste;
                          |
                          |


tasteenthaeltkommando ....|BOOL PROC taste enthaelt kommando (TEXT CONST taste) :
                          |  (belegung (code (taste) + 1) SUB 1) = kommandoidentifikation
                          |END PROC taste enthaelt kommando;
                          |
                          |


stdtastenbelegung ........|PROC std tastenbelegung:
                          |  lernsequenz auf taste legen ("(", ""91"");
                          |  lernsequenz auf taste legen (")", ""93"");
                          |  lernsequenz auf taste legen ("<", ""123"");
                          |  lernsequenz auf taste legen (">", ""125"");
                          |  lernsequenz auf taste legen ("A", ""214"");
                          |  lernsequenz auf taste legen ("O", ""215"");
                          |  lernsequenz auf taste legen ("U", ""216"");
                          |  lernsequenz auf taste legen ("a", ""217"");
                          |  lernsequenz auf taste legen ("o", ""218"");
                          |  lernsequenz auf taste legen ("u", ""219"");
```

```
  lernsequenz auf taste legen ("k", ""220"");
  lernsequenz auf taste legen ("-", ""221"");
  lernsequenz auf taste legen ("#", ""222"");
  lernsequenz auf taste legen (" ", ""223"");
  lernsequenz auf taste legen ("B", ""251"");
  lernsequenz auf taste legen ("s", ""251"");
END PROC std tastenbelegung;


END PACKET tasten verwaltung;
```

```
editorpaket ***************|PACKET editor paket DEFINES                             (* EDITOR
                           |     123   *)
                           |       (**********)                                     (* 19.07.85
                           |            -bk-  *)
                           |                                                        (* 10.09.85
                           |     -ws-  *)
                           |                                                        (* 25.04.86
                           |     -sh-  *)
                           |        edit, editget,                                  (* 10.06.86
                           |             -wk-  *)
                           |        quit, quit last,                                (* 04.06.86
                           |             -jl-  *)
                           |        push, type,
                           |        word wrap, margin,
                           |        write permission,
                           |        set busy indicator,
                           |        two bytes,
                           |        is kanji esc,
                           |        within kanji,
                           |        rubin mode,
                           |        is editget,
                           |        editget command,
                           |        getchar,                     nichts neu,
                           |        getcharety,                  satznr neu,
                           |        is incharety,                ueberschrift neu,
                           |        get window,                  zeile neu,
                           |        get editcursor,              abschnitt neu,
                           |        get editline,                bildabschnitt neu,
                           |        put editline,                bild neu,
                           |        aktueller editor,            alles neu,
                           |        groesster editor,            satznr zeigen,
                           |        open editor,                 ueberschrift zeigen,
                           |        editfile,                    bild zeigen:
                           |
                           |
                           |LET     hop            = ""1"",      right          = ""2"",
                           |        up char        = ""3"",      clear eop      = ""4"",
                           |        clear eol      = ""5"",      cursor pos     = ""6"",
                           |        piep           = ""7"",      left           = ""8"",
                           |        down char      = ""10"",     rubin          = ""11"",
                           |        rubout         = ""12"",     cr             = ""13"",
                           |        mark key       = ""16"",     abscr          = ""17"",
                           |        inscr          = ""18"",     dezimal        = ""19"",
                           |        backcr         = ""20"",     esc            = ""27"",
                           |        dach           = ""94"",     blank          = " ";
                           |
                           |
                           |LET     no output     = 0,           out zeichen    = 1,
                           |        out feldrest  = 2,           out feld       = 3,
                           |        clear feldrest = 4;
                           |
                           |LET     FELDSTATUS = STRUCT (INT  stelle, alte stelle, rand, limit,
                           |                                  anfang, marke, laenge, verschoben,
                           |                            BOOL einfuegen, fliesstext, write
                           |                                  access,
                           |                            TEXT tabulator);
                           |FELDSTATUS VAR feldstatus;
                           |
                           |TEXT VAR begin mark := ""15"",
                           |         end mark   := ""14"";
                           |
                           |TEXT VAR separator := "", kommando := "", audit := "", zeichen := "",
                                |        satzrest := "", merksatz := "", alter editsatz := "";
                                |
                                |INT  VAR kommando zeiger := 1, umbruchstelle, umbruch verschoben,
                                |         zeile, spalte, output mode := no output, postblanks := 0,
                                |         min schreibpos, max schreibpos, cpos, absatz ausgleich;
                                |
                                |BOOL VAR lernmodus := FALSE, separator eingestellt := FALSE,
                                |         invertierte darstellung := FALSE, absatzmarke steht,
                                |         cursor diff := FALSE, editget modus := FALSE,
                                |         two byte mode := FALSE, std fliesstext := TRUE,
                                |         editget kommando darf ausgeführt werden := TRUE;.
                                |
                                |
     schirmbreite               |schirmbreite : x size - 1 .
                                |
     schirmhoehe                |schirmhoehe  : y size .
                                |
     maxbreite                  |maxbreite    : schirmbreite - 2 .
                                |
     maxlaenge                  |maxlaenge    : schirmhoehe - 1 .
                                |
     marklength                 |marklength   : mark size .;
                                |
                                |initialisiere editor;
                                |
                                |.initialisiere editor :
                                |  anfang := 1; zeile := 0; verschoben := 0; tabulator := "";
                                |  einfuegen := FALSE; fliesstext := TRUE; zeileneinfuegen := FALSE;
                                |  marke := 0; bildmarke := 0; feldmarke := 0.;
                                |


  editgetcommand ...........    |PROC editget command (BOOL CONST schalter) :
                                |  editget kommando darf ausgeführt werden := schalter
                                |ENDPROC editget command ;
                                |
                                |(**********************************  editget
 +                              |     **********************************)
                                |


  editget ..................    |PROC editget (TEXT VAR editsatz, INT CONST editlimit, editlaenge,
                                |                TEXT CONST sep, res, TEXT VAR exit char) :
                                |  IF editlaenge < 1 THEN errorstop ("Fenster zu klein") FI;
                                |  separator := ""13""; separator CAT sep;
                                |  separator eingestellt := TRUE;
                                |  TEXT VAR reservierte editget tasten := ""11""12"" ;
                                |  reservierte editget tasten CAT res ;
                                |  disable stop;
                                |  absatz ausgleich := 0; exit char := ""; get cursor;
                                |  FELDSTATUS CONST alter feldstatus := feldstatus;
                                |  feldstatus := FELDSTATUS : (1, 1, spalte - 1, editlimit,
                                |                              1, 0, editlaenge, 0,
                                |                              FALSE, FALSE, TRUE, "");
                                |  konstanten neu berechnen;
                                |  output mode := out feld;
                                |  feld editieren;
                                |  zeile verlassen;
                                |  feldstatus := alter feldstatus;
                                |  konstanten neu berechnen;
                                |  separator := "";
                                |  separator eingestellt := FALSE .
```

```
                         |
                         |
feldeditieren            |feld editieren :
                         |  REP
                         |    feldeditor (editsatz, reservierte editget tasten);
                         |    IF   is error
                         |    THEN kommando zeiger := 1; kommando := ""; LEAVE feld editieren
                         |    FI ;
                         |    TEXT VAR t, zeichen; getchar (zeichen);
                         |    IF   zeichen ist separator
                         |    THEN exit char := zeichen; LEAVE feld editieren
                         |    ELIF zeichen = hop
                         |    THEN feldout (editsatz, stelle); getchar (zeichen)
                         |    ELIF zeichen = mark key
                         |    THEN output mode := out feld
                         |    ELIF zeichen = abscr
                         |    THEN exit char := cr; LEAVE feld editieren
                         |    ELIF zeichen = esc
                         |    THEN getchar (zeichen); auf exit pruefen;
                         |         IF   zeichen = rubout
                         |              (*sh*)
                         |         THEN IF   marke > 0
                         |              THEN merksatz := subtext (editsatz, marke, stelle - 1);
                         |                   change (editsatz, marke, stelle - 1, "");
                         |                   stelle := marke; marke := 0; konstanten neu
                         |                        berechnen
                         |              FI
                         |         ELIF zeichen = rubin
                         |         THEN t := subtext (editsatz, 1, stelle - 1);
                         |              t CAT merksatz;
                         |              satzrest := subtext (editsatz, stelle);
                         |              t CAT satzrest;
                         |              stelle INCR LENGTH merksatz;
                         |              merksatz := ""; editsatz := t
                         |         ELIF editget kommando darf ausgeführt werden
                         |                   CAND
                         |              zeichen ist kein esc kommando
                         |                   CAND
                         |              kommando auf taste (zeichen) <> ""
                         |         THEN editget kommando ausfuehren
                         |         FI ;
                         |         output mode := out feld
                         |    FI
                         |  PER .
                         |
                         |
zeichenistkeinesckomma   |zeichen ist kein esc kommando :
                         |    (*wk*)
                         |  pos (hop + left + right, zeichen) = 0 .
                         |
                         |
zeileverlassen           |zeile verlassen :
                         |  IF   marke > 0 OR verschoben <> 0
                         |  THEN stelle DECR verschoben; verschoben := 0; feldout (editsatz, 0);
                         |  ELSE cursor (rand + 1 + min (LENGTH editsatz, editlaenge), zeile)
                         |  FI .
                         |
                         |
zeichenistseparator      |zeichen ist separator : pos (separator, zeichen) > 0 .
                         |
      aufexitpruefen         |auf exit pruefen :
                             |  IF   pos (res, zeichen) > 0
                             |  THEN exit char := esc + zeichen; LEAVE feld editieren
                             |  FI .
                             |
                             |
      editgetkommandoausfueh |editget kommando ausfuehren :
                             |  editget zustaende sichern ;
                             |  do (kommando auf taste (zeichen)) ;
                             |  alte editget zustaende wieder herstellen ;
                             |  IF stelle < marke THEN stelle := marke FI;
                             |  konstanten neu berechnen .
                             |
                             |
      editgetzustaendesicher |editget zustaende sichern :
                             |      (*wk*)
                             |  BOOL VAR alter editget modus := editget modus;
                             |  FELDSTATUS VAR feldstatus vor do kommando := feldstatus ;
                             |  INT VAR zeile vor do kommando := zeile ;
                             |  TEXT VAR separator vor do kommando := separator ;
                             |  BOOL VAR separator eingestellt vor do kommando := separator
                             |         eingestellt ;
                             |  editget modus := TRUE ;
                             |  alter editsatz := editsatz .
                             |
                             |
      alteeditgetzustaendewi |alte editget zustaende wieder herstellen :
                             |  editget modus := alter editget modus ;
                             |  editsatz := alter editsatz;
                             |  feldstatus := feldstatus vor do kommando ;
                             |  zeile := zeile vor do kommando ;
                             |  separator := separator vor do kommando ;
                             |  separator eingestellt := separator eingestellt vor do kommando .
                             |
                             |END PROC editget;
                             |


    editget .................|PROC editget (TEXT VAR editsatz, INT CONST editlimit, TEXT VAR exit
                             |     char) :
                             |  editget (editsatz, editlimit, x size - x cursor, "", "", exit ch
                             |END PROC editget;                                        (* 05.07.84
                             |     -bk- *)
                             |


    editget .................|PROC editget (TEXT VAR editsatz, TEXT CONST sep, res, TEXT VAR exit
                             |     char) :
                             |  editget (editsatz, max text length, x size - x cursor, sep, res,
                             |        exit char)
                             |END PROC editget;                                        (* 05.07.84
                             |     -bk- *)
                             |


    editget .................|PROC editget (TEXT VAR editsatz) :
                             |  TEXT VAR exit char;                                    (* 05.07.84
                             |        -bk- *)
                             |  editget (editsatz, max text length, x size - x cursor, "", "",
                             |        exit char)
                             |END PROC editget;
                             |
editget ..................|PROC editget (TEXT VAR editsatz, INT CONST editlimit, editlaenge) :
                          |  TEXT VAR exit char;
                          |  editget (editsatz, editlimit, editlaenge, "", "", exit char)
                          |ENDPROC editget;
                          |
                          |(*******************************  feldeditor
                          |     *******************************)
                          |
                          |TEXT VAR reservierte feldeditor tasten ;
                          |     (*jl*)
                          |


feldeditor ...............|PROC feldeditor (TEXT VAR satz, TEXT CONST res) :
                          |  enable stop;
                          |  reservierte feldeditor tasten := ""1""2""8"" ;
                          |  reservierte feldeditor tasten CAT res;
                          |  absatzmarke steht := (satz SUB LENGTH satz) = blank;
                          |  alte stelle merken;
                          |  cursor diff bestimmen und ggf ausgleichen;
                          |  feld editieren;
                          |  absatzmarke updaten .
                          |
                          |
   altestellemerken       |alte stelle merken : alte stelle := stelle .
                          |
                          |
   cursordiffbestimmenund |cursor diff bestimmen und ggf ausgleichen :
                          |  IF   cursor diff
                          |  THEN stelle INCR 1; cursor diff := FALSE
                          |  FI ;
                          |  IF   stelle auf zweitem halbzeichen
                          |  THEN stelle DECR 1; cursor diff := TRUE
                          |  FI .
                          |
                          |
   feldeditieren          |feld editieren :
                          |  REP
                          |    feld optisch aufbereiten;
                          |    kommando annehmen und ausfuehren
                          |  PER .
                          |
                          |
   absatzmarkeupdaten     |absatzmarke updaten :
                          |  IF   absatzmarke soll stehen
                          |  THEN IF NOT absatzmarke steht THEN absatzmarke schreiben (TRUE) FI
                          |  ELSE IF     absatzmarke steht THEN absatzmarke schreiben (FALSE) F
                          |  FI .
                          |
                          |
   absatzmarkesollstehen  |absatzmarke soll stehen : (satz SUB LENGTH satz) = blank .
                          |
                          |
   feldoptischaufbereiten |feld optisch aufbereiten :
                          |  stelle korrigieren;
                          |  verschieben wenn erforderlich;
                          |  randausgleich fuer doppelzeichen;
                          |  output mode behandeln;
                          |  ausgabe verhindern .
                          |
randausgleichfuerdoppe |randausgleich fuer doppelzeichen :
                       |  IF   stelle = max schreibpos CAND stelle auf erstem halbzeichen
                       |  THEN verschiebe (1)
                       |  FI .
                       |
                       |
stellekorrigieren      |stelle korrigieren :
                       |  IF stelle auf zweitem halbzeichen THEN stelle DECR 1 FI .
                       |
                       |
stelleauferstemhalbzei |stelle auf erstem halbzeichen  : within kanji (satz, stelle + 1) .
                       |
                       |
stelleaufzweitemhalbze |stelle auf zweitem halbzeichen : within kanji (satz, stelle) .
                       |
                       |
outputmodebehandeln    |output mode behandeln :
                       |  SELECT output mode OF
                       |    CASE no output      : im markiermode markierung anpassen
                       |    CASE out zeichen    : zeichen ausgeben; LEAVE output mode
                       |         behandeln
                       |    CASE out feldrest   : feldrest neu schreiben
                       |    CASE out feld       : feldout (satz, stelle)
                       |    CASE clear feldrest : feldrest loeschen
                       |  END SELECT;
                       |  schreibmarke positionieren (stelle) .
                       |
                       |
ausgabeverhindern      |ausgabe verhindern : output mode := no output .
                       |
                       |
immarkiermodemarkierun |im markiermode markierung anpassen :
                       |  IF markiert THEN markierung anpassen FI .
                       |
                       |
markierunganpassen     |markierung anpassen :
                       |  IF   stelle > alte stelle
                       |  THEN markierung verlaengern
                       |  ELIF stelle < alte stelle
                       |  THEN markierung verkuerzen
                       |  FI .
                       |
                       |
markierungverlaengern  |markierung verlaengern :
                       |  invers out (satz, alte stelle, stelle, "", end mark) .
                       |
                       |
markierungverkuerzen   |markierung verkuerzen :
                       |  invers out (satz, stelle, alte stelle, end mark, "") .
                       |
                       |
zeichenausgeben        |zeichen ausgeben :
                       |  IF   NOT markiert
                       |  THEN out (zeichen)
                       |  ELIF mark refresh line mode
                       |  THEN feldout (satz, stelle); schreibmarke positionieren (stelle)
                       |  ELSE out (begin mark); markleft; out (zeichen); out (end mark);
                       |       markleft
                       |  FI .
                       |
markleft                 |markleft :
                         |  marklength TIMESOUT left .
                         |
                         |
feldrestneuschreiben     |feldrest neu schreiben :
                         |  IF   NOT markiert
                         |  THEN feldrest unmarkiert neu schreiben
                         |  ELSE feldrest markiert neu schreiben
                         |  FI ;
                         |  WHILE postblanks > 0 CAND x cursor <= rand + laenge REP
                         |    out (blank); postblanks DECR 1
                         |  PER ; postblanks := 0 .
                         |
                         |
feldrestunmarkiertneus   |feldrest unmarkiert neu schreiben :
                         |  schreibmarke positionieren (alte stelle);
                         |  out subtext mit randbehandlung (satz, alte stelle, stelle am ende)
                         |       .
                         |
                         |
feldrestmarkiertneusch   |feldrest markiert neu schreiben :
                         |  markierung verlaengern; out subtext mit randbehandlung
                         |                            (satz, stelle, stelle am ende - 2 *
                         |                                 marklength) .
                         |
                         |
kommandoannehmenundaus   |kommando annehmen und ausfuehren :
                         |  kommando annehmen; kommando ausfuehren .
                         |
                         |
kommandoannehmen         |kommando annehmen :
                         |  getchar (zeichen); kommando zurueckweisen falls noetig .
                         |
                         |
kommandozurueckweisenf   |kommando zurueckweisen falls noetig :
                         |  IF   NOT write access CAND zeichen ist druckbar
                         |  THEN benutzer warnen; kommando ignorieren
                         |  FI .
                         |
                         |
benutzerwarnen           |benutzer warnen : out (piep) .
                         |
                         |
kommandoignorieren       |kommando ignorieren :
                         |  zeichen := ""; LEAVE kommando annehmen und ausfuehren .
                         |
                         |
kommandoausfuehren       |kommando ausfuehren :
                         |  neue satzlaenge bestimmen;
                         |  alte stelle merken;
                         |  IF   zeichen ist separator
                         |  THEN feldeditor verlassen
                         |  ELIF zeichen ist druckbar
                         |  THEN fortschreiben
                         |  ELSE funktionstasten behandeln
                         |  FI .
                         |
                         |
neuesatzlaengebestimme   |neue satzlaenge bestimmen : INT VAR satzlaenge := LENGTH satz .
                         |
feldeditorverlassen     |feldeditor verlassen :
                        |  IF NOT absatzmarke steht THEN blanks abschneiden FI;
                        |       (*sh*)
                        |  push (zeichen); LEAVE feld editieren .
                        |
                        |
blanksabschneiden       |blanks abschneiden :
                        |  INT VAR letzte non blank pos := satzlaenge;
                        |  WHILE letzte non blank pos > 0 CAND (satz SUB letzte non blank
                        |       pos) = blank REP
                        |    letzte non blank pos DECR 1
                        |  PER; satz := subtext (satz, 1, letzte non blank pos) .
                        |
                        |
zeichenistdruckbar      |zeichen ist druckbar : zeichen >= blank .
                        |
                        |
zeichenistseparator     |zeichen ist separator :
                        |  separator eingestellt CAND pos (separator, zeichen) > 0 .
                        |
                        |
fortschreiben           |fortschreiben :
                        |  zeichen in satz eintragen;
                        |  IF is kanji esc (zeichen) THEN kanji zeichen schreiben FI;
                        |  bei erreichen von limit ueberlauf behandeln .
                        |
                        |
zeicheninsatzeintragen  |zeichen in satz eintragen :
                        |  IF   hinter dem satz
                        |  THEN satz mit leerzeichen auffuellen und zeichen anfuegen
                        |  ELIF einfuegen
                        |  THEN zeichen vor aktueller position einfuegen
                        |  ELSE altes zeichen ersetzen
                        |  FI .
                        |
                        |
hinterdemsatz           |hinter dem satz : stelle > satzlaenge .
                        |
                        |
satzmitleerzeichenauff  |satz mit leerzeichen auffuellen und zeichen anfuegen :
                        |  satz AUFFUELLENMIT blank;
                        |  zeichen anfuegen;
                        |  output mode := out zeichen .
                        |
                        |
zeichenanfuegen         |zeichen anfuegen   : satz CAT zeichen; neue satzlaenge bestimmen .
                        |
zeichenignorieren       |zeichen ignorieren : benutzer warnen; LEAVE kommando ausfuehren .
                        |
                        |
zeichenvoraktuellerpos  |zeichen vor aktueller position einfuegen :
                        |  insert char (satz, zeichen, stelle);
                        |  neue satzlaenge bestimmen;
                        |  output mode := out feldrest .
                        |
                        |
alteszeichenersetzen    |altes zeichen ersetzen :
                        |  replace (satz, stelle, zeichen);
                        |  IF   stelle auf erstem halbzeichen
                        |  THEN output mode := out feldrest; replace (satz, stelle + 1, blank)
                        |  ELSE output mode := out zeichen
                        |  FI .
```

```
                         |
                         |
kanjizeichenschreiben    |kanji zeichen schreiben :
                         |  alte stelle merken;
                         |  stelle INCR 1; getchar (zeichen);
                         |  IF   zeichen < ""64"" THEN zeichen := ""64"" FI;
                         |  IF   hinter dem satz
                         |  THEN zeichen anfuegen
                         |  ELIF einfuegen
                         |  THEN zeichen vor aktueller position einfuegen
                         |  ELSE replace (satz, stelle, zeichen)
                         |  FI ;
                         |  output mode := out feldrest .
                         |
                         |
beierreichenvonlimitue   |bei erreichen von limit ueberlauf behandeln :
                         |     (*sh*)
                         |  IF   satzlaenge kritisch
                         |  THEN in naechste zeile falls moeglich
                         |  ELSE stelle INCR 1
                         |  FI .
                         |
                         |
satzlaengekritisch       |satzlaenge kritisch :
                         |  IF   stelle >= satzlaenge
                         |  THEN satzlaenge = limit
                         |  ELSE satzlaenge = limit + 1
                         |  FI .
                         |
                         |
innaechstezeilefallsmo   |in naechste zeile falls moeglich :
                         |  IF   fliesstext AND umbruch moeglich OR NOT fliesstext AND stelle
                         |       >= satzlaenge
                         |  THEN in naechste zeile
                         |  ELSE stelle INCR 1
                         |  FI .
                         |
                         |
umbruchmoeglich          |umbruch moeglich :
                         |  INT CONST st := stelle; stelle := limit;
                         |  INT CONST ltzt wortanf := letzter wortanfang (satz);
                         |  stelle := st; einrueckposition (satz) < ltzt wortanf .
                         |
                         |
innaechstezeile          |in naechste zeile :
                         |  IF   fliesstext
                         |  THEN ueberlauf und oder umbruch
                         |  ELSE ueberlauf ohne umbruch
                         |  FI .
                         |
                         |
ueberlaufundoderumbruc   |ueberlauf und oder umbruch :
                         |  INT VAR umbruchpos := 1;
                         |  umbruchposition bestimmen;
                         |  loeschposition bestimmen;
                         |  IF   stelle = satzlaenge
                         |  THEN ueberlauf mit oder ohne umbruch
                         |  ELSE umbruch mit oder ohne ueberlauf
                         |  FI .
                         |
umbruchpositionbestimm |umbruchposition bestimmen :
                       |  umbruchstelle := stelle;
                       |  stelle := satzlaenge;
                       |  umbruchpos := max (umbruchpos, letzter wortanfang (satz));
                       |  stelle := umbruchstelle .
                       |
                       |
loeschpositionbestimme |loeschposition bestimmen :
                       |  INT VAR loeschpos := umbruchpos;
                       |  WHILE davor noch blank REP loeschpos DECR 1 PER .
                       |
                       |
davornochblank         |davor noch blank :
                       |  loeschpos > ganz links CAND (satz SUB (loeschpos - 1)) = blank .
                       |
                       |
ganzlinks              |ganz links : max (1, marke) .
                       |
                       |
ueberlaufmitoderohneum |ueberlauf mit oder ohne umbruch :
                       |  IF   zeichen = blank OR loeschpos = ganz links
                       |  THEN stelle := 1; ueberlauf ohne umbruch
                       |  ELSE ueberlauf mit umbruch
                       |  FI .
                       |
                       |
ueberlaufohneumbruch   |ueberlauf ohne umbruch : push (cr) .
                       |
                       |
ueberlaufmitumbruch    |ueberlauf mit umbruch :
                       |  ausgabe verhindern;
                       |  umbruchkommando aufbereiten;
                       |  auf loeschposition positionieren .
                       |
                       |
umbruchkommandoaufbere |umbruchkommando aufbereiten :
                       |  zeichen := hop + rubout + inscr;
                       |  satzrest := subtext (satz, umbruchpos);
                       |  zeichen CAT satzrest;
                       |  IF   stelle ist im umgebrochenen teil
                       |  THEN insert char (zeichen, backcr, max (stelle - umbruchpos + 1,
  +                    |       0) + 4);
                       |       zeichen CAT backcr
                       |  FI ;
                       |  push (zeichen) .
                       |
                       |
stelleistimumgebrochen |stelle ist im umgebrochenen teil : stelle >= loeschpos .
                       |
                       |
aufloeschpositionposit |auf loeschposition positionieren : stelle := loeschpos .
                       |
                       |
umbruchmitoderohneuebe |umbruch mit oder ohne ueberlauf :
                       |  umbruchposition anpassen;
                       |  IF   stelle ist im umgebrochenen teil
                       |  THEN umbruch mit ueberlauf
                       |  ELSE umbruch ohne ueberlauf
                       |  FI .
                       |
umbruchpositionanpasse |umbruchposition anpassen :
                       |  IF   zeichen = blank
                       |  THEN umbruchpos := stelle + 1;
                       |       umbruchposition bestimmen;
                       |       neue loeschposition bestimmen
                       |  FI .
                       |
                       |
neueloeschpositionbest |neue loeschposition bestimmen :
                       |  loeschpos := umbruchpos;
                       |  WHILE davor noch blank AND stelle noch nicht erreicht REP
                       |       loeschpos DECR 1 PER .
                       |
                       |
stellenochnichterreich |stelle noch nicht erreicht : loeschpos > stelle + 1 .
                       |
                       |
umbruchmitueberlauf    |umbruch mit ueberlauf : ueberlauf mit umbruch .
                       |
                       |
umbruchohneueberlauf   |umbruch ohne ueberlauf :
                       |  zeichen := inscr;
                       |  satzrest := subtext (satz, umbruchpos);
                       |  zeichen CAT satzrest;
                       |  zeichen CAT up char + backcr;
                       |  umbruchstelle INCR 1; umbruch verschoben := verschoben;
                       |  satz := subtext (satz, 1, loeschpos - 1);
                       |  schreibmarke positionieren (loeschpos); feldrest loeschen;
                       |  output mode := out feldrest;
                       |  push (zeichen) .
                       |
                       |
funktionstastenbehande |funktionstasten behandeln :
                       |  SELECT pos (kommandos, zeichen) OF
                       |    CASE c hop     : hop kommandos behandeln
                       |    CASE c esc     : esc kommandos behandeln
                       |    CASE c right   : nach rechts oder ueberlauf
                       |    CASE c left    : wenn moeglich ein schritt nach links
                       |    CASE c tab     : zur naechsten tabulator position
                       |    CASE c dezimal : dezimalen schreiben
                       |    CASE c rubin   : einfuegen umschalten
                       |    CASE c rubout  : ein zeichen loeschen
                       |    CASE c abscr, c inscr, c down : feldeditor verlassen
                       |    CASE c up      : eine zeile nach oben
                       |         (*sh*)
                       |    CASE c cr      : ggf absatz erzeugen
                       |    CASE c mark    : markieren umschalten
                       |    CASE c backcr  : zurueck zur umbruchstelle
                       |    OTHERWISE      : sondertaste behandeln
                       |  END SELECT .
                       |
                       |
kommandos              |kommandos :
                       |  LET c hop     =  1,              c right   =  2,
                       |      c up      =  3,              c left    =  4,
                       |      c tab     =  5,              c down    =  6,
                       |      c rubin   =  7,              c rubout  =  8,
                       |      c cr      =  9,              c mark    = 10,
                       |      c abscr   = 11,              c inscr   = 12,
                       |      c dezimal = 13,              c esc     = 14,
                       |      c backcr  = 15;
                       |
                          | ""1""2""3""8""9""10""11""12""13""16""17""18""19""27""20"" .
                          |
                          |
dezimalenschreiben        |dezimalen schreiben : IF write access THEN dezimaleditor (satz) FI .
                          |
                          |
zurueckzurumbruchstell    |zurueck zur umbruchstelle:
                          |  IF   umbruch stelle > 0 THEN stelle := umbruch stelle FI;
                          |  IF   verschoben <> umbruch verschoben
                          |  THEN verschoben := umbruch verschoben; output mode := out feld
                          |  FI .
                          |
                          |
hopkommandosbehandeln     |hop kommandos behandeln :
                          |  TEXT VAR zweites zeichen; getchar (zweites zeichen);
                          |  zeichen CAT zweites zeichen;
                          |  SELECT pos (hop kommandos, zweites zeichen) OF
                          |    CASE h hop    : nach links oben
                          |    CASE h right  : nach rechts blaettern
                          |    CASE h left   : nach links blaettern
                          |    CASE h tab    : tab position definieren oder loeschen
                          |    CASE h rubin  : zeile splitten
                          |    CASE h rubout : loeschen oder rekombinieren
                          |    CASE h cr, h up, h down : feldeditor verlassen
                          |    OTHERWISE     : zeichen ignorieren
                          |  END SELECT .
                          |
                          |
hopkommandos              |hop kommandos :
                          |  LET h hop    = 1,                h right  = 2,
                          |      h up     = 3,                h left   = 4,
                          |      h tab    = 5,                h down   = 6,
                          |      h rubin  = 7,                h rubout = 8,
                          |      h cr     = 9;
                          |
                          | ""1""2""3""8""9""10""11""12""13"" .
                          |
                          |
nachlinksoben             |nach links oben :
                          |  stelle := max (marke, anfang) + verschoben; feldeditor verlassen
                          |
                          |
nachrechtsblaettern       |nach rechts blaettern :
                          |  INT CONST rechter rand := stelle am ende - markierausgleich;
                          |  IF   stelle ist am rechten rand
                          |  THEN stelle INCR laenge - 2 * markierausgleich + ausgleich fuer
   +                      |       doppelzeichen
                          |  ELSE stelle := rechter rand
                          |  FI ;
                          |  IF satzlaenge <= limit THEN stelle := min (stelle, limit) FI;
                          |  alte einrueckposition mitziehen .
                          |
                          |
stelleistamrechtenrand    |stelle ist am rechten rand :
                          |  stelle auf erstem halbzeichen CAND stelle = rechter rand - 1
                          |                                COR  stelle = rechter rand .
                          |
                          |
ausgleichfuerdoppelzei    |ausgleich fuer doppelzeichen : stelle - rechter rand .
                          |
 605     nachlinksblaettern      |nach links blaettern :
 606                             |  INT CONST linker rand := stelle am anfang;
 607                             |  IF   stelle = linker rand
 608                             |  THEN stelle DECR laenge - 2 * markierausgleich
 609                             |  ELSE stelle := linker rand
 610                             |  FI ;
 611                             |  stelle := max (ganz links, stelle);
 612                             |  alte einrueckposition mitziehen .
 613                             |
                                 |
 614     tabpositiondefiniereno  |tab position definieren oder loeschen :
 615                             |  IF   stelle > LENGTH tabulator
 616                             |  THEN tabulator AUFFUELLENMIT right; tabulator CAT dach
 617                             |  ELSE replace (tabulator, stelle, neues tab zeichen)
 618                             |  FI ;
 619                             |  feldeditor verlassen .
 620                             |
                                 |
 621     neuestabzeichen         |neues tab zeichen :
 622                             |  IF (tabulator SUB stelle) = right THEN dach ELSE right FI .
 623                             |
                                 |
 624     zeilesplitten           |zeile splitten :
 625                             |  IF write access THEN feldeditor verlassen ELSE zeichen ignorieren
  +                              |       FI .
 626                             |
                                 |
 627     loeschenoderrekombinie  |loeschen oder rekombinieren :
 628                             |  IF   NOT write access
 629                             |  THEN zeichen ignorieren
 630                             |  ELIF hinter dem satz
 631                             |  THEN zeilen rekombinieren
 632                             |  ELIF auf erstem zeichen
 633                             |  THEN ganze zeile loeschen
 634                             |  ELSE zeilenrest loeschen
 635                             |  FI .
 636                             |
                                 |
 637     zeilenrekombinieren     |zeilen rekombinieren : feldeditor verlassen .
                                 |
 638     auferstemzeichen        |auf erstem zeichen   : stelle = 1 .
                                 |
 639     ganzezeileloeschen      |ganze zeile loeschen : satz := ""; feldeditor verlassen .
 640                             |
                                 |
 641     zeilenrestloeschen      |zeilenrest loeschen :
 642                             |  change (satz, stelle, satzlaenge, "");
 643                             |  output mode := clear feldrest .
 644                             |
                                 |
 645     esckommandosbehandeln   |esc kommandos behandeln :
 646                             |  getchar (zweites zeichen);
 647                             |  zeichen CAT zweites zeichen;
 648                             |  auf exit pruefen;
 649                             |  SELECT pos (esc kommandos, zweites zeichen) OF
 650                             |    CASE e hop   : lernmodus umschalten
 651                             |    CASE e right : zum naechsten wort
 652                             |    CASE e left  : zum vorigen wort
 653                             |    OTHERWISE    : belegte taste ausfuehren
 654                             |  END SELECT .
 655                             |
aufexitpruefen          |auf exit pruefen :
                        |  IF pos (res, zweites zeichen) > 0 THEN feldeditor verlassen FI .
                        |
                        |
esckommandos            |esc kommandos :
                        |  LET e hop      =  1,
                        |      e right    =  2,
                        |      e left     =  3;
                        |
                        |  ""1""2""8"" .
                        |
                        |
lernmodusumschalten     |lernmodus umschalten :
                        |  IF lernmodus THEN lernmodus ausschalten ELSE lernmodus einschalten
                        |       FI;
                        |  feldeditor verlassen .
                        |
                        |
lernmodusausschalten    |lernmodus ausschalten :
                        |  lernmodus := FALSE;
                        |  belegbare taste erfragen;
                        |  audit := subtext (audit, 1, LENGTH audit - 2);
                        |  IF   taste = hop
                        |  THEN (* lernsequenz nicht auf taste legen *)          (* 16.08.85
                        |       -ws- *)
                        |  ELSE lernsequenz auf taste legen (taste, audit)
                        |  FI ;
                        |  audit := "" .
                        |
                        |
belegbaretasteerfragen  |belegbare taste erfragen :
                        |  TEXT VAR taste; getchar (taste);
                        |  WHILE taste ist reserviert REP
                        |    benutzer warnen; getchar (taste)
                        |  PER .
                        |
                        |
tasteistreserviert      |taste ist reserviert :                                  (* 16.08.85
                        |      -ws- *)
                        |  taste <> hop CAND pos (reservierte feldeditor tasten, taste) > 0 .
                        |
                        |
lernmoduseinschalten    |lernmodus einschalten : audit := ""; lernmodus := TRUE .
                        |
                        |
zumvorigenwort          |zum vorigen wort :
                        |  IF   stelle > 1
                        |  THEN stelle DECR 1; stelle := letzter wortanfang (satz);
                        |       alte einrueckposition mitziehen;
                        |       IF (satz SUB stelle) <> blank THEN LEAVE zum vorigen wort FI
                        |  FI ;
                        |  feldeditor verlassen .
                        |
                        |
zumnaechstenwort        |zum naechsten wort :
                        |  IF kein naechstes wort THEN feldeditor verlassen FI .
                        |
                        |
keinnaechsteswort       |kein naechstes wort :
                        |  BOOL VAR im alten wort := TRUE;
                        |  INT VAR i;
                        |  FOR i FROM stelle UPTO satzlaenge REP
```

```
                         |    IF   im alten wort
                         |    THEN im alten wort := (satz SUB i) <> blank
                         |    ELIF (satz SUB i) <> blank
                         |    THEN stelle := i; LEAVE kein naechstes wort WITH FALSE
                         |    FI
                         |  PER;
                         |  TRUE .
                         |
                         |
belegtetasteausfuehren   |belegte taste ausfuehren :
                         |  IF   ist kommando taste
                         |  THEN feldeditor verlassen
                         |  ELSE gelerntes ausfuehren
                         |  FI .
                         |
                         |
istkommandotaste         |ist kommando taste : taste enthaelt kommando (zweites zeichen) .
                         |
                         |
gelerntesausfuehren      |gelerntes ausfuehren :
                         |  push (lernsequenz auf taste (zweites zeichen)) .
                         |       (*sh*)
                         |
                         |
nachrechtsoderueberlau   |nach rechts oder ueberlauf :
                         |  IF   fliesstext COR stelle < limit OR satzlaenge > limit
                         |  THEN nach rechts
                         |  ELSE auf anfang der naechsten zeile
                         |  FI .
                         |
                         |
nachrechts               |nach rechts :
                         |  IF stelle auf erstem halbzeichen THEN stelle INCR 2 ELSE stelle
                         |       INCR 1 FI;
                         |  alte einrueckposition mitziehen .
                         |
                         |
aufanfangdernaechstenz   |auf anfang der naechsten zeile : push (abscr) .
                         |
                         |
nachlinks                |nach links : stelle DECR 1; alte einrueckposition mitziehen .
                         |
                         |
alteeinrueckpositionmi   |alte einrueckposition mitziehen :
                         |  IF   satz ist leerzeile
                         |  THEN alte einrueckposition := stelle
                         |  ELSE alte einrueckposition := min (stelle, einrueckposition (satz))
                         |  FI .
                         |
                         |
satzistleerzeile         |satz ist leerzeile :
                         |  satz = "" OR satz = blank .
                         |
                         |
wennmoeglicheinschritt   |wenn moeglich ein schritt nach links :
                         |  IF   stelle = ganz links
                         |  THEN zeichen ignorieren
                         |  ELSE nach links
                         |  FI .
                         |
zurnaechstentabulatorp |zur naechsten tabulator position :
                       |  bestimme naechste explizite tabulator position;
                       |  IF   tabulator gefunden
                       |  THEN explizit tabulieren
                       |  ELIF stelle <= satzlaenge
                       |  THEN implizit tabulieren
                       |  ELSE auf anfang der naechsten zeile
                       |  FI .
                       |
                       |
bestimmenaechsteexpliz |bestimme naechste explizite tabulator position :
                       |  INT VAR tab position := pos (tabulator, dach, stelle + 1);
                       |  IF   tab position > limit AND satzlaenge <= limit
                       |  THEN tab position := 0
                       |  FI .
                       |
                       |
tabulatorgefunden      |tabulator gefunden : tab position <> 0 .
                       |
                       |
explizittabulieren     |explizit tabulieren : stelle := tab position; push (dezimal) .
                       |
                       |
implizittabulieren     |implizit tabulieren :
                       |  tab position := einrueckposition (satz);
                       |  IF   stelle <  tab position
                       |  THEN stelle := tab position
                       |  ELSE stelle := satzlaenge + 1
                       |  FI .
                       |
                       |
einfuegenumschalten    |einfuegen umschalten :
                       |  IF NOT write access THEN zeichen ignorieren FI;
                       |       (*sh*)
                       |  einfuegen := NOT einfuegen;
                       |  IF einfuegen THEN einfuegen optisch anzeigen FI;
                       |  feldeditor verlassen .
                       |
                       |
einfuegenoptischanzeig |einfuegen optisch anzeigen :
                       |  IF   markiert
                       |  THEN out (begin mark); markleft; out (dach left); warten;
                       |       out (end mark); markleft
                       |  ELSE out (dach left); warten;
                       |       IF   stelle auf erstem halbzeichen
                       |       THEN out text (satz, stelle, stelle + 1)
                       |       ELSE out text (satz, stelle, stelle)
                       |       FI
                       |  FI .
                       |
                       |
markiert               |markiert  : marke > 0 .
                       |
dachleft               |dach left : ""94""8"" .
                       |
                       |
warten                 |warten :
                       |  TEXT VAR t := incharety (2);
                       |  kommando CAT t; IF lernmodus THEN audit CAT t FI .
                       |
einzeichenloeschen       |ein zeichen loeschen :
                         |  IF   NOT write access THEN zeichen ignorieren FI;
                         |       (*sh*)
                         |  IF   zeichen davor soll geloescht werden
                         |  THEN nach links oder ignorieren
                         |  FI ;
                         |  IF   NOT hinter dem satz THEN aktuelles zeichen loeschen FI .
                         |
                         |
zeichendavorsollgeloes   |zeichen davor soll geloescht werden :
                         |  hinter dem satz COR markiert .
                         |
                         |
nachlinksoderignoriere   |nach links oder ignorieren :
                         |  IF   stelle > ganz links
                         |  THEN nach links
                         |       (*sh*)
                         |  ELSE zeichen ignorieren
                         |  FI .
                         |
                         |
aktuelleszeichenloesch   |aktuelles zeichen loeschen :
                         |  stelle korrigieren; alte stelle merken;
                         |  IF   stelle auf erstem halbzeichen
                         |  THEN delete char (satz, stelle);
                         |       postblanks INCR 1
                         |  FI ;
                         |  delete char (satz, stelle);
                         |  postblanks INCR 1;
                         |  neue satzlaenge bestimmen;
                         |  output mode := out feldrest .
                         |
                         |
einezeilenachoben        |eine zeile nach oben :
                         |     (*sh*)
                         |  IF   NOT absatzmarke steht CAND NOT ist teil eines
                         |       umbruchkommandos
                         |  THEN blanks abschneiden
                         |  FI ;
                         |  push (zeichen); LEAVE feld editieren .
                         |
                         |
istteileinesumbruchkom   |ist teil eines umbruchkommandos : (kommando SUB kommandozeiger) =
                         |     backcr .
                         |
                         |
ggfabsatzerzeugen        |ggf absatz erzeugen :
                         |     (*sh*)
                         |  IF   write access
                         |  THEN IF   NOT absatzmarke steht THEN blanks abschneiden FI;
                         |       IF   stelle > LENGTH satz AND fliesstext AND (satz SUB LENGTH
                         |            satz) <> blank
                         |       THEN satz CAT blank
                         |       FI
                         |  FI ; push (zeichen); LEAVE feld editieren .
                         |
                         |
markierenumschalten      |markieren umschalten :
                         |  IF   markiert
                         |  THEN marke := 0;      maxschreibpos INCR marklength; cpos DECR
                         |       marklength
                         |  ELSE marke := stelle; maxschreibpos DECR marklength; cpos INCR
                         |       marklength;
```

```
                           |       verschieben wenn erforderlich
                           |  FI ;
                           |  feldeditor verlassen .
                           |
                           |
   sondertastebehandeln    |sondertaste behandeln : push (esc + zeichen) .
                           |END PROC feldeditor;
                           |

dezimaleditor ............ |PROC dezimaleditor (TEXT VAR satz) :
                           |  INT VAR dezimalanfang := stelle;
                           |  zeichen einlesen;
                           |  IF dezimalstartzeichen CAND ueberschreibbar THEN dezimalen
                           |       schreiben FI;
                           |  push (zeichen) .
                           |
                           |
   zeicheneinlesen         |zeichen einlesen    : TEXT VAR zeichen; getchar (zeichen) .
                           |
   dezimalzeichen          |dezimalzeichen      : pos (dezimalen, zeichen) > 0 AND nicht
                           |     separator .
                           |
   dezimalstartzeichen     |dezimalstartzeichen : pos (startdezimalen, zeichen) > 0 AND nicht
                           |     separator .
                           |
   dezimalen               |dezimalen           : "0123456789" .
                           |
   startdezimalen          |startdezimalen      : "+-0123456789" .
                           |
   nichtseparator          |nicht separator     : pos (separator, zeichen) = 0 .
                           |
                           |
   ueberschreibbar         |ueberschreibbar :
                           |  dezimalanfang > LENGTH satz OR
                           |  pos (ueberschreibbare zeichen, satz SUB dezimalanfang) > 0 .
                           |
                           |
   ueberschreibbarezeiche  |ueberschreibbare zeichen : " ,.+-0123456789" .
                           |
                           |
   dezimalenschreiben      |dezimalen schreiben :
                           |  REP
                           |    dezimale in satz eintragen;
                           |    dezimalen zeigen;
                           |    zeichen einlesen;
                           |    dezimalanfang DECR 1
                           |  UNTIL dezimaleditor beendet PER;
                           |  stelle INCR 1 .
                           |
                           |
   dezimaleinsatzeintrage  |dezimale in satz eintragen :
                           |  IF   dezimalanfang > LENGTH satz
                           |  THEN satz AUFFUELLENMIT blank; satz CAT zeichen
                           |  ELSE delete char (satz, dezimalanfang); insert char (satz,
                           |       zeichen, stelle)
                           |  FI .
                           |
                           |
   dezimalenzeigen         |dezimalen zeigen :
                           |  INT VAR min dezimalschreibpos := max (min schreibpos,
                           |       dezimalanfang);
```

```
                                |  IF markiert THEN markiert zeigen ELSE unmarkiert zeigen FI;
                                |  schreibmarke positionieren (stelle) .
                                |
                                |
      markiert                  |markiert : marke > 0 .
                                |
                                |
      markiertzeigen            |markiert zeigen :
                                |  invers out (satz, min dezimalschreibpos, stelle, "", end mark);
                                |  out (zeichen) .
                                |
                                |
      unmarkiertzeigen          |unmarkiert zeigen :
                                |  schreibmarke positionieren (min dezimalschreibpos);
                                |  out subtext (satz, min dezimalschreibpos, stelle) .
                                |
                                |
      dezimaleditorbeendet      |dezimaleditor beendet :
                                |  NOT dezimalzeichen OR
                                |  dezimalanfang < max (min schreibpos, marke) OR
                                |  NOT ueberschreibbar .
                                |END PROC dezimaleditor;
                                |


   iseditget ...................|BOOL PROC is editget :
                                |  editget modus
                                |END PROC is editget ;
                                |


   geteditline .................|PROC get editline (TEXT VAR editline, INT VAR editpos, editmarke) :
                                |  IF   editget modus
                                |  THEN editline := alter editsatz;
                                |       editpos := stelle
                                |  FI ;
                                |  editmarke := marke
                                |END PROC get editline;
                                |


   puteditline .................|PROC put editline (TEXT CONST editline, INT CONST editpos,
                                |     editmarke) :
                                |  IF   editget modus
                                |  THEN alter editsatz := editline;
                                |       stelle := max (editpos, 1);
                                |       marke := max (editmarke, 0)
                                |  FI
                                |END PROC put editline;
                                |


   withinkanji .................|BOOL PROC within kanji (TEXT CONST satz, INT CONST stelle) :
                                |  count directly prefixing kanji esc bytes;
                                |  number of kanji esc bytes is odd .
                                |
                                |
      countdirectlyprefixing    |count directly prefixing kanji esc bytes :
                                |  INT VAR pos := stelle - 1, kanji esc bytes := 0;
                                |  WHILE pos > 0 CAND is kanji esc (satz SUB pos) REP
                                |    kanji esc bytes INCR 1; pos DECR 1
                                |  PER .
```

```
 942                              |
                                  |
 943      numberofkanjiescbytesi  |number of kanji esc bytes is odd :
 944                              |  (kanji esc bytes AND 1) <> 0 .
 945                              |END PROC within kanji;
 946                              |


 947   iskanjiesc ................|BOOL PROC is kanji esc (TEXT CONST char) :
  +                               |     (*sh*)
 948                              |  two byte mode CAND
 949                              |  (char >= ""129"" AND char <= ""159"" OR char >= ""224"" AND char
  +                               |       <= ""239"")
 950                              |END PROC is kanji esc;
 951                              |


 952   twobytes ..................|BOOL PROC two bytes : two byte mode END PROC two bytes;
 953                              |


 954   twobytes ..................|PROC two bytes (BOOL CONST new mode) :
 955                              |  two byte mode := new mode
 956                              |END PROC two bytes;
 957                              |


 958   outtext ...................|PROC outtext (TEXT CONST source, INT CONST from, to) :
 959                              |  out subtext mit randbehandlung (source, from, to);
 960                              |  INT VAR trailing;
 961                              |  IF   from <= LENGTH source
 962                              |  THEN trailing := to - LENGTH source
 963                              |  ELSE trailing := to - from + 1
 964                              |  FI ; trailing TIMESOUT blank
 965                              |END PROC outtext;
 966                              |


 967   outsubtextmitrandbehan ... |PROC out subtext mit randbehandlung (TEXT CONST satz, INT CONST von,
  +                               |     bis) :
 968                              |  IF   von > bis
 969                              |  THEN
 970                              |  ELIF bis >= LENGTH satz COR NOT within kanji (satz, bis + 1)
 971                              |  THEN out subtext mit anfangsbehandlung (satz, von, bis)
 972                              |  ELSE out subtext mit anfangsbehandlung (satz, von, bis - 1); out
  +                               |       (blank)
 973                              |  FI
 974                              |END PROC out subtext mit randbehandlung;
 975                              |


 976   outsubtextmitanfangsbe ... |PROC out subtext mit anfangsbehandlung (TEXT CONST satz, INT CONST
  +                               |     von, bis) :
 977                              |  IF   von > bis
 978                              |  THEN
 979                              |  ELIF von = 1 COR NOT within kanji (satz, von)
 980                              |  THEN out subtext (satz, von, bis)
 981                              |  ELSE out (blank); out subtext (satz, von + 1, bis)
 982                              |  FI
 983                              |END PROC out subtext mit anfangsbehandlung;
 984                              |
getcursor ................|PROC get cursor   :  get cursor (spalte, zeile)     END PROC get
                          |     cursor;
                          |


xcursor ..................|INT PROC x cursor :  get cursor; spalte             END PROC x
                          |     cursor;
                          |


writepermission ..........|BOOL PROC write permission :   write access   END PROC write
                          |     permission;
                          |


push .....................|PROC push (TEXT CONST ausfuehrkommando) :
                          |  IF   ausfuehrkommando = ""
                          |       (*sh*)
                          |  THEN
                          |  ELIF kommando = ""
                          |  THEN kommando := ausfuehrkommando
                          |  ELIF (kommando SUB kommando zeiger - 1) = ausfuehrkommando
                          |  THEN kommando zeiger DECR 1
                          |  ELIF replace moeglich
                          |  THEN kommando zeiger DECR laenge des ausfuehrkommandos;
                          |       replace (kommando, kommando zeiger, ausfuehrkommando)
                          |  ELSE insert char (kommando, ausfuehrkommando, kommando zeiger)
                          |  FI .
                          |
                          |
   replacemoeglich        |replace moeglich :
                          |  INT CONST laenge des ausfuehrkommandos := LENGTH ausfuehrkommando;
                          |  kommando zeiger > laenge des ausfuehrkommandos .
                          |END PROC push;
                          |


type .....................|PROC type (TEXT CONST ausfuehrkommando) :
                          |  kommando CAT ausfuehrkommando
                          |END PROC type;
                          |


stelleamanfang ...........|INT PROC stelle am anfang : anfang + verschoben END PROC stelle am
                          |     anfang;
                          |


stelleamende .............|INT PROC stelle am ende   : stelle am anfang+laenge-1 END PROC
                          |     stelle am ende;
                          |


markierausgleich .........|INT PROC markierausgleich : SIGN marke * marklength END PROC
                          |     markierausgleich;
                          |


verschiebenwennerforde ...|PROC verschieben wenn erforderlich :
                          |  IF   stelle > max schreibpos
                          |  THEN verschiebe (stelle - max schreibpos)
                          |  ELIF stelle < min schreibpos
```

```
                               |  THEN verschiebe (stelle - min schreibpos)
                               |  FI
                               |END PROC verschieben wenn erforderlich;
                               |


verschiebe ....................|PROC verschiebe (INT CONST i) :
                               |  verschoben     INCR i;
                               |  min schreibpos INCR i;
                               |  max schreibpos INCR i;
                               |  cpos           DECR i;
                               |  output mode := out feld;
                               |  schreibmarke positionieren (stelle)                 (* 11.05.85
                               |       -ws- *)
                               |END PROC verschiebe;
                               |


konstantenneuberechnen ...|PROC konstanten neu berechnen :
                               |  min schreibpos := anfang + verschoben;
                               |  IF   min schreibpos < 0                             (* 17.05.85
                               |       -ws- *)
                               |  THEN min schreibpos DECR verschoben; verschoben := 0
                               |  FI ;
                               |  max schreibpos := min schreibpos + laenge - 1 - markierausgleich;
                               |  cpos := rand + laenge - max schreibpos
                               |END PROC konstanten neu berechnen;
                               |


schreibmarkepositionie ...|PROC schreibmarke positionieren (INT CONST sstelle) :
                               |  cursor (cpos + sstelle, zeile)
                               |END PROC schreibmarke positionieren;
                               |


simplefeldout .................|PROC simple feldout (TEXT CONST satz, INT CONST dummy) :
                               |  (* PRECONDITION : NOT markiert AND verschoben = 0 *)
                               |  (*                AND feldrest schon geloescht    *)
                               |  schreibmarke an feldanfang positionieren;
                               |  out subtext mit randbehandlung (satz, anfang, anfang + laenge - 1);
                               |  IF (satz SUB LENGTH satz) = blank THEN absatzmarke schreiben
                               |       (TRUE) FI .
                               |
                               |
    schreibmarkeanfeldanfa |schreibmarke an feldanfang positionieren : cursor (rand + 1, zeile) .
                               |END PROC simple feldout;
                               |


feldout .......................|PROC feldout (TEXT CONST satz, INT CONST sstelle) :
                               |  schreibmarke an feldanfang positionieren;
                               |  feld ausgeben;
                               |  feldrest loeschen;
                               |  IF (satz SUB LENGTH satz) = blank THEN absatzmarke schreiben
                               |       (TRUE) FI .
                               |
                               |
    schreibmarkeanfeldanfa |schreibmarke an feldanfang positionieren : cursor (rand + 1, zeile) .
                               |
    feldausgeben           |feld ausgeben :
                           |  INT VAR von := anfang + verschoben, bis := von + laenge - 1;
                           |  IF   nicht markiert
                           |  THEN unmarkiert ausgeben
                           |  ELIF markiertes nicht sichtbar
                           |  THEN unmarkiert ausgeben
                           |  ELSE markiert ausgeben
                           |  FI .
                           |
                           |
    nichtmarkiert          |nicht markiert : marke <= 0 .
                           |
                           |
    markiertesnichtsichtba |markiertes nicht sichtbar :
                           |  bis DECR marklength * (1 + SIGN sstelle); marke > bis + 1 .
                           |
                           |
    unmarkiertausgeben     |unmarkiert ausgeben :
                           |  out subtext mit randbehandlung (satz, von, bis) .
                           |
                           |
    markiertausgeben       |markiert ausgeben :
                           |  INT VAR smarke := max (von, marke);
                           |  out text (satz, von, smarke - 1); out (begin mark);
                           |  verschiedene feldout modes behandeln .
                           |
                           |
    verschiedenefeldoutmod |verschiedene feldout modes behandeln :
                           |  IF   sstelle = 0
                           |  THEN out subtext mit randbehandlung (satz, smarke, bis); out (end
  +                        |       mark)
                           |  ELSE out text (satz, smarke, zeilenrand); out (end mark);
  +                        |       (*sh*)
                           |       out subtext mit randbehandlung (satz, sstelle, bis)
                           |  FI .
                           |
                           |
    zeilenrand             |zeilenrand : min (bis, sstelle - 1) .
                           |END PROC feldout;
                           |


absatzmarkeschreiben ..... |PROC absatzmarke schreiben (BOOL CONST schreiben) :
                           |  IF   fliesstext AND nicht markiert
                           |  THEN cursor (rand + 1 + laenge, zeile);
                           |       out (absatzmarke) ;
                           |       absatzmarke steht := TRUE
                           |  FI .
                           |
                           |
    nichtmarkiert          |nicht markiert : marke <= 0 .
                           |
                           |
    absatzmarke            |absatzmarke :
                           |  IF   NOT schreiben
                           |  THEN "  "
                           |  ELIF marklength > 0
                           |  THEN ""15""14""
                           |  ELSE ""15" "14" "
                           |  FI .
                           |END PROC absatzmarke schreiben;
                           |
inversout ...............|PROC invers out (TEXT CONST satz, INT CONST von, bis, TEXT CONST
                         |     pre, post) :
                         | IF   mark refresh line mode
                         | THEN feldout (satz, stelle)
                         | ELSE schreibmarke positionieren (von);
                         |      out (begin mark); markleft; out (pre);
                         |      out text (satz, von, bis - 1); out (post)
                         | FI .
                         |
                         |
   markleft              |markleft :
                         | marklength TIMESOUT left .
                         |
                         |END PROC invers out;
                         |

feldrestloeschen ........|PROC feldrest loeschen :
                         | IF   rand + laenge < maxbreite COR invertierte darstellung
                         | THEN INT VAR x; get cursor (x, zeile);
                         |      (rand + laenge - x + 1 + absatz ausgleich) TIMESOUT blank;
                         |           (*sh*)
                         |      cursor (x, zeile)
                         | ELSE out (clear eol); absatzmarke steht := FALSE
                         | FI
                         |END PROC feldrest loeschen;
                         |

AUFFUELLENMIT ...........|OP AUFFUELLENMIT (TEXT VAR satz, TEXT CONST fuellzeichen) :
                         | INT VAR i;
                         | FOR i FROM stelle - LENGTH satz DOWNTO 2 REP
                         |   satz CAT fuellzeichen
                         | PER
                         |END OP AUFFUELLENMIT;
                         |

einrueckposition ........|INT PROC einrueckposition (TEXT CONST satz) :
                         |     (*sh*)
                         | IF   fliesstext AND satz = blank
                         | THEN anfang
                         | ELSE max (pos (satz, ""33"", ""254"", 1), 1)
                         | FI
                         |END PROC einrueckposition;
                         |

letzterwortanfang .......|INT PROC letzter wortanfang (TEXT CONST satz) :
                         |     (*sh*)
                         | INT CONST ganz links := max (1, marke);
                         | BOOL VAR noch nicht im neuen wort := TRUE;
                         | INT  VAR i;
                         | FOR i FROM stelle DOWNTO ganz links REP
                         |     IF   noch nicht im neuen wort
                         |     THEN noch nicht im neuen wort := char = blank
                         |     ELIF is kanji esc (char)
                         |     THEN LEAVE letzter wortanfang WITH i
                         |     ELIF nicht mehr im neuen wort
                         |     THEN LEAVE letzter wortanfang WITH i + 1
                         |     FI
                         | PER ;
```

```
                              |  ganz links .
                              |
                              |
    char                      |char : satz SUB i .
                              |
                              |
    nichtmehrimneuenwort      |nicht mehr im neuen wort : char = blank COR within kanji (satz, i)
                              |END PROC letzter wortanfang;
                              |


getchar ......................|PROC getchar (TEXT VAR zeichen) :
                              |  IF   kommando = ""
                              |  THEN inchar (zeichen); IF lernmodus THEN audit CAT zeichen FI
                              |  ELSE zeichen := kommando SUB kommando zeiger;
                              |       kommando zeiger INCR 1;
                              |       IF   kommando zeiger > LENGTH kommando
                              |       THEN kommando zeiger := 1; kommando := ""
                              |       FI ;
                              |       IF   LENGTH kommando - kommando zeiger < 3
                              |       THEN kommando CAT inchety
                              |       FI
                              |  FI .
                              |END PROC getchar;
                              |


inchety ......................|TEXT PROC inchety :
                              |  IF   lernmodus
                              |  THEN TEXT VAR t := incharety; audit CAT t; t
                              |  ELSE incharety
                              |  FI
                              |END PROC inchety;
                              |


isincharety ..................|BOOL PROC is incharety (TEXT CONST muster) :
                              |  IF   kommando = ""
                              |  THEN TEXT CONST t := inchety;
                              |       IF t = muster THEN TRUE ELSE kommando := t; FALSE FI
                              |  ELIF (kommando SUB kommando zeiger) = muster
                              |  THEN kommando zeiger INCR 1;
                              |       IF   kommando zeiger > LENGTH kommando
                              |       THEN kommando zeiger := 1; kommando := ""
                              |       FI ;
                              |       TRUE
                              |  ELSE FALSE
                              |  FI
                              |END PROC is incharety;
                              |


getcharety ...................|TEXT PROC getcharety :
                              |  IF   kommando = ""
                              |  THEN inchety
                              |  ELSE TEXT CONST t := kommando SUB kommando zeiger;
                              |       kommando zeiger INCR 1;
                              |       IF   kommando zeiger > LENGTH kommando
                              |       THEN kommando zeiger := 1; kommando := ""
                              |       FI ; t
                              |  FI
                              |END PROC getcharety;
```

```
                             |
                             |
geteditcursor ...............|PROC get editcursor (INT VAR x, y) :
                             |     (*sh*)
                             |  IF actual editor > 0 THEN aktualisiere bildparameter FI;
                             |  x := rand - (anfang + verschoben - 1 - markierausgleich) + stelle;
                             |  y := zeile .
                             |
                             |
   aktualisierebildparame    |  aktualisiere bildparameter :
                             |    INT VAR old x, old y; get cursor (old x, old y);
                             |    dateizustand holen; bildausgabe steuern; satznr zeigen;
                             |    fenster zeigen; zeile := bildrand + zeilennr; cursor (old x, old
                             |         y) .
                             |END PROC get editcursor;
                             |
                             |(************************** Zugriff auf Feldstatus
                             |     **************************).
                             |
                             |
   stelle                    |stelle       : feldstatus.stelle .
                             |
   altestelle                |alte stelle  : feldstatus.alte stelle .
                             |
   rand                      |rand         : feldstatus.rand .
                             |
   limit                     |limit        : feldstatus.limit .
                             |
   anfang                    |anfang       : feldstatus.anfang .
                             |
   marke                     |marke        : feldstatus.marke .
                             |
   laenge                    |laenge       : feldstatus.laenge .
                             |
   verschoben                |verschoben   : feldstatus.verschoben .
                             |
   einfuegen                 |einfuegen    : feldstatus.einfuegen .
                             |
   fliesstext                |fliesstext   : feldstatus.fliesstext .
                             |
   writeaccess               |write access : feldstatus.write access .
                             |
   tabulator                 |tabulator    : feldstatus.tabulator .
                             |
                             |(**********************************************************************
                             |     *******)
                             |
                             |LET undefinierter bereich = 0,    nix           = 1,
                             |    bildzeile             = 2,    akt satznr    = 2,
                             |    abschnitt             = 3,    ueberschrift  = 3,
                             |    bild                  = 4,    fehlermeldung = 4;
                             |
                             |LET BILDSTATUS = STRUCT (INT feldlaenge, kurze feldlaenge,
                             |                             bildrand, bildlaenge, kurze bildlaenge,
                             |                             ueberschriftbereich, bildbereich,
                             |                             erster neusatz, letzter neusatz,
                             |                             old zeilennr, old lineno, old mark
                             |                                  lineno,
                             |                        BOOL zeileneinfuegen, old line update,
                             |                        TEXT satznr pre, ueberschrift pre,
                             |                             ueberschrift text, ueberschrift post,
                             |                                 old satz,
                                  |                             FRANGE old range,
                                  |                             FILE file),
                                  |      EDITSTATUS = STRUCT (FELDSTATUS feldstatus, BILDSTATUS
                                  |           bildstatus),
                                  |      max editor = 10,
                                  |      EDITSTACK  = ROW max editor EDITSTATUS;
                                  |
                                  |BILDSTATUS VAR bildstatus ;
                                  |EDITSTACK VAR editstack;
                                  |
                                  |ROW max editor INT VAR einrueckstack;
                                  |
                                  |BOOL VAR markiert;
                                  |TEXT VAR filename, tab, bildsatz, bildzeichen, fehlertext,
                                  |          akt bildsatz ;
                                  |INT  VAR zeilennr, satznr, bildanfang, bildmarke, feldmarke,
                                  |          actual editor := 0, max used editor := 0,
                                  |          letzer editor auf dieser datei,
                                  |          alte einrueckposition := 1;
                                  |


aktuellereditor ..........|INT PROC aktueller editor :  actual editor  END PROC aktueller
                          |     editor;
                          |


groesstereditor ..........|INT PROC groesster editor : max used editor END PROC groesster
                          |     editor;
                          |
                          |(******************************** bildeditor
                          |     *********************************)
                          |


bildeditor ...............|PROC bildeditor (TEXT CONST res, PROC (TEXT CONST) kommando
                          |     interpreter) :
                          |  evtl fehler behandeln;
                          |  enable stop;
                          |  TEXT VAR reservierte tasten := ""11""12""27"bf" ;
                          |  reservierte tasten CAT res ;
                          |  INT CONST my highest editor := max used editor;
                          |  laenge := feldlaenge;
                          |  konstanten neu berechnen;
                          |  REP
                          |    markierung justieren;
                          |    altes feld nachbereiten;
                          |    feldlaenge einstellen;
                          |    ueberschrift zeigen;
                          |    fenster zeigen ;
                          |    zeile bereitstellen;
                          |    zeile editieren;
                          |    kommando ausfuehren
                          |  PER .
                          |
                          |
    evtlfehlerbehandeln   |evtl fehler behandeln :
                          |  IF   is error
                          |  THEN fehlertext := errormessage;
                          |       IF fehlertext <> "" THEN neu (fehlermeldung, nix) FI;
                          |       clear error
                          |  ELSE fehlertext := ""
```

```
                        | FI .
                        |
                        |
markierungjustieren     |markierung justieren :
                        | IF   bildmarke > 0
                        | THEN IF   satznr <= bildmarke
                        |      THEN bildmarke := satznr;
                        |           stelle := max (stelle, feldmarke);
                        |           marke := feldmarke
                        |      ELSE marke := 1
                        |      FI
                        | FI .
                        |
                        |
zeilebereitstellen      |zeile bereitstellen : IF hinter letztem satz THEN insert record
                        |    (file) FI .
                        |
hinterletztemsatz       |hinter letztem satz : lineno (file) > lines (file) .
                        |
                        |
altesfeldnachbereiten   |altes feld nachbereiten :
                        | IF   old line update AND lineno (file) <> old lineno
                        | THEN IF   verschoben <> 0
                        |      THEN verschoben := 0; konstanten neu berechnen;
                        |      FI ;
                        |      INT CONST alte zeilennr := old lineno - bildanfang + 1;
                        |      IF   alte zeilennr > 0 AND alte zeilennr <= aktuelle
                        |           bildlaenge
                        |      THEN INT CONST m := marke;
                        |           IF   lineno (file) < old lineno
                        |           THEN marke := 0
                        |           ELIF old lineno = bildmarke
                        |           THEN marke := min (feldmarke, LENGTH old satz + 1)
                        |           ELSE marke := min (marke,     LENGTH old satz + 1)
                        |           FI ;
                        |           zeile := bildrand + alte zeilennr;
                        |           feldout (old satz, 0); marke := m
                        |      FI
                        | FI ;
                        | old line update := FALSE; old satz := "" .
                        |
                        |
feldlaengeeinstellen    |feldlaenge einstellen :
                        | INT CONST alte laenge := laenge;
                        | IF   zeilennr > kurze bildlaenge
                        | THEN laenge := kurze feldlaenge
                        | ELSE laenge := feldlaenge
                        | FI ;
                        | IF   laenge <> alte laenge
                        | THEN konstanten neu berechnen
                        | FI .
                        |
                        |
zeileeditieren          |zeile editieren :
                        | zeile := bildrand + zeilennr;
                        | exec (PROC (TEXT VAR, TEXT CONST) feldeditor, file, reservierte
                        |      tasten);
                        | old lineno := satznr;
                        | IF   markiert oder verschoben
                        | THEN old line update := TRUE; read record (file, old satz)
                        | FI .
                        |
markiertoderverschoben |markiert oder verschoben : markiert COR verschoben <> 0 .
                       |
                       |
kommandoausfuehren     |kommando ausfuehren :
                       |  getchar (bildzeichen);
                       |  SELECT pos (kommandos, bildzeichen) OF
                       |    CASE x hop   : hop kommando verarbeiten
                       |    CASE x esc   : esc kommando verarbeiten
                       |    CASE x up    : zum vorigen satz
                       |    CASE x down  : zum folgenden satz
                       |    CASE x rubin : zeicheneinfuegen umschalten
                       |    CASE x mark  : markierung umschalten
                       |    CASE x cr    : eingerueckt mit cr                  (*  08.06.8
                       |        -ws- *)
                       |    CASE x inscr : eingerueckt zum folgenden satz
                       |    CASE x abscr : zum anfang des folgenden satzes
                       |  END SELECT .
                       |
                       |
kommandos              |kommandos :
                       |  LET x hop   = 1,               x up    = 2,
                       |      x down  = 3,               x rubin = 4,
                       |      x cr    = 5,               x mark  = 6,
                       |      x abscr = 7,               x inscr = 8,
                       |      x esc   = 9;
                       |
                       |  ""1""3""10""11""13""16""17""18""27"" .
                       |
                       |
zeicheneinfuegenumscha |zeicheneinfuegen umschalten :
                       |  rubin segment in ueberschrift eintragen;
                       |  neu (ueberschrift, nix) .
                       |
                       |
rubinsegmentinuebersch |rubin segment in ueberschrift eintragen :
                       |  replace (ueberschrift text, 9, rubin segment) .
                       |
                       |
rubinsegment           |rubin segment :
                       |  IF einfuegen THEN "RUBIN" ELSE "....." FI .
                       |
                       |
hopkommandoverarbeiten |hop kommando verarbeiten :
                       |  getchar (bildzeichen);
                       |  read record (file, bildsatz);
                       |  SELECT pos (hop kommandos, bildzeichen) OF
                       |    CASE y hop    : nach oben
                       |    CASE y cr     : neue seite
                       |    CASE y up     : zurueckblaettern
                       |    CASE y down   : weiterblaettern
                       |    CASE y tab    : put tabs (file, tabulator); neu (ueberschrift,
                       |        nix)
                       |    CASE y rubout : zeile loeschen
                       |    CASE y rubin  : zeileneinfuegen umschalten
                       |  END SELECT .
                       |
                       |
hopkommandos           |hop kommandos :
                       |  LET y hop    = 1,              y up     = 2,
                       |      y tab    = 3,              y down   = 4,
                       |      y rubin  = 5,              y rubout = 6,
                       |      y cr     = 7;
```

```
                        |
                        |  ""1""3""9""10""11""12""13"" .
                        |
                        |
zeileneinfuegenumschal  |zeileneinfuegen umschalten :
                        |  zeileneinfuegen := NOT zeileneinfuegen;
                        |  IF   zeileneinfuegen
                        |  THEN zeile aufspalten; logisches eof setzen
                        |  ELSE leere zeile am ende loeschen; logisches eof loeschen
                        |  FI ; restbild zeigen .
                        |
                        |
zeileaufspalten         |zeile aufspalten :
                        |  IF   stelle <= LENGTH bildsatz OR stelle = 1
                        |  THEN loesche ggf trennende blanks und spalte zeile
                        |  FI .
                        |
                        |
loescheggftrennendebla  |loesche ggf trennende blanks und spalte zeile:          (* 26.06.84
                        |     -bk- *)
                        |  INT VAR first non blank pos := stelle;
                        |  WHILE first non blank pos <= length (bildsatz) CAND
                        |        (bildsatz SUB first non blank pos) = blank REP
                        |    first non blank pos INCR 1
                        |  PER ;
                        |  split line and indentation;
                        |        (*sh*)
                        |  first non blank pos := stelle - 1;
                        |  WHILE first non blank pos >= 1 CAND
                        |        (bildsatz SUB first non blank pos) = blank REP
                        |    first non blank pos DECR 1
                        |  PER;
                        |  bildsatz := subtext (bildsatz, 1, first non blank pos);
                        |  write record (file, bildsatz) .
                        |
                        |
splitlineandindentatio  |split line and indentation :
                        |  split line (file, first non blank pos, TRUE) .
                        |
                        |
logischeseofsetzen      |logisches eof setzen :
                        |  down (file); col (file, 1);
                        |  set range (file, 1, 1, old range); up (file) .
                        |
                        |
leerezeileamendeloesch  |leere zeile am ende loeschen :
                        |  to line (file, lines (file));
                        |  IF len (file) = 0 THEN delete record (file) FI;
                        |  to line (file, satznr) .
                        |
                        |
logischeseofloeschen    |logisches eof loeschen :
                        |  col (file, stelle); set range (file, old range) .
                        |
                        |
restbildzeigen          |restbild zeigen :
                        |  erster neusatz := satznr;
                        |  letzter neusatz := bildanfang + bildlaenge - 1;
                        |  rest segment in ueberschrift eintragen;
                        |  neu (ueberschrift, abschnitt) .
                        |
restsegmentinueberschr |rest segment in ueberschrift eintragen :
                       |  replace (ueberschrift text, feldlaenge - 25, rest segment) .
                       |
                       |
restsegment            |rest segment :
                       |  IF zeileneinfuegen THEN "REST" ELSE "...." FI .
                       |
                       |
esckommandoverarbeiten |esc kommando verarbeiten :
                       |  getchar (bildzeichen);
                       |  eventuell zeichen zurueckweisen;                         (* 04.05.85 -ws- *)
                       |  IF   taste ist reserviert
                       |  THEN belegte taste ausfuehren
                       |  ELSE fest vordefinierte esc funktion
                       |  FI ; ende nach quit .
                       |
                       |
eventuellzeichenzuruec |eventuell zeichen zurueckweisen :                          (* 04.05.85 -ws- *)
                       |  IF   NOT write access CAND NOT erlaubte taste
                       |  THEN benutzer warnen; LEAVE kommando ausfuehren
                       |  FI .
                       |
                       |
erlaubtetaste          |erlaubte taste     : pos (zulaessige zeichen, bildzeichen) > 0 .
                       |
zulaessigezeichen      |zulaessige zeichen : res + ""1""2""8""27"bfq" .
                       |
benutzerwarnen         |benutzer warnen    : out (piep) .
                       |
                       |
endenachquit           |ende nach quit :
                       |  IF max used editor < my highest editor THEN LEAVE bildeditor FI .
                       |
                       |
tasteistreserviert     |taste ist reserviert : pos (res, bildzeichen) > 0 .
                       |
                       |
festvordefinierteescfu |fest vordefinierte esc funktion :
                       |  read record (file, bildsatz);
                       |  SELECT pos (esc kommandos, bildzeichen) OF
                       |    CASE z hop    : lernmodus umschalten
                       |    CASE z esc    : kommandodialog versuchen
                       |    CASE z left   : zum vorigen wort
                       |    CASE z right  : zum naechsten wort
                       |    CASE z b      : bild an aktuelle zeile angleichen
                       |    CASE z f      : belegte taste ausfuehren
                       |    CASE z rubout : markiertes vorsichtig loeschen
                       |    CASE z rubin  : vorsichtig geloeschtes einfuegen
                       |    OTHERWISE     : belegte taste ausfuehren
                       |  END SELECT .
                       |
                       |
esckommandos           |esc kommandos :
                       |  LET z hop    =  1,               z right  =  2,
                       |      z left   =  3,               z rubin  =  4,
                       |      z rubout =  5,               z esc    =  6,
                       |      z b      =  7,               z f      =  8;
                       |
                       |  ""1""2""8""11""12""27"bf" .
                       |
zumvorigenwort            |zum vorigen wort :
                          |  IF   vorgaenger erlaubt
                          |  THEN vorgaenger; read record (file, bildsatz);
                          |       stelle := LENGTH bildsatz + 1; push (esc + left)
                          |  FI .
                          |
                          |
vorgaengererlaubt         |vorgaenger erlaubt :
                          |  satznr > max (1, bildmarke) .
                          |
                          |
zumnaechstenwort          |zum naechsten wort :
                          |  IF nicht auf letztem satz THEN weitersuchen wenn nicht gefunden FI
                          |       .
                          |
                          |
nichtaufletztemsatz       |nicht auf letztem satz : line no (file) < lines (file) .
                          |
                          |
weitersuchenwennnichtg    |weitersuchen wenn nicht gefunden :
                          |  nachfolgenden satz holen;
                          |  IF   (nachfolgender satz SUB anfang) = blank
                          |  THEN push (abscr + esc + right)
                          |  ELSE push (abscr)
                          |  FI .
                          |
                          |
nachfolgendensatzholen    |nachfolgenden satz holen :
                          |  down (file); read record (file, nachfolgender satz); up (file) .
                          |
                          |
bildanaktuellezeileang    |bild an aktuelle zeile angleichen :
                          |  anfang INCR verschoben; verschoben := 0;
                          |  margin segment in ueberschrift eintragen;
                          |  neu (ueberschrift, bild) .
                          |
                          |
marginsegmentinuebersc    |margin segment in ueberschrift eintragen :
                          |  replace (ueberschrift text, 2, margin segment) .
                          |
                          |
marginsegment             |margin segment :
                          |  IF   anfang <= 1
                          |  THEN "......"
                          |  ELSE TEXT VAR margin text := "M" + text (anfang);
                          |       (6 - LENGTH margin text) * "." + margin text
                          |  FI .
                          |
                          |
belegtetasteausfuehren    |belegte taste ausfuehren :
                          |  kommando analysieren (bildzeichen, PROC(TEXT CONST) kommando
                          |       interpreter) .
                          |
                          |
kommandodialogversuche    |kommandodialog versuchen:
                          |  IF   fenster ist zu schmal fuer dialog
                          |  THEN kommandodialog ablehnen
                          |  ELSE kommandodialog fuehren
                          |  FI .
                          |
fensteristzuschmalfuer |fenster ist zu schmal fuer dialog : laenge < 20 .
                       |
                       |
kommandodialogablehnen |kommandodialog ablehnen :
                       |  fehlertext := "zu schmal fuer ESC ESC"; neu (fehlermeldung, nix)
                       |
                       |
kommandodialogfuehren  |kommandodialog fuehren:
                       |  INT VAR x0, x1, x2, x3, y;
                       |  get cursor (x0, y);
                       |  cursor (rand + 1, bildrand + zeilennr);
                       |  get cursor (x1, y);
                       |  out (begin mark); out (monitor meldung);
                       |  get cursor (x2, y);
                       |  (laenge - LENGTH monitor meldung - marklength) TIMESOUT blank;
                       |  get cursor (x3, y);
                       |  out (end mark); out (blank);
                       |  kommandozeile editieren;
                       |  ueberschrift zeigen;
                       |  absatz ausgleich := 2;
                       |       (*sh*)
                       |  IF kommandotext = "" THEN LEAVE kommandodialog fuehren FI;
                       |  kommando auf taste legen ("f", kommandotext);
                       |  kommando analysieren ("f", PROC(TEXT CONST) kommando interpreter)
                       |  IF   fehlertext <> ""
                       |  THEN push (esc + esc + esc + "k")
                       |  ELIF markiert
                       |  THEN zeile neu
                       |  FI .
                       |
                       |
kommandozeileeditieren |kommandozeile editieren :
                       |  TEXT VAR kommandotext := "";
                       |  cursor (x1, y); out (begin mark);
                       |  disable stop;
                       |  darstellung invertieren;
                       |  editget schleife;
                       |  darstellung invertieren;
                       |  enable stop;
                       |  cursor (x3, y); out (end mark);
                       |  exec (PROC (TEXT CONST, INT CONST) feldout, file, stelle);
                       |  cursor (x0, y) .
                       |
                       |
darstellunginvertieren |darstellung invertieren :
                       |  TEXT VAR dummy := begin mark; begin mark := end mark; end mark :=
                       |       dummy;
                       |  invertierte darstellung := NOT invertierte darstellung .
                       |
                       |
editgetschleife        |editget schleife :
                       |  TEXT VAR exit char;
                       |  REP
                       |    cursor (x2, y);
                       |    editget (kommandotext, max textlength, rand + laenge - x cursor
                       |             "", "k?!", exit char);
                       |    neu (ueberschrift, nix);
                       |    IF   exit char = ""27"k"
                       |    THEN kommando text := kommando auf taste ("f")
                       |    ELIF exit char = ""27"?"
                       |    THEN TEXT VAR taste; getchar (taste);
                       |         kommando text := kommando auf taste (taste)
```

```
                        |    ELIF exit char = ""27"!"
                        |    THEN getchar (taste);
                        |         IF   ist reservierte taste
                        |         THEN set busy indicator;
                        |              (*sh*)
                        |              out ("FEHLER: """ + taste + """ ist reserviert"7"")
                        |         ELSE kommando auf taste legen (taste, kommandotext);
                        |              kommandotext := ""; LEAVE editget schleife
                        |         FI
                        |    ELSE LEAVE editget schleife
                        |    FI
                        |  PER .
                        |
                        |
istreserviertetaste     |ist reservierte taste : pos (res, taste) > 0 .
                        |
monitormeldung          |monitor meldung       : "gib kommando : " .
                        |
                        |
neueseite               |neue seite : bildanfang := satznr; zeilennr := 1; neu (akt satznr,
                        |     bild) .
                        |
                        |
weiterblaettern         |weiterblaettern :
                        |  INT CONST akt bildlaenge := aktuelle bildlaenge;
                        |  IF   nicht auf letztem satz
                        |  THEN erster neusatz := satznr;
                        |       IF   zeilennr >= akt bildlaenge
                        |       THEN bildanfang INCR akt bildlaenge; neu (akt satznr, bild)
                        |       FI ;
                        |       satznr := min (lines (file), bildanfang + akt bildlaenge - 1);
                        |       letzter neusatz := satznr;
                        |       toline (file, satznr);
                        |       stelle DECR verschoben;
                        |       neu (akt satznr, nix);
                        |       zeilennr := satznr - bildanfang + 1;
                        |       IF markiert THEN neu (nix, abschnitt) FI;
                        |       einrueckposition bestimmen
                        |  FI .
                        |
                        |
zurueckblaettern        |zurueckblaettern :
                        |  IF   vorgaenger erlaubt
                        |  THEN IF   zeilennr <= 1
                        |       THEN bildanfang := max (1, bildanfang - aktuelle bildlaenge);
                        |            neu (akt satznr, bild)
                        |       FI ;
                        |       nach oben; einrueckposition bestimmen
                        |  FI .
                        |
                        |
zeileloeschen           |zeile loeschen :
                        |  IF   stelle = 1
                        |  THEN delete record (file);
                        |       erster neusatz := satznr;
                        |       letzter neusatz := bildanfang + bildlaenge - 1;
                        |       neu (nix, abschnitt)
                        |  ELSE zeilen rekombinieren
                        |  FI .
                        |
zeilenrekombinieren     |zeilen rekombinieren :
                        |  IF   nicht auf letztem satz
                        |  THEN aktuellen satz mit blanks auffuellen;
                        |       delete record (file);
                        |       nachfolgenden satz lesen;
                        |       bildsatz CAT nachfolgender satz ohne fuehrende blanks;
                        |       write record (file, bildsatz);
                        |       erster neusatz := satznr;
                        |       letzter neusatz := bildanfang + bildlaenge - 1;
                        |       neu (nix, abschnitt)
                        |  FI .
                        |
                        |
aktuellensatzmitblanks  |aktuellen satz mit blanks auffuellen :
                        |  bildsatz AUFFUELLENMIT blank .
                        |
                        |
nachfolgendensatzlesen  |nachfolgenden satz lesen :
                        |  TEXT VAR nachfolgender satz;
                        |  read record (file, nachfolgender satz) .
                        |
                        |
nachfolgendersatzohnef  |nachfolgender satz ohne fuehrende blanks :
                        |  satzrest := subtext (nachfolgender satz,
                        |     einrueckposition (nachfolgender satz)); satzrest .
                        |
                        |
zeileaufsplitten        |zeile aufsplitten :
                        |  nachfolgender satz := "";
                        |  INT VAR i;
                        |  FOR i FROM 2 UPTO min (stelle, einrueckposition (bildsatz)) REP
                        |    nachfolgender satz CAT blank
                        |  PER;
                        |  satzrest := subtext (bildsatz, naechste non blank position);
                        |  nachfolgender satz CAT satzrest;
                        |  bildsatz := subtext (bildsatz, 1, stelle - 1);
                        |  write record (file, bildsatz);
                        |  down (file); insert record (file);
                        |  write record (file, nachfolgender satz); up (file) .
                        |
                        |
naechstenonblankpositi  |naechste non blank position :
                        |  INT VAR non blank pos := stelle;
                        |  WHILE (bildsatz SUB non blank pos) = blank REP
                        |    non blank pos INCR 1
                        |  PER; non blank pos .
                        |
                        |
zumvorigensatz          |zum vorigen satz :
                        |  IF vorgaenger erlaubt THEN vorgaenger; einrueckposition bestimmen
                        |       FI .
                        |
                        |
zumfolgendensatz        |zum folgenden satz :                                   (* 12.09.85
                        |     -ws- *)
                        |  IF nachfolger erlaubt THEN nachfolger; einrueckposition bestimmen
                        |                        ELSE col (file, len (file) + 1); neu (nix,
                        |                             nix)
                        |  FI .
                        |
einrueckpositionbestim |einrueckposition bestimmen :                          (* 27.08.85
                       |     -ws- *)
                       |  read record (file, akt bildsatz);
                       |  INT  VAR neue einrueckposition := einrueckposition (akt bildsatz);
                       |  IF   akt bildsatz ist leerzeile
                       |  THEN alte einrueckposition := max (stelle, neue einrueckposition)
                       |  ELSE alte einrueckposition := min (stelle, neue einrueckposition)
                       |  FI .
                       |
                       |
aktbildsatzistleerzeil |akt bildsatz ist leerzeile :
                       |  akt bildsatz = "" OR akt bildsatz = blank .
                       |
                       |
zumanfangdesfolgendens |zum anfang des folgenden satzes :
                       |  IF nachfolger erlaubt THEN nachfolger; stelle := anfang FI .
                       |
                       |
nachfolgererlaubt      |nachfolger erlaubt :
                       |  write access COR nicht auf letztem satz .
                       |
                       |
eingeruecktmitcr       |eingerueckt mit cr :
                       |  IF NOT nachfolger erlaubt THEN LEAVE eingerueckt mit cr FI;
                       |       (*sh*)
                       |  read record (file, bildsatz);
                       |  INT VAR epos := einrueckposition (bildsatz);
                       |  nachfolger; col (file, 1);
                       |  IF   eof (file)
                       |  THEN IF   LENGTH bildsatz <= epos
                       |       THEN stelle := alte einrueckposition
                       |       ELSE stelle := epos
                       |       FI
                       |  ELSE read record (file, bildsatz);
                       |       stelle := einrueckposition (bildsatz);
                       |       IF   bildsatz ist leerzeile                  (* 29.08.85
                       |            -ws- *)
                       |       THEN stelle := alte einrueckposition;
                       |            aktuellen satz mit blanks auffuellen
                       |       FI
                       |  FI ;
                       |  alte einrueckposition := stelle .
                       |
                       |
bildsatzistleerzeile   |bildsatz ist leerzeile :
                       |  bildsatz = "" OR bildsatz = blank .
                       |
                       |
eingeruecktzumfolgende |eingerueckt zum folgenden satz :
                       |     (*sh*)
                       |  IF NOT nachfolger erlaubt OR NOT write access
                       |    THEN LEAVE eingerueckt zum folgenden satz
                       |  FI;
                       |  alte einrueckposition merken;
                       |  naechsten satz holen;
                       |  neue einrueckposition bestimmen;
                       |  alte einrueckposition := stelle .
                       |
                       |
alteeinrueckpositionme |alte einrueckposition merken :
                       |  read record (file, bildsatz);
                       |  epos := einrueckposition (bildsatz);
```

```
                          | auf aufzaehlung pruefen;
                          | IF epos > LENGTH bildsatz THEN epos := anfang FI.
                          |
                          |
  aufaufzaehlungpruefen   |auf aufzaehlung pruefen :
                          | BOOL CONST aufzaehlung gefunden :=
                          |      ist aufzaehlung CAND vorher absatzzeile CAND wort folgt;
                          | IF   aufzaehlung gefunden THEN epos := anfang des naechsten wortes
                          |      FI .
                          |
                          |
  istaufzaehlung          |ist aufzaehlung :
                          | INT CONST wortende := pos (bildsatz, blank, epos, epos + 20) - 1;
                          | SELECT pos ("-*).:" , bildsatz SUB wortende) OF
                          |   CASE 1,2 : wortende =  epos
                          |   CASE 3,4 : wortende <= epos + 7
                          |   CASE 5   : TRUE
                          |   OTHERWISE: FALSE
                          | ENDSELECT .
                          |
                          |
  vorherabsatzzeile       |vorher absatzzeile :
                          | IF   satznr = 1
                          | THEN TRUE
                          | ELSE up (file);
                          |      INT  CONST vorige satzlaenge := len (file);
                          |      BOOL CONST vorher war absatzzeile :=
                          |      subtext (file, vorige satzlaenge, vorige satzlaenge) = blank;
                          |      down (file); vorher war absatzzeile
                          | FI .
                          |
                          |
  wortfolgt               |wort folgt :
                          | INT CONST anfang des naechsten wortes :=
                          | pos (bildsatz, ""33"", ""254"", wortende + 1);
                          | anfang des naechsten wortes > wortende .
                          |
                          |
  naechstensatzholen      |naechsten satz holen :
                          | nachfolger; col (file, 1);
                          | IF   eof (file)
                          | THEN bildsatz := ""
                          | ELSE IF   neue zeile einfuegen erforderlich
                          |      THEN insert record (file); bildsatz := "";
                          |           letzter neusatz := bildanfang + bildlaenge - 1
                          |      ELSE read record (file, bildsatz);
                          |           letzter neusatz := satznr;
                          |           ggf trennungen zurueckwandeln und umbruch indikator
                          |                einfuegen
                          |      FI ;
                          |      erster neusatz := satznr;
                          |      neu (nix, abschnitt)
                          | FI .
                          |
                          |
  neuezeileeinfuegenerfo  |neue zeile einfuegen erforderlich :
                          | BOOL CONST war absatz := war absatzzeile;
                          | war absatz COR neuer satz ist zu lang .
                          |
warabsatzzeile         |war absatzzeile :
                       |  INT VAR wl := pos (kommando, up backcr, kommando zeiger);
                       |  wl = 0 COR (kommando SUB (wl - 1)) = blank .
                       |
                       |
neuersatzistzulang     |neuer satz ist zu lang : laenge des neuen satzes >= limit .
                       |
                       |
laengedesneuensatzes   |laenge des neuen satzes :
                       |  IF   len (file) > 0
                       |  THEN len (file) + wl
                       |  ELSE wl + epos
                       |  FI .
                       |
                       |
upbackcr               |up backcr : ""3""20"" .
                       |
                       |
ggftrennungenzurueckwa |ggf trennungen zurueckwandeln und umbruch indikator einfuegen :
                       |  LET trenn k      = ""220"",
                       |      trenn strich = ""221"";
                       |  TEXT VAR umbruch indikator;
                       |  IF   letztes zeichen ist trenn strich
                       |  THEN entferne trenn strich;
                       |       IF   letztes zeichen = trenn k
                       |       THEN wandle trenn k um
                       |       FI ;
                       |       umbruch indikator := up backcr
                       |  ELIF letztes umgebrochenes zeichen ist kanji
                       |  THEN umbruch indikator := up backcr
                       |  ELSE umbruch indikator := blank + up backcr
                       |  FI ;
                       |  change (kommando, wl, wl+1, umbruch indikator) .
                       |
                       |
letztesumgebrochenesze |letztes umgebrochenes zeichen ist kanji : within kanji (kommando,
                       |     wl-1) .
                       |
                       |
letzteszeichenisttrenn |letztes zeichen ist trenn strich :
                       |  TEXT CONST last char := letztes zeichen;
                       |  last char = trenn strich COR
                       |  last char = "-" CAND wl > 2 CAND (kommando SUB (wl-2)) <> blank .
                       |
                       |
letzteszeichen         |letztes zeichen      : kommando SUB (wl-1) .
                       |
entfernetrennstrich    |entferne trenn strich : delete char (kommando, wl-1); wl DECR 1 .
                       |
wandletrennkum         |wandle trenn k um     : replace (kommando, wl-1, "c") .
                       |
loescheindikator       |loesche indikator     : delete char (kommando, wl) .
                       |
                       |
neueeinrueckpositionbe |neue einrueckposition bestimmen :
                       |  IF   aufzaehlung gefunden CAND bildsatz ist leerzeile
                       |  THEN stelle := epos
                       |  ELIF NOT bildsatz ist leerzeile
                       |  THEN stelle := einrueckposition (bildsatz)
                       |  ELIF war absatz COR auf letztem satz
                       |  THEN stelle := epos
                       |  ELSE down (file); read record (file, nachfolgender satz);
```

```
                          |      up   (file); stelle := einrueckposition (nachfolgender satz)
                          |  FI ;
                          |  IF   ist einfuegender aber nicht induzierter umbruch
                          |  THEN loesche indikator;
                          |       umbruchstelle := stelle + wl - kommando zeiger - anzahl der
                          |           stz;
                          |       umbruchverschoben := 0
                          |  FI .
                          |
                          |
aufletztemsatz            |auf letztem satz : NOT nicht auf letztem satz .
                          |
                          |
isteinfuegenderabernic    |ist einfuegender aber nicht induzierter umbruch :
                          |  wl := pos (kommando, backcr, kommando zeiger);
                          |  wl > 0 CAND (kommando SUB (wl - 1)) <> up char .
                          |
                          |
anzahlderstz              |anzahl der stz :
                          |  TEXT CONST umgebrochener anfang := subtext (kommando, kommando
                          |       zeiger, wl-1);
                          |  INT VAR anz := 0, anf := pos (umgebrochener anfang, ""1"", ""31"",
                          |       1);
                          |  WHILE anf > 0 REP
                          |    anz INCR 1; anf := pos (umgebrochener anfang, ""1"", ""31"", anf
                          |         + 1)
                          |  PER; anz .
                          |
                          |
markiertesvorsichtiglo    |markiertes vorsichtig loeschen :
                          |  IF   write access CAND markiert
                          |  THEN clear removed (file);
                          |       IF   nur im satz markiert
                          |       THEN behandle einen satz
                          |       ELSE behandle mehrere saetze
                          |       FI
                          |  FI .
                          |
                          |
nurimsatzmarkiert         |nur im satz markiert : line no (file) = bildmarke .
                          |
                          |
behandleeinensatz         |behandle einen satz :
                          |  insert record (file);
                          |  satzrest := subtext (bildsatz, marke, stelle - 1);
                          |  write record (file, satzrest);
                          |  remove (file, 1);
                          |  change (bildsatz, marke, stelle - 1, "");
                          |  stelle := marke;
                          |  marke := 0; bildmarke := 0; feldmarke := 0;
                          |  markiert := FALSE; mark (file, 0, 0);
                          |  konstanten neu berechnen;
                          |  IF   bildsatz = ""
                          |  THEN delete record (file);
                          |       erster neusatz := satznr;
                          |       letzter neusatz := bildanfang + bildlaenge - 1;
                          |       neu (nix, abschnitt)
                          |  ELSE write record (file, bildsatz);
                          |       neu (nix, bildzeile)
                          |  FI .
                          |
behandlemehreresaetze  |behandle mehrere saetze :
                       |  erster neusatz := bildmarke;
                       |  letzter neusatz := bildanfang + bildlaenge - 1;
                       |  zeile an aktueller stelle auftrennen;
                       |  ersten markierten satz an markieranfang aufspalten;
                       |  markierten bereich entfernen;
                       |  bild anpassen .
                       |
                       |
zeileanaktuellerstelle |zeile an aktueller stelle auftrennen :
                       |  INT VAR markierte saetze := line no (file) - bildmarke + 1;
                       |  IF   nicht am ende der zeile
                       |  THEN IF   nicht am anfang der zeile
                       |       THEN zeile aufsplitten
                       |       ELSE up (file); markierte saetze DECR 1
                       |       FI
                       |  FI .
                       |
                       |
nichtamanfangderzeile  |nicht am anfang der zeile : stelle > 1 .
                       |
nichtamendederzeile    |nicht am ende der zeile   : stelle <= LENGTH bildsatz .
                       |
                       |
erstenmarkiertensatzan |ersten markierten satz an markieranfang aufspalten :
                       |  to line (file, line no (file) - (markierte saetze - 1));
                       |  read record (file, bildsatz);
                       |  stelle := feldmarke;
                       |  IF   nicht am anfang der zeile
                       |  THEN IF   nicht am ende der zeile
                       |       THEN zeile aufsplitten
                       |       ELSE markierte saetze DECR 1
                       |       FI ;
                       |       to line (file, line no (file) + markierte saetze)
                       |  ELSE to line (file, line no (file) + markierte saetze - 1)
                       |  FI ;
                       |  read record (file, bildsatz) .
                       |
                       |
markiertenbereichentfe |markierten bereich entfernen :
                       |  zeilen nr := line no (file) - markierte saetze - bildanfang + 2;
                       |  remove (file, markierte saetze);
                       |  marke := 0; bildmarke := 0; feldmarke := 0;
                       |  markiert := FALSE; mark (file, 0, 0);
                       |  konstanten neu berechnen;
                       |  stelle := 1 .
                       |
                       |
bildanpassen           |bild anpassen :
                       |  satz nr := line no (file);
                       |  IF   zeilen nr <= 1
                       |  THEN bildanfang := line no (file); zeilen nr := 1;
                       |       neu (akt satznr, bild)
                       |  ELSE neu (akt satznr, abschnitt)
                       |  FI .
                       |
                       |
vorsichtiggeloeschtese |vorsichtig geloeschtes einfuegen :
                       |  IF   NOT write access OR removed lines (file) = 0
                       |  THEN LEAVE vorsichtig geloeschtes einfuegen
                       |  FI ;
                       |  IF   nur ein satz
```

```
                        |  THEN in aktuellen satz einfuegen
                        |  ELSE aktuellen satz aufbrechen und einfuegen
                        |  FI .
                        |
                        |
nureinsatz              |nur ein satz : removed lines (file) = 1 .
                        |
                        |
inaktuellensatzeinfueg  |in aktuellen satz einfuegen :
                        |  reinsert (file);
                        |  read record (file, nachfolgender satz);
                        |  delete record (file);
                        |  TEXT VAR t := bildsatz;
                        |  bildsatz := subtext (t, 1, stelle - 1);
                        |  aktuellen satz mit blanks auffuellen;
                        |       (*sh*)
                        |  bildsatz CAT nachfolgender satz;
                        |  satzrest := subtext (t, stelle);
                        |  bildsatz CAT satzrest;
                        |  write record (file, bildsatz);
                        |  stelle INCR LENGTH nachfolgender satz;
                        |  neu (nix, bildzeile) .
                        |
                        |
aktuellensatzaufbreche  |aktuellen satz aufbrechen und einfuegen :
                        |  INT CONST alter bildanfang := bildanfang;
                        |  old lineno := satznr;
                        |  IF   stelle = 1
                        |  THEN reinsert (file);
                        |       read record (file, bildsatz)
                        |  ELIF stelle > LENGTH bildsatz
                        |  THEN down (file);
                        |       reinsert (file);
                        |       read record (file, bildsatz)
                        |  ELSE INT VAR von := stelle;
                        |       WHILE (bildsatz SUB von) = blank REP von INCR 1 PER;
                        |       satzrest := subtext (bildsatz, von, LENGTH bildsatz);
                        |       INT VAR bis := stelle - 1;
                        |       WHILE (bildsatz SUB bis) = blank REP bis DECR 1 PER;
                        |       bildsatz := subtext (bildsatz, 1, bis);
                        |       write record (file, bildsatz);
                        |       down (file);
                        |       reinsert (file);
                        |       read record (file, bildsatz);
                        |       nachfolgender satz := einrueckposition (bildsatz) * blank;
                        |       nachfolgender satz CAT satzrest;
                        |       down (file); insert record (file);
                        |       write record (file, nachfolgender satz); up (file)
                        |  FI ;
                        |  stelle := max (1, LENGTH bildsatz);                  (* 22.06.84
                        |       -bk- *)
                        |  satz nr := line no (file);
                        |  zeilennr INCR satznr - old lineno;
                        |  zeilennr := min (zeilennr, aktuelle bildlaenge);
                        |  bildanfang := satznr - zeilennr + 1;
                        |  IF   bildanfang veraendert
                        |  THEN abschnitt neu (bildanfang, 9999)
                        |  ELSE abschnitt neu (old lineno, 9999)
                        |  FI ;
                        |  neu (akt satznr, nix).
                        |
     bildanfangveraendert   |bildanfang veraendert : bildanfang <> alter bildanfang .
                            |
                            |
     lernmodusumschalten    |lernmodus umschalten :
                            |  learn segment in ueberschrift eintragen; neu (ueberschrift, nix) .
                            |
                            |
     learnsegmentinuebersch |learn segment in ueberschrift eintragen :
                            |  replace (ueberschrift text, feldlaenge - 19, learn segment) .
                            |
                            |
     learnsegment           |learn segment :
                            |  IF lernmodus THEN "LEARN" ELSE "....." FI .
                            |
                            |
     markierungumschalten   |markierung umschalten :
                            |  IF markiert THEN markierung ausschalten ELSE markierung
                            |       einschalten FI .
                            |
                            |
     markierungeinschalten  |markierung einschalten :
                            |  bildmarke := satznr; feldmarke := marke; markiert := TRUE;
                            |  mark (file, bildmarke, feldmarke);
                            |  neu (nix, bildzeile) .
                            |
                            |
     markierungausschalten  |markierung ausschalten :
                            |  erster neusatz := max (bildmarke, bildanfang);
                            |  letzter neusatz := satznr;
                            |  bildmarke := 0; feldmarke := 0; markiert := FALSE;
                            |  mark (file, 0, 0);
                            |  IF   erster neusatz = letzter neusatz
                            |  THEN neu (nix, bildzeile)
                            |  ELSE neu (nix, abschnitt)
                            |  FI .
                            |END PROC bildeditor;
                            |


 neu .......................|PROC neu (INT CONST ue bereich, b bereich) :
                            |  ueberschriftbereich := max (ueberschriftbereich, ue bereich);
                            |  bildbereich := max (bildbereich, b bereich)
                            |END PROC neu;
                            |
                            |


 nachoben ..................|PROC nach oben :
                            |  letzter neusatz := satznr;
                            |  satznr := max (bildanfang, bildmarke);
                            |  toline (file, satznr);
                            |  stelle DECR verschoben;
                            |  zeilennr := satznr - bildanfang + 1;
                            |  erster neusatz := satznr;
                            |  IF   markiert
                            |  THEN neu (akt satznr, abschnitt)
                            |  ELSE neu (akt satznr, nix)
                            |  FI
                            |END PROC nach oben;
                            |
aktuellebildlaenge .......|INT PROC aktuelle bildlaenge :
                          |  IF stelle - stelle am anfang < kurze feldlaenge
                          |    AND feldlaenge > 0
                          |  THEN bildlaenge
                          |       (*wk*)
                          |  ELSE kurze bildlaenge
                          |  FI
                          |END PROC aktuelle bildlaenge;
                          |


vorgaenger ...............|PROC vorgaenger :
                          |  up (file); satznr DECR 1;
                          |  marke := 0; stelle DECR verschoben;
                          |  IF   zeilennr = 1
                          |  THEN bildanfang DECR 1; neu (ueberschrift, bild)
                          |  ELSE zeilennr DECR 1; neu (akt satznr, nix);
                          |       (*sh*)
                          |       IF markiert THEN neu (nix, bildzeile) FI
                          |  FI
                          |END PROC vorgaenger;
                          |


nachfolger ...............|PROC nachfolger :
                          |  down (file); satznr INCR 1;
                          |  stelle DECR verschoben;
                          |  IF   zeilennr = aktuelle bildlaenge
                          |  THEN bildanfang INCR 1;
                          |       IF   rollup erlaubt
                          |       THEN rollup
                          |       ELSE neu (ueberschrift, bild)
                          |       FI
                          |  ELSE neu (akt satznr, nix); zeilennr INCR 1
                          |       (*sh*)
                          |  FI ;
                          |  IF   markiert THEN neu (nix, bildzeile) FI .
                          |
                          |
   rolluperlaubt          |rollup erlaubt :
                          |  kurze bildlaenge = maxlaenge AND kurze feldlaenge = maxbreite .
                          |
                          |
   rollup                 |rollup :
                          |  out (down char);
                          |  IF   bildzeichen = inscr
                          |  THEN neu (ueberschrift, nix)
                          |  ELIF is cr or down CAND (write access COR nicht auf letztem satz)
                          |       (*sh*)
                          |  THEN neu (nix, bildzeile)
                          |  ELSE neu (ueberschrift, bildzeile)
                          |  FI .
                          |
                          |
   iscrordown             |is cr or down :
                          |  IF kommando = "" THEN kommando := inchety FI;
                          |  kommando char = down char COR kommando char = cr .
                          |
                          |
   kommandochar           |kommando char : kommando SUB kommando zeiger .
                          |
   nichtaufletztemsatz   |nicht auf letztem satz : line no (file) < lines (file) .
                         |END PROC nachfolger;
                         |

nextincharetyis .........|BOOL PROC next incharety is (TEXT CONST muster) :
                         |  INT CONST klen := LENGTH kommando - kommando zeiger + 1,
                         |            mlen := LENGTH muster;
                         |  INT VAR i; FOR i FROM 1 UPTO mlen - klen REP kommando CAT inchety
                         |       PER;
                         |  subtext (kommando, kommando zeiger, kommando zeiger + mlen - 1) =
                         |       muster
                         |END PROC next incharety is;
                         |

quitlast ................|PROC quit last:                                        (* 22.06.84
                         |     -bk- *)
                         |  IF  actual editor > 0 AND actual editor < max used editor
                         |  THEN verlasse alle groesseren editoren
                         |  FI .
                         |
                         |
   verlasseallegroesseren |verlasse alle groesseren editoren :
                         |  open editor (actual editor + 1); quit .
                         |END PROC quit last;
                         |

quit ....................|PROC quit :
                         |  IF actual editor > 0 THEN verlasse aktuellen editor FI .
                         |
                         |
   verlasseaktuellenedito |verlasse aktuellen editor :
                         |  disable stop;
                         |  INT CONST aktueller editor := actual editor;
                         |  in innersten editor gehen;
                         |  REP
                         |    IF zeileneinfuegen THEN hop rubin simulieren FI;
                         |    ggf bildschirmdarstellung korrigieren;
                         |    innersten editor schliessen
                         |  UNTIL aktueller editor > max used editor PER;
                         |  actual editor := max used editor .
                         |
                         |
   ininnersteneditorgehen |in innersten editor gehen : open editor (max used editor) .
                         |
                         |
   hoprubinsimulieren    |hop rubin simulieren :
                         |  zeileneinfuegen := FALSE;
                         |  leere zeilen am dateiende loeschen;
                         |       (*sh*)
                         |  ggf bildschirmdarstellung korrigieren;
                         |  logisches eof loeschen .
                         |
                         |
   innersteneditorschlies |innersten editor schliessen :
                         |  max used editor DECR 1;
                         |  IF  max used editor > 0
                         |  THEN open editor (max used editor);
                         |       bildeinschraenkung aufheben
                         |  FI .
```

```
                             |
                             |
   logischeseofloeschen      |logisches eof loeschen :
                             |  col (file, stelle); set range (file, old range) .
                             |
                             |
   leerezeilenamdateiende    |leere zeilen am dateiende loeschen :                    (* 15.08.85
                             |      -ws- *)
                             |  satz nr := line no (file) ;
                             |  to line (file, lines (file)) ;
                             |  WHILE lines (file) > 1 AND bildsatz ist leerzeile REP
                             |    delete record (file);
                             |    to line (file, lines (file))
                             |  PER;
                             |  toline (file, satznr) .
                             |
                             |
   bildsatzistleerzeile      |bildsatz ist leerzeile :
                             |  TEXT VAR bildsatz;
                             |  read record (file, bildsatz);
                             |  ist leerzeile .
                             |
                             |
   istleerzeile              |ist leerzeile :
                             |  bildsatz = "" OR bildsatz = blank .
                             |
                             |
   ggfbildschirmdarstellu    |ggf bildschirmdarstellung korrigieren :
                             |  satz nr DECR 1;                              (* für
                             |       Bildschirmkorrektur *)
                             |  IF   satznr > lines (file)
                             |  THEN zeilen nr DECR satz nr - lines (file);
                             |       satz nr := lines (file);
                             |       dateizustand retten
                             |  FI .
                             |
                             |
   bildeinschraenkungaufh    |bildeinschraenkung aufheben :
                             |  laenge := feldlaenge;
                             |  kurze feldlaenge := feldlaenge;
                             |  kurze bildlaenge := bildlaenge;
                             |  neu (nix, bild) .
                             |END PROC quit;
                             |


nichtsneu ...................|PROC nichts neu       : neu (nix,          nix)  END PROC nichts ne
                             |


satznrneu ...................|PROC satznr neu       : neu (akt satznr,   nix)  END PROC satznr ne
                             |


ueberschriftneu .............|PROC ueberschrift neu : neu (ueberschrift, nix)  END PROC
                             |     ueberschrift neu;
                             |


zeileneu ....................|PROC zeile neu :
                             |  INT CONST zeile := line no (file);
                             |  abschnitt neu (zeile, zeile)
                             |END PROC zeile neu;
                             |

abschnittneu ............|PROC abschnitt neu (INT CONST von satznr, bis satznr) :
                             |  IF   von satznr <= bis satznr
                             |  THEN erster neusatz  := min (erster neusatz, von satznr);
                             |       letzter neusatz := max (letzter neusatz, bis satznr);
                             |       neu (nix, abschnitt)
                             |  ELSE abschnitt neu (bis satznr, von satznr)
                             |  FI
                             |END PROC abschnitt neu;
                             |

bildabschnittneu ........|PROC bildabschnitt neu (INT CONST von zeile, bis zeile) :
                             |     (*sh*)
                             |  IF   von zeile <= bis zeile
                             |  THEN erster neusatz  := max (1, von zeile + bildanfang - 1);
                             |       letzter neusatz := min (bildlaenge, bis zeile + bildanfang -
                             |           1);
                             |       IF   von zeile < 1
                             |       THEN neu (ueberschrift, abschnitt)
                             |       ELSE neu (nix         , abschnitt)
                             |       FI
                             |  ELSE bildabschnitt neu (bis zeile, von zeile)
                             |  FI
                             |END PROC bildabschnitt neu;
                             |

bildneu .................|PROC bild neu : neu (nix, bild) END PROC bild neu;
                             |     (*sh*)
                             |

bildneu .................|PROC bild neu (FILE VAR f) :
                             |  INT CONST editor no := abs (editinfo (f)) DIV 256;
                             |  IF   editor no > 0 AND editor no <= max used editor
                             |  THEN IF   editor no = actual editor
                             |       THEN bild neu
                             |       ELSE editstack (editor no).bildstatus.bildbereich := bild
                             |       FI
                             |  FI
                             |END PROC bild neu;
                             |

allesneu ................|PROC alles neu :
                             |  neu (ueberschrift, bild);
                             |  INT VAR i;
                             |  FOR i FROM 1 UPTO max used editor REP
                             |    editstack (i).bildstatus.bildbereich := bild;
                             |    editstack (i).bildstatus.ueberschriftbereich := ueberschrift
                             |  PER
                             |END PROC alles neu;
                             |

satznrzeigen ............|PROC satznr zeigen :
                             |  out (satznr pre); out (text (text (lineno (file)), 4))
                             |END PROC satznr zeigen;
```

```
                              |

ueberschriftzeigen ...........|PROC ueberschrift zeigen :
                              |  SELECT ueberschriftbereich OF
                              |    CASE akt satznr    : satznr zeigen;
                              |                         ueberschriftbereich := nix
                              |    CASE ueberschrift  : ueberschrift schreiben;
                              |                         ueberschriftbereich := nix
                              |    CASE fehlermeldung : fehlermeldung schreiben;
                              |                         ueberschriftbereich := ueberschrift
                              |  END SELECT
                              |END PROC ueberschrift zeigen;
                              |

fensterzeigen ................|PROC fenster zeigen :
                              |  SELECT bildbereich OF
                              |    CASE bildzeile :
                              |         zeile := bildrand + zeilennr;
                              |         IF   line no (file) > lines (file)
                              |         THEN feldout ("", stelle)
                              |         ELSE exec (PROC (TEXT CONST, INT CONST) feldout, file,
                              |              stelle)
                              |         FI
                              |    CASE abschnitt :
                              |         bild ausgeben
                              |    CASE bild :
                              |         erster neusatz := 1;
                              |         letzter neusatz := 9999;
                              |         bild ausgeben
                              |    OTHERWISE :
                              |         LEAVE fenster zeigen
                              |  END SELECT;
                              |  erster neusatz := 9999;
                              |  letzter neusatz := 0;
                              |  bildbereich := nix
                              |END PROC fenster zeigen ;
                              |

bildausgeben .................|PROC bild ausgeben :
                              |  BOOL CONST schreiben ist ganz einfach := NOT markiert AND
                              |       verschoben = 0;
                              |  INT  CONST save marke := marke,
                              |             save verschoben := verschoben,
                              |             save laenge := laenge,
                              |             act lineno := lineno (file),
                              |             von := max (1, erster neusatz - bildanfang + 1);
                              |  INT   VAR bis := min (letzter neusatz - bildanfang + 1,
                              |       bildlaenge);
                              |  IF kurze feldlaenge <= 0 THEN bis := min (bis, kurze bildlaenge)
                              |       FI;
                              |  IF von > bis THEN LEAVE bild ausgeben FI;
                              |  verschoben := 0;
                              |  IF   markiert
                              |  THEN IF   mark lineno (file) < bildanfang + von - 1
                              |       THEN marke := anfang
                              |       ELSE marke := 0
                              |       FI
                              |  FI ;
                              |  abschnitt loeschen und neuschreiben;
```

```
                         | to line (file, act lineno);
                         | laenge := save laenge;
                         | verschoben := save verschoben;
                         | marke := save marke .
                         |
                         |
markiert                 |markiert : mark lineno (file) > 0 .
                         |
                         |
abschnittloeschenundne   |abschnitt loeschen und neuschreiben :
                         | abschnitt loeschen;
                         | INT VAR line number := bildanfang + von - 1;
                         | to line (file, line number);
                         | abschnitt schreiben .
                         |
                         |
abschnittloeschen        |abschnitt loeschen :
                         | cursor (rand + 1, bildrand + von);
                         | IF  bildrest darf komplett geloescht werden
                         | THEN out (clear eop)
                         | ELSE zeilenweise loeschen
                         | FI .
                         |
                         |
bildrestdarfkomplettge   |bildrest darf komplett geloescht werden :
                         | bis = maxlaenge AND kurze bildlaenge = maxlaenge
                         |                 AND kurze feldlaenge = maxbreite .
                         |
                         |
zeilenweiseloeschen      |zeilenweise loeschen :
                         | INT VAR i;
                         | FOR i FROM von UPTO bis REP
                         |   check for interrupt;
                         |   feldlaenge einstellen;
                         |   feldrest loeschen;
                         |   IF i < bis THEN out (down char) FI
                         | PER .
                         |
                         |
feldlaengeeinstellen     |feldlaenge einstellen :
                         | IF  ganze zeile sichtbar
                         | THEN laenge := feldlaenge
                         | ELSE laenge := kurze feldlaenge
                         | FI .
                         |
                         |
ganzezeilesichtbar       |ganze zeile sichtbar : i <= kurze bildlaenge .
                         |
                         |
abschnittschreiben       |abschnitt schreiben :
                         | INT CONST last line := lines (file);
                         | FOR i FROM von UPTO bis
                         | WHILE line number <= last line REP
                         |   check for interrupt;
                         |   feldlaenge einstellen;
                         |   zeile schreiben;
                         |   down (file);
                         |   line number INCR 1
                         | PER .
                         |
      checkforinterrupt       |check for interrupt :
                              |  kommando CAT inchety;
                              |  IF   kommando <> ""
                              |  THEN IF   zeilen nr = 1 CAND up command CAND vorgaenger erlaubt
                              |          THEN LEAVE abschnitt loeschen und neuschreiben
                              |          ELIF zeilen nr = bildlaenge CAND down command CAND nicht
                              |               letzter satz
                              |          THEN LEAVE abschnitt loeschen und neuschreiben
                              |          FI
                              |  FI .
                              |
                              |
      vorgaengererlaubt       |vorgaenger erlaubt :
                              |  satznr > max (1, bildmarke) .
                              |
                              |
      upcommand               |up command : next incharety is (""3"") COR next incharety is
                              |     (""1""3"") .
                              |
                              |
      downcommand             |down command :
                              |  next incharety is (""10"") CAND bildlaenge < maxlaenge
                              |  COR next incharety is (""1""10"") .
                              |
                              |
      nichtletztersatz        |nicht.letzter satz : act lineno < lines (file) .
                              |
                              |
      zeileschreiben          |zeile schreiben :
                              |  zeile := bildrand + i;
                              |  IF   schreiben ist ganz einfach
                              |  THEN exec (PROC (TEXT CONST, INT CONST) simple feldout, file, 0)
                              |  ELSE zeile kompliziert schreiben
                              |  FI ;
                              |  IF   line number = old lineno THEN old line update := FALSE FI .
                              |
                              |
      zeilekompliziertschrei  |zeile kompliziert schreiben :
                              |  IF   line number = mark lineno (file) THEN marke := mark col
                              |       (file) FI;
                              |  IF   line number = act lineno
                              |  THEN verschoben := save verschoben;
                              |       exec (PROC (TEXT CONST, INT CONST) feldout, file, stelle);
                              |       verschoben := 0; marke := 0
                              |  ELSE exec (PROC (TEXT CONST, INT CONST) feldout, file, 0);
                              |       IF line number = mark lineno (file) THEN marke := anfang FI
                              |  FI .
                              |END PROC bild ausgeben;
                              |


   bildzeigen ................|PROC bild zeigen :
                              |     (* wk *)
                              |
                              |  dateizustand holen ;
                              |  ueberschrift zeigen ;
                              |  bildausgabe steuern ;
                              |  bild neu ;
                              |  fenster zeigen ;
                              |  oldline no := satznr ;
                              |  old line update := FALSE ;
                              |  old satz := "" ;
```

```
                                | old zeilennr := satznr - bildanfang + 1 ;
                                | dateizustand retten .
                                |
                                |ENDPROC bild zeigen ;
                                |


ueberschriftinitialisi ...|PROC ueberschrift initialisieren :
                                |     (*sh*)
                                | satznr pre  :=
                                |      cursor pos + code (bildrand - 1) + code (rand + feldlaenge -
                                |           6);
                                | ueberschrift pre :=
                                |      cursor pos + code (bildrand - 1) + code (rand) + mark anf;
                                | ueberschrift text := ""; INT VAR i;
                                | FOR i FROM 16 UPTO feldlaenge REP ueberschrift text CAT "." PER;
                                | ueberschrift post :=  blank + mark end + "Zeile     " + mark anf;
                                | ueberschrift post CAT blank + mark end + "  ";
                                | filename := headline (file);
                                | filename := subtext (filename, 1, feldlaenge - 24);
                                | insert char (filename, blank, 1); filename CAT blank;
                                | replace (ueberschrift text, filenamepos, filename);
                                | rubin segment in ueberschrift eintragen;
                                | margin segment in ueberschrift eintragen;
                                | rest segment in ueberschrift eintragen;
                                | learn segment in ueberschrift eintragen .
                                |
                                |
     filenamepos                |filenamepos    : (LENGTH ueberschrift text - LENGTH filename + 3)
                                |     DIV 2 .
                                |
     markanf                    |mark anf       : begin mark + mark ausgleich.
                                |
     markend                    |mark end       :  end  mark + mark ausgleich.
                                |
     markausgleich              |mark ausgleich : (1 - sign (max (mark size, 0))) * blank .
                                |
                                |
     rubinsegmentinuebersch     |rubin segment in ueberschrift eintragen :
                                | replace (ueberschrift text, 9, rubin segment) .
                                |
                                |
     rubinsegment               |rubin segment :
                                | IF einfuegen THEN "RUBIN" ELSE "....." FI .
                                |
                                |
     marginsegmentinuebersc     |margin segment in ueberschrift eintragen :
                                | replace (ueberschrift text, 2, margin segment) .
                                |
                                |
     marginsegment              |margin segment :
                                | IF   anfang <= 1
                                | THEN "......"
                                | ELSE TEXT VAR margin text := "M" + text (anfang);
                                |      (6 - LENGTH margin text) * "." + margin text
                                | FI .
                                |
                                |
     restsegmentinueberschr     |rest segment in ueberschrift eintragen :
                                | replace (ueberschrift text, feldlaenge - 25, rest segment) .
                                |
   restsegment             |rest segment :
                           |  IF zeileneinfuegen THEN "REST" ELSE "...." FI .
                           |
                           |
   learnsegmentinuebersch  |learn segment in ueberschrift eintragen :
                           |  replace (ueberschrift text, feldlaenge - 19, learn segment) .
                           |
                           |
   learnsegment            |learn segment :
                           |  IF lernmodus THEN "LEARN" ELSE "....." FI .
                           |
                           |END PROC ueberschrift initialisieren;
                           |


ueberschriftschreiben .... |PROC ueberschrift schreiben :
                           |  replace (ueberschrift post, satznr pos, text (text (lineno
                           |       (file)), 4));
                           |  out (ueberschrift pre); out (ueberschrift text); out (ueberschrift
                           |       post);
                           |  get tabs (file, tab);
                           |  IF   pos (tab, dach) > 0
                           |  THEN out (ueberschrift pre);
                           |       out subtext (tab, anfang + 1, anfang + feldlaenge - 1);
                           |       cursor (rand + 1 + feldlaenge, bildrand); out (end mark)
                           |  FI .
                           |
                           |
   satznrpos               |  satznr pos : IF mark size > 0 THEN 9 ELSE 10 FI .
                           |       (*sh*)
                           |END PROC ueberschrift schreiben;
                           |


fehlermeldungschreiben ... |PROC fehlermeldung schreiben :
                           |  ueberschrift schreiben;
                           |  out (ueberschrift pre);
                           |  out ("FEHLER: ");
                           |  out subtext (fehlertext, 1, feldlaenge - 21);
                           |  out (blank);
                           |  out (piep);
                           |  cursor (rand + 1 + feldlaenge, bildrand); out (end mark)
                           |END PROC fehlermeldung schreiben;
                           |


setbusyindicator ......... |PROC set busy indicator :
                           |  cursor (rand + 2, bildrand)
                           |END PROC set busy indicator;
                           |


kommandoanalysieren ...... |PROC kommando analysieren (TEXT CONST taste,
                           |                            PROC (TEXT CONST) kommando interpreter) :
                           |  disable stop;
                           |  bildausgabe normieren;
                           |  zustand in datei sichern;
                           |  editfile modus setzen;
                           |  kommando interpreter (taste);
                           |  editfile modus zuruecksetzen;
                           |  IF actual editor <= 0 THEN LEAVE kommando analysieren FI;
                           |  absatz ausgleich := 2;
                           |       (*sh*)
                           | konstanten neu berechnen;
                           | neues bild bei undefinierter benutzeraktion;
                           | evtl fehler behandeln;
                           | zustand aus datei holen;
                           | bildausgabe steuern .
                           |
                           |
    editfilemodussetzen    |editfile modus setzen :
                           | BOOL VAR alter editget modus := editget modus ;
                           | editget modus := FALSE .
                           |
                           |
    editfilemoduszurueckse |editfile modus zuruecksetzen :
                           | editget modus := alter editget modus .
                           |
                           |
    evtlfehlerbehandeln    |evtl fehler behandeln :
                           | IF   is error
                           | THEN fehlertext := errormessage;
                           |      IF fehlertext <> "" THEN neu (fehlermeldung, nix) FI;
                           |      clear error
                           | ELSE fehlertext := ""
                           | FI .
                           |
                           |
    zustandindateisichern  |zustand in datei sichern :
                           | old zeilennr := zeilennr;
                           | old mark lineno := bildmarke;
                           | dateizustand retten .
                           |
                           |
    zustandausdateiholen   |zustand aus datei holen :
                           | dateizustand holen;
                           | IF   letzer editor auf dieser datei <> actual editor
                           | THEN zurueck auf alte position; neu (ueberschrift, bild)
                           | FI .
                           |
                           |
    zurueckaufalteposition |zurueck auf alte position :
                           | to line (file, old lineno);
                           | col (file, alte stelle);
                           | IF   fliesstext
                           | THEN editinfo (file,   old zeilennr)
                           | ELSE editinfo (file, - old zeilennr)
                           | FI ; dateizustand holen .
                           |
                           |
    bildausgabenormieren   |bildausgabe normieren :
                           | bildbereich := undefinierter bereich;
                           | erster neusatz := 9999;
                           | letzter neusatz := 0 .
                           |
                           |
    neuesbildbeiundefinier |neues bild bei undefinierter benutzeraktion :
                           | IF bildbereich = undefinierter bereich THEN alles neu FI .
                           |END PROC kommando analysieren;
                           |


bildausgabesteuern ........|PROC bildausgabe steuern :
                           | IF   markiert
                           | THEN IF   old mark lineno = 0
```

```
                         |       THEN abschnitt neu (bildmarke, satznr);
                         |            konstanten neu berechnen
                         |       ELIF stelle veraendert
                         |            (*sh*)
                         |       THEN zeile neu
                         |       FI
                         |  ELIF old mark lineno > 0
                         |  THEN abschnitt neu (old mark lineno, (max (satznr, old lineno)));
                         |       konstanten neu berechnen
                         |  FI ;
                         |  IF   satznr <> old lineno
                         |  THEN neu (akt satznr, nix);
                         |       neuen bildaufbau bestimmen
                         |  ELSE zeilennr := old zeilennr
                         |  FI ;
                         |  zeilennr := min (min (zeilennr, satznr), aktuelle bildlaenge);
                         |  bildanfang := satznr - zeilennr + 1 .
                         |
                         |
   stelleveraendert      |stelle veraendert : stelle <> alte stelle .
                         |
                         |
   neuenbildaufbaubestimm |neuen bildaufbau bestimmen :
                         |  zeilennr := old zeilennr + satznr - old lineno;
                         |  IF   1 <= zeilennr AND zeilennr <= aktuelle bildlaenge
                         |  THEN im fenster springen
                         |  ELSE bild neu aufbauen
                         |  FI .
                         |
                         |
   imfensterspringen     |im fenster springen :
                         |  IF markiert THEN abschnitt neu (old lineno, satznr) FI .
                         |
                         |
   bildneuaufbauen       |bild neu aufbauen :
                         |  neu (nix, bild); zeilennr := max (1, aktuelle bildlaenge DIV 2) .
                         |END PROC bildausgabe steuern;
                         |


wordwrap ................|PROC word wrap (BOOL CONST b) :
                         |  IF   actual editor = 0
                         |  THEN std fliesstext := b
                         |  ELSE fliesstext in datei setzen
                         |  FI .
                         |
                         |
   fliesstextindateisetze |fliesstext in datei setzen :
                         |  fliesstext := b;
                         |  IF fliesstext veraendert THEN editinfo (file, - editinfo (file))
                         |       FI;
                         |  neu (ueberschrift, bild) .
                         |
                         |
   fliesstextveraendert  |fliesstext veraendert :
                         |  fliesstext AND editinfo (file) < 0 OR
                         |  NOT fliesstext AND editinfo (file) > 0 .
                         |END PROC word wrap;
                         |
wordwrap ................|BOOL PROC word wrap :
                         |     (*sh*)
                         |  IF   actual editor = 0
                         |  THEN std fliesstext
                         |  ELSE fliesstext
                         |  FI
                         |END PROC word wrap;
                         |


margin ..................|INT PROC margin :   anfang   END PROC margin;
                         |


margin ..................|PROC margin (INT CONST i) :
                         |     (*sh*)
                         |  IF   anfang <> i CAND i > 0 AND i < 16001
                         |  THEN anfang := i; neu (ueberschrift, bild);
                         |       margin segment in ueberschrift eintragen
                         |  ELSE IF i >= 16001 OR i < 0
                         |         THEN errorstop ("ungueltige Anfangsposition (1 - 16000)")
                         |       FI
                         |  FI .
                         |
                         |
   marginsegmentinuebersc |margin segment in ueberschrift eintragen :
                         |  replace (ueberschrift text, 2, margin segment) .
                         |
                         |
   marginsegment          |margin segment :
                         |  IF   anfang <= 1
                         |  THEN "......"
                         |  ELSE TEXT VAR margin text := "M" + text (anfang);
                         |       (6 - LENGTH margin text) * "." + margin text
                         |  FI .
                         |
                         |END PROC margin;
                         |


rubinmode ...............|BOOL PROC rubin mode :   rubin mode (actual editor)   END PROC rubi
                         |     mode;
                         |


rubinmode ...............|BOOL PROC rubin mode (INT CONST editor nr) :
                         |     (*sh*)
                         |  IF   editor nr < 1 OR editor nr > max used editor
                         |  THEN errorstop ("Editor nicht eroeffnet")
                         |  FI ;
                         |  IF   editor nr = actual editor
                         |  THEN einfuegen
                         |  ELSE editstack (editor nr).feldstatus.einfuegen
                         |  FI
                         |END PROC rubin mode;
                         |


edit ....................|PROC edit (INT CONST i, TEXT CONST res,
                         |           PROC (TEXT CONST) kommando interpreter) :
                         |  edit (i, i, i, res, PROC (TEXT CONST) kommando interpreter)
                         |END PROC edit;
```

```
                               |

edit .....................|PROC edit (INT CONST von, bis, start, TEXT CONST res,
                          |         PROC (TEXT CONST) kommando interpreter) :
                          | disable stop;
                          | IF   von < bis
                          | THEN edit (von+1, bis, start, res, PROC (TEXT CONST) kommando
                          |      interpreter);
                          |      IF max used editor < von THEN LEAVE edit FI;
                          |      open editor (von)
                          | ELSE open editor (start)
                          | FI ;
                          | absatz ausgleich := 2;
                          | bildeditor (res, PROC (TEXT CONST) kommando interpreter);
                          | cursor (1, schirmhoehe);
                          | IF   is error
                          | THEN kommando zeiger := 1; kommando := ""; quit
                          | FI ;
                          | IF   lernmodus CAND actual editor = 0 THEN warnung ausgeben FI .
                          |      (*sh*)
                          |
                          |
   warnungausgeben        | warnung ausgeben :
                          |   out (clear eop); out ("WARNUNG: Lernmodus nicht
                          |        ausgeschaltet"13""10"") .
                          |END PROC edit;
                          |

dateizustandholen ........|PROC dateizustand holen :
                          | modify (file);
                          | get tabs (file, tabulator);
                          | zeilennr und fliesstext und letzter editor aus editinfo decodieren;
                          | limit := max line length (file);
                          | stelle := col (file);
                          | markiert := mark (file);
                          | IF   markiert
                          | THEN markierung holen
                          | ELSE keine markierung
                          | FI ;
                          | satz nr := lineno (file);
                          | IF   zeilennr > aktuelle bildlaenge
                          |      (*sh*)
                          | THEN zeilennr := min (satznr, aktuelle bildlaenge); bild neu
                          | ELIF zeilennr > satznr
                          | THEN zeilennr := min (satznr, aktuelle bildlaenge)
                          | FI ; zeilennr := max (zeilennr, 1);
                          | bildanfang := satz nr - zeilennr + 1 .
                          |
                          |
   zeilennrundfliesstextu |zeilennr und fliesstext und letzter editor aus editinfo decodieren :
                          | zeilennr := edit info (file);
                          | IF   zeilennr = 0
                          | THEN zeilennr := 1;
                          |      fliesstext := std fliesstext
                          | ELIF zeilennr > 0
                          | THEN fliesstext := TRUE
                          | ELSE zeilennr := - zeilennr;
                          |      fliesstext := FALSE
                          | FI ;
                          | letzer editor auf dieser datei := zeilennr DIV 256;
```

```
                            |  zeilennr := zeilennr MOD 256 .
                            |
                            |
    markierungholen         |markierung holen :
                            |  bildmarke := mark lineno (file);
                            |  feldmarke := mark col (file);
                            |  IF   line no (file) <= bildmarke
                            |  THEN to line (file, bildmarke);
                            |       marke := feldmarke;
                            |       stelle := max (stelle, feldmarke)
                            |  ELSE marke := 1
                            |  FI .
                            |
                            |
    keinemarkierung         |keine markierung :
                            |  bildmarke := 0;
                            |  feldmarke := 0;
                            |  marke     := 0 .
                            |END PROC dateizustand holen;
                            |


dateizustandretten .......|PROC dateizustand retten :
                            |  put tabs (file, tabulator);
                            |  IF   fliesstext
                            |  THEN editinfo (file,    zeilennr + actual editor * 256)
                            |  ELSE editinfo (file, - (zeilennr + actual editor * 256))
                            |  FI ;
                            |  max line length (file, limit);
                            |  col (file, stelle);
                            |  IF   markiert
                            |  THEN mark (file, bildmarke, feldmarke)
                            |  ELSE mark (file, 0, 0)
                            |  FI
                            |END PROC dateizustand retten;
                            |


openeditor ...............|PROC open editor (FILE CONST new file, BOOL CONST access) :
                            |  disable stop; quit last;
                            |  neue bildparameter bestimmen;
                            |  open editor (actual editor + 1, new file, access, x, y, x len, y
   +                        |       len).
                            |
                            |
    neuebildparameterbesti  |neue bildparameter bestimmen :
                            |  INT VAR x, y, x len, y len;
                            |  IF   actual editor > 0
                            |  THEN teilbild des aktuellen editors
                            |  ELSE volles bild
                            |  FI .
                            |
                            |
    teilbilddesaktuellened  |teilbild des aktuellen editors :
                            |  get editcursor (x, y); bildgroesse bestimmen;
                            |  IF   fenster zu schmal
   +                        |       (*sh*)
                            |  THEN enable stop; errorstop ("Fenster zu klein")
                            |  ELIF fenster zu kurz
                            |  THEN verkuerztes altes bild nehmen
                            |  FI .
                            |
  bildgroessebestimmen   |bildgroesse bestimmen :
                         |  x len := rand + feldlaenge - x + 3;
                         |  y len := bildrand + bildlaenge - y + 1 .
                         |
                         |
  fensterzuschmal        |fenster zu schmal : x > schirmbreite - 17 .
                         |
  fensterzukurz          |fenster zu kurz   : y > schirmhoehe  -  1 .
                         |
                         |
  verkuerztesaltesbildne |verkuerztes altes bild nehmen :
                         |  x := rand + 1; y := bildrand + 1;
                         |  IF fenster zu kurz THEN enable stop; errorstop ("Fenster zu
   +                     |       klein") FI;
                         |  x len := feldlaenge + 2;
                         |  y len := bildlaenge;
                         |  kurze feldlaenge := 0;
                         |  kurze bildlaenge := 1 .
                         |
                         |
  vollesbild             |volles bild :
                         |  x := 1; y := 1; x len := schirmbreite; y len := schirmhoehe .
                         |END PROC open editor;
                         |


openeditor ..............|PROC open editor (INT CONST editor nr,
                         |                  FILE CONST new file, BOOL CONST access,
                         |                  INT CONST x start, y, x len start, y len) :
                         |  INT VAR x := x start,
                         |  x len := x len start;
                         |  IF   editor nr > max editor
                         |  THEN errorstop ("zu viele Editor-Fenster")
                         |  ELIF editor nr > max used editor + 1 OR editor nr < 1
                         |  THEN errorstop ("Editor nicht eroeffnet")
                         |  ELIF fenster ungueltig
                         |  THEN errorstop ("Fenster ungueltig")
                         |  ELSE neuen editor stacken
                         |  FI .
                         |
                         |
  fensterungueltig       |fenster ungueltig :
                         |  x < 1  COR  x > schirmbreite  COR  y < 1  COR  y > schirmhoehe  COR
                         |  x len - 2 <= 15  COR  y len - 1 < 1  COR
                         |  x + x len - 1 > schirmbreite  COR  y + y len - 1 > schirmhoehe .
                         |
                         |
  neueneditorstacken     |neuen editor stacken :
                         |  disable stop;
                         |  IF   actual editor > 0 AND ist einschraenkung des alten bildes
                         |  THEN dateizustand holen;
                         |       aktuelles editorbild einschraenken;
                         |       arbeitspunkt in das restbild positionieren;
                         |       abgrenzung beruecksichtigen
                         |  FI ;
                         |  aktuellen zustand retten;
                         |  neuen zustand setzen;
                         |  neues editorbild zeigen;
                         |  actual editor := editor nr;
                         |  IF   actual editor > max used editor
                         |  THEN max used editor := actual editor
                         |  FI .
```

```
                          |
                          |
isteinschraenkungdesal    |ist einschraenkung des alten bildes :
                          |  x > rand      CAND  x + x len = rand + feldlaenge + 3  CAND
                          |  y > bildrand  CAND  y + y len = bildrand + bildlaenge + 1 .
                          |
                          |
aktuelleseditorbildein    |aktuelles editorbild einschraenken :
                          |  kurze feldlaenge := x - rand - 3;
                          |  kurze bildlaenge := y - bildrand - 1 .
                          |
                          |
arbeitspunktindasrestb    |arbeitspunkt in das restbild positionieren :
                          |  IF   stelle > 3
                          |  THEN stelle DECR 3; alte stelle := stelle
                          |  ELSE WHILE zeilennr > 1 AND zeilennr > kurze bildlaenge REP
                          |         vorgaenger
                          |       PER; old lineno := satznr
                          |  FI .
                          |
                          |
abgrenzungberuecksicht    |abgrenzung beruecksichtigen :
                          |  IF   x - rand > 1
                          |  THEN balken malen;
                          |       x INCR 2;
                          |       x len DECR 2
                          |  FI .
                          |
                          |
balkenmalen               |balken malen :
                          |  INT VAR i;
                          |  FOR i FROM 0 UPTO y len-1 REP
                          |    cursor (x, y+i); out (kloetzchen)
                          |        (*sh*)
                          |  PER .
                          |
                          |
kloetzchen                |kloetzchen : IF mark size > 0 THEN ""15""14"" ELSE ""15" "14" " FI .
                          |
                          |
aktuellenzustandretten    |aktuellen zustand retten :
                          |  IF   actual editor > 0
                          |  THEN dateizustand retten;
                          |       editstack (actual editor).feldstatus := feldstatus;
                          |       editstack (actual editor).bildstatus := bildstatus;
                          |       einrueckstack (actual editor) := alte einrueckposition
                          |  FI .
                          |
                          |
neuenzustandsetzen        |neuen zustand setzen :
                          |  FRANGE VAR frange;
                          |  feldstatus := FELDSTATUS :
                          |    (1, 1, x-1, 0, 1, 0, x len-2, 0, FALSE, TRUE, access, "");
                          |  bildstatus := BILDSTATUS :
                          |    (x len-2, x len-2, y, y len-1, y len-1, ueberschrift, bild,
                          |     0, 0, 1, 0, 0, FALSE, FALSE, "", "", "", "", "", frange, new
                          |         file);
                          |  alte einrueckposition := 1;
                          |  dateizustand holen;
                          |  ueberschrift initialisieren .
                          |
     neueseditorbildzeigen  |neues editorbild zeigen :
                            |  ueberschrift zeigen; fenster zeigen
                            |END PROC open editor;
                            |


  openeditor ...............|PROC open editor (INT CONST i) :
                            |  IF   i < 1 OR i > max used editor
                            |  THEN errorstop ("Editor nicht eroeffnet")
                            |  ELIF actual editor <> i
                            |  THEN switch editor
                            |  FI .
                            |
                            |
     switcheditor           |switch editor :
                            |  aktuellen zustand retten;
                            |  actual editor := i;
                            |  neuen zustand setzen;
                            |  IF   kein platz mehr fuer restfenster
                            |  THEN eingeschachtelte editoren vergessen;
                            |       bildeinschraenkung aufheben
                            |  ELSE neu (nix, nix)
                            |  FI .
                            |
                            |
     aktuellenzustandretten |aktuellen zustand retten :
                            |  IF   actual editor > 0
                            |  THEN editstack (actual editor).feldstatus := feldstatus;
                            |       editstack (actual editor).bildstatus := bildstatus;
                            |       einrueckstack (actual editor) := alte einrueckposition;
                            |       dateizustand retten
                            |  FI .
                            |
                            |
     neuenzustandsetzen     |neuen zustand setzen :
                            |  feldstatus := editstack (i).feldstatus;
                            |  bildstatus := editstack (i).bildstatus;
                            |  alte einrueckposition := einrueckstack (i);
                            |  dateizustand holen .
                            |
                            |
     keinplatzmehrfuerrestf |kein platz mehr fuer restfenster :
                            |  kurze feldlaenge < 1 AND kurze bildlaenge < 1 .
                            |
                            |
     eingeschachtelteeditor |eingeschachtelte editoren vergessen :
                            |  IF actual editor < max used editor
                            |    THEN open editor (actual editor + 1) ;
                            |         quit
                            |  FI ;
                            |  open editor (i) .
                            |
                            |
     bildeinschraenkungaufh |bildeinschraenkung aufheben :
                            |  laenge := feldlaenge;
                            |  kurze feldlaenge := feldlaenge;
                            |  kurze bildlaenge := bildlaenge;
                            |  neu (ueberschrift, bild) .
                            |END PROC open editor;
                            |
editfile ...............|FILE PROC editfile :
                        |  IF   actual editor = 0 OR editget modus
                        |  THEN errorstop ("Editor nicht eroeffnet")
                        |  FI ; file
                        |END PROC editfile;
                        |


getwindow ..............|PROC get window (INT VAR x, y, x size, y size) :
                        |  x := rand + 1;
                        |  y := bildrand;
                        |  x size := feldlaenge + 2;
                        |  y size := bildlaenge + 1
                        |ENDPROC get window;
                        |
                        |(************************* Zugriff auf Bildstatus
                        |     *************************).
                        |
                        |
   feldlaenge           |feldlaenge          : bildstatus.feldlaenge .
                        |
   kurzefeldlaenge      |kurze feldlaenge    : bildstatus.kurze feldlaenge .
                        |
   bildrand             |bildrand            : bildstatus.bildrand .
                        |
   bildlaenge           |bildlaenge          : bildstatus.bildlaenge .
                        |
   kurzebildlaenge      |kurze bildlaenge    : bildstatus.kurze bildlaenge .
                        |
   ueberschriftbereich  |ueberschriftbereich : bildstatus.ueberschriftbereich .
                        |
   bildbereich          |bildbereich         : bildstatus.bildbereich .
                        |
   ersterneusatz        |erster neusatz      : bildstatus.erster neusatz .
                        |
   letzterneusatz       |letzter neusatz     : bildstatus.letzter neusatz .
                        |
   oldzeilennr          |old zeilennr        : bildstatus.old zeilennr .
                        |
   oldlineno            |old lineno          : bildstatus.old lineno .
                        |
   oldmarklineno        |old mark lineno     : bildstatus.old mark lineno .
                        |
   zeileneinfuegen      |zeileneinfuegen     : bildstatus.zeileneinfuegen .
                        |
   oldlineupdate        |old line update     : bildstatus.old line update .
                        |
   satznrpre            |satznr pre          : bildstatus.satznr pre .
                        |
   ueberschriftpre      |ueberschrift pre    : bildstatus.ueberschrift pre .
                        |
   ueberschrifttext     |ueberschrift text   : bildstatus.ueberschrift text .
                        |
   ueberschriftpost     |ueberschrift post   : bildstatus.ueberschrift post .
                        |
   oldsatz              |old satz            : bildstatus.old satz .
                        |
   oldrange             |old range           : bildstatus.old range .
                        |
   file                 |file                : bildstatus.file .
                        |
                        |END PACKET editor paket;
```

```
editorfunctions ***********|PACKET  editor functions  DEFINES                   (* FUNCTIONS
                           |     052 *)
                           |        (**************)                           (*  17.07.85
                           |             -bk-  *)
                           |                                                   (*  10.09.85
                           |     -ws-  *)
                           |        edit,                                      (*  25.04.86
                           |             -sh-  *)
                           |        show,                                      (*  27.05.86
                           |             -wk-  *)
                           |        U,
                           |        D,
                           |        T,
                           |        up,
                           |        down,
                           |        downety,
                           |        uppety,
                           |        to line,
                           |        PUT,
                           |        GET,
                           |        P,
                           |        G,
                           |        limit,
                           |        len,
                           |        eof,
                           |        C,
                           |        change to,
                           |        CA,
                           |        change all,
                           |        lines,
                           |        line no,
                           |        col,
                           |        mark,
                           |        at,
                           |        word,
                           |        std kommando interpreter,
                           |        note,
                           |        note line,
                           |        note edit,
                           |        anything noted,
                           |        note file:
                           |
                           |
                           |LET     marker        = "^",
                           |        ersatzmarker  = "'",
                           |        schritt       = 50,
                           |        file size     = 4072,
                           |        write acc     = TRUE,
                           |        read acc      = FALSE;
                           |
                           |LET     bold          = 2,
                           |        integer       = 3,
                           |        string        = 4,
                           |        end of file   = 7;
                           |
                           |LET     std res       = "eqvw19dpgn"9"";
                           |
                           |FILE VAR edfile;
                           |BOOL VAR from scratchfile :: FALSE;
                           |TEXT VAR kommandotext, tabulator, zeile;
                           |
                           |
stdkommandointerpreter ...|PROC std kommando interpreter (TEXT CONST taste) :
                          |  enable stop ;
                          |  edfile := editfile;
                          |  set busy indicator;
                          |  SELECT pos (std res, taste) OF
                          |    CASE 1 (*e*)  : edit
                          |    CASE 2 (*q*)  : quit
                          |    CASE 3 (*v*)  : quit last
                          |    CASE 4 (*w*)  : open editor (next editor)
                          |    CASE 5 (*1*)  : toline (1); col (1)
                          |    CASE 6 (*9*)  : toline (lines); col (len+1)
                          |    CASE 7 (*d*)  : d case
                          |    CASE 8 (*p*)  : p case
                          |    CASE 9 (*g*)  : g case
                          |    CASE 10(*n*)  : note edit
                          |    CASE 11(*tab*): change tabs
                          |    OTHERWISE     : echtes kommando analysieren
                          |  END SELECT .
                          |
                          |
   dcase                  |d case :
                          |  IF   mark
                          |  THEN PUT ""; mark (FALSE); from scratchfile := TRUE
                          |  ELSE textzeile auf taste legen
                          |  FI .
                          |
                          |
   pcase                  |p case :
                          |  IF   mark
                          |       (*sh*)
                          |  THEN IF   write permission
                          |       THEN PUT ""; push(""27""12""); from scratchfile := TRUE
                          |       ELSE out (""7"")
                          |       FI
                          |  ELSE textzeile auf taste legen
                          |  FI .
                          |
                          |
   gcase                  |g case :
                          |  IF   write permission
                          |       (*sh*)
                          |  THEN IF   from scratchfile
                          |       THEN GET ""
                          |       ELSE IF is editget
                          |              THEN push (lernsequenz auf taste ("g")); nichts neu
                          |            FI
                          |       FI
                          |  ELSE out (""7"")
                          |  FI .
                          |
                          |
   textzeileauftastelegen |textzeile auf taste legen :
                          |  read record (edfile, zeile);
                          |  zeile := subtext (zeile, col);
                          |  lernsequenz auf taste legen ("g", zeile);
                          |  from scratchfile := FALSE; zeile neu .
                          |
                          |
   nexteditor             |next editor :
                          |  (aktueller editor MOD groesster editor) + 1 .
                          |
changetabs              |change tabs :
                        |  get tabs (edfile, tabulator) ;
                        |  IF   pos (tabulator, marker) <> 0
                        |  THEN change all (tabulator, marker, ersatzmarker)
                        |  ELSE change all (tabulator, ersatzmarker, marker)
                        |  FI ;
                        |  put tabs (edfile, tabulator) ;
                        |  ueberschrift neu .
                        |
                        |
echteskommandoanalysie  |echtes kommando analysieren :
                        |  kommandotext := kommando auf taste (taste);
                        |  IF   kommandotext = ""
                        |  THEN nichts neu; LEAVE std kommando interpreter
                        |  FI ;
                        |  scan (kommandotext);
                        |  TEXT VAR s1; INT VAR t1; next symbol (s1, t1);
                        |  TEXT VAR s2; INT VAR t2; next symbol (s2, t2);
                        |  IF   t1 = integer AND t2 = end of file THEN toline (int (s1))
                        |  ELIF t1 = string  AND t2 = end of file THEN down (s1)
                        |  ELIF perhaps simple up or down         THEN
                        |  ELIF perhaps simple changeto           THEN
                        |  ELSE do (kommandotext)
                        |  FI .
                        |
                        |
perhapssimpleupordown   |perhaps simple up or down :
                        |  IF   t1 = bold
                        |  THEN TEXT VAR s3; INT VAR t3; next symbol (s3, t3);
                        |       IF   t3 <> end of file THEN FALSE
                        |       ELIF s1 = "U"          THEN perhaps simple up
                        |       ELIF s1 = "D"          THEN perhaps simple down
                        |                              ELSE FALSE
                        |       FI
                        |  ELSE FALSE
                        |  FI .
                        |
                        |
perhapssimpleup         |perhaps simple up :
                        |  IF   t2 = string  THEN up (s2);       TRUE
                        |  ELIF t2 = integer THEN up (int (s2)); TRUE
                        |                    ELSE                FALSE
                        |  FI .
                        |
                        |
perhapssimpledown       |perhaps simple down :
                        |  IF   t2 = string  THEN down (s2);       TRUE
                        |  ELIF t2 = integer THEN down (int (s2)); TRUE
                        |                    ELSE                  FALSE
                        |  FI .
                        |
                        |
perhapssimplechangeto   |perhaps simple changeto :
                        |  IF   t1 = string AND s2 = "C" AND t3 is string AND t4 is eof
                        |  THEN s1 C s3; TRUE
                        |  ELSE FALSE
                        |  FI .
                        |
                        |
t3isstring              |t3 is string :
                        |  next symbol (s3, t3);
                        |  t3 = string .
```

```
                          |
                          |
   t4iseof                |t4 is eof :
                          |  TEXT VAR s4; INT VAR t4;
                          |  next symbol (s4, t4);
                          |  t4 = end of file .
                          |END PROC std kommando interpreter;
                          |
                          |


edit .....................|PROC edit (FILE VAR f) :
                          |  enable stop;
                          |  IF aktueller editor > 0
                          |       (*wk*)
                          |    THEN ueberschrift neu
                          |  FI ;
                          |  open editor (f, write acc);
                          |  edit (groesster editor, std res, PROC(TEXT CONST) std kommando
                          |       interpreter)
                          |END PROC edit;
                          |
                          |


edit .....................|PROC edit (FILE VAR f, INT CONST x, y, x size, y size) :
                          |  enable stop;
                          |  open editor (groesster editor + 1, f, write acc, x, y, x size, y
                          |       size);
                          |  edit (groesster editor, std res, PROC(TEXT CONST) std kommando
                          |       interpreter)
                          |END PROC edit;
                          |
                          |


edit .....................|PROC edit (FILE VAR f, TEXT CONST res, PROC (TEXT CONST) kdo
                          |     interpreter) :
                          |  enable stop;
                          |  open editor (f, write acc);
                          |  edit (groesster editor, res, PROC(TEXT CONST) kdo interpreter)
                          |END PROC edit;
                          |
                          |


edit .....................|PROC edit :
                          |  IF   aktueller editor > 0
                          |  THEN dateiname einlesen;
                          |       edit (dateiname)
                          |  ELSE edit (last param)
                          |  FI .
                          |
                          |
   dateinameeinlesen      |dateiname einlesen :
                          |  INT VAR x, y; get editcursor (x, y);
                          |  IF x < x size - 17
                          |       (*sh*)
                          |  THEN cursor (x, y);
                          |       out (""15"Dateiname:"14"");
                          |       (x size-14-x) TIMESOUT " ";
                          |       (x size-14-x) TIMESOUT ""8"";
```

```
                              |      TEXT VAR dateiname := std;
                              |      editget (dateiname);
                              |      trailing blanks entfernen;
                              |      quotes entfernen
                              |  ELSE errorstop ("Fenster zu klein")
                              |  FI .
                              |
                              |
     trailingblanksentferne  |trailing blanks entfernen:
                              |  INT VAR i := LENGTH dateiname;
                              |  WHILE (dateiname SUB i) = " " REP i DECR 1 PER;
                              |  dateiname := subtext (dateiname, 1, i) .
                              |
                              |
     quotesentfernen          |quotes entfernen :
                              |  IF   (dateiname SUB 1) = """" AND (dateiname SUB LENGTH dateiname)
                              |       = """"
                              |  THEN dateiname := subtext (dateiname, 2, LENGTH dateiname - 1)
                              |  FI .
                              |END PROC edit;
                              |
                              |


  edit .......................|PROC edit (TEXT CONST filename) :
                              |  IF   filename <> ""
                              |  THEN edit named file
                              |  ELSE errorstop ("Name ungueltig")
                              |  FI .
                              |
                              |
     editnamedfile            |edit named file :
                              |  last param (filename);
                              |  IF   exists (filename) COR yes ("""" + filename + """ neu
                              |       einrichten")
                              |  THEN IF aktueller editor > 0 THEN ueberschrift neu FI;
                              |       (*sh*)
                              |       FILE VAR f := sequential file (modify, filename);
                              |       headline (f, filename); edit (f); last param (filename)
                              |  ELSE errorstop ("")
                              |  FI .
                              |END PROC edit;
                              |
                              |


  edit .......................|PROC edit (TEXT CONST filename, INT CONST x, y, x size, y size) :
                              |  last param (filename);
                              |  IF exists (filename) COR yes ("""" + filename + """ neu
                              |       einrichten")
                              |  THEN FILE VAR f := sequential file (modify, filename);
                              |       headline (f, filename); edit (f, x, y, x size, y size);
                              |       last param (filename)
                              |  ELSE errorstop ("")
                              |  FI
                              |END PROC edit;
                              |
                              |
edit .....................|PROC edit (INT CONST i) :
                          |  edit (i, std res, PROC (TEXT CONST) std kommando interpreter)
                          |END PROC edit;
                          |
                          |


show .....................|PROC show (FILE VAR f) :
                          |  enable stop;
                          |  open editor (f, read acc);
                          |  edit(groesster editor, std res, PROC(TEXT CONST) std kommando
                          |       interpreter);
                          |END PROC show;
                          |
                          |


show .....................|PROC show (TEXT CONST filename) :
                          |     (*sh*)
                          |  last param (filename);
                          |  IF   exists (filename)
                          |  THEN FILE VAR f := sequential file (modify, filename);
                          |       show (f); last param (filename)
                          |  ELSE errorstop ("""" + filename + """ gibt es nicht")
                          |  FI
                          |END PROC show;
                          |
                          |


show .....................|PROC show :
                          |  show (last param)
                          |END PROC show;
                          |
                          |
                          |DATASPACE VAR local space;
                          |INT  VAR zeilenoffset;
                          |TEXT VAR kopierzeile;
                          |
                          |


PUT ......................|OP PUT (TEXT CONST filename) :
                          |  nichts neu;
                          |  IF   mark
                          |  THEN markierten bereich in datei schreiben
                          |  FI .
                          |
                          |
   markiertenbereichindat |markierten bereich in datei schreiben :
                          |  disable stop;
                          |  zieldatei vorbereiten;
                          |  quelldatei oeffnen;
                          |  IF   noch genuegend platz in der zieldatei
                          |       (*sh*)
                          |  THEN zeilenweise kopieren
                          |  ELSE errorstop ("FILE-Ueberlauf")
                          |  FI ;
                          |  quelldatei schliessen;
                          |  zieldatei schliessen;
                          |  set busy indicator .
                          |
zieldateivorbereiten    |zieldatei vorbereiten :
                        |  FRANGE VAR ganze zieldatei;
                        |  IF exists (filename) THEN forget (filename); ueberschrift neu FI;
                        |  FILE VAR destination;
                        |  IF   filename = ""
                        |  THEN forget (local space); local space := nilspace;
                        |       destination := sequential file (output, local space)
                        |  ELSE destination := sequential file (modify, filename) ;
                        |       INT CONST groesse der zieldatei := lines (destination);
                        |           (*sh*)
                        |       set marked range (destination, ganze zieldatei) ;
                        |       output (destination)
                        |  FI .
                        |
                        |
quelldateioeffnen       |quelldatei oeffnen :
                        |  zeilenoffset := mark line no (edfile) - 1;
                        |  INT CONST old line := line no, old col := col;
                        |  FRANGE VAR ganze datei;
                        |  set range (edfile, mark lineno (edfile), mark col (edfile), ganze
                        |       datei);
                        |  input (edfile) .
                        |
                        |
nochgenuegendplatzinde  |noch genuegend platz in der zieldatei :
                        |  lines + groesse der zieldatei < file size .
                        |
                        |
zeilenweisekopieren     |zeilenweise kopieren :
                        |  enable stop;
                        |  satznr neu;
                        |  INT VAR zeile;
                        |  FOR zeile FROM 1 UPTO lines (edfile) REP
                        |    getline (edfile, kopierzeile);
                        |    putline (destination, kopierzeile);
                        |    satznr zeigen
                        |  PER .
                        |
                        |
quelldateischliessen    |quelldatei schliessen :
                        |  modify (edfile);
                        |  set range (edfile, ganze datei);
                        |  to line (old line);
                        |  col (old col) .
                        |
                        |
zieldateischliessen     |zieldatei schliessen :
                        |  IF   filename <> ""
                        |  THEN INT CONST last line written := line no (destination) ;
                        |       modify (destination) ;
                        |       to line (destination, last line written) ;
                        |       col (destination, len (destination) + 1) ;
                        |       bild neu (destination) ;
                        |       set range (destination, ganze zieldatei)
                        |  FI .
                        |END OP PUT;
                        |
                        |
P ......................|OP P (TEXT CONST filename) :
                        |  PUT filename
                        |END OP P ;
                        |
                        |


GET ....................|OP GET (TEXT CONST filename) :
                        |     (*sh*)
                        |  IF   NOT write permission
                        |  THEN errorstop ("Schreibversuch auf 'show'-Datei")
                        |  FI ;
                        |  quelldatei oeffnen;
                        |  IF   nicht mehr genuegend platz im editfile
                        |  THEN quelldatei schliessen; errorstop ("FILE-Ueberlauf")
                        |  FI ;
                        |  disable stop;
                        |  zieldatei oeffnen;
                        |  zeilenweise kopieren ;
                        |  zieldatei schliessen;
                        |  quelldatei schliessen;
                        |  set busy indicator .
                        |
                        |
   quelldateioeffnen    |quelldatei oeffnen :
                        |  FILE VAR source;
                        |  FRANGE VAR ganze quelldatei;
                        |  IF   filename = ""
                        |  THEN source := sequential file (input, local space)
                        |  ELSE IF NOT exists (filename)
                        |       THEN errorstop ("""" + filename + """ gibt es nicht")
                        |       FI ;
                        |       source := sequential file (modify, filename);
                        |       INT CONST old line := line no (source),
                        |                 old col  := col (source);
                        |       set marked range (source, ganze quelldatei);
                        |       input (source)
                        |  FI .
                        |
                        |
   nichtmehrgenuegendplat |nicht mehr genuegend platz im editfile :
                        |  lines (source) + lines >= file size .
                        |
                        |
   zeilenweisekopieren  |zeilenweise kopieren :
                        |  enable stop;
                        |  satznr neu;
                        |  INT VAR zeile;
                        |  FOR zeile FROM 1 UPTO lines (source) REP
                        |    getline (source, kopierzeile);
                        |    putline (edfile, kopierzeile);
                        |    satznr zeigen
                        |  PER .
                        |
                        |
   zieldateioeffnen     |zieldatei oeffnen :
                        |  zeilenoffset := line no - 1;
                        |  leere datei in editfile einschachteln;
                        |  output (edfile) .
                        |
    leeredateiineditfileei |leere datei in editfile einschachteln :
                           |  INT CONST range start col := col;
                           |  FRANGE VAR ganze datei;
                           |  set range (edfile, line no, col, ganze datei);
                           |  IF lines = 1 THEN delete record (edfile) FI .
                           |
                           |
    quelldateischliessen   |quelldatei schliessen :
                           |  IF   filename <> ""
                           |  THEN modify (source);
                           |       set range (source, ganze quelldatei);
                           |       to line (source, old line);
                           |       col (source, old col)
                           |  FI .
                           |
                           |
    zieldateischliessen    |zieldatei schliessen :
                           |  modify (edfile);
                           |  to line (lines);
                           |  col (range start col);
                           |  set range (edfile, ganze datei);
                           |  abschnitt neu (zeilenoffset + 1, lines) .
                           |END OP GET;
                           |
                           |


 G ........................|OP G (TEXT CONST filename) :
                           |  GET filename
                           |END OP G;
                           |
                           |


 len ......................|INT PROC len :
                           |  len (edfile)
                           |END PROC len;
                           |
                           |


 col ......................|PROC col (INT CONST stelle) :
                           |  nichts neu; col (edfile, stelle)
                           |END PROC col;
                           |
                           |


 col ......................|INT PROC col :
                           |  col (edfile)
                           |END PROC col;
                           |
                           |


 limit ....................|PROC limit (INT CONST limit) :
                           |  nichts neu; max line length (edfile, limit)
                           |END PROC limit;
                           |
                           |
limit ....................|INT PROC limit :
                          |  max line length (edfile)
                          |END PROC limit;
                          |
                          |


lines ....................|INT PROC lines :
                          |  lines (edfile)
                          |END PROC lines;
                          |
                          |


lineno ...................|INT PROC line no :
                          |  line no (edfile)
                          |END PROC line no;
                          |
                          |


toline ...................|PROC to line (INT CONST satz nr) :
                          |  satznr neu;
                          |  edfile := editfile;
                          |  IF   satz nr > lines
                          |  THEN toline (edfile, lines); col (len + 1)
                          |  ELSE to line (edfile, satz nr)
                          |  FI
                          |END PROC to line;
                          |
                          |


T ........................|OP T (INT CONST satz nr) :
                          |  to line (satz nr)
                          |END OP T;
                          |
                          |


down .....................|PROC down (INT CONST anz) :
                          |  nichts neu; down (edfile, anz)
                          |END PROC down;
                          |
                          |


D ........................|OP D (INT CONST anz) :
                          |  down (anz)
                          |END OP D;
                          |
                          |


up .......................|PROC up (INT CONST anz) :
                          |  nichts neu; up (edfile, anz)
                          |END PROC up;
                          |
                          |
U .......................|OP U (INT CONST anz) :
                         |  up (anz)
                         |END OP U;
                         |
                         |

down ....................|PROC down (TEXT CONST muster) :
                         |  nichts neu;
                         |  REP
                         |    down (muster, schritt - line no MOD schritt);
                         |    IF   pattern found
                         |    THEN LEAVE down
                         |    ELSE satznr zeigen
                         |    FI
                         |  UNTIL eof PER
                         |END PROC down;
                         |
                         |

D .......................|OP D (TEXT CONST muster) :
                         |  down (muster)
                         |END OP D;
                         |
                         |

down ....................|PROC down (TEXT CONST muster, INT CONST anz) :
                         |  nichts neu; down (edfile, muster, anz)
                         |END PROC down;
                         |
                         |

up ......................|PROC up (TEXT CONST muster) :
                         |  nichts neu;
                         |  REP
                         |    up (muster, (line no - 1) MOD schritt + 1);
                         |    IF   pattern found
                         |    THEN LEAVE up
                         |    ELSE satznr zeigen
                         |    FI
                         |  UNTIL line no = 1 PER
                         |END PROC up;
                         |
                         |

U .......................|OP U (TEXT CONST muster) :
                         |  up (muster)
                         |END OP U;
                         |
                         |

up ......................|PROC up (TEXT CONST muster, INT CONST anz) :
                         |  nichts neu; up (edfile, muster, anz)
                         |END PROC up;
                         |
                         |
downety ..................|PROC downety (TEXT CONST muster) :
                          |  nichts neu;
                          |  IF   NOT at (muster)
                          |  THEN down (muster)
                          |  FI
                          |END PROC downety;
                          |
                          |

downety ..................|PROC downety (TEXT CONST muster, INT CONST anz) :
                          |  nichts neu; downety (edfile, muster, anz)
                          |END PROC downety;
                          |
                          |

uppety ...................|PROC uppety (TEXT CONST muster) :
                          |  nichts neu;
                          |  IF   NOT at (muster)
                          |  THEN up (muster)
                          |  FI
                          |END PROC uppety;
                          |
                          |

uppety ...................|PROC uppety (TEXT CONST muster, INT CONST anz) :
                          |  nichts neu; uppety (edfile, muster, anz)
                          |END PROC uppety;
                          |
                          |

C ........................|OP C (TEXT CONST old, new) :
                          |  change to (old, new)
                          |END OP C;
                          |

C ........................|OP C (TEXT CONST replacement) :
                          |  IF   NOT write permission
                          |       (*sh*)
                          |  THEN errorstop ("Schreibversuch auf 'show'-Datei")
                          |  FI ;
                          |  IF   at (edfile, match(0))
                          |  THEN zeile neu; change (edfile, matchpos(0), matchend(0),
                          |       replacement)
                          |  FI
                          |END OP C;
                          |

changeto .................|PROC change to (TEXT CONST old, new) :
                          |  IF   NOT write permission
                          |       (*sh*)
                          |  THEN errorstop ("Schreibversuch auf 'show'-Datei")
                          |  FI ;
                          |  nichts neu;
                          |  REP
                          |    downety (old, schritt - line no MOD schritt);
                          |    IF   pattern found
```

```
                         |    THEN change (edfile, matchpos(0), matchend(0), new);
                         |         col (col + LENGTH new); zeile neu;
                         |         LEAVE changeto
                         |    ELSE satznr zeigen
                         |    FI
                         |  UNTIL eof PER
                         |END PROC change to;
                         |
                         |

CA ......................|OP CA (TEXT CONST old, new) :
                         |  change all (old, new)
                         |END OP CA;
                         |
                         |

changeall ...............|PROC change all (TEXT CONST old, new) :
                         |  WHILE NOT eof REP old C new PER
                         |END PROC change all;
                         |
                         |

eof .....................|BOOL PROC eof :
                         |  eof (edfile)
                         |END PROC eof;
                         |
                         |

mark ....................|BOOL PROC mark :
                         |  mark (edfile)
                         |END PROC mark;
                         |
                         |

mark ....................|PROC mark (BOOL CONST mark on) :
                         |  nichts neu;
                         |  IF   mark on
                         |  THEN mark (edfile, line no, col)
                         |  ELSE mark (edfile, 0, 0)
                         |  FI
                         |END PROC mark;
                         |
                         |

at ......................|BOOL PROC at (TEXT CONST pattern) :
                         |  at (edfile, pattern)
                         |END PROC at;
                         |

word ....................|TEXT PROC word :
                         |  word (edfile)
                         |END PROC word;
                         |
                         |
word .....................|TEXT PROC word (TEXT CONST sep) :
                          |  word (edfile, sep)
                          |END PROC word;
                          |
                          |

word .....................|TEXT PROC word (INT CONST len) :
                          | word (edfile, len)
                          |END PROC word;
                          |
                          |
                          |LET no access = 0,
                          |    edit access = 1,
                          |    output access = 2;
                          |
                          |INT VAR last note file mode;
                          |FILE VAR notebook;
                          |INITFLAG VAR this packet := FALSE;
                          |DATASPACE VAR note ds;
                          |
                          |

note .....................|PROC note (TEXT CONST text) :
                          |  access note file (output access);
                          |  write (notebook, text)
                          |END PROC note;
                          |
                          |

note .....................|PROC note (INT CONST number) :
                          |  access note file (output access);
                          |  put (notebook, number)
                          |END PROC note;
                          |
                          |

noteline .................|PROC note line :
                          |  access note file (output access);
                          |  line (notebook)
                          |END PROC note line;
                          |
                          |

anythingnoted ............|BOOL PROC anything noted :
                          |  access note file (no access);
                          |  last note file mode = output access
                          |END PROC anything noted;
                          |
                          |

notefile .................|FILE PROC note file :
                          |  access note file (output access);
                          |  notebook
                          |END PROC note file;
                          |
                          |
noteedit .................|PROC note edit (FILE VAR context) :
                          |     (*sh*)
                          |  access note file (edit access);
                          |  make notebook erasable;
                          |  IF aktueller editor = 0
                          |    THEN open editor (1, context, write acc, 1, 1, x size - 1, y
                          |         size)
                          |  FI ;
                          |  get window size;
                          |  IF   window large enough
                          |  THEN include note editor;
                          |       edit (aktueller editor-1, aktueller editor, aktueller
                          |            editor-1,
                          |             std res, PROC (TEXT CONST) std kommando interpreter)
                          |  FI .
                          |
                          |
   getwindowsize          |get window size :
                          |  INT VAR x, y, windows x size, windows y size;
                          |  get window (x, y, windows x size, windows y size) .
                          |
                          |
   windowlargeenough      |window large enough :
                          |  windows y size > 4 .
                          |
                          |
   includenoteeditor      |include note editor :
                          |  open editor (aktueller editor + 1, notebook, write acc,
                          |               x, y + (windows y size + 1) DIV 2,
                          |               windows x size, windows y size DIV 2) .
                          |
                          |
   makenotebookerasable   |make notebook erasable :
                          |  last note file mode := edit access .
                          |END PROC note edit;
                          |
                          |


noteedit .................|PROC note edit :
                          |  access note file (edit access);
                          |  make notebook erasable;
                          |  edit (notebook) .
                          |
                          |
   makenotebookerasable   |make notebook erasable :
                          |  last note file mode := edit access .
                          |END PROC note edit;
                          |
                          |


accessnotefile ...........|PROC access note file (INT CONST new mode) :
                          |  disable stop;
                          |  initialize note ds if necessary;
                          |  IF   last note file mode < new mode
                          |  THEN forget (note ds);
                          |       note ds := nilspace;
                          |       notebook := sequential file (output, note ds);
                          |       headline (notebook, "notebook");
                          |       last note file mode := new mode
                          |  FI .
```

```
                           |
                           |
initializenotedsifnece     |initialize note ds if necessary :
                           |  IF   NOT initialized (this packet)
                           |  THEN note ds := nilspace;
                           |       last note file mode := no access
                           |  FI .
                           |END PROC access note file;
                           |
                           |END PACKET editor functions;
```

```
                              |(* ------------------- VERSION 2    06.03.86 ------------------- *)
stdtransput ***************   |PACKET std transput DEFINES
                              |
                              |    sysout ,
                              |    sysin ,
                              |    put ,
                              |    putline ,
                              |    line ,
                              |    page ,
                              |    write ,
                              |    get ,
                              |    getline ,
                              |    get secret line :
                              |
                              |
                              |LET cr               = ""13"" ,
                              |    cr lf            = ""13""10"" ,
                              |    home clear       = ""1""4"" ,
                              |    esc              = ""27"" ,
                              |    rubout           = ""12"" ,
                              |    bell             = ""7"" ,
                              |    back blank back = ""8" "8"" ,
                              |    del line cr lf  = ""5""13""10"" ;
                              |
                              |TEXT VAR number word , exit char ;
                              |
                              |BOOL VAR console output := TRUE, console input := TRUE ;
                              |
                              |FILE VAR outfile, infile ;
                              |TEXT VAR outfile name := "", infile name := "" ;
                              |
                              |


sysout ...................    |PROC sysout (TEXT CONST file name) :
                              |
                              |  outfile name := file name ;
                              |  IF file name = ""
                              |    THEN console output := TRUE
                              |    ELSE outfile := sequential file (output, file name) ;
                              |         console output := FALSE
                              |  FI
                              |
                              |ENDPROC sysout ;
                              |


sysout ...................    |TEXT PROC sysout :
                              |  outfile name
                              |ENDPROC sysout ;
                              |


sysin ....................    |PROC sysin (TEXT CONST file name) :
                              |
                              |  infile name := file name ;
                              |  IF file name = ""
                              |    THEN console input := TRUE
                              |    ELSE infile := sequential file (input, file name) ;
                              |         console input := FALSE
                              |  FI
                              |
                        |ENDPROC sysin ;
                        |

sysin ..................|TEXT PROC sysin :
                        |  infile name
                        |ENDPROC sysin ;
                        |
                        |

put ....................|PROC put (TEXT CONST word) :
                        |
                        |  IF console output
                        |    THEN out (word) ; out (" ")
                        |    ELSE put (outfile, word)
                        |  FI
                        |
                        |ENDPROC put ;
                        |

put ....................|PROC put (INT CONST number) :
                        |
                        |  put (text (number))
                        |
                        |ENDPROC put ;
                        |

put ....................|PROC put (REAL CONST number) :
                        |
                        |  put (text (number))
                        |
                        |ENDPROC put ;
                        |

putline ................|PROC putline (TEXT CONST textline) :
                        |
                        |  IF console output
                        |    THEN out (textline) ; out (cr lf)
                        |    ELSE putline (outfile, textline)
                        |  FI
                        |
                        |ENDPROC putline ;
                        |

line ...................|PROC line :
                        |
                        |  IF console output
                        |    THEN out (cr lf)
                        |    ELSE line (outfile)
                        |  FI
                        |
                        |ENDPROC line ;
                        |
line .....................|PROC line (INT CONST times) :
                          |
                          |  INT VAR i ;
                          |  FOR i FROM 1 UPTO times REP
                          |    line
                          |  PER
                          |
                          |ENDPROC line ;
                          |


page .....................|PROC page :
                          |
                          |  IF console output
                          |    THEN out (home clear)
                          |  FI
                          |
                          |ENDPROC page ;
                          |


write ....................|PROC write (TEXT CONST word) :
                          |
                          |  IF console output
                          |    THEN out (word)
                          |    ELSE write (outfile, word)
                          |  FI
                          |
                          |ENDPROC write ;
                          |
                          |


get ......................|PROC get (TEXT VAR word) :
                          |
                          |  IF console input
                          |    THEN get from console
                          |    ELSE get (infile, word)
                          |  FI .
                          |
                          |
   getfromconsole         |get from console :
                          |  REP
                          |    word := "" ;
                          |    editget (word, " ", "", exit char) ;
                          |    echoe exit char
                          |  UNTIL word <> "" AND word <> " " PER ;
                          |  delete leading blanks .
                          |
                          |
   deleteleadingblanks    |delete leading blanks :
                          |  WHILE (word SUB 1) = " " REP
                          |    word := subtext (word,2)
                          |  PER .
                          |
                          |ENDPROC get ;
                          |


get ......................|PROC get (TEXT VAR word, TEXT CONST separator) :
                          |
                          |  IF console input
                          |    THEN get from console
                          |    ELSE get (infile, word, separator)
                          |  FI .
                          |
                          |
   getfromconsole         |get from console :
                          |  word := "" ;
                          |  editget (word, separator, "", exit char) ;
                          |  echoe exit char .
                          |
                          |ENDPROC get ;
                          |


echoeexitchar ............|PROC echoe exit char :
                          |
                          |  IF exit char = ""13""
                          |    THEN out (""13""10"")
                          |    ELSE out (exit char)
                          |  FI
                          |
                          |ENDPROC echoe exit char ;
                          |


get ......................|PROC get (INT VAR number) :
                          |
                          |  get (number word) ;
                          |  number := int (number word)
                          |
                          |ENDPROC get ;
                          |


get ......................|PROC get (REAL VAR number) :
                          |
                          |  get (number word) ;
                          |  number := real (number word)
                          |
                          |ENDPROC get ;
                          |


get ......................|PROC get (TEXT VAR word, INT CONST length) :
                          |
                          |  IF console input
                          |    THEN get from console
                          |    ELSE get (infile, word, length)
                          |  FI .
                          |
                          |
   getfromconsole         |get from console :
                          |  word := "" ;
                          |  editget (word, length, exit char) ;
                          |  echoe exit char .
                          |
                          |ENDPROC get ;
                          |
getline ..................|PROC getline (TEXT VAR textline) :
                          |
                          |  IF console input
                          |    THEN get from console
                          |    ELSE getline (infile, textline)
                          |  FI .
                          |
                          |
   getfromconsole         |get from console :
                          |  textline := "" ;
                          |  editget (textline, "", "", exit char) ;
                          |  echoe exit char
                          |
                          |ENDPROC getline ;
                          |


getsecretline ............|PROC get secret line (TEXT VAR textline) :
                          |
                          |  TEXT VAR char ;
                          |  textline := "" ;
                          |  get start cursor position ;
                          |  get line very secret ;
                          |  IF char = esc
                          |    THEN get line little secret
                          |  FI ;
                          |  cursor to start position ;
                          |  out (del line cr lf) .
                          |
                          |
   getlineverysecret      |get line very secret :
                          |  REP
                          |    inchar (char) ;
                          |    IF char = esc OR char = cr
                          |      THEN LEAVE get line very secret
                          |    ELIF char = rubout
                          |      THEN delete last char
                          |    ELIF char >= " "
                          |      THEN textline CAT char ;
                          |           out (".")
                          |    ELSE   out (bell)
                          |    FI
                          |  PER .
                          |
                          |
   deletelastchar         |delete last char :
                          |  IF LENGTH textline = 0
                          |    THEN out (bell)
                          |    ELSE out (back blank back) ;
                          |         delete char (textline, LENGTH textline)
                          |  FI .
                          |
                          |
   getlinelittlesecret    |get line little secret :
                          |  cursor to start position ;
                          |  editget (textline, "", "", exit char) .
                          |
                          |
   getstartcursorposition |get start cursor position :
                          |  INT VAR x, y;
                          |  get cursor (x, y) .
                          |
 258      cursortostartposition  |cursor to start position :
 259                             |  cursor (x, y) .
 260                             |
 261                             |ENDPROC get secret line ;
 262                             |
 263                             |ENDPACKET std transput ;
```

```
                             |
localmanagerpart2 ********|PACKET local manager part 2 DEFINES           (* Autor: J.Liedtke *
                             |                                              (* Stand: 25.02.85  *
                             |  list :
                             |
                             |
                             |TEXT VAR file name, status text;
                             |
                             |


list .....................|PROC list :
                             |
                             |  disable stop ;
                             |  DATASPACE VAR ds := nilspace ;
                             |  FILE VAR list file := sequential file (output, ds) ;
                             |  headline (list file, "list") ;
                             |  list (list file) ;
                             |  show (list file) ;
                             |  forget (ds) .
                             |
                             |ENDPROC list ;
                             |


list .....................|PROC list (FILE VAR f) :
                             |
                             |  enable stop ;
                             |  begin list ;
                             |  putline (f, "") ;
                             |  REP
                             |    get list entry (file name, status text) ;
                             |    IF file name = ""
                             |      THEN LEAVE list
                             |    FI ;
                             |    write (f, status text + "  """ ) ;
                             |    write (f, file name) ;
                             |    write (f, """") ;
                             |    line (f)
                             |  PER .
                             |
                             |ENDPROC list ;
                             |
                             |ENDPACKET local manager part 2 ;
```

```
eumelcoderpart1 ***********|PACKET eumel coder part 1                          (* Autor: U.
                           |     Bartling *)
                           |       DEFINES run, run again,
                           |               insert,
                           |               prot, prot off,
                           |               check, check on, check off,
                           |               warnings, warnings on, warnings off,
                           |
                           |               help, bulletin, packets
                           |               :
                           |
                           |(*********************************************************************
                           |     ******)
                           |(*
                           |          *)
                           |(*                         E U M E L  -  C O D E R
                           |          *)
                           |(*
                           |          *)
                           |(*
                           |          *)
                           |(*   Zur Beschreibung des Coders siehe
                           |          *)
                           |(*         U.Bartling, J. Liedtke: EUMEL-Coder-Interface
                           |          *)
                           |(*
                           |          *)
                           |(*   Stand der Dokumentation  : 13.02.1986
                           |          *)
                           |(*   Stand der Implementation : 16.04.1986
                           |          *)
                           |(*
                           |          *)
                           |(*
                           |          *)
                           |(*********************************************************************
                           |     ******)
                           |
                           |
                           |                (***** Globale Variable *****)
                           |
                           |TEXT VAR object name;
                           |
                           |FILE VAR bulletin file;
                           |
                           |INT VAR hash table pointer, nt link, permanent pointer, param link,
                           |        index, mode, word;
                           |
                           |BOOL VAR found, end of params;
                           |
(***************************************************************************)
(*                                                                         *)
(*                   1. Interface zum ELAN-Compiler          10.04.1986    *)
(*                                  1.7.5.4                                *)
(*                                                                         *)
(*  Beschreibung der     Tabellen (-groessen),                             *)
(*                       internen Vercodung von Typen                      *)
(*                  und  Kennungen .                                       *)
(*  Initialisieren und Beenden des Compilers,                              *)
(*  Lesen aus und Schreiben in Namens- bzw. Permanent-Tabelle              *)
(*                                                                         *)
(***************************************************************************)


LET begin of hash table          =     0 ,
    end of hash table            =  1023 ,

    begin of permanent table     = 22784 ,
      before first pt entry      = 22784 ,
      first permanent entry      = 22785 ,
    end of permanent table       = 32767 ,

    wordlength                   =     1 ,  (* compile  u n d  run time *)
    two word length              =     2 ,
    three word length            =     3 ,

    permanent param const              = 10000 ,
    permanent param var                = 20000 ,
    permanent proc op                  = 30000 ,
    permanent type                     = 30000 ,
    permanent row                      =    10 ,
    permanent struct                   =    11 ,
    permanent param proc               =    12 ,
(*  permanent param proc end marker    =     0 , *)
    permanent type field               =     0 ,

    ptt limit                          = 10000 ,
    begin of pt minus ptt limit        = 12784 ,

    void                               =     0 ,
    int                                =     1 ,
    real                               =     2 ,
    string                             =     3 ,
    bool                               =     5 ,
    bool result                        =     6 ,
    dataspace                          =     7 ,
    row                                =    10 ,
    struct                             =    11 ,

                              |     const                          =    1 ,
                              |     var                            =    2 ,
                              |(*   proc                           =    3 , *)
                              |(*   denoter                        =    5 , *)
                              |     bold                           =    2 ,
                              |
                              |     ins                            = TRUE ,
                              |     no ins                         = FALSE ,
                              |     no lst                         = FALSE ,
                              |     sermon                         = TRUE  ,
                              |     no sermon                      = FALSE ,
                              |
                              |     run again mode                 =    0 ,
                              |     compile file mode              =    1 ,
                              |
                              |     warning message                =    2 ,
                              |     error message                  =    4 ,
                              |
                              |     point line                     = "..............." ;
                              |
                              |INT CONST permanent packet         :=    -2 ,
                              |          permanent end            :=    -3 ;
                              |
                              |
                              |INT  VAR run again mod nr          :=    0 ;
                              |
                              |
                              |                (***** Start/Ende *****)
                              |


elan .........................|PROC elan (INT CONST mode, FILE VAR source, TEXT CONST line,
                              |           INT VAR start module number, BOOL CONST ins, lst, rtc,
   +                          |                ser) :
                              |  EXTERNAL 256
                              |ENDPROC elan ;
                              |
                              |              (***** Hash/Namenstabelle *****)
                              |.
                              |
   nexthashentry              |next hash entry :
                              |     hash table pointer INCR wordlength .
                              |
                              |
   endofhashtablereached      |end of hash table reached :
                              |     hash table pointer > end of hash table .
                              |
                              |
   yetanotherntentry          |yet another nt entry :
                              |     nt link := cdb int (nt link) ;
                              |     nt link <> 0 . ;
                              |


declareobject ................|PROC declare object (TEXT CONST name, INT VAR nt link, pt pointer) :
                              |    EXTERNAL 10031
                              |ENDPROC declare object ;
                              |
toobject ................|PROC to object (TEXT CONST searched object) :
                         |     hash ;
                         |     search nt entry .
                         |
                         |
   hash                  |hash :
                         |     hash code := 0 ;
                         |     FOR index FROM 1 UPTO LENGTH searched object REP
                         |         addmult cyclic
                         |     ENDREP .
                         |
                         |
   addmultcyclic         |addmult cyclic :
                         |     hash code INCR hash code ;
                         |     IF hash code > end of hash table THEN wrap around FI ;
                         |     hash code := (hash code + code (searched object SUB index)) MOD
                         |          1024 .
                         |
                         |
   wraparound            |wrap around :
                         |     hash code DECR end of hash table .
                         |
                         |
   hashcode              |hash code : nt link .
                         |
                         |
   searchntentry         |search nt entry :
                         |     found := FALSE ;
                         |     WHILE yet another nt entry REP
                         |          read current entry ;
                         |          IF object name = searched object
                         |             THEN found := TRUE ;
                         |                  LEAVE to object
                         |          FI
                         |     PER .
                         |
                         |
   readcurrententry      |read current entry :
                         |     permanent pointer := cdb int (nt link + wordlength) ;
                         |     object name := cdb text (nt link + two word length)
                         |ENDPROC to object ;
                         |
                         |
                         |               (***** Permanent Tabelle *****)
                         |.
                         |
   nextprocedure         |next procedure :
                         |     permanent pointer := cdb int (permanent pointer) . ;
                         |


nextptparam .............|PROC next pt param :
                         |     mode := cdb int (param link) MOD ptt limit ;
                         |     param link INCR wordlength ;
                         |     IF   mode = permanent row    THEN skip over permanent row
                         |     ELIF mode = permanent struct THEN skip over permanent struct
                         |     FI ;
                         |     set end marker if end of list .
                         |
   skipoverpermanentrow   |skip over permanent row :
                          |     param link INCR wordlength ;
                          |     next pt param .
                          |
                          |
   skipoverpermanentstruc |skip over permanent struct :
                          |     REP
                          |         next pt param ;
                          |         mode := cdb int (param link)
                          |     UNTIL mode = permanent type field PER ;
                          |     param link INCR wordlength
                          |ENDPROC next pt param ;
                          |


setendmarkerifendoflis ...|PROC set end marker if end of list :
                          |     mode := cdb int (param link) ;
                          |     end of params := mode >= permanent proc op OR mode <= 0
                          |ENDPROC set end marker if end of list ;
                          |


gettypeandmode ...........|PROC get type and mode (INT VAR type) :
                          |     mode := cdb int (param link) ;
                          |     IF mode = permanent param proc THEN type of param proc
                          |                                    ELSE type of object
                          |     FI .
                          |
                          |
   typeofparamproc        |type of param proc :
                          |     param link INCR wordlength ;
                          |     get type and mode (type) ;
                          |     mode := permanent param proc .
                          |
                          |
   typeofobject           |type of object :
                          |     IF mode < 0 THEN type := 2769 + (32767 + mode) ;
                          |                      mode := 0
                          |                 ELSE type := mode MOD ptt limit ;
                          |                      mode DECR type ;
                          |                      translate type if necessary ;
                          |                      translate mode if necessary
                          |     FI .
                          |
                          |
   translatetypeifnecessa |translate type if necessary :
                          |     IF permanent row or struct THEN translate type FI .
                          |
                          |
   translatetype          |translate type :
                          |     type := param link - begin of pt minus ptt limit .
                          |
                          |
   translatemodeifnecessa |translate mode if necessary :
                          |    IF   mode = permanent param const THEN mode := const
                          |    ELIF mode = permanent param var   THEN mode := var
                          |    FI .
                          |
                          |
   permanentroworstruct   |permanent row or struct :
                          |     type = permanent row OR type = permanent struct
                          |ENDPROC get type and mode ;
```

```
                              |
                              |
                              |                 (***** Allgemeine Zugriffsprozeduren *****)
                              |


    cdbint ...................|INT PROC cdb int (INT CONST index) :
                              |   EXTERNAL 116
                              |ENDPROC cdb int ;
                              |


    cdbtext ..................|TEXT PROC cdb text (INT CONST index) :
                              |   EXTERNAL 117
                              |ENDPROC cdb text ;
                              |
                               |(*********************************************************************
                               |     ******)
                               |(*
                               |          *)
                               |(*              10. Inspector
                               |     16.04.1986  *)
                               |(*
                               |          *)
                               |(*********************************************************************
                               |     ******)
                               |
                               |
                               |
                               |INT VAR line number, pattern length, packet link,
                               |        begin of packet, last packet entry, indentation;
                               |
                               |TEXT VAR bulletin name, type and mode, pattern, buffer;
                               |
                               |DATASPACE VAR bulletin ds :: nilspace ;
                               |
                               |.packet name :
                               |     cdb text (cdb int(packet link + wordlength) + two word length) .
                               |
                               |.within editor :
                               |    aktueller editor > 0 .  ;
                               |

nameoftype ....................|PROC name of type (INT CONST type) :
                               |            SELECT type OF
                               |     CASE void               :
                               |     CASE int                : type and mode CAT "INT"
                               |     CASE real               : type and mode CAT "REAL"
                               |     CASE string             : type and mode CAT "TEXT"
                               |     CASE bool, bool result  : type and mode CAT "BOOL"
                               |     CASE dataspace          : type and mode CAT "DATASPACE"
                               |     CASE row                : type and mode CAT "ROW "
                               |     CASE struct             : type and mode CAT "STRUCT"
                               |     OTHERWISE               : complex type
                               |            ENDSELECT .
                               |
                               |
   complextype                 |complex type :
                               |     IF type > ptt limit THEN perhaps permanent struct or row
                               |                         ELSE get complex type
                               |     FI .
                               |
                               |
   perhapspermanentstruct      |perhaps permanent struct or row :
                               |     index := type + begin of pt minus ptt limit ;
                               |     mode := cdb int (index) MOD ptt limit ;
                               |     IF   mode = permanent row    THEN get permanent row
                               |     ELIF mode = permanent struct THEN get permanent struct
                               |                                  ELSE type and mode CAT "-"
                               |     FI .
                               |
                               |
   getcomplextype              |get complex type :
                               |     index := type + begin of permanent table ;
                               |     IF is complex type THEN get name
                               |                        ELSE type and mode CAT "-"
                               |     FI .
```

```
                              |
                              |
     iscomplextype            |is complex type :
                              |     permanent type definition mode = permanent type .
                              |
                              |
     getname                  |get name :
                              |     type and mode CAT cdb text (link to type name + two word
   +                          |          length) .
                              |
                              |
     linktotypename           |link to type name :
                              |     cdb int (index + three word length) .
                              |
                              |
     permanenttypedefinitio   |permanent type definition mode :
                              |     cdb int (index + wordlength) .
                              |
                              |
     getpermanentrow          |get permanent row :
                              |     INT VAR t;
                              |     type and mode CAT "ROW " ;
                              |     type and mode CAT text (cdb int (index + wordlength)) ;
                              |     type and mode CAT " " ;
                              |     param link := index + two wordlength ;
                              |     get type and mode (t) ;
                              |     name of type (t) .
                              |
                              |
     getpermanentstruct       |get permanent struct :
                              |     type and mode CAT "STRUCT ( ... )"
                              |ENDPROC name of type ;
                              |


  help .......................|PROC help (TEXT CONST proc name) :
                              |     prep bulletin ;
                              |     prep help ;
                              |     scan (object name) ;
                              |     next symbol (pattern) ;
                              |     packet link := end of permanent table ;
                              |     IF function = 0 THEN standard help
                              |                     ELSE asterisk help
                              |     FI .
                              |
                              |
     prephelp                 |prep help :
                              |     object name := compress (proc name) ;
                              |     INT VAR function :: 0 ;
                              |     INT CONST l :: LENGTH object name ;
                              |     IF l > 1 AND object name <> "**"
                              |        THEN IF (object name SUB 1) = "*"
                              |               THEN function INCR 2 ;
                              |                    delete char (object name, 1)
                              |             FI ;
                              |             IF (object name SUB 1) = "*"
                              |               THEN function INCR 1 ;
                              |                    delete char (object name, 1)
                              |             FI ;
                              |             IF another asterisk THEN wrong function FI
                              |     FI.
                              |
anotherasterisk          |another asterisk :
                         |     pos (object name, "*") <> 0 .
                         |
                         |
wrongfunction            |wrong function :
                         |     errorstop ("unzulaessige Sternfunktion") .
                         |
                         |
standardhelp             |standard help :
                         |     to object (pattern) ;
                         |     IF found THEN display
                         |              ELSE error stop ("unbekannt: " + proc name)
                         |     FI .
                         |
                         |
display                  |display :
                         |     WHILE permanent pointer <> 0 REP
                         |         put name of packet if necessary ;
                         |         put specifications (pattern) ;
                         |         next procedure
                         |     ENDREP ;
                         |     show bulletin file .
                         |
                         |
putnameofpacketifneces   |put name of packet if necessary :
                         |     IF new packet THEN packet link := permanent pointer ;
                         |                        find begin of packet ;
                         |                        writeline (2) ;
                         |                        write packet name
                         |     FI .
                         |
                         |
findbeginofpacket        |find begin of packet :
                         |     REP
                         |         packet link DECR wordlength
                         |     UNTIL begin of packet found PER .
                         |
                         |
beginofpacketfound       |begin of packet found :
                         |     cdb int (packet link) = permanent packet .
                         |
                         |
newpacket                |new packet :
                         |     permanent pointer < packet link .
                         |
                         |
asteriskhelp             |asterisk help :
                         |     hash table pointer := begin of hash table ;
                         |     pattern length := LENGTH pattern - 1 ;
                         |     REP
                         |        list all objects in current hash table chain ;
                         |        next hash entry
                         |     UNTIL end of hash table reached ENDREP ;
                         |     show bulletin file .
                         |
                         |
listallobjectsincurren   |list all objects in current hash table chain :
                         |     nt link := hash table pointer ;
                         |     WHILE yet another nt entry REP
                         |        permanent pointer := cdb int (nt link + wordlength) ;
                         |        object name := cdb text (nt link + two word length) ;
                         |        IF matching THEN into bulletin FI
```

```
                             |     PER .
                             |
                             |
   matching                  |matching :
                             |     INT CONST p :: pos (object name, pattern) ;
                             |     SELECT function OF
                             |         CASE 1 :   p <> 0 AND p = LENGTH object name - pattern
                             |              length
                             |         CASE 2 :   p = 1
                             |         CASE 3 :   p <> 0
                             |         OTHERWISE  FALSE
                             |     ENDSELECT .
                             |
                             |
   intobulletin              |into bulletin :
                             |     object names into bulletin (BOOL PROC not end of chain)
                             |ENDPROC help ;
                             |


notendofchain ...............|BOOL PROC not end of chain :
                             |     permanent pointer <> 0
                             |ENDPROC not end of chain ;
                             |


writepacketname .............|PROC write packet name :
                             |     indentation := 0 ;
                             |     write line ;
                             |     write bulletin line ("PACKET ") ;
                             |     indentation := 7 ;
                             |     object name := packet name ;
                             |     write bulletin line (object name) ;
                             |     write bulletin line (":") ;
                             |     writeline (2)
                             |ENDPROC write packet name ;
                             |


putspecifications ...........|PROC put specifications (TEXT CONST proc name) :
                             |     put obj name (proc name) ;
                             |     to first param ;
                             |     IF NOT end of params THEN put param list FI ;
                             |     put result ;
                             |     writeline .
                             |
                             |
   tofirstparam              |to first param :
                             |     param link := permanent pointer + word length ;
                             |     set end marker if end of list .
                             |
                             |
   putresult                 |put result :
                             |     INT VAR type;
                             |     get type and mode (type) ;
                             |     IF type <> void THEN type and mode := "  --> " ;
                             |                          name of type (type) ;
                             |                          write bulletin line (type and mode)
                             |     FI
                             |ENDPROC put specifications ;
                             |
putparamlist ............|PROC put param list :
                         |     write bulletin line (" (") ;
                         |     REP
                         |         INT VAR type, param mode;
                         |         get type and mode (type) ;
                         |         param mode := mode ;
                         |         put type and mode ;
                         |         maybe param proc ;
                         |         next pt param ;
                         |         IF end of params THEN write bulletin line (")") ;
                         |                               LEAVE put param list
                         |         FI ;
                         |         write bulletin line (", ") ;
                         |     PER .
                         |
                         |
   puttypeandmode        |put type and mode :
                         |     type and mode := "" ;
                         |     name of type (type) ;
                         |     type and mode CAT name of mode ;
                         |     write bulletin line (type and mode) .
                         |
                         |
   nameofmode            |name of mode :
                         |     IF   param mode = const THEN " CONST"
                         |     ELIF param mode = var   THEN " VAR"
                         |                             ELSE " PROC"
                         |     FI .
                         |
                         |
   maybeparamproc        |maybe param proc :
                         |     IF mode = permanent param proc THEN put virtual params FI .
                         |
                         |
   putvirtualparams      |put virtual params :
                         |     skip over result type if complex type ;
                         |     IF NOT end of virtual params THEN put param list FI.
                         |
                         |
   skipoverresulttypeifco |skip over result type if complex type :
                         |     next pt param .
                         |
                         |
   endofvirtualparams    |end of virtual params :
                         |    end of params
                         |ENDPROC put param list ;
                         |


nextpacket ..............|PROC next packet :
                         |     REP
                         |         packet link INCR wordlength ;
                         |         word := cdb int (packet link) ;
                         |         IF   word = permanent packet THEN true return
                         |         ELIF end of permanents       THEN false return
                         |         FI ;
                         |     ENDREP .
                         |
                         |
   truereturn            |true return :
                         |     found := TRUE ;
                         |     LEAVE next packet .
```

```
                             |
                             |
    falsereturn              |false return :
                             |     found := FALSE ;
                             |     LEAVE next packet .
                             |
                             |
    endofpermanents          |end of permanents :
                             |     word = permanent end OR packet link > end of permanent table
                             |ENDPROC next packet ;
                             |


prepbulletin ................|PROC prep bulletin :
                             |     forget (bulletin ds) ;
                             |     bulletin ds := nilspace ;
                             |     bulletin file := sequential file (output, bulletin ds) ;
                             |     line number := 0 ;
                             |     buffer := ""
                             |ENDPROC prep bulletin ;
                             |


showbulletinfile ............|PROC show bulletin file :
                             |     IF within editor THEN ueberschrift neu FI ;
                             |     DATASPACE VAR local ds :: bulletin ds ;
                             |     FILE VAR local file :: sequential file (modify, local ds) ;
                             |     show (local file) ;
                             |     forget (local ds)
                             |ENDPROC show bulletin file ;
                             |


writebulletinline ...........|PROC write bulletin line (TEXT CONST line) :
                             |     IF LENGTH buffer + LENGTH line > 75 THEN writeline FI ;
                             |     buffer CAT line
                             |ENDPROC write bulletin line ;
                             |


writeline ...................|PROC writeline  :
                             |     write (bulletin file, buffer) ;
                             |     line (bulletin file) ;
                             |     line number INCR 1 ;
                             |     cout (line number) ;
                             |     buffer := indentation * " "
                             |ENDPROC writeline ;
                             |


writeline ...................|PROC writeline (INT CONST times) :
                             |     IF LENGTH compress(buffer) <> 0 THEN index := times - 1 ;
                             |                                          writeline
                             |                                     ELSE index := times
                             |     FI ;
                             |     line (bulletin file, index) ;
                             |     line number INCR index;
                             |     indentation := 0 ;
                             |     cout (line number)
                             |ENDPROC writeline ;
                             |
bulletin ................|PROC bulletin (TEXT CONST packet name) :
                         |     prep bulletin ;
                         |     scan (packet name) ;
                         |     next symbol (pattern) ;
                         |     to packet ;
                         |     IF found THEN list packet ;
                         |                   show bulletin file
                         |              ELSE error stop (packet name + " ist kein Paketname")
                         |     FI .
                         |
                         |
   topacket              |to packet :
                         |     last packet entry := 0 ;
                         |     get nametab link of packet name ;
                         |     packet link := before first pt entry ;
                         |     REP
                         |         packet link INCR wordlength ;
                         |         word := cdb int (packet link) ;
                         |         IF word < 0 THEN IF   word = permanent packet THEN packet
                         |              found
                         |                         ELIF word = permanent end    THEN return
                         |                         FI
                         |         FI
                         |     ENDREP .
                         |
                         |
   getnametablinkofpacket |get nametab link of packet name :
                         |     to object (pattern) ;
                         |     IF NOT found THEN error stop ("unbekanntes Paket :" + packet
                         |          name) ;
                         |                       LEAVE to packet
                         |     FI .
                         |
                         |
   packetfound           |packet found :
                         |     IF cdb int (packet link + wordlength) = nt link
                         |        THEN last packet entry := packet link FI .
                         |
                         |
   return                |return :
                         |     IF last packet entry <> 0 THEN found := TRUE ;
                         |                                    packet link := last packet entry
                         |                               ELSE found := FALSE
                         |     FI ;
                         |     LEAVE to packet
                         |ENDPROC bulletin ;
                         |


listpacket ..............|PROC list packet :
                         |     begin of packet := packet link + word length ;
                         |     write packet name ;
                         |     find end of packet ;
                         |     run through nametab and list all packet objects .
                         |
                         |
   findendofpacket       |find end of packet :
                         |     last packet entry := begin of packet ;
                         |     REP
                         |         last packet entry INCR wordlength ;
                         |         word := cdb int (last packet entry) ;
                         |     UNTIL end of packet entries PER .
```

```
                                |
                                |
      endofpacketentries        |end of packet entries :
                                |     word = permanent packet OR word = permanent end .
                                |
                                |
      runthroughnametabandli    |run through nametab and list all packet objects :
                                |     hashtable pointer := begin of hashtable ;
                                |     REP
                                |         nt link := hashtable pointer ;
                                |         list objects of current packet in this chain ;
                                |         next hash entry
                                |     UNTIL end of hashtable reached ENDREP .
                                |
                                |
      listobjectsofcurrentpa    |list objects of current packet in this chain :
                                |     WHILE yet another nt entry REP
                                |         permanent pointer := cdb int (nt link + wordlength) ;
                                |         put objects of this name
                                |     PER .
                                |
                                |
      putobjectsofthisname      |put objects of this name :
                                |     IF there is at least one object of this name in the current
                                |          packet
                                |       THEN into bulletin FI .
                                |
                                |
      thereisatleastoneobjec    |there is at least one object of this name in the current packet :
                                |     REP
                                |        IF permanent pointer >= begin of packet AND
                                |           permanent pointer < last packet entry
                                |           THEN LEAVE there is at least one object of this name
                                |                      in the current packet WITH TRUE FI ;
                                |       next procedure
                                |     UNTIL permanent pointer = 0 PER ;
                                |     FALSE .
                                |
                                |
      intobulletin              |into bulletin :
                                |     object name := cdb text (nt link + two word length) ;
                                |     object names into bulletin (BOOL PROC within packet)
                                |ENDPROC list packet ;
                                |


   withinpacket ............    |BOOL PROC within packet :
                                |     permanent pointer >= begin of packet AND
                                |     permanent pointer < last packet entry
                                |ENDPROC within packet ;
                                |


   objectnamesintobulleti ...   |PROC object names into bulletin (BOOL PROC link ok) :
                                |     scan (object name) ;
                                |     next symbol (object name, mode) ;
                                |     IF type definition THEN put type definition
                                |                        ELSE put object definitions
                                |     FI .
                                |
     typedefinition         |type definition :
                            |     mode = bold AND no params .
                            |
                            |
     noparams               |no params :
                            |     cdb int (permanent pointer + word length) >= permanent type .
                            |
                            |
     puttypedefinition      |put type definition :
                            |     put obj name (object name) ;
                            |     write bulletin line ("TYPE ") ;
                            |     writeline (1) .
                            |
                            |
     putobjectdefinitions   |put object definitions :
                            |     WHILE link ok REP
                            |         put specifications (object name) ;
                            |         next procedure
                            |     ENDREP
                            |ENDPROC object names into bulletin ;
                            |


  bulletin .................|PROC bulletin :
                            |     prep bulletin ;
                            |     packet link := first permanent entry ;
                            |     REP
                            |         list packet ;
                            |         write line (4) ;
                            |         next packet
                            |     UNTIL NOT found PER ;
                            |     show bulletin file
                            |ENDPROC bulletin ;
                            |


  putobjname ...............|PROC put obj name (TEXT CONST name) :
                            |     buffer := "      " ;
                            |     bulletin name := point line ;
                            |     change (bulletin name, 1, end of line or name, name) ;
                            |     buffer CAT bulletin name ;
                            |     indentation := LENGTH buffer + 1 .
                            |
                            |
     endoflineorname        |end of line or name :
                            |     min (LENGTH name, LENGTH bulletin name)
                            |ENDPROC put obj name ;
                            |


  packets ..................|PROC packets :
                            |     prep bulletin ;
                            |     packet link := first permanent entry ;
                            |     REP
                            |         object name := packet name ;
                            |         put obj name (object name) ;
                            |         write line ;
                            |         next packet
                            |     UNTIL NOT found PER ;
                            |     show bulletin file
                            |ENDPROC packets ;
                            |
                             |(**********************************************************************
                             |    ******)
                             |(*
                             |         *)
                             |(*                 11. ELAN Run-Interface
                             |    09.01.1986  *)
                             |(*
                             |         *)
                             |(*  Uebersetzen von ELAN-Programmen
                             |         *)
                             |(*  Bereitstellen der Ausgabeprozeduren fuer den ELAN-Compiler
                             |         *)
                             |(*
                             |         *)
                             |(**********************************************************************
                             |    ******)
                             |
                             |
                             |
                             |BOOL VAR list option     := FALSE ,
                             |         check option    :=  TRUE ,
                             |         warning option  := FALSE ,
                             |         listing enabled := FALSE ;
                             |
                             |FILE VAR listing file ;
                             |
                             |TEXT VAR listing file name := "" ;
                             |
                             |


run .........................|PROC run (TEXT CONST file name) :
                             |  enable stop ;
                             |  IF NOT exists (file name)
                             |    THEN errorstop ("""" + file name + """ gibt es nicht")
                             |  FI ;
                             |  last param (file name) ;
                             |  run elan (file name, no ins)
                             |END PROC run;
                             |


run .........................|PROC run :
                             |  run (last param)
                             |ENDPROC run ;
                             |


runagain ....................|PROC run again :
                             |  IF run again mod nr <> 0
                             |    THEN elan (run again mode, bulletin file, "", run again mod nr,
                             |               no ins, no lst, check option, no sermon)
                             |    ELSE errorstop ("'run again' nicht moeglich")
                             |  FI
                             |ENDPROC run again ;
                             |


insert ......................|PROC insert (TEXT CONST file name) :
                             |  enable stop ;
                             |  IF NOT exists (file name)
                             |    THEN errorstop ("""" + file name + """ gibt es nicht")
                             |  FI ;
                             |  last param (file name) ;
                             |  run elan (file name, ins)
                             |ENDPROC insert ;
                             |


insert ......................|PROC insert :
                             |  insert (last param)
                             |ENDPROC insert ;
                             |


runelan .....................|PROC run elan (TEXT CONST file name, BOOL CONST insert option) :
                             |  FILE VAR source := sequential file (modify, file name) ;
                             |  IF listing enabled
                             |    THEN open listing file
                             |  FI ;
                             |
                             |  disable stop ;
                             |  no do again ;
                             |  elan (compile file mode, source, "" , run again mod nr,
                             |        insert option, list option, check option, sermon) ;
                             |
                             |  IF anything noted AND command dialogue
                             |    THEN ignore halt during compiling ;
                             |         note edit (source) ;
                             |         last param (file name) ;
                             |         errorstop ("")
                             |  FI .
                             |
                             |
   ignorehaltduringcompil    |ignore halt during compiling :
                             |  IF is error
                             |    THEN put error ;
                             |         clear error ;
                             |         pause (5)
                             |  FI .
                             |
                             |
   openlistingfile           |open listing file :
                             |  listing file := sequential file (output, listing file name) ;
                             |  max line length (listing file, 130)
                             |
                             |ENDPROC run elan ;
                             |


outtext .....................|PROC out text (TEXT CONST text, INT CONST out type) :
                             |  INTERNAL 257 ;
                             |  IF online
                             |    THEN out (text)
                             |  FI ;
                             |  IF out type = error message OR (warning option AND out type =
                             |      warning message)
                             |    THEN note (text) ;
                             |  FI ;
                             |  IF listing enabled
                             |    THEN write (listing file, text)
                             |  FI
                             |ENDPROC out text ;
                             |
outline ..................|PROC out line (INT CONST out type) :
                          |  INTERNAL 258 ;
                          |  IF online
                          |    THEN out (""13""10"")
                          |  FI ;
                          |  IF out type = error message
                          |       OR (warning option AND out type = warning message)
                          |    THEN note line
                          |  ELIF listing enabled
                          |    THEN line (listing file)
                          |  FI
                          |ENDPROC out line ;
                          |

prot .....................|PROC prot (TEXT CONST file name) :
                          |  list option := TRUE ;
                          |  listing file name := file name ;
                          |  listing enabled := TRUE
                          |ENDPROC prot ;
                          |

protoff ..................|PROC prot off :
                          |  list option := FALSE ;
                          |  listing enabled := FALSE
                          |ENDPROC prot off ;
                          |

prot .....................|BOOL PROC prot :
                          |  list option
                          |ENDPROC prot ;
                          |

checkon ..................|PROC check on :
                          |  check option := TRUE
                          |ENDPROC check on ;
                          |

checkoff .................|PROC check off :
                          |  check option := FALSE
                          |ENDPROC check off ;
                          |

check ....................|BOOL PROC check :
                          |  check option
                          |ENDPROC check ;
                          |

warningson ...............|PROC warnings on :
                          |  warning option := TRUE
                          |ENDPROC warnings on ;
                          |
warningsoff ..............|PROC warnings off :
                          |  warning option := FALSE
                          |ENDPROC warnings off ;
                          |

warnings .................|BOOL PROC warnings :
                          |  warning option
                          |ENDPROC warnings ;
                          |
                          |ENDPACKET eumel coder part 1 ;
```

```
                            |(* ------------------- VERSION 2     06.03.86 -------------------
mathlib ********************|PACKET mathlib DEFINES sqrt, **, exp, ln, log2, log10, e, pi,
                            |                       sin, cos, tan, sind, cosd, tand,
                            |                       arctan, arctand, random, initializerandom :
                            |
                            |LET  pii    =  3.141592653589793238462,
                            |     pi2    =  1.570796326794896619231,
                            |     pi3    =  1.047197551196597746154,
                            |     pi6    =  0.523598775598298873077,
                            |     pi4    =  1.27323954473516268651,
                            |     ln2    =  0.693147180559945309417,
                            |     lg2    =  0.301029995663981195213,
                            |     ln10   =  2.302585092994045684018,
                            |     lge    =  0.434294481903251827651,
                            |     ei     =  2.718281828459045235360,
                            |     pi180  = 57.295779513082320876798,
                            |     sqrt3  =  1.732050807568877293527,
                            |     sqr3   =  0.577350269189625764509,
                            |     sqr3p2=   3.732050807568877293527,
                            |     sqr3m2=   0.267949192431122706473,
                            |     sqr2   =  0.707106781186547524400;
                            |
                            |REAL VAR rdg::0.4711;
                            |


pi .........................|REAL PROC pi:   pii     END PROC pi;


e ..........................|REAL PROC e :   ei      END PROC e;
                            |


ln .........................|REAL PROC  ln  ( REAL CONST x ):
                            |     log2(x) * ln2
                            |END PROC ln;
                            |


log10 ......................|REAL PROC log10( REAL CONST x ):
                            |     log2(x) * lg2
                            |END PROC log10;
                            |


log2 .......................|REAL PROC log2 ( REAL CONST z ):
                            | REAL VAR t, summe::0.0, x::z;
                            | IF   x=1.0  THEN 0.0
                            | ELIF x>0.0  THEN normal
                            |             ELSE errorstop("log2: " + text (x,20)); 0.0   FI.
                            |
                            |
   normal                   |normal:
                            |  IF   x >= 0.5   THEN  normalise downwards
                            |                  ELSE  normalise upwards      FI;
                            |  IF   x < sqr2   THEN  summe := summe - 0.75; t := trans8
                            |                  ELSE  summe := summe - 0.25; t := trans2  FI;
                            |  summe + reihenentwicklung.
                            |
   normalisedownwards    | normalise downwards:
                         |  WHILE  x >= 8.0 REP    x := 0.0625 * x;  summe:=summe+4.0  PER;
                         |  WHILE  x >= 1.0 REP    x :=    0.5 * x;  summe:=summe+1.0  PER.
                         |
                         |
   normaliseupwards      | normalise upwards:
                         |  WHILE x<=0.0625 REP    x :=   16.0 * x;  summe:=summe-4.0  PER;
                         |  WHILE x<= 0.5   REP    x :=    2.0 * x;  summe:=summe-1.0  PER.
                         |
                         |
   trans8                | trans8: (x - 0.5946035575013605)/(x + 0.5946035575013605).
                         |
   trans2                | trans2: (x - 0.8408964152537145)/(x + 0.8408964152537145).
                         |
                         |
   reihenentwicklung     | reihenentwicklung:  x := t * t;         t * 0.06405572387119384648
    +                    |       *
                         |  ((((((3.465*x+4.095)*x+5.005)*x+6.435)*x+9.009)*x+15.015)*x+45.045)
                         |END PROC log2;
                         |


sqrt ....................|REAL PROC sqrt ( REAL CONST z ):
                         |  REAL VAR y0, y1, x::z;
                         |  INT  VAR p :: decimal exponent(x) DIV 2;
                         |  IF   p <= -64  THEN 0.0
                         |  ELIF x < 0.0  THEN errorstop("sqrt: " + text (x,20)); 0.0
                         |                ELSE nontrivial    FI.
                         |
                         |
   nontrivial            | nontrivial:
                         |  set exp (decimal exponent (x) -p-p, x);
                         |  IF x<10.0 THEN x := 5.3176703 - 40.760905/( 8.408065 + x )
                         |            ELSE x := 16.81595  - 1288.973 /( 84.08065 + x )  FI;
                         |  y0 := x;
                         |  set exp (decimal exponent (x) + p, y0);
                         |  y1 := 0.5 * ( y0 + z/y0 );
                         |  y0 := 0.5 * ( y1 + z/y1 );
                         |  y1 := 0.5 * ( y0 + z/y0 );
                         |        0.5 * ( y1 + z/y1 )
                         |END PROC sqrt;
                         |


exp .....................|REAL PROC exp ( REAL CONST z ):
                         | REAL VAR x::z, a::1.0;  BOOL VAR negativ :: x<0.0;
                         | IF   negativ        THEN x := -x  FI;
                         | IF   x>292.42830676
                         |   THEN IF NOT negativ THEN errorstop ("REAL-Ueberlauf") FI ; 0.0
                         | ELIF x<=0.0001
                         |   THEN ( 0.5*z + 1.0 ) * z + 1.0
                         |   ELSE approx
                         | FI.
                         |
                         |
   approx                | approx:
                         | IF x > ln10
                         | THEN x := lge*x;
                         |      a := 1.0;
                         |      set exp (int(x), a);
                         |      x := frac(x)*ln10
                         | FI;
```

```
                        | IF x >= 2.0    THEN a := 7.389056098930650227230*a; x := x-2.0    FI;
                        | IF x >= 1.0    THEN a := 2.718281828459045235360*a; x := x-1.0    FI;
                        | IF x >= 0.5    THEN a := 1.648721270700128146848*a; x := x-0.5    FI;
                        | IF x >= 0.25   THEN a := 1.284025416687741484073*a; x := x-0.25   FI;
                        | IF x >= 0.125  THEN a := 1.133148453066826316829*a; x := x-0.125  FI;
                        | IF x >= 0.0625THEN a := 1.064494458917859429563*a; x := x-0.0625FI;
                        |
                        |      a:=a/50.4*(((((((0.01*x+0.07)*x+0.42)*x+2.1)*x+8.4)*x+25.2)*x+5
                        |      0.4)*x+50.4);
                        | IF negativ THEN 1.0/a ELSE a FI .
                        |
                        |ENDPROC exp ;
                        |


tan ....................|REAL PROC tan (REAL CONST x):
                        |  IF x < 0.0 THEN - tg( -x * pi4)
                        |              ELSE   tg(  x * pi4)  FI
                        |END PROC tan;
                        |


tand ...................|REAL PROC tand (REAL CONST x):
                        |  IF x < 0.0 THEN - tg( -x / 45.0)
                        |              ELSE   tg(  x / 45.0) FI
                        |END PROC tand;
                        |


tg .....................|REAL PROC tg (REAL CONST x ):
                        |  REAL VAR q::floor(x), s::x-q; INT VAR n;
                        |  q := q - floor(0.25*q) * 4.0 ;
                        |  IF q < 2.0
                        |  THEN IF q < 1.0
                        |       THEN n:=0;
                        |       ELSE n:=1; s := 1.0 - s FI
                        |  ELSE IF q < 3.0
                        |       THEN n:=2;
                        |       ELSE n:=3; s := 1.0 - s FI
                        |  FI;
                        |  q := s * s;
                        |  q := (((((((((-5.116186989653120e-11*q-5.608325022830701e-10)*q-
                        |
                        |       9.526170109403018e-9)*q-1.517906721393745e-7)*q-2.430939946375
                        |       515e-6)*q-
                        |
                        |       3.901461426385464e-5)*q-6.324811612385572e-4)*q-1.076606829172
                        |       646e-2)*q-
                        |  0.2617993877991508)*q+pi4);
                        |
                        |  SELECT n OF
                        |  CASE 0 : s/q
                        |  CASE 1 : q/s
                        |  CASE 2 : -q/s
                        |  OTHERWISE : -s/q ENDSELECT  .
                        |
                        |END PROC tg;
                        |
sin ......................|REAL PROC sin ( REAL CONST x ):
                          |  REAL VAR y, r, q;
                          |  IF x < 0.0 THEN y := -x; q := 4.0 ELSE y := x; q := 0.0 FI;
                          |  y := y * pi4;
                          |  r := floor(y);
                          |  sincos( q+r , y-r )
                          |END PROC sin;
                          |

sind .....................|REAL PROC sind ( REAL CONST x ):
                          |  REAL VAR y, r, q;
                          |  IF x < 0.0 THEN y := -x; q := 4.0 ELSE y := x; q := 0.0 FI;
                          |  y := y / 45.0;
                          |  r := floor(y);
                          |  sincos( q+r , y-r )
                          |END PROC sind;
                          |

cos ......................|REAL PROC cos ( REAL CONST x ):
                          |  REAL VAR y, q;
                          |  IF x < 0.0 THEN y := -x ELSE y := x FI;
                          |  y := y * pi4;
                          |  q := floor(y);
                          |  sincos( q+2.0, y-q )
                          |END PROC cos;
                          |

cosd .....................|REAL PROC cosd ( REAL CONST x ):
                          |  REAL VAR y, q;
                          |  IF x < 0.0 THEN y := -x ELSE y := x FI;
                          |  y := y / 45.0;
                          |  q := floor(y);
                          |  sincos( q+2.0, y-q )
                          |END PROC cosd;
                          |

sincos ...................|REAL PROC sincos ( REAL CONST q, y ):
                          | REAL VAR r :: q - floor( 0.125*q + 0.1 ) * 8.0;
                          | IF r >= 4.0 THEN IF r >= 6.0 THEN IF r >= 7.0 THEN - sin
                          |      approx(1.0-y)
                          |                                                ELSE - cos approx(y)
                          |                                                     FI
                          |                                ELSE IF r >= 5.0 THEN - cos
                          |                                     approx(1.0-y)
                          |                                                ELSE - sin approx(y)
                          |                                                     FI FI
                          |              ELSE IF r >= 2.0 THEN IF r >= 3.0 THEN   sin
                          |                   approx(1.0-y)
                          |                                                ELSE   cos approx(y)
                          |                                                     FI
                          |                                ELSE IF r >= 1.0 THEN   cos
                          |                                     approx(1.0-y)
                          |                                                ELSE   sin approx(y
                          |                                                     FI FI FI
                          |END PROC sincos;
                          |
sinapprox ...............|REAL PROC sin approx ( REAL CONST x ):
                         | REAL VAR z::x*x;
                         |
                         |     x*((((((0.6877101540593035e-11*z-0.1757149296873372e-8)*z+0.313
                         |     3616216672568
                         |
                         |     e-6)*z-0.3657620415845891e-4)*z+0.2490394570188737e-2)*z-0.8074
                         |     55121882e-1)*
                         | z+0.7853981633974483)
                         |END PROC sin approx;
                         |


cosapprox ...............|REAL PROC cos approx ( REAL CONST x ):
                         | REAL VAR z::x*x;
                         |
                         |     ((((((-0.3857761864560276e-12*z+0.115004970178141e-9)*z-0.24611
                         |     3638267419e-7
                         |
                         |     )*z+0.3590860445885748e-5)*z-0.3259918869266875e-3)*z+0.1585434
                         |     424381541e-1)
                         | *z-0.3084251375340425)*z+1.0
                         |END PROC cos approx;
                         |


arctan ..................|REAL PROC arctan ( REAL CONST y ):
                         | REAL VAR f, z, x; BOOL VAR neg :: y < 0.0;
                         | IF   neg   THEN x := -y ELSE x := y FI;
                         | IF   x>1.0 THEN f := a ELSE f := -b; neg := NOT neg FI;
                         | z := x * x;
                         | x := x/(((((((0.0107090276046822*z-0.01647757182108040)*z
                         |
                         |          +0.02177846332482151)*z-0.03019339673273880)*z+0.046560835
                         |          61183398)*z
                         |     -0.08888888888888888)*z+0.3333333333333333)*z+1.0);
                         | IF   neg   THEN  x - f  ELSE  f - x  FI.
                         |
                         |
    a                    | a:IF x>sqr3p2 THEN x := 1.0/x; pi2 ELSE x :=
                         |     4.0/(sqrt3+x+x+x)-sqr3; pi3 FI.
                         |
    b                    | b:IF x<sqr3m2 THEN   0.0           ELSE x := sqrt3 - 4.0/(sqrt3+x);
                         |       pi6 FI
                         |END PROC arctan;
                         |


arctand .................|REAL PROC arctand ( REAL CONST x ):
                         |  arctan(x) * pi180
                         |END PROC arctand;
                         |


** ......................|REAL OP ** ( REAL CONST b, e ):
                         | IF b=0.0
                         |   THEN IF e=0.0 THEN 1.0 ELSE 0.0 FI
                         | ELIF b < 0.0
                         |   THEN errorstop("("+text(b,20)+") ** "+text(e)); (-b) ** e
                         | ELSE   exp( e * log2( b ) * ln2 )
                         | FI
                         |END OP **;
```

```
                            |

** ........................|REAL OP ** ( REAL CONST a, INT CONST b ) :
                            |
                            |  REAL VAR p := 1.0 ,
                            |           r := a ;
                            |  INT VAR  n := ABS b ,
                            |           m ;
                            |  IF (a = 0.0 OR a = -0.0)
                            |    THEN IF b = 0
                            |           THEN 1.0
                            |           ELSE 0.0
                            |         FI
                            |    ELSE WHILE n>0 REP
                            |           m := n DIV 2 ;
                            |           IF m + m = n
                            |             THEN n := m ;
                            |                  r := r*r
                            |             ELSE n DECR 1 ;
                            |                  p := p*r
                            |           FI
                            |         END REP ;
                            |         IF b>0
                            |           THEN p
                            |           ELSE 1.0 / p
                            |         FI
                            |  FI .
                            |
                            |END OP ** ;
                            |

random ....................|REAL PROC random:
                            |
  +                         |      rdg:=rdg+pii;rdg:=rdg*rdg;rdg:=rdg*rdg;rdg:=rdg*rdg;rdg:=frac(
  +                         |      rdg);rdg
                            |END PROC random;
                            |

initializerandom .........|PROC initializerandom ( REAL CONST z ):
                            |  rdg := frac(z)
                            |END PROC initializerandom;
                            |
                            |END PACKET mathlib;
                            |(* -------------------- VERSION 2     05.05.86 -------------------- *)
commandhandler ************|PACKET command handler DEFINES                  (* Autor: J.Liedtke *)
                            |
                            |      get command ,
                            |      analyze command ,
                            |      do command ,
                            |      command error ,
                            |      cover tracks :
                            |
                            |
                            |LET cr lf         = ""4""13""10"" ,
                            |   esc k          = ""27"k" ,
                            |   command pre  = ""4""13"       " ,
                            |   command post = ""13""10"      " ,
                            |
                            |   max command length = 2010 ,
                            |
                            |   tag type = 1 ,
                            |   texttype = 4 ,
                            |   eof type = 7 ;
                            |
                            |
                            |TEXT VAR command handlers own command line := "" ,
                            |        previous command line := "" ,
                            |        symbol ,
                            |        procedure ,
                            |        pattern ,
                            |        error note := "" ;
                            |
                            |INT VAR  symbol type ;
                            |
                            |


getcommand ..............|PROC get command (TEXT CONST command text) :
                            |
                            |  get command (command text, command handlers own command line)
                            |
                            |ENDPROC get command ;
                            |


getcommand ..............|PROC get command (TEXT CONST command text, TEXT VAR command line) :
                            |
                            |  set line nr (0) ;
                            |  error protocoll ;
                            |  get command from console .
                            |
                            |
    errorprotocoll          |error protocoll :
                            |  IF is error
                            |    THEN put error ;
                            |         clear error
                            |    ELSE command line := "" ;
                            |  FI .
                            |
                            |
    getcommandfromconsole   |get command from console :
                            |  normalize cursor ;
                            |  REP
                            |    out (command pre) ;
                            |    out (command text) ;
```

```
                          |     out (command post) ;
                          |     editget command
                          |   UNTIL command line <> "" PER ;
                          |   param position (LENGTH command line) ;
                          |   out (command post) .
                          |
                          |
    editgetcommand        |editget command :
                          |   TEXT VAR exit char ;
                          |   REP
                          |     get cursor (x, y) ;
                          |     editget (command line, max command length, x size - x,
                          |              "", "k", exit char) ;
                          |     ignore halt errors during editget ;
                          |     break quiet if command line is too long ;
                          |     IF exit char = esc k
                          |       THEN cursor to begin of command input ;
                          |            command line := previous command line
                          |     ELIF LENGTH command line > 1
                          |       THEN previous command line := command line ;
                          |            LEAVE editget command
                          |     ELSE   LEAVE editget command
                          |     FI
                          |   PER .
                          |
                          |
    normalizecursor       |normalize cursor :
                          |   INT VAR x, y;
                          |   out (crlf) ;
                          |   get cursor (x, y) ;
                          |   cursor (x, y) .
                          |
                          |
    ignorehalterrorsduring|ignore halt errors during editget :
                          |   IF is error
                          |     THEN clear error
                          |   FI .
                          |
                          |
    breakquietifcommandlin|break quiet if command line is too long :
                          |   IF command line is too long
                          |     THEN command line := "break (quiet)"
                          |   FI .
                          |
                          |
    commandlineistoolong  |command line is too long :
                          |   LENGTH command line = max command length .
                          |
                          |
    cursortobeginofcommand|cursor to begin of command input :
                          |   out (command pre) .
                          |
                          |ENDPROC get command ;
                          |
                          |


analyzecommand ...........|PROC analyze command ( TEXT CONST command list,
                          |                       INT CONST permitted type,
                          |                       INT VAR command index, number of params,
                          |                       TEXT VAR param 1, param 2) :
                          |
                          |  analyze command (command list, command handlers own command line,
                          |                   permitted type, command index,
                          |                   number of params, param 1, param 2)
                          |
                          |ENDPROC analyze command ;
                          |

analyzecommand ...........|PROC analyze command ( TEXT CONST command list, command line,
                          |                       INT CONST permitted type,
                          |                       INT VAR command index, number of params,
                          |                       TEXT VAR param 1, param 2) :
                          |
                          |  error note := "" ;
                          |  scan (command line) ;
                          |  next symbol ;
                          |  IF symbol type <> tag type AND symbol <> "?"
                          |    THEN error ("Name ungueltig") ;
                          |         impossible command
                          |  ELIF pos (command list, symbol) > 0
                          |    THEN procedure name ;
                          |         parameter list pack option ;
                          |         nothing else in command line ;
                          |         decode command
                          |    ELSE impossible command
                          |  FI .
                          |
                          |
   procedurename          |procedure name :
                          |  procedure := symbol ;
                          |  next symbol .
                          |
                          |
   parameterlistpackoptio |parameter list pack option :
                          |  number of params := 0 ;
                          |  param 1 := "" ;
                          |  param 2 := "" ;
                          |  IF symbol = "("
                          |    THEN next symbol ;
                          |         parameter list ;
                          |         IF symbol <> ")" AND error note = ""
                          |           THEN error (") fehlt")
                          |         FI
                          |  ELIF symbol type <> eof type
                          |    THEN error ("( fehlt")
                          |  FI .
                          |
                          |
   parameterlist          |parameter list :
                          |  parameter (param 1, number of params, permitted type) ;
                          |  IF symbol = ","
                          |    THEN next symbol ;
                          |         parameter (param 2, number of params, permitted type) ;
                          |  FI .
                          |
                          |
   nothingelseincommandli |nothing else in command line :
                          |  next symbol ;
                          |  IF symbol <> ""
                          |    THEN error ("Kommando zu schwierig")
                          |  FI .
                          |
     decodecommand          |decode command :
                            |  command index := index (command list, procedure, number of params(
                            |        .
                            |
                            |
     impossiblecommand      |impossible command :
                            |  command index := 0 .
                            |
                            |ENDPROC analyze command ;
                            |


parameter ..................|PROC parameter (TEXT VAR param, INT VAR number of params,
                            |                INT CONST permitted type) :
                            |
                            |  IF symbol type = text type OR symbol type = permitted type
                            |    THEN param := symbol ;
                            |         number of params INCR 1 ;
                            |         next symbol
                            |    ELSE error ("Parameter ist kein TEXT ("" fehlt)")
                            |  FI
                            |
                            |ENDPROC parameter ;
                            |


index ......................|INT PROC index (TEXT CONST list, procedure, INT CONST params) :
                            |
                            |  pattern := procedure ;
                            |  pattern CAT ":" ;
                            |  IF procedure name found
                            |    THEN get colon pos ;
                            |         get dot pos ;
                            |         get end pos ;
                            |         get command index ;
                            |         get param index ;
                            |         IF param index >= 0
                            |           THEN command index + param index
                            |           ELSE - command index
                            |         FI
                            |    ELSE 0
                            |  FI .
                            |
                            |
     procedurenamefound     |procedure name found :
                            |  INT VAR index pos := pos (list, pattern) ;
                            |  WHILE index pos > 0 REP
                            |    IF index pos = 1  COR  (list SUB index pos - 1) <= "9"
                            |      THEN LEAVE procedure name found WITH TRUE
                            |    FI ;
                            |    index pos := pos (list, pattern, index pos + 1)
                            |  PER ;
                            |  FALSE .
                            |
                            |
     getparamindex          |get param index :
                            |  INT CONST param index :=
                            |          pos (list, text (params), dot pos, end pos) - dot pos -
                            |              1 .
                            |
    getcommandindex          |get command index :
                             |  INT CONST command index :=
                             |            int ( subtext (list, colon pos + 1, dot pos - 1) ) .
                             |
                             |
    getcolonpos              |get colon pos :
                             |  INT CONST colon pos := pos (list, ":", index pos) .
                             |
                             |
    getdotpos                |get dot pos :
                             |  INT CONST dot pos := pos (list, ".", index pos) .
                             |
                             |
    getendpos                |get end pos :
                             |  INT CONST end pos := dot pos + 4 .
                             |
                             |ENDPROC index ;
                             |


docommand ...................|PROC do command :
                             |
                             |  do (command handlers own command line)
                             |
                             |ENDPROC do command ;
                             |


error .......................|PROC error (TEXT CONST message) :
                             |
                             |  error note := message ;
                             |  scan ("") ;
                             |  procedure := "-"
                             |
                             |ENDPROC error ;
                             |


commanderror ................|PROC command error :
                             |
                             |  disable stop ;
                             |  IF error note <> ""
                             |    THEN errorstop (error note) ;
                             |         error note := ""
                             |  FI ;
                             |  enable stop
                             |
                             |ENDPROC command error ;
                             |
                             |


nextsymbol ..................|PROC next symbol :
                             |
                             |  next symbol (symbol, symbol type)
                             |
                             |ENDPROC next symbol ;
                             |
                             |
covertracks ..............|PROC cover tracks :
                          |
                          |  cover tracks (command handlers own command line) ;
                          |  cover tracks (previous command line) ;
                          |  erase buffers of compiler and do packet .
                          |
                          |
   erasebuffersofcompiler |erase buffers of compiler and do packet :
                          |  do (command handlers own command line) .
                          |
                          |ENDPROC cover tracks ;
                          |


covertracks ..............|PROC cover tracks (TEXT VAR secret) :
                          |
                          |  INT VAR i ;
                          |  FOR i FROM 1 UPTO LENGTH secret REP
                          |    replace (secret, i, " ")
                          |  PER ;
                          |  WHILE LENGTH secret < 13 REP
                          |    secret CAT " "
                          |  PER
                          |
                          |ENDPROC cover tracks ;
                          |
                          |ENDPACKET command handler ;
```

```
advertising ***************|PACKET advertising DEFINES eumel must advertise : (* Autor:
                           |     J.Liedtke *)
                           |                                                   (* Stand: 08.03.85
                           |      *)
                           |


eumelmustadvertise .......|PROC eumel must advertise :
                           |
                           |  IF online AND channel <= 15
                           |    THEN out (""1""4"") ;
                           |         cursor (22,5) ;
                           |         out ("E U M E L    Version 1.8.0/S     "13""10""10""10"") ;
                           |  FI .
                           |
                           |ENDPROC eumel must advertise ;
                           |
                           |ENDPACKET advertising ;
```

```
                                 |
taskssingle ***************|PACKET tasks single DEFINES                          (* Autor: J.Liedtke *)
                                 |                                                     (* Stand: 01.06.84 *)
                                 |    TASK ,
                                 |    := ,
                                 |    = ,
                                 |    niltask ,
                                 |    is niltask ,
                                 |    myself ,
                                 |    archive ,
                                 |    father ,
                                 |
                                 |    dataspaces ,
                                 |    pcb ,
                                 |    status ,
                                 |    channel ,
                                 |    clock ,
                                 |    storage ,
                                 |    continue :
                                 |
                                 |
                                 |LET nil = 0 ,
                                 |
                                 |    hex ff   = 255 ,
                                 |    hex 7f00 = 32512 ,
                                 |
                                 |    channel field = 4 ,
                                 |    myself no field = 9 ,
                                 |    myself version field = 10 ,
                                 |
                                 |    lowest ds number      = 4 ,
                                 |    highest ds number     = 255 ,
                                 |
                                 |    max channel = 32 ;
                                 |
                                 |
                                 |
                                 |TYPE TASK = STRUCT (INT no, version) ;
                                 |
                                 |TASK CONST niltask := TASK: (0,0) ,
                                 |           archive := TASK: (4711,4711) ,
                                 |           father  := archive ;
                                 |


myself ..................|TASK PROC myself :
                                 |
                                 |  TASK: (pcb (myself no field), pcb (myself version field))
                                 |
                                 |ENDPROC myself ;
                                 |
                                 |


:= .......................|OP := (TASK VAR dest, TASK CONST source) :
                                 |
                                 |  CONCR (dest) := CONCR (source)
                                 |
                                 |ENDOP := ;
                                 |
= .........................|BOOL OP = (TASK CONST left, right) :
                           |
                           |  left.no = right.no  AND  left.version = right.version
                           |
                           |ENDOP = ;
                           |

isniltask .................|BOOL PROC is niltask (TASK CONST t) :
                           |
                           |  t.no = 0
                           |
                           |ENDPROC is niltask ;
                           |

pcb .......................|INT PROC pcb (TASK CONST id, INT CONST field) :
                           |
                           |  EXTERNAL 104
                           |
                           |ENDPROC pcb ;
                           |

status ....................|INT PROC status (TASK CONST id) :
                           |
                           |  EXTERNAL 107
                           |
                           |ENDPROC status ;
                           |

channel ...................|INT PROC channel (TASK CONST id) :
                           |
                           |  pcb (id, channel field)
                           |
                           |ENDPROC channel ;
                           |

clock .....................|REAL PROC clock (TASK CONST id) :
                           |
                           |  EXTERNAL 106
                           |
                           |ENDPROC clock ;
                           |

storage ...................|INT PROC storage (TASK CONST id) :
                           |
                           |  INT VAR ds number, storage sum := 0, ds size;
                           |  FOR ds number FROM lowest ds number UPTO highest ds number REP
                           |    ds size := pages (ds number, id) ;
                           |    IF ds size >= 0
                           |      THEN storage sum INCR ((ds size + 1) DIV 2)
                           |    FI
                           |  PER ;
                           |  storage sum
                           |
                           |ENDPROC storage ;
                           |
pages ....................|INT PROC pages (INT CONST ds number, TASK CONST id) :
                          |
                          |  EXTERNAL 88
                          |
                          |ENDPROC pages ;
                          |

continue .................|PROC continue (INT CONST channel no) :
                          |
                          |  IF channel no > 0 AND channel no <= max channel
                          |    THEN write pcb (myself, channel field, channel no)
                          |    ELSE errorstop ("ungueltige Kanalnummer")
                          |  FI
                          |
                          |ENDPROC continue ;
                          |

dataspaces ...............|INT PROC dataspaces :
                          |
                          |  INT VAR ds number, spaces := 0 ;
                          |  FOR ds number FROM lowest ds number UPTO highest ds number REP
                          |    IF pages (ds number, pcb (myself no field)) >= 0
                          |      THEN spaces INCR 1
                          |    FI
                          |  PER ;
                          |  spaces
                          |
                          |ENDPROC dataspaces ;
                          |

pages ....................|INT PROC pages (INT CONST ds number, INT CONST task no) :
                          |  EXTERNAL 88
                          |ENDPROC pages ;
                          |

writepcb .................|PROC write pcb (TASK CONST task, INT CONST field, value) :
                          |  EXTERNAL  105
                          |ENDPROC write pcb ;
                          |
                          |ENDPACKET tasks single ;
```

```
fontstore ****************|PACKET font store                                  (* Autor : Rudolf
                          |     Ruland *)
                          |                                                   (* Stand :
                          |     18.02.86      *)
                          |        DEFINES font table,
                          |                list font tables,
                          |                list fonts,
                          |
                          |                x step conversion,
                          |                y step conversion,
                          |                on string,
                          |                off string,
                          |
                          |                font,
                          |                font exists,
                          |                next larger font exists,
                          |                next smaller font exists,
                          |                font lead,
                          |                font height,
                          |                font depth,
                          |                indentation pitch,
                          |                char pitch,
                          |                extended char pitch,
                          |                replacement,
                          |                extended replacement,
                          |                font string,
                          |                y offsets,
                          |                bold offset,
                          |                get font,
                          |                get replacements :
                          |
                          |
                          |LET underline        =    1,
                          |    bold             =    2,
                          |    italics          =    4,
                          |    reverse          =    8,
                          |
                          |    first font       =    1,
                          |    max fonts        =   50,
                          |    max extensions   =  120,
                          |    font table type  = 3009,
                          |
                          |    FONTTABLE = STRUCT (
                          |
                          |        THESAURUS font names,
                          |
                          |        TEXT replacements, font name links,
                          |             extension chars, extension indexes,
                          |
                          |        ROW 4 TEXT on strings, off strings,
                          |
                          |        REAL x unit, y unit,
                          |
                          |        ROW 256 INT replacements table,
                          |
                          |        INT last font, last extension
                          |
                          |        ROW max fonts STRUCT (
                          |            TEXT font string, font name indexes, replacements,
                          |                 extension chars, extension indexes, y offsets,
                          |            ROW 256 INT pitch table, replacements table,
                          |            INT  indentation pitch, font lead, font height, font
                          |                 depth,
                              |         next larger font, next smaller font, bold offset )
                              |            fonts ,
                              |
                              |    ROW max extensions STRUCT (
                              |      TEXT replacements,
                              |      ROW 256 INT pitch table, replacements table,
                              |      INT std pitch                                 )
                              |         extensions ,
                              |
                              |             );
                              |
                              |INT VAR font nr, list index, last font,
                              |    link nr, font store replacements length;
                              |
                              |TEXT VAR fo table := "", old font table, font name links, buffer;
                              |
                              |THESAURUS VAR font tables, font names;
                              |
                              |INITFLAG VAR in this task := FALSE,
                              |             init font ds := FALSE,
                              |             init ds      := FALSE;
                              |
                              |BOUND FONTTABLE VAR font store;
                              |
                              |
                              |DATASPACE VAR font ds, ds;
                              |
                              |(*****************************************************************)
                              |


fonttable ....................|PROC font table (TEXT CONST new font table) :
                              |
                              | disable stop;
                              | get font table (new font table);
                              | in this task := NOT (font table = "" OR type (font ds) <> font
                              |    table type);
                              |
                              |END PROC font table;
                              |
                              |


getfonttable .................|PROC get font table (TEXT CONST new font table) :
                              |
                              | enable stop;
                              | buffer := new font table;
                              | change all (buffer, " ", "");
                              | IF   exists (buffer) CAND type (old (buffer)) = font table type
                              |      THEN get font table from own task
                              |      ELSE errorstop ("Fonttabelle """ + buffer + """ gibt es
                              |         nicht")
                              | FI;
                              |
                              | . get font table from own task :
                              |     IF NOT initialized (init ds) THEN ds := nilspace FI;
                              |     forget (ds); ds := old (buffer);
                              |     new font store;
                              |
                              |     . new font store :
                              |         disable stop;
                              |         IF NOT initialized (init font ds) THEN font ds := nilspace
                              |            FI;
```

```
                          |          forget (font ds);
                          |          font ds := ds;
                          |          forget (ds);
                          |          font store      := font ds;
                          |          fo table        := buffer;
                          |          font names      := font store. font names;
                          |          font name links := font store. font name links;
                          |          last font       := font store. last font;
                          |          font store replacements length := LENGTH font store.
  +                       |               replacements;
                          |
                          |END PROC get font table;
                          |
                          |


fonttable ................|TEXT PROC font table :
                          |
                          |  fo table
                          |
                          |END PROC  font table;
                          |
                          |


listfonttables ...........|PROC list font tables :
                          |
                          |  enable stop;
                          |  font tables := empty thesaurus;
                          |  font tables in own task;
                          |  note font tables;
                          |  note edit;
                          |
                          |  . font tables in own task :
                          |      list index := 0;
                          |      REP get (all, buffer, list index);
                          |          IF buffer = "" THEN LEAVE font tables in own task FI;
                          |          IF type (old (buffer)) = font table type
                          |                  AND NOT (font tables CONTAINS buffer)
                          |            THEN insert (font tables, buffer) FI;
                          |      PER;
                          |
                          |  . note font tables :
                          |      list index := 0;
                          |      REP get (font tables, buffer, list index);
                          |          IF buffer = ""
                          |             THEN LEAVE note font tables;
                          |             ELSE note (buffer); note line;
                          |          FI;
                          |      PER;
                          |
                          |END PROC list font tables;
                          |
                          |


listfonts ................|PROC list fonts (TEXT CONST name):
                          |
                          |  initialize if necessary;
                          |  disable stop;
                          |  old font table := font table;
                          |  font table (name);
```

```
                         |  list fonts;
                         |  font table (old font table);
                         |
                         |END PROC list fonts;
                         |
                         |

listfonts ...............|PROC list fonts :
                         |
                         |  enable stop;
                         |  initialize if necessary;
                         |  note font table;
                         |  FOR font nr FROM first font UPTO last font REP note font PER;
                         |  note edit;
                         |
                         |. note font table :
                         |    note ("FONTTABELLE  :  """); note (font table);
                         |         note (""";"); noteline;
                         |    note ("  x einheit  =  ");   note (text (font store. x unit));
                         |         note (";"); noteline;
                         |    note ("  y einheit  =  ");   note (text (font store. y unit));
                         |         note (";"); noteline;
                         |
                         |. note font :
                         |    cout (font nr);
                         |    noteline;
                         |    note ("  FONT  :  ");                note font names;
                         |                   note (";"); noteline;
                         |    note ("    einrueckbreite   =  ");   note (text(font.
                         |         indentation pitch));  note (";"); noteline;
                         |    note ("    durchschuss      =  ");   note (text(font. font
                         |         lead));           note (";"); noteline;
                         |    note ("    fonthoehe        =  ");   note (text(font. font
                         |         height));         note (";"); noteline;
                         |    note ("    fonttiefe        =  ");   note (text(font. font
                         |         depth));          note (";"); noteline;
                         |    note ("    groesserer font  =  """); note (next larger);
                         |                 note (""";"); noteline;
                         |    note ("    kleinerer font   =  """); note (next smaller);
                         |                 note (""";"); noteline;
                         |
                         |    . font          : font store. fonts (font nr)
                         |    . next larger   : name (font store. font names, font. next larger
                         |         font)
                         |    . next smaller : name (font store. font names, font. next
                         |         smaller font)
                         |
                         |    . note font names :
                         |        INT VAR index;
                         |        note ("""");
                         |        note (name (font names, font. font name indexes ISUB 1));
                         |        note ("""");
                         |        FOR index FROM 2 UPTO LENGTH font. font name indexes DIV 2
                         |        REP note (",  """);
                         |            note (name (font names, font. font name indexes ISUB
                         |                 index));
                         |            note ("""");
                         |        PER;
                         |
                         |END PROC list fonts;
                         |
                           |

xstepconversion ..........|INT PROC x step conversion (REAL CONST cm) :
                          |
                          |  initialize if necessary;
                          |  IF cm >= 0.0
                          |     THEN int (cm * font store. x unit + 0.5 )
                          |     ELSE int (cm * font store. x unit - 0.5 )
                          |  FI
                          |
                          |END PROC x step conversion;
                          |
                          |

xstepconversion ..........|REAL PROC x step conversion (INT CONST steps) :
                          |
                          |  initialize if necessary;
                          |  real (steps) / font store. x unit
                          |
                          |END PROC x step conversion;
                          |
                          |

ystepconversion ..........|INT PROC y step conversion (REAL CONST cm) :
                          |
                          |  initialize if necessary;
                          |  IF cm >= 0.0
                          |     THEN int (cm * font store. y unit + 0.5 )
                          |     ELSE int (cm * font store. y unit - 0.5 )
                          |  FI
                          |
                          |END PROC y step conversion;
                          |
                          |

ystepconversion ..........|REAL PROC y step conversion (INT CONST steps) :
                          |
                          |  initialize if necessary;
                          |  real (steps) / font store. y unit
                          |
                          |END PROC y step conversion;
                          |
                          |

onstring .................|TEXT PROC on string (INT CONST modification) :
                          |
                          |  initialize if necessary;
                          |  SELECT modification OF
                          |    CASE underline : font store. on strings (1)
                          |    CASE bold      : font store. on strings (2)
                          |    CASE italics   : font store. on strings (3)
                          |    CASE reverse   : font store. on strings (4)
                          |         OTHERWISE : errorstop ("unzulaessige Modifikation"); ""
                          |  END SELECT
                          |
                          |END PROC on string;
                          |
                           |

offstring ................|TEXT PROC  off string (INT CONST modification) :
                          |
                          |  initialize if necessary;
                          |  SELECT modification OF
                          |    CASE underline : font store.  off strings (1)
                          |    CASE bold      : font store.  off strings (2)
                          |    CASE italics   : font store.  off strings (3)
                          |    CASE reverse   : font store.  off strings (4)
                          |         OTHERWISE : errorstop ("unzulaessige Modifikation"); ""
                          |  END SELECT
                          |
                          |END PROC  off string;
                          |
                          |

font .....................|INT PROC font (TEXT CONST font name) :
                          |
                          |  initialize if necessary;
                          |  buffer := font name;
                          |  change all (buffer, " ", "");
                          |  INT CONST link nr := link (font names, buffer)
                          |  IF link nr <> 0
                          |     THEN font name links ISUB link nr
                          |     ELSE 0
                          |  FI
                          |
                          |END PROC font;
                          |
                          |

font .....................|TEXT PROC font (INT CONST font number) :
                          |
                          |  initialize if necessary;
                          |  IF font number >= first font AND font number <= last font
                          |     THEN name (font names, fonts. font name indexes ISUB 1)
                          |     ELSE ""
                          |  FI
                          |
                          |  . fonts : font store. fonts (font number)
                          |
                          |END PROC font;
                          |
                          |

fontexists ...............|BOOL PROC font exists (TEXT CONST font name) :
                          |
                          |  font (font name) <> 0
                          |
                          |END PROC font exists;
                          |
                          |

nextlargerfontexists .....|BOOL PROC next larger font exists(INT CONST font number,
                          |                                  INT VAR   next larger font) :
                          |
                                 |  initialize if necessary;
                                 |  IF font number >= first font AND font number <= last font
                                 |     THEN next larger font := fonts. next larger font;
                                 |          IF next larger font <> 0
                                 |             THEN next larger font := font name links ISUB next
                                 |                  larger font;
                                 |                  next larger font <> 0
                                 |             ELSE FALSE
                                 |          FI
                                 |     ELSE errorstop ("Font " + text (font number) + " gibt es
                                 |          nicht");
                                 |          FALSE
                                 |  FI
                                 |
                                 |  . fonts : font store. fonts (font number)
                                 |
                                 |END PROC next larger font exists;
                                 |
                                 |


nextsmallerfontexists ....|BOOL PROC next smaller font exists (INT CONST font number,
                                 |                                     INT VAR   next smaller font) :
                                 |
                                 |  initialize if necessary;
                                 |  IF font number >= first font AND font number <= last font
                                 |     THEN next smaller font := fonts. next smaller font;
                                 |          IF next smaller font <> 0
                                 |             THEN next smaller font := font name links ISUB next
                                 |                  smaller font;
                                 |                  next smaller font <> 0
                                 |             ELSE FALSE
                                 |          FI
                                 |     ELSE errorstop ("Font " + text (font number) + " gibt es
                                 |          nicht");
                                 |          FALSE
                                 |  FI
                                 |
                                 |  . fonts : font store. fonts (font number)
                                 |
                                 |END PROC next smaller font exists;
                                 |
                                 |


fontlead ................|INT PROC font lead (INT CONST font number) :
                                 |
                                 |  initialize if necessary;
                                 |  IF font number >= first font AND font number <= last font
                                 |     THEN fonts. font lead
                                 |     ELSE errorstop ("Font " + text (font number) + " gibt es
                                 |          nicht"); 0
                                 |  FI
                                 |
                                 |  . fonts : font store. fonts (font number)
                                 |
                                 |END PROC font lead;
                                 |
                                 |
fontheight ..............|INT PROC font height (INT CONST font number) :
                         |
                         | initialize if necessary;
                         | IF font number >= first font AND font number <= last font
                         |    THEN fonts. font height
                         |    ELSE errorstop ("Font " + text (font number) + " gibt es
                         |         nicht"); 0
                         | FI
                         |
                         | . fonts : font store. fonts (font number)
                         |
                         |END PROC font height;
                         |
                         |


fontdepth ...............|INT PROC font depth (INT CONST font number) :
                         |
                         | initialize if necessary;
                         | IF font number >= first font AND font number <= last font
                         |    THEN fonts. font depth
                         |    ELSE errorstop ("Font " + text (font number) + " gibt es
                         |         nicht"); 0
                         | FI
                         |
                         | . fonts : font store. fonts (font number)
                         |
                         |END PROC font depth;
                         |
                         |


indentationpitch ........|INT PROC indentation pitch (INT CONST font number) :
                         |
                         | initialize if necessary;
                         | IF font number >= first font AND font number <= last font
                         |    THEN fonts. indentation pitch
                         |    ELSE errorstop ("Font " + text (font number) + " gibt es
                         |         nicht"); 0
                         | FI
                         |
                         | . fonts : font store. fonts (font number)
                         |
                         |END PROC indentation pitch;
                         |
                         |


charpitch ...............|INT PROC char pitch (INT  CONST font number,
                         |                      TEXT CONST char ) :
                         |
                         | initialize if necessary;
                         | IF font number >= first font AND font number <= last font
                         |    THEN INT CONST pitch := font. pitch table (code (char SUB 1) +
                         |         1);
                         |         IF   pitch = maxint
                         |              THEN extended char pitch (font number, char SUB 1,
                         |                   char SUB 2)
                         |         ELIF pitch < 0
                         |              THEN pitch XOR (-maxint-1)
                         |              ELSE pitch
                         |         FI
```

```
                              |     ELSE errorstop ("Font " + text (font number) + " gibt es
                              |          nicht"); 0
                              |  FI
                              |
                              |  . font :      font store. fonts (font number)
                              |
                              |END PROC char pitch;
                              |
                              |


extendedcharpitch ........|INT PROC extended char pitch (INT  CONST font number,
                              |                               TEXT CONST esc char, char) :
                              |
                              |  initialize if necessary;
                              |  IF font number >= first font AND font number <= last font
                              |     THEN extension. pitch table (code (char) + 1)
                              |     ELSE errorstop ("Font " + text (font number) + " gibt es
                              |          nicht"); 0
                              |  FI
                              |
                              |  . font      : font store. fonts (font number)
                              |
                              |  . extension : font store. extensions (extension number)
                              |
                              |  . extension number :
                              |      INT CONST index := pos (font. extension chars, esc char);
                              |      IF index = 0
                              |         THEN errorstop ("""" + esc char + char + """ hat keine
                              |              Erweiterung") FI;
                              |      font. extension indexes ISUB index
                              |
                              |END PROC extended char pitch;
                              |
                              |


replacement .............|TEXT PROC replacement (INT  CONST font number,
                              |                       TEXT CONST char ) :
                              |
                              |  initialize if necessary;
                              |  IF font number >= first font AND font number <= last font
                              |     THEN link nr := font. replacements table (code (char SUB 1) +
                              |          1);
                              |          IF link nr = maxint
                              |             THEN extended replacement (font number, char SUB 1,
                              |                  char SUB 2)
                              |             ELSE process font replacement
                              |          FI
                              |     ELSE errorstop ("Font " + text (font number) + " gibt es
                              |          nicht"); ""
                              |  FI
                              |
                              |  . font      : font store. fonts (font number)
                              |
                              |  . process font replacement :
                              |      IF link nr < 0 THEN link nr := link nr XOR (-maxint-1) FI;
                              |      IF   link nr = 0
                              |           THEN char
                              |      ELIF link nr > font store replacements length
                              |           THEN link nr DECR font store replacements length;
                              |                replacement text (font. replacements)
                              |          ELSE replacement text (font store. replacements)
                              |     FI
                              |
                              |END PROC replacement;
                              |
                              |


extendedreplacement ......|TEXT PROC extended replacement (INT  CONST font number,
                              |                                  TEXT CONST esc char, char ) :
                              |
                              |  initialize if necessary;
                              |  IF font number >= first font AND font number <= last font
                              |     THEN process extension replacement
                              |     ELSE errorstop ("Font " + text (font number) + " gibt es
                              |          nicht"); ""
                              |  FI
                              |
                              |  . process extension replacement :
                              |      determine extension link nr;
                              |      IF   link nr = 0
                              |           THEN char
                              |      ELIF link nr > font store extension replacements length
                              |           THEN link nr DECR font store extension replacements
                              |                length;
                              |                replacement text (font extension. replacements)
                              |           ELSE replacement text (font store extension. replacements)
                              |      FI
                              |
                              |      . determine extension link nr :
                              |          INT CONST index 1 := pos (font. extension chars, esc
                              |               char);
                              |          INT CONST index 2 := pos (font store. extension chars, esc
                              |               char);
                              |          IF   index 1 <> 0
                              |               THEN link nr := font extension. replacements table
                              |                    (code (char) + 1);
                              |          ELIF index 2 <> 0
                              |               THEN link nr := font store extension. replacements
                              |                    table (code (char) + 1);
                              |               ELSE errorstop ("""" + esc char + char + """ hat
                              |                    keine Erweiterung")
                              |          FI;
                              |
                              |      . font extension             : font store. extensions (font
                              |           extension number)
                              |
                              |      . font extension number      : font. extension indexes ISUB
                              |           index 1
                              |
                              |      . font                       : font store. fonts (font number)
                              |
                              |      . font store extension       : font store. extensions (font
                              |           store extension number)
                              |
                              |      . font store extension number : font store. extension indexes
                              |           ISUB index 2
                              |
                              |      . font store extension replacements length :
                              |          IF index 2 = 0
                              |            THEN 0
                              |            ELSE LENGTH font store extension. replacements
```

```
                           |          FI
                           |
                           |END PROC extended replacement;
                           |
                           |

replacementtext ...........|TEXT PROC replacement text (TEXT CONST replacements) :
                           |
                           |  buffer := subtext (replacements, link nr + 1,
                           |                                  link nr + code (replacements SUB
                           |                                       link nr));
                           |  buffer
                           |
                           |END PROC replacement text;
                           |
                           |

fontstring ................|TEXT PROC font string (INT CONST font number) :
                           |
                           |  initialize if necessary;
                           |  IF font number >= first font AND font number <= last font
                           |     THEN fonts. font string
                           |     ELSE errorstop ("Font " + text (font number) + " gibt es
                           |          nicht"); ""
                           |  FI
                           |
                           |  . fonts : font store. fonts (font number)
                           |
                           |END PROC font string;
                           |
                           |

yoffsets ..................|TEXT PROC y offsets (INT CONST font number) :
                           |
                           |  initialize if necessary;
                           |  IF font number >= first font AND font number <= last font
                           |     THEN fonts. y offsets
                           |     ELSE errorstop ("Font " + text (font number) + " gibt es
                           |          nicht"); ""
                           |  FI
                           |
                           |  . fonts : font store. fonts (font number)
                           |
                           |END PROC y offsets;
                           |
                           |

boldoffset ................|INT PROC bold offset (INT CONST font number) :
                           |
                           |  initialize if necessary;
                           |  IF font number >= first font AND font number <= last font
                           |     THEN fonts. bold offset
                           |     ELSE errorstop ("Font " + text (font number) + " gibt es
                           |          nicht"); 0
                           |  FI
                           |
                           |  . fonts : font store. fonts (font number)
                           |
                             |END PROC bold offset;
                             |
                             |


getfont .....................|PROC get font (INT CONST font number,
                             |               INT VAR indentation pitch, font lead, font height,
                             |                    font depth,
                             |               ROW 256 INT VAR pitch table ) :
                             |
                             |  initialize if necessary;
                             |  IF font number >= first font AND font number <= last font
                             |     THEN indentation pitch := fonts. indentation pitch;
                             |          pitch table       := fonts. pitch table;
                             |          font lead         := fonts. font lead;
                             |          font height       := fonts. font height;
                             |          font depth        := fonts. font depth;
                             |     ELSE errorstop ("Font " + text (font number) + " gibt es
                             |          nicht");
                             |  FI;
                             |
                             |  . fonts : font store. fonts (font number)
                             |
                             |END PROC get font;
                             |
                             |


getreplacements .............|PROC get replacements (INT CONST font number,
                             |                       TEXT VAR replacements,
                             |                       ROW 256 INT VAR replacements table) :
                             |
                             |  initialize if necessary;
                             |  IF font number >= first font AND font number <= last font
                             |     THEN replacements       := font store. replacements;
                             |          replacements      CAT fonts. replacements;
                             |          replacements table := fonts. replacements table;
                             |     ELSE errorstop ("Font " + text (font number) + " gibt es
                             |          nicht");
                             |  FI;
                             |
                             |  . fonts : font store. fonts (font number)
                             |
                             |END PROC get replacements;
                             |
                             |


initializeifnecessary .......|PROC initialize if necessary :
                             |
                             |  IF NOT initialized (in this task)
                             |     THEN IF font table = ""
                             |             THEN in this task := FALSE;
                             |                  errorstop ("Fonttabelle noch nicht eingestellt");
                             |             ELSE font table (font table);
                             |          FI;
                             |  FI;
                             |
                             |END PROC initialize if necessary;
                             |
                             |
                             |END PACKET font store;
```

```
                               |(* ------------------- VERSION 3     17.03.86 ------------------- *)
nameset *******************|PACKET name set DEFINES                         (* Autor: J.Liedtke *)
                               |
                               |       ALL ,
                               |       SOME ,
                               |       LIKE ,
                               |       + ,
                               |       - ,
                               |       / ,
                               |       do ,
                               |       FILLBY ,
                               |       remainder ,
                               |
                               |       fetch ,
                               |       save ,
                               |       fetch all ,
                               |       save all ,
                               |       forget ,
                               |       erase ,
                               |       insert ,
                               |       edit :
                               |
                               |
                               |LET cr lf = ""13""10"" ;
                               |
                               |TEXT VAR name ;
                               |DATASPACE VAR edit space ;
                               |
                               |THESAURUS VAR remaining thesaurus := empty thesaurus ;
                               |
                               |


+ ........................|THESAURUS OP + (THESAURUS CONST left, right) :
                               |
                               |  THESAURUS VAR union := left ;
                               |  INT VAR index := 0 ;
                               |  get (right, name, index) ;
                               |  WHILE name <> "" REP
                               |    IF NOT (union CONTAINS name)
                               |      THEN insert (union, name)
                               |    FI ;
                               |    get (right, name, index)
                               |  PER ;
                               |  union .
                               |
                               |ENDOP + ;
                               |


+ ........................|THESAURUS OP + (THESAURUS CONST left, TEXT CONST right) :
                               |
                               |  THESAURUS VAR union := left ;
                               |  IF NOT (union CONTAINS right)
                               |    THEN insert (union, right)
                               |  FI ;
                               |  union .
                               |
                               |ENDOP + ;
                               |
-  ........................|THESAURUS OP - (THESAURUS CONST left, right) :
                           |
                           |  THESAURUS VAR difference := empty thesaurus ;
                           |  INT VAR index := 0 ;
                           |  get (left, name, index) ;
                           |  WHILE name <> "" REP
                           |    IF NOT (right CONTAINS name)
                           |      THEN insert (difference, name)
                           |    FI ;
                           |    get (left, name, index)
                           |  PER ;
                           |  difference .
                           |
                           |ENDOP - ;
                           |


-  ........................|THESAURUS OP - (THESAURUS CONST left, TEXT CONST right) :
                           |
                           |  THESAURUS VAR difference := left ;
                           |  INT VAR index ;
                           |  delete (difference, right, index) ;
                           |  difference .
                           |
                           |ENDOP - ;
                           |


/  ........................|THESAURUS OP / (THESAURUS CONST left, right) :
                           |
                           |  THESAURUS VAR intersection := empty thesaurus ;
                           |  INT VAR index := 0 ;
                           |  get (left, name, index) ;
                           |  WHILE name <> "" REP
                           |    IF right CONTAINS name
                           |      THEN insert (intersection, name)
                           |    FI ;
                           |    get (left, name, index)
                           |  PER ;
                           |  intersection .
                           |
                           |ENDOP / ;
                           |


ALL .......................|THESAURUS OP ALL (TEXT CONST file name) :
                           |
                           |  FILE VAR file := sequential file (input, file name) ;
                           |  THESAURUS VAR thesaurus := empty thesaurus ;
                           |  thesaurus FILLBY file ;
                           |  thesaurus .
                           |
                           |ENDOP ALL ;
                           |


SOME ......................|THESAURUS OP SOME (THESAURUS CONST thesaurus) :
                           |
                           |  copy thesaurus into file ;
                           |  edit file ;
                           |  copy file into thesaurus .
                           |
    copythesaurusintofile  |copy thesaurus into file :
                           |  forget (edit space) ;
                           |  edit space := nilspace ;
                           |  FILE VAR file := sequential file (output, edit space) ;
                           |  file FILLBY thesaurus .
                           |
                           |
    editfile               |edit file :
                           |  modify (file) ;
                           |  edit (file) .
                           |
                           |
    copyfileintothesaurus  |copy file into thesaurus :
                           |  THESAURUS VAR result := empty thesaurus ;
                           |  input (file) ;
                           |  result FILLBY file ;
                           |  forget (edit space) ;
                           |  result .
                           |
                           |ENDOP SOME ;
                           |


SOME ......................|THESAURUS OP SOME (TASK CONST task) :
                           |
                           |  SOME ALL task
                           |
                           |ENDOP SOME ;
                           |


SOME ......................|THESAURUS OP SOME (TEXT CONST file name) :
                           |
                           |  SOME ALL file name
                           |
                           |ENDOP SOME ;
                           |


LIKE ......................|THESAURUS OP LIKE (THESAURUS CONST thesaurus, TEXT CONST pattern) :
                           |
                           |  THESAURUS VAR result:= empty thesaurus ;
                           |  INT VAR index:= 0 ;
                           |  REP get (thesaurus, name, index) ;
                           |    IF name = ""
                           |      THEN LEAVE LIKE WITH result
                           |    ELIF name LIKE pattern
                           |      THEN insert (result, name)
                           |    FI
                           | PER ;
                           | result .
                           |
                           |ENDOP LIKE ;
                           |


remainder .................|THESAURUS PROC remainder :
                           |
                           |  remaining thesaurus
                           |
                           |ENDPROC remainder ;
                           |
do .......................|PROC do (PROC (TEXT CONST) operate, THESAURUS CONST thesaurus) :
                          |
                          |  INT VAR index := 0 , operation number := 0 ;
                          |  TEXT VAR name ;
                          |
                          |  remaining thesaurus := empty thesaurus ;
                          |  disable stop ;
                          |  work off thesaurus ;
                          |  fill leftover with remainder .
                          |
                          |
   workoffthesaurus       |work off thesaurus :
                          |  REP
                          |    get (thesaurus, name, index) ;
                          |    IF name = ""
                          |      THEN LEAVE work off thesaurus
                          |    FI ;
                          |    operation number INCR 1 ;
                          |    cout (operation number) ;
                          |    execute (PROC (TEXT CONST) operate, name)
                          |  UNTIL is error ENDREP .
                          |
                          |
   fillleftoverwithremain |fill leftover with remainder :
                          |  WHILE name <> "" REP
                          |    insert (remaining thesaurus, name) ;
                          |    get (thesaurus, name, index)
                          |  PER .
                          |
                          |ENDPROC do ;
                          |


execute ..................|PROC execute (PROC (TEXT CONST) operate, TEXT CONST name) :
                          |
                          |  enable stop ;
                          |  operate (name)
                          |
                          |ENDPROC execute ;
                          |


do .......................|PROC do (PROC (TEXT CONST, TASK CONST) operate, THESAURUS CONST
                          |     thesaurus,
                          |         TASK CONST task) :
                          |
                          |  INT VAR index := 0 , operation number := 0 ;
                          |  TEXT VAR name ;
                          |
                          |  remaining thesaurus := empty thesaurus ;
                          |  disable stop ;
                          |  work off thesaurus ;
                          |  fill leftover with remainder .
                          |
                          |
   workoffthesaurus       |work off thesaurus :
                          |  REP
                          |    get (thesaurus, name, index) ;
                          |    IF name = ""
                          |      THEN LEAVE work off thesaurus
                          |    FI ;
                          |    operation number INCR 1 ;
```

```
                            |    cout (operation number) ;
                            |    execute (PROC (TEXT CONST, TASK CONST) operate, name, task)
                            |  UNTIL is error ENDREP .
                            |
                            |
      fillleftoverwithremain|fill leftover with remainder :
                            |  WHILE name <> "" REP
                            |    insert (remaining thesaurus, name) ;
                            |    get (thesaurus, name, index)
                            |  PER .
                            |
                            |ENDPROC do ;
                            |


  execute ..................|PROC execute (PROC (TEXT CONST, TASK CONST) operate,
                            |                TEXT CONST name, TASK CONST task) :
                            |
                            |  enable stop ;
                            |  operate (name, task)
                            |
                            |ENDPROC execute ;
                            |


  FILLBY ...................|OP FILLBY (THESAURUS VAR thesaurus, FILE VAR file) :
                            |
                            |  WHILE NOT eof (file) REP
                            |    getline (file, name) ;
                            |    delete trailing blanks ;
                            |    IF name <> "" CAND NOT (thesaurus CONTAINS name)
                            |      THEN insert (thesaurus, name)
                            |    FI
                            |  PER .
                            |
                            |
      deletetrailingblanks  |delete trailing blanks :
                            |  WHILE (name SUB LENGTH name) = " " REP
                            |    name := subtext (name, 1, LENGTH name - 1)
                            |  PER .
                            |
                            |ENDOP FILLBY ;
                            |


  FILLBY ...................|OP FILLBY (FILE VAR file, THESAURUS CONST thesaurus) :
                            |
                            |  INT VAR index := 0 ;
                            |  REP
                            |    get (thesaurus, name, index) ;
                            |    IF name = ""
                            |      THEN LEAVE FILLBY
                            |    FI ;
                            |    putline (file, name)
                            |  PER .
                            |
                            |ENDOP FILLBY ;
                            |
FILLBY ..................|OP FILLBY (TEXT CONST file name, THESAURUS CONST thesaurus) :
                         |
                         |  FILE VAR f := sequential file (output, file name) ;
                         |  f FILLBY thesaurus
                         |
                         |ENDOP FILLBY ;
                         |
                         |
                         |


fetch ...................|PROC fetch (THESAURUS CONST nameset) :
                         |
                         |  do (PROC (TEXT CONST) fetch, nameset)
                         |
                         |ENDPROC fetch ;
                         |


fetch ...................|PROC fetch (THESAURUS CONST nameset, TASK CONST task) :
                         |
                         |  do (PROC (TEXT CONST, TASK CONST) fetch, nameset, task)
                         |
                         |ENDPROC fetch ;
                         |


save ....................|PROC save (THESAURUS CONST nameset) :
                         |
                         |  do (PROC (TEXT CONST) save, nameset)
                         |
                         |ENDPROC save ;
                         |


save ....................|PROC save (THESAURUS CONST nameset, TASK CONST task) :
                         |
                         |  do (PROC (TEXT CONST, TASK CONST) save, nameset, task)
                         |
                         |ENDPROC save ;
                         |


fetchall ................|PROC fetch all :
                         |
                         |  fetch all (father)
                         |
                         |ENDPROC fetch all ;
                         |


fetchall ................|PROC fetch all (TASK CONST manager) :
                         |
                         |  fetch (ALL manager, manager)
                         |
                         |ENDPROC fetch all ;
                         |


saveall .................|PROC save all :
                         |
                         |  save all (father)
                         |
                         |ENDPROC save all ;
                         |

saveall .................|PROC save all (TASK CONST manager) :
                         |
                         |  save (ALL myself, manager)
                         |
                         |ENDPROC save all ;
                         |

forget ..................|PROC forget (THESAURUS CONST nameset) :
                         |
                         |  do (PROC (TEXT CONST) forget, nameset)
                         |
                         |ENDPROC forget ;
                         |

erase ...................|PROC erase (THESAURUS CONST nameset) :
                         |
                         |  do (PROC (TEXT CONST) erase, nameset)
                         |
                         |ENDPROC erase ;
                         |

erase ...................|PROC erase (THESAURUS CONST nameset, TASK CONST task) :
                         |
                         |  do (PROC (TEXT CONST, TASK CONST) erase, nameset, task)
                         |
                         |ENDPROC erase ;
                         |

insert ..................|PROC insert (THESAURUS CONST nameset) :
                         |
                         |  do (PROC (TEXT CONST) insert, nameset)
                         |
                         |ENDPROC insert ;
                         |

edit ....................|PROC edit (THESAURUS CONST nameset) :
                         |
                         |  do (PROC (TEXT CONST) edit, nameset)
                         |
                         |ENDPROC edit ;
                         |
                         |ENDPACKET name set ;
```

```
                              |
systeminfo ***************** |PACKET system info DEFINES                      (* Autor: J.Liedt
                              |     *)
                              |                                                 (* Stand: 22.09.84
                              |                                                      *)
                              |    task status ,
                              |    storage info ,
                              |    help :
                              |
                              |
                              |LET channel field  = 4 ,
                              |    prio field     = 6 ,
                              |
                              |    cr lf      = ""13""10"" ,
                              |    cr         = ""13"" ,
                              |    page       = ""1""4"" ,
                              |    begin mark= ""15"" ,
                              |    end mark  = ""14"" ,
                              |    bell       = ""7"" ,
                              |    esc        = ""27"" ;
                              |
                              |
                              |


cputimeof ................   |TEXT PROC cpu time of (TASK CONST actual task) :
                              |
                              |  disable stop ;
                              |  TEXT VAR result := subtext (time (clock (actual task), 12), 1, 10)
                              |       ;
                              |  IF is error
                              |    THEN clear error ;
                              |         result := 10 * "*"
                              |  FI ;
                              |  result
                              |
                              |ENDPROC cpu time of ;
                              |


taskstatus ...............   |PROC task status :
                              |
                              |  line ;
                              |  put (date); put (time of day) ;
                              |  line (2) ;
                              |  put ("Speicher:"); put (storage (myself)); putline ("K");
                              |  put ("CPU-Zeit:"); put (cpu time of (myself)) ; line;
                              |  line .
                              |
                              |ENDPROC task status ;
                              |


storageinfo ..............   |PROC storage info :
                              |
                              |  INT VAR size, used ;
                              |  storage (size, used) ;
                              |  out (""13""10"      ") ;
                              |  put (used) ;
                              |  put ("K von") ;
                              |  put (size plus reserve) ;
                              |  putline ("K sind belegt!") .
```

```
                                  |
                                  |
     sizeplusreserve              |size plus reserve :
                                  |  int (real (size + 24) * 64.0 / 63.0 ) .
                                  |
                                  |ENDPROC storage info ;
                                  |
                                  |


help .....................        |PROC help :
                                  |
                                  |  IF exists ("help")
                                  |    THEN FILE VAR f := sequential file (modify, "help") ;
                                  |         help (f)
                                  |    ELSE errorstop ("""help"" gibt es nicht")
                                  |  FI .
                                  |
                                  |ENDPROC help ;
                                  |


help .....................        |PROC help (FILE VAR help file) :
                                  |
                                  |  initialize help command ;
                                  |  REP
                                  |    out (page) ;
                                  |    to paragraph ;
                                  |    show paragraph ;
                                  |    get show command
                                  |  UNTIL is quit command PER .
                                  |
                                  |
     initializehelpcommand        |initialize help command :
                                  |   TEXT VAR
                                  |   help command := getcharety ;
                                  |   IF help command = ""
                                  |     THEN help command := "0"
                                  |   FI .
                                  |
                                  |
     toparagraph                  |to paragraph :
                                  |  col (help file, 1) ;
                                  |  to line (help file, 1) ;
                                  |  downety (help file, "#" + help command + "#") ;
                                  |  IF eof (help file)
                                  |    THEN to line (help file, 1) ;
                                  |         out (bell)
                                  |  FI .
                                  |
                                  |
     showparagraph                |show paragraph :
                                  |  show headline ;
                                  |  WHILE NOT end of help subfile REP
                                  |    show help line
                                  |  PER ;
                                  |  show bottom line .
                                  |
                                  |
     showheadline                 |show headline :
                                  |  out (begin mark) ;
                                  |  INT CONST dots := (x size - len (help file) - 5) DIV 2 ;
                                  |  dots TIMESOUT "." ;
                                  |  exec (PROC show line, help file, 4) ;
                                  |  dots TIMESOUT "." ;
                                  |  out (end mark) ;
                                  |  down (help file) .
                                  |
                                  |
        showhelpline              |show help line :
                                  |  out (cr lf) ;
                                  |  exec (PROC show line, help file, 1) ;
                                  |  down (help file) .
                                  |
                                  |
        showbottomline            |show bottom line :
                                  |  cursor (5, y size) ;
                                  |  exec (PROC show line, help file, 3) ;
                                  |  out (cr) .
                                  |
                                  |
        getshowcommand            |get show command :
                                  |  TEXT VAR char ;
                                  |  get char (char) ;
                                  |  IF char = esc
                                  |    THEN get char (char)
                                  |  FI ;
                                  |  IF char >= " "
                                  |    THEN help command := char
                                  |    ELSE out (bell)
                                  |  FI .
                                  |
                                  |
        endofhelpsubfile          |end of help subfile : pos (help file,"##",1) <> 0 OR eof (help file)
  +                               |      .
                                  |
                                  |
        isquitcommand             |is quit command     : help command = "q" OR help command = "Q" .
                                  |
                                  |ENDPROC help ;
                                  |

    showline .................... |PROC show line (TEXT CONST line, INT CONST from) :
                                  |
                                  |  outsubtext (line, from, x size - from)
                                  |
                                  |ENDPROC show line ;
                                  |
                                  |ENDPACKET system info ;
```

```
                              |(* -------------------- VERSION 2     26.05.86 -------------------- *)
 singleusermonitor ********* |PACKET single user monitor DEFINES            (* Autor: J.Liedtke *)
                              |
                              |    monitor ,
                              |    shutup ,
                              |    save system ,
                              |    fixpoint ,
                              |    collect garbage blocks ,
                              |    set clock ,
                              |    set date :
                              |
                              |
                              |LET command list =
                              |
                              |"edit:1.01run:3.01runagain:5.0insert:6.01forget:8.01rename:10.2copy:1
                              |     1.2
                              |
    list                      |list:12.0storageinfo:13.0fetch:14.1save:15.01saveall:17.0shutup:18.0
                              |
    help                      |help:19.0 " ;
                              |
                              |LET text param type = 4 ,
                              |    main channel    = 1 ,
                              |    cr = ""13"" ,
                              |
                              |    garbage collect code = 1 ,
                              |    fixpoint code        = 2 ,
                              |    shutup code          = 4 ,
                              |    shutup and save code = 12 ;
                              |
                              |
                              |INT VAR command index , number of params , previous heap size ,
                              |        old session := session ;
                              |TEXT VAR param 1, param 2 , date text;
                              |
                              |


 monitor .................... |PROC monitor :
                              |
                              |  monitor (PROC set up)
                              |
                              |ENDPROC monitor ;
                              |


 monitor .................... |PROC monitor (PROC init system) :
                              |
                              |  disable stop ;
                              |  previous heap size := heap size ;
                              |  REP
                              |    continue (main channel) ;
                              |    command dialogue (TRUE) ;
                              |    sysin ("") ;
                              |    sysout ("") ;
                              |    reset editor ;
                              |    init system if necessary ;
                              |    cry if not enough storage ;
                              |    get command ("gib kommando :") ;
                              |    analyze command (command list, text param type,
                              |                     command index, number of params, param1,
                              |                          param2) ;
                              |    execute command ;
                              |    collect heap garbage if necessary
                              |  PER .
                              |
                              |
     collectheapgarbageifne   |collect heap garbage if necessary :
                              |  IF heap size > previous heap size + 6
                              |    THEN collect heap garbage ;
                              |         previous heap size := heap size
                              |  FI .
                              |
                              |
     initsystemifnecessary    |init system if necessary :
                              |  IF session <> old session
                              |    THEN old session := session ;
                              |         continue (main channel) ;
                              |         clear error ;
                              |         init system ;
                              |         eumel must advertise ;
                              |         set date ;
                              |         storage info
                              |  FI .
                              |
                              |
     cryifnotenoughstorage    |cry if not enough storage :
                              |  INT VAR size, used ;
                              |  storage (size, used) ;
                              |  IF used > size
                              |    THEN out (""7"Speicher Engpass! Dateien loeschen!"13""10"")
                              |  FI .
                              |
                              |
     reseteditor              |reset editor :
                              |  WHILE aktueller editor > 0 REP
                              |    quit
                              |  PER .
                              |
                              |ENDPROC monitor ;
                              |


   executecommand ..........  |PROC execute command :
                              |
                              |  enable stop ;
                              |  SELECT command index OF
                              |    CASE 1 : edit
                              |    CASE 2 : edit (param1)
                              |    CASE 3 : run
                              |    CASE 4 : run (param1)
                              |    CASE 5 : run again
                              |    CASE 6 : insert
                              |    CASE 7 : insert (param1)
                              |    CASE 8 : forget
                              |    CASE 9 : forget (param1)
                              |    CASE 10: rename (param1, param2)
                              |    CASE 11: copy (param1, param2)
                              |    CASE 12: list
                              |    CASE 13: storage info
                              |    CASE 14: fetch (param1)
                              |    CASE 15: save
                              |    CASE 16: save (param1)
                              |    CASE 17: save all
```

```
                             |    CASE 18: shutup
                             |    CASE 19: help
                             |    OTHERWISE do command
                             |  ENDSELECT .
                             |
                             |ENDPROC  execute command ;
                             |
                             |BOOL VAR hardware clock ok ;
                             |REAL VAR now ;
                             |


setdate .................... |PROC set date :
                             |
                             |  hardware clock ok := TRUE ;
                             |  try to get date and time from hardware ;
                             |  IF NOT hardware clock ok
                             |    THEN get date and time from user
                             |  FI ;
                             |  define date and time .
                             |
                             |
   trytogetdateandtimefro    |try to get date and time from hardware :
                             |  disable stop ;
                             |  REAL VAR previous now ;
                             |  now := 0.0 ;
                             |  INT VAR try ;
                             |  FOR try FROM 1 UPTO 3 WHILE hardware clock ok REP
                             |    previous now := now ;
                             |    now := date (hardwares today) + time (hardwares time)
                             |  UNTIL now = previous now OR is error PER ;
                             |  clear error ;
                             |  enable stop .
                             |
                             |
   getdateandtimefromuser    |get date and time from user :
                             |  line (2) ;
                             |  put ("      Bitte geben Sie das heutige Datum ein :") ;
                             |  date text := date ;
                             |  TEXT VAR exit char ;
                             |  editget (date text, cr, "", exit char) ;
                             |  now := date (date text) ;
                             |  line ;
                             |  put ("      und die aktuelle Uhrzeit :") ;
                             |  date text := time of day ;
                             |  editget (date text, cr, "", exit char) ;
                             |  now INCR time (date text) ;
                             |  IF NOT last conversion ok
                             |    THEN errorstop ("Falsche Zeitangabe")
                             |  FI .
                             |
                             |
   hardwarestoday            |hardwares today :  calendar (3) + "." + calendar (4) + "." +
                             |     calendar (5) .
                             |
                             |
   hardwarestime             |hardwares time  :  calendar (2) + ":" + calendar (1) .
                             |
                             |
   definedateandtime         |define date and time :
                             |  set clock (now) .
                             |
                         |ENDPROC set date ;
                         |


calendar ................|TEXT PROC calendar (INT CONST index) :
                         |
                         |  INT VAR bcd ;
                         |  control (10, index, 0, bcd) ;
                         |  IF bcd < 0
                         |    THEN hardware clock ok := FALSE ; ""
                         |    ELSE text (low digit + 10 * high digit)
                         |  FI .
                         |
                         |
   lowdigit              |low digit :  bcd AND 15 .
                         |
                         |
   highdigit             |high digit:  (bcd AND (15*256)) DIV 256 .
                         |
                         |ENDPROC calendar ;
                         |


shutup ..................|PROC shutup :
                         |
                         |  page ;
                         |  cursor (32, 15) ;
                         |  put ("bitte warten") ;
                         |  cursor (35, 12) ;
                         |  system operation (shutup code)
                         |
                         |ENDPROC shutup ;
                         |


savesystem ..............|PROC save system :
                         |
                         |  archive ("save") ;
                         |  IF yes ("Leere Floppy eingelegt")
                         |  THEN
                         |    system operation (shutup and save code) ;
                         |  FI.
                         |
                         |ENDPROC save system ;
                         |


collectgarbageblocks .....|PROC collect garbage blocks :
                         |
                         |  system operation (garbage collect code)
                         |
                         |ENDPROC collect garbage blocks ;
                         |


fixpoint ................|PROC fixpoint :
                         |
                         |  system operation (fixpoint code)
                         |
                         |ENDPROC fixpoint ;
                         |
systemoperation ..........|PROC system operation (INT CONST code) :
                          |
                          |  INT VAR size, used ;
                          |  storage (size, used) ;
                          |  IF used <= size
                          |    THEN disable stop ;
                          |         sys op (code) ;
                          |         ignore start message error
                          |    ELSE errorstop ("nicht genuegend System - Speicher vorhanden")
                          |  FI .
                          |
                          |
   ignorestartmessageerro |ignore start message error :
                          |  pause (5) ;
                          |  clear error .
                          |
                          |ENDPROC system operation ;
                          |


sysop ....................|PROC sys op (INT CONST code) :
                          |  EXTERNAL 90
                          |ENDPROC sys op ;
                          |


setclock .................|PROC set clock (REAL CONST time) :
                          |  EXTERNAL 103
                          |ENDPROC set clock ;
                          |
                          |ENDPACKET single user monitor ;
```

```
                            |ke ; (* maintenance ke *)
                            |

sysgenoff ..................|PROC sysgen off (INT CONST mode, INT VAR a,b,c,d,e,f,g,h,i,j,k)
                            |  EXTERNAL 256
                            |ENDPROC sysgen off ;
                            |
                            |INT VAR x := 0 ;
                            |sysgen off (3,x,x,x,x,x,x,x,x,x,x,x) ;
```

```

check on ;
command dialogue (TRUE) ;
set clock (date ("19.06.86")) ;
disable stop ;
save system ;
REP UNTIL yes ("help") PER ;
archive ("help") ;
fetch ("help", archive) ;
REP UNTIL yes ("dev") PER ;
archive ("dev") ;
fetch all (archive) ;
release (archive) ;
save system ;
configurate ;
set up ;
monitor ;
```

```
                             |(* ------------------- VERSION 11     06.03.86 --------------
basicarchive ****************|PACKET basic archive DEFINES
                             |
                             |  archive blocks ,
                             |  block number ,
                             |  check read ,
                             |  format archive ,
                             |  read block ,
                             |  read ,
                             |  rewind ,
                             |  search dataspace ,
                             |  seek ,
                             |  size ,
                             |  skip dataspace ,
                             |  write block ,
                             |  write :
                             |
                             |INT VAR blocknr  := 0 ,
                             |        rerun    := 0 ,
                             |        page     := -1 ,
                             |        bit word := 1 ,
                             |        unreadable sequence length := 0 ;
                             |INT CONST all ones :=-1 ;
                             |
                             |
                             |DATASPACE VAR label ds ;
                             |
                             |LET write normal            = 0 ,
                             |    archive version         = 1 ,
                             |    first page stored       = 2 ,
                             |    dr size                 = 3 ,
                             |    first bit word          = 4 ,
                             |(*  write deleted data mark = 64 , *)
                             |    inconsistent            = 90 ,
                             |    read error              = 92 ,
                             |    label size              = 131 ;
                             |
                             |BOUND STRUCT (ALIGN dummy for page1,
                             |              (* Page 2 begins: *)
                             |              ROW label size INT lab) VAR label;
                             |
                             |


blocknumber .................|INT PROC block number :
                             |  block nr
                             |ENDPROC block number ;
                             |


seek ........................|PROC seek (INT CONST block) :
                             |  block nr := block
                             |ENDPROC seek ;
                             |


rewind ......................|PROC rewind :
                             |  forget (label ds);
                             |  label ds := nilspace;
                             |  label := label ds;
                             |  block nr := 0;
                             |  rerun := session
                             |END PROC rewind;
                             |

skipdataspace ...............|PROC skip dataspace:
                             |  check rerun;
                             |  get label;
                             |  IF is error
                             |    THEN
                             |  ELIF olivetti
                             |    THEN block nr INCR label.lab (dr size+1)
                             |    ELSE block nr INCR label.lab (dr size)
                             |  FI
                             |END PROC skip dataspace;
                             |

read ........................|PROC read (DATASPACE VAR ds):
                             |  read (ds, 30000, FALSE)
                             |ENDPROC read ;
                             |

read ........................|PROC read (DATASPACE VAR ds, INT CONST max pages, BOOL CONST error
   +                         |     accept) :
                             |  enable stop ;
                             |  check rerun;
                             |  get label;
                             |  init next page;
                             |  INT VAR i ;
                             |  FOR i FROM 1 UPTO max pages REP
                             |    next page;
                             |    IF no further page THEN  LEAVE read FI;
                             |    check storage ;
                             |    check rerun ;
                             |    read block ;
                             |    block nr INCR 1;
                             |  PER .
                             |
                             |
   readblock                 |read block :
                             |  disable stop ;
                             |  get external block (ds, page, block nr) ;
                             |  ignore read error if no errors accepted ;
                             |  enable stop .
                             |
                             |
   ignorereaderrorifnoerr    |ignore read error if no errors accepted :
                             |  IF is error CAND error code = read error CAND NOT error accept
                             |    THEN clear error
                             |  FI .
                             |
                             |
   checkstorage              |check storage :
                             |  INT VAR size, used ;
                             |  storage (size, used) ;
                             |  IF used > size
                             |    THEN forget (ds) ;
                             |         ds := nilspace ;
                             |         errorstop ("Speicherengpass") ;
                             |         LEAVE read
                             |  FI .
```

```
                              |
                              |
   checkrerun                 |check rerun :
                              |    IF rerun <> session
                              |      THEN errorstop ("RERUN beim Archiv-Zugriff") ;
                              |           LEAVE read
                              |    FI .
                              |
                              |END PROC read;
                              |


checkread ................... |PROC check read :
                              |
                              |  enable stop ;
                              |  get label ;
                              |  INT VAR pages, i;
                              |  IF olivetti
                              |    THEN pages := label.lab (dr size+1)
                              |    ELSE pages := label.lab (dr size)
                              |  FI ;
                              |  FOR i FROM 1 UPTO pages REP
                              |    get external block (label ds, 2, block nr) ;
                              |    block nr INCR 1
                              |  PER .
                              |
                              |ENDPROC check read ;
                              |


write ....................... |PROC write (DATASPACE CONST ds):
                              |  enable stop ;
                              |  check rerun;
                              |  INT VAR label block nr := block nr;
                              |  block nr INCR 1;init label;
                              |  INT VAR page := -1,i;
                              |  FOR i FROM 1 UPTO ds pages (ds) REP
                              |    check rerun ;
                              |    page := next ds page(ds,page);
                              |    put external block (ds, page, block nr) ;
                              |    reset archive bit;
                              |    label.lab(dr size) INCR 1;
                              |    block nr INCR 1
                              |  PER;
                              |  put label.
                              |
                              |
                              |
   initlabel                  |  init label:
                              |  label.lab(archive version) := 0 ;
                              |  label.lab(first page stored) := 0 ;
                              |  label.lab(dr size) := 0;
                              |  INT VAR j;
                              |  FOR j FROM first bit word UPTO label size REP
                              |    label.lab (j) := all ones
                              |  PER.
                              |
                              |
   putlabel                   |  put label:
                              |  put external block (label ds, 2, label block nr).
                              |
   resetarchivebit         |  reset archive bit:
                           |  reset bit (label.lab (page DIV 16+first bit word), page MOD 16).
                           |
                           |END PROC write;
                           |


getlabel ..................|PROC get label:
                           |
                           |  enable stop ;
                           |  get external block (label ds, 2, block nr)  ;
                           |  block nr INCR 1;
                           |  check label.
                           |
                           |
   checklabel              |check label:
                           |  IF may be z80 format label OR may be old olivetti format label
                           |    THEN
                           |    ELSE errorstop (inconsistent, "Archiv inkonsistent")
                           |  FI.
                           |
                           |
   maybez80formatlabel     |may be z80 format label :
                           |  z80 archive AND label.lab(dr size) > 0 .
                           |
                           |
   maybeoldolivettiformat  |may be old olivetti format label :
                           |  olivetti AND label.lab(first page stored)=0 AND label.lab(dr
                           |       size+1) > 0 .
                           |
                           |END PROC get label;
                           |


nextpage ..................|PROC next page:
                           |  IF z80 archive
                           |  THEN
                           |    WHILE labelbits = all ones REP
                           |      bitword INCR 1;
                           |      IF bitword >= label size THEN
                           |      no further page := true; LEAVE next page FI
                           |    PER;
                           |    INT VAR p := lowest reset (labelbits);
                           |    set bit (labelbits, p);
                           |    page := 16*(bitword-first bit word)+p
                           |  ELSE
                           |    WHILE oli bits = 0 REP
                           |      bitword INCR 1;
                           |      IF bitword >= labelsize-64 THEN
                           |      no further page := true; LEAVE next page FI
                           |    PER;
                           |    p := lowest set (oli bits);
                           |    reset bit (olibits, p);
                           |    page := 16*(bitword-firstbitword)+p;
                           |  FI.
                           |
                           |
   labelbits               |  label bits : label.lab (bitword).
                           |
   olibits                 |  oli bits : label.lab (bitword+1).
                           |
                           |END PROC next page;
```

```
                              |.
                              |
    olivetti                  |olivetti : label.lab (archive version) = -1.
                              |
                              |
    z80archive                |z80 archive : label.lab (archive version) = 0.
                              |
                              |
    initnextpage              |init next page:
                              |   BOOL VAR no further page := false;
                              |   bitword := first bit word.
                              |
                              |
    checkrerun                |check rerun :
                              |    IF rerun <> session
                              |      THEN errorstop ("RERUN beim Archiv-Zugriff")
                              |    FI .
                              |


getexternalblock .........|PROC get external block (DATASPACE VAR ds, INT CONST page,
                              |                                INT CONST block nr):
                              |
                              |  INT VAR error ;
                              |  read block (ds, page, block nr, error) ;
                              |  SELECT error OF
                              |    CASE 0: read succeeded
                              |    CASE 1: error stop ("Lesen unmoeglich (Archiv)")
                              |    CASE 2: read failed
                              |    CASE 3: error stop ("Archiv-Ueberlauf")
                              |    OTHERWISE error stop ("??? (Archiv)")
                              |  END SELECT .
                              |
                              |
    readsucceeded             |read succeeded :
                              |  unreadable sequence length := 0 .
                              |
                              |
    readfailed                |read failed :
                              |  unreadable sequence length INCR 1 ;
                              |  IF unreadable sequence length >= 30
                              |    THEN errorstop ("30 unlesbare Bloecke hintereinander")
                              |    ELSE error stop (read error, "Lesefehler (Archiv)")
                              |  FI .
                              |
                              |END PROC get external block;
                              |


putexternalblock .........|PROC put external block (DATASPACE CONST ds, INT CONST page,
                              |                                INT CONST block nr):
                              |  INT VAR error;
                              |  write block (ds, page, write normal, block nr, error) ;
                              |  SELECT error OF
                              |    CASE 0:
                              |    CASE 1: error stop ("Schreiben unmoeglich (Archiv)")
                              |    CASE 2: error stop ("Schreibfehler (Archiv)")
                              |    CASE 3: error stop ("Archiv-Ueberlauf")
                              |    OTHERWISE error stop ("??? (Archiv)")
                              |  END SELECT .
                              |
                              |END PROC put external block;
                              |
readblock ...............     |PROC read block (DATASPACE VAR ds,
                              |                  INT CONST ds page no,
                              |                  INT CONST block no,
                              |                  INT VAR return code) :
                              |  read block;
                              |  retry if read error.
                              |
                              |
   readblock                  |read block:
                              |  block in (ds, ds page no, 0, block no, return code).
                              |
                              |
   retryifreaderror           |retry if read error:
                              |  INT VAR retry;
                              |  FOR retry FROM 1 UPTO 10 WHILE return code = 2 REP
                              |    reset to block 0 if fifth try;
                              |    read block
                              |  PER.
                              |
                              |
   resettoblock0iffifthtr     |reset to block 0 if fifth try:
                              |  IF retry = 5
                              |    THEN block in (ds, ds page no, 0, 0, return code)
                              |  FI.
                              |
                              |END PROC read block;
                              |


writeblock ..............     |PROC write block (DATASPACE CONST ds,
                              |                  INT CONST ds page no,
                              |                  INT CONST mode,
                              |                  INT CONST block no,
                              |                  INT VAR return code):
                              |  write block;
                              |  retry if write error.
                              |
                              |
   writeblock                 |write block:
                              |  block out (ds, ds page no, mode * 256, block no, return code) .
                              |
                              |
   retryifwriteerror          |retry if write error:
                              |  INT VAR retry;
                              |  FOR retry FROM 1 UPTO 10 WHILE return code = 2 REP
                              |    reset to block 0 if fifth try;
                              |    write block
                              |  PER.
                              |
                              |
   resettoblock0iffifthtr     |reset to block 0 if fifth try:
                              |  IF retry = 5
                              |    THEN disable stop;
                              |         DATASPACE VAR dummy ds := nilspace;
                              |         block in (dummy ds, 2, 0, 0, return code);
                              |         forget (dummy ds);
                              |         enable stop
                              |  FI.
                              |
                           |END PROC write block;
                           |


size ......................|INT PROC size (INT CONST key) :
                           |
                           |  INT VAR return code ;
                           |  control (5, key, 0, return code) ;
                           |  return code .
                           |
                           |ENDPROC size ;
                           |


archiveblocks .............|INT PROC archive blocks :
                           |  size (0)
                           |ENDPROC archive blocks ;
                           |


searchdataspace ...........|PROC search dataspace (INT VAR ds pages) :
                           |
                           |  disable stop ;
                           |  ds pages := -1 ;
                           |  INT CONST last block := archive blocks ;
                           |
                           |  WHILE block nr < last block REP
                           |    IF block is dataspace label
                           |      THEN ds pages := pages counted ;
                           |           LEAVE search dataspace
                           |    FI ;
                           |    block nr INCR 1
                           |  UNTIL is error PER .
                           |
                           |
   blockisdataspacelabel   |block is dataspace label :
                           |  look at label block ;
                           |  IF is error
                           |    THEN IF error code = read error OR error code = inconsistent
                           |           THEN clear error
                           |         FI ;
                           |         FALSE
                           |    ELSE count pages ;
                           |         pages counted = number of pages as label says
                           |  FI .
                           |
                           |
   lookatlabelblock        |look at label block :
                           |  INT CONST
                           |  old block nr := block nr ;
                           |  get label ;
                           |  block nr := old block nr.
                           |
                           |
   countpages              |count pages :
                           |  INT VAR
                           |  pages counted := 0 ;
                           |  init next page ;
                           |  next page ;
                           |  WHILE NOT no further page REP
                           |    pages counted INCR 1 ;
                           |    next page
```

```
                              |  PER .
                              |
                              |
   numberofpagesaslabelsa     |number of pages as label says :   label.lab (dr size) .
                              |
                              |ENDPROC search dataspace ;
                              |

formatarchive ............... |PROC format archive (INT CONST format code) :
                              |
                              |  IF format is possible
                              |    THEN format
                              |    ELSE errorstop ("'format' ist hier nicht implementiert")
                              |  FI .
                              |
                              |
   formatispossible           |format is possible :
                              |  INT VAR return code ;
                              |  control (1,0,0, return code) ;
                              |  bit (return code, 4) .
                              |
                              |
   format                     |format :
                              |  control (7, format code, 0, return code) ;
                              |  IF return code = 1
                              |    THEN errorstop ("Formatieren unmoeglich")
                              |  ELIF return code > 1
                              |    THEN errorstop ("Schreibfehler (Archiv)")
                              |  FI .
                              |
                              |ENDPROC format archive ;
                              |
                              |END PACKET basic archive;
```

```
archivesingle *************|PACKET archive single DEFINES                        (* Autor: J.Liedtke*)
                           |                                                     (* Stand: 31.07.85 *)
                           |     archive ,
                           |     release ,
                           |     save ,
                           |     fetch ,
                           |     erase ,
                           |     check ,
                           |     exists ,
                           |     ALL ,
                           |     clear ,
                           |     list ,
                           |     format :
                           |
                           |
                           |
                           |LET archive channel = 31 ,
                           |    main channel    = 1  ,
                           |
                           |    read error       = 92 ,
                           |
                           |    max files = 200 ,
                           |
                           |    start of volume = 1000 ,
                           |    end of volume = 1 ,
                           |    file header   = 3 ,
                           |
                           |    number of header blocks = 2 ,
                           |
                           |    quote          = """" ,
                           |    dummy name     = "-" ,
                           |    dummy date     = "        " ,
                           |
                           |
                           |    HEADER = STRUCT (TEXT name, date, INT type, TEXT password) ;
                           |
                           |
                           |
                           |TEXT VAR archive name := "" , write stamp ;
                           |
                           |REAL VAR last access time := 0.0 ;
                           |
                           |BOOL VAR was already write access := FALSE ;
                           |
                           |
                           |DATASPACE VAR header space := nilspace , ds := nilspace ;
                           |BOUND HEADER VAR header ;
                           |
                           |TEXT VAR file name := "" ;
                           |
                           |LET invalid   = 0 ,
                           |    read only = 1 ,
                           |    valid     = 2 ;
                           |
                           |LET accept read errors = TRUE ,
                           |    ignore read errors = FALSE ;
                           |
                           |
                           |INT VAR directory state := invalid ;
                           |
                              |THESAURUS VAR directory , all names ;
                              |INT VAR dir index ;
                              |
                              |INT VAR archive size ;
                              |
                              |INT VAR end of volume block ;
                              |ROW max files INT VAR header block ;
                              |ROW max files TEXT VAR header date ;
                              |
                              |
                              |


archive ......................|PROC archive (TEXT CONST name) :
                              |
                              |  disable stop ;
                              |  directory state := invalid ;
                              |  archive name := name ;
                              |  last access time := clock (1) .
                              |
                              |ENDPROC archive ;
                              |


release ......................|PROC release (TASK CONST t) :
                              |
                              |  directory state := invalid
                              |
                              |ENDPROC release ;
                              |
                              |


accessarchive ................|PROC access archive :
                              |
                              |  IF directory state = invalid
                              |    THEN open archive
                              |  ELIF last access more than two seconds ago
                              |    THEN check volume name ;
                              |         new open if somebody changed medium
                              |  FI .
                              |
                              |
   lastaccessmorethantwos     |last access more than two seconds ago :
                              |  abs (clock (1) - last access time) > 2.0 .
                              |
                              |
   newopenifsomebodychang     |new open if somebody changed medium :
                              |  IF header.date <> write stamp
                              |    THEN directory state := invalid ;
                              |         access archive
                              |  FI .
                              |
                              |
   openarchive                |open archive :
                              |  directory state := invalid ;
                              |  check volume name ;
                              |  write stamp := header.date ;
                              |  was already write access := FALSE ;
                              |  read directory ;
                              |  make directory valid if no read errors occurred .
                              |
     readdirectory           |read directory :
                             |  directory := empty thesaurus ;
                             |  rewind ;
                             |  get next header ;
                             |  WHILE header.type = file header REP
                             |    IF directory CONTAINS header.name
                             |      THEN rename (directory, header.name, dummy name)
                             |    FI ;
                             |    insert (directory, header.name, dir index) ;
                             |    header block (dir index) := end of volume block ;
                             |    header date (dir index)  := header.date ;
                             |    get next header ;
                             |  PER .
                             |
     makedirectoryvalidifno  |make directory valid if no read errors occurred :
                             |  IF directory state = invalid
                             |    THEN directory state := valid
                             |  FI .
                             |
                             |ENDPROC access archive ;
                             |


  accessfile ................|PROC access file (TEXT CONST name) :
                             |
                             |  file name := name ;
                             |  dir index := link (directory, file name) .
                             |
                             |ENDPROC access file ;
                             |
                             |


  checkvolumename ...........|PROC check volume name :
                             |
                             |  disable stop ;
                             |  archive size := archive blocks ;
                             |  read volume header ;
                             |  IF header.type <> start of volume
                             |    THEN simulate header (start of volume, "?????")
                             |  ELIF header.name <> archive name
                             |    THEN errorstop ("Archiv heisst """ + header.name + """")
                             |  FI .
                             |
     readvolumeheader        |read volume header :
                             |  rewind ;
                             |  read header ;
                             |  IF is error
                             |    THEN clear error ;
                             |         simulate header (start of volume, "?????")
                             |  FI .
                             |
                             |ENDPROC check volume name ;
                             |


  getnextheader .............|PROC get next header :
                             |
                             |  disable stop ;
                             |  skip dataspace ;
```

```
                          |  IF NOT is error
                          |    THEN read header
                          |  FI ;
                          |  IF is error
                          |    THEN clear error ;
                          |         directory state := read only ;
                          |         search header
                          |  FI ;
                          |  end of volume block := block number - number of header blocks .
                          |
                          |
     searchheader         |search header :
                          |  INT VAR ds pages ;
                          |  search dataspace (ds pages) ;
                          |  IF ds pages < 0
                          |    THEN simulate header (end of volume, "")
                          |  ELIF NOT is header space
                          |    THEN simulate header (file header, "????? " + text (block
                          |         number))
                          |  FI .
                          |
                          |
     isheaderspace        |is header space :
                          |  IF ds pages <> 1
                          |    THEN FALSE
                          |    ELSE remember position ;
                          |         read header ;
                          |         IF read error occurred
                          |           THEN clear error; back to old position; FALSE
                          |         ELIF header format looks ok
                          |           THEN TRUE
                          |           ELSE back to old position ; FALSE
                          |         FI
                          |  FI .
                          |
                          |
     readerroroccurred    |read error occurred :
                          |  is error CAND error code = read error .
                          |
                          |
     headerformatlooksok  |header format looks ok :
                          |  header.type = file header OR header.type = end of volume .
                          |
                          |
     rememberposition     |remember position :
                          |  INT CONST old block nr := block number .
                          |
                          |
     backtooldposition    |back to old position :
                          |  seek (old block nr) .
                          |
                          |ENDPROC get next header ;
                          |
                          |


   fetch .................|PROC fetch (TEXT CONST file name) :
                          |
                          |  fetch (file name, archive)
                          |
                          |ENDPROC fetch ;
                          |
fetch ....................|PROC fetch (TEXT CONST file name, TASK CONST from) :
                          |
                          |  enable stop;
                          |  IF NOT (from = archive)
                          |    THEN errorstop ("Task gibt es nicht")
                          |  ELIF NOT exists (file name) COR overwrite permitted
                          |     THEN get archive file
                          |  FI .
                          |
                          |
   getarchivefile         |get archive file:
                          |  last param (file name) ;
                          |  disable stop ;
                          |  continue (archive channel) ;
                          |  fetch file (file name) ;
                          |  last access time := clock (1) ;
                          |  continue (main channel) ;
                          |  IF NOT is error
                          |    THEN forget (file name, quiet) ;
                          |         copy (ds, file name)
                          |  FI ;
                          |  forget (ds) .
                          |
                          |
   overwritepermitted     |overwrite permitted :
                          |  say ("eigene datei """) ;
                          |  say (file name) ;
                          |  yes (""" ueberschreiben") .
                          |
                          |ENDPROC fetch ;
                          |


fetchfile ................|PROC fetch file (TEXT CONST name) :
                          |
                          |  enable stop ;
                          |  access archive ;
                          |  access file (name) ;
                          |  IF no read error remarked
                          |    THEN disable stop ;
                          |         fetch ds (accept read errors) ;
                          |         IF read error occurred
                          |           THEN remark read error
                          |         FI ;
                          |         enable stop
                          |    ELSE fetch ds (ignore read errors)
                          |  FI .
                          |
                          |
   noreaderrorremarked    |no read error remarked :
                          |  pos (name, " mit Lesefehler") = 0 .
                          |
                          |
   readerroroccurred      |read error occurred :
                          |  is error AND error code = read error .
                          |
                          |
   remarkreaderror        |remark read error :
                          |  dir index := link (directory, file name) ;
                          |  REP
                          |    file name CAT " mit Lesefehler" ;
                          |  UNTIL NOT (directory CONTAINS file name) PER ;
```

```
                             |  IF LENGTH file name < 100
                             |    THEN rename (directory, dir index, file name)
                             |  FI .
                             |
                             |ENDPROC fetch file ;
                             |


fetchds .....................|PROC fetch ds (BOOL CONST error accept) :
                             |
                             |  enable stop ;
                             |  IF file name <> dummy name
                             |    THEN fetch from archive
                             |    ELSE error ("Name unzulaessig")
                             |  FI .
                             |
                             |
   fetchfromarchive          |fetch from archive :
                             |  IF file in directory
                             |    THEN position to file ;
                             |         forget (ds) ;
                             |         ds := nilspace ;
                             |         read (ds, 30000, error accept) ;
                             |  ELIF directory state = read only
                             |    THEN error ("gibt es nicht (oder Lesefehler)")
                             |    ELSE error ("gibt es nicht")
                             |  FI .
                             |
                             |
   positiontofile            |position to file :
                             |  seek (header block (dir index) + number of header blocks) .
                             |
                             |
   fileindirectory           |file in directory :  dir index > 0 .
                             |
                             |ENDPROC fetch ds ;
                             |


erase .......................|PROC erase :
                             |
                             |  erase (last param)
                             |
                             |ENDPROC erase ;
                             |


erase .......................|PROC erase (TEXT CONST file name) :
                             |
                             |  erase (file name, archive)
                             |
                             |ENDPROC erase ;
                             |


erase .......................|PROC erase (TEXT CONST file name, TASK CONST dest) :
                             |
                             |  IF dest = archive
                             |    THEN disable stop ;
                             |         continue (archive channel) ;
                             |         erase on archive (file name) ;
                             |         last access time := clock (1) ;
```

```
                              |         continue (main channel)
                              |    ELSE errorstop ("Task gibt es nicht")
                              |  FI
                              |
                              |ENDPROC erase ;
                              |

eraseonarchive ...............|PROC erase on archive (TEXT CONST file name) :
                              |
                              |  enable stop ;
                              |  access archive ;
                              |  access file (file name) ;
                              |  continue (main channel) ;
                              |  IF NOT file in directory
                              |    THEN putline ("gibt es nicht") ;
                              |         LEAVE erase on archive
                              |  ELIF NOT yes (""""+file name+""" loeschen")
                              |    THEN LEAVE erase on archive
                              |  FI ;
                              |  continue (archive channel) ;
                              |  erase archive entry .
                              |
                              |
     fileindirectory          |file in directory :  dir index > 0 .
                              |
                              |ENDPROC erase on archive ;
                              |

erasearchiveentry ............|PROC erase archive entry :
                              |
                              |  IF directory state = read only
                              |    THEN errorstop ("'save'/'erase' wegen Lesefehler verboten")
                              |    ELSE update write stamp if first write access ;
                              |         erase archive
                              |  FI .
                              |
                              |
     updatewritestampiffirs   |update write stamp if first write access :
                              |  IF NOT was already write access
                              |    THEN rewind ;
                              |         write stamp := text (clock (1), 13, 1) ;
                              |         write header (archive name, write stamp, start of volume) ;
                              |         was already write access := TRUE
                              |  FI .
                              |
                              |
     erasearchive             |erase archive :
                              |  IF file in directory
                              |    THEN IF is last file of archive
                              |           THEN cut off all erased files
                              |           ELSE rename to dummy
                              |         FI
                              |  FI .
                              |
                              |
     fileindirectory          |file in directory :        dir index > 0 .
                              |
  islastfileofarchive   |is last file of archive :  dir index = highest entry (directory) .
                        |
                        |
  cutoffallerasedfiles  |cut off all erased files :
                        |  directory state := invalid ;
                        |  REP
                        |    delete (directory, dir index) ;
                        |    dir index DECR 1
                        |  UNTIL dir index = 0 COR name (directory, dir index) <> dummy name
                        |      PER ;
                        |  behind last valid file ;
                        |  write end of volume ;
                        |  directory state := valid .
                        |
                        |
  behindlastvalidfile   |behind last valid file :
                        |  seek (header block (dir index + 1)) ;
                        |  end of volume block := block number .
                        |
                        |
  renametodummy         |rename to dummy :
                        |    directory state := invalid ;
                        |    to file header ;
                        |    read header ;
                        |    to file header ;
                        |    header.name := dummy name ;
                        |    header.date := dummy date ;
                        |    write (header space) ;
                        |    rename (directory, file name, dummy name) ;
                        |    header date (dir index) := dummy date ;
                        |    directory state := valid .
                        |
                        |
  tofileheader          |to file header :
                        |  seek (header block (dir index)) .
                        |
                        |ENDPROC erase archive entry ;
                        |


save ...................|PROC save :
                        |
                        |  save (last param)
                        |
                        |ENDPROC save ;
                        |


save ...................|PROC save (TEXT CONST file name) :
                        |
                        |  save (file name, archive)
                        |
                        |ENDPROC save ;
                        |


save ...................|PROC save (TEXT CONST file name, TASK CONST to) :
                        |
                        |  IF to = archive
                        |    THEN disable stop ;
                        |         continue (archive channel) ;
                        |         save to archive (file name) ;
```

```
                              |          last access time := clock (1) ;
                              |          continue (main channel)
                              |     ELSE errorstop ("Task gibt es nicht")
                              |   FI .
                              |
                              |ENDPROC save ;
                              |

savetoarchive ................|PROC save to archive (TEXT CONST file name) :
                              |
                              |   enable stop ;
                              |   access archive ;
                              |   access file (file name) ;
                              |   continue (main channel) ;
                              |   IF file in directory
                              |     THEN IF NOT yes (""""+file name+""" ueberschreiben")
                              |            THEN LEAVE save to archive
                              |          FI
                              |   FI ;
                              |   continue (archive channel) ;
                              |   access archive ;
                              |   access file (file name) ;
                              |   erase archive entry ;
                              |   IF file name = dummy name
                              |     THEN error ("Name unzulaessig")
                              |   ELIF file too large OR highest entry (directory) >= max files
                              |     THEN error ("kann nicht geschrieben werden (Archiv voll)")
                              |     ELSE write new file
                              |   FI .
                              |
                              |
   fileindirectory            |file in directory :  dir index > 0 .
                              |
                              |
   filetoolarge               |file too large :
                              |   end of volume block + ds pages (ds) + 5 > archive size .
                              |
                              |
   writenewfile               |write new file :
                              |   seek (end of volume block) ;
                              |   disable stop ;
                              |   write file (file name, old (file name)) ;
                              |   IF is error
                              |     THEN seek (end of volume block)
                              |     ELSE insert (directory, file name, dir index) ;
                              |          remember begin of header block ;
                              |          remember date
                              |   FI ;
                              |   write end of volume .
                              |
                              |
   rememberbeginofheaderb     |remember begin of header block :
                              |   header block (dir index) := end of volume block .
                              |
                              |
   rememberdate               |remember date :
                              |   header date (dir index) := date .
                              |
                              |ENDPROC save to archive ;
                              |
writefile ...............|PROC write file (TEXT CONST file name, DATASPACE CONST ds) :
                         |
                         |  enable stop ;
                         |  write header (file name, date, file header) ;
                         |  write (ds)
                         |
                         |ENDPROC write file ;
                         |


writeendofvolume ........|PROC write end of volume :
                         |
                         |  disable stop ;
                         |  end of volume block := block number ;
                         |  write header ("", "", end of volume)
                         |
                         |ENDPROC write end of volume ;
                         |


writeheader .............|PROC write header (TEXT CONST name, date, INT CONST header type) :
                         |
                         |  forget (header space) ;
                         |  header space := nilspace ;
                         |  header := header space ;
                         |
                         |  header.name := subtext (name,1,100) ;
                         |  header.date := date ;
                         |  header.type := header type ;
                         |
                         |  write (header space)
                         |
                         |ENDPROC write header ;
                         |


readheader ..............|PROC read header :
                         |
                         |  forget (header space) ;
                         |  header space := nilspace ;
                         |  read (header space, 1, accept read errors) ;
                         |  header := header space .
                         |
                         |ENDPROC read header ;
                         |


simulateheader ..........|PROC simulate header (INT CONST type, TEXT CONST name) :
                         |
                         |  forget (header space) ;
                         |  header space := nilspace ;
                         |  header := header space ;
                         |  header.name := name ;
                         |  header.date := "??.??.??" ;
                         |  header.type := type ;
                         |  header.password := ""
                         |
                         |ENDPROC simulate header ;
                         |
check ....................|PROC check (TEXT CONST name, TASK CONST from) :
                          |
                          |  IF from = archive
                          |    THEN check file
                          |    ELSE errorstop ("Task gibt es nicht")
                          |  FI .
                          |
                          |
   checkfile              |check file :
                          |  access archive ;
                          |  access file (name) ;
                          |  IF file in directory
                          |    THEN position to file ;
                          |         disable stop ;
                          |         check read ;
                          |         IF is error
                          |           THEN clear error; error ("fehlerhaft")
                          |           ELSE last access time := clock (1) ;
                          |                putline ("""" + file name + """ ohne Fehler gelese
                          |         FI
                          |    ELSE error ("gibt es nicht")
                          |  FI .
                          |
                          |
   fileindirectory        |file in directory :  dir index > 0 .
                          |
                          |
   positiontofile         |position to file :
                          |  seek (header block (dir index) + number of header blocks) .
                          |
                          |ENDPROC check ;
                          |

exists ...................|BOOL PROC exists (TEXT CONST name, TASK CONST from) :
                          |
                          |  IF from = archive
                          |    THEN access archive ;
                          |         access file (name) ;
                          |         file in directory
                          |    ELSE FALSE
                          |  FI .
                          |
                          |
   fileindirectory        |file in directory :  dir index > 0 .
                          |
                          |ENDPROC exists ;
                          |

list .....................|PROC list (TASK CONST from) :
                          |
                          |  forget (ds) ;
                          |  ds := nilspace ;
                          |  FILE VAR list file := sequential file (output, ds) ;
                          |  list (list file, from) ;
                          |  modify (list file) ;
                          |  show (list file) ;
                          |  forget (ds) .
                          |
                          |ENDPROC list ;
                          |
list ....................|PROC list (FILE VAR list file, TASK CONST from) :
                         |
                         |  IF from = archive
                         |    THEN disable stop ;
                         |         continue (archive channel) ;
                         |         list archive (list file) ;
                         |         last access time := clock (1) ;
                         |         continue (main channel)
                         |    ELIF from = myself                          (* R. Nolting
                         |         25.10.84 *)
                         |      THEN list(listfile)
                         |      ELSE errorstop ("Task gibt es nicht")
                         |  FI .
                         |
                         |ENDPROC list ;
                         |


listarchive .............|PROC list archive (FILE VAR list file) :
                         |
                         |  enable stop ;
                         |  access archive ;
                         |  open list file ;
                         |  INT VAR file number := 0 ;
                         |  get (directory, file name, file number) ;
                         |  WHILE file number > 0 REP
                         |    generate list line ;
                         |    get (directory, file name, file number)
                         |  PER ;
                         |  IF directory state = read only
                         |    THEN putline (list file, "Lesefehler: Evtl. fehlen Einträge")
                         |  FI ;
                         |  write list head .
                         |
                         |
   openlistfile          |open list file :
                         |  output (list file) ;
                         |  putline (list file, "") .
                         |
                         |
   generatelistline      |generate list line :
                         |  write (list file, header date (file number)) ;
                         |  write (list file, text (file blocks DIV 2, 5)) ;
                         |  write (list file, " K  ") ;
                         |  IF header.name = dummy name
                         |    THEN write (list file, dummy name)
                         |    ELSE write (list file, quote) ;
                         |         write (list file, file name ) ;
                         |         write (list file, quote)
                         |  FI ;
                         |  line (list file) .
                         |
                         |
   fileblocks            |file blocks :
                         |  IF file number < highest entry (directory)
                         |    THEN header block (file number+1) - header block (file number)
                         |    ELSE end of volume block - header block (file number)
                         |  FI .
                         |
  writelisthead           |write list head :
                          |  headline (list file, archive name +
                          |           " (" + used + " K belegt von " + text (archive size DI
                          |               2) +" K)") .
                          |
                          |
  used                    |used :  text ((end of volume block + 3) DIV 2) .
                          |
                          |ENDPROC list archive ;
                          |


ALL ......................|THESAURUS OP ALL (TASK CONST from) :
                          |
                          |  IF from = myself
                          |    THEN all
                          |  ELIF from = archive
                          |    THEN disable stop ;
                          |         continue (archive channel) ;
                          |         get all from archive ;
                          |         last access time := clock (1) ;
                          |         continue (main channel) ;
                          |         enable stop ;
                          |         all names
                          |  ELSE   errorstop ("Task gibt es nicht") ;
                          |         empty thesaurus
                          |  FI .
                          |
                          |ENDOP ALL ;
                          |


getallfromarchive ........|PROC get all from archive :
                          |
                          |  enable stop ;
                          |  access archive ;
                          |  all names := directory ;
                          |  WHILE all names CONTAINS dummy name REP
                          |    delete (all names, dummy name, dir index)
                          |  PER .
                          |
                          |ENDPROC get all from archive ;
                          |


clear ....................|PROC clear (TASK CONST dest) :
                          |
                          |  IF dest = archive
                          |    THEN disable stop ;
                          |         continue (archive channel) ;
                          |         clear archive ;
                          |         continue (main channel)
                          |    ELSE errorstop ("Task gibt es nicht")
                          |  FI .
                          |
                          |ENDPROC clear ;
                          |


cleararchive .............|PROC clear archive :
                          |
                          |  archive size := archive blocks ;
                          | ask for erase all ;
                          | directory state := invalid ;
                          | rewind ;
                          | write header (archive name, text (clock(1),13,1), start of volume)
                          | write end of volume .
                          |
                          |
     askforeraseall       |ask for erase all :
                          | rewind ;
                          | disable stop ;
                          | read header ;
                          | IF is error OR
                          |    LENGTH header.name < 0 OR LENGTH header.name > 100 OR is error
                          |   THEN header.name := "" ;
                          |        clear error
                          | FI ;
                          | enable stop ;
                          | continue (main channel) ;
                          | IF header.name <> ""
                          |   THEN IF NOT yes ("archiv """+header.name+""" loeschen")
                          |          THEN LEAVE clear archive
                          |        FI
                          |   ELSE IF NOT yes ("archiv initialisieren")
                          |          THEN LEAVE clear archive
                          |        FI
                          | FI ;
                          | continue (archive channel) .
                          |
                          |ENDPROC clear archive ;
                          |


   format ................|PROC format (INT CONST format code, TASK CONST dest) :
                          |
                          | IF dest = archive
                          |   THEN IF yes (""7"Formatieren ueberschreibt alles! Richtige
+                         |        Diskette eingelegt")
                          |          THEN disable stop ;
                          |               continue (archive channel) ;
                          |               format archive (format code) ;
                          |               directory state := invalid ;
                          |               rewind ;
                          |               write header ( archive name, text (clock (1), 13, 1)
+                         |                    ,start of volume) ;
                          |               write end of volume ;
                          |               continue (main channel)
                          |        FI
                          |   ELSE errorstop ("Task gibt es nicht")
                          | FI .
                          |
                          |ENDPROC format ;
                          |


   format ................|PROC format (TASK CONST dest) :
                          |
                          | format (0, dest)
                          |
                          |ENDPROC format ;
                          |
error ....................|PROC error (TEXT CONST error msg) :
                          |
                          |  errorstop ("""" + file name + """ " + error msg)
                          |
                          |ENDPROC error ;
                          |
                          |ENDPACKET archive single ;
```

```
                               |(* ------------------- VERSION 4     22.04.86 ------------------- *)
konfigurieren *************|PACKET konfigurieren DEFINES                  (* Autor: D.Heinrichs *
                               |
                               |
                               |
                               |   ansi cursor,
                               |   baudrate ,
                               |   bits ,
                               |   cursor logic ,
                               |   elbit cursor ,
                               |   enter incode ,
                               |   enter outcode ,
                               |   flow ,
                               |   input buffer size ,
                               |   link ,
                               |   new configuration ,
                               |   new type ,
                               |   ysize :
                               |
                               |LET max dtype nr = 5,  (* maximum number of active device tables *)
                               |   device table = 32000,
                               |   ack = 0 ;
                               |
                               |
                               |INT VAR next outstring,
                               |       next instring;
                               |
                               |BOUND STRUCT (ALIGN space,                          (*
                               |    umsetzcodetabelle *)
                               |            ROW 128 INT outcodes,
                               |            ROW  64 INT outstrings,
                               |            ROW  64 INT instrings) VAR x;
                               |
                               |
                               |ROW max dtype nr DATASPACE VAR device code table;
                               |
                               |THESAURUS VAR dtypes ;
                               |
                               |


newconfiguration .........|PROC new configuration :
                               |
                               |  dtypes := empty thesaurus ;
                               |  INT VAR i ;
                               |  insert (dtypes, "psi", i) ;
                               |  insert (dtypes, "transparent", i) ;
                               |  FOR i FROM 1 UPTO max dtype nr REP
                               |    forget (device code table (i))
                               |  PER .
                               |
                               |ENDPROC new configuration ;
                               |
                               |


blockout ..................|PROC block out (DATASPACE CONST ds, INT CONST page, code):
                               |  INT VAR err;
                               |  block out (ds,page,0,code,err);
                               |  announce error (err)
                               |END PROC block out;
                               |
announceerror ...........|PROC announce error (INT CONST err):
                         |  SELECT err OF
                         |  CASE 0:
                         |  CASE 1: errorstop ("unbekanntes Terminalkommando")
                         |  CASE 2: errorstop ("Nummer der Terminal-Typ-Tabelle falsch")
                         |  CASE 3: errorstop ("falsche Terminalnummer")
                         |  OTHERWISE errorstop ("blockout: unzulaessiger Kanal")
                         |  ENDSELECT
                         |END PROC announce error;
                         |

flow ....................|PROC flow (INT CONST nr, INT CONST dtype):
                         |  control (6, dtype, nr)
                         |END PROC flow;
                         |

ysize ...................|PROC ysize (INT CONST channel ,new size, INT VAR old size) :
                         |  control (11, channel, new size, old size)
                         |ENDPROC ysize ;
                         |

inputbuffersize .........|PROC input buffer size (INT CONST nr,size):
                         |  INT VAR err;
                         |  control (2,nr,size,err)
                         |END PROC input buffer size;
                         |

baudrate ................|PROC baudrate (INT CONST nr, rate) :
                         |  control (8, rate, nr)
                         |ENDPROC baudrate ;
                         |

bits ....................|PROC bits (INT CONST channel, number, parity) :
                         |  bits (channel, number-1 + 8*parity)
                         |ENDPROC bits ;
                         |

bits ....................|PROC bits (INT CONST channel, key) :
                         |  control (9, key, channel)
                         |ENDPROC bits ;
                         |

control .................|PROC control (INT CONST function, key, channel) :
                         |
                         |  INT VAR err ;
                         |  IF key > -128 AND key < 127
                         |    THEN control (function, channel, key, err)
                         |  ELIF key = -128
                         |    THEN control (function, channel, -maxint-1, err)
                         |  FI
                         |
                         |ENDPROC control ;
                         |
                         |
newtype .................|PROC new type (TEXT CONST dtype):
                         |  x := new (dtype);
                         |  type (old (dtype), device table);
                         |  next outstring := 4;
                         |  next instring := 0;
                         |  INT VAR i;
                         |  (* Defaults, damit trmpret den cursor mitfuehrt: *)
                         |    FOR i FROM 1 UPTO 6 REP
                         |    enter outcode (i,i)
                         |    PER;
                         |    enter outcode (8,8);
                         |    enter outcode (10,10);
                         |    enter outcode (13,13);
                         |    enter outcode (14,126);
                         |    enter outcode (15,126);
                         |END PROC new type;
                         |

activatedtype ...........|INT PROC activate dtype (TEXT CONST dtype):
                         |
                         | INT VAR i := link (dtypes, dtype);
                         | IF (exists (dtype) CAND type (old (dtype)) = device table)
                         |    THEN IF i <= 0
                         |            THEN insert (dtypes, dtype, i);
                         |         FI;
                         |         forget(device code table (i-2));
                         |         device code table (i-2) := old (dtype)
                         | FI;
                         | IF i > max dtype nr +2  (* 5 neue Typen erlaubt *)
                         |    THEN delete (dtypes,i);
                         |         error stop ("Anzahl Terminaltypen > "+text (i));0
                         |    ELIF i <= 0
                         |         THEN error stop ("Unbekannter Terminaltyp" + dtype); 0
                         |    ELSE i
                         | FI.
                         |
                         |END PROC activate dtype;
                         |

link ....................|PROC link (INT CONST nr, TEXT CONST dtype):
                         |
                         |  INT VAR lst nr := activate dtype (dtype)-3;
                         |  IF lst nr < 0
                         |     THEN lst nr INCR 256  (* fuer std terminal und std device *)
                         |     ELSE blockout (device code table(lst nr+1), 2, lst nr);
                         |  FI;
                         |  INT VAR err := 0;
                         |  control (1,nr,lst nr,err) ;
                         |  announce error(err)
                         |
                         |END PROC link;
                         |
                         |

enteroutcode ............|PROC enter outcode (INT CONST eumel code, ziel code):
                         |
                         |  IF ziel code < 128
                         |    THEN simple entry (eumel code, ziel code)
                         |    ELSE enter outcode (eumel code, 0, code (ziel code))
```

```
                             |  FI .
                             |
                             |ENDPROC enter outcode ;
                             |

simpleentry .................|PROC simple entry (INT CONST eumel code, ziel code) :
                             |
                             |  INT CONST position := eumel code DIV 2 +1,
                             |            teil := eumel code - 2*position + 2;
                             |  TEXT VAR h :="    ";
                             |  replace (h,1,out word);
                             |  replace (h,1+teil,code (ziel code));
                             |  out word := (h ISUB 1).
                             |
                             |
   outword                   |  out word: x.outcodes (position).
                             |
                             |END PROC simple entry ;
                             |

enteroutcode ................|PROC enter outcode (INT CONST eumel code, wartezeit,
                             |                    TEXT CONST sequenz):
                             |
                             |     INT VAR i;
                             |     simple entry (eumel code, next outstring + 128);
                             |     enter part (x.outstrings, next outstring, wartezeit);
                             |     FOR i FROM 1 UPTO length (sequenz) REP
                             |       enter part (x.outstrings, next outstring + i, code
                             |            (sequenzSUBi))
                             |     PER;
                             |     next outstring INCR length (sequenz)+2;
                             |     abschluss.
                             |
                             |
   abschluss                 |     abschluss:
                             |     enter part (x.outstrings, next outstring-1, 0)
                             |END PROC enter outcode;
                             |

enteroutcode ................|PROC enter outcode (INT CONST eumelcode, TEXT CONST wert):
                             |  enter outcode (eumelcode,code(wert))
                             |END PROC enter outcode;
                             |

enterpart ...................|PROC enter part (ROW 64 INT VAR a,INT CONST index, wert):
                             |  INT CONST position := index DIV 2 +1,
                             |            teil := index - 2*position + 2;
                             |  IF position > 64 THEN errorstop ("Ueberlauf der
                             |       Terminaltyptabelle") FI;
                             |  TEXT VAR h :="    ";
                             |  replace (h,1,out word);
                             |  replace (h,1+teil,code (wert));
                             |  out word := (h ISUB 1).
                             |
                             |
   outword                   |  out word: a (position).
                             |END PROC enter part;
                             |
                                 |

enterincode .............|PROC enter incode (INT CONST elan code, TEXT CONST sequenz):
                         |     IF elan code > 254 OR elan code < 0 THEN errorstop ("kein
                         |          Eingabecode")
                         |     ELSE
                         |     INT VAR i;
                         |     enter part (x.instrings, next instring, elan code);
                         |     FOR i FROM 1 UPTO length (sequenz) REP
                         |       enter part (x.instrings, next instring + i, code
                         |            (sequenzSUBi))
                         |     PER;
                         |     next instring INCR length (sequenz)+2;
                         |
                         |     FI
                         |
                         |END PROC enter incode;
                         |

cursorlogic .............|PROC cursor logic (INT CONST dist, TEXT CONST pre, mid, post):
                         |
                         |     cursor logic (dist,255,pre,mid,post)
                         |
                         |END PROC cursor logic;
                         |

ansicursor ..............|PROC ansi cursor (TEXT CONST pre, mid, post):
                         |
                         |     cursor logic (0, 1, pre, mid, post)
                         |
                         |END PROC ansi cursor;
                         |

cursorlogic .............|PROC cursor logic (INT CONST dist, modus, TEXT CONST pre, mid, post)
                         |
                         |     enter part (x.outstrings,2,dist);
                         |     enter part (x.outstrings,3,dist);
                         |     enter part (x.outstrings,0,modus);
                         |     enter part (x.outstrings,1,modus);
                         |     enter outcode (6,0,pre+""0"y"+mid+""0"x"+post+""0"")
                         |
                         |END PROC cursor logic;
                         |

elbitcursor .............|PROC elbit cursor:
                         |     cursor logic (0,""27"","","");
                         |     enter part (x.outstrings,0,2);
                         |     enter part (x.outstrings,1,255);
                         |END PROC elbit cursor;
                         |
                         |ENDPACKET konfigurieren;
```

```
                                |(* ------------------- VERSION 11    10.06.86 ------------------- *)
configuratorsingle *******      |PACKET configurator single DEFINES
                                |
                                |       configurate ,
                                |       exec configuration ,
                                |       setup :
                                |
                                |LET baudrates      = ""1"50"2"75"3"110"4"134.5"5"150"6"300"7"600
                                |"8"1200"9"1800"10"2400"11"3600"12"4800"13"7200
                                |"14"9600"15"19200"16"38400"17"",
                                |    parities       = ""0"no"1"odd"2"even"3"" ,
                                |    bits per char  = ""0"1"1"2"2"3"3"4"4"5"5"6"6"7"7"8"8"" ,
                                |    stopbits       = ""0"1"1"1.5"2"2"3"" ,
                                |    flow modes     = ""0"ohne Protokoll"1"XON/XOFF"2"RTS/CTS
                                |"3""4""5"XON/XOFF - ausgabeseitig"6"RTS/CTS - ausgabeseitig"7""8"
                                |"9"XON/XOFF - eingabeseitig"10"RTS/CTS - eingabeseitig"11"" ,
                                |
                                |    ok             = "j" ,
                                |    esc            = ""27"" ,
                                |    cr             = ""13"" ,
                                |    right          = ""2"" ,
                                |
                                |    psi            = "psi" ,
                                |    transparent    = "transparent" ,
                                |
                                |    std rate          = 14 ,
                                |    std bits          = 22 ,
                                |    std flow          = 0 ,
                                |    std inbuffer size = 16 ,
                                |
                                |    device table      = 32000 ,
                                |
                                |    max edit terminal      = 15 ,
                                |    configuration channel  = 32 ,
                                |
                                |    CONF = STRUCT (TEXT dev type,
                                |                   INT baud, bits par stop, flow control, inbuffer
                                |                        size) ;
                                |
                                |
                                |BOUND ROW max edit terminal CONF VAR conf ;
                                |
                                |INT VAR channel no ;
                                |
                                |TEXT VAR prelude , last feature , answer ;
                                |
                                |
                                |


shardpermits ............       |BOOL PROC shard permits (INT CONST code, key) :
                                |
                                |  INT VAR reply ;
                                |  IF key > -128
                                |    THEN control (code, channel no, key, reply)
                                |    ELSE control (code, channel no, -maxint-1, reply)
                                |  FI ;
                                |  reply = 0 .
                                |
                                |ENDPROC shard permits ;
                                |
askuser ..................|PROC ask user (TEXT CONST feature, question) :
                          |
                          |  last feature := feature ;
                          |  put question ;
                          |  skip pretyped chars ;
                          |  get valid answer .
                          |
                          |
   putquestion            |put question :
                          |  clear line ;
                          |  out (prelude) ;
                          |  out (feature) ;
                          |  out (question) ;
                          |  out (" (j/n) ") .
                          |
                          |
   clearline              |clear line :
                          |  out (cr) ;
                          |  79 TIMESOUT " " ;
                          |  out (cr) .
                          |
                          |
   skippretypedchars      |skip pretyped chars :
                          |  REP UNTIL incharety = "" PER .
                          |
                          |
   getvalidanswer         |get valid answer :
                          |  REP
                          |    inchar (answer)
                          |  UNTIL pos ("jJyYnN"27"", answer) > 0 PER ;
                          |  IF answer > ""31""
                          |    THEN out (answer)
                          |  FI ;
                          |  out (cr) ;
                          |  normalize answer .
                          |
                          |
   normalizeanswer        |normalize answer :
                          |  IF pos ("jJyY", answer) > 0
                          |    THEN answer := ok
                          |  FI .
                          |
                          |ENDPROC ask user ;
                          |


yes ......................|BOOL PROC yes (TEXT CONST question) :
                          |
                          |  ask user ("", question) ;
                          |  answer = ok
                          |
                          |ENDPROC yes ;
                          |


chosekey .................|PROC chose key (INT VAR old key, INT CONST max key, TEXT CONST key
 +                        |     string,
                          |                key entity, BOOL PROC (INT CONST) shard permits):
                          |
                          |  IF shard permits at least one standard key
                          |    THEN try all keys
                          |  FI .
```

```
                         |
                         |
shardpermitsatleastone   |shard permits at least one standard key :
                         |  INT VAR key ;
                         |  FOR key FROM 0 UPTO max key REP
                         |    IF shard permits (key)
                         |      THEN LEAVE shard permits at least one standard key WITH TRUE
                         |    FI
                         |  PER ;
                         |  FALSE .
                         |
                         |
tryallkeys               |try all keys :
                         |  key := old key ;
                         |  REP
                         |    examine this key ;
                         |    next key
                         |  PER .
                         |
                         |
examinethiskey           |examine this key :
                         |  IF shard permits (key) CAND key value <> ""
                         |    THEN ask user (key value, key entity) ;
                         |         IF answer = ok
                         |           THEN chose this key
                         |         ELIF answer = esc
                         |           THEN key := -129
                         |         FI
                         |  FI .
                         |
                         |
keyvalue                 |key value :
                         |  IF key >= 0
                         |    THEN subtext (key string, key pos + 1, next key pos - 1)
                         |    ELSE text (key)
                         |  FI .
                         |
                         |
keypos                   |key pos      : pos (key string, code (key)) .
                         |
nextkeypos               |next key pos  : pos (key string, code (key+1)) .
                         |
                         |
chosethiskey             |chose this key :
                         |  remember calibration ;
                         |  old key := key ;
                         |  LEAVE chose key .
                         |
                         |
nextkey                  |next key :
                         |  IF key < max key
                         |    THEN key INCR 1
                         |    ELSE key := 0
                         |  FI .
                         |
                         |
remembercalibration      |remember calibration :
                         |  prelude CAT last feature ;
                         |  prelude CAT ", " .
                         |
                         |ENDPROC chose key ;
                         |
rateok ..................|BOOL PROC rate ok (INT CONST key) :
                         |
                         |  shard permits (8, key)
                         |
                         |ENDPROC rate ok ;
                         |


bitsok ..................|BOOL PROC bits ok (INT CONST key) :
                         |
                         |  IF key < 0
                         |    THEN shard permits (9, key)
                         |    ELSE some standard combination ok
                         |  FI .
                         |
                         |
   somestandardcombinatio |some standard combination ok :
                         |  INT VAR combined := key ;
                         |  REP
                         |    IF shard permits (9, combined)
                         |      THEN LEAVE bits ok WITH TRUE
                         |    FI ;
                         |    combined INCR 8
                         |  UNTIL combined > 127 PER ;
                         |  FALSE
                         |
                         |ENDPROC bits ok ;
                         |


parityok ................|BOOL PROC parity ok (INT CONST key) :
                         |
                         |  INT VAR combined := 8 * key + data bits ;
                         |  key >= 0 AND (shard permits (9, combined)      OR
                         |                shard permits (9, combined + 32) OR
                         |                shard permits (9, combined + 64)    )
                         |
                         |ENDPROC parity ok ;
                         |


stopbitsok ..............|BOOL PROC stopbits ok (INT CONST key) :
                         |
                         |  key >= 0 AND shard permits (9, 32 * key + 8 * parity + data bits)
                         |
                         |ENDPROC stopbits ok ;
                         |


flowmodeok ..............|BOOL PROC flow mode ok (INT CONST key) :
                         |
                         |  shard permits (6, key)
                         |
                         |ENDPROC flow mode ok ;
                         |
                         |
                         |
                         |INT VAR operators channel ,
                         |        data bits ,
                         |        parity ,
                         |        stop ;
                         |
                          |TEXT VAR table name, dummy ;
                          |
                          |

configurate ..............|PROC configurate :
                          |
                          |  new configuration ;
                          |  access configuration table ;
                          |  show all device types ;
                          |  channel no := 1 ;
                          |  REP
                          |    IF channel hardware exists
                          |      THEN try this channel ;
                          |           setup this channel
                          |    FI ;
                          |    channel no INCR 1
                          |  UNTIL channel no > 15 PER ;
                          |  prelude := "" ;
                          |  IF yes ("Koennen unbenutzte Geraetetypen geloescht werden")
                          |    THEN forget unused device tables
                          |  FI .
                          |
                          |
   accessconfigurationtab |access configuration table :
                          |  IF exists ("configuration")
                          |    THEN conf := old ("configuration")
                          |    ELSE conf := new ("configuration") ;
                          |         initialize configuration
                          |  FI .
                          |
                          |
   initializeconfiguratio |initialize configuration :
                          |  FOR channel no FROM 1 UPTO max edit terminal REP
                          |    conf (channel no) :=
                          |    CONF:(transparent, std rate, std bits, std flow, std inbuffer
                          |         size)
                          |  PER ;
                          |  conf (1).dev type := psi .
                          |
                          |
   showalldevicetypes     |show all device types :
                          |  show prelude ;
                          |  begin list ;
                          |  get list entry (table name, dummy) ;
                          |  WHILE table name <> "" REP
                          |    IF dataspace is device table
                          |      THEN show table name
                          |    FI ;
                          |    get list entry (table name, dummy)
                          |  PER ;
                          |  line (2) .
                          |
                          |
   showprelude            |show prelude :
                          |  line (30) ;
                          |  outtext (psi, 1, 20) ;
                          |  outtext (transparent, 1, 20) .
                          |
dataspaceisdevicetable |dataspace is device table :
                       |  type (old (table name)) = device table .
                       |
                       |
showtablename          |show table name :
                       |  outtext (table name, 1, 20) .
                       |
                       |
trythischannel         |try this channel :
                       |  prelude := "Kanal " ;
                       |  ask user ("", text (channel no)) ;
                       |  IF answer = ok
                       |    THEN prelude CAT text (channel no) + ": " ;
                       |         get configuration from user (conf (channel no)) ;
                       |         line
                       |  FI .
                       |
                       |
channelhardwareexists  |channel hardware exists :
                       |  operators channel := channel ;
                       |  INT VAR channel type ;
                       |  disable stop ;
                       |  continue (channel no) ;
                       |  IF is error
                       |    THEN IF error message = "kein Kanal"
                       |           THEN channel type := 0
                       |           ELSE channel type := inout mask
                       |         FI
                       |    ELSE get channel type from shard
                       |  FI ;
                       |  clear error ;
                       |  disable stop ;
                       |  continue operators channel ;
                       |  (channel type AND inout mask) <> 0 .
                       |
                       |
getchanneltypefromshar |get channel type from shard :
                       |  control (1, 0, 0, channel type) .
                       |
                       |
inoutmask              |inout mask : 3 .
                       |
                       |
forgetunuseddevicetabl |forget unused device tables :
                       |  begin list ;
                       |  get list entry (table name, dummy) ;
                       |  WHILE table name <> "" REP
                       |    IF type (old (table name)) = device table
                       |      THEN forget if unused
                       |    FI ;
                       |    get list entry (table name, dummy)
                       |  PER .
                       |
                       |
forgetifunused         |forget if unused :
                       |  FOR channel no FROM 1 UPTO max edit terminal REP
                       |    IF conf (channel no).dev type = table name
                       |      THEN LEAVE forget if unused
                       |    FI
                       |  PER ;
                       |  forget (table name, quiet) .
                       |
   setupthischannel       |setup this channel :
                          |  operators channel := channel ;
                          |  disable stop ;
                          |  continue (configuration channel) ;
                          |  set up channel (channel no, conf (channel no)) ;
                          |  continue operators channel .
                          |
                          |
   continueoperatorschann |continue operators channel :
                          |  continue (operators channel) ;
                          |  IF is error
                          |    THEN clear error ;
                          |         LEAVE configurate
                          |  FI ;
                          |  enable stop .
                          |
                          |ENDPROC configurate ;
                          |


getconfigurationfromus ...|PROC get configuration from user (CONF VAR conf) :
                          |
                          |  get device type ;
                          |  get baud rate ;
                          |  get bits and parity and stopbits ;
                          |  get protocol ;
                          |  get buffer size .
                          |
                          |
                          |
   getdevicetype          |get device type :
                          |  begin list ;
                          |  table name := conf.dev type ;
                          |  IF NOT is valid device type
                          |    THEN next device type
                          |  FI ;
                          |  REP
                          |    IF NOT (table name = transparent AND channel no = 1)
                          |      THEN ask user ("", table name) ;
                          |           IF answer = ok COR was esc followed by type table name
                          |             THEN IF is valid device type
                          |                    THEN remember device type ;
                          |                         LEAVE get device type
                          |                    ELSE out (""7" unbekannter Typ"); pause (20)
                          |                  FI
                          |           FI
                          |    FI ;
                          |    next device type
                          |  PER .
                          |
                          |
   wasescfollowedbytypeta |was esc followed by type table name :
                          |  IF answer = esc
                          |    THEN 9 TIMESOUT right ;
                          |         put ("Typ:") ;
                          |         editget (table name) ;
                          |         TRUE
                          |    ELSE FALSE
                          |  FI .
                          |
isvaliddevicetype       |is valid device type :
                        |  table name = psi OR table name = transparent OR
                        |  (exists (table name) CAND type (old (table name)) = device table)
                        |
                        |
rememberdevicetype      |remember device type :
                        |  prelude CAT table name ;
                        |  conf.dev type := table name ;
                        |  prelude CAT ", " .
                        |
                        |
nextdevicetype          |next device type :
                        |  IF table name = psi
                        |    THEN table name := transparent
                        |    ELSE IF table name = transparent
                        |           THEN begin list
                        |         FI ;
                        |         search next device type space
                        |  FI .
                        |
                        |
searchnextdevicetypesp  |search next device type space :
                        |  REP
                        |    get list entry (table name, dummy)
                        |  UNTIL table name = "" COR type (old (table name)) = device table
                        |      PER;
                        |  IF table name = ""
                        |    THEN table name := psi
                        |  FI .
                        |
                        |
getbaudrate             |get baudrate :
                        |  chose key (conf.baud, 16, baudrates, " Baud", PROC rate ok) .
                        |
                        |
getbitsandparityandsto  |get bits and parity and stopbits :
                        |  data bits := conf.bits par stop MOD 8 ;
                        |  parity := (conf.bits par stop DIV 8) MOD 4 ;
                        |  stop := (conf.bits par stop DIV 32) MOD 4 ;
                        |  chose key (data bits, 7, bits per char, " Bits", PROC bits ok) ;
                        |  IF data bits >= 0
                        |    THEN chose key (parity, 2, parities, " parity", PROC parity ok)
                        |         chose key (stop, 2, stopbits, " Stopbits", PROC stopbits
                        |              ok);
                        |         conf.bits par stop := data bits + 8 * parity + 32 * stop
                        |    ELSE conf.bits par stop := data bits
                        |  FI .
                        |
                        |
getprotocol             |get protocol :
                        |  chose key (conf.flow control, 10, flow modes,
                        |             "", PROC flow mode ok) .
                        |
                        |
getbuffersize           |get buffer size :
                        |  IF dev type is transparent
                        |    THEN chose buffer size
                        |    ELSE conf.inbuffer size := std inbuffer size
                        |  FI .
                        |
   devtypeistransparent   |dev type is transparent :
                          |  conf.dev type = "transparent" .
                          |
                          |
   chosebuffersize        |chose buffer size :
                          |  REP
                          |    IF conf.inbuffer size = 16 CAND yes ("normaler Puffer")
                          |      THEN LEAVE chose buffer size
                          |    FI ;
                          |    conf.inbuffer size := 512 ;
                          |    IF yes ("grosser Puffer")
                          |      THEN LEAVE chose buffer size
                          |    FI ;
                          |    conf.inbuffer size := 16
                          |  PER .
                          |
                          |ENDPROC get configuration from user ;
                          |


execconfiguration ........|PROC exec configuration :
                          |
                          |  setup
                          |
                          |ENDPROC exec configuration ;
                          |


setup ....................|PROC setup :
                          |
                          |  conf := old ("configuration") ;
                          |  continue (configuration channel) ;
                          |  FOR channel no FROM 1 UPTO max edit terminal REP
                          |    set up channel (channel no, conf (channel no))
                          |  PER ;
                          |  continue (operators channel) .
                          |
                          |ENDPROC set up ;
                          |


setupchannel .............|PROC set up channel (INT CONST channel no, CONF CONST conf) :
                          |
                          |  link (channel no, conf.dev type) ;
                          |  baudrate (channel no, conf.baud) ;
                          |  bits (channel no, conf.bits par stop) ;
                          |  flow (channel no, conf.flow control) ;
                          |  input buffer size (channel no, conf.inbuffer size) .
                          |
                          |ENDPROC setup channel ;
                          |
                          |ENDPACKET configurator single ;
                          |
```