Graph2D Library --- DOS ---

Generated by Doxygen 1.8.19

Chapter 1

Graph2D / Plot10 & AG II- DOS Port

Graphics Driver for DOS

The library was developed with the Microsoft FTN-77 compiler and the MASM assembler, based on the CP/M version. In the beginning the basic graphics library graphics.lib, which was part of the MS compiler package, was used . Later, the system was ported to the free Open Watcom compiler/assembler and its graph.lib library. To keep the ability to use the MS-compiler, the include files fgraph.fd and fgraph.fi adapt the correspondent procedure calls to the Watcom library.

How to build the library:
Copy the sources to the /build subdirectory by running "$getfiles.bat DOS" and use the Watcom workspace files.

How to use the library:
After building the library and linking it to the applications, the main features could be changed by the following files:
graphlib.fon: Fontfile for the graphic text
graphlib.lng: Translations of the messages

Hardcopies are created as standard $*$.bmp-files.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 AG2.for File Reference

Graph2D: Tektronix Advanced Graphing II Emulation.

Functions/Subroutines

- subroutine ag2lev (ilevel)
- subroutine line (ipar)
- subroutine symbl (ipar)
- subroutine steps (ipar)
- subroutine infin (par)
- real function ag2infin ()
- subroutine npts (ipar)
- subroutine stepl (ipar)
- subroutine sizes (par)
- subroutine sizel (par)
- subroutine xneat (ipar)
- subroutine yneat (ipar)
- subroutine xzero (ipar)
- subroutine yzero (ipar)
- subroutine xloc (ipar)
- subroutine yloc (ipar)
- subroutine xloctp (ipar)
- subroutine ylocrt (ipar)
- subroutine xlab (ipar)
- subroutine ylab (ipar)
- subroutine xden (ipar)
- subroutine yden (ipar)
- subroutine xtics (ipar)
- subroutine ytics (ipar)
- subroutine xlen (ipar)
- subroutine ylen (ipar)
- subroutine xfrm (ipar)
- subroutine yfrm (ipar)
- subroutine xmtcs (ipar)
- subroutine ymtcs (ipar)

- subroutine xmfrm (ipar)
- subroutine ymfrm (ipar)
- subroutine dlimx (xmin, xmax)
- subroutine dlimy (ymin, ymax)
- subroutine slimx (ixmin, ixmax)
- subroutine slimy (iymin, iymax)
- subroutine place (ipar)
- subroutine xtype (ipar)
- subroutine ytype (ipar)
- subroutine xwdth (ipar)
- subroutine ywdth (ipar)
- subroutine xetyp (ipar)
- subroutine yetyp (ipar)
- subroutine setwin
- subroutine dinitx
- subroutine dinity
- subroutine hbarst (ishade, iwbar, idbar)
- subroutine vbarst (ishade, iwbar, idbar)
- subroutine binitt
- subroutine check (x, y)
- subroutine typck (ixy, arr)
- subroutine rgchek (ixy, arr)
- subroutine mnmx (arr, amin, amax)
- subroutine cmnmx (arr, amin, amax)
- subroutine optim (ixy)
- subroutine loptim (ixy)
- subroutine coptim (ixy)
- real function calpnt (arr, i)
- subroutine calcon (amin, amax, labtyp, ubgc)
- subroutine ymdyd (iJulYrOut, iJulDayOut, iGregYrIn, iGregMonIn, iGregDayIn)
- integer function leap (iyear)
- subroutine iubgc (iyear, iday, iubgcO)
- subroutine oubgc (iyear, iday, iubgcI)
- subroutine frame
- subroutine dsplay (x, y)
- subroutine cplot (x, y)
- subroutine keyset (array, key)
- real function datget (arr, i, key)
- subroutine bar (x, y, line)
- subroutine filbox (minx, miny, maxx, maxy, ishade, lspace)
- subroutine bsyms (x, y, isym)
- subroutine symout (isym, fac)
- subroutine teksym (isym, amult)
- subroutine teksym1 (istart, iend, incr, siz)
- subroutine grid
- subroutine logtix (nbase, start, tintvl, mstart, mend)
- subroutine tset (nbase)
- subroutine tset2 (newloc, nfar, nlen, nfrm, kstart, kend)
- subroutine monpos (nbase, iy1, dpos, spos)
- subroutine gline (nbase, datapt, spos)
- subroutine label (nbase)
- subroutine numsetc (fnum, iwidth, nbase, outstr)
- subroutine iformc (fnum, iwidth, outstr)
- subroutine fformc (fnum, iwidth, idec, outstr)
- subroutine fonlyc (fnum, iwidth, idec, outstr)

- subroutine eformc (fnum, iwidth, idec, outstr)
- subroutine esplit (fnum, iwidth, idec, iexpon)
- subroutine expoutc (nbase, iexp, outstr)
- subroutine alfsetc (fnum, labtyp, string)
- subroutine notatec (ix, iy, string)
- subroutine vlablc (string)
- subroutine justerc (string, iPosFlag, iOff)
- subroutine width (nbase)
- subroutine lwidth (nbase)
- subroutine remlab (nbase, iloc, labtyp, ix, iy)
- subroutine spread (nbase)
- real function findge (val, tab, iN)
- real function findle (val, tab, iN)
- integer function locge (ival, itab, iN)
- integer function locle (ival, itab, iN)
- real function roundd (value, finterval)
- real function roundu (value, finterval)
- subroutine savcom (Array)
- subroutine rescom (Array)
- integer function iother (ipar)

3.1.1 Detailed Description

Graph2D: Tektronix Advanced Graphing II Emulation.

Version

(2025,347, x)

Author

(C) 2025 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Layer 2: scientific 2-D graphic subroutines

Note

The control character for exponent (originally -1) is now SOH=char(1) and for index (originally -2) STX=char(2).

```
Package:
 - AG2.for:         chart plotting routines
 - AG2Holerith.for: deprecated routines
 - AG2USR.for:      default userroutines
 - G2dAG2.fd:       commonblock
```

Definition in file AG2.for.

3.1.2 Function/Subroutine Documentation

3.1.2.1 ag2infin()

```
real function ag2infin
```

Definition at line 155 of file AG2.for.

3.1.2.2 ag2lev()

```
subroutine ag2lev (
             integer, dimension(3) ilevel )
```

Definition at line 94 of file AG2.for.

3.1.2.3 alfsetc()

```
subroutine alfsetc (
             real fnum,
             integer labtyp,
             character *(*) string )
```

Definition at line 2574 of file AG2.for.

3.1.2.4 bar()

```
subroutine bar (
             real x,
             real y,
             integer line )
```

Definition at line 1698 of file AG2.for.

3.1.2.5 binitt()

```
subroutine binitt
```

Definition at line 724 of file AG2.for.

3.1.2.6 bsyms()

```
subroutine bsyms (
             real x,
             real y,
             integer isym )
```

Definition at line 1850 of file AG2.for.

3.1.2.7 calcon()

```
subroutine calcon (
             real amin,
             real amax,
             integer labtyp,
             logical ubgc )
```

Definition at line 1336 of file AG2.for.

3.1.2.8 calpnt()

```
real function calpnt (
             real, dimension(5) arr,
             integer i )
```

Definition at line 1281 of file AG2.for.

3.1.2.9 check()

```
subroutine check (
             real, dimension(5) x,
             real, dimension(5) y )
```

Definition at line 808 of file AG2.for.

3.1.2.10 cmnmx()

```
subroutine cmnmx (
             real, dimension(5) arr,
             real amin,
             real amax )
```

Definition at line 930 of file AG2.for.

3.1.2.11 coptim()

```
subroutine coptim (
             integer ixy )
```

Definition at line 1125 of file AG2.for.

3.1.2.12 cplot()

```
subroutine cplot (
             real, dimension(5) x,
             real, dimension(5) y )
```

Definition at line 1548 of file AG2.for.

3.1.2.13 datget()

```
real function datget (
             real, dimension(5) arr,
             integer i,
             integer key )
```

Definition at line 1670 of file AG2.for.

3.1.2.14 dinitx()

```
subroutine dinitx
```

Definition at line 654 of file AG2.for.

3.1.2.15 dinity()

```
subroutine dinity
```

Definition at line 668 of file AG2.for.

3.1.2.16 dlimx()

```
subroutine dlimx (
             real xmin,
             real xmax )
```

Definition at line 474 of file AG2.for.

3.1.2.17 dlimy()

```
subroutine dlimy (
             real ymin,
             real ymax )
```

Definition at line 486 of file AG2.for.

3.1.2.18 dsplay()

```
subroutine dsplay (
             real, dimension(5) x,
             real, dimension(5) y )
```

Definition at line 1534 of file AG2.for.

3.1.2.19 eformc()

```
subroutine eformc (
             real fnum,
             integer iwidth,
             integer idec,
             character, dimension(*) outstr )
```

Definition at line 2445 of file AG2.for.

3.1.2.20 esplit()

```
subroutine esplit (
             real fnum,
             integer iwidth,
             integer idec,
             integer iexpon )
```

Definition at line 2478 of file AG2.for.

3.1.2.21 expoutc()

```
subroutine expoutc (
             integer nbase,
             integer iexp,
             character, dimension(*) outstr )
```

Definition at line 2498 of file AG2.for.

3.1.2.22 fformc()

```
subroutine fformc (
             real fnum,
             integer iwidth,
             integer idec,
             character, dimension(*) outstr )
```

Definition at line 2385 of file AG2.for.

3.1.2.23 filbox()

```
subroutine filbox (
             integer minx,
             integer miny,
             integer maxx,
             integer maxy,
             integer ishade,
             integer lspace )
```

Definition at line 1765 of file AG2.for.

3.1.2.24 findge()

```
real function findge (
             real val,
             real, dimension(1) tab,
             integer iN )
```

Definition at line 2933 of file AG2.for.

3.1.2.25 findle()

```
real function findle (
             real val,
             real, dimension(1) tab,
             integer iN )
```

Definition at line 2952 of file AG2.for.

3.1.2.26 fonlyc()

```
subroutine fonlyc (
             real fnum,
             integer iwidth,
             integer idec,
             character, dimension(*) outstr )
```

Definition at line 2414 of file AG2.for.

3.1.2.27 frame()

```
subroutine frame
```

Definition at line 1520 of file AG2.for.

3.1.2.28 gline()

```
subroutine gline (
             integer nbase,
             real datapt,
             integer spos )
```

Definition at line 2183 of file AG2.for.

3.1.2.29 grid()

```
subroutine grid
```

Definition at line 1966 of file AG2.for.

3.1.2.30 hbarst()

```
subroutine hbarst (
             integer ishade,
             integer iwbar,
             integer idbar )
```

Definition at line 682 of file AG2.for.

3.1.2.31 iformc()

```
subroutine iformc (
             real fnum,
             integer iwidth,
             character, dimension(*) outstr )
```

Definition at line 2353 of file AG2.for.

3.1.2.32 infin()

```
subroutine infin (
             real par )
```

Definition at line 142 of file AG2.for.

3.1.2.33 iother()

```
integer function iother (
             integer ipar )
```

Definition at line 3077 of file AG2.for.

3.1.2.34 iubgc()

```
subroutine iubgc (
             integer iyear,
             integer iday,
             integer iubgc0 )
```

Definition at line 1483 of file AG2.for.

3.1.2.35 justerc()

```
subroutine justerc (
             character, dimension(*) string,
             integer iPosFlag,
             integer iOff )
```

Definition at line 2677 of file AG2.for.

3.1.2.36 keyset()

```
subroutine keyset (
             real, dimension(1) array,
             integer key )
```

Definition at line 1644 of file AG2.for.

3.1.2.37 label()

```
subroutine label (
             integer nbase )
```

Definition at line 2210 of file AG2.for.

3.1.2.38 leap()

```
integer function leap (
             integer iyear )
```

Definition at line 1469 of file AG2.for.

3.1.2.39 line()

```
subroutine line (
             integer ipar )
```

Definition at line 109 of file AG2.for.

3.1.2.40 locge()

```
integer function locge (
             integer ival,
             integer, dimension(1) itab,
             integer iN )
```

Definition at line 2974 of file AG2.for.

3.1.2.41 locle()

```
integer function locle (
             integer ival,
             integer, dimension(1) itab,
             integer iN )
```

Definition at line 2992 of file AG2.for.

3.1.2.42 logtix()

```
subroutine logtix (
             integer nbase,
             real start,
             real tintvl,
             integer mstart,
             integer mend )
```

Definition at line 2052 of file AG2.for.

3.1.2.43 loptim()

```
subroutine loptim (
             integer ixy )
```

Definition at line 998 of file AG2.for.

3.1.2.44 lwidth()

```
subroutine lwidth (
             integer nbase )
```

Definition at line 2743 of file AG2.for.

3.1.2.45 mnmx()

```
subroutine mnmx (
             real, dimension(5) arr,
             real amin,
             real amax )
```

Definition at line 891 of file AG2.for.

3.1.2.46 monpos()

```
subroutine monpos (
             integer nbase,
             integer iyl,
             real dpos,
             integer spos )
```

Definition at line 2169 of file AG2.for.

3.1.2.47 notatec()

```
subroutine notatec (
             integer ix,
             integer iy,
             character *(*) string )
```

Definition at line 2629 of file AG2.for.

3.1.2.48 npts()

```
subroutine npts (
             integer ipar )
```

Definition at line 165 of file AG2.for.

3.1.2.49 numsetc()

```
subroutine numsetc (
             real fnum,
             integer iwidth,
             integer nbase,
             character, dimension(*) outstr )
```

Definition at line 2326 of file AG2.for.

3.1.2.50 optim()

```
subroutine optim (
             integer ixy )
```

Definition at line 981 of file AG2.for.

3.1.2.51 oubgc()

```
subroutine oubgc (
             integer iyear,
             integer iday,
             integer iubgcI )
```

Definition at line 1497 of file AG2.for.

3.1.2.52 place()

```
subroutine place (
             integer ipar )
```

Definition at line 522 of file AG2.for.

3.1.2.53 remlab()

```
subroutine remlab (
             integer nbase,
             integer iloc,
             integer labtyp,
             integer ix,
             integer iy )
```

Definition at line 2818 of file AG2.for.

3.1.2.54 rescom()

```
subroutine rescom (
             integer, dimension(1) Array )
```

Definition at line 3061 of file AG2.for.

3.1.2.55 rgchek()

```
subroutine rgchek (
             integer ixy,
             real, dimension(5) arr )
```

Definition at line 864 of file AG2.for.

3.1.2.56 roundd()

```
real function roundd (
              value,
             real, value finterval )
```

Definition at line 3010 of file AG2.for.

3.1.2.57 roundu()

```
real function roundu (
              value,
             real, value finterval )
```

Definition at line 3026 of file AG2.for.

3.1.2.58 savcom()

```
subroutine savcom (
             integer, dimension(1) Array )
```

Definition at line 3045 of file AG2.for.

3.1.2.59 setwin()

```
subroutine setwin
```

Definition at line 632 of file AG2.for.

3.1.2.60 sizel()

```
subroutine sizel (
             real par )
```

Definition at line 198 of file AG2.for.

3.1.2.61 sizes()

```
subroutine sizes (
             real par )
```

Definition at line 187 of file AG2.for.

3.1.2.62 slimx()

```
subroutine slimx (
             integer ixmin,
             integer ixmax )
```

Definition at line 498 of file AG2.for.

3.1.2.63 slimy()

```
subroutine slimy (
             integer iymin,
             integer iymax )
```

Definition at line 510 of file AG2.for.

3.1.2.64 spread()

```
subroutine spread (
             integer nbase )
```

Definition at line 2881 of file AG2.for.

3.1.2.65 stepl()

```
subroutine stepl (
             integer ipar )
```

Definition at line 176 of file AG2.for.

3.1.2.66 steps()

```
subroutine steps (
             integer ipar )
```

Definition at line 131 of file AG2.for.

3.1.2.67 symbl()

```
subroutine symbl (
             integer ipar )
```

Definition at line 120 of file AG2.for.

3.1.2.68 symout()

```
subroutine symout (
             integer isym,
             real fac )
```

Definition at line 1867 of file AG2.for.

3.1.2.69 teksym()

```
subroutine teksym (
             integer isym,
             real amult )
```

Definition at line 1892 of file AG2.for.

3.1.2.70 teksym1()

```
subroutine teksym1 (
             integer istart,
             integer iend,
             integer incr,
             real siz )
```

Definition at line 1940 of file AG2.for.

3.1.2.71 tset()

```
subroutine tset (
             integer nbase )
```

Definition at line 2099 of file AG2.for.

3.1.2.72 tset2()

```
subroutine tset2 (
             integer newloc,
             integer nfar,
             integer nlen,
             integer nfrm,
             integer kstart,
             integer kend )
```

Definition at line 2137 of file AG2.for.

3.1.2.73 typck()

```
subroutine typck (
             integer ixy,
             real, dimension(5) arr )
```

Definition at line 833 of file AG2.for.

3.1.2.74 vbarst()

```
subroutine vbarst (
             integer ishade,
             integer iwbar,
             integer idbar )
```

Definition at line 702 of file AG2.for.

3.1.2.75 vlablc()

```
subroutine vlablc (
             character, dimension(*) string )
```

Definition at line 2654 of file AG2.for.

3.1.2.76 width()

```
subroutine width (
             integer nbase )
```

Definition at line 2702 of file AG2.for.

3.1.2.77 xden()

```
subroutine xden (
             integer ipar )
```

Definition at line 322 of file AG2.for.

3.1.2.78 xetyp()

```
subroutine xetyp (
             integer ipar )
```

Definition at line 606 of file AG2.for.

3.1.2.79 xfrm()

```
subroutine xfrm (
             integer ipar )
```

Definition at line 400 of file AG2.for.

3.1.2.80 xlab()

```
subroutine xlab (
             integer ipar )
```

Definition at line 300 of file AG2.for.

3.1.2.81 xlen()

```
subroutine xlen (
             integer ipar )
```

Definition at line 374 of file AG2.for.

3.1.2.82 xloc()

```
subroutine xloc (
             integer ipar )
```

Definition at line 256 of file AG2.for.

3.1.2.83 xloctp()

```
subroutine xloctp (
             integer ipar )
```

Definition at line 278 of file AG2.for.

3.1.2.84 xmfrm()

```
subroutine xmfrm (
             integer ipar )
```

Definition at line 448 of file AG2.for.

3.1.2.85 xmtcs()

```
subroutine xmtcs (
             integer ipar )
```

Definition at line 426 of file AG2.for.

3.1.2.86 xneat()

```
subroutine xneat (
             integer ipar )
```

Definition at line 212 of file AG2.for.

3.1.2.87 xtics()

```
subroutine xtics (
             integer ipar )
```

Definition at line 352 of file AG2.for.

3.1.2.88 xtype()

```
subroutine xtype (
             integer ipar )
```

Definition at line 554 of file AG2.for.

3.1.2.89 xwdth()

```
subroutine xwdth (
             integer ipar )
```

Definition at line 580 of file AG2.for.

3.1.2.90 xzero()

```
subroutine xzero (
             integer ipar )
```

Definition at line 234 of file AG2.for.

3.1.2.91 yden()

```
subroutine yden (
             integer ipar )
```

Definition at line 337 of file AG2.for.

3.1.2.92 yetyp()

```
subroutine yetyp (
             integer ipar )
```

Definition at line 619 of file AG2.for.

3.1.2.93 yfrm()

```
subroutine yfrm (
             integer ipar )
```

Definition at line 413 of file AG2.for.

3.1.2.94 ylab()

```
subroutine ylab (
             integer ipar )
```

Definition at line 311 of file AG2.for.

3.1.2.95 ylen()

```
subroutine ylen (
             integer ipar )
```

Definition at line 387 of file AG2.for.

3.1.2.96 yloc()

```
subroutine yloc (
             integer ipar )
```

Definition at line 267 of file AG2.for.

3.1.2.97 ylocrt()

```
subroutine ylocrt (
             integer ipar )
```

Definition at line 289 of file AG2.for.

3.1.2.98 ymdyd()

```
subroutine ymdyd (
             integer iJulYrOut,
             integer iJulDayOut,
             integer iGregYrIn,
             integer iGregMonIn,
             integer iGregDayIn )
```

entry subroutine YMDYD (iJulYrIn,iJulDayIn,iGregYrOut,iGregMonOut,iGregDayOut)

Definition at line 1414 of file AG2.for.

3.1.2.99 ymfrm()

```
subroutine ymfrm (
             integer ipar )
```

Definition at line 461 of file AG2.for.

3.1.2.100 ymtcs()

```
subroutine ymtcs (
             integer ipar )
```

Definition at line 437 of file AG2.for.

3.1.2.101 yneat()

```
subroutine yneat (
             integer ipar )
```

Definition at line 223 of file AG2.for.

3.1.2.102 ytics()

```
subroutine ytics (
             integer ipar )
```

Definition at line 363 of file AG2.for.

3.1.2.103 ytype()

```
subroutine ytype (
             integer ipar )
```

Definition at line 567 of file AG2.for.

3.1.2.104 ywdth()

```
subroutine ywdth (
             integer ipar )
```

Definition at line 593 of file AG2.for.

3.1.2.105 yzero()

```
subroutine yzero (
             integer ipar )
```

Definition at line 245 of file AG2.for.

3.2 AG2.for

```
00001 C> \file       AG2.for
00002 C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
00003 C> \version    (2025,347, x)
00004 C> \author     (C) 2025 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C>  Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
00009 C> \note
00010 C>     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00011 C>     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00012 C>
00013 C> \~english
00014 C> Layer 2: scientific 2-D graphic subroutines
00015 C> \note
00016 C>     The control character for exponent (originally -1) is now SOH=char(1)
00017 C>     and for index (originally -2) STX=char(2).
00018 C>
00019 C> \~
00020 C> \note \verbatim
00021 C>   Package:
00022 C>    - AG2.for:         chart plotting routines
00023 C>    - AG2Holerith.for: deprecated routines
00024 C>    - AG2USR.for:      default userroutines
00025 C>    - G2dAG2.fd:       commonblock
00026 C> \endverbatim
00027 C
00028 C
00029 C  Tektronix Advanced Graphics 2 - Version 2.x
00030 C
00031 C
00032 C     Neuer Code in Fortran 77. Die Verwendung der im Manual dokumentierten
00033 C     Unterprogramme bleibt unveraendert, die direkte Manipulation von
00034 C     Variablen des zugrundeliegenden Commonblockes ist jedoch nicht mehr
00035 C     empfehlenswert. IBASEX (iPar) und IBASEY(iPar) mit ipar <>0,
00036 C     IBASEC, COMGET und COMSET sollten in neuen Programmen nicht verwendet
00037 C     werden.
00038 C
00039 C     Die Zwischenspeicherung der Statusvariablen ueber
```

```
00040 C           SAVCOM und RESCOM
00041 C     und die Achsensteuerung ueber
00042 C           IBASEX(0), IBASEY(0) und IOTHER
00043 C     werden weiterhin unterstuetzt.
00044 C
00045 C     Die Implementation der Unterprogramme COMGET und COMSET setzt die gleiche
00046 C     Laenge von REAL und INTEGER-Variablen voraus.
00047 C
00048 C     Da Holerithvariablen von modernen Compilern uneinheitlich unterstuetzt
00049 C     werden (4Habcd entweder als gepackte Integervariable oder als Character-
00050 C     variable interpretiert), wurden die folgenden Routinen angepasst:
00051 C      - subroutine PLACE (Lit): Lit wird nur noch als Ordnungszahl (1..13)
00052 C        und nicht mehr alternativ als Literal ('STD', 'UPH') interpretiert.
00053 C
00054 C     subroutine LEAP (iyear): Die Schaltjahrkorrektur erfolgt nicht mehr
00055 C     als SUBROUTINE ueber einen Common-Block, sondern direkt als
00056 C     integer function LEAP (iyear) ! = 1: Schaltjahr, sonst 0
00057 C
00058 C     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00059 C     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00060 C
00061 C     Intern erfolgt die Stringverarbeitung ueber Charactervariablen als
00062 C     nullterminierte C-Strings.
00063 C
00064 C     Der User-API wurden die folgenden Unterprogramme als Charactervarianten
00065 C     der Original-Holerithroutinen hinzugefuegt:
00066 C      - subroutine NUMSETC (fnum,nbase, outstr,fillstr)
00067 C      - subroutine FONLYC (fnum,iwidth,idec, outstr,fillstr)
00068 C      - subroutine EFORMC (fnum,iwidth,idec, outstr,fillstr)
00069 C      - subroutine EXPOUTC (nbase,iexp, outstr,fillstr)
00070 C      - subroutine ALFSETC (fnum,iwidth,labtyp,outstr)
00071 C      - subroutine NOTATEC (IX,IY,LENCHR,IARRAY)
00072 C      - subroutine JUSTERC
00073 C
00074 C      - subroutine USESETC (fnum, iwidth, nbase, labstr)
00075 C
00076 C      subroutine MONPOS (nbase,iy1,dpos, spos) ! spos ist INTEGER
00077 C      subroutine GLINE (nbase,datapt,spos) ! spos ist INTEGER
00078 C
00079 C     Der Code ab Version 2.0 wird nicht mehr fuer CP/M entwickelt. Letzte
00080 C     unter CP/M compilierbare Version: (2006, 013, 1)
00081 C
00082 C     Zugehoerige Module:
00083 C      - AG2.FOR:     Basisfunktionen
00084 C      - AG2Holerith: Veraltete Unterprogramme zur Wahrung der Kompatibilitaet
00085 C                     (Unterstuetzung Holerithvariablen und vektorisierter Zu-
00086 C                     griff auf den Commonblock)
00087 C      - AG2USR.FOR:  Userroutinen
00088 C      - G2dAG2.fd:   Commonblockdefinition
00089 C
00090
00091 C
00092 C  Ausgabe der Softwareversion
00093 C
00094       subroutine ag2lev (ilevel)
00095       implicit none
00096       integer ilevel(3)
00097
00098       call tcslev (ilevel) ! level(3)= System aus TCS
00099       ilevel(1)=2025       ! Aenderungsjahr
00100       ilevel(2)=  70       ! Aenderungstag
00101       return
00102       end
00103
00104
00105
00106 C
00107 C  Setzen allgemeiner Commonvariablen
00108 C
00109       subroutine line (ipar)
00110       implicit none
00111       integer ipar
00112       include 'G2dAG2.fd'
00113
00114       cline= ipar
00115       return
00116       end
00117
00118
00119
00120       subroutine symbl (ipar)
00121       implicit none
00122       integer ipar
00123       include 'G2dAG2.fd'
00124
00125       csymbl= ipar
00126       return
```

```
00127       end
00128
00129
00130
00131       subroutine steps (ipar)
00132       implicit none
00133       integer ipar
00134       include 'G2dAG2.fd'
00135
00136       csteps= ipar
00137       return
00138       end
00139
00140
00141
00142       subroutine infin (par)
00143       implicit none
00144       real par
00145       include 'G2dAG2.fd'
00146
00147       if (par .gt. 0.) then
00148        cinfin= par
00149       end if
00150       return
00151       end
00152
00153
00154
00155       real function ag2infin ()
00156       implicit none
00157       include 'G2dAG2.fd'
00158
00159       ag2infin= cinfin
00160       return
00161       end
00162
00163
00164
00165       subroutine npts (ipar)
00166       implicit none
00167       integer ipar
00168       include 'G2dAG2.fd'
00169
00170       cnpts= ipar
00171       return
00172       end
00173
00174
00175
00176       subroutine stepl (ipar)
00177       implicit none
00178       integer ipar
00179       include 'G2dAG2.fd'
00180
00181       cstepl= ipar
00182       return
00183       end
00184
00185
00186
00187       subroutine sizes (par)
00188       implicit none
00189       real par
00190       include 'G2dAG2.fd'
00191
00192       csizes= par
00193       return
00194       end
00195
00196
00197
00198       subroutine sizel (par)
00199       implicit none
00200       real par
00201       include 'G2dAG2.fd'
00202
00203       csizel= par
00204       return
00205       end
00206
00207
00208
00209 C
00210 C  Setzen der achsenbezogenen Commonvariablen
00211 C
00212       subroutine xneat (ipar)
00213       implicit none
```

```
00214       integer ipar
00215       include 'G2dAG2.fd'
00216
00217       cxyneat(1) = ipar .ne. 0
00218       return
00219       end
00220
00221
00222
00223       subroutine yneat (ipar)
00224       implicit none
00225       integer ipar
00226       include 'G2dAG2.fd'
00227
00228       cxyneat(2) = ipar .ne. 0
00229       return
00230       end
00231
00232
00233
00234       subroutine xzero (ipar)
00235       implicit none
00236       integer ipar
00237       include 'G2dAG2.fd'
00238
00239       cxyzero(1) = ipar .ne. 0
00240       return
00241       end
00242
00243
00244
00245       subroutine yzero (ipar)
00246       implicit none
00247       integer ipar
00248       include 'G2dAG2.fd'
00249
00250       cxyzero(2) = ipar .ne. 0
00251       return
00252       end
00253
00254
00255
00256       subroutine xloc (ipar)
00257       implicit none
00258       integer ipar
00259       include 'G2dAG2.fd'
00260
00261       cxyloc(1)= ipar
00262       return
00263       end
00264
00265
00266
00267       subroutine yloc (ipar)
00268       implicit none
00269       integer ipar
00270       include 'G2dAG2.fd'
00271
00272       cxyloc(2)= ipar
00273       return
00274       end
00275
00276
00277
00278       subroutine xloctp (ipar)
00279       implicit none
00280       integer ipar
00281       include 'G2dAG2.fd'
00282
00283       cxyloc(1)= ipar+abs(cxysmax(2)-cxysmin(2))
00284       return
00285       end
00286
00287
00288
00289       subroutine ylocrt (ipar)
00290       implicit none
00291       integer ipar
00292       include 'G2dAG2.fd'
00293
00294       cxyloc(2)= ipar + abs(cxysmax(1)-cxysmin(1))
00295       return
00296       end
00297
00298
00299
00300       subroutine xlab (ipar)
```

```
00301       implicit none
00302       integer ipar
00303       include 'G2dAG2.fd'
00304
00305       cxylab(1)= ipar
00306       return
00307       end
00308
00309
00310
00311       subroutine ylab (ipar)
00312       implicit none
00313       integer ipar
00314       include 'G2dAG2.fd'
00315
00316       cxylab(2)= ipar
00317       return
00318       end
00319
00320
00321
00322       subroutine xden (ipar)
00323       implicit none
00324       integer ipar
00325       include 'G2dAG2.fd'
00326
00327       if ((ipar .ge. 0) .and. (ipar .le. 10)) then
00328        cxyden(1)= ipar
00329        cxytics(1)= 0
00330        cxymtcs(1)= 0
00331       end if
00332       return
00333       end
00334
00335
00336
00337       subroutine yden (ipar)
00338       implicit none
00339       integer ipar
00340       include 'G2dAG2.fd'
00341
00342       if ((ipar .ge. 0) .and. (ipar .le. 10)) then
00343        cxyden(2)= ipar
00344        cxytics(2)= 0
00345        cxymtcs(2)= 0
00346       end if
00347       return
00348       end
00349
00350
00351
00352       subroutine xtics (ipar)
00353       implicit none
00354       integer ipar
00355       include 'G2dAG2.fd'
00356
00357       cxytics(1)= abs(ipar)
00358       return
00359       end
00360
00361
00362
00363       subroutine ytics (ipar)
00364       implicit none
00365       integer ipar
00366       include 'G2dAG2.fd'
00367
00368       cxytics(2)= abs(ipar)
00369       return
00370       end
00371
00372
00373
00374       subroutine xlen (ipar)
00375       implicit none
00376       integer ipar
00377       include 'G2dAG2.fd'
00378
00379       if (ipar .ge. 0) then
00380        cxylen(1)= ipar
00381       end if
00382       return
00383       end
00384
00385
00386
00387       subroutine ylen (ipar)
```

```
00388       implicit none
00389       integer ipar
00390       include 'G2dAG2.fd'
00391
00392       if (ipar .ge. 0) then
00393        cxylen(2)= ipar
00394       end if
00395       return
00396       end
00397
00398
00399
00400       subroutine xfrm (ipar)
00401       implicit none
00402       integer ipar
00403       include 'G2dAG2.fd'
00404
00405       if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00406        cxyfrm(1)= ipar
00407       end if
00408       return
00409       end
00410
00411
00412
00413       subroutine yfrm (ipar)
00414       implicit none
00415       integer ipar
00416       include 'G2dAG2.fd'
00417
00418       if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00419        cxyfrm(2)= ipar
00420       end if
00421       return
00422       end
00423
00424
00425
00426       subroutine xmtcs (ipar)
00427       implicit none
00428       integer ipar
00429       include 'G2dAG2.fd'
00430
00431       cxymtcs(1)= abs(ipar)
00432       return
00433       end
00434
00435
00436
00437       subroutine ymtcs (ipar)
00438       implicit none
00439       integer ipar
00440       include 'G2dAG2.fd'
00441
00442       cxymtcs(2)= abs(ipar)
00443       return
00444       end
00445
00446
00447
00448       subroutine xmfrm (ipar)
00449       implicit none
00450       integer ipar
00451       include 'G2dAG2.fd'
00452
00453       if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00454        cxymfrm(1)= ipar
00455       end if
00456       return
00457       end
00458
00459
00460
00461       subroutine ymfrm (ipar)
00462       implicit none
00463       integer ipar
00464       include 'G2dAG2.fd'
00465
00466       if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00467        cxymfrm(2)= ipar
00468       end if
00469       return
00470       end
00471
00472
00473
00474       subroutine dlimx (xmin,xmax)
```

```
00475       implicit none
00476       real xmin,xmax
00477       include 'G2dAG2.fd'
00478
00479       cxydmin(1)= xmin
00480       cxydmax(1)= xmax
00481       return
00482       end
00483
00484
00485
00486       subroutine dlimy (ymin,ymax)
00487       implicit none
00488       real ymin,ymax
00489       include 'G2dAG2.fd'
00490
00491       cxydmin(2)= ymin
00492       cxydmax(2)= ymax
00493       return
00494       end
00495
00496
00497
00498       subroutine slimx (ixmin,ixmax)
00499       implicit none
00500       integer ixmin,ixmax
00501       include 'G2dAG2.fd'
00502
00503       cxysmin(1)= ixmin
00504       cxysmax(1)= ixmax
00505       return
00506       end
00507
00508
00509
00510       subroutine slimy (iymin,iymax)
00511       implicit none
00512       integer iymin,iymax
00513       include 'G2dAG2.fd'
00514
00515       cxysmin(2)= iymin
00516       cxysmax(2)= iymax
00517       return
00518       end
00519
00520
00521
00522       subroutine place (ipar)
00523       implicit none
00524       include 'G2dAG2.fd'
00525       integer ipar
00526
00527       integer postab (4,13)        ! Koordinaten des Zeichenbereiches
00528       data postab /150,900, 125,700,
00529      2             150,850, 525,700,
00530      3             150,850, 150,325,
00531      4             150,450, 525,700,
00532      5             650,950, 525,700,
00533      6             150,450, 150,325,
00534      7             650,950, 150,325,
00535      8             150,325, 525,700,
00536      9             475,650, 525,700,
00537      a             800,975, 525,700,
00538      1             150,325, 150,325,
00539      2             475,650, 150,325,
00540      3             800,975, 150,325/
00541       save postab
00542
00543       if ((ipar .ge. 1) .and. (ipar.le.13)) then
00544        cxysmin(1)= postab(1,ipar)
00545        cxysmax(1)= postab(2,ipar)
00546        cxysmin(2)= postab(3,ipar)
00547        cxysmax(2)= postab(4,ipar)
00548       end if
00549       return
00550       end
00551
00552
00553
00554       subroutine xtype (ipar)
00555       implicit none
00556       integer ipar
00557       include 'G2dAG2.fd'
00558
00559       if ((ipar .ge. 1) .and. (ipar .le. 8)) then
00560        cxytype(1)= ipar
00561       end if
```

```
00562       return
00563       end
00564
00565
00566
00567       subroutine ytype (ipar)
00568       implicit none
00569       integer ipar
00570       include 'G2dAG2.fd'
00571
00572       if ((ipar .ge. 1) .and. (ipar .le. 8)) then
00573        cxytype(2)= ipar
00574       end if
00575       return
00576       end
00577
00578
00579
00580       subroutine xwdth (ipar)
00581       implicit none
00582       integer ipar
00583       include 'G2dAG2.fd'
00584
00585       if (ipar .ge. 0) then
00586        cxywdth(1)= ipar
00587       end if
00588       return
00589       end
00590
00591
00592
00593       subroutine ywdth (ipar)
00594       implicit none
00595       integer ipar
00596       include 'G2dAG2.fd'
00597
00598       if (ipar .ge. 0) then
00599        cxywdth(2)= ipar
00600       end if
00601       return
00602       end
00603
00604
00605
00606       subroutine xetyp (ipar)
00607       implicit none
00608       integer ipar
00609       include 'G2dAG2.fd'
00610
00611       if ((ipar .ge. 0) .and. (ipar .le. 4)) then
00612        cxyetyp(1)= ipar
00613       end if
00614       return
00615       end
00616
00617
00618
00619       subroutine yetyp (ipar)
00620       implicit none
00621       integer ipar
00622       include 'G2dAG2.fd'
00623
00624       if ((ipar .ge. 0) .and. (ipar .le. 4)) then
00625        cxyetyp(2)= ipar
00626       end if
00627       return
00628       end
00629
00630
00631
00632       subroutine setwin
00633       implicit none
00634       include 'G2dAG2.fd'
00635
00636       call twindo (cxysmin(1),cxysmax(1), cxysmin(2),cxysmax(2))
00637       call dwindo (cxydmin(1),cxydmax(1), cxydmin(2),cxydmax(2))
00638       if (cxytype(1) .eq. 2) then
00639        if (cxytype(2) .eq. 2) then
00640         call logtrn (3)
00641        else
00642         call logtrn (1)
00643        end if
00644       else if (cxytype(2) .eq. 2) then
00645         call logtrn (2)
00646       else
00647        call lintrn
00648       end if
```

```
00649       return
00650       end
00651
00652
00653
00654       subroutine dinitx
00655       implicit none
00656       include 'G2dAG2.fd'
00657
00658       cxydmin(1)= 0.        ! Datenbereich
00659       cxydmax(1)= 0.
00660       cxywdth(1)= 0         ! Dezimalstellen
00661       cxydec(1)=  0         ! Dezimalstellen
00662       cxyepon(1)= 0         ! Exponent Label
00663       return
00664       end
00665
00666
00667
00668       subroutine dinity
00669       implicit none
00670       include 'G2dAG2.fd'
00671
00672       cxydmin(2)= 0.        ! Datenbereich
00673       cxydmax(2)= 0.
00674       cxywdth(2)= 0         ! Dezimalstellen
00675       cxydec(2)=  0         ! Dezimalstellen
00676       cxyepon(2)= 0         ! Exponent Label
00677       return
00678       end
00679
00680
00681
00682       subroutine hbarst (ishade,iwbar,idbar)
00683       implicit none
00684       integer ishade,iwbar,idbar
00685       include 'G2dAG2.fd'
00686
00687       cline= -3
00688       if ((ishade .ge. 0).and. (ishade .le. 15)) csymbl= ishade
00689       csizes= real(idbar)
00690       csizel= real(iwbar)
00691
00692       if (cxyfrm(2) .eq. 5) then
00693        cxyfrm(2)= 2
00694       else if (cxyfrm(2) .eq. 6) then
00695        cxyfrm(2)= 1
00696       end if
00697       return
00698       end
00699
00700
00701
00702       subroutine vbarst (ishade,iwbar,idbar)
00703       implicit none
00704       integer ishade,iwbar,idbar
00705       include 'G2dAG2.fd'
00706
00707       cline=  -2
00708       if ((ishade .ge. 0) .and. (ishade .le. 15)) csymbl= ishade
00709       csizes= real(idbar)
00710       csizel= real(iwbar)
00711       if (cxyfrm(1) .eq. 5) then
00712        cxyfrm(1)= 2
00713       else if (cxyfrm(1) .eq. 6) then
00714        cxyfrm(1)= 1
00715       end if
00716       return
00717       end
00718
00719
00720
00721 C
00722 C  Berechnung der Commonvariablen
00723 C
00724       subroutine binitt
00725       implicit none
00726       integer ih
00727       include 'G2dAG2.fd'
00728
00729       cline= 0
00730       csymbl= 0
00731       csteps= 1
00732       cinfin= 1.e30
00733       cnpts= 0
00734       cstepl= 1
00735       cnumbr= 0
```

```
00736       csizes= 1.
00737       csizel= 1.
00738
00739       cxyneat(1)= .true.
00740       cxyneat(2)= .true.
00741       cxyzero(1)= .true.
00742       cxyzero(2)= .true.
00743       cxyloc(1)= 0
00744       cxyloc(2)= 0
00745       cxylab(1)= 1
00746       cxylab(2)= 1
00747       cxyden(1)= 8
00748       cxyden(2)= 8
00749       cxytics(2)= 0
00750       cxytics(2)= 0
00751
00752       call csize (ih,cxylen(1))
00753       cxylen(2)= cxylen(1)
00754
00755       cxyfrm(1)= 5
00756       cxyfrm(2)= 5
00757       cxymtcs(1)= 0
00758       cxymtcs(2)= 0
00759       cxymfrm(1)= 2
00760       cxymfrm(2)= 2
00761       cxydec(1)= 0
00762       cxydec(2)= 0
00763       cxydmin(1)= 0.
00764       cxydmin(2)= 0.
00765       cxydmax(1)= 0.
00766       cxydmax(2)= 0.
00767
00768       cxysmin(1)= 150
00769       cxysmin(2)= 125
00770       cxysmax(1)= 900
00771       cxysmax(2)= 700
00772
00773       cxytype(1)= 1
00774       cxytype(2)= 1
00775       cxylsig(1)= 0
00776       cxylsig(2)= 0
00777       cxywdth(1)= 0
00778       cxywdth(2)= 0
00779       cxyepon(1)= 0
00780       cxyepon(2)= 0
00781       cxystep(1)= 1
00782       cxystep(2)= 1
00783       cxystag(1)= 1
00784       cxystag(2)= 1
00785       cxyetyp(1)= 0
00786       cxyetyp(2)= 0
00787       cxybeg(1)= 0
00788       cxybeg(2)= 0
00789       cxyend(1)= 0
00790       cxyend(2)= 0
00791       cxymbeg(1)= 0
00792       cxymbeg(2)= 0
00793       cxymend(1)= 0
00794       cxymend(2)= 0
00795       cxyamin(1)= 0.
00796       cxyamin(2)= 0.
00797       cxyamax(1)= 0.
00798       cxyamax(2)= 0.
00799       return
00800       end
00801
00802
00803
00804 C
00805 C  Datenanalyse
00806 C
00807
00808       subroutine check (x,y)
00809       implicit none
00810       real  x(5),y(5)
00811       include 'G2dAG2.fd'
00812
00813       external SPREAD ! External wg. Namenskonflikt FTN90-Intrinsic
00814
00815       call typck (1,x)
00816       call rgchek(1,x)
00817       call optim (1)
00818       call width (1)
00819       if (cxystag(1) .eq. 1) call spread (1)
00820       call tset (1)
00821
00822       call typck (2,y)
```

```
00823       call rgchek(2,y)
00824       call optim(2)
00825       call width(2)
00826       if (cxystag(2) .eq. 1) call spread (2)
00827       call tset (2)
00828       return
00829       end
00830
00831
00832
00833       subroutine typck (ixy, arr)
00834       implicit none
00835       integer ixy
00836       real arr(5)
00837       integer i
00838       include 'G2dAG2.fd'
00839
00840       if ((cxytype(ixy) .lt. 3) .or. (nint(arr(1)) .lt. -1 )) then
00841        if ((cnpts .ne. 0) .or. (nint(arr(1)) .ne. -2) ) return
00842        i= nint(arr(3))
00843        if ( i .eq. 1) then
00844         cxytype(ixy)= 8
00845        else if ( i .eq. 4) then
00846         cxytype(ixy)= 7
00847        else if ( i .eq. 12) then
00848         cxytype(ixy)= 6
00849        else if ( i .eq. 13) then
00850         cxytype(ixy)= 5
00851        else if ( i .eq. 52) then
00852         cxytype(ixy)= 4
00853        else if ( i .eq. 365) then
00854         cxytype(ixy)= 3
00855        end if
00856       else
00857        cxytype(ixy)= 1
00858       end if
00859       return
00860       end
00861
00862
00863
00864       subroutine rgchek (ixy,arr)
00865       implicit none
00866       integer ixy
00867       real arr(5)
00868       real amin, amax
00869       include 'G2dAG2.fd'
00870
00871       if (cxydmax(ixy) .eq. cxydmin(ixy)) then ! Bereich schon bestimmt?
00872        if (cxyzero(ixy)) then ! Nullpunktunterdrueckung?
00873         amin= cinfin
00874        else
00875         amin= 0.
00876        end if
00877        amax= -amin
00878        call mnmx (arr, amin, amax)
00879        if (amax .eq. amin) then
00880         amin= amin - 0.5
00881         amax= amax + 0.5
00882        end if
00883        cxydmin(ixy)= amin
00884        cxydmax(ixy)= amax
00885       end if
00886       return
00887       end
00888
00889
00890
00891       subroutine mnmx (arr,amin,amax)
00892       implicit none
00893       real arr(5), amin,amax, aminmax
00894       integer i, itype, nstart,nlim
00895       include 'G2dAG2.fd'
00896
00897       if (cnpts .eq. 0) then                          ! Tek Standard-Format
00898        nlim= nint(arr(1)) + 1
00899        nstart= 2
00900       else
00901        nlim= cnpts
00902        nstart= 1
00903       end if
00904       if ((arr(1) .lt. 0.) .and. (cnpts .eq. 0))  then ! Kurzformate
00905        itype= abs(arr(1))
00906        if (itype .eq. 1) then
00907         aminmax= arr(3) + (arr(2)-1.) * arr(4)
00908         amin= amin1(arr(3),aminmax,amin)
00909         amax= amax1(arr(3),aminmax,amax)
```

```
00910        else if (itype .eq. 2) then
00911         call cmnmx (arr,amin,amax)
00912        else
00913         call umnmx (arr,amin,amax)
00914        end if
00915       else                                                ! Langformate
00916        if (nstart .le. nlim) then
00917         do 100 i= nstart, nlim
00918          if (arr(i) .lt. cinfin) then
00919           if (arr(i).lt. amin) amin= arr(i)
00920           if (arr(i).gt. amax) amax= arr(i)
00921          end if
00922 100     continue
00923        end if
00924       end if
00925       return
00926       end
00927
00928
00929
00930       subroutine cmnmx (arr,amin,amax)
00931       implicit none
00932       real arr(5), amin, amax
00933       integer nTage, iStUBGC, nIntv, iadj, imin,imax
00934       integer minTg,minJr, maxTg,maxJr
00935
00936
00937       nintv= nint(arr(3))
00938       if ((nintv .eq. 52).or.(nintv .eq. 13).or.(nintv .eq. 4)) then
00939        if (nintv .eq. 52) then          ! Wochen
00940         ntage=7
00941        else if (nintv .eq. 13) then     ! 28 Tagemonat
00942         ntage= 28
00943        else if (nintv .eq. 4) then      ! Quartal
00944         ntage=91
00945        end if
00946        call iubgc (nint(arr(4)),1, istubgc)    ! Start: Jahr=arr(4), Tag=1
00947        iadj= mod(istubgc,7)
00948        if (iadj .gt. 3) iadj=iadj-7
00949        imin= istubgc-iadj + nint(arr(5))*ntage ! Min= f(Startjahr,StartIntervall)
00950        imax= imin + nint(arr(2))*ntage
00951
00952       else
00953        if (nintv .eq. 1) then ! Jahre
00954         mintg= 1
00955         maxtg= 1
00956         minjr= nint(arr(4))+1
00957         maxjr= nint(arr(4)+arr(2))
00958        else if ( nintv .eq. 12) then ! Monate
00959         call ymdyd (minjr,mintg, nint(arr(4)),nint(arr(5))+1,1)
00960         call ymdyd (maxjr,maxtg, nint(arr(4)),nint(arr(5)+arr(2)),1)
00961        else if ( nintv .eq. 365) then ! Tage
00962         minjr= nint(arr(4))
00963         mintg= nint(arr(5))
00964         maxjr= nint(arr(4))
00965         maxtg= nint(arr(5)+arr(2)) -1
00966        end if
00967        call iubgc (minjr,mintg, imin)
00968        call iubgc (maxjr,maxtg, imax)
00969       end if
00970       if (real(imax) .gt. amax) amax= real(imax)
00971       if (real(imin) .lt. amin) amin= real(imin)
00972       return
00973       end
00974
00975
00976
00977 C
00978 C  Ticmarkoptimierung
00979 C
00980
00981       subroutine optim (ixy)
00982       implicit none
00983       integer ixy
00984       include 'G2dAG2.fd'
00985
00986       if (cxytype(ixy) .eq. 2) cxylab(ixy)= 2
00987       if (cxylab(ixy) .eq. 2) cxylab(ixy)= cxytype(ixy)
00988       if (cxytype(ixy) .le. 2) then
00989        call loptim (ixy) ! Tic-Mark Optimierung fuer lineare und log. Daten
00990       else
00991        call coptim (ixy) ! Tic-Mark Optimierung fuer Kalenderdaten
00992       end if
00993       return
00994       end
00995
00996
```

```
00997
00998       subroutine loptim (ixy)
00999       implicit none
01000       integer ixy ,i, labtyp, ntics, lsig, mtcs
01001       real dataint, amin,amax, aminor,amaxor, sigfac
01002       integer idataint
01003       integer mintic
01004       integer LINWDT, LINHGT
01005       real ROUNDD, ROUNDU
01006       include 'G2dAG2.fd'
01007
01008       labtyp=abs( cxylab(ixy)) ! <0: Userlabel
01009       if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp
01010
01011       amin= cxydmin(ixy)
01012       amax= cxydmax(ixy)
01013       ntics= abs(cxytics(ixy)) ! Anzahl >=1, 0= Flag fuer autoscale
01014       mintic= 0
01015
01016       if (labtyp .eq. 2) then ! logarithmische Achsen
01017        amin= log10(max(amin,1./cinfin)) + 1.e-7  ! !> 0 => log10 definiert
01018        amax= log10(amax)
01019       end if
01020
01021       aminor= amin
01022       amaxor= amax
01023
01024       if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
01025        if (ixy.eq.1) then
01026         i= linwdt(8) ! 100 + LINWDT(3)
01027        else
01028         i= linhgt(3) ! 50 + LINHGT(3)
01029        end if
01030        ntics= (cxysmax(ixy) - cxysmin(ixy)) / i
01031        if (ntics .lt. 1) ntics= 1
01032       end if
01033       dataint= abs(amax-amin) / real(ntics)
01034
01035 310   continue ! repeat...
01036        if (labtyp .eq. 2) dataint= roundu(dataint,1.) ! logarithmische Achsen
01037        lsig= rounddd(log10(dataint),1.) ! Anzahl signifikanter Nachkommastellen
01038        sigfac=10.**(lsig)
01039        if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
01040         if(labtyp .ne. 2) then ! nicht bei log. Achsen
01041          if ((dataint/sigfac) .le. 1.) then
01042           dataint= 1. * sigfac
01043           mintic= 10
01044          else if ((dataint/sigfac) .le. 2.) then
01045           dataint= 2. * sigfac
01046           mintic= 2
01047          else if ((dataint/sigfac) .le. 2.5) then
01048           dataint= 2.5 * sigfac
01049           mintic= 5
01050           lsig=lsig-1
01051          else if ((dataint/sigfac) .le. 5.) then
01052           dataint= 5. * sigfac
01053           mintic=  5
01054          else if ((dataint/sigfac) .le. 10.) then
01055           dataint= 10. * sigfac
01056           mintic= 10
01057           lsig=lsig+1
01058          else
01059           dataint= cinfin
01060           mintic= 0
01061          end if
01062         end if ! log. Achse
01063        else ! .not. neat
01064         lsig=lsig-2
01065        end if
01066        if (lsig .ge. 0) lsig=lsig+1
01067       if (cxyneat(ixy) .or. (labtyp .eq. 2) ) then ! ... until
01068        amin= rounddd(amin+.01*sigfac,dataint) !  runde auf TicIntervall
01069        amax= roundu(amax-.01*sigfac,dataint) ! .01*sigfac= Genauigkeit Plot
01070        ntics= int(abs(amax-amin)/dataint+.0001)
01071        if(cxytics(ixy) .ne. 0) then ! until: ntics nicht vorbesetzt oder = vorbesetzt
01072         if(abs(cxytics(ixy)) .lt. ntics) then
01073          dataint= dataint * 1.1
01074          amin=aminor
01075          amax=amaxor
01076          goto 310 ! noch eine Iterationsschleife
01077         else if (abs(cxytics(ixy)) .gt. ntics) then
01078          ntics= abs(cxytics(ixy))
01079          amax= amin + real(ntics) * dataint
01080         end if ! abs(cxytics(ixy)) .eq. ntics: no action
01081        end if
01082       end if
01083       cxytics(ixy)= ntics
```

```
01084
01085       if ((cxymtcs(ixy) .eq. 0) .and. (cxyden(ixy) .ge. 6)) then ! unbesetzt oder wenig TICS
01086        mtcs= mintic ! Bestimmung Minor TicMarcs
01087        if((mtcs .eq. 10) .or. (labtyp .eq. 2)) then
01088         if(cxyden(ixy) .lt. 9) mtcs=5
01089         if(cxyden(ixy) .lt. 7) mtcs=2
01090         if(labtyp .eq. 2) then ! log. Achsen
01091          idataint= nint(dataint)
01092          if (idataint .ne. 1) then ! mehrere Achsenintervalle
01093           i= 1
01094 320       continue ! repeat...
01095            mtcs= idataint/i
01096           if ((mtcs*i .ne. idataint) .and. (i .lt. (idataint-1))) then ! ...until
01097            i= i+1
01098            goto 320
01099           else if (mtcs .gt. 10 ) then
01100            mtcs= 0  ! Failure
01101           end if
01102          else ! einzelne logarithmische Dekade
01103           if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 100* ntics) mtcs=-1 ! logarithm. Tics
01104           if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 20* linhgt(1)) mtcs=-2 ! Label
01105          end if
01106         end if
01107        end if
01108        cxymtcs(ixy)= mtcs
01109       end if
01110
01111       cxylsig(ixy)= lsig
01112       cxyamin(ixy)= amin
01113       cxyamax(ixy)= amax
01114       if (labtyp .eq. 2) then ! logarithmische Achsen: Wiederherstellung der Originalwerte
01115        amax=10.**amax
01116        amin=10.**amin
01117       end if
01118       cxydmin(ixy)= amin
01119       cxydmax(ixy)= amax
01120       return
01121       end
01122
01123
01124
01125       subroutine coptim (ixy)
01126       implicit none
01127       integer ixy , labtyp, ntics
01128       real dataint, amin,amax, aminor,amaxor
01129       integer LINWDT
01130       real ROUNDD, ROUNDU
01131       include 'G2dAG2.fd'
01132
01133       if (cxytics(ixy) .eq. 1) cxytics(ixy)= 2 ! Minimum manuelle Ticwahl: 2
01134       labtyp=abs( cxylab(ixy)) ! <0: Userlabel
01135       if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp
01136       amin= cxydmin(ixy)
01137       amax= cxydmax(ixy)
01138       call calcon (amin,amax,labtyp,.true.) ! Konvertiere UBGC -> Labelzeiteinheit
01139       ntics= cxytics(ixy)
01140       aminor=amin
01141       amaxor=amax
01142       if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
01143        ntics= (cxysmax(ixy) - cxysmin(ixy)) / (25 + linwdt(1))
01144        if (ntics .lt. 2) ntics= 2
01145       end if
01146       dataint= abs(amax-amin) / real(ntics)
01147
01148       if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
01149 310    continue ! repeat...
01150         if (cxytics(ixy) .eq. 0) then ! keine manuelle Belegung erfolgt
01151          if (labtyp.eq.3) then ! Labeltyp: Tage
01152           if (dataint .le. 1.) then
01153            dataint= 1.
01154           else if (dataint .le. 7.) then
01155            dataint= 7.
01156           else if (dataint .le. 14.) then
01157            dataint= 14.
01158           else if (dataint .le. 28.) then
01159            dataint= 28.
01160           else if (dataint .le. 56.) then
01161            dataint= 56.
01162           else if (dataint .le. 128.) then
01163            dataint= 128.
01164           end if ! dataint > 128 -> unveraendert
01165          else if (labtyp.eq.4) then ! Labeltyp: Wochen
01166           if (dataint .le. 1.) then
01167            dataint= 1.
01168           else if (dataint .le. 2.) then
01169            dataint= 2.
01170           else if (dataint .le. 4.) then
```

```
01171           dataint= 4.
01172          else if (dataint .le. 8.) then
01173           dataint= 8.
01174          else if (dataint .le. 16.) then
01175           dataint= 16.
01176          else if (dataint .le. 26.) then
01177           dataint= 26.
01178          else if (dataint .le. 52.) then
01179           dataint= 52.
01180          else if (dataint .le. 104.) then
01181           dataint= 104.
01182          end if ! dataint -> unveraendert
01183         else if (labtyp.eq.5) then ! Labeltyp: Kalenderabschnitte
01184          if (dataint .le. 1.) then
01185           dataint= 1.
01186          else if (dataint .le. 2.) then
01187           dataint= 2.
01188          else if (dataint .le. 13.) then
01189           dataint= 13.
01190          else if (dataint .le. 26.) then
01191           dataint= 26.
01192          else if (dataint .le. 52.) then
01193           dataint= 52.
01194          end if ! dataint -> unveraendert
01195         else if (labtyp.eq.6) then ! Labeltyp: Monate
01196          if (dataint .le. 1.) then
01197           dataint= 1.
01198          else if (dataint .le. 2.) then
01199           dataint= 2.
01200          else if (dataint .le. 3.) then
01201           dataint= 3.
01202          else if (dataint .le. 4.) then
01203           dataint= 4.
01204          else if (dataint .le. 6.) then
01205           dataint= 6.
01206          else if (dataint .le. 12.) then
01207           dataint= 12.
01208          else if (dataint .le. 24.) then
01209           dataint= 24.
01210          else if (dataint .le. 36.) then
01211           dataint= 36.
01212          end if ! dataint -> unveraendert
01213         else if (labtyp.eq.7) then ! Labeltyp: Quartale
01214          if (dataint .le. 1.) then
01215           dataint= 1.
01216          else if (dataint .le. 2.) then
01217           dataint= 2.
01218          else if (dataint .le. 4.) then
01219           dataint= 4.
01220          else if (dataint .le. 8.) then
01221           dataint= 8.
01222          else if (dataint .le. 12.) then
01223           dataint= 12.
01224          else if (dataint .le. 16.) then
01225           dataint= 16.
01226          else if (dataint .le. 24.) then
01227           dataint= 24.
01228          end if ! dataint -> unveraendert
01229         else if (labtyp.eq.8) then ! Labeltyp: Jahre
01230          if (dataint .le. 1.) then
01231           dataint= 1.
01232          else if (dataint .le. 2.) then
01233           dataint= 2.
01234          else if (dataint .le. 5.) then
01235           dataint= 5.
01236          else if (dataint .le. 10.) then
01237           dataint= 10.
01238          else if (dataint .le. 20.) then
01239           dataint= 20.
01240          else if (dataint .le. 50.) then
01241           dataint= 50.
01242          else if (dataint .le. 100.) then
01243           dataint= 100.
01244          end if ! dataint -> unveraendert
01245         end if ! labtyp 3..8
01246        end if ! manuelle Vorbesetzung
01247        amin= roundd(amin,dataint) !  runde auf TicIntervall
01248        amax= roundu(amax,dataint)
01249        ntics= ifix(abs(amax-amin)/dataint+.0001)
01250        if (ntics .eq. 0) ntics = 2
01251       if(cxytics(ixy) .ne. 0) then ! until: ntics nicht oder = vorbesetzt
01252        if(abs(cxytics(ixy)) .lt. ntics) then ! Verringere Ticanzahl
01253         dataint= dataint * 1.1
01254         amin=aminor
01255         amax=amaxor
01256         goto 310 ! noch eine Iterationsschleife
01257        else if (abs(cxytics(ixy)) .gt. ntics) then ! Vergroessere Ticanzahl
```

```
01258          ntics= abs(cxytics(ixy))
01259          amax= amin + real(ntics) * dataint
01260         end if ! abs(cxytics(ixy)) .eq. ntics: no action
01261        end if ! Ende der Schleife
01262       end if ! neat
01263       cxytics(ixy)= ntics
01264       cxylsig(ixy)= 0
01265       cxyamin(ixy)= amin
01266       cxyamax(ixy)= amax
01267       call calcon (amin,amax,labtyp,.false.) ! Labelzeiteinheit -> UBGC
01268       cxydmin(ixy)= amin
01269       cxydmax(ixy)= amax
01270       return
01271       end
01272
01273
01274
01275 C
01276 C  Kalenderroutinen
01277 C
01278
01279
01280
01281       real function calpnt (arr,i)
01282       implicit none
01283       integer i
01284       real arr(5)
01285       integer iy,idays, itmp
01286       integer icltyp, istyr, istper, iubg1, iweek1, nodays
01287       save icltyp, istyr, istper, iubg1, iweek1, nodays
01288
01289       if (i .eq. 1) then ! 1. Datenpunkt: Formatanalyse, Parameterberechnung
01290        istyr= nint(arr(4))
01291        istper= nint(arr(5))
01292        itmp= nint(arr(3)) ! Laenge Intervall in Tagen
01293        if (itmp .eq. 12) then ! Zeitintervall Monat
01294         icltyp= 2
01295        else if (itmp .eq. 365) then ! Zeitintervall Tage
01296         icltyp=3
01297         call iubgc (istyr,istper,iubg1)
01298        else if (itmp .eq. 52) then ! Zeitintervall Wochen
01299         icltyp= 4
01300         nodays= 7
01301        else if (itmp .eq. 13) then ! Zeitintervall 4 Wochen
01302         icltyp= 5
01303         nodays= 28
01304        else if (itmp .eq. 4) then  ! Zeitintervall Quartal
01305         icltyp= 6
01306         nodays= 91
01307        else ! Zeitintervall Jahre
01308         icltyp= 1
01309        end if
01310        if (icltyp .ge. 4) then
01311         call iubgc (istyr,1,iubg1)
01312         itmp= mod(iubg1+1,7)
01313         if(itmp .gt. 3) itmp= itmp-7
01314         iweek1= iubg1-itmp
01315         iubg1= iweek1+(istper-1)*nodays
01316        end if
01317       end if ! Ende Initialisierung, jetzt Berechnung
01318
01319       if (icltyp .eq. 1) then ! Zeitintervall Jahr
01320        call iubgc (istyr+i,1,iubg1)
01321        calpnt= iubg1
01322       else if (icltyp .eq. 2) then ! Zeitintervall Monat
01323        call ymdyd (iy,idays,istyr,istper+i,1)
01324        call iubgc (iy,idays,iubg1)
01325        calpnt= iubg1 ! Zeitintervall Tage
01326       else if (icltyp .eq. 3) then
01327        calpnt= iubg1+i-1
01328       else ! Zeitintervall Wochen oder 4 Wochen
01329        calpnt= iweek1+(istper-1+i)*nodays
01330       end if
01331       return
01332       end
01333
01334
01335
01336       subroutine calcon (amin,amax,labtyp,ubgc)
01337       implicit none
01338       real amin, amax
01339       integer labtyp
01340       logical ubgc
01341       integer iubg1, iubg2, iday1, iadj, id, month1,month2 , imin,imax
01342       real dimin, dimax
01343       integer iweek1
01344       real fnoday
```

```
01345       integer iy1,iy2, iy3,iy4, idays
01346       save iweek1, fnoday
01347       save iy1,iy2, iy3, iy4, idays
01348
01349       real ROUNDD, ROUNDU
01350
01351       if (labtyp .le. 3) return ! nicht Kalender, bzw.Tage: keine Transformation
01352
01353       if (ubgc) then ! Konvertierung UBGC in Labeltype
01354        if ( (labtyp .eq. 4).or.(labtyp .eq. 5).or.(labtyp .eq. 7) ) then
01355         if (labtyp .eq. 4) fnoday= 7.
01356         if (labtyp .eq. 5) fnoday= 28.
01357         if (labtyp .eq. 7) fnoday= 91.
01358         iubg1=amin
01359         iubg2=amax
01360         call oubgc (iy1,idays,iubg1) ! Wochenanfang der 1.KW Startjahr
01361         iday1=iubg1-idays+1
01362         iadj=mod(iday1+1,7)
01363         if(iadj .gt. 3) iadj=iadj-7
01364         iweek1= iday1-iadj           ! Merken in iweek1
01365         dimin= roundd(real(iubg1-iweek1),fnoday)
01366         dimin= dimin/fnoday+1.
01367         call oubgc (iy2,idays,iubg2)
01368         dimax= roundu(real(iubg2-iweek1),fnoday)
01369         dimax= dimax/fnoday
01370        else if (labtyp .eq. 6) then
01371         call oubgc (iy1,idays,nint(amin))
01372         call ydymd (iy1,idays,iy3,month1,id)
01373         dimin= month1
01374         call oubgc (iy2,idays,nint(amax))
01375         call ydymd (iy2,idays,iy4,month2,id)
01376         dimax= (iy4-iy3)*12+month2
01377         if(id .gt. 1) dimax=dimax+1.
01378        else if (labtyp .eq. 8) then
01379         call oubgc (iy1,idays,nint(amin))
01380         dimin= iy1
01381         call oubgc(iy2,idays,nint(amax))
01382         dimax= iy2
01383         if(idays .gt. 1) dimax=dimax+1.
01384        end if
01385        amin= dimin-1.
01386        amax= dimax-1.
01387        return
01388
01389       else ! Konvertierung Labeltype in UBGC
01390        amin=amin+1.
01391        amax=amax+1.
01392        if ((labtyp .eq. 4).or.(labtyp .eq. 5).or.(labtyp .eq. 7)) then
01393         amin= iweek1 + (nint(amin)-1) * nint(fnoday)
01394         amax= iweek1+(nint(amax)-1)*nint(fnoday)
01395        else if (labtyp .eq. 6)then
01396         iy4= iy3
01397         call ymdyd (iy1,idays,iy3,nint(amin),1)
01398         call iubgc (iy1,idays,imin)
01399         amin= imin
01400         call ymdyd (iy2,idays,iy4,nint(amax),1)
01401         call iubgc (iy2,idays,imax)
01402         amax= imax
01403        else if (labtyp .eq. 8) then
01404         call iubgc (nint(amin),1,imin)
01405         amin= imin
01406         call iubgc (nint(amax),1,imax)
01407         amax= imax
01408        end if
01409       endif
01410       return
01411       end
01412
01413
01414       subroutine ymdyd (iJulYrOut,iJulDayOut,
01415      1                                 iGregYrIn,iGregMonIn,iGregDayIn)
01416        implicit none
01417        integer iJulYrOut,iJulDayOut, iGregYrIn,iGregMonIn,iGregDayIn
01418        integer iJulYrIn,iJulDayIn, iGregYrOut,iGregMonOut,iGregDayOut
01419        integer iMon, LEAP
01420        integer iDatTab(12)
01421        save idattab
01422        data idattab /0,31,59,90,120,151,181,212,243,273,304,334/
01423
01424        ijulyrout= igregyrin
01425        imon= igregmonin
01426 100    if (imon .lt. 1) then ! while iMon .not. in [1..12]
01427         imon= imon + 12
01428         ijulyrout= ijulyrout-1
01429         goto 100
01430        else if (imon .gt. 12) then
01431         imon= imon -12
```

```
01432        ijulyrout= ijulyrout+1
01433        goto 100
01434       end if
01435       ijuldayout= igregdayin + idattab(imon)
01436       if (imon .gt.2) ijuldayout= ijuldayout + leap(ijulyrout)
01437       return
01438
01439 C> entry subroutine YMDYD (iJulYrIn,iJulDayIn,iGregYrOut,iGregMonOut,iGregDayOut)
01440       entry ydymd(ijulyrin,ijuldayin,
01441      1                       igregyrout,igregmonout,igregdayout)
01442
01443       igregdayout= ijuldayin
01444       igregyrout= ijulyrin
01445 110   if (igregdayout .lt. 1) then ! while iGregDayOut .not. in [1..365(366)]
01446        igregyrout= igregyrout-1
01447        igregdayout= igregdayout + 365 + leap(igregyrout)
01448        goto 110
01449       else if (igregdayout .gt. 365+ leap(igregyrout)) then
01450        igregyrout= igregyrout+1
01451        igregdayout= igregdayout - 365 - leap(igregyrout)
01452        goto 110
01453       end if
01454
01455       igregmonout= int( real(igregdayout)/29.5+1.)
01456       if (igregdayout .le. idattab(igregmonout)) then
01457        if ((igregmonout .le. 2) .or.
01458      1   (igregdayout.le.(idattab(igregmonout)+leap(igregyrout)))) then
01459          igregmonout= igregmonout-1
01460        end if
01461       end if
01462       igregdayout= igregdayout- idattab(igregmonout)
01463       if (igregmonout .gt. 2) igregdayout= igregdayout -leap(igregyrout)
01464       return
01465       end
01466
01467
01468
01469       integer function  leap (iyear)
01470       implicit none
01471       integer iyear
01472       if (  (mod(iyear,4) .eq. 0) .and.
01473      1     ((mod(iyear,100).ne.0) .or. (mod(iyear,400).eq.0))  ) then
01474        leap= 1
01475       else
01476        leap= 0
01477       end if
01478       return
01479       end
01480
01481
01482
01483       subroutine iubgc(iyear,iday, iubgcO)
01484       implicit none
01485       integer iyear,iday,iubgcO
01486       integer iYr1
01487
01488       iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
01489       iubgco= 365* (iyear-1901) ! Verhinderung Overflow: Offset im Faktor
01490       iubgco= iubgco + int(iyr1/4) - int(iyr1/100) + int(iyr1/400)
01491       iubgco= iubgco + iday -460 ! Bezugsdatum 1.1.1901= 365*1901 + 460 Schalttage
01492       return
01493       end
01494
01495
01496
01497       subroutine oubgc(iyear,iday,iubgcI)
01498       implicit none
01499       integer iyear,iday,iubgcI
01500       integer iYr1
01501
01502       iyear= int( (real(iubgci) + 694325.99) / 365.2425 )
01503 100   continue ! Schleife der evtl. Nachiteration
01504        iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
01505        iday= iubgci + 460 - 365*(iyear-1901)
01506        iday= iday + int(iyr1/100) - int(iyr1/4) - int(iyr1/400)
01507       if (iday .lt. 1) then ! Nachiteration?
01508        iyear= iyear-1
01509        goto 100
01510       end if
01511       return
01512       end
01513
01514
01515
01516 C
01517 C  Zeichenroutinen
01518 C
```

```
01519
01520       subroutine frame
01521       implicit none
01522       include 'G2dAG2.fd'
01523
01524       call movabs (cxysmax(1),cxysmin(2))
01525       call drwabs (cxysmax(1),cxysmax(2))
01526       call drwabs (cxysmin(1),cxysmax(2))
01527       call drwabs (cxysmin(1),cxysmin(2))
01528       call drwabs (cxysmax(1),cxysmin(2))
01529       return
01530       end
01531
01532
01533
01534       subroutine dsplay (x,y)
01535       implicit none
01536       real x(5),y(5)
01537
01538       call setwin
01539       call cplot (x,y)
01540       call grid
01541       call label (1)
01542       call label (2)
01543       return
01544       end
01545
01546
01547
01548       subroutine cplot (x,y)
01549       implicit none
01550       real x(5),y(5)
01551       logical symbol
01552       integer i,i1, keyx, keyy, lines, linsav, icount, imax
01553       real xpoint(1), ypoint(1)
01554       real DATGET
01555       include 'G2dAG2.fd'
01556
01557       call keyset (x,keyx)
01558       call keyset (y,keyy)
01559       if (keyx .eq. 1) then ! standard long
01560        imax= x(1)
01561       else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
01562        imax= x(2)
01563       else ! nonstandard
01564        imax= cnpts
01565       end if
01566       if (keyy .eq. 1) then ! standard long
01567        if (imax .lt. y(1)) imax= y(1)
01568       else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
01569        if (imax .lt. y(2)) imax= y(2)
01570       else ! nonstandard
01571        if (imax .lt. cnpts) imax= cnpts
01572       end if
01573
01574       symbol= (csymbl .ne. 0) .and.(cline .ne.-2) .and.(cline .ne.-3)
01575
01576       i= 1 ! Suche Startpunkt
01577 100   continue ! repeat
01578        if (i .gt. imax) return ! kein Punkt zu zeichnen
01579        xpoint(1)= datget(x,i,keyx)
01580        ypoint(1)= datget(y,i,keyy)
01581       if ((xpoint(1) .ge. cinfin) .or. (ypoint(1) .ge. cinfin)) then ! while
01582        i= i+cstepl
01583        goto 100
01584       end if
01585
01586       call movea (xpoint(1),ypoint(1))
01587       if (cline .eq. -4) call pointa (xpoint(1),ypoint(1))
01588       if (cline .lt. -10) call uline (xpoint(1),ypoint(1),1)
01589       if (cline .eq.-2 .or. cline .eq.-3) then
01590        call bar (xpoint(1),ypoint(1),cline)
01591       end if
01592       if (symbol) call bsyms (xpoint(1),ypoint(1),csymbl)
01593
01594       if (cline .eq. -1) then
01595        lines= 2
01596       else if ((cline .eq. -2) .or. (cline .eq. -3)) then
01597        lines= 3
01598       else if (cline .eq. -4) then
01599        lines=4
01600       else if (cline .lt. -10) then
01601        lines=5
01602       else
01603        lines=1 ! bei cline = 0: dash ergibt durchgezogene Linie
01604       end if
01605
```

```
01606       i1= i+cstepl
01607       if (i1 .ge. imax) return
01608       icount= csteps
01609       linsav= lines
01610
01611       do 900 i=i1,imax,cstepl
01612        xpoint(1)= datget(x,i,keyx)
01613        ypoint(1)= datget(y,i,keyy)
01614        if ((xpoint(1) .ge. cinfin) .or. (ypoint(1) .ge. cinfin)) then
01615         if (i.gt.imax-cstepl) return ! Der letzte Punkt ist ungueltig -> done
01616         if ((cline .ne. -2) .and. (cline .ne. 3)) lines= 2
01617        else
01618         if (lines .eq. 1 ) then
01619          call dasha (xpoint(1),ypoint(1), cline) ! dashed or solid
01620         else if (lines .eq. 2 ) then
01621          call movea (xpoint(1),ypoint(1))
01622          lines=linsav ! restore after missing data
01623         else if (lines .eq. 3 ) then
01624          call bar (xpoint(1),ypoint(1),0)
01625         else if (lines .eq. 4 ) then
01626          call pointa (xpoint(1),ypoint(1))
01627         else
01628          call uline (xpoint(1),ypoint(1),i)
01629         end if
01630         if (symbol) then
01631          icount=icount-1
01632          if(icount .le. 0) then
01633           icount= csteps
01634           call bsyms (xpoint(1),ypoint(1),csymbl)
01635          end if
01636         end if
01637        end if
01638 900   continue
01639       return
01640       end
01641
01642
01643
01644       subroutine keyset (array,key)
01645       implicit none
01646       integer key
01647       integer npts
01648       real array(1)
01649       include 'G2dAG2.fd'
01650
01651       if (cnpts .ne. 0) then       ! nonstandard array
01652        key= 5
01653       else
01654        npts= nint(array(1))
01655        if (npts .ge. 0) then       ! standard long
01656         key= 1
01657        else if (npts .eq. -1) then ! short
01658         key= 2
01659        else if (npts .eq. -2) then ! short calendar
01660         key= 3
01661        else                        ! short user
01662         key= 4
01663        end if
01664       end if
01665       return
01666       end
01667
01668
01669
01670       real function datget (arr,i,key)
01671       implicit none
01672       integer i, key
01673       real calpnt, upoint
01674       real arr(5) ! Dimension 5 sonst GNU-Compilerwarnung bei dat= ...arr(5)...
01675       real dat, olddat
01676       save olddat
01677
01678       if (key.eq.1) then ! standard long
01679        dat= arr(i+1)
01680       else if (key.eq.2) then ! standard short
01681        dat= arr(3) + arr(4)*real(i-1)
01682       else if (key.eq.3) then ! short calendar
01683        dat= calpnt(arr,i)
01684       else if (key.eq.4) then ! user
01685        dat= upoint(arr,i,olddat)
01686       else if (key.eq.5) then ! non standard
01687        dat= arr(i)
01688       endif
01689       olddat= dat
01690       datget= dat
01691       return
01692       end
```

```
01693
01694
01695
01696 C  Balkendiagramme
01697
01698       subroutine bar (x,y,line)
01699       implicit none
01700       real x, y
01701       integer line
01702       integer key, ix,iy, ixl,iyl,ixh,iyh
01703       real xfac, yfac
01704       logical VerticalBar
01705       integer isymb, ihalf, lspace, minx,maxx,miny,maxy, ibegx,ibegy
01706       SAVE isymb, ihalf, lspace, minx,maxx,miny,maxy, ibegx,ibegy
01707       SAVE verticalbar
01708       include 'G2dAG2.fd'
01709
01710       if (line .ne. 0) then ! Erster Aufruf -> Parameterbestimmung
01711        verticalbar= line .ne. -3
01712        isymb= csymbl
01713        ihalf= .5 * csizel
01714        lspace= csizes
01715        if (lspace .le. 1) lspace=20 ! Default: 20 Pixel Schraffur
01716        if (ihalf .lt. 2) ihalf=20 ! Default: 40 Pixel Balkenbreite
01717        if (cxysmin(1) .le. cxysmax(1)) then
01718         minx= cxysmin(1)
01719         maxx= cxysmax(1)
01720        else
01721         minx= cxysmax(1)
01722         maxx= cxysmin(1)
01723        end if
01724        if (cxysmin(2) .le. cxysmax(2)) then
01725         miny= cxysmin(2)
01726         maxy= cxysmax(2)
01727        else
01728         miny= cxysmax(2)
01729         maxy= cxysmin(2)
01730        end if
01731
01732        call seetrn(xfac,yfac, key)
01733        if (key .eq. 2) then ! logarithmische Werte
01734         ibegx= cxysmin(1)
01735         ibegy= cxysmin(2)
01736        else
01737         call wincot (0.,0.,ibegx,ibegy)
01738        end if
01739       end if
01740
01741       call wincot (x,y,ix,iy)
01742       if (verticalbar) then ! vertikale Balken
01743        iyl= min0(ibegy,iy)
01744        iyh= max0(ibegy,iy)
01745        ixl= min0(ix-ihalf,ix+ihalf)
01746        ixh= max0(ix-ihalf,ix+ihalf)
01747       else ! horizontale Balken
01748        iyl= min0(iy-ihalf,iy+ihalf)
01749        iyh= max0(iy-ihalf,iy+ihalf)
01750        ixl= min0(ibegx,ix)
01751        ixh= max0(ibegx,ix)
01752       end if
01753       ixl=max0(ixl,minx)
01754       ixh=min0(ixh,maxx)
01755       iyl=max0(iyl,miny)
01756       iyh=min0(iyh,maxy)
01757       if ((ixh-ixl .ge. 2) .and. (iyh-iyl .ge. 2)) then ! mindestens 2x2 Pxl
01758        call filbox(ixl,iyl,ixh,iyh,isymb,lspace)
01759       end if
01760       return
01761       end
01762
01763
01764
01765       subroutine filbox (minx,miny,maxx,maxy,ishade,lspace)
01766       implicit none
01767       integer minx,miny,maxx,maxy,ishade,lspace
01768       integer iminx,imaxx,iminy,imaxy
01769       integer i, ishift, idely, iymax
01770       real ximin, ximax
01771       real savcom (60)
01772
01773       iminx= min0(minx,maxx)        ! zeichne Rechteck
01774       iminy= min0(miny,maxy)
01775       imaxx= max0(minx,maxx)
01776       imaxy= max0(miny,maxy)
01777
01778       call movabs (iminx,iminy)
01779       call drwabs (imaxx,iminy)
```

```
01780       call drwabs (imaxx,imaxy)
01781       call drwabs (iminx,imaxy)
01782       call drwabs (iminx,iminy)
01783
01784       if ((ishade .le.0) .or. (ishade .gt. 15)) return ! ohne Schraffur
01785
01786       ishift= ishade / 2
01787       if ((ishade-ishift*2) .ne. 0) then ! Bit0: horizontale Schraffur
01788        i= iminy
01789 100    continue ! repeat...
01790         i= i+lspace
01791        if (i .lt. imaxy) then
01792         call movabs (iminx,i)
01793         call drwabs (imaxx,i)
01794         goto 100 ! ... until
01795        end if
01796       end if ! horizontale Schraffur gezeichnet
01797
01798       if (mod(ishift,2) .ne. 0) then ! Bit1: vertikale Schraffur
01799        i= iminx
01800 110    continue ! repeat
01801         i= i+lspace
01802        if(i .lt. imaxx) then
01803         call movabs (i,iminy)
01804         call drwabs (i,imaxy)
01805         goto 110
01806        end if ! vertikale Schraffur gezeichnet
01807       end if
01808
01809       if (ishade .ge. 4) then ! diagonale Schraffuren
01810        ximin= real(iminx)
01811        ximax= real(imaxx)
01812        call svstat (savcom) ! verwende TCS-Clipping
01813        call lintrn
01814        call dwindo (ximin,ximax,real(iminy),real(imaxy))
01815        call twindo (iminx,imaxx,iminy,imaxy)
01816
01817        if (ishade .ge. 8) then ! Bit3: diagonal fallend
01818         idely= iminx-imaxx
01819         iymax= imaxy+imaxx-iminx
01820         i= iminy+lspace
01821 120     continue ! repeat ...
01822          call movea (ximin,real(i))
01823          call drawa (ximax,real(i+idely))
01824          i= i+lspace
01825         if (i .lt. iymax) goto 120 ! ... until
01826         ishift= ishade -8
01827        else
01828         ishift= ishade
01829        end if
01830
01831        if (ishift .ge. 4) then ! Bit2: diagonal steigend
01832         idely= imaxx-iminx
01833         iymax= real(imaxy)
01834         i= iminy - idely + lspace
01835 130     continue ! repeat...
01836          call movea (ximin,real(i))
01837          call drawa (ximax,real(i+idely))
01838          i= i+lspace
01839         if (i .lt. iymax) goto 130 ! ...until
01840        end if
01841        call restat (savcom)
01842       end if ! Diagonalen
01843       return
01844       end
01845
01846
01847
01848 C  Zeichnen von Symbolen
01849
01850       subroutine bsyms (x,y,isym)
01851       implicit none
01852       real x,y
01853       integer isym
01854       include 'G2dAG2.fd'
01855
01856       if (isym .ge. 0) then
01857        call symout (isym, csizes)
01858       else
01859        call users (x,y,isym)
01860       end if
01861       call movea (x,y)
01862       return
01863       end
01864
01865
01866
```

```
01867       subroutine symout (isym,fac)
01868       implicit none
01869       integer isym
01870       real fac
01871       integer ix,iy, ihorz,ivert
01872
01873       call seeloc (ix,iy)
01874       if (isym .gt. 127) then
01875        call softek (isym)
01876       else if (isym .ge. 33) then
01877        call csize (ihorz,ivert)
01878        ihorz= int( real(ihorz)*.3572)
01879        ivert= int( real(ivert)*.3182)
01880        call movrel (-ihorz,-ivert)
01881        call alfmod
01882        call toutpt (isym)
01883       else if (isym .le. 11) then
01884        call teksym (isym,fac)
01885       end if
01886       call movabs (ix,iy)
01887       return
01888       end
01889
01890
01891
01892       subroutine teksym (isym,amult)
01893       implicit none
01894       integer isym
01895       real amult
01896       integer ihalf, ifull
01897
01898       ihalf= nint(8.* amult)
01899       ifull=ihalf * 2
01900       if (isym .eq. 1) then ! Kreis
01901        call teksym1 (0, 360, 30, 8.*amult)
01902       else if (isym .eq. 2) then ! X
01903        call movrel (ihalf,ihalf)
01904        call drwrel (-ifull,-ifull)
01905        call movrel (0,ifull)
01906        call drwrel (ifull,-ifull)
01907       else if (isym .eq. 3) then ! Dreieck
01908        call teksym1 (90, 450, 120, 8.*amult)
01909       else if (isym .eq. 4) then ! Quadrat
01910        call teksym1 (45, 405, 90, 8.*amult)
01911       else if (isym .eq. 5) then ! Stern
01912        call teksym1 (90, 810, 144, 8.*amult)
01913       else if (isym .eq. 6) then ! Raute
01914        call teksym1 (90, 450, 90, 8.*amult)
01915       else if (isym .eq. 7) then ! vertikaler Balken
01916        call teksym1 (90, 270, 180, 8.*amult)
01917       else if (isym .eq. 8) then ! Kreuz
01918        call movrel (0,ihalf)
01919        call drwrel (0,-ifull)
01920        call movrel (-ihalf,ihalf)
01921        call drwrel (ifull,0)
01922       else if (isym .eq. 9) then ! Pfeil nach oben
01923        call drwrel (-2,-6)
01924        call drwrel (4,0)
01925        call drwrel (-2,6)
01926        call drwrel (0,-ifull)
01927       else if (isym .eq. 10) then ! Pfeil nach unten
01928        call drwrel (-2,6)
01929        call drwrel (4,0)
01930        call drwrel (-2,-6)
01931        call drwrel (0,ifull)
01932       else if (isym .eq. 11) then ! Durchstreichung
01933        call teksym1 (270, 630, 120, 8.*amult)
01934       end if
01935       return
01936       end
01937
01938
01939
01940       subroutine teksym1 (istart, iend, incr, siz)
01941       implicit none
01942       integer istart, iend, incr
01943       real siz
01944       integer i, mx,my,mix,miy
01945       real b
01946
01947       b= real(istart)*.01745
01948       mx= nint(siz*cos(b))
01949       my= nint(siz*sin(b))
01950       call movrel (mx,my)
01951       do 100 i= istart+incr, iend, incr
01952        b= real(i)*.01745
01953        mix= nint(siz*cos(b))
```

```
01954         miy= nint(siz*sin(b))
01955         call drwrel (mix-mx,miy-my)
01956         mx= mix
01957         my= miy
01958 100    continue
01959        return
01960        end
01961
01962
01963
01964 C Netz und Ticmarks
01965
01966        subroutine grid
01967        implicit none
01968        integer i, mlim
01969        real xyext,xyextm, tintvl,tmntvl
01970        include 'G2dAG2.fd'
01971
01972        if (cxyfrm(2) .ne. 0) then ! Zeichnen der y-Achse
01973         i= min0(cxysmin(1),cxysmax(1)) + cxyloc(2)
01974         call movabs (i, cxysmax(2))
01975         call drwabs (i, cxysmin(2))
01976         if (cxybeg(2) .ne. cxyend(2)) then ! Zeichnen y-Ticmarks
01977          i= cxylab(2) ! Labeltyp
01978          if (i .eq. 1) i= cxytype(2) ! =1: Typ entsprechend Daten
01979          if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
01980           if(cxytics(2) .ne. 0) then
01981            tintvl= real(cxysmax(2)-cxysmin(2)) / real( cxytics(2))
01982           end if
01983           if (cxymtcs(2) .gt. 0) tmntvl= tintvl / real(cxymtcs(2))
01984           call movabs(cxybeg(2),cxysmin(2))
01985           call drwabs(cxyend(2),cxysmin(2))
01986           xyext= real(cxysmin(2))
01987           do 100, i=1,cxytics(2)
01988            if (cxymbeg(2) .ne. cxymend(2)) then ! Zeichnen Minor Ticmarks
01989             mlim= cxymtcs(2)-1
01990             xyextm= xyext
01991 110         continue ! repeat...
01992             if (mlim.gt.0) then ! ...until mlim <= 0
01993              xyextm= xyextm+tmntvl
01994              call movabs (cxymbeg(2), nint(xyextm))
01995              call drwabs (cxymend(2), nint(xyextm))
01996              mlim=mlim-1
01997              goto 110
01998             else if (mlim. lt. 0) then
01999              call logtix (2,xyext,tintvl,cxymbeg(2),cxymend(2))
02000             end if
02001            end if
02002            xyext= xyext+tintvl
02003            call movabs (cxybeg(2), nint(xyext))
02004            call drwabs (cxyend(2), nint(xyext))
02005 100       continue
02006          end if ! Labtyp=6: Monate
02007         end if ! Ende Zeichnen Ticmarks
02008        end if ! Ende Zeichnen der Achse
02009
02010        if (cxyfrm(1) .ne. 0) then ! Zeichnen der x-Achse
02011         i= min0(cxysmin(2),cxysmax(2)) + cxyloc(1)
02012         call movabs (cxysmin(1), i)
02013         call drwabs (cxysmax(1), i)
02014         if (cxybeg(1) .ne. cxyend(1)) then ! Zeichnen y-Ticmarks
02015          i= cxylab(1) ! Labeltyp
02016          if (i .eq. 1) i= cxytype(1) ! =1: Typ entsprechend Daten
02017          if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
02018           if(cxytics(1) .ne. 0) then
02019            tintvl= real(cxysmax(1)-cxysmin(1)) / real( cxytics(1))
02020           end if
02021           if (cxymtcs(1) .gt. 0) tmntvl= tintvl / real(cxymtcs(1))
02022           call movabs(cxysmin(1), cxybeg(1))
02023           call drwabs(cxysmin(1), cxyend(1))
02024           xyext= real(cxysmin(1))
02025           do 120, i=1,cxytics(1)
02026            if (cxymbeg(1) .ne. cxymend(1)) then ! Zeichnen Minor Ticmarks
02027             mlim= cxymtcs(1)-1
02028             xyextm= xyext
02029 130         continue ! repeat...
02030             if (mlim.gt.0) then ! ...until mlim <= 0
02031              xyextm= xyextm+tmntvl
02032              call movabs (nint(xyextm), cxymbeg(1))
02033              call drwabs (nint(xyextm), cxymend(1))
02034              mlim=mlim-1
02035              goto 130
02036             else if (mlim. lt. 0) then
02037              call logtix (1,xyext,tintvl,cxymbeg(1),cxymend(1))
02038             end if
02039            end if
02040            xyext= xyext+tintvl
```

```
02041           call movabs (nint(xyext), cxybeg(1))
02042           call drwabs (nint(xyext), cxyend(1))
02043 120      continue
02044         end if ! Labtyp=6: Monate
02045        end if ! Ende Zeichnen Ticmarks
02046       end if ! Ende Zeichnen der Achse
02047       return
02048       end
02049
02050
02051
02052       subroutine logtix (nbase,start,tintvl,mstart,mend)
02053       implicit none
02054       integer nbase,mstart,mend
02055       real start, tintvl
02056       integer i, logtic, ihorz, ivert, idx,idy
02057       character*1 loglab
02058       include 'G2dAG2.fd'
02059
02060       call csize (ihorz,ivert)
02061       do 100 i=2,9
02062        write (unit=loglab, fmt='(i1)') i ! Unicodefaehig durch Compilerfeature
02063        logtic= nint(log10(real(i))*tintvl + start)
02064        if (nbase .eq. 1) then ! x-Achse
02065         idx=  -ihorz/3
02066         if  (mstart .gt. mend) then
02067          idy= ivert
02068         else
02069          idy= -ivert
02070         end if
02071         call movabs (logtic,mend)
02072         call drwabs (logtic,mstart)
02073         if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
02074          call movrel (idx,idy)
02075          call toutstc (loglab)
02076         end if
02077
02078        else if (nbase .eq. 2) then ! y-Achse
02079         if  (mstart .gt. mend) then
02080          idx= ihorz
02081         else
02082          idx= -ihorz
02083         end if
02084         idy=  -ivert / 3
02085         call movabs (mend,logtic)
02086         call drwabs (mstart,logtic)
02087        end if
02088
02089        if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
02090         call movrel (idx,idy)
02091         call toutstc (loglab)
02092        end if
02093 100   continue
02094       return
02095       end
02096
02097
02098
02099       subroutine tset (nbase)
02100       implicit none
02101       integer nbase
02102       integer IOTHER
02103       integer otherbase, near, nfar, newloc, nlen
02104       include 'G2dAG2.fd'
02105
02106       otherbase= iother(nbase)
02107       near= min0(cxysmin(otherbase), cxysmax(otherbase))
02108       nfar= max0(cxysmin(otherbase), cxysmax(otherbase))
02109       newloc= near + cxyloc(nbase)
02110       if (cxyfrm(nbase) .ne. 1) then
02111        if (newloc .lt. ((nfar+near)/2)) then
02112         nlen= cxylen(nbase)
02113        else
02114         nlen= -cxylen(nbase)
02115         nfar= near
02116        end if
02117        call tset2 (newloc,nfar,nlen,cxyfrm(nbase),
02118      1                              cxybeg(nbase),cxyend(nbase))
02119       else
02120        cxybeg(nbase)= 0
02121        cxyend(nbase)= 0
02122       end if
02123
02124       if ((cxymfrm(nbase) .ne. 1) .and. (cxymtcs(nbase) .ne. 0)) then
02125        nlen= nlen / 2
02126        call tset2 (newloc,nfar,nlen,cxymfrm(nbase),
02127      1                              cxymbeg(nbase),cxymend(nbase))
```

```
02128       else
02129        cxymbeg(nbase)= 0
02130        cxymend(nbase)= 0
02131       end if
02132       return
02133       end
02134
02135
02136
02137       subroutine tset2 (newloc,nfar,nlen,nfrm,kstart,kend)
02138       implicit none
02139       integer newloc,nfar,nlen,nfrm,kstart,kend
02140
02141       if (nfrm .eq. 3 .or. nfrm .eq. 6) then
02142        kstart= newloc
02143       else
02144        kstart=newloc-nlen
02145       end if
02146       if (kstart .lt. 0) then
02147        kstart= 0
02148       else if (kend .gt. 1023) then
02149        kstart= 1023
02150       end if
02151
02152       if (nfrm .eq. 2) then
02153        kend= newloc
02154       else if (nfrm .eq. 5 .or. nfrm .eq. 6) then
02155        kend = nfar
02156       else
02157        kend=newloc+nlen
02158       end if
02159       if (kend .lt. 0) then
02160        kend= 0
02161       else if (kend .gt. 1023) then
02162        kend= 1023
02163       end if
02164       return
02165       end
02166
02167
02168
02169       subroutine monpos (nbase,iy1,dpos, spos)
02170       implicit none
02171       integer nbase, iy1, spos
02172       integer iy,idays,iubgc1
02173       real dpos
02174
02175       call ymdyd (iy,idays,iy1, nint(dpos)+1,1)
02176       call iubgc (iy,idays, iubgc1)
02177       call gline (nbase, real(iubgc1), spos)
02178       return
02179       end
02180
02181
02182
02183       subroutine gline (nbase,datapt,spos)
02184       implicit none
02185       integer nbase, spos
02186       real datapt
02187       integer i
02188       include 'G2dAG2.fd'
02189
02190       if (nbase .eq. 1) then ! x-Achsengrid
02191        call wincot (datapt,1., spos,i)
02192        if (iabs(cxyend(1)-cxybeg(1)) .ge. 2) then
02193         call movabs(spos,cxybeg(1))
02194         call drwabs(spos,cxyend(1))
02195        end if
02196       else ! y-Achsengrid
02197        call wincot (1.,datapt, i,spos)
02198        if (iabs(cxyend(2)-cxybeg(2)) .ge. 2) then
02199         call movabs(cxybeg(2),spos)
02200         call drwabs(cxyend(2),spos)
02201        end if
02202       end if
02203       return
02204       end
02205
02206
02207
02208 C Label
02209
02210       subroutine label (nbase)
02211       implicit none
02212       integer nbase
02213       logical even, stag
02214       integer i, icv, igap, iquadrant, labtyp, ilim, iposflag, ioff, iy
```

```
02215       integer ispos,isintv, iyear
02216       integer level1, level2
02217       real fnum, fac, dpos, dintv
02218       character *(255) labstr
02219       integer IOTHER
02220       include 'G2dAG2.fd'
02221
02222       labtyp= cxylab(nbase)
02223       if(labtyp .eq. 1) labtyp= cxytype(nbase) ! LabTyp=1: = dataType
02224       if (labtyp .eq. 0) return ! LabTyp=0: keine Label
02225
02226       fac= 10.**(-cxyepon(nbase))
02227
02228       dintv= real(cxystep(nbase)) / real(cxytics(nbase)) ! Zwischenergebnis
02229       isintv= nint(real(cxysmax(nbase)-cxysmin(nbase)) * dintv)
02230       dintv= (cxyamax(nbase)-cxyamin(nbase)) * dintv
02231
02232       call csize (i,icv) ! nur icv = vertikale Hoehe benoetigt
02233       igap= icv / 3
02234       if (nbase.eq.1) igap= 2*igap
02235       if (iabs(cxysmax(iother(nbase))-cxysmin(iother(nbase)))
02236      1                                      .gt. 2* cxyloc(nbase)) then
02237        iquadrant= -1 ! untere Haelfte
02238       else
02239        iquadrant= +1
02240       end if
02241       level1= min0(cxysmax(iother(nbase)),cxysmin(iother(nbase)))
02242      1                           - (igap-icv/3 ) + cxyloc(nbase)
02243      2                       + isign(igap+cxylen(nbase),iquadrant)
02244       level2= level1 + isign(icv+igap, iquadrant)
02245
02246       if (nbase .eq. 1) then ! Label links/zentriert/rechts?
02247        iposflag= 0 ! x-Achse: zentriert
02248       else
02249        iposflag= -iquadrant
02250       end if
02251
02252       stag= cxystag(nbase) .eq. 2 ! Verwendung in Schleife
02253       even= .false.
02254       ilim= cxytics(nbase) + 1
02255
02256       dpos= cxyamin(nbase)
02257       ispos= cxysmin(nbase)
02258
02259       if (iabs(labtyp) .ge. 3 .and. iabs(labtyp) .le. 8) then ! Kalenderdaten
02260        call oubgc (iyear,i,ifix(cxydmin(nbase))) ! i: Tag nicht benoetigt
02261        dpos= dpos+dintv ! 1. Tic ungelabelt
02262        ispos= ispos+isintv
02263        ilim=ilim-1
02264        if (nbase .eq. 1) iposflag= 1 ! x-Achse Kalender: rechtsbuendig
02265       end if
02266
02267       do 100 i=1,ilim, cxystep(nbase)
02268        if ((labtyp .le. 2) .or. (labtyp .ge. 8)) then
02269         fnum= dpos
02270        else ! Kalendertyp ohne Jahr
02271         if (labtyp.eq.3) then ! Tage
02272          fnum= 7.
02273         else if (labtyp.eq.4) then ! Wochen
02274          fnum= 52.
02275         else if (labtyp.eq.5) then ! Periods
02276          fnum= 13.
02277         else if (labtyp.eq.6) then ! Monate
02278          fnum= 12.
02279         else if (labtyp.eq.7) then ! Quartal
02280          fnum= 4.
02281         end if ! Jahr wird wie linear behandelt
02282         fnum= amod(dpos-1.,fnum)+1.
02283        end if
02284
02285        if (labtyp .lt. 0) then
02286         call usesetc (fnum, cxywdth(nbase), nbase, labstr)
02287        else if ((labtyp .eq. 6) .OR. (labtyp .eq. 3)) then
02288         call alfsetc (fnum, labtyp, labstr)
02289         if (cxywdth(nbase) .lt. len(labstr)) then
02290          labstr(cxywdth(nbase)+1:cxywdth(nbase)+1)= char(0)
02291         end if
02292         if (labtyp .eq. 6) call monpos (nbase,iyear,dpos,ispos)
02293        else
02294         call numsetc (fnum*fac,cxywdth(nbase),nbase,labstr)
02295        end if
02296        call justerc (labstr, iposflag, ioff)
02297
02298        if (nbase .eq. 1) then ! x-Achse
02299         iy= level1
02300         if(stag .and. even) iy= level2
02301         even= .not. even
```

```
02302         call notatec (ispos+ioff,iy, labstr)
02303        else ! y-Achse
02304         call notatec (level1+ioff,ispos-igap,labstr)
02305        end if
02306        dpos= dpos+dintv
02307        ispos= ispos+isintv
02308 100   continue ! end do
02309
02310       if ((labtyp .ne. 2) .and. (cxyetyp(2) .ge. 0)) then ! nicht logarithm.
02311        if (nbase .eq. 1) then ! x-Achse
02312         if (stag) level2= level2 + isign(icv+igap,iquadrant)
02313         i=(cxysmin(nbase)+cxysmax(nbase))/2.
02314         iy=level2
02315        else
02316         i= level1
02317         iy= max0(cxysmin(nbase),cxysmax(nbase)) +icv+igap
02318        end if
02319        call remlab (nbase,cxyloc(nbase),labtyp,i,iy)
02320       end if
02321       return
02322       end
02323
02324
02325
02326       subroutine numsetc (fnum,iwidth,nbase, outstr)
02327       implicit none
02328       real fnum
02329       integer iwidth,nbase
02330       character outstr *(*)
02331       integer iexp
02332       include 'G2dAG2.fd'
02333
02334       if (cxytype(nbase) .eq. 2) then
02335        if (fnum .gt. 0.) then
02336         iexp= fnum + .00005
02337        else if (fnum .lt. 0.) then
02338         iexp= fnum - .00005
02339        else
02340         iexp= 0
02341        end if
02342        call expoutc (nbase,iexp, outstr)
02343       else if ((cxytype(nbase).eq.1) .and. (cxydec(nbase).gt.0)) then
02344        call fformc (fnum,iwidth, cxydec(nbase), outstr)
02345       else
02346        call iformc (fnum,iwidth, outstr)
02347       end if
02348       return
02349       end
02350
02351
02352
02353       subroutine iformc (fnum,iwidth, outstr)
02354       implicit none
02355       real fnum
02356       integer iwidth
02357       character outstr *(*)
02358       character fmtstr *(11)
02359
02360       if (iwidth .le. 0) then ! iwidth=0: ohne Label
02361        outstr= char(0)
02362        return
02363       end if
02364
02365       if (iwidth .gt. 99) goto 200 ! Errorhandler
02366       write (unit=fmtstr,fmt=100, err=200) iwidth
02367       if (len(outstr) .gt. iwidth) then
02368        write (unit= outstr, fmt=fmtstr, err=200) nint(fnum),0 ! 0: End of String
02369       else
02370        write (unit= outstr, fmt=fmtstr, err=200) nint(fnum) ! evtl. ohne EoS?
02371       end if
02372
02373       return
02374
02375 200   continue ! Error Handler
02376       outstr= '?I?'
02377       if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
02378       return
02379
02380 100   format ('(SS,I' ,i2.2, ',A1)')
02381       end
02382
02383
02384
02385       subroutine fformc (fnum,iwidth,idec, outstr)
02386       implicit none
02387       real fnum
02388       integer iwidth,idec
```

```
02389       character outstr *(*)
02390       integer nDgtM
02391       real fa
02392       include 'G2dAG2.fd'
02393
02394       ndgtm= iwidth-idec
02395       if (fnum .ge. 0.) then
02396        ndgtm= ndgtm -1  ! Ziffern Mantisse
02397       else
02398        ndgtm= ndgtm-2   ! 1 Ziffer Vorzeichen
02399       end if
02400       fa= abs(fnum) ! Skalierung mindestens 2 signfikante Stellen: .1*abs(fnum)
02401
02402       if ( (((fa .lt. 10./cinfin) .or. (fa .gt. .1**idec)) ! Zahl mit Dezimalen darstellbar
02403      1                         .and.(fa .lt. 10.**ndgtm))   ! Zahl mit Mantisse darstellbar
02404      2      .or. ((iwidth.lt.idec+7))              ) then  ! oder Laenge zu kurz fuer E-Format
02405         call fonlyc (fnum,iwidth,idec, outstr)
02406       else
02407        call eformc (fnum,iwidth,idec, outstr)
02408       end if
02409       return
02410       end
02411
02412
02413
02414       subroutine fonlyc (fnum,iwidth,idec, outstr)
02415       implicit none
02416       real fnum
02417       integer iwidth,idec
02418       character outstr *(*)
02419       character fmtstr *(14)
02420
02421       if (iwidth .le. 0) then ! iwidth=0: ohne Label
02422        outstr= char(0)
02423        return
02424       end if
02425
02426       if ((idec .gt. iwidth-1) .or. (iwidth .gt. 99)) goto 200 ! Errorhandler
02427       write (unit=fmtstr,fmt=100, err=200) iwidth,idec
02428       if (len(outstr) .gt. iwidth) then
02429        write (unit= outstr, fmt=fmtstr, err=200) fnum,0 ! 0: End of String
02430       else
02431        write (unit= outstr, fmt=fmtstr, err=200) fnum ! evtl. ohne EoS?
02432       end if
02433       return
02434
02435 200   continue ! Error Handler
02436       outstr= '?F?'
02437       if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
02438       return
02439
02440 100   format ('(SS,F' ,i2.2,'.', i2.2,',A1)')
02441       end
02442
02443
02444
02445       subroutine eformc (fnum,iwidth,idec, outstr)
02446       implicit none
02447       real fnum
02448       integer iwidth,idec
02449       character outstr *(*)
02450       integer iexpon
02451       character fmtstr *(18)
02452
02453       if (iwidth .le. 0) then ! iwidth=0: ohne Label
02454        outstr= char(0)
02455        return
02456       end if
02457
02458       call esplit (fnum,iwidth,idec,iexpon)
02459       if ((idec .gt. iwidth-7) .or. (iwidth .gt. 99)) goto 200 ! Errorhandler
02460       write (unit=fmtstr,fmt=100, err=200) iwidth-idec-6,iwidth,iwidth-7
02461       if (len(outstr) .gt. iwidth) then
02462        write (unit= outstr, fmt=fmtstr, err=200) fnum,0 ! 0: End of String
02463       else
02464        write (unit= outstr, fmt=fmtstr, err=200) fnum ! evtl. ohne EoS?
02465       end if
02466       return
02467
02468 200   continue ! Error Handler
02469       outstr= '?E?'
02470       if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
02471       return
02472
02473 100   format ('(SS,' ,i2.2,'P,E' ,i2.2,'.', i2.2,',A1)')
02474       end
02475
```

```
02476
02477
02478       subroutine esplit (fnum,iwidth,idec,iexpon)
02479       implicit none
02480       real fnum
02481       integer iwidth,idec,iexpon
02482       real fabs
02483       include 'G2dAG2.fd'
02484
02485       fabs= abs(fnum)
02486       if (fabs .ge. 1.) then
02487        iexpon= ifix( alog10(fabs)+1.000005) - iwidth+idec+6 ! 6: Vorz.-Pkt-Exp(4)
02488       else if (fabs .ge. 10./cinfin) then
02489        iexpon= alog10(fabs)
02490       else
02491        iexpon= -alog10(cinfin)
02492       end if
02493       return
02494       end
02495
02496
02497
02498       subroutine expoutc (nbase,iexp, outstr)
02499       implicit none
02500       integer nbase,iexp, i, iL, nexp
02501       character outstr *(*), tmpstr *(4)
02502       include 'G2dAG2.fd'
02503
02504       il= len(outstr)
02505       nexp= abs(iexp)
02506
02507       if ( (cxyetyp(nbase).eq.2) .and. (il.gt. 5)
02508      1             .and. (mod(nexp,3) .eq. 0)
02509      2             .and. (iexp.ge.1)  .and. (iexp.le.9) ) then ! MMMs
02510        do 20 i=3,nexp,3
02511         outstr(i/3:i/3)= 'M'
02512 20     continue
02513        outstr(nexp/3+1:)= char(39) // 'S' // char(0)
02514
02515       else if ( (cxyetyp(nbase).eq.3) .and. (il.gt.17)
02516      1                  .and. (iexp.ge.1) .and. (iexp.le.6)) then ! TENS
02517        if (nexp .eq. 1) then
02518         outstr= 'TENS' // char(0)
02519        else if (nexp .eq. 2) then
02520         outstr= 'HUNDREDS' // char(0)
02521        else if (nexp .eq. 3) then
02522         outstr= 'THOUSANDS' // char(0)
02523        else if (nexp .eq. 4) then
02524         outstr= 'TEN THOUSANDS' // char(0)
02525        else if (nexp .eq. 5) then
02526         outstr= 'HUNDRED THOUSANDS' // char(0)
02527        else if (nexp .eq. 6) then
02528         outstr= 'MILLIONS' // char(0)
02529        end if
02530       else if( (cxyetyp(nbase).eq.4) ! 10000
02531      1    .and. (iexp.ge.1)  .and. (iexp.le.9)
02532      2                  .and. (il.ge.nexp+2)) then
02533        do 30 i=2,nexp+1
02534         outstr(i:i)= '0'
02535  30    continue
02536        outstr(1:1)= '1'
02537        outstr(nexp+2:)= char(0)
02538
02539       else if (il .gt. 7) then ! Default: Superscript EXP
02540        if (iexp .ne. 1) then
02541         if (nexp .lt. 10) then
02542          i=1
02543         else
02544          i=2
02545         end if
02546         if (iexp .lt. 0) then
02547          i= i+1
02548         end if
02549         call iformc (real(iexp), i, tmpstr)
02550        else
02551         tmpstr= char(0) ! 10 wird ohne Exponenten 1 ausgegeben
02552        end if
02553        if (iexp .ne. 0) then
02554         if (cxytype(nbase) .ne. 2) then
02555          outstr(1:1)= 'x'
02556          i= 2
02557         else
02558          i= 1
02559         end if
02560         outstr(i:)= '10' // char(1) ! Index UP
02561         outstr(i+3:)= tmpstr ! char(0) wird bei IFORMC angehaengt
02562        else
```

```
02563         outstr(1:)= '1' // char(0) ! 1 wird nicht als 10**0 ausgegeben
02564        end if
02565       else ! outstr zu kurz
02566        outstr= '?X?'
02567       end if
02568
02569       return
02570       end
02571
02572
02573
02574       subroutine alfsetc (fnum, labtyp, string)
02575       implicit none
02576       integer inum, labtyp
02577       real fnum
02578       character *(*) string
02579
02580       inum= fnum + .001 ! truncate real to integer
02581       if (labtyp .eq. 3) then ! Tage
02582        if ((inum .eq. 0) .or. (inum .eq. 7)) then
02583         string= 'MONDAY' // char(0)
02584        else if (inum .eq. 1) then
02585         string= 'TUESDAY' // char(0)
02586        else if (inum .eq. 2) then
02587         string= 'WEDNESDAY' // char(0)
02588        else if (inum .eq. 3) then
02589         string= 'THURSDAY' // char(0)
02590        else if (inum .eq. 4) then
02591         string= 'FRIDAY' // char(0)
02592        else if (inum .eq. 5) then
02593         string= 'SATURDAY' // char(0)
02594        else if (inum .eq. 6) then
02595         string= 'SUNDAY' // char(0)
02596        end if
02597       else if (labtyp .eq. 6) then ! Monate
02598        if (inum .eq. 1) then
02599         string= 'JANUARY' // char(0)
02600        else if (inum .eq. 2) then
02601         string= 'FEBRUARY' // char(0)
02602        else if (inum .eq. 3) then
02603         string= 'MARCH' // char(0)
02604        else if (inum .eq. 4) then
02605         string= 'APRIL' // char(0)
02606        else if (inum .eq. 5) then
02607         string= 'MAY' // char(0)
02608        else if (inum .eq. 6) then
02609         string= 'JUNE' // char(0)
02610        else if (inum .eq. 7) then
02611         string= 'JULY' // char(0)
02612        else if (inum .eq. 8) then
02613         string= 'AUGUST' // char(0)
02614        else if (inum .eq. 9) then
02615         string= 'SEPTEMBER' // char(0)
02616        else if (inum .eq. 10) then
02617         string= 'OCTOBER' // char(0)
02618        else if (inum .eq. 11) then
02619         string= 'NOVEMBER' // char(0)
02620        else if (inum .eq. 12) then
02621         string= 'DECEMBER' // char(0)
02622        end if
02623       end if
02624       return
02625       end
02626
02627
02628
02629       subroutine notatec (ix,iy, string)
02630       implicit none
02631       integer ix, iy
02632       character *(*) string
02633       integer i, iv, is
02634       integer ISTRINGLEN
02635
02636       call csize(i,iv)        ! nur iv benoetigt
02637       call movabs(ix,iy)
02638
02639       is= 1
02640       do 100 i=1, istringlen(string)
02641        if (string(i:i) .lt. char(31) ) then
02642         if (i.gt.is) call toutstc (string(is:i-is))
02643         if (string(i:i) .eq. char(1)) call movrel (0, iv/2)  ! Hochindex
02644         if (string(i:i) .eq. char(2)) call movrel (0, -iv/2) ! Index
02645         is= i+1
02646        end if
02647 100   continue
02648       if (is .le. istringlen(string)) call toutstc (string(is:))
02649       return
```

```
02650       end
02651
02652
02653
02654       subroutine vlablc (string)
02655 C
02656 C  Sollte in das TCS verlagert werden, um vertikale Schrift zu erzeugen
02657 C
02658       implicit none
02659       character string*(*)
02660       integer i, icy, ix,iy
02661       integer ISTRINGLEN
02662
02663       if (istringlen(string) .le. 0) return
02664       call csize (i,icy)
02665       call seeloc (ix,iy)
02666       do 100 i=1,istringlen(string)
02667        iy= iy-icy
02668        if (iy .lt. 0) return
02669        call movabs (ix,iy)
02670        call toutpt (ichar(string(i:i)))
02671 100   continue
02672       return
02673       end
02674
02675
02676
02677       subroutine justerc (string, iPosFlag, iOff)
02678       implicit none
02679       integer iPosFlag, iOff
02680       character string*(*)
02681       integer i, iLen, nCtrl
02682       integer ISTRINGLEN, LINWDT
02683
02684       ilen= istringlen(string)
02685       nctrl= 0    ! Zaehlen der Ctrlcharacter
02686       do 100 i=1, ilen
02687        if (string(i:i) .lt. char(31) ) nctrl= nctrl+1
02688 100   continue
02689
02690       if (iposflag .lt. 0) then ! linksbuendig
02691        ioff= 0
02692       else ! rechtsbuendig und zentriert
02693        ioff= -linwdt((ilen-nctrl)*8-2)/8       ! rechtsbuendig
02694        if (iposflag.eq.0) ioff= ioff / 2        ! zentriert
02695       end if
02696
02697       return
02698       end
02699
02700
02701
02702       subroutine width (nbase)
02703       implicit none
02704       integer nbase
02705       integer labtyp
02706       include 'G2dAG2.fd'
02707
02708       labtyp= cxylab(nbase)
02709       if(labtyp .eq. 1) labtyp= cxytype(nbase) ! LabTyp=1: = dataType
02710
02711       if ((cxywdth(nbase).ne.0) .and. (labtyp.ne.1)) return ! Manuelle Vorgabe nichtlinear
02712
02713       if (labtyp.le.1) then ! lineare Achsen und anwenderdefinierte Label
02714        call lwidth (nbase)
02715
02716       else if (labtyp .eq. 2) then ! logarithmische Achsen
02717        if (cxyetyp(nbase) .le. 1) then ! 10 mit Exponent
02718         cxywdth(nbase)= 6
02719        else if (cxyetyp(nbase) .eq. 2) then ! M, MM...
02720         cxywdth(nbase)= int(alog10(abs(cxydmax(nbase)))/3. ) + 6
02721        else if (cxyetyp(nbase) .eq. 3) then ! Ausgeschriebene Worte
02722         cxywdth(nbase)= 20
02723         cxystep(nbase)= 1
02724         cxystag(nbase)= 2
02725        else if (cxyetyp(nbase) .eq. 4) then ! 1 mit 0
02726         cxywdth(nbase)= max(abs(alog10(abs(cxydmin(nbase)))),
02727      1                      abs(alog10(abs(cxydmin(nbase)))) ) + 2
02728        end if
02729
02730       else if (labtyp .gt. 2) then ! Kalenderachsen
02731        if ((labtyp .eq. 3) .or. (labtyp .eq. 6)) then ! Tage oder Monate
02732         cxywdth(nbase)= 9
02733        else
02734         cxywdth(nbase)= 4
02735        end if
02736       end if
```

```
02737
02738       return
02739       end
02740
02741
02742
02743       subroutine lwidth (nbase)
02744       implicit none
02745       integer nbase
02746       integer iadj, most, least, isign,iwidth, idelta, ndec, iexp
02747       real xmax
02748       real ROUNDD
02749       include 'G2dAG2.fd'
02750
02751       iadj= 0
02752       xmax= amax1(abs(cxydmin(nbase)),abs(cxydmax(nbase)))
02753       if (xmax .gt. 1.) then
02754        most= int(alog10(xmax) + 1.00005) ! Position Most Significant Digit
02755        iadj= 1
02756       else if (xmax .eq. 1.) then
02757        most= 0
02758       else
02759        most= int(alog10(xmax) - 0.00005)
02760       end if
02761
02762       ndec= cxydec(nbase)
02763       if (cxydec(nbase) .ne. 0) then ! Anzahl Dezimalstellen vorgegeben
02764        least= -ndec ! Entspricht Position LeastSignificant Digit
02765       else
02766        least= cxylsig(nbase)
02767       end if
02768
02769       if (cxydmin(nbase) .lt. 0.) then
02770        isign=1    ! 1 Buchstabe Vorzeichen
02771       else
02772        isign=0
02773       end if
02774
02775       if ((most .lt. 0) .or. (least .ge. 0)) then
02776        iwidth= max0(1,most)- min0(0,least) + isign
02777        if (most .lt. 0) iwidth= iwidth+1 ! 1 Dezimalpunkt
02778        if ((iwidth .gt. 5 ) .and. (cxyetyp(nbase) .ge. 0)) then
02779         if (cxyetyp(nbase).eq.2) then
02780          iexp= int( roundd(real(most-iadj),3.))
02781         else
02782          iexp= int( roundd(real(most-iadj),1.))
02783         end if
02784         iwidth= most-least+isign+ 2
02785         ndec= max0(0,iexp-least+iadj)
02786        else
02787         ndec= max(0,-least)
02788         iexp= 0
02789        end if
02790       else
02791        iexp= 0
02792        ndec= max(0,-least)
02793        iwidth= most-least+isign+1
02794        if (most .eq. 0) iwidth= iwidth+1 ! Einbezug fuehrende Null
02795       end if
02796
02797       if ((cxywdth(nbase) .ne. 0).and.(cxywdth(nbase).lt. iwidth)) then
02798        idelta= iwidth - cxywdth(nbase) - ndec
02799        if ((ndec .gt. 0) .and. (idelta .lt. 1) ) then
02800         ndec= max0(0,-idelta)
02801         iwidth= cxywdth(nbase)
02802        else
02803         iexp= iexp+idelta
02804         if(ndec .gt. 0) iexp=iexp-1
02805         iwidth= cxywdth(nbase)
02806         ndec=0
02807        end if
02808       end if
02809
02810       cxywdth(nbase)= iwidth
02811       cxydec(nbase)= ndec
02812       cxyepon(nbase)= iexp
02813       return
02814       end
02815
02816
02817
02818       subroutine remlab (nbase,iloc,labtyp,ix,iy)
02819       implicit none
02820       integer nbase, iloc, labtyp, ix, iy
02821       integer iyear1,iday1, iyear2,iday2
02822       integer iyear,imon,iday, ioff, iposflag
02823       character label *(25)
```

```
02824       include 'G2dAG2.fd'
02825
02826       if (iabs(labtyp) .eq. 1) then ! lineare Daten
02827        if (cxyepon(nbase) .eq. 0) return ! kein Exponent
02828        call expoutc (nbase,cxyepon(nbase), label)
02829       else ! Kalenderdaten
02830        if ((labtyp .ge. 4) .and. (labtyp.ne.6)) then ! Wochen, Quartale, Jahre
02831         ioff= 4 ! Überlappung der Jahre vermeiden
02832        else
02833         ioff= 0
02834        end if
02835        call oubgc (iyear1,iday1, nint(cxydmin(nbase))+ioff)
02836        call oubgc (iyear2,iday2, nint(cxydmax(nbase))-ioff)
02837        if (iday2 .le. 1) iyear2=iyear2-1
02838        iday2=iday2-1
02839        call ydymd(iyear1,iday1,iyear,imon,iday)
02840
02841        if (iabs(labtyp).eq. 3) then
02842         call iformc (real(iday), 2, label(1:2))
02843         label(3:3)= ' ' ! 'dd '
02844         call alfsetc (real(imon), 6, label(4:6)) ! labtyp 6= Monate, Laenge 3
02845         label(7:7)= ' ' ! 'dd mmm '
02846         call iformc (real(iyear), 4, label(7:10)) ! 'dd mm yyyy'
02847         label(11:11)= char(0) ! evtl. Labelende
02848         if (iyear1 .lt. iyear2) then ! bei Bedarf Start und Endjahr
02849          label(11:11)= '-' ! 'dd mm yyyy-'
02850          call ydymd(iyear2,iday2,iyear,imon,iday)
02851          call iformc (real(iday), 2, label(12:13)) ! 'dd'
02852          label(14:14)= ' ' ! 'dd mm yyyy-dd '
02853          call alfsetc (real(imon), 6, label(15:17)) ! 'dd mmm'
02854          label(18:18)= ' ' ! 'dd mm yyyy-dd mmm '
02855          call iformc (real(iyear), 4, label(19:22)) ! 'dd mm yyyy-'
02856          label(23:23)= char(0)
02857         end if
02858        else
02859         call iformc (real(iyear), 4, label(1:4)) ! 'yyyy'
02860         label(5:5)= char(0)
02861         if (iyear1 .lt. iyear2) then ! bei Bedarf Start und Endjahr
02862          label(5:5)= '-' ! 'yyyy-'
02863          call iformc (real(iyear2), 4, label(6:9)) ! 'yyyy-yyyy'
02864          label(10:10)= char(0)
02865         end if
02866        end if
02867       end if
02868
02869       if ((nbase.eq.1) .or. (iloc.eq.1)) then ! X-Achse oder y Zentriert
02870        iposflag= 0
02871       else
02872        iposflag= isign(1,1-iloc)
02873       end if
02874       call justerc (label, iposflag, ioff)
02875       call notatec (ix+ioff, iy,label)
02876       return
02877       end
02878
02879
02880
02881       subroutine spread (nbase)
02882       implicit none
02883       integer nbase
02884       integer ih, labtyp, iwidth, iMaxWid
02885       integer LINWDT
02886       include 'G2dAG2.fd'
02887
02888       if (cxystag(nbase) .ne. 1) return
02889
02890       labtyp= cxylab(nbase)
02891       if ((labtyp .eq. 1) .or. (labtyp .eq. 0)) labtyp= cxytype(nbase)
02892
02893 100   continue ! outer loop
02894        if (nbase .eq. 1) then ! x-Achse
02895         iwidth= linwdt(cxywdth(nbase))
02896        else
02897         call csize(ih, iwidth)
02898        end if
02899
02900        imaxwid= iabs(cxysmax(nbase)-cxysmin(nbase))- 2*iwidth
02901        imaxwid= imaxwid* cxystep(nbase)* cxystag(nbase) / cxytics(nbase)
02902
02903        cxystep(nbase)= 1
02904        cxystag(nbase)= 1
02905
02906        if (iwidth .lt. imaxwid) return ! exit loop
02907
02908        if (nbase .eq. 1) then ! x-Achse
02909         cxystag(nbase)= 2
02910        else
```

```
02911          cxystep(nbase)= cxystep(nbase) + 1
02912         end if
02913
02914 110     continue ! inner loop
02915          if(iwidth .lt. imaxwid) return ! exit loop
02916          if(cxystep(nbase) .gt. cxytics(nbase)) return ! exit loop
02917         if (labtyp .ne. 3 .and. labtyp .ne. 6) then ! cycle inner loop
02918          cxystep(nbase)= cxystep(nbase)+1
02919          goto 110
02920        else ! cycle outer loop
02921         if (cxywdth(nbase) .eq. 3) return
02922         cxywdth(nbase)=3
02923         goto 100
02924        end if ! cycle until force exit
02925        end
02926
02927
02928
02929 C
02930 C  Tabellensuche und Rundungen
02931 C
02932
02933        real function findge (val,tab,in)
02934        implicit none
02935        integer in
02936        real val, tab(1)
02937
02938 100    if (tab(in) .lt. val) goto 110 ! while
02939         in= in-1
02940         goto 100
02941 110    continue ! endwhile
02942
02943 120    continue ! repeat
02944         in= in+1
02945        if (tab(in) .lt. val) goto 120 ! end repeat
02946        findge= tab(in)
02947        return
02948        end
02949
02950
02951
02952        real function findle (val,tab,in)
02953        implicit none
02954        integer in
02955        real val, tab(1)
02956        real valeps
02957
02958        valeps= val+ 1.e-7 ! Vergleich um 0 ermoeglichen (Rechengenauigkeit!)
02959
02960 100    if (tab(in) .le. valeps) goto 110 ! while
02961         in= in-1
02962         goto 100
02963 110    continue ! endwhile
02964
02965 120    continue ! repeat
02966         in= in+1
02967        if (tab(in) .lt. valeps) goto 120 ! end repeat
02968        findle= tab(in-1)
02969        return
02970        end
02971
02972
02973
02974        integer function locge (ival,itab,iN)
02975        implicit none
02976        integer ival, itab(1), in
02977
02978 100    if (itab(in) .lt. ival) goto 110 ! while
02979         in= in-1
02980         goto 100
02981 110    continue ! endwhile
02982
02983 120    continue ! repeat
02984         in= in+1
02985        if (itab(in) .lt. ival) goto 120 ! end repeat
02986        locge= itab(in)
02987        return
02988        end
02989
02990
02991
02992        integer function locle (ival,itab,iN)
02993        implicit none
02994        integer ival, itab(1), in
02995
02996 100    if (itab(in) .le. ival) goto 110 ! while
02997         in= in-1
```

```
02998         goto 100
02999 110    continue ! endwhile
03000
03001 120    continue ! repeat
03002         in= in+1
03003        if (itab(in) .le. ival) goto 120 ! end repeat
03004        locle= itab(in-1)
03005        return
03006        end
03007
03008
03009
03010        real function roundd (value,finterval)
03011        implicit none
03012        real value,finterval
03013        integer ifrac
03014        real frac
03015
03016        frac= value/finterval
03017        ifrac= int(frac)
03018        if (real(ifrac) .gt. frac) ifrac= ifrac-1 ! Abrunden bei frac neg.
03019        roundd = real(ifrac) * finterval
03020        if (roundd .gt. value) roundd= value
03021        return
03022        end
03023
03024
03025
03026        real function roundu (value,finterval)
03027        implicit none
03028        real value,finterval
03029        integer ifrac
03030        real frac
03031
03032        frac= value/finterval
03033        ifrac= int(frac)
03034        if (real(ifrac) .lt. frac) ifrac= ifrac+1 ! Aufrunden bei frac pos.
03035        roundu = real(ifrac) * finterval
03036        if (roundu .lt. value) roundu= value
03037        return
03038        end
03039
03040
03041
03042 C
03043 C  Generelle Manipulationen der Commonvariablen
03044 C
03045        subroutine savcom (Array)
03046        implicit none
03047        integer array(1)
03048        include 'G2dAG2.fd'
03049
03050        integer i
03051        integer arr(1)
03052        equivalence(arr(1),cline)
03053        do 10 i=1,g2dag2l
03054         array(i)= arr(i)
03055 10     continue
03056        return
03057        end
03058
03059
03060
03061        subroutine rescom (Array)
03062        implicit none
03063        integer array(1)
03064        include 'G2dAG2.fd'
03065
03066        integer i
03067        integer arr(1)
03068        equivalence(arr(1),cline)
03069        do 10 i=1,g2dag2l
03070         arr(i)= array(i)
03071 10     continue
03072        return
03073        end
03074
03075
03076
03077        integer function iother (ipar)
03078        implicit none
03079        integer ipar
03080
03081        if (mod(ipar,2) .eq. 1) then ! ungerader Parameter=x-Achse
03082         iother= ipar+1
03083        else
03084         iother= ipar-1
```

```
03085       end if
03086       return
03087       end
```

3.3 AG2Holerith.for File Reference

Graph2D: deprecated AG2 routines.

Functions/Subroutines

- subroutine notate (ix, iy, lenchr, iarray)
- subroutine alfset (fnum, kwidth, labtyp, ilabel)
- subroutine numset (fnum, iwidth, nbase, ilabel, ifill)
- subroutine expout (nbase, iexp, ilabel, nchars, ifill)
- subroutine hstrin (iString)
- subroutine hlabel (iLen, iString)
- subroutine vstrin (iarray)
- subroutine vlabel (iLen, iString)
- subroutine juster (iLen, iString, iposflag, ifill, lenchr, ioff)
- subroutine eform (fnum, iwidth, idec, ilabel, ifill)
- subroutine fform (fnum, iwidth, idec, ilabel, ifill)
- subroutine fonly (fnum, iwidth, idec, ilabel, ifill)
- subroutine iform (fnum, iwidth, ilabel, ifill)
- integer function ibasec (iPar)
- integer function ibasex (ipar)
- integer function ibasey (ipar)
- real function comget (iPar)
- subroutine comset (iPar, val)
- subroutine comdmp

3.3.1 Detailed Description

Graph2D: deprecated AG2 routines.

Version

2.2

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Compatibility routines dealing with holerith characters and direct manipulation of common variables.

Definition in file AG2Holerith.for.

3.3.2 Function/Subroutine Documentation

3.3.2.1 alfset()

```
subroutine alfset (
             real fnum,
             integer kwidth,
             integer labtyp,
             integer, dimension(kwidth) ilabel )
```

Definition at line 45 of file AG2Holerith.for.

3.3.2.2 comdmp()

```
subroutine comdmp
```

Definition at line 328 of file AG2Holerith.for.

3.3.2.3 comget()

```
real function comget (
             integer iPar )
```

Definition at line 271 of file AG2Holerith.for.

3.3.2.4 comset()

```
subroutine comset (
             integer iPar,
             real val )
```

Definition at line 299 of file AG2Holerith.for.

3.3.2.5 eform()

```
subroutine eform (
             real fnum,
             integer iwidth,
             integer idec,
             integer, dimension(iwidth) ilabel,
             integer ifill )
```

Definition at line 173 of file AG2Holerith.for.

3.3.2.6 expout()

```
subroutine expout (
             integer nbase,
             integer iexp,
             integer, dimension(nchars) ilabel,
             integer nchars,
             integer ifill )
```

Definition at line 90 of file AG2Holerith.for.

3.3.2.7 fform()

```
subroutine fform (
             real fnum,
             integer iwidth,
             integer idec,
             integer, dimension(255) ilabel,
             integer ifill )
```

Definition at line 189 of file AG2Holerith.for.

3.3.2.8 fonly()

```
subroutine fonly (
             real fnum,
             integer iwidth,
             integer idec,
             integer, dimension(iwidth) ilabel,
             integer ifill )
```

Definition at line 205 of file AG2Holerith.for.

3.3.2.9 hlabel()

```
subroutine hlabel (
             integer iLen,
             integer, dimension(ilen) iString )
```

Definition at line 121 of file AG2Holerith.for.

3.3.2.10 hstrin()

```
subroutine hstrin (
             integer, dimension(2) iString )
```

Definition at line 112 of file AG2Holerith.for.

3.3.2.11 ibasec()

```
integer function ibasec (
             integer iPar )
```

Definition at line 241 of file AG2Holerith.for.

3.3.2.12 ibasex()

```
integer function ibasex (
             integer ipar )
```

Definition at line 251 of file AG2Holerith.for.

3.3.2.13 ibasey()

```
integer function ibasey (
             integer ipar )
```

Definition at line 261 of file AG2Holerith.for.

3.3.2.14 iform()

```
subroutine iform (
             real fnum,
             integer iwidth,
             integer, dimension(iwidth) ilabel,
             integer ifill )
```

Definition at line 221 of file AG2Holerith.for.

3.3.2.15 juster()

```
subroutine juster (
             integer iLen,
             integer, dimension(ilen) iString,
             integer iposflag,
             integer ifill,
             integer lenchr,
             integer ioff )
```

Definition at line 154 of file AG2Holerith.for.

3.3.2.16 notate()

```
subroutine notate (
             integer ix,
             integer iy,
             integer lenchr,
             integer, dimension(lenchr) iarray )
```

Definition at line 30 of file AG2Holerith.for.

3.3.2.17 numset()

```
subroutine numset (
             real fnum,
             integer iwidth,
             integer nbase,
             integer, dimension(iwidth) ilabel,
             integer ifill )
```

Definition at line 67 of file AG2Holerith.for.

3.3.2.18 vlabel()

```
subroutine vlabel (
             integer iLen,
             integer, dimension(ilen) iString )
```

Definition at line 139 of file AG2Holerith.for.

3.3.2.19 vstrin()

```
subroutine vstrin (
             integer, dimension(2) iarray )
```

Definition at line 130 of file AG2Holerith.for.

3.4 AG2Holerith.for

```
00001 C> \file       AG2Holerith.for
00002 C> \version    2.2
00003 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00004 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00005 C> \~german
00006 C> \brief   Graph2D: obsolete AG2 Routinen
00007 C> \~english
00008 C> \brief   Graph2D: deprecated AG2 routines
00009 C> \~
00010 C>
00011 C> \~german
00012 C>     Unterprogramme zur Behandlung von Holerithvariablen und direkter
00013 C>     Manipulation des Commonblocks
00014 C>
00015 C> \~english
00016 C>     Compatibility routines dealing with holerith characters
00017 C>     and direct manipulation of common variables.
00018 C>
00019 C
00020 C
00021 C  Tektronix Advanced Graphics 2 - Version 2.x
00022 C
00023 C     Optionale Unterprogramme
00024 C
00025
00026 C
00027 C Stringfunktionen fuer Holerithvariablen
00028 C
00029
00030       subroutine notate (ix,iy,lenchr,iarray)
00031       implicit none
00032       integer ix,iy,lenchr, iarray(lenchr)
00033       integer i
00034       character *(255) buf
00035
00036       do 100 i=1,lenchr
00037        buf(i:i)= char(iarray(i))
00038 100   continue
00039       call notatec (ix,iy,buf(1:lenchr))
00040       return
00041       end
00042
00043
00044
00045       subroutine alfset (fnum,kwidth,labtyp,ilabel)
00046       implicit none
00047       integer kwidth,labtyp, ilabel(kwidth)
00048       real fnum
00049       integer i, buflen
00050       character *(255) buf
00051       integer ISTRINGLEN
00052
00053       call alfsetc (fnum, labtyp, buf)
00054       buflen= istringlen(buf)
00055       do 100 i=1,kwidth
00056        if (i .le. buflen) then
00057         ilabel(i)= ichar(buf(i:i))
00058        else
00059         ilabel(i)= ichar(' ')
00060        end if
00061 100   continue
00062       return
00063       end
00064
00065
00066
00067       subroutine numset (fnum,iwidth,nbase,ilabel,ifill)
00068       implicit none
00069       integer iwidth,nbase,ilabel(iwidth),ifill
00070       real fnum
00071       integer i, iLeadFill
```

```
00072       character *(255) buf
00073       integer ISTRINGLEN
00074
00075       call numsetc (fnum,iwidth,nbase, buf)
00076       ileadfill= max(0,iwidth-istringlen(buf))
00077       do 100 i=1,iwidth
00078        ilabel(ileadfill+i)= ichar(buf(i:i))
00079 100   continue
00080       i=1 ! iLabel ist rechtsjustiert!
00081       if (i.gt.ileadfill) goto 110 ! while
00082        ilabel(i)= ifill
00083        i= i+1
00084 110   continue ! endwhile
00085       return
00086       end
00087
00088
00089
00090       subroutine expout (nbase,iexp,ilabel,nchars,ifill)
00091       implicit none
00092       integer nbase,iexp, nchars, ilabel(nchars), ifill
00093       integer i, iLeadFill
00094       character *(255) buf
00095       integer ISTRINGLEN
00096
00097       call expoutc (nbase,iexp, buf(1:nchars))
00098       ileadfill= max(0,nchars-istringlen(buf))
00099       do 100 i=1,nchars
00100        ilabel(ileadfill+i)= ichar(buf(i:i))
00101 100   continue
00102       i=1 ! iLabel ist rechtsjustiert!
00103       if (i.gt.ileadfill) goto 110 ! while
00104        ilabel(i)= ifill
00105        i= i+1
00106 110   continue ! endwhile
00107       return
00108       end
00109
00110
00111
00112       subroutine hstrin (iString)
00113       implicit none
00114       integer iString(2)
00115       call anstr (istring(1),istring(2))
00116       return
00117       end
00118
00119
00120
00121       subroutine hlabel (iLen, iString)
00122       implicit none
00123       integer iLen, iString(iLen)
00124       call anstr (ilen, istring)
00125       return
00126       end
00127
00128
00129
00130       subroutine vstrin (iarray)
00131       implicit none
00132       integer iarray(2)
00133       call vlabel (iarray(1),iarray(2))
00134       return
00135       end
00136
00137
00138
00139       subroutine vlabel (iLen,iString)
00140       implicit none
00141       integer iLen, iString(iLen)
00142       integer i
00143       character *(255) buf
00144       integer ISTRINGLEN
00145       do 100 i=1, ilen
00146        buf(i:i)= char(istring(i))
00147 100   continue
00148       call vlablc (buf(:ilen))
00149       return
00150       end
00151
00152
00153
00154       subroutine juster (iLen,iString,iposflag,ifill,lenchr, ioff)
00155       implicit none
00156       integer iLen,iString(iLen), iposflag,ifill, lenchr, ioff
00157       integer i
00158       character *(255) buf
```

```
00159
00160       lenchr= 0
00161       do 100 i=1, ilen
00162        if ( (i .gt. 1) .or. (istring(i) .ne. ifill) ) then ! Ueberlese Startfillchars
00163         lenchr= lenchr+1
00164         buf(lenchr:lenchr)= char(abs(istring(i))) ! Tek Index -1,-2 -> char(1),char(2)
00165        end if
00166 100   continue
00167       call justerc (buf, iposflag, ioff)
00168       return
00169       end
00170
00171
00172
00173       subroutine eform (fnum,iwidth,idec,ilabel,ifill)
00174       implicit none
00175       integer iwidth,idec, ilabel(iwidth), ifill
00176       real fnum
00177       integer i
00178       character *(255) buf
00179
00180       call eformc (fnum,iwidth,idec, buf)
00181       do 100 i=1,iwidth
00182        ilabel(i)= ichar(buf(i:i))
00183 100   continue
00184       return
00185       end
00186
00187
00188
00189       subroutine fform (fnum,iwidth,idec,ilabel,ifill)
00190       implicit none
00191       integer iwidth,idec, ilabel(255), ifill
00192       real fnum
00193       integer i
00194       character *(255) buf
00195
00196       call fformc (fnum,iwidth,idec, buf)
00197       do 100 i=1,iwidth
00198        ilabel(i)= ichar(buf(i:i))
00199 100   continue
00200       return
00201       end
00202
00203
00204
00205       subroutine fonly (fnum,iwidth,idec,ilabel,ifill)
00206       implicit none
00207       integer iwidth,idec, ilabel(iwidth), ifill
00208       real fnum
00209       integer i
00210       character *(255) buf
00211
00212       call fonlyc (fnum,iwidth,idec, buf)
00213       do 100 i=1,iwidth
00214        ilabel(i)= ichar(buf(i:i))
00215 100   continue
00216       return
00217       end
00218
00219
00220
00221       subroutine iform (fnum,iwidth,ilabel,ifill)
00222       implicit none
00223       integer iwidth,idec, ilabel(iwidth), ifill
00224       real fnum
00225       integer i
00226       character *(255) buf
00227
00228       call iformc (fnum,iwidth,idec, buf)
00229       do 100 i=1,iwidth
00230        ilabel(i)= ichar(buf(i:i))
00231 100   continue
00232       return
00233       end
00234
00235
00236
00237 C
00238 C  Direkte Manipulation des Commonblocks
00239 C
00240
00241       integer function ibasec (iPar)
00242       implicit none
00243       integer ipar
00244
00245       ibasec= -1-ipar
```

```
00246       return
00247       end
00248
00249
00250
00251       integer function ibasex (ipar)
00252       implicit none
00253       integer ipar
00254
00255       ibasex= 1 + 2*ipar
00256       return
00257       end
00258
00259
00260
00261       integer function ibasey (ipar)
00262       implicit none
00263       integer ipar
00264
00265       ibasey= 2 + 2*ipar
00266       return
00267       end
00268
00269
00270
00271       real function comget (ipar)
00272       implicit none
00273       integer ipar
00274       include 'G2dAG2.fd'
00275
00276       integer iarr(1), iarr2(1)
00277       real arr(1), arr2(1)
00278       equivalence(iarr(1),cline), (iarr2(1),cxyneat)
00279       equivalence(arr(1),cline), (arr2(1),cxyneat)
00280
00281       if ((ipar.lt.0) .and. (ipar.ge. -9))then
00282        if ((ipar .eq. -4) .or. (ipar .le. -8)) then
00283         comget= arr(-ipar)
00284        else
00285         comget= real(iarr(-ipar))
00286        end if
00287       else if ((ipar.gt.0) .and. (ipar.le.56)) then
00288        if ((ipar.le.22) .or. ((ipar .ge. 27).and.(ipar.le.52))) then
00289         comget= real(iarr2(ipar))
00290        else
00291         comget= arr2(ipar)
00292        end if
00293       end if
00294       return
00295       end
00296
00297
00298
00299       subroutine comset (iPar,val)
00300       implicit none
00301       integer iPar
00302       real val
00303       include 'G2dAG2.fd'
00304
00305       integer iarr(1), iarr2(1)
00306       real arr(1), arr2(1)
00307       equivalence(iarr(1),cline), (iarr2(1),cxyneat)
00308       equivalence(arr(1),cline), (arr2(1),cxyneat)
00309
00310       if ((ipar.lt.0) .and. (ipar.ge. -9))then
00311        if ((ipar.eq.-4) .or. (ipar .le. -8)) then
00312         arr(-ipar)= val
00313        else
00314         iarr(-ipar)= int(val)
00315        end if
00316       else if ((ipar.gt.0) .and. (ipar.le.56)) then
00317        if ((ipar.le.22) .or. ((ipar .ge. 27).and.(ipar.le.52))) then
00318         iarr2(ipar)= int(val)
00319        else
00320         arr2(ipar)= val
00321        end if
00322       end if
00323       return
00324       end
00325
00326
00327
00328       subroutine comdmp
00329       implicit none
00330       integer i
00331       character *80 buf
00332       include 'G2dAG2.fd'
```

```
00333
00334       call erase
00335       call home
00336
00337       write (unit= buf,fmt=600, err=200) (cxyneat(i),i=1,2), cline
00338 600   format (1x,' 0:  cxneat(1)=',i14,', (2)=',i14,',   cline=',i14)
00339       call toutstc (buf)
00340       call newlin
00341       write (unit= buf,fmt=601, err=200) (cxyzero(i),i=1,2), csymbl
00342 601   format (1x,' 1: cxyzero(1)=',i14,', (2)=',i14,',  csymbl=',i14)
00343       call toutstc (buf)
00344       call newlin
00345       write (unit= buf,fmt=602, err=200) (cxyloc(i),i=1,2), csteps
00346 602   format (1x,' 2:  cxyloc(1)=',i14,', (2)=',i14,',  csteps=',i14)
00347       call toutstc (buf)
00348       call newlin
00349       write (unit= buf,fmt=603, err=200) (cxylab(i),i=1,2), cinfin
00350 603   format (1x,' 3:  cxylab(1)=',i14,', (2)=',i14,',  cinfin=',e14.7)
00351       call toutstc (buf)
00352       call newlin
00353       write (unit= buf,fmt=604, err=200) (cxyden(i),i=1,2), cnpts
00354 604   format (1x,' 4:  cxyden(1)=',i14,', (2)=',i14,',   cnpts=',i14)
00355       call toutstc (buf)
00356       call newlin
00357       write (unit= buf,fmt=605, err=200) (cxytics(i),i=1,2), cstepl
00358 605   format (1x,' 5: cxytics(1)=',i14,', (2)=',i14,',  cstepl=',i14)
00359       call toutstc (buf)
00360       call newlin
00361       write (unit= buf,fmt=606, err=200) (cxylen(i),i=1,2), cnumbr
00362 606   format (1x,' 6:  cxylen(1)=',i14,', (2)=',i14,',  cnumbr=',i14)
00363       call toutstc (buf)
00364       call newlin
00365       write (unit= buf,fmt=607, err=200) (cxyfrm(i),i=1,2), csizes
00366 607   format (1x,' 7:  cxyfrm(1)=',i14,', (2)=',i14,',  csizes=',e14.7)
00367       call toutstc (buf)
00368       call newlin
00369       write (unit= buf,fmt=608, err=200) (cxymtcs(i),i=1,2), csizel
00370 608   format (1x,' 8: cxymtcs(1)=',i14,', (2)=',i14,',  csizel=',e14.7)
00371       call toutstc (buf)
00372       call newlin
00373       write (unit= buf,fmt=609, err=200) (cxymfrm(i),i=1,2)
00374 609   format (1x,' 9: cxymfrm(1)=',i14,', (2)=',i14)
00375       call toutstc (buf)
00376       call newlin
00377       write (unit= buf,fmt=610, err=200) (cxydec(i),i=1,2)
00378 610   format (1x,'10:  cxydec(1)=',i14,', (2)=',i14)
00379       call toutstc (buf)
00380       call newlin
00381       write (unit= buf,fmt=611, err=200) (cxydmin(i),i=1,2)
00382 611   format (1x,'11: cxydmin(1)=',e14.7,', (2)=',e14.7)
00383       call toutstc (buf)
00384       call newlin
00385       write (unit= buf,fmt=612, err=200) (cxydmax(i),i=1,2)
00386 612   format (1x,'12: cxydmax(1)=',e14.7,', (2)=',e14.7)
00387       call toutstc (buf)
00388       call newlin
00389       write (unit= buf,fmt=613, err=200) (cxysmin(i),i=1,2)
00390 613   format (1x,'13: cxysmin(1)=',i14,', (2)=',i14)
00391       call toutstc (buf)
00392       call newlin
00393       write (unit= buf,fmt=614, err=200) (cxysmax(i),i=1,2)
00394 614   format (1x,'14: cxysmax(1)=',i14,', (2)=',i14)
00395       call toutstc (buf)
00396       call newlin
00397       write (unit= buf,fmt=615, err=200) (cxytype(i),i=1,2)
00398 615   format (1x,'15: cxytype(1)=',i14,', (2)=',i14)
00399       call toutstc (buf)
00400       call newlin
00401       write (unit= buf,fmt=616, err=200) (cxylsig(i),i=1,2)
00402 616   format (1x,'16: cxylsig(1)=',i14,', (2)=',i14)
00403       call toutstc (buf)
00404       call newlin
00405       write (unit= buf,fmt=617, err=200) (cxywdth(i),i=1,2)
00406 617   format (1x,'17: cxywdth(1)=',i14,', (2)=',i14)
00407       call toutstc (buf)
00408       call newlin
00409       write (unit= buf,fmt=618, err=200) (cxyepon(i),i=1,2)
00410 618   format (1x,'18: cxyepon(1)=',i14,', (2)=',i14)
00411       call toutstc (buf)
00412       call newlin
00413       write (unit= buf,fmt=619, err=200) (cxystep(i),i=1,2)
00414 619   format (1x,'19: cxystep(1)=',i14,', (2)=',i14)
00415       call toutstc (buf)
00416       call newlin
00417       write (unit= buf,fmt=620, err=200) (cxystag(i),i=1,2)
00418 620   format (1x,'20: cxystag(1)=',i14,', (2)=',i14)
00419       call toutstc (buf)
```

```
00420       call newlin
00421       write (unit= buf,fmt=621, err=200) (cxyetyp(i),i=1,2)
00422 621   format (1x,'21: cxyetyp(1)=',i14,', (2)=',i14)
00423       call toutstc (buf)
00424       call newlin
00425       write (unit= buf,fmt=622, err=200) (cxybeg(i),i=1,2)
00426 622   format (1x,'22:  cxybeg(1)=',i14,', (2)=',i14)
00427       call toutstc (buf)
00428       call newlin
00429       write (unit= buf,fmt=623, err=200) (cxyend(i),i=1,2)
00430 623   format (1x,'23:  cxyend(1)=',i14,', (2)=',i14)
00431       call toutstc (buf)
00432       call newlin
00433       write (unit= buf,fmt=624, err=200) (cxymbeg(i),i=1,2)
00434 624   format (1x,'24: cxymbeg(1)=',i14,', (2)=',i14)
00435       call toutstc (buf)
00436       call newlin
00437       write (unit= buf,fmt=625, err=200) (cxymend(i),i=1,2)
00438 625   format (1x,'25: cxymend(1)=',i14,', (2)=',i14)
00439       call toutstc (buf)
00440       call newlin
00441       write (unit= buf,fmt=626, err=200) (cxyamin(i),i=1,2)
00442 626   format (1x,'26: cxyamin(1)=',e14.7,', (2)=',e14.7)
00443       call toutstc (buf)
00444       call newlin
00445       write (unit= buf,fmt=627, err=200) (cxyamax(i),i=1,2)
00446 627   format (1x,'27: cxyamax(1)=',e14.7,', (2)=',e14.7)
00447       call toutstc (buf)
00448
00449       call graphicerror (11,char(0))
00450       call erase
00451
00452 200   continue
00453       return
00454       end
```

3.5 AG2uline.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine uline (x, y, i)

3.5.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2uline.for.

3.5.2 Function/Subroutine Documentation

3.5.2.1 uline()

```
subroutine uline (
              x,
              y,
              i )
```

Definition at line 10 of file AG2uline.for.

3.6 AG2uline.for

```
00001 C> \file    AG2uline.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009
00010       subroutine uline (x,y,i)
00011       return
00012       end
00013
```

3.7 AG2umnmx.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine umnmx (array, amin, amax)

3.7.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2umnmx.for.

3.7.2 Function/Subroutine Documentation

3.7.2.1 umnmx()

```
subroutine umnmx (
               array,
               amin,
               amax )
```

Definition at line 9 of file AG2umnmx.for.

3.8 AG2umnmx.for

```
00001 C> \file    AG2umnmx.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009       subroutine umnmx (array,amin,amax)
00010       return
00011       end
00012
```

3.9 AG2upoint.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- real function upoint (arr, ii, oldone)

3.9.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2upoint.for.

3.9.2 Function/Subroutine Documentation

3.9.2.1 upoint()

```
real function upoint (
               arr,
               ii,
               oldone )
```

Definition at line 9 of file AG2upoint.for.

3.10 AG2upoint.for

```
00001 C> \file    AG2upoint.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009       real function upoint (arr,ii,oldone)
00010       upoint=0.
00011       return
00012       end
```

3.11 AG2users.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine users (x, y, i)

3.11.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2users.for.

3.11.2 Function/Subroutine Documentation

3.11.2.1 users()

```
subroutine users (
              x,
              y,
              i )
```

Definition at line 9 of file AG2users.for.

3.12 AG2users.for

```
00001 C> \file    AG2users.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009       subroutine users (x,y,i)
00010       return
00011       end
```

3.13 AG2useset.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine useset (fnum, iwidth, nbase, labeli)

3.13.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2useset.for.

3.13.2 Function/Subroutine Documentation

3.13.2.1 useset()

```
subroutine useset (
             real fnum,
             integer iwidth,
             integer nbase,
             integer, dimension(1) labeli )
```

Definition at line 9 of file AG2useset.for.

3.14 AG2useset.for

```
00001 C> \file    AG2useset.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009       subroutine useset (fnum,iwidth,nbase,labeli)
00010       implicit none
00011       real fnum
00012       integer iwidth, nbase
00013       integer labeli(1)
00014       integer i
00015
00016       do 100 i=1, iwidth
00017        labeli(i)= 32 ! Blank
00018 100   continue
00019       return
00020       end
00021
```

3.15 AG2usesetC.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine usesetc (fnum, iwidth, nbase, labstr)

3.15.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2usesetC.for.

3.15.2 Function/Subroutine Documentation

3.15.2.1 usesetc()

```
subroutine usesetc (
             real fnum,
             integer iwidth,
             integer nbase,
             character *(*) labstr )
```

Definition at line 9 of file AG2usesetC.for.

3.16 AG2usesetC.for

```
00001 C> \file    AG2usesetC.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009       subroutine usesetc (fnum,iwidth, nbase, labstr)
00010       implicit none
00011       real fnum
00012       integer iwidth, nbase
00013       character *(*) labstr
00014       integer labeli(20)
00015       integer i, i1, iw, ISTRINGLEN
00016
00017       iw= min(20, iwidth, istringlen(labstr))
00018       call useset (fnum,iw,nbase,labeli)
00019
00020       i1= 0
00021       do 100 i=1,iw
00022        i1= i1+1
00023        labstr(i1:i1)= char(labeli(i))
00024 100   continue
00025       if (i1 .lt. iw) labstr(i1+1:i1+1)= char(0)
00026       return
00027       end
00028
```

3.17 AG2UsrSoftek.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine softek (isym)

3.17.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2UsrSoftek.for.

3.17.2 Function/Subroutine Documentation

3.17.2.1 softek()

```
subroutine softek (
              isym )
```

Definition at line 9 of file AG2UsrSoftek.for.

3.18 AG2UsrSoftek.for

```
00001 C> \file    AG2UsrSoftek.for
00002 C> \brief   Graph2D: Dummy User Routine
00003 C
00004 C  Tektronix Advanced Graphics 2 - Version 2.0
00005 C
00006 C     User Subroutinen
00007 C
00008
00009       subroutine softek (isym)
00010       return
00011       end
```

3.19 Fgraph.fd File Reference

DOS Port: Declarations OW graph.lib.

3.19.1 Detailed Description

DOS Port: Declarations OW graph.lib.

Functions and constants of the Watcom DOS Graphic Library. Substitution for the INCLUDE-file of the Microsoft Fortran Compiler, derived from the Watcom Headerfile graph.fi.

Author

Dr.-Ing. Klaus Friedewald

Note

Watcom-FTN77 variable names are allowed to be 32 characters long and may contain $ and _. That for $notruncate und $notstrict are superfluous.

Hexadecimal numbers are represented by 'ff'x instead of #ff.

The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to be linked to the main programs: -libr graph.

Definition in file Fgraph.fd.

3.20 Fgraph.fd

```
00001 C> \file    Fgraph.fd
00002 C> \brief   DOS Port: Declarations OW graph.lib
00003 C>
00004 C> \~german
00005 C> Konstanten und Funktionen der Watcom DOS Graphik-Library. Ersatz für das zum
00006 C> Microsoft Fortan-Compiler gehörende INCLUDE-File, abgeleitet aus dem
00007 C> Watcom-Headerfile graph.fi.
00008 C>
00009 C> \~english
00010 C> Functions and constants of the Watcom DOS Graphic Library. Substitution for
00011 C> the INCLUDE-file of the Microsoft Fortran Compiler, derived from the
00012 C> Watcom Headerfile graph.fi.
00013 C>
00014 C> \~
00015 C> \author  Dr.-Ing. Klaus Friedewald
00016 C>
00017 C> \~german
00018 C> \note
00019 C> Der Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
00020 C> von $ und _. Deswegen sind $notruncate und $notstrict überflüssig.
00021 C>
00022 C> \note
00023 C> Hex-Zahlen werden nicht durch \#ff sondern durch 'ff'x dargestellt.
00024 C>
00025 C> \note
00026 C> Die OpenWatcom Library graph.lib ist nicht Bestandteil von Graph2Ddos.lib
00027 C> und muss bei den Linkoptionen der Hauptprogramme aufgeführt werden:
00028 C> -libr graph.
00029 C> \~english
00030 C> \note
00031 C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
00032 C> contain $ and _. That for $notruncate und $notstrict are superfluous.
00033 C>
00034 C> \note
00035 C> Hexadecimal numbers are represented by 'ff'x instead of \#ff.
00036 C>
00037 C> \note
00038 C> The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to
00039 C> be linked to the main programs:
00040 C> -libr graph.
00041 C> \~
00042 C>
00043 C> \cond
00044
00045       structure/videoconfig/      ! structure for getvideoconfig
00046         integer*2 numxpixels
00047         integer*2 numypixels
00048         integer*2 numtextcols
00049         integer*2 numtextrows
00050         integer*2 numcolors
00051         integer*2 bitsperpixel
00052         integer*2 numvideopages
00053         integer*2 mode
00054         integer*2 adapter
00055         integer*2 monitor
00056         integer*2 memory
00057       end structure
00058
00059       structure/xycoord/          ! structure for pixel position
00060         integer*2 xcoord
00061         integer*2 ycoord
00062       end structure
00063
00064       structure/rccoord/          ! structure for text position
00065         integer*2 row
00066         integer*2 col
00067       end structure
00068
00069 C Videomodes
00070
00071       integer*2, $MAXRESMODE, $MAXCOLORMODE, $DEFAULTMODE,$TEXTBW40,
00072      1           $TEXTC40,$TEXTBW80,$TEXTC80, $MRES4COLOR,$MRESNOCOLOR,
00073      2           $HRESBW,$TEXTMONO,$HERCMONO, $MRES16COLOR,$HRES16COLOR,
00074      3           $ERESNOCOLOR,$ERESCOLOR, $VRES2COLOR,$VRES16COLOR,
00075      4           $MRES256COLOR,$ORESCOLOR
00076       parameter($maxresmode   =-3)    ! graphics mode with highest resolution
00077       parameter($maxcolormode =-2)    ! graphics mode with most colors
00078       parameter($defaultmode  =-1)    ! restore screen to original mode
00079       parameter($textbw40      =0)    ! 40 x 25 text, 16 grey
00080       parameter($textc40       =1)    ! 40 x 25 text, 16/8 color
00081       parameter($textbw80      =2)    ! 80 x 25 text, 16 grey
00082       parameter($textc80       =3)    ! 80 x 25 text, 16/8 color
00083       parameter($mres4color    =4)    ! 320 x 200, 4 color
00084       parameter($mresnocolor   =5)    ! 320 x 200, 4 grey
00085       parameter($hresbw        =6)    ! 640 x 200, BW
```

```
00086       parameter($textmono      =7)    ! 80 x 25 text, BW
00087       parameter($hercmono      =8)    ! 720 x 348, BW for HGC
00088       parameter($mres16color   =13)   ! 320 x 200, 16 color
00089       parameter($hres16color   =14)   ! 640 x 200, 16 color
00090       parameter($eresnocolor   =15)   ! 640 x 350, BW
00091       parameter($erescolor     =16)   ! 640 x 350, 4 or 16 color
00092       parameter($vres2color    =17)   ! 640 x 480, BW
00093       parameter($vres16color   =18)   ! 640 x 480, 16 color
00094       parameter($mres256color  =19)   ! 320 x 200, 256 color
00095       parameter($orescolor     =64)   ! 640 x 400, 1 of 16 colors (Olivetti)
00096
00097       integer*4 $MDPA,$CGA,$EGA,$MCGA,$VGA,$HGC,$OCGA,$OEGA,$OVGA
00098       parameter($mdpa     ='0001'x)    ! Monochrome Display Adapter (MDPA)
00099       parameter($cga      ='0002'x)    ! Color Graphics Adapter     (CGA)
00100       parameter($ega      ='0004'x)    ! Enhanced Graphics Adapter  (EGA)
00101       parameter($vga      ='0008'x)    ! Video Graphics Array       (VGA)
00102       parameter($mcga     ='0010'x)    ! MultiColor Graphics Array  (MCGA)
00103       parameter($hgc      ='0020'x)    ! Hercules Graphics Card     (HGC)
00104       parameter($ocga     ='0042'x)    ! Olivetti Color Graphics Adapter (OCGA)
00105       parameter($oega     ='0044'x)    ! Olivetti Enhanced Graphics Adapter (OEGA)
00106       parameter($ovga     ='0048'x)    ! Olivetti Video Graphics Array (OVGA)
00107
00108       integer*4 $MONO,$COLOR,$ENHCOLOR,$ANALOGMONO,$ANALOGCOLOR,$ANALOG
00109       parameter($mono       ='0001'x)     ! Monochrome
00110       parameter($color      ='0002'x)     ! Color (or Enhanced emulating color)
00111       parameter($enhcolor   ='0004'x)     ! Enhanced Color
00112       parameter($analogmono ='0008'x)     ! Analog Monochrome only
00113       parameter($analogcolor='0010'x)     ! Analog Color only
00114       parameter($analog     ='0018'x)     ! Analog
00115
00116 C Plotting Action
00117
00118       integer*2 $GBORDER,$GFILLINTERIOR,
00119      1          $GCLEARSCREEN, $GVIEWPORT,$GWINDOW
00120
00121       parameter($gborder       =2)      ! draw outline only
00122       parameter($gfillinterior =3)      ! fill using current fill mask
00123
00124       parameter($gclearscreen=0)
00125       parameter($gviewport   =1)
00126       parameter($gwindow     =2)
00127
00128       integer*4 $GCURSOROFF,$GCURSORON,$GWRAPOFF,$GWRAPON
00129       parameter($gcursoroff=0)
00130       parameter($gcursoron =1)
00131
00132       parameter($gwrapoff  =0)
00133       parameter($gwrapon   =1)
00134
00135       integer*4 $GSCROLLUP, $GSCROLLDOWN
00136       parameter($gscrollup   =1)
00137       parameter($gscrolldown =-1)
00138
00139       integer*4 $MAXTEXTROWS
00140       parameter($maxtextrows =-1)
00141
00142       integer*4 $GPSET,$GPRESET,$GAND,$GOR,$GXOR
00143       parameter($gpset         =3)
00144       parameter($gpreset       =2)
00145       parameter($gand          =1)
00146       parameter($gor           =0)
00147       parameter($gxor          =4)
00148
00149       integer*4 $BLACK,$BLUE,$GREEN,$CYAN,$RED,$MAGENTA,$BROWN,
00150      1          $WHITE,$GRAY, $LIGHTBLUE,$LIGHTGREEN,$LIGHTCYAN,
00151      2          $LIGHTRED,$LIGHTMAGENTA, $LIGHTYELLOW,$BRIGHTWHITE
00152       parameter($black         ='000000'x)
00153       parameter($blue          ='2a0000'x)
00154       parameter($green         ='002a00'x)
00155       parameter($cyan          ='2a2a00'x)
00156       parameter($red           ='00002a'x)
00157       parameter($magenta       ='2a002a'x)
00158       parameter($brown         ='00152a'x)
00159       parameter($white         ='2a2a2a'x)
00160       parameter($gray          ='151515'x)
00161       parameter($lightblue     ='3F1515'x)
00162       parameter($lightgreen    ='153f15'x)
00163       parameter($lightcyan     ='3f3f15'x)
00164       parameter($lightred      ='15153f'x)
00165       parameter($lightmagenta  ='3f153f'x)
00166       parameter($lightyellow   ='153f3f'x)
00167       parameter($brightwhite   ='3f3f3f'x)
00168
00169       integer*4 $MODEFOFF,$MODEFOFFTOON,$MODEFOFFTOHI,$MODEFONTOOFF,
00170      1          $MODEFON,$MODEFONTOHI,$MODEFHITOOFF,$MODEFHITOON,
00171      2          $MODEFHI
00172       parameter($modefoff      =0)
```

```
00173       parameter($modefofftoon  =1)
00174       parameter($modefofftohi  =2)
00175       parameter($modefontooff  =3)
00176       parameter($modefon       =4)
00177       parameter($modefontohi   =5)
00178       parameter($modefhitooff  =6)
00179       parameter($modefhitoon   =7)
00180       parameter($modefhi       =8)
00181
00182       integer*4 $MODE7OFF,$MODE7ON,$MODE7HI
00183       parameter($mode7off      =0)
00184       parameter($mode7on       =1)
00185       parameter($mode7hi       =2)
00186
00187 C external functions
00188
00189       external setvideomode
00190       integer*2 setvideomode
00191
00192       external setvideomoderows
00193       integer*2 setvideomoderows
00194
00195       external setactivepage
00196       integer*2 setactivepage
00197
00198       external setvisualpage
00199       integer*2 setvisualpage
00200
00201       external getactivepage
00202       integer*2 getactivepage
00203
00204       external getvisualpage
00205       integer*2 getvisualpage
00206
00207       external getvideoconfig
00208       external setvieworg
00209       external getviewcoord
00210       external getphyscoord
00211       external setcliprgn
00212       external setviewport
00213       external clearscreen
00214       external moveto
00215       external getcurrentposition
00216
00217       external lineto
00218       integer*2 lineto
00219
00220       external rectangle
00221       integer*2 rectangle
00222
00223       external ellipse
00224       integer*2 ellipse
00225
00226       external arc
00227       integer*2 arc
00228
00229       external pie
00230       integer*2 pie
00231
00232       external setpixel
00233       integer*2 setpixel
00234
00235       external getpixel
00236       integer*2 getpixel
00237
00238       external floodfill
00239       integer*2 floodfill
00240
00241       external setcolor
00242       integer*2 setcolor
00243
00244       external getcolor
00245       integer*2 getcolor
00246
00247       external setlinestyle
00248
00249       external getlinestyle
00250       integer*2 getlinestyle
00251
00252       external setfillmask
00253       external getfillmask
00254
00255       external setbkcolor
00256       integer*4 setbkcolor
00257
00258       external getbkcolor
00259       integer*4 getbkcolor
```

```
00260
00261       external remappalette
00262       integer*4 remappalette
00263
00264       external remapallpalette
00265       integer*2 remapallpalette
00266
00267       external selectpalette
00268       integer*2 selectpalette
00269
00270       external settextrows
00271       integer*2 settextrows
00272
00273       external settextwindow
00274       external scrolltextwindow
00275       external outtext
00276
00277       external wrapon
00278       integer*2 wrapon
00279
00280       external displaycursor
00281       integer*2 displaycursor
00282
00283       external settextcursor
00284       integer*2 settextcursor
00285
00286       external gettextcursor
00287       integer*2 gettextcursor
00288
00289       external settextposition
00290       external gettextposition
00291
00292       external settextcolor
00293       integer*2 settextcolor
00294
00295       external gettextcolor
00296       integer*2 gettextcolor
00297
00298       external getimage
00299       external putimage
00300
00301       external imagesize
00302       integer*4 imagesize
00303
00304
00305
00306       structure/wxycoord/       ! window coordinates
00307         double precision wx
00308         double precision wy
00309       end structure
00310
00311       external setwindow
00312       integer*2 setwindow
00313
00314       external getwindowcoord
00315       external getviewcoord_w
00316       external getcurrentposition_w
00317
00318
00319       external arc_w
00320       integer*2 arc_w
00321
00322       external ellipse_w
00323       integer*2 ellipse_w
00324
00325       external floodfill_w
00326       integer*2 floodfill_w
00327
00328       external getpixel_w
00329       integer*2 getpixel_w
00330
00331       external lineto_w
00332       integer*2 lineto_w
00333
00334       external moveto_w
00335
00336       external pie_w
00337       integer*2 pie_w
00338
00339       external rectangle_w
00340       integer*2 rectangle_w
00341
00342       external setpixel_w
00343       integer*2 setpixel_w
00344
00345       external getimage_w
00346
```

```
00347       external imagesize_w
00348       integer*2 imagesize_w
00349
00350       external putimage_w
00351
00352       structure/fontinfo/
00353         integer*2 type          ! b0 set = vector,clear = bit map
00354         integer*2 ascent        ! pix dist from top to baseline
00355         integer*2 pixwidth      ! character width in pixels, 0=prop
00356         integer*2 pixheight     ! character height in pixels
00357         integer*2 avgwidth      ! average character width in pixels
00358         character*81 filename   ! file name including path
00359         character*32 facename   ! font name
00360       end structure
00361
00362
00363       integer*2 $NO_SPACE, $FIXED_SPACE, $PROP_SPACE
00364       parameter($no_space    = 0)
00365       parameter($fixed_space = 1)
00366       parameter($prop_space  = 2)
00367
00368       integer*2 $NO_FONT_MAP, $VECTOR_MAP, $BIT_MAP
00369       parameter($no_font_map = 0)
00370       parameter($vector_map  = 1)
00371       parameter($bit_map     = 2)
00372
00373       external registerfonts
00374       integer*2 registerfonts
00375
00376       external unregisterfonts
00377
00378       external setfont
00379       integer*2 setfont
00380
00381       external getfontinfo
00382       integer*2 getfontinfo
00383
00384       external outgtext
00385
00386       external getgtextextent
00387       integer*2 getgtextextent
00388 C
00389 C> \endcond
```

3.21 Fgraph.fi File Reference

DOS Port: Interface OW graph.lib.

3.21.1 Detailed Description

DOS Port: Interface OW graph.lib.

Interface definition for the Watcom DOS Graphic Library. Substitutes the INCLUDE-file of the Microsoft Fortran Compiler, derived from the Watcom headerfile graphapi.fi.

Author

Dr.-Ing. Klaus Friedewald

Note

Watcom-FTN77 variable names are allowed to be 32 characters long and may contain $ and _. That for $notruncate und $notstrict are superfluous.

The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to be linked to the main programs: -libr graph.

Definition in file Fgraph.fi.

3.22 Fgraph.fi

```
00001 C> \file    Fgraph.fi
00002 C> \brief   DOS Port: Interface OW graph.lib
00003 C>
00004 C> \~german
00005 C> Interfacedeklaration der Watcom DOS Graphik-Library. Ersatz für das zum
00006 C> Microsoft Fortran-Compiler gehörende INCLUDE-File, abgeleitet aus dem
00007 C> Watcom-Headerfile graphapi.fi.
00008 C>
00009 C> \~english
00010 C> Interface definition for the Watcom DOS Graphic Library. Substitutes
00011 C> the INCLUDE-file of the Microsoft Fortran Compiler, derived from the
00012 C> Watcom headerfile graphapi.fi.
00013 C>
00014 C> \~
00015 C> \author  Dr.-Ing. Klaus Friedewald
00016 C>
00017 C> \~german
00018 C> \note
00019 C> Der Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
00020 C> von $ und _. Deswegen sind $notruncate und $notstrict überflüssig.
00021 C>
00022 C> \note
00023 C> Die OpenWatcom Library graph.lib ist nicht Bestandteil von Graph2Ddos.lib
00024 C> und muss bei den Linkoptionen der Hauptprogramme aufgeführt werden:
00025 C> -libr graph.
00026 C> \~english
00027 C> \note
00028 C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
00029 C> contain $ and _. That for $notruncate und $notstrict are superfluous.
00030 C>
00031 C> \note
00032 C> The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to
00033 C> be linked to the main programs:
00034 C> -libr graph.
00035 C> \~
00036 C>
00037
00038
00039 c$pragma aux arc "_arc_" parm (VALUE*2)
00040
00041 c$pragma aux arc_w "_arc_w_" parm (VALUE*8)
00042
00043 c$pragma aux clearscreen "_clearscreen_" parm (VALUE*2)
00044
00045 c$pragma aux displaycursor "_displaycursor_" parm (VALUE*2)
00046
00047 c$pragma aux ellipse  "_ellipse_" parm (VALUE*2)
00048
00049 c$pragma aux ellipse_w "_ellipse_w_" parm (VALUE*2, VALUE*8)
00050
00051 c$pragma aux floodfill  "_floodfill_" parm (VALUE*2)
00052
00053 c$pragma aux floodfill_w "_floodfill_w_" parm (VALUE*8, VALUE*8, VALUE*2)
00054
00055 c$pragma aux getactivepage  "_getactivepage_"
00056
00057 c$pragma aux getbkcolor  "_getbkcolor_"
00058
00059 c$pragma aux getcolor "_getcolor_"
00060
00061 c$pragma aux getcurrentposition "_getcurrentposition_" parm (REFERENCE FAR)
00062
00063 c$pragma aux getcurrentposition_w "_getcurrentposition_w_" parm (REFERENCE FAR)
00064
00065 c$pragma aux getfillmask "_getfillmask_" parm (REFERENCE FAR)
00066
00067 c$pragma aux getimage "_getimage_" parm (VALUE*2,VALUE*2,VALUE*2,VALUE*2, \
00068 c REFERENCE FAR)
00069
00070 c$pragma aux getimage_w "_getimage_w_" parm (VALUE*8,VALUE*8,VALUE*8, \
00071 c  VALUE*8,REFERENCE FAR)
00072
00073 c$pragma aux  getlinestyle "_getlinestyle_"
00074
00075 c$pragma aux getphyscoord "_getphyscoord_" parm (VALUE*2,VALUE*2, \
00076 c  REFERENCE FAR)
00077
00078 c$pragma aux getpixel "_getpixel_" parm (VALUE*2)
00079
00080 c$pragma aux getpixel_w "_getpixel_w_" parm (VALUE*8)
00081
00082 c$pragma aux gettextcolor "_gettextcolor_"
00083
00084 c$pragma aux gettextcursor "_gettextcursor_"
00085
```

```
00086 c$pragma aux gettextposition "_gettextposition_" parm (REFERENCE FAR)
00087
00088 c$pragma aux getvideoconfig "_getvideoconfig_" parm (REFERENCE FAR)
00089
00090 c$pragma aux getviewcoord "_getviewcoord_" parm (VALUE*2,VALUE*2, \
00091 c  REFERENCE FAR)
00092
00093 c$pragma aux getviewcoord_w "_getviewcoord_w_" parm (VALUE*8,VALUE*8, \
00094 c  REFERENCE FAR)
00095
00096 c$pragma aux getvisualpage "_getvisualpage_"
00097
00098 c$pragma aux getwindowcoord "_getwindowcoord_" parm (VALUE*2,VALUE*2, \
00099 c  REFERENCE FAR)
00100
00101 c$pragma aux imagesize "_imagesize_" parm (VALUE*2)
00102
00103 c$pragma aux imagesize_w "_imagesize_w_" parm (VALUE*8)
00104
00105 c$pragma aux lineto "_lineto_" parm (VALUE*2)
00106
00107 c$pragma aux lineto_w "_lineto_w_" parm (VALUE*8)
00108
00109 c$pragma aux moveto "_moveto_" parm (VALUE*2,VALUE*2,REFERENCE FAR)
00110
00111 c$pragma aux moveto_w "_moveto_w_" parm (VALUE*8,VALUE*8,REFERENCE FAR)
00112
00113 c$pragma aux _outtext "_outtext_" parm (DATA_REFERENCE FAR)
00114
00115 c$pragma aux pie "_pie_" parm (VALUE*2)
00116
00117 c$pragma aux pie_w "_pie_w_" parm (VALUE*2,VALUE*8)
00118
00119 c$pragma aux putimage "_putimage_" parm (VALUE*2,VALUE*2,REFERENCE FAR,VALUE*2)
00120
00121 c$pragma aux putimage_w "_putimage_w_" parm (VALUE*8,VALUE*8, \
00122 c  REFERENCE FAR,VALUE*2)
00123
00124 c$pragma aux rectangle "_rectangle_" parm (VALUE*2)
00125
00126 c$pragma aux rectangle_w "_rectangle_w_" parm (VALUE*2,VALUE*8)
00127
00128 c$pragma aux remappalette "_remappalette_" parm (VALUE*2,VALUE*4)
00129
00130 c$pragma aux remapallpalette "_remapallpalette_" parm (VALUE*4)
00131
00132 c$pragma aux scrolltextwindow "_scrolltextwindow_" parm (VALUE*2)
00133
00134 c$pragma aux selectpalette "_selectpalette_" parm (VALUE*2)
00135
00136 c$pragma aux setactivepage "_setactivepage_" parm (VALUE*2)
00137
00138 c$pragma aux setbkcolor "_setbkcolor_" parm (VALUE*4)
00139
00140 c$pragma aux setcliprgn "_setcliprgn_" parm (VALUE*2)
00141
00142 c$pragma aux setcolor "_setcolor_" parm (VALUE*2)
00143
00144 c$pragma aux setfillmask "_setfillmask_" parm (REFERENCE FAR)
00145
00146 c$pragma aux setlinestyle "_setlinestyle_" parm (VALUE*2)
00147
00148 c$pragma aux setpixel "_setpixel_" parm (VALUE*2)
00149
00150 c$pragma aux setpixel_w"_setpixel_w_" parm (VALUE*8)
00151
00152 c$pragma aux settextcolor "_settextcolor_" parm (VALUE*2)
00153
00154 c$pragma aux settextcursor "_settextcursor_" parm (VALUE*2)
00155
00156 c$pragma aux settextposition "_settextposition_" parm (VALUE*2,VALUE*2, \
00157 c  REFERENCE FAR)
00158
00159 c$pragma aux settextrows "_settextrows_" parm (VALUE*2)
00160
00161 c$pragma aux settextwindow "_settextwindow_" parm (VALUE*2)
00162
00163 c$pragma aux setvideomode "_setvideomode_" parm (VALUE*2)
00164
00165 c$pragma aux setvideomoderows "_setvideomoderows_" parm (VALUE*2)
00166
00167 c$pragma aux setvieworg "_setvieworg_" parm (VALUE*2, VALUE*2,REFERENCE FAR)
00168
00169 c$pragma aux setviewport "_setviewport_" parm (VALUE*2)
00170
00171 c$pragma aux setvisualpage "_setvisualpage_" parm (VALUE*2)
00172
```

```
00173 c$pragma aux setwindow "_setwindow_" parm (VALUE*2,VALUE*8)
00174
00175 c$pragma aux wrapon "_wrapon_" parm (VALUE*2)
00176
00177
00178 c$pragma aux getfontinfo "_getfontinfo_" parm (REFERENCE FAR)
00179
00180 c$pragma aux getgtextextent "_getgtextextent_" parm (DATA_REFERENCE FAR)
00181
00182 c$pragma aux outgtext "_outgtext_" parm (DATA_REFERENCE FAR)
00183
00184 c$pragma aux registerfonts "_registerfonts_" parm (DATA_REFERENCE FAR)
00185
00186 c$pragma aux setfont "_setfont_" parm (DATA_REFERENCE FAR)
00187
00188 c$pragma aux unregisterfonts "_unregisterfonts_"
```

3.23 G2dAG2.fd File Reference

Graph2D: AG2 Common Block G2dAG2.

3.23.1 Detailed Description

Graph2D: AG2 Common Block G2dAG2.

Version

2.0

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Definition in file G2dAG2.fd.

3.24 G2dAG2.fd

```
00001 C> \file       G2dAG2.fd
00002 C> \brief      Graph2D: AG2 Common Block G2dAG2
00003 C> \version    2.0
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C
00007 C  Da die folgende Definition kein Bestandteil eines Moduls
00008 C  ist versagt der DOXYGEN-Parser bei der Kombination von
00009 C  COMMON und integer. Workaraound: \\cond ... \\endcond
00010 C> \cond
00011
00012 C Common Block G2dAG2, Version 2.0 für AG2
00013 C     Die Funktion der Variablen entspricht dem Tektronix AG2 User-Manual,
00014 C     jedoch sind die achsenbezogenen Variablen in einem Feld zusammenge-
00015 C     fasst. Die x-Achse wird durch Index=1, y durch Index=2 beschrieben.
00016 C
00017       integer     cline,csymbl,csteps ! ibase+ 0..2
00018       real        cinfin ! 3
00019       integer     cnpts,cstepl,cnumbr ! 4..6
00020       real        csizes,csizel ! 7,8
00021
```

```
00022       logical     cxyneat(2),cxyzero(2) ! nbase+ 0, 1
00023       integer     cxyloc(2),cxylab(2),cxyden(2),cxytics(2) ! nbase+ 2..5
00024       integer     cxylen(2),cxyfrm(2),cxymtcs(2),cxymfrm(2),cxydec(2) ! 6..10
00025       real        cxydmin(2),cxydmax(2) ! 11,12
00026       integer     cxysmin(2),cxysmax(2),cxytype(2) ! 13..15
00027       integer     cxylsig(2),cxywdth(2),cxyepon(2) ! 16..18
00028       integer     cxystep(2),cxystag(2),cxyetyp(2) ! 19..21
00029       integer     cxybeg(2),cxyend(2),cxymbeg(2),cxymend(2) ! 22..25
00030       real        cxyamin(2),cxyamax(2) ! 26,27
00031
00032       common /g2dag2/
00033 C     & extent,cvectr,xvectr,yvectr,
00034 C     & xtentc,xtentx,xtenty,
00035 C
00036      & cline,csymbl,csteps,
00037      & cinfin,
00038      & cnpts,cstepl,cnumbr,csizes,csizel,
00039 C
00040      & cxyneat,cxyzero,cxyloc,cxylab,cxyden,cxytics,
00041      & cxylen,cxyfrm,cxymtcs,cxymfrm,cxydec,
00042      & cxydmin,cxydmax,cxysmin,cxysmax,cxytype,
00043      & cxylsig,cxywdth,cxyepon,cxystep,cxystag,cxyetyp,
00044      & cxybeg,cxyend,cxymbeg,cxymend,cxyamin,cxyamax
00045 C
00046 C     & reserv(8)
00047       save /g2dag2/
00048
00049       integer G2dAG2L         ! Benoetigt von SAVCOM, RESCOM
00050       parameter(g2dag2l=65)  ! integer, real und logical gleich lang!
00051 C> \endcond
```

3.25 hdcopy.for File Reference

DOS Port: Hardcopy.

Functions/Subroutines

- subroutine hdcopy
- subroutine writebuf (iHandle, Buf, iPtr, iWrite)

3.25.1 Detailed Description

DOS Port: Hardcopy.

Version

1.35

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

TCS Hardcopy from Screen

Definition in file hdcopy.for.

3.25.2 Function/Subroutine Documentation

3.25.2.1 hdcopy()

```
subroutine hdcopy
```

Definition at line 40 of file hdcopy.for.

3.25.2.2 writebuf()

```
subroutine writebuf (
             integer*2 iHandle,
             integer*1, dimension(1) Buf,
             integer iPtr,
             integer iWrite )
```

Definition at line 241 of file hdcopy.for.

3.26 hdcopy.for

```
00001 C> \file       hdcopy.for
00002 C> \brief      DOS Port: Hardcopy
00003 C> \version    1.35
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C> TCS Bildschirmhardcopy
00009 C> \~english
00010 C> TCS Hardcopy from Screen
00011 C> \~
00012 C>
00013 C
00014 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00015 C
00016 C  TCS Graphik Hardcopy für DOS
00017 C
00018 C     Version 1.1
00019 C
00020 C           subroutine HDCOPY: Erzeugt Windows-Bitmapfile der Form HDCxxx.bmp
00021 C
00022 C      21.11.01         Dr.-Ing. K. Friedewald
00023 C
00024 C     08.02.02 Version 1.2
00025 C           Implementierung multilinguale Meldungen
00026 C
00027 C     31.05.02 Version 1.3:
00028 C           Ersatz Hex-Konstante durch Dezimalkonstante zur Erzielung Kompatibilität mit
       WATCOM-Kompiler
00029 C           INCLUDE Interface TCSDOSA.FI zur Anpassung an den WATCOM-Compiler
00030 C
00031 C     19.10.02 Version 1.34
00032 C           Umbenennung TKTRNX.FOR in TKTRNX.FD zur Kompatibilität CP/M
00033 C
00034 C     06.02.03 Version 1.35
00035 C           Interne Umbenennung lib$movc3 in lib_movc3
00036 C
00037       include 'FGRAPH.FI'
00038       include 'TCSdDOSa.FI'
00039
00040       subroutine hdcopy
00041       include 'TKTRNX.FD'
00042       include  'FGRAPH.FD'
```

```
00043       structure /bitmapfileheader/
00044             integer*2   DatKennung  ! = $4d42
00045             integer*4   DatSize     ! Bilddateigroesse in Byte
00046             integer*2   Reserved1
00047             integer*2   Reserved2
00048             integer*4   GraphDatDst ! Entfernung BITMAPFILEHEADER zu Graphikdaten (Byte)
00049       end structure
00050       structure /bitmapinfoheader/
00051             integer*4   BMpInfHdSiz ! Größe Bitmapinfoheader in Byte
00052             integer*4   PicWidth    ! Bildbreite Pixel, abgespeicherte Bytes durch 4 teilbar!
00053             integer*4   PicHeight   ! Bildhöhe in Pixel
00054             integer*2   iLayer      ! = 1
00055             integer*2   iBitPix     ! Bits per Pixel (1,4,8,24)
00056             integer*4   Kompr       ! Komprimierung =0(ohne),1(RLE8),2(RLE4)
00057             integer*4   PicSiz      ! Bildgroesse in Byte
00058             integer*4   HorPixDen   ! Horizontale Auflösung Pixel/ Meter
00059             integer*4   VerPixDen   ! Vertikale Auflösung Pixel/ Meter
00060             integer*4   iCol        ! Anzahl benutzte Farben
00061             integer*4   iVIPCol     ! Anzahl wichtige Farben =0(alle)
00062       end structure
00063       structure /rgbquad/
00064             integer*1   Blue
00065             integer*1   Green
00066             integer*1   Red
00067             integer*1   Reserved    ! =0
00068       end structure
00069       structure /fileheader/
00070             record /bitmapfileheader/  bfh
00071             record /bitmapinfoheader/  bih
00072             record /rgbquad/           palette(16)
00073       end structure
00074
00075       record /fileheader/ filhead
00076
00077       integer iWrtBuf
00078       parameter(iwrtbuf=650)
00079       integer*1 Buf(iWrtBuf)            ! > 2* (VGA-Auflösung/2)
00080       equivalence(buf,filhead)
00081
00082
00083       integer nByteRow
00084       integer iPtr, iPathlen
00085       integer*2 iHandle, ierr
00086       character*10 FilNam, Path*80
00087
00088       call graphicerror (10,' ') ! Hardcopy in progress
00089 c
00090 c  Initialisierung Fileheader
00091 c
00092       nbyterow=(kscrx+7-mod(kscrx-1,8))/2 ! Byte pro Zeile durch 4 teilbar
00093       if (2*nbyterow.gt.iwrtbuf) then
00094        call graphicerror (8, ' ') ! Hardcopy: Write Buffer Overflow
00095       end if
00096
00097       filhead.bfh.datkennung= 19778 ! = 4d42h
00098
00099       filhead.bfh.reserved1= 0
00100       filhead.bfh.reserved2= 0
00101
00102       filhead.bfh.graphdatdst= 118 ! = 76h
00103       filhead.bfh.datsize=nbyterow*(kscry+1) + filhead.bfh.graphdatdst
00104
00105       filhead.bih.bmpinfhdsiz= 40 ! = 28h
00106       filhead.bih.picwidth= kscrx+1
00107       filhead.bih.picheight= kscry+1
00108
00109       filhead.bih.ilayer= 1
00110       filhead.bih.ibitpix=4         ! Auch bei Monochrom???
00111       filhead.bih.kompr= 0
00112       filhead.bih.picsiz= 0         ! nicht verwendet
00113       filhead.bih.horpixden= 0
00114       filhead.bih.verpixden= 0
00115       filhead.bih.icol= 0
00116       filhead.bih.ivipcol= 0
00117
00118       filhead.palette(1).red= 0
00119       filhead.palette(1).green= 0
00120       filhead.palette(1).blue= 0
00121
00122       filhead.palette(2).red= 0
00123       filhead.palette(2).green= 0
00124       filhead.palette(2).blue= 160
00125
00126       filhead.palette(3).red= 0
00127       filhead.palette(3).green= 160
00128       filhead.palette(3).blue= 0
00129
```

```
00130       filhead.palette(4).red= 0
00131       filhead.palette(4).green= 160
00132       filhead.palette(4).blue=160
00133
00134       filhead.palette(5).red= 160
00135       filhead.palette(5).green= 0
00136       filhead.palette(5).blue= 0
00137
00138       filhead.palette(6).red= 160
00139       filhead.palette(6).green= 0
00140       filhead.palette(6).blue= 160
00141
00142       filhead.palette(7).red= 160
00143       filhead.palette(7).green= 80
00144       filhead.palette(7).blue= 0
00145
00146       filhead.palette(8).red= 160
00147       filhead.palette(8).green= 160
00148       filhead.palette(8).blue= 160
00149
00150       filhead.palette(9).red= 80
00151       filhead.palette(9).green= 80
00152       filhead.palette(9).blue= 80
00153
00154       filhead.palette(10).red= 80
00155       filhead.palette(10).green= 80
00156       filhead.palette(10).blue= 240
00157
00158       filhead.palette(11).red= 80
00159       filhead.palette(11).green= 240
00160       filhead.palette(11).blue= 80
00161
00162       filhead.palette(12).red= 80
00163       filhead.palette(12).green= 240
00164       filhead.palette(12).blue= 240
00165
00166       filhead.palette(13).red= 240
00167       filhead.palette(13).green= 80
00168       filhead.palette(13).blue= 80
00169
00170       filhead.palette(14).red= 240
00171       filhead.palette(14).green= 80
00172       filhead.palette(14).blue= 240
00173
00174       filhead.palette(15).red= 240
00175       filhead.palette(15).green= 240
00176       filhead.palette(15).blue= 80
00177
00178       filhead.palette(16).red= 240
00179       filhead.palette(16).green= 240
00180       filhead.palette(16).blue= 240
00181
00182       do 3 i=1,16
00183  3     filhead.palette(i).reserved= 0
00184 c
00185 c Create Filename and open
00186 c
00187       path= 'SPL='//char(0)
00188       call getenv (path, len(path))
00189       ipathlen=istringlen(path)
00190
00191       i=0
00192 5     continue
00193        i= i+1
00194        write (filnam,fmt=300) i
00195        if (ipathlen.gt.0) then
00196         call openbytfil(ierr,ihandle,
00197      1                  path(:ipathlen)//'\'//filnam//char(0))
00198        else
00199         call openbytfil(ierr,ihandle, filnam//char(0))
00200        end if
00201       if (ierr.eq.80) goto 5  ! File exists - increase FilNam
00202       if (ierr.ne.0) call graphicerror (6, ' ') ! Hardcopy: Error during OPEN
00203 c
00204 c Zeilenweises Auslesen Bildschirmspeicher, Puffern und Fileausgabe
00205 c
00206       iptr= filhead.bfh.graphdatdst +1
00207
00208       do 20 iy=kscry,0,-1      ! oder 1?
00209        ix=0
00210  10    continue                    ! repeat
00211         buf(iptr)= ishl(getpixel(ix,iy),4)
00212         ix= ix+1
00213         if(ix.le.kscrx)buf(iptr)=buf(iptr).or.(getpixel(ix,iy).and.15)
00214         iptr= iptr+1
00215         ix=ix+1
00216        if (ix.le.kscrx) goto 10
```

```
00217         ix=ix                          ! Anzahl belegter Halfbytes
00218   15    if (ix.lt.2*nbyterow) then     ! do while
00219          buf(iptr)= 0
00220          iptr= iptr+1
00221          ix=ix+2
00222          goto 15
00223         end if                          ! end while
00224         call writebuf (ihandle, buf(1),iptr, 256)
00225   20    continue
00226 c
00227 c Empty Buffer and Close File
00228 c
00229       call wrtbytfil (ierr, ihandle, buf(1), iptr)
00230       if (ierr.ne.0) call graphicerror (7, ' ') ! Hardcopy: Error during WRITE
00231
00232       call closebytfil (ihandle)
00233       call statst (' ')
00234       return
00235
00236 300   format ('HDC',i3.3,'.BMP')
00237       end
00238
00239
00240
00241       subroutine writebuf (iHandle, Buf, iPtr, iWrite)
00242       integer*1 Buf(1)
00243       integer iPtr, iWrite
00244       integer*2 iHandle
00245       integer*2 iErr
00246   10  continue
00247        if (iptr.le.iwrite) return
00248        call wrtbytfil (ierr, ihandle, buf(1), iwrite)
00249        if (ierr.ne.0) call graphicerror (7, ' ') ! Hardcopy: Error during WRITE
00250        call lib_movc3 (iptr-iwrite,buf(iwrite+1), buf(1))
00251        iptr= iptr-iwrite
00252       goto 10
00253       end
00254
00255
```

3.27 Mainpage.dox File Reference

3.28 outtext.for File Reference

DOS Port: alphanumeric output to the graphic screen.

Functions/Subroutines

- subroutine outtext (text)

3.28.1 Detailed Description

DOS Port: alphanumeric output to the graphic screen.

Version

1.0

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Version

1.0

Unification of the Watcom and Microsoft version

Definition in file outtext.for.

3.28.2 Function/Subroutine Documentation

3.28.2.1 outtext()

```
subroutine outtext (
             character *(*) text )
```

Definition at line 23 of file outtext.for.

3.29 outtext.for

```
00001 C> \file       outtext.for
00002 C> \version    1.0
00003 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00004 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00005 C>
00006 C> \~german
00007 C> \brief   DOS Port: Textausgabe in den Grafikbereich
00008 C> \~english
00009 C> \brief   DOS Port: alphanumeric output to the graphic screen
00010 C> \~
00011 C> \version 1.0
00012 C> \~german
00013 C> Angleichung der Watcom-Graphikroutine an die MS-Version
00014 C> \~english
00015 C> Unification of the Watcom and Microsoft version
00016 C> \~
00017 C>
00018 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00019 C OUTTEXT.FOR - Angleichung der Watcom-Graphikroutine an die MS-Version
00020 C
00021       include 'FGRAPH.FI'
00022
00023       subroutine outtext (text) ! Angleichung an MS-Version
00024       character *(*) text
00025       character *(81) TextBuf
00026       textbuf= text//char(0)
00027       call _outtext (textbuf)
00028       return
00029       end
00030
```

3.30 Strings.for File Reference

TCS: String functions.

Functions/Subroutines

- subroutine substitute (Source, Destination, Old1, New1)
- integer function istringlen (String)
- character ∗(∗) function printstring (String)
- integer function itrimlen (string)

3.30.1 Detailed Description

TCS: String functions.

Version

1.26

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Fortran utility functions for string processing

Definition in file Strings.for.

3.30.2 Function/Subroutine Documentation

3.30.2.1 istringlen()

```
integer function istringlen (
             character ∗(∗) String )
```

Definition at line 94 of file Strings.for.

3.30.2.2 itrimlen()

```
integer function itrimlen (
             character ∗(∗) string )
```

Definition at line 133 of file Strings.for.

3.30.2.3 printstring()

```
character*(*) function printstring (
             character, dimension(*) String )
```

Definition at line 114 of file Strings.for.

3.30.2.4 substitute()

```
subroutine substitute (
             character *(*) Source,
             character *(*) Destination,
             character *(*) Old1,
             character *(*) New1 )
```

Definition at line 30 of file Strings.for.

3.31 Strings.for

```
00001 C> \file       Strings.for
00002 C> \brief      TCS: String functions
00003 C> \version    1.26
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Hilfsfunktionen zur Fortran Stringverarbeitung
00008 C> \~english
00009 C> Fortran utility functions for string processing
00010 C> \~
00011 C>
00012 C
00013 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
00014 C
00015 C  Unterprogramme zur Behandlung von Fortran-Strings.
00016 C  Die Stringenden werden entweder durch CHAR(0) markiert oder
00017 C  ueber die Deklaration ermittelt.
00018 C
00019 C     9.11.88     K. Friedewald
00020 C
00021 C  Ergaenzungen:
00022 C     iTrimLen
00023 C
00024 C     7.12.01     K. Friedewald
00025 C
00026 C  Version: 1.26
00027 C
00028 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
00029
00030       subroutine substitute (Source, Destination, Old1, New1)
00031 C
00032 C  Durchsucht SOURCE nach den Substrings OLD, ersetzt sie durch NEW
00033 C  und uebergibt das Ergebniss in DESTINATION. Wenn New=CHAR(0), werden
00034 C  die vorkommenden OLD nur geloescht.
00035 C
00036 C  Stringenden koennen durch CHAR(0) markiert werden.
00037 C
00038       implicit none
00039       integer iNext, iNext2, TempLen
00040       integer iStringLen
00041       character *(*) Source, Destination, Old1, New1
00042       character*255 temp, old, new
00043
00044       if (istringlen(old1).le.0) return
00045       if (istringlen(source) .le. 0) then
00046        destination= char(0)
00047        return
00048       end if
00049
00050       old= old1 // char(0)          ! old evtl. = Destination
```

```
00051       new= new1 // char(0)          ! => retten!
00052
00053       temp= source(1:istringlen(source)) // char(0) ! evtl. Ueberlappung!
00054       destination= temp
00055       inext= index( destination(:istringlen(destination)),
00056      1                                        old(:istringlen(old)) )
00057       do while (inext.gt.0)
00058        if (inext.eq.1) then
00059         temp= destination
00060         if (new.eq.char(0)) then
00061          destination= temp(istringlen(old)+1:)
00062         else
00063          destination= new(:istringlen(new)) // temp(istringlen(old)+1:)
00064         end if
00065        else
00066         temp= destination(1:inext-1)
00067         templen= inext-1
00068         if (new.ne.char(0)) then
00069          temp= temp(1:templen)//new
00070          templen= templen+istringlen(new)
00071         end if
00072         if (inext+istringlen(old).lt.len(destination)) then
00073          temp= temp(1:templen)//destination(inext+istringlen(old):)
00074         end if
00075         destination= temp
00076        end if
00077        inext2= inext+istringlen(new)
00078        if (inext2.lt.len(destination)) then
00079         inext2= index(destination(inext2:), old(:istringlen(old)) )
00080        else
00081         inext2=0
00082        end if
00083        if (inext2.gt.0) then
00084         inext= inext+istringlen(new)+inext2-1
00085        else
00086         inext=0
00087        end if
00088       end do
00089       return
00090       end
00091
00092
00093
00094       function istringlen (String)
00095 C
00096 C Ermittelt die Stringlänge bei durch char(0) abgeschlossenen STRINGs.
00097 C Falls kein char(0) vorhanden ist, wird die Gesamtlänge übergeben.
00098 C
00099       implicit none
00100       character *(*) string
00101       integer istringlen, i
00102
00103       i= index(string,char(0))-1
00104       if (i.ge.0) then
00105        istringlen=i
00106       else
00107        istringlen= len(string)
00108       end if
00109       return
00110       end
00111
00112
00113
00114       character*(*) function printstring (String)
00115 C
00116 C  Kopiert STRING in einen variabel langen PRINTSTRING. Hierdurch wird
00117 C  der Ausdruck von Nullstrings (Fortran-Fehler!) vermieden.
00118 C
00119       implicit none
00120       character string *(*)
00121       integer istringlen
00122
00123       if (istringlen(string).gt.0) then
00124        printstring= string(1:istringlen(string))
00125       else
00126        printstring= ' '
00127       end if
00128       return
00129       end
00130
00131
00132
00133       integer function itrimlen (string)
00134 C
00135 C  Bestimmt die Länge des Strings ohne angehängte Leerzeichen.
00136 C  Bei Bedarf wird ein Char(0) angehaengt. Es darf in Ftn77 nie ein
00137 C  Nullstring erzeugt werden, da sonst die RTL-Library abstuerzt. Deswegen
```

```
00138 C  ist der kleinste erzeugte String ein Blank ' '.
00139 C
00140       implicit none
00141       character *(*) string
00142       integer i, istringlen
00143
00144       i=istringlen(string) +1
00145
00146  10   continue
00147        i= i-1
00148        if (i.ge.1) then
00149         if (string(i:i).eq.' ') goto 10
00150       end if
00151       itrimlen=i
00152       if ((i.lt.len(string)).and.(len(string).gt.1)) then
00153        string(i+1:i+1)= char(0) ! .gt.1: Achtung, nie Nullstring erzeugen!
00154       end if
00155       return
00156       end
00157
```

3.32 TCS.for File Reference

TCS: Tektronix Plot 10 Emulation.

Functions/Subroutines

- subroutine vcursr (IC, X, Y)
- subroutine drawr (X, Y)
- subroutine mover (X, Y)
- subroutine pointr (X, Y)
- subroutine dashr (X, Y, iL)
- subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
- subroutine drawa (X, Y)
- subroutine movea (X, Y)
- subroutine pointa (X, Y)
- subroutine dasha (X, Y, iL)
- subroutine wincot (X, Y, IX, IY)
- subroutine revcot (IX, IY, X, Y)
- subroutine anstr (NChar, IStrin)
- subroutine ancho (ichar)
- subroutine newlin
- subroutine cartn
- subroutine linef
- subroutine baksp
- subroutine newpag
- function linhgt (Numlin)
- function linwdt (NumChr)
- subroutine lintrn
- subroutine logtrn (IMODE)
- subroutine twindo (IX1, IX2, IY1, IY2)
- subroutine swindo (IX, LX, IY, LY)
- subroutine dwindo (X1, X2, Y1, Y2)
- subroutine vwindo (X, XL, Y, YL)
- subroutine rescal
- subroutine rrotat (Grad)
- subroutine rscale (Faktor)
- subroutine home
- subroutine setmrg (Mlinks, Mrecht)
- subroutine seetrm (IBaud, Iterm, ICSize, MaxScr)
- subroutine seetrn (xf, yf, key)
- logical function genflg (ITEM)

3.32.1 Detailed Description

TCS: Tektronix Plot 10 Emulation.

Version

4.1

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

System independent subroutines

Definition in file TCS.for.

3.32.2 Function/Subroutine Documentation

3.32.2.1 ancho()

```
subroutine ancho (
              ichar )
```

Definition at line 339 of file TCS.for.

3.32.2.2 anstr()

```
subroutine anstr (
              NChar,
             dimension(1) IStrin )
```

Definition at line 329 of file TCS.for.

3.32.2.3 baksp()

```
subroutine baksp
```

Definition at line 384 of file TCS.for.

3.32.2.4 cartn()

```
subroutine cartn
```

Definition at line 365 of file TCS.for.

3.32.2.5 dasha()

```
subroutine dasha (
              X,
              Y,
              iL )
```

Definition at line 290 of file TCS.for.

3.32.2.6 dashr()

```
subroutine dashr (
              X,
              Y,
              iL )
```

Definition at line 236 of file TCS.for.

3.32.2.7 drawa()

```
subroutine drawa (
              X,
              Y )
```

Definition at line 257 of file TCS.for.

3.32.2.8 drawr()

```
subroutine drawr (
              X,
              Y )
```

Definition at line 212 of file TCS.for.

3.32.2.9 dwindo()

```
subroutine dwindo (
              X1,
              X2,
              Y1,
              Y2 )
```

Definition at line 462 of file TCS.for.

3.32.2.10 genflg()

```
logical function genflg (
              ITEM )
```

Definition at line 558 of file TCS.for.

3.32.2.11 home()

```
subroutine home
```

Definition at line 518 of file TCS.for.

3.32.2.12 linef()

```
subroutine linef
```

Definition at line 374 of file TCS.for.

3.32.2.13 linhgt()

```
function linhgt (
              Numlin )
```

Definition at line 400 of file TCS.for.

3.32.2.14 lintrn()

```
subroutine lintrn
```

Definition at line 418 of file TCS.for.

3.32.2.15 linwdt()

```
function linwdt (
              NumChr )
```

Definition at line 408 of file TCS.for.

3.32.2.16 logtrn()

```
subroutine logtrn (
              IMODE )
```

Definition at line 428 of file TCS.for.

3.32.2.17 movea()

```
subroutine movea (
              X,
              Y )
```

Definition at line 268 of file TCS.for.

3.32.2.18 mover()

```
subroutine mover (
              X,
              Y )
```

Definition at line 220 of file TCS.for.

3.32.2.19 newlin()

```
subroutine newlin
```

Definition at line 357 of file TCS.for.

3.32.2.20 newpag()

```
subroutine newpag
```

Definition at line 392 of file TCS.for.

3.32.2.21 pointa()

```
subroutine pointa (
              X,
              Y )
```

Definition at line 279 of file TCS.for.

3.32.2.22 pointr()

```
subroutine pointr (
              X,
              Y )
```

Definition at line 228 of file TCS.for.

3.32.2.23 rel2ab()

```
subroutine rel2ab (
              Xrel,
              Yrel,
              Xabs,
              Yabs )
```

Definition at line 244 of file TCS.for.

3.32.2.24 rescal()

```
subroutine rescal
```

Definition at line 481 of file TCS.for.

3.32.2.25 revcot()

```
subroutine revcot (
              IX,
              IY,
              X,
              Y )
```

Definition at line 314 of file TCS.for.

3.32.2.26 rrotat()

```
subroutine rrotat (
              Grad )
```

Definition at line 501 of file TCS.for.

3.32.2.27 rscale()

```
subroutine rscale (
              Faktor )
```

Definition at line 510 of file TCS.for.

3.32.2.28 seetrm()

```
subroutine seetrm (
              IBaud,
              Iterm,
              ICSize,
              MaxScr )
```

Definition at line 536 of file TCS.for.

3.32.2.29 seetrn()

```
subroutine seetrn (
              xf,
              yf,
              key )
```

Definition at line 547 of file TCS.for.

3.32.2.30 setmrg()

```
subroutine setmrg (
              Mlinks,
              Mrecht )
```

Definition at line 527 of file TCS.for.

3.32.2.31 swindo()

```
subroutine swindo (
              IX,
              LX,
              IY,
              LY )
```

Definition at line 450 of file TCS.for.

3.32.2.32 twindo()

```
subroutine twindo (
              IX1,
              IX2,
              IY1,
              IY2 )
```

Definition at line 443 of file TCS.for.

3.32.2.33 vcursr()

```
subroutine vcursr (
              IC,
              X,
              Y )
```

Definition at line 202 of file TCS.for.

3.32.2.34 vwindo()

```
subroutine vwindo (
               X,
               XL,
               Y,
               YL )
```

Definition at line 469 of file TCS.for.

3.32.2.35 wincot()

```
subroutine wincot (
               X,
               Y,
               IX,
               IY )
```

Definition at line 301 of file TCS.for.

3.33 TCS.for

```
00001 C> \file       TCS.for
00002 C> \brief      TCS: Tektronix Plot 10 Emulation
00003 C> \version    4.1
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Systemübergreifende TCS-Routinen
00008 C> \~english
00009 C> System independent subroutines
00010 C> \~
00011 C
00012 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00013 C
00014 C      26.07.23 Version 5.0:
00015 C               Einheitliche Version CPM/DOS/Windows/SDL2/wX
00016 C
00017 C      27.11.20 Version 4.0:
00018 C               Einheitliche Version CPM/DOS/Windows/SDL2
00019 C
00020 C      17.08.20 Version 3.2
00021 C               Harmonisierung der Verwendung des Commonblocks TKTRNX
00022 C               Variable KHOMEY wird jetzt (analog alter DOS-Version) verwendet.
00023 C               Da KHOMEY nicht in der CP/M Version vorhanden ist, muss ab dieser
00024 C               Version fuer eine Complilation unter CP/M die entsprechende Zeile
00025 C               in der SUBROUTINE HOME geändert werden.
00026 C
00027 C      13.11.17 Version 3.1
00028 C               Anpassung an OpenWatcom 2.0
00029 C               Bugfix: Unterscheidung Aufrufe ueber windowsx.h (win16) und GDI (win32)
00030 C                - SelectPen -> SelectObject
00031 C                - DeletePen -> DeleteObject
00032 C                - DeleteBrush -> DeleteObject
00033 C                - GetStockBrush -> GetStockObject
00034 C                - DeleteRgn -> DeleteObject
00035 C                - SelectFont -> SelectObject
00036 C                - DeleteFont -> DeleteObject
00037 C
00038 C      27.03.13 Version 3.0
00039 C               Anpassung an Windows 7 und OpenWatcom 1.9
00040 C               Anpassung an gfortran anstelle von g77 der GCC
00041 C
00042 C      22.12.05 Version 2.19
00043 C               Elimination berechnetes GOTO in LOGTRN
00044 C
00045 C      18.10.05 Version 2.18
00046 C               Anpassung der Windowsversionen zur gemeinsamen Verwendung SDL2:
```

```
C                 TCSdrWIN.for
C                 TCSdWINc.h
C                 - Überfuehrung der Deklaration aus TCSdWIN.c nach *.h:
C                   GraphicError und CreateMainWindow_IfNecessary
C                 - Definition der Fehlernummern als Konstante statt enum
C               Abhaengigkeit Watcom-Defaultwindowsystem eliminiert
C               - TCSdWINc.c: Kein Abbruch bei OpenWatcom > 1.3 und
C                 definiertem Symbol trace_calls
C
C      26.10.04 Version 2.17
C               Bugfix Windows-System: Größe und Defaultposition des Status-
C                fensters wird bei der Erzeugung berechnet -> 1. RESTORE nach
C                Verkleinern des Graphikfensters entspricht dem vorherigen
C                Bild. 2. Angleichung des Verhaltens von 16- und 32bit Windows
C               Bei Definition des Symbols STAT_WINDOW_PRIVATE erhält das
C                Statusfenster einen privaten Devicekontext.
C               Zusammenfuehrung Initialisierung der Windows-Library und
C                Windows-DLL -> zusaetzliche Sourcefiles
C                TCSinitt.for, CreateMainWindow.c, GetMainInstance.c
C
C      23.06.04 Version 2.16:
C               Anpassungen an GNU-Compiler fuer Win32. Zusätzliches Sourcefile
C                fuer die GNU-Version: WinMain.c
C               CSIZE in Windows-Version: Korrektur Rundungsfehler
C
C      08.06.04 Version 2.15:
C               Umbenennung lib$movc3 in lib_movc3 (entsprechend ANSI-Fortran)
C               Modul STRINGS.FOR: Version 1.24
C
C      27.06.03 Version 2.14:
C               Verarbeitung Steuerzeichen in ANCHO
C
C      21.10.02 Version 2.13:
C               Einheitliche Version CPM/DOS/Windows
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Grundversion fuer C128 / Version 1.0:
C
C      Zugehoerige Module:
C              TKTRNX.FOR    Common-Block TKTRNX
C              TCSBASIC.ASM  Low-Level Routinen in Bank 0, C128 spezifisch
C              TCSDRIVR.ASM  Treiber fuer TCSBASIC
C              TCSGIN.ASM    Treiber des Gin-Cursors
C
C      20.4.88         Dr.-Ing. K. Friedewald
C                      4000 Duesseldorf 1
C                      Gerresheimerstr. 84
C
C      21.10.02 Version 2.13:
C               Vereinheitlichung CPM/DOS/Windowsversion
C               Zusätzliches Modul: TCSdrCPM.FOR: früher Teil von TCS.FOR
C               Ausschließliche Verwendung von durch grosses "C" eingeleiteten
C                Kommentaren zur Kompatibilität mit FORTRAN 4
C               Umbenennung des Includefiles in Tktrnx.fd. So kann unter CP/M
C                das als Teil des Filenamens interpretierte "/" der INCLUDE-
C                Anweisung entsprechend der 8.3 Filenamen umgesetzt werden.
C               Implementierung Unterprogramm TCSLEV
C               Bugfix: Kommentar in Tktrnx.fd wurde falsch gekennzeichnet
C                       (c statt C) -> SVSTAT und RESTAT fehlerhaft, da nicht
C                       erkannte Kommentare zusaetzliche Variablen erzeugten.
C
C      TBD: Implementierung vertikale Auflösung von 400 Pixeln
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassung an DOS:
C
C      Aenderungen gegenueber CP/M-Version:
C               SEELOC, DCURSR, SVSTAT, RESTAT, CSIZE in TCSdrDOS.FOR
C      Bugfix:  DASHA, DASHR - Korrektur Parameterliste
C               SEETRM - ibaud statt ibaudr
C
C      Zugehoerige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TCSdrDOS.FOR  Bildschirmtreiber
C               TCSdDOSa.ASM  Betriebssystemspezifische Low-Level Routinen
C               HDCOPY.FOR    Hardcopyroutine
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C               OUTTEXT.FOR   nur für WATCOM-Compiler
C
C      25.10.01 Version 2.00:  Dr.-Ing. K. Friedewald
C
C      07.02.02 Version 2.10:
C               Implementierung multilinguale Fehlermeldungen
C
C      11.10.02 Version 2.12:
```

```
C               Vereinheitlichung DOS/Windowsversion
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an Microsoft-Windows:
C
C      Aenderungen gegenueber DOS-Version:
C               INITT befinden sich jetzt in TCSdrWIN.FOR bzw. TCSinitt.FOR
C
C      Zugehoerige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TKTRNX.h      Common-Block TKTRNX für Zugriff durch C
C               TCSdrWIN.FOR  Bildschirmtreiber
C               TCSdWINc.c    Windowspezifische API-Routinen
C               TCSdWINc.h    Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C
C      27.10.01 Version 2.11: Dr.-Ing. K. Friedewald
C
C      11.10.02 Version 2.12:
C               Vereinheitlichung DOS/Windowsversion
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an SDL2:
C
C      Aenderungen gegenueber Windows-Version:
C               Fehlerausgabe in den Windows-Debug-Channel (bzw. *ix Fehlerkanal)
C               Statusfenster analog DOS nur einzeilig ohne Scrollmöglichkeit
C
C      Zugehoerige Module:
C               TKTRNX.FOR    identisch mit Windows-Version
C               TKTRNX.h      identisch mit Windows-Version
C               TCSdrSDL.FOR  SDL2-spezifische API-Routinen
C               TCSdSDLc.c    SDL2-spezifische API-Routinen
C               TCSdSDLc.h    Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR   identisch mit Windows-Version
C
C      27.11.20 Version 4.00: Dr.-Ing. K. Friedewald
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an WXwidgets:
C
C      Aenderungen gegenueber SDL2-Version:
C               Fehlerausgabe in den wxLogStatus
C               Statusfenster durch initt1() konfigurierbar
C
C      Zugehoerige Module:
C               TKTRNX.FOR     identisch mit Windows-Version
C               TKTRNX.hpp     identisch mit Windows-Version
C               TCSdrWXfor.f08 WX-spezifische API-Routinen
C               TCSdrWXcpp.cpp WX-spezifische API-Routinen
C               TCSdrWXcpp.hpp Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR    identisch mit Windows-Version
C               Graph2D.f08    Interfacemodul Anwenderprogramme ab Fortran 2003
C               graph2d.h      Header fuer C/Cpp Anwenderprogramme
C
C      26.07.23 Version 5.00: Dr.-Ing. K. Friedewald
C



C
C Graphic Input
C

      subroutine vcursr (IC,X,Y)
      call dcursr (ic,ix,iy)
      call revcot (ix,iy,x,y)
      return
      end

C
C  Virtuelle Graphik, relativ
C

      subroutine drawr (X,Y)
      call rel2ab (x,y,xabs,yabs)
      call drawa (xabs,yabs)
      return
      end



      subroutine mover (X,Y)
```

```
00221       call rel2ab (x,y,xabs,yabs)
00222       call movea (xabs,yabs)
00223       return
00224       end
00225
00226
00227
00228       subroutine pointr (X,Y)
00229       call rel2ab (x,y,xabs,yabs)
00230       call pointa (xabs,yabs)
00231       return
00232       end
00233
00234
00235
00236       subroutine dashr (X,Y, iL)
00237       call rel2ab (x,y,xabs,yabs)
00238       call dasha (xabs,yabs, il)
00239       return
00240       end
00241
00242
00243
00244       subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
00245       include 'Tktrnx.fd'
00246       call seeloc (ix,iy)
00247       call revcot (ix,iy,xabs,yabs)
00248       xabs= (( xrel*trcosf - yrel*trsinf)*trscal)+xabs
00249       yabs= (( xrel*trsinf + yrel*trcosf)*trscal)+yabs
00250       return
00251       end
00252
00253 C
00254 C  Virtuelles Zeichnen, absolut
00255 C
00256
00257       subroutine drawa (X,Y)
00258       include 'Tktrnx.fd'
00259       call wincot (x,y,ix,iy)
00260       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00261       call drwabs (ix,iy)
00262       call swind1 (0,0,1023,780)
00263       return
00264       end
00265
00266
00267
00268       subroutine movea (X,Y)
00269       include 'Tktrnx.fd'
00270       call wincot (x,y,ix,iy)
00271       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00272       call movabs (ix,iy)
00273       call swind1 (0,0,1023,780)
00274       return
00275       end
00276
00277
00278
00279       subroutine pointa (X,Y)
00280       include 'Tktrnx.fd'
00281       call wincot (x,y,ix,iy)
00282       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00283       call pntabs (ix,iy)
00284       call swind1 (0,0,1023,780)
00285       return
00286       end
00287
00288
00289
00290       subroutine dasha (X,Y, iL)
00291       include 'Tktrnx.fd'
00292       call wincot (x,y,ix,iy)
00293       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00294       call dshabs (ix,iy, il)
00295       call swind1 (0,0,1023,780)
00296       return
00297       end
00298
00299
00300
00301       subroutine wincot (X,Y,IX,IY)
00302       include 'Tktrnx.fd'
00303       dx= x-tminvx
00304       dy= y-tminvy
00305       if ((xlog.lt.255.).and.(x.gt.0.)) dx= alog(x)-xlog
00306       if ((ylog.lt.255.).and.(y.gt.0.)) dy= alog(y)-ylog
00307       ix= ifix(dx*xfac+.5)+kminsx
```

```
00308       iy= ifix(dy*yfac+.5)+kminsy
00309       return
00310       end
00311
00312
00313
00314       subroutine revcot (IX,IY,X,Y)
00315       include 'Tktrnx.fd'
00316       dx= float(ix-kminsx) / xfac
00317       dy= float(iy-kminsy) / yfac
00318       x= dx + tminvx
00319       y= dy + tminvy
00320       if (xlog.lt.255.) x= 2.718282**(dx+xlog)
00321       if (ylog.lt.255.) y= 2.718282**(dy+ylog)
00322       return
00323       end
00324
00325 C
00326 C  Alphanumerische Ausgabe
00327 C
00328
00329       subroutine anstr (NChar, IStrin)
00330       dimension istrin(1)
00331       do 10 i=1,nchar
00332        call ancho (istrin(i))
00333 10    continue
00334       return
00335       end
00336
00337
00338
00339       subroutine ancho (ichar)
00340       include 'Tktrnx.fd'
00341
00342       if (ichar.gt.31) goto 10
00343       if (ichar.eq.7) call bell
00344       if (ichar.eq.10) call linef
00345       if (ichar.eq.13) call cartn
00346       return
00347
00348  10   call seeloc (ix,k)
00349       call csize (ixlen,k)
00350       if (ix.gt.krmrgn-ixlen) call newlin
00351       call toutpt (ichar)
00352       return
00353       end
00354
00355
00356
00357       subroutine newlin
00358       call cartn
00359       call linef
00360       return
00361       end
00362
00363
00364
00365       subroutine cartn
00366       include 'Tktrnx.fd'
00367       call seeloc (ix,iy)
00368       call movabs (klmrgn,iy)
00369       return
00370       end
00371
00372
00373
00374       subroutine linef
00375       call seeloc (j,iy)
00376       call csize (j,iylen)
00377       if (iy.lt.iylen) call home
00378       call movrel (0,-iylen)
00379       return
00380       end
00381
00382
00383
00384       subroutine baksp
00385       call csize (ix,iy)
00386       call movrel (-ix,0)
00387       return
00388       end
00389
00390
00391
00392       subroutine newpag
00393       call erase
00394       call home
```

```
00395       return
00396       end
00397
00398
00399
00400       function linhgt (Numlin)
00401       call csize (ix,iy)
00402       linhgt= numlin*iy
00403       return
00404       end
00405
00406
00407
00408       function linwdt (NumChr)
00409       call csize (ix,iy)
00410       linwdt= numchr*ix
00411       return
00412       end
00413
00414 C
00415 C  Initialisierungsroutinen
00416 C
00417
00418       subroutine lintrn
00419       include 'Tktrnx.fd'
00420       xlog= 255.
00421       ylog= 255.
00422       call rescal
00423       return
00424       end
00425
00426
00427
00428       subroutine logtrn (IMODE)
00429       include 'Tktrnx.fd'
00430       call lintrn
00431       if ((imode .eq. 1) .or. (imode .eq. 3)) then
00432        xlog= 0.
00433       end if
00434       if ((imode .eq. 2) .or. (imode .eq. 3)) then
00435        ylog= 0.
00436       end if
00437       call rescal
00438       return
00439       end
00440
00441
00442
00443       subroutine twindo (IX1,IX2,IY1,IY2)
00444       call swindo (ix1,ix2-ix1,iy1,iy2-iy1)
00445       return
00446       end
00447
00448
00449
00450       subroutine swindo (IX,LX,IY,LY)
00451       include 'Tktrnx.fd'
00452       kminsx= ix
00453       kmaxsx= ix+lx
00454       kminsy= iy
00455       kmaxsy= iy+ly
00456       call rescal
00457       return
00458       end
00459
00460
00461
00462       subroutine dwindo (X1,X2,Y1,Y2)
00463       call vwindo (x1,x2-x1,y1,y2-y1)
00464       return
00465       end
00466
00467
00468
00469       subroutine vwindo (X,XL,Y,YL)
00470       include 'Tktrnx.fd'
00471       tminvx= x
00472       tmaxvx= x+xl
00473       tminvy= y
00474       tmaxvy= y+yl
00475       call rescal
00476       return
00477       end
00478
00479
00480
00481       subroutine rescal
```

```
00482       include 'Tktrnx.fd'
00483       xfac= 0.
00484       yfac= 0.
00485       if ((tmaxvx.eq.tminvx) .or. (tmaxvy.eq.tminvy)) return
00486       dx= tmaxvx-tminvx
00487       dy= tmaxvy-tminvy
00488       if ((xlog.eq.255.).or.(amin1(tminvx,tmaxvx).le.0.)) goto 10
00489        xlog= alog(tminvx)
00490        dx= alog(tmaxvx)-xlog
00491 10    if ((ylog.eq.255.).or.(amin1(tminvy,tmaxvy).le.0.)) goto 20
00492        ylog= alog(tminvy)
00493        dy= alog(tmaxvy)-ylog
00494 20    xfac= float(kmaxsx-kminsx) / dx
00495       yfac= float(kmaxsy-kminsy) / dy
00496       return
00497       end
00498
00499
00500
00501       subroutine rrotat (Grad)
00502       include 'Tktrnx.fd'
00503       trsinf= sin(grad/57.29578)
00504       trcosf= cos(grad/57.29578)
00505       return
00506       end
00507
00508
00509
00510       subroutine rscale (Faktor)
00511       include 'Tktrnx.fd'
00512       trscal= faktor
00513       return
00514       end
00515
00516
00517
00518       subroutine home
00519       include 'Tktrnx.fd'
00520 C      call movabs(klmrgn,750) Fuer CP/M (kein khomey verfuegbar, -> !=750)
00521       call movabs(klmrgn,khomey)
00522       return
00523       end
00524
00525
00526
00527       subroutine setmrg (Mlinks, Mrecht)
00528       include 'Tktrnx.fd'
00529       klmrgn= mlinks
00530       krmrgn= mrecht
00531       return
00532       end
00533
00534
00535
00536       subroutine seetrm (IBaud,Iterm,ICSize,MaxScr)
00537       include 'Tktrnx.fd'
00538       ibaud= 0
00539       iterm= 1
00540       icsize= 1
00541       maxscr= 1023
00542       return
00543       end
00544
00545
00546
00547       subroutine seetrn (xf,yf,key)
00548       include 'Tktrnx.fd'
00549       xf= xfac
00550       yf= yfac
00551       key= 1
00552       if ((xlog.lt.255.).or.(ylog.lt.255.)) key=2
00553       return
00554       end
00555
00556
00557
00558       logical function genflg (ITEM)
00559       genflg= item.eq.0
00560       return
00561       end
```

3.34 TCSdDosa.asm File Reference

DOS Port: x86 Assembler Routinen.

Functions

- int ktinput ()
  Tastaturabfrage.
- void bell ()
  Signalton.
- void GinCrsIn (bool iAvail, int iButton, int iXmin, int iXmax, int iYmin, int iYmax)
  Initialisierung Graphikmaus.
- void GinCrs (int ic, int ix, int iy)
  Abfrage Graphikmaus.
- void GinCrsEx ()
  Reset Graphikmaus.
- void GetEnv (char Buf, int BufLen)
  Abfrage Enviromentvariable
- void lib_movc3 (int iByte, char Source, char Dest)
  Kopieren eines Feldes
- void OpenBytFil (int iErr, int iHandle, char FilNam)
  Oeffnen eines Bytefiles.
- void WrtBytFil (int iErr, int iHandle, char buf, int iWrite)
  WrtBytFil Byteweises Schreiben ohne Steuerzeichen.
- void CloseBytFil (int iHandle)
  Schliesen eines Bytefiles.

3.34.1 Detailed Description

DOS Port: x86 Assembler Routinen.

Version

1.4 ;

Author

(C) 2022 Dr.-Ing. Klaus Friedewald ;

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Definition in file TCSdDosa.asm.

3.34.2 Function Documentation

3.34.2.1 bell()

```
void bell ( )
```

Signalton.

3.34.2.2 CloseBytFil()

```
void CloseBytFil (
            int iHandle )
```

Schliesen eines Bytefiles.

Parameters

in	*iHandle*	Filehandle

3.34.2.3 GetEnv()

```
void GetEnv (
            char Buf,
            int BufLen )
```

Abfrage Enviromentvariable

Parameters

in,out	*Buf*	in=Variable out=Uebersetzung
in	*BufLen*	

3.34.2.4 GinCrs()

```
void GinCrs (
            int ic,
            int ix,
            int iy )
```

Abfrage Graphikmaus.

Parameters

out	*ic*	Gedrueckte Taste
out	*ix,iy*	Cursorposition

3.34.2.5 GinCrsEx()

```
void GinCrsEx ( )
```

Reset Graphikmaus.

3.34.2.6 GinCrsIn()

```
void GinCrsIn (
            bool iAvail,
            int iButton,
            int iXmin,
            int iXmax,
            int iYmin,
            int iYmax )
```

Initialisierung Graphikmaus.

Parameters

out	*iAvail*	Maus vorhanden
out	*iButton*	Anzahl Tasten
in	*iXmin,iXmax,iYmin,iYmax*	Zeichenfläche

3.34.2.7 ktinput()

```
int ktinput ( )
```

Tastaturabfrage.

Parameters

out	*[↩ AX]*	Funktionsrückgabe ASCII

3.34.2.8 lib_movc3()

```
void lib_movc3 (
            int iByte,
            char Source,
            char Dest )
```

Kopieren eines Feldes

Parameters

in	*iByte*	Anzahl verschiebender Bytes (0 zulässig)
in	*Source*	zu kopierende Daten
out	*Dest*	Zielfeld, kann auch Source überlappen

3.34.2.9 OpenBytFil()

```
void OpenBytFil (
            int iErr,
            int iHandle,
            char FilNam )
```

Oeffnen eines Bytefiles.

Parameters

out	*iErr*	Errorflag
out	*iHandle*	Filehandle
in	*FilNam*	Dateiname

3.34.2.10 WrtBytFil()

```
void WrtBytFil (
            int iErr,
            int iHandle,
            char buf,
            int iWrite )
```

WrtBytFil Byteweises Schreiben ohne Steuerzeichen.

Parameters

out	*iErr*	Errorflag
in	*iHandle*	Filehandle
in	*buf*	Daten
in	*iWrite*	Anzahl zu schreibender Bytes

3.35 TCSdDosa.asm

```
00001 ; // DOXYGEN Dokumentation TCSdDOS.asm: als C-Programm möglich da ";" C-Leerbefehl entspricht
00002 ; /** \file TCSdDosa.asm \brief DOS Port: x86 Assembler Routinen \version 1.4
00003 ; \author  (C) 2022 Dr.-Ing. Klaus Friedewald
00004 ; \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3   */
00005
00006 ; //! \brief Tastaturabfrage \param[out] [AX] Funktionsrückgabe ASCII
00007 ; (int) ktinput ()
00008
00009 ; //! \brief Signalton
00010 ; (void) bell ()
00011
00012 ; //! \brief Initialisierung Graphikmaus
00013 ; //! \param[out] iAvail Maus vorhanden
00014 ; //! \param[out] iButton Anzahl Tasten
00015 ; //! \param[in] iXmin, iXmax, iYmin, iYmax Zeichenfläche
00016 ; (void) GinCrsIn (bool iAvail,int iButton,int iXmin,int iXmax,int iYmin,int iYmax)
00017
00018 ; //! \brief Abfrage Graphikmaus
00019 ; //! \param[out] ic Gedrueckte Taste
00020 ; //! \param[out] ix, iy  Cursorposition
00021 ; (void) GinCrs (int ic,int ix,int iy)
00022
00023 ; //! \brief Reset Graphikmaus
00024 ; (void) GinCrsEx ()
00025
00026 ; //! \brief Abfrage Enviromentvariable
00027 ; //! \param[in,out] Buf in=Variable out=Uebersetzung
00028 ; //! \param[in]   BufLen
00029
00030 ; (void) GetEnv (char Buf, int BufLen)
00031 ; //! \brief Kopieren eines Feldes
00032 ; //! \param[in] iByte Anzahl verschiebender Bytes (0 zulässig)
00033 ; //! \param[in] Source zu kopierende Daten
00034 ; //! \param[out] Dest Zielfeld, kann auch Source überlappen
00035
00036 ; (void) lib_movc3 (int iByte, char Source,char Dest)
00037 ; //! \brief Oeffnen eines Bytefiles
00038 ; //! \param[out] iErr Errorflag
00039 ; //! \param[out] iHandle Filehandle
00040 ; //! \param[in] FilNam Dateiname
00041
00042 ; (void) OpenBytFil(int iErr,int iHandle,char FilNam)
00043 ; //! \brief WrtBytFil   Byteweises Schreiben ohne Steuerzeichen
00044 ; //! \param[out] iErr Errorflag
00045 ; //! \param[in] iHandle Filehandle
00046 ; //! \param[in] buf Daten
00047 ; //! \param[in] iWrite Anzahl zu schreibender Bytes
00048
00049 ; (void) WrtBytFil (int iErr,int iHandle, char buf, int iWrite)
00050 ; //! \brief Schliesen eines Bytefiles
00051 ; //! \param[in] iHandle Filehandle
00052
00053 ; (void) CloseBytFil (int iHandle)
00054 ; //! \cond
00055 ; ---------------------------   Changelog   ------------------------------
00056 ;
00057 ;  Version 1.2
00058 ;     25.10.01               Dr. Ing. K. Friedewald
00059 ;
00060 ;           ktinput:    Tastaturabfrage
00061 ;           bell:       Signalton
00062 ;           GinCrsIn:   Initialisierung Graphikmaus
00063 ;           GinCrs:     Abfrage Graphikmaus
00064 ;           GinCrsEx:   Wiederherstellen Graphikmaus
00065 ;
00066 ;           GetEnv:     Abfrage Enviromentvariable (C-Characterformat!)
00067 ;                        Input: Pufferfeld, Vorbesetzt mit Variablenname
00068 ;                               max. Länge Pufferfeld (einschliesslich char(0))
00069 ;                        Output:Pufferfeld, Übersetzter Wert
00070 ;
00071 ;           Lib_movC3   Kopieren eines Feldes
00072 ;                        Input: iByte, Anzahl verschiebender Bytes (0 zulässig)
00073 ;                               Source, zu kopierende Daten
00074 ;                        Output:Dest, Zielfeld, kann auch Source überlappen
00075 ;
00076 ;           OpenBytFil  Oeffnen eines Bytefiles
00077 ;                        Input: FilNam
00078 ;                        Output:iErr, iHandle
00079 ;
00080 ;           WrtBytFil   Byteweises Schreiben ohne Steuerzeichen
00081 ;                        Input: iHandle, Buf(*), iCount
00082 ;                        Output:iErr
00083 ;
00084 ;           CloseBytFil Schliesen eines Bytefiles
00085 ;                         Input: iHandle
```

```
00086 ;
00087 ;
00088 ;
00089 ;  Version 1.31
00090 ;     30.05.02                Dr. Ing. K. Friedewald
00091 ;
00092 ;           Anpassung an WATCOM-Assembler:
00093 ;            Auskommentieren der Microsoft-spezifischen Assemblerdirektiven
00094 ;            .no87, .list, title, subtitle, page
00095 ;           Bugfix: Fehlerhafte Parameterübergabe WRTBYTFIL:
00096 ;                    DS von Buf wurde überschrieben
00097 ;                    iErr jetzt übergeben (Programm: MOV, Deklaration:Offset)
00098 ;
00099 ;
00100 ;  Version 1.32
00101 ;     25.10.02                Dr. Ing. K. Friedewald
00102 ;
00103 ;           Bugfix: Schnell aufeinanderfolgende GINCRS-Aufrufe fehlerhaft
00104 ;                   Warten auf nicht gedrueckte Maustaste ergaenzt
00105 ;
00106 ;  Version 1.33
00107 ;     29.10.04                Dr. Ing. K. Friedewald
00108 ;
00109 ;           Anpassung an OpenWatcom-Linker 1.3: Großschreibung PUBLIC-Symbole
00110 ;
00111 ;  Version 1.4
00112 ;     04.12.20                Dr. Ing. K. Friedewald
00113 ;
00114 ;           Dokumentation durch DOXYGEN
00115 ;
00116 ;
00117
00118 ;           title       'TCS Assembler Routinen'
00119             .8086
00120 ;            .no87
00121 ;            .list
00122             .model large
00123
00124             public      KTINPUT     ; FORTRAN: integer*2 function ktinput ()
00125
00126             public      BELL        ; FORTRAN: call bell ()
00127
00128             public      GINCRS      ; FORTRAN: call gincrs (ic,ix,iy)
00129 iC          equ   [BP] + 14         ; Integer*2 (Rückgabe 1,2: linke,rechte Maustaste sonst ASCII
00130 iX          equ   [BP] + 10         ; Integer*2
00131 iY          equ   [BP] + 6          ; Integer*2
00132
00133             public      GINCRSIN    ; FORTRAN: call gincrsIn (iAvail, iButton, iX0,iX1,iY0,iY1)
00134 iAvail      equ   [BP] + 26         ; Integer*2 oder Logical*2
00135 iButton     equ   [BP] + 22         ; Integer*2
00136 iX0         equ   [BP] + 18         ; Integer*2
00137 iX1         equ   [BP] + 14         ; Integer*2
00138 iY0         equ   [BP] + 10         ; Integer*2
00139 iY1         equ   [BP] + 6          ; Integer*2
00140
00141             public      GINCRSEX    ; FORTRAN: call GinCrsEx ()
00142
00143             public      GETENV      ; FORTRAN: call GetEnv (CHARBUF, CharBufL)
00144 CharBuf     equ   [BP] + 10         ; Vorbesetzt mit "NAME="//char(0)
00145 CharBufL    equ   [BP] + 6
00146
00147             public      OPENBYTFIL  ; FORTRAN: call OpenBytFil (iErr, iHandle, Filnam)
00148 iErrO       equ   [BP] + 14
00149 iHandleO    equ   [BP] + 10         ; integer*2 iHandle <> 0 falls o.k.
00150 FilNam      equ   [BP] + 6          ; C-String
00151
00152             public      WRTBYTFIL   ; FORTRAN: call WrtBytFil (iErr, iHandle, Buf, iCount)
00153 iErr        equ   [BP] + 18
00154 iHandle     equ   [BP] + 14         ; Integer*2
00155 Buf         equ   [BP] + 10         ; byte array
00156 iCount      equ   [BP] + 6          ; Integer*2
00157
00158             public      CLOSEBYTFIL ; FORTRAN: call CloseBytFil (iHandle)
00159 iHandleC    equ   [BP] + 6
00160
00161             public      LIB_MOVC3_  ; FORTRAN: call Lib_MovC3_ (iByte, Source, Dest)
00162 iByte       equ   [BP] + 14
00163 Source      equ   [BP] + 10
00164 Dest        equ   [BP] + 6
00165
00166 TCSdDosA_data segment public 'DATA'   ; obligatorischer Name für MS-Compiler
00167
00168
00169 CrsDefHotX  equ   0                   ; Definition Graphikmousecursor
00170 CrsDefHotY  equ   0                   ; Vorsicht, Cursor kann nicht über linke, obere Ecke geclippt
       werden!
00171 CrsDef      dw    16 dup (0ffffh)   ; Screenmask (wird AND verküpft)
```

```
00172             dw    07c00h, 0c000h    ; Cursorform (wird XOR verknüpft)
00173             dw    0a000h, 09000h
00174             dw    08800h, 08400h
00175             dw    00200h, 00100h
00176             dw    00080h, 00000h
00177             dw    00000h, 00000h
00178             dw    00000h, 00000h
00179             dw    00000h, 00000h
00180
00181 TCSdDosA_data  ends
00182
00183 DGROUP      group TCSdDosA_data
00184
00185 ;             subtitle    'TCS Basisfunktionen'
00186 ;             page
00187
00188 TcsdDosA_text segment public 'code'   ; obligatorischer Name für MS-Compiler
00189
00190             assume CS:TcsdDosA_text, DS:DGROUP, SS:DGROUP
00191
00192 DOS         equ   021h              ; DOS-Interrupt
00193 MOUSE       equ   033h              ; Mousedriver
00194 VideoBIOS   equ   010h
00195
00196 ;
00197 ; ********************
00198 ; *                  *
00199 ; * Function KTINPUT *
00200 ; *                  *
00201 ; ********************
00202 ;
00203
00204 ktinput     proc far
00205
00206             push  bp
00207             mov   bp,sp             ; lokale Basis
00208             push  ds
00209
00210             mov   ah, 07h           ; DOS 7: Zeichen ohne Echo einlesen
00211             int   DOS
00212             mov   ah,0h
00213
00214             pop   ds
00215             pop   bp
00216             ret
00217
00218 ktinput     endp
00219 ;
00220 ; ******************
00221 ; *                *
00222 ; * Subroutine BELL *
00223 ; *                *
00224 ; ******************
00225 ;
00226 bell        proc far
00227
00228             push  bp
00229             mov   bp,sp             ; lokale Basis
00230             push  ds
00231
00232             mov   ah, 0eh           ; Video-Bios: TTY Out
00233             mov   al, 07h           ; Bell
00234             mov   bh,0              ; Bildschirmnummer
00235             mov   bl,0              ; Grafik-Vordergrundfarbe
00236             int   VideoBIOS
00237
00238             pop   ds
00239             pop   bp
00240             ret
00241
00242 bell        endp
00243
00244 ;             subtitle    'Graphic Input Cursor'
00245 ;             page
00246 ;
00247 ; ***********************
00248 ; *                     *
00249 ; * Subroutine GINCRSIN *
00250 ; *                     *
00251 ; ***********************
00252 ;
00253 ginCrsIn    proc far
00254
00255             push  bp
00256             mov   bp,sp             ; lokale Basis
00257             push  ds
00258             push  es
```

```
00259
00260             mov   ax, 00h           ; FN : Reset Mouse
00261             int   MOUSE
00262             push  bx                ; Freimachen Indexregister
00263             lds   bx, iAvail        ; Adresse iAvail nach BX laden
00264             mov   [bx],ax           ; Wert AX nach iAvail
00265             lds   bx, iButton       ; Adresse iButton nach BX laden
00266             pop   ax
00267             mov   [bx],ax           ; Wert AX nach iButton
00268
00269             mov   ax, 07h           ; FN : Setzen iXmin und iXmax
00270             lds   bx, iX0
00271             mov   cx,[bx]
00272             lds   bx, iX1
00273             mov   dx,[bx]
00274             int   MOUSE
00275
00276             mov   ax, 08h           ; FN : Setzen iYmin und iYmax
00277             lds   bx, iY0
00278             mov   cx,[bx]
00279             lds   bx, iY1
00280             mov   dx,[bx]
00281             int   MOUSE
00282
00283             mov   ax, 09h           ; FN : Definition Cursorform
00284             mov   bx, CrsDefHotX
00285             mov   cx, CrsDefHotY
00286             mov   dx, seg CrsDef    ; Mousedriver: Adressangabe über ES!
00287             mov   es, dx
00288             mov   dx, offset CrsDef
00289             int   MOUSE
00290
00291             pop   es
00292             pop   ds
00293             pop   bp
00294             ret   24                ; Parameteranzahl * 4 Bytes freigeben
00295 gincrsIn    endp
00296 ;
00297 ; ***********************
00298 ; *                     *
00299 ; * Subroutine GINCRSEX *
00300 ; *                     *
00301 ; ***********************
00302 ;
00303 ginCrsEx    proc far
00304
00305             push  bp
00306             mov   bp,sp             ; lokale Basis
00307             push  ds
00308
00309             mov   ax, 00h           ; FN : Reset Mouse
00310             int   MOUSE
00311
00312             pop   ds
00313             pop   bp
00314             ret   0                 ; Parameteranzahl * 4 Bytes freigeben
00315 gincrsEx    endp
00316 ;
00317 ; *********************
00318 ; *                   *
00319 ; * Subroutine GINCRS *
00320 ; *                   *
00321 ; *********************
00322 ;
00323 gincrs      proc far
00324
00325             push  bp
00326             mov   bp,sp             ; lokale Basis
00327             push  ds
00328
00329             mov   ax, 01h           ; FN : Show Cursor
00330             int   MOUSE
00331
00332 WaitUp:     mov   ax, 03h           ; FN: Get Button Status
00333             int   MOUSE
00334             test  bx,bx             ; Taste noch gedrueckt?
00335             jnz   WaitUp            ; noch vom letzten mal -> Warte
00336
00337 KeyLoop:    mov   ax, 03h           ; FN : Get Button Status
00338             int   MOUSE             ; MouseDriver-Call
00339             test  bx,bx             ; Bit0 linke, Bit 1 rechte Maustaste
00340             jnz   ExitKeyLp         ; Taste gedrückt -> fertig
00341
00342             mov   ah,06h            ; DOS 6: Zeichen ohne Warten einlesen
00343             mov   dl,0ffh
00344             int   DOS
00345             jz    KeyLoop           ; keine Keyboardtaste gedrückt -> weiter
```

```
00346
00347             mov   ah,0h
00348             push  ax                ; Terminator
00349             mov   ax, 03h           ; FN : Get Mouse Koordinaten
00350             int   MOUSE
00351             pop   bx                ; Terminator ASCII
00352
00353 ExitKeyLp:  push  bx                ; Terminator
00354             lds   bx, iX            ; Adresse iX nach BX laden
00355             mov   [bx],cx           ; CX: horizontale Mauskoordinate
00356             lds   bx, iY            ; Adresse iY nach BX laden
00357             mov   [bx],dx           ; DX: vertikale Mauskoordinate
00358             pop   ax                ; Terminator
00359             lds   bx, iC            ; Adresse iC nach BX laden
00360             mov   [bx],ax           ; Übergabe in iC
00361
00362
00363             mov   ax, 02h           ; FN : Hide Cursor
00364             int   MOUSE
00365
00366             pop   ds
00367             pop   bp
00368             ret   12                ; Parameteranzahl * 4 Bytes freigeben
00369 gincrs      endp
00370
00371 ;            subtitle    'Get Enviroment'
00372 ;            page
00373 ;
00374 ; *********************
00375 ; *                   *
00376 ; * Subroutine GETENV *
00377 ; *                   *
00378 ; *********************
00379 ;
00380 GetEnv      proc far
00381
00382             push  bp
00383             mov   bp,sp             ; lokale Basis
00384             push  ds
00385             push  es
00386             push  di
00387             push  si
00388             pushf                   ; Rette Direction Flag!
00389
00390             cld                     ; Stringsuche aufwärts
00391 ;
00392 ; Bestimmung Stringlänge Suchstring
00393 ;
00394             mov   cx, 0             ; Counter
00395             lds   si, CharBuf       ; Buffer = Suchstring
00396 LenLoop:    mov   al,byte ptr ds:[si]; nächstes Zeichen
00397             or    al,al             ; Char(0) = Ende?
00398             jz    LenDone           ; ja
00399             inc   cx
00400             inc   si
00401             jmp   LenLoop
00402
00403 LenDone:    push  cx                ; Länge des Suchstrings
00404 ;
00405 ; Get Enviroment
00406 ;
00407             mov   ah, 62h           ; DOS 62h: Get PSP
00408             int   DOS
00409             mov   es,bx             ; ES:00 jetzt auf PSP
00410             mov   bx,es:[2ch]       ; PSP Element 2c: Enviroment
00411             mov   es, bx
00412             xor   di,di             ; Jetzt: ES:DI auf 1. Eintrag Enviroment
00413
00414 SearchLoop: lds   si, CharBuf       ; Suchstring in DS:AX
00415             pop   cx                ; Länge Suchstring
00416             push  cx
00417             repe  cmpsb             ; vergleichen mit Enviroment
00418             jz    Found
00419             xor   al,al             ; Ende Enviromenteintrag suchen
00420             mov   cx,-1
00421             repnz scasb
00422             cmp   byte ptr es:[di],0; letzter Eintrag?
00423             jnz   SearchLoop
00424             jmp   NotFound
00425 ;
00426 ; Abspeichern in den Puffer
00427 ;
00428 NotFound:                           ; ES:DI auf Char(0)
00429 Found:                              ; ES:DI auf Inhalt Enviromentvariable
00430
00431             lds   bx, CharBufL      ; Parameter Bufferlänge
00432             mov   cx,[bx]           ; Counter = Bufferlänge
```

```
00433
00434             lds   si, CharBuf       ; Zieladresse
00435 StoreLoop:  mov   al,byte ptr es:[di]; nächstes Zeichen
00436             mov   byte ptr ds:[si],al; speichern
00437             or    al,al             ; Char(0) = Ende?
00438             jz    StoreDone         ; ja
00439             inc   di
00440             inc   si
00441             dec   cx
00442             jz    StoreDone         ; Bufferende erreicht
00443             jmp   StoreLoop
00444
00445 StoreDone:  pop   ax                ; Clear Stack, Suchstringlänge
00446
00447             popf                    ; Restore Status
00448             pop   si
00449             pop   di
00450             pop   es
00451             pop   ds
00452             pop   bp
00453             ret   8
00454
00455 GetEnv      endp
00456
00457 ;            subtitle    'Byte Files'
00458 ;            page
00459 ;
00460 ; ***********************
00461 ; *                     *
00462 ; * Function OpenBytFil *
00463 ; *                     *
00464 ; ***********************
00465 ;
00466 OpenBytFil  proc far
00467
00468             push  bp
00469             mov   bp,sp             ; lokale Basis
00470             push  ds
00471
00472             lds   dx,FilNam
00473             xor   cx,cx             ; Löschen Attribut -> unbeschränkter Zugriff
00474             mov   ah,05bh           ; Open New File
00475             int   DOS
00476
00477             lds   bx, iHandleO      ; Adresse iButton nach BX laden
00478             mov   [bx],ax           ; FileHandle nach iHandle
00479
00480             lds   bx, iErrO
00481             jc    ErrO              ; kein Carryflag -> iErr=0: i.O.
00482             xor   ax,ax             ; iErr=3: path not found, =4 too many open files
00483 ErrO:       mov   [bx],ax           ; =5 access denied, =50h file exists
00484
00485             pop   ds
00486             pop   bp
00487             ret   12                ; 12 = 3 Parameter
00488
00489 OpenBytFil  endp
00490 ;
00491 ;
00492 ; ***********************
00493 ; *                     *
00494 ; * Function WrtBytFil *
00495 ; *                     *
00496 ; ***********************
00497 ;
00498
00499 WrtBytFil   proc far
00500
00501             push  bp
00502             mov   bp,sp             ; lokale Basis
00503             push  ds
00504
00505             lds   bx,iCount
00506             mov   cx,[bx]
00507             jcxz  NoWrt             ; keine Bytes zu schreiben
00508
00509             lds   bx,iHandle
00510             mov   bx,[bx]
00511
00512             lds   dx,Buf            ; letzter Befehl vor DOS-call, DS auf Buf!
00513
00514             mov   ah,040h           ; Write File
00515             int   DOS
00516
00517             lds   bx,iCount
00518             mov   cx,[bx]
00519             xor   dx,dx             ; Clear Error-Flag
```

```
00520             cmp   ax,cx             ; Count IST < Count SOLL?
00521             jnl   WrtIO
00522             mov   dx,0ffffh         ; SET Error-Flag
00523 WrtIO:      lds   bx,iErr           ; Store Error-Flag
00524             mov   [bx],dx
00525
00526 NoWrt:      pop   ds
00527             pop   bp
00528             ret   16                ; 16 = 4 Parameter
00529
00530 WrtBytFil   endp
00531 ;
00532 ; ************************
00533 ; *                      *
00534 ; * Function CloseBytFil *
00535 ; *                      *
00536 ; ************************
00537 ;
00538 CloseBytFil proc far
00539
00540             push  bp
00541             mov   bp,sp             ; lokale Basis
00542             push  ds
00543
00544             lds   bx,iHandleC
00545             mov   bx,[bx]
00546             mov   ah,03eh           ; Close File
00547             int   DOS
00548
00549             pop   ds
00550             pop   bp
00551             ret   4                 ; 4 = 1 Parameter
00552
00553 CloseBytFil endp
00554
00555 ;             subtitle    'lib$MoveC3'
00556 ;             page
00557 ;
00558 ; ************************
00559 ; *                      *
00560 ; * Subroutine lib_MovC3 *
00561 ; *                      *
00562 ; ************************
00563 ;
00564 lib_movc3_  proc far
00565
00566             push  bp
00567             mov   bp,sp             ; lokale Basis
00568             push  ds
00569             push  es
00570             push  di
00571             push  si
00572             pushf                   ; Rette Direction Flag!
00573
00574 ;
00575 ; Kopieren des Strings
00576 ;
00577
00578             lds   bx,iByte
00579             mov   cx,[bx]           ; Counter
00580             lds   si, Source        ; Buffer = Suchstring
00581             les   di, Dest
00582
00583             cld                     ; aufwärts
00584             cmp   di,si
00585             jb    domove
00586
00587             add   di,cx
00588             dec   di
00589             add   si,cx
00590             dec   si
00591             std                     ; abwärts
00592
00593 domove:     rep   movsb
00594
00595             popf                    ; Restore Status
00596             pop   si
00597             pop   di
00598             pop   es
00599             pop   ds
00600             pop   bp
00601             ret   12
00602
00603 lib_movc3_  endp
00604
00605 TcsdDosA_text ends
00606
```

```
00607             end
00608 ;
00609 ; //! \endcond
00610
```

3.36 TCSdDosa.fi File Reference

DOS Port: FORTRAN-Interface TCSdDOSa.asm.

3.36.1 Detailed Description

DOS Port: FORTRAN-Interface TCSdDOSa.asm.

Interface definitions for the Watcom Fortran Compiler

Author

Dr.-Ing. Klaus Friedewald

Version

1.32

Date

06.02.2003

Note

Assemblerroutines are written according to the Microsoft Procedure Call Standard.

Watcom-FTN77 variable names are allowed to be 32 characters long and may contain $ and _. That for $notruncate und $notstrict are superfluous.

Hexadecimal numbers are represented by 'ff'x instead of #ff.

Definition in file TCSdDosa.fi.

3.37 TCSdDosa.fi

```
00001 C> \file    TCSdDosa.fi
00002 C> \brief   DOS Port: FORTRAN-Interface TCSdDOSa.asm
00003 C>
00004 C> \~german
00005 C> Interfacedeklarationen fuer den Watcom Fortran-Compiler
00006 C> \~english
00007 C> Interface definitions for the Watcom Fortran Compiler
00008 C> \~
00009 C> \author  Dr.-Ing. Klaus Friedewald
00010 C> \version 1.32
00011 C> \date 06.02.2003
00012 C> \~german
00013 C> \note
00014 C> Assemblerroutinen entsprechend Microsoft Procedure Call Standard
00015 C>
00016 C> \note
00017 C> Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
00018 C> von $ und _. Deswegen $notruncate und $notstrict ueberfluessig.
00019 C>
00020 C> \note
00021 C> Hex-Zahlen werden nicht durch \#ff sondern durch \'ff\'x dargestellt
00022 C> \~english
00023 C> \note
00024 C> Assemblerroutines are written according to the Microsoft Procedure Call Standard.
00025 C>
00026 C> \note
00027 C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
00028 C> contain $ and _. That for $notruncate und $notstrict are superfluous.
00029 C>
00030 C> \note
00031 C> Hexadecimal numbers are represented by 'ff'x instead of \#ff.
00032 C> \~
00033 C>
00034 C
00035 C Interfacedeklarationen fuer den Watcom Fortran-Compiler
00036 C Assemblerroutinen entsprechend Microsoft Procedure Call Standard
00037 C
00038 C
00039 C   ktinput:    Tastaturabfrage [AX] dos7h
00040 C   bell:       Signalton [ax,bx] video bios tty out
00041 C   GinCrsIn:   Initialisierung Graphikmaus [ax,bx,cx,dx] int mouse
00042 C   GinCrsEX:   Wiederherstellen Graphikmaus [ax] int mouse
00043 C   GinCrs:     Abfrage Graphikmaus [ax,bx,cx,dx] int mouse
00044 C
00045 C   GetEnv:     Abfrage Enviroment (C-Characterformat!)[ax,bx,cx,dx] int dos
00046 C
00047 C   Lib_movC3_: Kopieren eines Feldes [ax,bx,cx]
00048 C
00049 C   OpenBytFil [ax,bx,cd,dx] dos
00050 C   WrtBytFil [ax,bx,cd,dx] dos
00051 C   CloseBytFil [ax,bx]
00052 C   i.O.: kTinput, bell
00053 C
00054 C \cond
00055
00056 c$pragma aux kTinput value [ax] modify exact [ax]
00057
00058 c$pragma aux bell parm [] modify exact [ax bx]
00059
00060 c$pragma aux GetEnv parm reverse (DATA_REFERENCE FAR, REFERENCE FAR) []\
00061 c  modify exact [ax bx cx dx]
00062
00063 c$pragma aux GinCrsIn parm reverse (REFERENCE FAR, reference far, \
00064 c  reference far) [] modify exact [ax bx cx dx]
00065
00066 c$pragma aux GinCrs parm reverse (REFERENCE FAR) [] \
00067 c  modify exact [ax bx cx dx]
00068
00069 c$pragma aux GinCrsEx modify exact [ax]
00070
00071 c$pragma aux lib_movC3_ parm reverse (REFERENCE FAR, DATA_REFERENCE FAR, \
00072 c  DATA_REFERENCE FAR) [] modify exact [ax bx cx]
00073
00074 c$pragma aux OpenBytFil parm reverse (REFERENCE FAR, REFERENCE FAR, \
00075 c  DATA_REFERENCE FAR) [] modify exact [ax bx cx dx]
00076
00077 c$pragma aux WrtBytFil parm reverse (REFERENCE FAR, REFERENCE FAR, \
00078 c  DATA_REFERENCE FAR, REFERENCE FAR) [] modify exact [ax bx cx dx]
00079
00080 c$pragma aux CloseBytFil parm reverse (REFERENCE FAR) [] modify exact [ax bx]
00081 C
00082 C \endcond
```

3.38 TCSdrDOS.for File Reference

DOS Port: High-Level Driver.

Functions/Subroutines

- subroutine tcslev (LEVEL)
- subroutine initt (iDummy)
- subroutine initt1
- subroutine italic
- subroutine graphicerrorinit
- subroutine lincol (iCol)
- subroutine txtcol (iCol)
- subroutine bckcol (iCol)
- subroutine defaultcolour
- integer function icolcode (iCol)
- integer function iscreenxcoord (iX)
- integer function iscreenycoord (iY)
- integer function irevscreenxcoord (iX)
- integer function irevscreenycoord (iY)
- subroutine erase
- subroutine finitt
- subroutine svstat (Array)
- subroutine restat (Array)
- subroutine movabs (ix, iy)
- subroutine pntabs (ix, iy)
- subroutine drwabs (ix, iy)
- subroutine dshabs (ix, iy, iMask)
- subroutine movrel (iX, iY)
- subroutine pntrel (iX, iY)
- subroutine drwrel (iX, iY)
- subroutine dshrel (iX, iY, iMask)
- subroutine seeloc (IX, IY)
- subroutine swind1 (ix1, iy1, ix2, iy2)
- subroutine alpha
- subroutine csize (Ixlen, iylen)
- subroutine toutpt (iChr)
- subroutine toutst (nChr, iChrArr)
- subroutine toutstc (String)
- subroutine statst (String)
- subroutine tinput (iChr)
- subroutine dcursr (IC, IX, IY)
- subroutine lib_movc3 (iLen, sou, dst)
- subroutine anmode

  Entry Dummyroutinen.
- logical function winselect (iDummy)

3.38.1 Detailed Description

DOS Port: High-Level Driver.

Version

(2005, 45,2)

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Note

```
Extensions of the Tektronix TCS:
 subroutine TOUTSTC (String): Output Fortran-String
 subroutine LINCOL (iCol): Set line color (iCol=0..15)
 subroutine TXTCOL (iCol): Set text color
 subroutine BCKCOL (iCol): Set background color (visible after ERASE)
 subroutine DefaultColour: Reset default colors
```

Definition in file TCSdrDOS.for.

3.38.2 Function/Subroutine Documentation

3.38.2.1 alpha()

```
subroutine alpha
```

Definition at line 686 of file TCSdrDOS.for.

3.38.2.2 anmode()

```
subroutine anmode
```

Entry Dummyroutinen.

AlfMod

pClipt

ioWait

Definition at line 800 of file TCSdrDOS.for.

3.38.2.3 bckcol()

```
subroutine bckcol (
             integer iCol )
```

Definition at line 427 of file TCSdrDOS.for.

3.38.2.4 csize()

```
subroutine csize (
              Ixlen,
              iylen )
```

Definition at line 698 of file TCSdrDOS.for.

3.38.2.5 dcursr()

```
subroutine dcursr (
             integer IC,
             integer IX,
             integer IY )
```

Definition at line 767 of file TCSdrDOS.for.

3.38.2.6 defaultcolour()

```
subroutine defaultcolour
```

Definition at line 436 of file TCSdrDOS.for.

3.38.2.7 drwabs()

```
subroutine drwabs (
              ix,
              iy )
```

Definition at line 587 of file TCSdrDOS.for.

3.38.2.8 drwrel()

```
subroutine drwrel (
              iX,
              iY )
```

Definition at line 645 of file TCSdrDOS.for.

3.38.2.9 dshabs()

```
subroutine dshabs (
              ix,
              iy,
              iMask )
```

Definition at line 599 of file TCSdrDOS.for.

3.38.2.10 dshrel()

```
subroutine dshrel (
              iX,
              iY,
              iMask )
```

Definition at line 655 of file TCSdrDOS.for.

3.38.2.11 erase()

```
subroutine erase
```

Definition at line 500 of file TCSdrDOS.for.

3.38.2.12 finitt()

```
subroutine finitt
```

Definition at line 513 of file TCSdrDOS.for.

3.38.2.13 graphicerrorinit()

```
subroutine graphicerrorinit
```

Definition at line 254 of file TCSdrDOS.for.

3.38.2.14 icolcode()

```
integer function icolcode (
              iCol )
```

Definition at line 444 of file TCSdrDOS.for.

3.38.2.15 initt()

```
subroutine initt (
              iDummy )
```

Definition at line 121 of file TCSdrDOS.for.

3.38.2.16 initt1()

```
subroutine initt1
```

Definition at line 135 of file TCSdrDOS.for.

3.38.2.17 irevscreenxcoord()

```
integer function irevscreenxcoord (
              iX )
```

Definition at line 484 of file TCSdrDOS.for.

3.38.2.18 irevscreenycoord()

```
integer function irevscreenycoord (
              iY )
```

Definition at line 492 of file TCSdrDOS.for.

3.38.2.19 iscreenxcoord()

```
integer function iscreenxcoord (
              iX )
```

Definition at line 468 of file TCSdrDOS.for.

3.38.2.20 iscreenycoord()

```
integer function iscreenycoord (
              iY )
```

Definition at line 476 of file TCSdrDOS.for.

3.38.2.21 italic()

```
subroutine italic
```

Definition at line 219 of file TCSdrDOS.for.

3.38.2.22 lib_movc3()

```
subroutine lib_movc3 (
             integer iLen,
             character *(*) sou,
             character *(*) dst )
```

Definition at line 790 of file TCSdrDOS.for.

3.38.2.23 lincol()

```
subroutine lincol (
             integer iCol )
```

Definition at line 406 of file TCSdrDOS.for.

3.38.2.24 movabs()

```
subroutine movabs (
              ix,
              iy )
```

Definition at line 557 of file TCSdrDOS.for.

3.38.2.25 movrel()

```
subroutine movrel (
              iX,
              iY )
```

Definition at line 625 of file TCSdrDOS.for.

3.38.2.26 pntabs()

```
subroutine pntabs (
              ix,
              iy )
```

Definition at line 570 of file TCSdrDOS.for.

3.38.2.27 pntrel()

```
subroutine pntrel (
              iX,
              iY )
```

Definition at line 635 of file TCSdrDOS.for.

3.38.2.28 restat()

```
subroutine restat (
            integer, dimension(1) Array )
```

Definition at line 541 of file TCSdrDOS.for.

3.38.2.29 seeloc()

```
subroutine seeloc (
              IX,
              IY )
```

Definition at line 667 of file TCSdrDOS.for.

3.38.2.30 statst()

```
subroutine statst (
             character *(*) String )
```

Definition at line 744 of file TCSdrDOS.for.

3.38.2.31 svstat()

```
subroutine svstat (
             integer, dimension(1) Array )
```

Definition at line 529 of file TCSdrDOS.for.

3.38.2.32 swind1()

```
subroutine swind1 (
              ix1,
              iy1,
              ix2,
              iy2 )
```

Definition at line 676 of file TCSdrDOS.for.

3.38.2.33 tcslev()

```
subroutine tcslev (
             integer, dimension(3) LEVEL )
```

Definition at line 104 of file TCSdrDOS.for.

3.38.2.34 tinput()

```
subroutine tinput (
              iChr )
```

Definition at line 760 of file TCSdrDOS.for.

3.38.2.35 toutpt()

```
subroutine toutpt (
              iChr )
```

Definition at line 707 of file TCSdrDOS.for.

3.38.2.36 toutst()

```
subroutine toutst (
              nChr,
             integer, dimension (1) iChrArr )
```

Definition at line 725 of file TCSdrDOS.for.

3.38.2.37 toutstc()

```
subroutine toutstc (
             character *(*) String )
```

Definition at line 735 of file TCSdrDOS.for.

3.38.2.38 txtcol()

```
subroutine txtcol (
             integer iCol )
```

Definition at line 418 of file TCSdrDOS.for.

3.38.2.39 winselect()

```
logical function winselect (
              iDummy )
```

Definition at line 812 of file TCSdrDOS.for.

3.39 TCSdrDOS.for

```
00001 C> \file       TCSdrDOS.for
00002 C> \brief      DOS Port: High-Level Driver
00003 C> \version    (2005, 45,2)
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C> \note \verbatim
00009 C>    Erweiterungen gegenüber Tektronix:
00010 C>     subroutine TOUTSTC (String): Ausgabe Fortran-String
00011 C>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
00012 C>     subroutine TXTCOL (iCol): Setzen Textfarbe
00013 C>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00014 C>     subroutine DefaultColour: Wiederherstellung Defaultfarben
00015 C> \endverbatim
00016 C> \~english
00017 C> \note \verbatim
00018 C>    Extensions of the Tektronix TCS:
00019 C>     subroutine TOUTSTC (String): Output Fortran-String
00020 C>     subroutine LINCOL (iCol): Set line color (iCol=0..15)
00021 C>     subroutine TXTCOL (iCol): Set text color
00022 C>     subroutine BCKCOL (iCol): Set background color (visible after ERASE)
00023 C>     subroutine DefaultColour: Reset default colors
00024 C> \endverbatim
00025 C> \~
00026 C>
00027 C
00028 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00029 C
00030 C     07.02.02 Version 1.2:
00031 C           Implementierung multilinguale Fehlermeldungen
00032 C
00033 C     24.05.02 Version 1.3:
00034 C           Umgebungsvariablen werten auch mit ";" getrennte Pfade aus
00035 C           INCLUDE Interface TCSDOSA.FI zur Anpassung an den WATCOM-Compiler
00036 C           TKTRNX.for: geänderte Common-Blocklänge aufgrund INTEGER*4
00037 C                       bei WATCOM (MS: INTEGER*2)
00038 C           DSHABS:     Ersatz Hex-Konstante durch Dezimalkonstante zur
00039 C                       Erzielung Kompatibilität mit WATCOM-Compiler
00040 C           INITT1:     Anpassung WATCOM-Compiler:
00041 C                       - Apostrophe innerhalb von Strings durch 2 Apostrophe
00042 C                       - Strings muessen mit CHAR(0) abgeschlossen werden
00043 C                       BugFix: SETVIEWPORT erwartet INTEGER*2 statt REAL*4!
00044 C           TOUTPT:     Anpassung WATCOM: auszugebender Character mit CHAR (0)
00045 C           GraphicError: Format 900 ist bei den *.lng-Files streng zu be-
00046 C                       folgen, d.h. "_12,Text" . Ausgabe OUTTEXT mit char(0).
00047 C
00048 C     23.07.02 Version 1.31:
00049 C           Erweiterung: subroutine WINLBL (GraphicWinLbl, StatusWindLbl):
00050 C                       Kompatibilität zu Windowsversion.
00051 C           Eigenes Modul OUTTEXT zur Kompatibilität MS-WATCOM
00052 C                       (Watcom benötigt char(0), dann MS Zeilenüberlauf)
00053 C
00054 C     24.08.02 Version 1.32:
00055 C           ITALIC, ITALIR, DBLSIZ, NRMSIZ: Schriftarten Graphikausgabe.
00056 C                       Kombination groß/kursiv nicht vorgesehen.
00057 C                       Implementiert durch Fontfile GraphLib.FON
00058 C                       (Quelle: Programm SOFTY und Arial Terminal TTF-Basis)
00059 C           DSHABS:     Standardisierung Dash-Linestyles DOS-Windows:
00060 C                       0: solid, 1: dotted, 2: dash-dotted, 3:dashed
00061 C           DEFAULTCOLOUR: Bugfix Namensgebung, nicht DEFAULTCOLOURS
00062 C
00063 C     10.10.02 Version 1.33:
00064 C           INITT:      Zur Vereinheitlichung DOS/Windows jetzt in diesem File
00065 C           TCSLEV:     neu, zur Angleichung mit AG2LEV und Systemerkennung
00066 C
00067 C     19.10.02 Version 1.34 bzw. (2002,292,2)
00068 C           Umbenennung TKTRNX.FOR in TKTRNX.FD zur Kompatibilität CP/M
00069 C
00070 C     25.10.02 Version (2002,298,2)
00071 C           Entprellen Mousetaste bei GIN-Cursoreingabe
```

```
00072 C
00073 C     06.02.03 Version (2003, 37,2)
00074 C           Vereinheitlichtes Interface lib$movc3 (Kompatibilitaet Windows)
00075 C
00076 C     12.01.04 Version (2004, 12,2)
00077 C           INITT1:     Bugfix Endlosschleife bei fehlerhaftes Fontfile und
00078 C                       Severity 5
00079 C           GRAPHICERRORINIT: Defaultseverity 10 bei EXIT (FINITT, iErr=12)
00080 C           Anmerkung: Die Subroutine GRAPHICERROR ruft sich bei Programm-
00081 C                 abbruch über FINITT implizit selber rekursiv auf (nicht
00082 C                 FORTRAN-konform!). Da jedoch keine lokalen Variablen ver-
00083 C                 wendet werden, ist dies in der Regeln nicht kritisch.
00084 C
00085 C     25.10.04 Version (2004,299,2)
00086 C           WINLBL:     Wertet jetzt den 3. Parameter (Initilisierungsfile)
00087 C                       analog zur Windowsversion aus (einschliesslich Ueber-
00088 C                       setzung '%:' und '.%'
00089 C           LIB$MOVC3:  Umbenannt in LIB_MOVC3. Alte Assemblerroutine heisst
00090 C                       jetzt LIB_MOVC3_.
00091 C
00092 C     15.02.05 Version (2005, 45,2)
00093 C           GRAPHICERROR: Bugfix ErrSeverity=0 entspricht jetzt NO ACTION.
00094 C
00095
00096       include 'FGRAPH.FI'
00097       include 'TCSdDOSa.FI'
00098
00099
00100
00101 C
00102 C  Ausgabe der Softwareversion
00103 C
00104       subroutine tcslev(LEVEL)
00105       integer LEVEL(3)
00106       level(1)=2005     ! Aenderungsjahr
00107       level(2)=  45     ! Aenderungstag
00108       level(3)=   2     ! System= DOS
00109
00110       return
00111       end
00112
00113
00114
00115 C
00116 C  Bildschirm Verwaltung
00117 C
00118
00119
00120
00121       subroutine initt (iDummy)
00122       call lintrn
00123       call swindo (0,1023,0,780)
00124       call vwindo (0.,1023.,0.,780.)
00125       call rrotat (0.)
00126       call rscale (1.)
00127       call setmrg (0,1023)
00128       call initt1
00129       call home
00130       return
00131       end
00132
00133
00134
00135       subroutine initt1
00136       include 'FGRAPH.FD'
00137       include 'TKTRNX.FD'
00138       integer*2 iErr, iAvail, iButton, kScrX2, kScrY2
00139       integer iLen, iTrimLen, iParse
00140
00141       character*80 cBuf, cBuf1*80
00142       record /videoconfig/ myscreen
00143       record /fontinfo/ myfont
00144
00145       character *13 cFontFile           ! Graphikfontfile
00146       parameter(cfontfile='GRAPHLIB.FON'//char(0))
00147
00148       character*5 cEnv                  ! Logischer Name für den Fontfilepfad
00149       parameter(cenv='LIB='//char(0))
00150
00151       call graphicerrorinit
00152
00153       ierr= setvideomode($maxresmode)
00154
00155       if (ierr .eq. 0) then
00156        call graphicerror (2,' ') ! TCS-Initt: unknown graphic adapter
00157       end if
00158
```

```
00159       call getvideoconfig (myscreen)
00160       kscrx= myscreen.numxpixels-1
00161       kscry= myscreen.numypixels-1-
00162      1    (myscreen.numypixels/myscreen.numtextrows)      ! Höhe Statuszeile
00163
00164       call setviewport (0,0, kscrx, kscry)
00165
00166       call settextwindow (myscreen.numtextrows,1,myscreen.numtextrows,
00167      1  myscreen.numtextcols)                              ! Statuszeile
00168       kstcol= myscreen.numtextcols - 1 ! Verhindere Scrollen durch -1
00169
00170       if (registerfonts(cfontfile).lt.0) then
00171        cbuf= cenv                              ! Abfrage Enviroment
00172        call getenv (cbuf, len(cbuf))
00173        ilenpath= itrimlen(cbuf)
00174        iparse=1
00175    10  continue ! while
00176         if (iparse.le.ilenpath) then
00177          ilen= index(cbuf(iparse:ilenpath), ';')-1
00178          if (ilen.le.0) ilen=ilenpath-iparse+1
00179         else
00180          ilen= -1
00181         end if
00182         if ((ilen.lt.1).or.(iparse.gt.ilenpath)) then
00183          cbuf1= cenv    ! Notwendig zur Bildung des Substrings aus PARAMETER
00184          cbuf1=cbuf1(1:istringlen(cbuf1))//':'//cfontfile
00185          call graphicerror (3,cbuf1(1:istringlen(cbuf1))) !openerror fontfile
00186          goto 15 ! ENDWHILE falls Errorseverity(3) < 10 (STOP)
00187         else
00188          cbuf1= cbuf(iparse:iparse+ilen-1)//'\'//cfontfile ! Chr0 in cFontFile
00189          call substitute (cbuf1,cbuf1, '\\', '\') ! kein doppelter Backslash!
00190         end if
00191        if (registerfonts(cbuf1(1:istringlen(cbuf1))).lt.0) then ! end while
00192         if (ilen.lt.ilenpath) then
00193          iparse= iparse+ilen+1
00194          goto 10        ! nächster Eintrag im Pfad
00195         else
00196          call graphicerror (3,cbuf1(1:istringlen(cbuf1)))
00197         end if
00198    15  end if
00199       end if
00200
00201       call nrmsiz               ! Standardschrift: normalgroß, nicht kursiv
00202
00203       kscrx2= kscrx             ! Konvertierung in int*2 durch WATCOM-Compiler
00204       kscry2= kscry
00205       call gincrsin (iavail, ibutton, 0, kscrx2, 0, kscry2)
00206       if (iavail.eq.-1) then
00207        imouse= ibutton
00208       else
00209        imouse= 0
00210       end if
00211       call defaultcolour
00212       call erase
00213
00214       return
00215       end
00216
00217
00218
00219       subroutine italic
00220 C
00221 C Verändern des Graphik-Fonts
00222 C
00223       include 'FGRAPH.FD'
00224       include 'TKTRNX.FD'
00225       integer*2 iErr
00226       record /fontinfo/ myfont
00227
00228       ierr= setfont('t''Italic'"//char(0))
00229       goto 10
00230
00231       entry dblsiz
00232       ierr= setfont('t''Double'"//char(0))
00233       goto 10
00234
00235       entry italir
00236       entry nrmsiz
00237       ierr= setfont('t''Normal'"//char(0))
00238
00239   10  continue          ! identischer Code für ITALIC und ITALIR
00240       if (ierr.lt.0) then
00241        call graphicerror (4,'Normal/Italic/Double') ! TCS-Initt: unknown font
00242       end if
00243       ierr= getfontinfo(myfont)
00244       khorsz= isign(irevscreenxcoord(int(myfont.pixwidth))
00245      1  - irevscreenxcoord(0),1)
```

```
00246       kversz= isign(irevscreenycoord(int(myfont.pixheight))
00247      1  - irevscreenycoord(0),1)
00248       khomey= 780-(1.1*kversz)
00249       return
00250       end
00251
00252
00253
00254       subroutine graphicerrorinit
00255 C
00256 C  SUBROUTINE GraphicErrorInit, ENTRIES WinLbl, GraphicError
00257 C  Internationalisierung der Fehlermeldungen
00258 C
00259       implicit none
00260       include 'FGRAPH.FD'
00261       save errseverity, errmsg, filnam
00262
00263       integer MaxErr
00264       parameter(maxerr=12)
00265       character *(*) Mssg
00266       character *(*) WinLblDummy, StatLblDummy, MessageFile
00267       integer iErr, i, iTrimLen,iStringLen, iErrSev
00268       integer iLenPath, iParse, iLen
00269
00270       character*132 cEnv, FilNam, cBuf
00271       integer ErrSeverity (MaxErr)
00272       character*80 ErrMsg (MaxErr)
00273       data cenv,filnam /'LIB=','GRAPHLIB.LNG'/
00274       data errmsg/'GRAPHLIB %%% INITT: Incompatible message file - Press
00275      1 any key',
00276      2      'GRAPHLIB %%% INIT: Unknown graphic adapter',
00277      3      'GRAPHLIB %%% INIT: Error opening fontfile $$',
00278      4      'GRAPHLIB %%% INIT: Unknown font $$',
00279      5      'GRAPHLIB %%% INPUT: No mousedriver available, use keyboard'
00280      6     ,'GRAPHLIB %%% HARDCOPY: Error during OPEN',
00281      7      'GRAPHLIB %%% HARDCOPY: Error during WRITE',
00282      8      'GRAPHLIB %%% HARDCOPY: Internal error (buffer overflow)',
00283      9      '$$','Hardcopy in progress','Press any key to continue',
00284      2      'Press any key to exit program'/
00285
00286       data errseverity /5,10,10,10, 1, 5, 5, 5, 1, 1, 5, 10/
00287
00288       external iGetArg           ! Watcom Library-Funktion
00289       integer iGetArg
00290
00291       cenv=cenv(1:itrimlen(cenv))//char(0)
00292       filnam= filnam(1:itrimlen(filnam))//char(0)
00293
00294 C
00295 C  1.Priorität: Message-File durch WinLbl spezifiziert
00296 C  2.Priorotät: GRAPHLIB.LNG im Arbeitsdirectory
00297 C
00298
00299       open (unit=9,form='FORMATTED', err=5, status='OLD', file=
00300      1                          filnam(1:istringlen(filnam)))
00301       goto 7      ! File gefunden -> Einlesen
00302
00303 C
00304 C  3.Priorität: Message-File GRAPHLIB.LNG in LIB:
00305 C
00306
00307 5     call getenv (cenv, len(cenv))
00308       ilenpath= itrimlen(cenv)
00309       iparse=1
00310    10 continue ! while
00311        if (iparse.le.ilenpath) then
00312         ilen= index(cenv(iparse:ilenpath), ';')-1
00313         if (ilen.le.0) ilen=ilenpath-iparse+1
00314        else
00315         goto 99         ! benutze Default
00316        end if
00317        if ((ilen.ge.1).and.(iparse.le.ilenpath)) then
00318         cbuf= cenv(iparse:iparse+ilen-1)//'\'//filnam ! Chr0 bereits in FilNam
00319         call substitute (cbuf,cbuf, '\\', '\') ! kein doppelter Backslash !
00320        end if
00321        open (unit=9,form='FORMATTED', err=6, status='OLD', file=
00322      1                          cbuf(1:istringlen(cbuf)))
00323       goto 7      ! File gefunden -> Einlesen
00324 6     if (ilen.lt.ilenpath) then ! end while
00325        iparse= iparse+ilen+1
00326        goto 10  ! nächster Eintrag im Pfad
00327       else
00328        goto 99 ! kein File vorhanden - > benutze Default
00329       end if
00330
00331 7     do 20 i=1,maxerr
00332        read (unit=9, err=90, fmt=900) errseverity(i),errmsg(i)
```

```
00333 20    continue
00334
00335       close (unit=9)
00336
00337 99    return
00338 C
00339 C Ausgabe Fehlermeldung Messagefile
00340 C
00341 90    call outtext (errmsg(1)) ! Graphiksystem wurde noch nicht initialisiert!
00342       call tinput (i)
00343       return
00344
00345
00346
00347       entry winlbl(winlbldummy, statlbldummy, messagefile)
00348 C
00349 C  Setzen des Messagefiles und Uebersetzung '%:' bzw. '.%'
00350 C
00351       if (istringlen(messagefile).le.0) return
00352       filnam= messagefile
00353       i= igetarg(0, cbuf) ! Arg. 0: Programmname mit Directory
00354       if (i.gt.1) then
00355    30  continue ! repeat
00356         i= i-1
00357        if ((cbuf(i:i).ne.'\').and.(i.gt.1)) goto 30
00358        cbuf(i+1:i+1)= char(0)
00359        call substitute (filnam, filnam,'%:',cbuf)
00360       end if
00361       call substitute (filnam, filnam,'.%','.lng')
00362       return
00363
00364
00365
00366       entry graphicerror(ierr,mssg)
00367 C
00368 C  Ausgabe der Fehlermeldung
00369 C
00370       if (ierr.eq.99) then             ! Programmabbruch aus FINITT (2. Aufruf)
00371        if (errseverity(12).eq.10) then
00372         ierrsev= 99                                ! STOP
00373        else if (errseverity(12).eq.5) then
00374         ierrsev= 1                                 ! TINPUT bereits durchgeführt
00375        else
00376         ierrsev= errseverity(12)
00377        end if
00378       else
00379        ierrsev= errseverity(ierr)
00380        if (ierrsev.gt.0) then
00381         call bell
00382         call substitute (errmsg(ierr),cbuf, '$$', mssg)
00383         call statst (cbuf)
00384        end if
00385       end if
00386
00387       if (ierrsev.le.1) then               ! =1: Statusmeldung
00388        return
00389       else if (ierrsev.eq.99) then
00390        stop                                ! =99: aus FINITT
00391       else
00392        call tinput (i)
00393        if (ierrsev.eq.5) then              ! =5: Warnung
00394         return
00395        else if (ierrsev.eq.10) then        ! =10: Abbruch
00396         if (ierr.ne.12) call finitt ()     ! Rekursion iErr=12 verhindern
00397        end if
00398       end if
00399
00400       return
00401 900   format (1x,i2,1x,a)
00402       end
00403
00404
00405
00406       subroutine lincol (iCol)
00407       include 'FGRAPH.FD'
00408       include 'TKTRNX.FD'
00409       integer iColCode, iCol
00410       integer *2 iErr
00411       ilincol= icolcode(icol)
00412       ierr= setcolor(ilincol)
00413       return
00414       end
00415
00416
00417
00418       subroutine txtcol (iCol)
00419       include 'TKTRNX.FD'
```

```
00420       integer iColCode, iCol
00421       itxtcol= icolcode(icol)
00422       return
00423       end
00424
00425
00426
00427       subroutine bckcol (iCol)
00428       include 'TKTRNX.FD'
00429       integer iColCode, iCol
00430       ibckcol= icolcode(icol)
00431       return
00432       end
00433
00434
00435
00436       Subroutine defaultcolour
00437       call bckcol (0)
00438       call lincol (1)
00439       call txtcol (1)
00440       return
00441       end
00442
00443
00444       integer function icolcode (iCol)
00445       include 'FGRAPH.FD'
00446       integer icoltab (15)     ! Anpassung Farbindex an VGA-Palette
00447       data icoltab/  15       ,12         ,10         ,11         ,9
00448 C        iCol=       1        2           3           4           5
00449 C        entspricht: weiss    rot         gruen       blau        lila
00450      1               ,14      ,7          ,13         ,4          ,2
00451 C        iCol=       6        7           8           9           10
00452 C        entspricht: gelb     grau        violett     mattrot     mattgruen
00453      2               ,1       ,3          ,6          ,8          ,5/
00454 C        iCol=       11       12          13          14          15
00455 C        entspricht: mattblau mattlila    orange      mattgrau    mattviolett
00456       if (icol.le.0) then
00457        icolcode= 0
00458       else if (icol.gt.15) then
00459        icolcode= icoltab(1)
00460       else
00461        icolcode= icoltab(icol)
00462       end if
00463       return
00464       end
00465
00466
00467
00468       integer function iscreenxcoord (iX)
00469       include 'TKTRNX.FD'
00470       iscreenxcoord= (ix*kscrx)/1023
00471       return
00472       end
00473
00474
00475
00476       integer function iscreenycoord (iY)
00477       include 'TKTRNX.FD'
00478       iscreenycoord= kscry-(kscry*iy)/780
00479       return
00480       end
00481
00482
00483
00484       integer function irevscreenxcoord (iX)
00485       include 'TKTRNX.FD'
00486       irevscreenxcoord= (ix*1023)/kscrx
00487       return
00488       end
00489
00490
00491
00492       integer function irevscreenycoord (iY)
00493       include 'TKTRNX.FD'
00494       irevscreenycoord= 780-(780*iy)/kscry
00495       return
00496       end
00497
00498
00499
00500       subroutine erase
00501       include 'FGRAPH.FD'
00502       include 'TKTRNX.FD'
00503       call clearscreen ($gclearscreen)
00504       ierr= setcolor(ibckcol)
00505       ierr= rectangle( $gfillinterior, 0, 0, kscrx, kscry)
00506       ierr= setcolor(ilincol)
```

```
00507       call movabs (kbeamx, kbeamy)      ! Cursorposition wiederherstellen
00508       return
00509       end
00510
00511
00512
00513       subroutine finitt
00514       implicit none
00515       include 'FGRAPH.FD'
00516       integer*2 iErr
00517       call graphicerror (12,' ')        ! Press any key to exit program
00518       call unregisterfonts ()
00519       ierr= setvideomode($defaultmode)
00520       call gincrsex
00521       call graphicerror (99,' ')        ! Jetzt auch STOP möglich
00522       return
00523       end
00524
00525 C
00526 C  Abspeichern Terminal Status Area
00527 C
00528
00529       subroutine svstat (Array)
00530       integer array(1)
00531       include 'TKTRNX.FD'
00532       integer arr(1)
00533       equivalence(arr(1),khomey)
00534       do 10 i=1,itktrnxl
00535 10     array(i)= arr(i)
00536       return
00537       end
00538
00539
00540
00541       subroutine restat (Array)
00542       integer array(1)
00543       include 'TKTRNX.FD'
00544       integer arr(1)
00545       equivalence(arr(1),khomey)
00546       do 10 i=1,itktrnxl
00547 10     arr(i)= array(i)
00548       call movabs (kbeamx, kbeamy)
00549       return
00550       end
00551
00552
00553 C
00554 C  Absolute Zeichenbefehle
00555 C
00556
00557       subroutine movabs (ix,iy)
00558       include 'FGRAPH.FD'
00559       include 'TKTRNX.FD'
00560       record /xycoord/ oldxy
00561       integer iScreenXcoord, iScreenYcoord
00562       call moveto (iscreenxcoord(ix),iscreenycoord(iy), oldxy)
00563       kbeamx= ix
00564       kbeamy= iy
00565       return
00566       end
00567
00568
00569
00570       subroutine pntabs (ix,iy)
00571       include 'FGRAPH.FD'
00572       include 'TKTRNX.FD'
00573       integer iScreenXcoord, iScreenYcoord
00574       integer oldPixel,ixs,iys
00575       record /xycoord/ oldxy
00576       ixs= iscreenxcoord(ix)
00577       iys= iscreenycoord(iy)
00578       call moveto (ixs,iys, oldxy)
00579       oldpixel= setpixel(ixs,iys)
00580       kbeamx= ix
00581       kbeamy= iy
00582       return
00583       end
00584
00585
00586
00587       subroutine drwabs (ix,iy)
00588       include 'FGRAPH.FD'
00589       include 'TKTRNX.FD'
00590       integer iScreenXcoord, iScreenYcoord
00591       ierr= lineto(iscreenxcoord(ix), iscreenycoord(iy))
00592       kbeamx= ix
00593       kbeamy= iy
```

```
00594       return
00595       end
00596
00597
00598
00599       subroutine dshabs (ix,iy, iMask)
00600       include 'FGRAPH.FD'
00601       include 'TKTRNX.FD'
00602       integer iScreenXcoord, iScreenYcoord
00603       integer*2 iErr
00604       if (imask.eq.0) then      ! solid line
00605        imask= 65535             ! 1111 1111 1111 1111
00606       else if (imask.eq.1) then ! dotted line
00607        imask= 43690             ! 1010 1010 1010 1010
00608       else if (imask.eq.2) then ! dash-dotted line
00609        imask= 58596             ! 1110 0100 1110 0100
00610       else if (imask.eq.3) then ! dashed line
00611        imask= 61680             ! 1111 0000 1111 0000
00612       end if
00613       call setlinestyle (imask)
00614       ierr= lineto(iscreenxcoord(ix), iscreenycoord(iy))
00615       call setlinestyle (65535) ! =#ffff, so zu WATCOM-Compiler kompatibel
00616       kbeamx= ix
00617       kbeamy= iy
00618       return
00619       end
00620
00621 C
00622 C  Relative Zeichenbefehle
00623 C
00624
00625       subroutine movrel (iX, iY)
00626       include 'TKTRNX.FD'
00627       ixx= kbeamx + ix
00628       iyy= kbeamy + iy
00629       call movabs (ixx, iyy)
00630       return
00631       end
00632
00633
00634
00635       subroutine pntrel (iX, iY)
00636       include 'TKTRNX.FD'
00637       ixx= kbeamx + ix
00638       iyy= kbeamy + iy
00639       call pntabs (ixx, iyy)
00640       return
00641       end
00642
00643
00644
00645       subroutine drwrel (iX, iY)
00646       include 'TKTRNX.FD'
00647       ixx= kbeamx + ix
00648       iyy= kbeamy + iy
00649       call drwabs (ixx, iyy)
00650       return
00651       end
00652
00653
00654
00655       subroutine dshrel (iX, iY, iMask)
00656       include 'TKTRNX.FD'
00657       ixx= kbeamx + ix
00658       iyy= kbeamy + iy
00659       call dshabs (ixx, iyy, imask)
00660       return
00661       end
00662
00663 C
00664 C   Ersatz SEELOC der CP/M-Version, SEELOC1 unnötig
00665 C
00666
00667       subroutine seeloc (IX,IY)
00668       include 'TKTRNX.FD'
00669       ix= kbeamx
00670       iy= kbeamy
00671       return
00672       end
00673
00674
00675
00676       Subroutine swind1 (ix1,iy1, ix2,iy2)
00677       include 'FGRAPH.FD'
00678       integer iScreenXcoord, iScreenYcoord
00679       call setcliprgn (iscreenxcoord(ix1),iscreenycoord(iy1),
00680      1                     iscreenxcoord(ix2),iscreenycoord(iy2))
```

```
00681       return
00682       end
00683
00684
00685
00686       Subroutine alpha
00687       implicit none
00688       include 'FGRAPH.FD'
00689       integer*2 iErr
00690       ierr= setvideomode($defaultmode)
00691       return
00692       end
00693
00694 C
00695 C  Textausgabe
00696 C
00697
00698       subroutine csize (Ixlen,iylen)
00699       include 'TKTRNX.FD'
00700       ixlen= khorsz
00701       iylen= kversz
00702       return
00703       end
00704
00705
00706
00707       subroutine toutpt (iChr)
00708       include 'FGRAPH.FD'
00709       include 'TKTRNX.FD'
00710       record /xycoord/ oldxy
00711       integer iScreenXcoord, iScreenYcoord
00712       integer*2 iErr
00713       call moveto (iscreenxcoord(kbeamx),iscreenycoord(kbeamy+kversz)
00714      1      , oldxy)
00715       ierr= setcolor(itxtcol)
00716       call outgtext (char(ichr)//char(0))
00717       ierr= setcolor(ilincol)
00718       kbeamx= kbeamx+khorsz
00719       call moveto (iscreenxcoord(kbeamx), iscreenycoord(kbeamy), oldxy)
00720       return
00721       end
00722
00723
00724
00725       subroutine toutst (nChr, iChrArr)
00726       integer iChrArr (1)
00727       if (nchr.eq.0) return
00728       do 10 i=1,nchr
00729 10     call toutpt (ichrarr(i))
00730       return
00731       end
00732
00733
00734
00735       subroutine toutstc (String)
00736       character *(*) String
00737       do 10 i=1,istringlen(string)
00738  10    call toutpt (ichar(string(i:i)))
00739       return
00740       end
00741
00742
00743
00744       subroutine statst (String)
00745       include 'FGRAPH.FD'
00746       include 'TKTRNX.FD'
00747       record /rccoord/ s
00748       character *(*) String
00749       character *80 Buf
00750       buf= string(1:istringlen(string)) ! Mit Blanks auf 80 Zeichen aufgefüllt
00751       call settextposition (1,1,s)
00752       call outtext (buf(1:min(80,kstcol)))
00753       return
00754       end
00755
00756 C
00757 C  Eingabe
00758 C
00759
00760       subroutine tinput (iChr)
00761       integer *2 kTinput
00762       ichr= ktinput()  ! Konversion Integer*2 nach *4 durch Compiler
00763       return
00764       end
00765
00766
00767       subroutine dcursr (IC,IX,IY)
```

```
00768       include 'TKTRNX.FD'
00769       integer ic, ix, iy
00770       integer*2 ic2, ix2, iy2
00771       if (imouse.ne.0) then
00772        call gincrs (ic2,ix2,iy2)
00773        ix= ix2                    ! Watcom: Konvertierung int*2 in int*4
00774        iy= iy2
00775        ic= ic2
00776       else
00777        call graphicerror (5, ' ') ! No Mousedriver available, use Keyboard
00778        call tinput (ic)
00779        ix= 0
00780        iy= 0
00781       end if
00782       ix= irevscreenxcoord(ix)
00783       iy= irevscreenycoord(iy)
00784       return
00785       end
00786
00787 C
00788 C  Interface lib$movc3 (Anpassung Parameterübergabe durch "TcsDDosA.FI"
00789 C
00790       subroutine lib_movc3 (iLen, sou, dst)
00791       integer iLen
00792       character *(*) sou,dst
00793       call lib_movc3_ (ilen, sou, dst)
00794       return
00795       end
00796
00797 C
00798 C>  Entry Dummyroutinen
00799 C
00800       subroutine  anmode
00801 C> AlfMod
00802       entry       alfmod
00803 C> pClipt
00804       entry       pclipt
00805 C> ioWait
00806       entry       iowait
00807       return
00808       end
00809
00810
00811
00812       logical function winselect (iDummy)
00813       winselect= .false.
00814       return
00815       end
```

3.40 TKTRNX.fd File Reference

DOS Port: TCS Common Block TKTRNX.

3.40.1 Detailed Description

DOS Port: TCS Common Block TKTRNX.

Version

1.0

Author

Dr.-Ing. Klaus Friedewald

Common Block TKTRNX, version for DOS and INTEGER∗4 variables (WATCOM-Compiler)

Because the following declaration not beeing part of a module, DOXYGEN could not interpret the combinattion COMMON / INTEGER. Workaround: \cond ... \endcond

Definition in file TKTRNX.fd.

3.41 TKTRNX.fd

```
00001 C> \file TKTRNX.fd
00002 C> \brief   DOS Port: TCS Common Block TKTRNX
00003 C> \version 1.0
00004 C> \author  Dr.-Ing. Klaus Friedewald
00005 C> \~german
00006 C> Common Block TKTRNX, Version für DOS und INTEGER*4 Variablen (WATCOM-Compiler)
00007 C> \~english
00008 C> Common Block TKTRNX, version for DOS and INTEGER*4 variables (WATCOM-Compiler)
00009 C> \~german
00010 C> \note
00011 C> Da die folgende Definition kein Bestandteil eines Moduls
00012 C> ist, versagt der DOXYGEN-Parser bei der Kombination von
00013 C> COMMON und integer. Workaround: \\cond ... \\endcond
00014 C> \~english
00015 C> Because the following declaration not beeing part of a module, DOXYGEN could
00016 C> not interpret the combinattion COMMON / INTEGER.
00017 C> Workaround: \\cond ... \\endcond
00018 C> \~
00019 C> \cond
00020 C>
00021 C Common Block TKTRNX, Version für DOS und INTEGER*4 Variablen (WATCOM-Compiler)
00022 C
00023       COMMON /tktrnx/
00024 c           kbaudr,kerror,kgrafl,
00025      1 khomey,
00026 c           kkmode,
00027      2 khorsz,kversz,
00028 c           kitalc,ksizef,
00029      3 klmrgn,krmrgn, kscrx,kscry,
00030 c           ktblsz,khorzt(10),kvertt(10),
00031      4 kbeamx,kbeamy,
00032 c           kmovef,kpchar(4),kdasht,
00033      5 kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
00034 c      trealx,trealy,timagx,timagy,
00035      6 trcosf,trsinf,trscal
00036      u ,xfac,yfac,xlog,ylog,kstcol,
00037      u ilincol, ibckcol, itxtcol, imouse
00038       SAVE /tktrnx/
00039
00040       integer iTktrnxL
00041       parameter(itktrnxl=29) ! +11)
00042
00043 c Neue Variablen:
00044 c     kScrX, kScrY: Zeichenfläche in Pixeln
00045 c           Unterer Bildschirmrand für eine Statuszeile freigehalten
00046 c     kBeamX, kBeamY: Aktuelle Strahlposition im (1024/780) Koordinatensystem
00047 c     kStCol: Maximale Zeichenzahl in der Statuszeile
00048 c     iLinCol, iBckCol, iTxtCol: Farbindices
00049 c     iMouse: Anzahl der Maustasten. iMouse=0: keine Maus vorhanden
00050 c
00051 c Achtung:
00052 c       Anpassung Parameters iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
00053 c     Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
00054 c
00055 C
00056 C> \endcond
```

Index