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

(2024,347, x)

Author

(C) 2022 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 2573 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 2444 of file AG2.for.

3.1.2.20 esplit()

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

Definition at line 2477 of file AG2.for.

3.1.2.21 expoutc()

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

Definition at line 2497 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 2932 of file AG2.for.

3.1.2.25 findle()

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

Definition at line 2951 of file AG2.for.

3.1.2.26 fonlyc()

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

Definition at line 2413 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 3076 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 2676 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 2973 of file AG2.for.

3.1.2.41 locle()

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

Definition at line 2991 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 2742 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 iy1,
             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 2628 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 2817 of file AG2.for.

3.1.2.54 rescom()

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

Definition at line 3060 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 3009 of file AG2.for.

3.1.2.57 roundu()

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

Definition at line 3025 of file AG2.for.

3.1.2.58 savcom()

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

Definition at line 3044 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 2880 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 2653 of file AG2.for.

3.1.2.76 width()

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

Definition at line 2701 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

```
C> \file       AG2.for
C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
C> \version    (2024,347, x)
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C>  Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
C> \note
C>     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
C>     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
C>
C> \~english
C> Layer 2: scientific 2-D graphic subroutines
C> \note
C>     The control character for exponent (originally -1) is now SOH=char(1)
C>     and for index (originally -2) STX=char(2).
C>
C> \~
C> \note \verbatim
C>   Package:
C>    - AG2.for:          chart plotting routines
C>    - AG2Holerith.for:  deprecated routines
C>    - AG2USR.for:       default userroutines
C>    - G2dAG2.fd:        commonblock
C> \endverbatim
C
C
C  Tektronix Advanced Graphics 2 - Version 2.x
C
C
C     Neuer Code in Fortran 77. Die Verwendung der im Manual dokumentierten
C     Unterprogramme bleibt unveraendert, die direkte Manipulation von
C     Variablen des zugrundeliegenden Commonblockes ist jedoch nicht mehr
C     empfehlenswert. IBASEX (iPar) und IBASEY(iPar) mit ipar <>0,
C     IBASEC, COMGET und COMSET sollten in neuen Programmen nicht verwendet
C     werden.
C
C     Die Zwischenspeicherung der Statusvariablen ueber
```

```
C          SAVCOM und RESCOM
C     und die Achsensteuerung ueber
C          IBASEX(0), IBASEY(0) und IOTHER
C     werden weiterhin unterstuetzt.
C
C     Die Implementation der Unterprogramme COMGET und COMSET setzt die gleiche
C     Laenge von REAL und INTEGER-Variablen voraus.
C
C     Da Holerithvariablen von modernen Compilern uneinheitlich unterstuetzt
C     werden (4Habcd entweder als gepackte Integervariable oder als Character-
C     variable interpretiert), wurden die folgenden Routinen angepasst:
C      - subroutine PLACE (Lit): Lit wird nur noch als Ordnungszahl (1..13)
C        und nicht mehr alternativ als Literal ('STD', 'UPH') interpretiert.
C
C     subroutine LEAP (iyear): Die Schaltjahrkorrektur erfolgt nicht mehr
C     als SUBROUTINE ueber einen Common-Block, sondern direkt als
C     integer function LEAP (iyear) ! = 1: Schaltjahr, sonst 0
C
C     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
C     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
C
C     Intern erfolgt die Stringverarbeitung ueber Charactervariablen als
C     nullterminierte C-Strings.
C
C     Der User-API wurden die folgenden Unterprogramme als Charactervarianten
C     der Original-Holerithroutinen hinzugefuegt:
C      - subroutine NUMSETC (fnum,nbase, outstr,fillstr)
C      - subroutine FONLYC (fnum,iwidth,idec, outstr,fillstr)
C      - subroutine EFORMC (fnum,iwidth,idec, outstr,fillstr)
C      - subroutine EXPOUTC (nbase,iexp, outstr,fillstr)
C      - subroutine ALFSETC (fnum,iwidth,labtyp,outstr)
C      - subroutine NOTATEC (IX,IY,LENCHR,IARRAY)
C      - subroutine JUSTERC
C
C      - subroutine USESETC (fnum, iwidth, nbase, labstr)
C
C      subroutine MONPOS (nbase,iy1,dpos, spos) ! spos ist INTEGER
C      subroutine GLINE (nbase,datapt,spos) ! spos ist INTEGER
C
C     Der Code ab Version 2.0 wird nicht mehr fuer CP/M entwickelt. Letzte
C     unter CP/M compilierbare Version: (2006, 013, 1)
C
C     Zugehoerige Module:
C      - AG2.FOR:     Basisfunktionen
C      - AG2Holerith: Veraltete Unterprogramme zur Wahrung der Kompatibilitaet
C                     (Unterstuetzung Holerithvariablen und vektorisierter Zu-
C                     griff auf den Commonblock)
C      - AG2USR.FOR:  Userroutinen
C      - G2dAG2.fd:   Commonblockdefinition
C

C
C  Ausgabe der Softwareversion
C
      subroutine ag2lev (ilevel)
      implicit none
      integer ilevel(3)

      call tcslev (ilevel) ! level(3)= System aus TCS
      ilevel(1)=2024       ! Aenderungsjahr
      ilevel(2)= 347       ! Aenderungstag
      return
      end



C
C  Setzen allgemeiner Commonvariablen
C
      subroutine line (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cline= ipar
      return
      end



      subroutine symbl (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      csymbl= ipar
      return
      end



      subroutine steps (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      csteps= ipar
      return
      end



      subroutine infin (par)
      implicit none
      real par
      include 'G2dAG2.fd'

      if (par .gt. 0.) then
       cinfin= par
      end if
      return
      end



      real function ag2infin ()
      implicit none
      include 'G2dAG2.fd'

      ag2infin= cinfin
      return
      end



      subroutine npts (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cnpts= ipar
      return
      end



      subroutine stepl (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cstepl= ipar
      return
      end



      subroutine sizes (par)
      implicit none
      real par
      include 'G2dAG2.fd'

      csizes= par
      return
      end



      subroutine sizel (par)
      implicit none
      real par
      include 'G2dAG2.fd'

      csizel= par
      return
      end



C
C  Setzen der achsenbezogenen Commonvariablen
C
      subroutine xneat (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyneat(1) = ipar .ne. 0
      return
      end



      subroutine yneat (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyneat(2) = ipar .ne. 0
      return
      end



      subroutine xzero (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyzero(1) = ipar .ne. 0
      return
      end



      subroutine yzero (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyzero(2) = ipar .ne. 0
      return
      end



      subroutine xloc (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyloc(1)= ipar
      return
      end



      subroutine yloc (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyloc(2)= ipar
      return
      end



      subroutine xloctp (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyloc(1)= ipar+abs(cxysmax(2)-cxysmin(2))
      return
      end



      subroutine ylocrt (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxyloc(2)= ipar + abs(cxysmax(1)-cxysmin(1))
      return
      end



      subroutine xlab (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxylab(1)= ipar
      return
      end



      subroutine ylab (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxylab(2)= ipar
      return
      end



      subroutine xden (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 10)) then
       cxyden(1)= ipar
       cxytics(1)= 0
       cxymtcs(1)= 0
      end if
      return
      end



      subroutine yden (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 10)) then
       cxyden(2)= ipar
       cxytics(2)= 0
       cxymtcs(2)= 0
      end if
      return
      end



      subroutine xtics (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxytics(1)= abs(ipar)
      return
      end



      subroutine ytics (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxytics(2)= abs(ipar)
      return
      end



      subroutine xlen (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if (ipar .ge. 0) then
       cxylen(1)= ipar
      end if
      return
      end



      subroutine ylen (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if (ipar .ge. 0) then
       cxylen(2)= ipar
      end if
      return
      end



      subroutine xfrm (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 6)) then
       cxyfrm(1)= ipar
      end if
      return
      end



      subroutine yfrm (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 6)) then
       cxyfrm(2)= ipar
      end if
      return
      end



      subroutine xmtcs (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxymtcs(1)= abs(ipar)
      return
      end



      subroutine ymtcs (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      cxymtcs(2)= abs(ipar)
      return
      end



      subroutine xmfrm (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 6)) then
       cxymfrm(1)= ipar
      end if
      return
      end



      subroutine ymfrm (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 6)) then
       cxymfrm(2)= ipar
      end if
      return
      end



      subroutine dlimx (xmin,xmax)
      implicit none
      real xmin,xmax
      include 'G2dAG2.fd'

      cxydmin(1)= xmin
      cxydmax(1)= xmax
      return
      end



      subroutine dlimy (ymin,ymax)
      implicit none
      real ymin,ymax
      include 'G2dAG2.fd'

      cxydmin(2)= ymin
      cxydmax(2)= ymax
      return
      end



      subroutine slimx (ixmin,ixmax)
      implicit none
      integer ixmin,ixmax
      include 'G2dAG2.fd'

      cxysmin(1)= ixmin
      cxysmax(1)= ixmax
      return
      end



      subroutine slimy (iymin,iymax)
      implicit none
      integer iymin,iymax
      include 'G2dAG2.fd'

      cxysmin(2)= iymin
      cxysmax(2)= iymax
      return
      end



      subroutine place (ipar)
      implicit none
      include 'G2dAG2.fd'
      integer ipar

      integer postab (4,13)        ! Koordinaten des Zeichenbereiches
      data postab /150,900, 125,700,
     2             150,850, 525,700,
     3             150,850, 150,325,
     4             150,450, 525,700,
     5             650,950, 525,700,
     6             150,450, 150,325,
     7             650,950, 150,325,
     8             150,325, 525,700,
     9             475,650, 525,700,
     a             800,975, 525,700,
     1             150,325, 150,325,
     2             475,650, 150,325,
     3             800,975, 150,325/
      save postab

      if ((ipar .ge. 1) .and. (ipar.le.13)) then
       cxysmin(1)= postab(1,ipar)
       cxysmax(1)= postab(2,ipar)
       cxysmin(2)= postab(3,ipar)
       cxysmax(2)= postab(4,ipar)
      end if
      return
      end



      subroutine xtype (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 1) .and. (ipar .le. 8)) then
       cxytype(1)= ipar
      end if
      return
      end



      subroutine ytype (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 1) .and. (ipar .le. 8)) then
       cxytype(2)= ipar
      end if
      return
      end



      subroutine xwdth (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if (ipar .ge. 0) then
       cxywdth(1)= ipar
      end if
      return
      end



      subroutine ywdth (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if (ipar .ge. 0) then
       cxywdth(2)= ipar
      end if
      return
      end



      subroutine xetyp (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 4)) then
       cxyetyp(1)= ipar
      end if
      return
      end



      subroutine yetyp (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      if ((ipar .ge. 0) .and. (ipar .le. 4)) then
       cxyetyp(2)= ipar
      end if
      return
      end



      subroutine setwin
      implicit none
      include 'G2dAG2.fd'

      call twindo (cxysmin(1),cxysmax(1), cxysmin(2),cxysmax(2))
      call dwindo (cxydmin(1),cxydmax(1), cxydmin(2),cxydmax(2))
      if (cxytype(1) .eq. 2) then
       if (cxytype(2) .eq. 2) then
        call logtrn (3)
       else
        call logtrn (1)
       end if
      else if (cxytype(2) .eq. 2) then
        call logtrn (2)
      else
       call lintrn
      end if
      return
      end



      subroutine dinitx
      implicit none
      include 'G2dAG2.fd'

      cxydmin(1)= 0.        ! Datenbereich
      cxydmax(1)= 0.
      cxywdth(1)= 0         ! Dezimalstellen
      cxydec(1)=  0         ! Dezimalstellen
      cxyepon(1)= 0         ! Exponent Label
      return
      end



      subroutine dinity
      implicit none
      include 'G2dAG2.fd'

      cxydmin(2)= 0.        ! Datenbereich
      cxydmax(2)= 0.
      cxywdth(2)= 0         ! Dezimalstellen
      cxydec(2)=  0         ! Dezimalstellen
      cxyepon(2)= 0         ! Exponent Label
      return
      end



      subroutine hbarst (ishade,iwbar,idbar)
      implicit none
      integer ishade,iwbar,idbar
      include 'G2dAG2.fd'

      cline= -3
      if ((ishade .ge. 0).and. (ishade .le. 15)) csymbl= ishade
      csizes= real(idbar)
      csizel= real(iwbar)

      if (cxyfrm(2) .eq. 5) then
       cxyfrm(2)= 2
      else if (cxyfrm(2) .eq. 6) then
       cxyfrm(2)= 1
      end if
      return
      end



      subroutine vbarst (ishade,iwbar,idbar)
      implicit none
      integer ishade,iwbar,idbar
      include 'G2dAG2.fd'

      cline=  -2
      if ((ishade .ge. 0) .and. (ishade .le. 15)) csymbl= ishade
      csizes= real(idbar)
      csizel= real(iwbar)
      if (cxyfrm(1) .eq. 5) then
       cxyfrm(1)= 2
      else if (cxyfrm(1) .eq. 6) then
       cxyfrm(1)= 1
      end if
      return
      end



C
C  Berechnung der Commonvariablen
C
      subroutine binitt
      implicit none
      integer ih
      include 'G2dAG2.fd'

      cline= 0
      csymbl= 0
      csteps= 1
      cinfin= 1.e30
      cnpts= 0
      cstepl= 1
      cnumbr= 0
      csizes= 1.
      csizel= 1.

      cxyneat(1)= .true.
      cxyneat(2)= .true.
      cxyzero(1)= .true.
      cxyzero(2)= .true.
      cxyloc(1)= 0
      cxyloc(2)= 0
      cxylab(1)= 1
      cxylab(2)= 1
      cxyden(1)= 8
      cxyden(2)= 8
      cxytics(2)= 0
      cxytics(2)= 0

      call csize (ih,cxylen(1))
      cxylen(2)= cxylen(1)

      cxyfrm(1)= 5
      cxyfrm(2)= 5
      cxymtcs(1)= 0
      cxymtcs(2)= 0
      cxymfrm(1)= 2
      cxymfrm(2)= 2
      cxydec(1)= 0
      cxydec(2)= 0
      cxydmin(1)= 0.
      cxydmin(2)= 0.
      cxydmax(1)= 0.
      cxydmax(2)= 0.

      cxysmin(1)= 150
      cxysmin(2)= 125
      cxysmax(1)= 900
      cxysmax(2)= 700

      cxytype(1)= 1
      cxytype(2)= 1
      cxylsig(1)= 0
      cxylsig(2)= 0
      cxywdth(1)= 0
      cxywdth(2)= 0
      cxyepon(1)= 0
      cxyepon(2)= 0
      cxystep(1)= 1
      cxystep(2)= 1
      cxystag(1)= 1
      cxystag(2)= 1
      cxyetyp(1)= 0
      cxyetyp(2)= 0
      cxybeg(1)= 0
      cxybeg(2)= 0
      cxyend(1)= 0
      cxyend(2)= 0
      cxymbeg(1)= 0
      cxymbeg(2)= 0
      cxymend(1)= 0
      cxymend(2)= 0
      cxyamin(1)= 0.
      cxyamin(2)= 0.
      cxyamax(1)= 0.
      cxyamax(2)= 0.
      return
      end



C
C  Datenanalyse
C

      subroutine check (x,y)
      implicit none
      real  x(5),y(5)
      include 'G2dAG2.fd'

      external SPREAD ! External wg. Namenskonflikt FTN90-Intrinsic

      call typck (1,x)
      call rgchek(1,x)
      call optim (1)
      call width (1)
      if (cxystag(1) .eq. 1) call spread (1)
      call tset (1)

      call typck (2,y)
      call rgchek(2,y)
      call optim(2)
      call width(2)
      if (cxystag(2) .eq. 1) call spread (2)
      call tset (2)
      return
      end



      subroutine typck (ixy, arr)
      implicit none
      integer ixy
      real arr(5)
      integer i
      include 'G2dAG2.fd'

      if ((cxytype(ixy) .lt. 3) .or. (nint(arr(1)) .lt. -1 )) then
       if ((cnpts .ne. 0) .or. (nint(arr(1)) .ne. -2) ) return
       i= nint(arr(3))
       if ( i .eq. 1) then
        cxytype(ixy)= 8
       else if ( i .eq. 4) then
        cxytype(ixy)= 7
       else if ( i .eq. 12) then
        cxytype(ixy)= 6
       else if ( i .eq. 13) then
        cxytype(ixy)= 5
       else if ( i .eq. 52) then
        cxytype(ixy)= 4
       else if ( i .eq. 365) then
        cxytype(ixy)= 3
       end if
      else
       cxytype(ixy)= 1
      end if
      return
      end



      subroutine rgchek (ixy,arr)
      implicit none
      integer ixy
      real arr(5)
      real amin, amax
      include 'G2dAG2.fd'

      if (cxydmax(ixy) .eq. cxydmin(ixy)) then ! Bereich schon bestimmt?
       if (cxyzero(ixy)) then ! Nullpunktunterdrueckung?
        amin= cinfin
       else
        amin= 0.
       end if
       amax= -amin
       call mnmx (arr, amin, amax)
       if (amax .eq. amin) then
        amin= amin - 0.5
        amax= amax + 0.5
       end if
       cxydmin(ixy)= amin
       cxydmax(ixy)= amax
      end if
      return
      end



      subroutine mnmx (arr,amin,amax)
      implicit none
      real arr(5), amin,amax, aminmax
      integer i, itype, nstart,nlim
      include 'G2dAG2.fd'

      if (cnpts .eq. 0) then                           ! Tek Standard-Format
       nlim= nint(arr(1)) + 1
       nstart= 2
      else
       nlim= cnpts
       nstart= 1
      end if
      if ((arr(1) .lt. 0.) .and. (cnpts .eq. 0))  then ! Kurzformate
       itype= abs(arr(1))
       if (itype .eq. 1) then
        aminmax= arr(3) + (arr(2)-1.) * arr(4)
        amin= amin1(arr(3),aminmax,amin)
        amax= amax1(arr(3),aminmax,amax)
       else if (itype .eq. 2) then
        call cmnmx (arr,amin,amax)
       else
        call umnmx (arr,amin,amax)
       end if
      else                                             ! Langformate
       if (nstart .le. nlim) then
        do 100 i= nstart, nlim
         if (arr(i) .lt. cinfin) then
          if (arr(i).lt. amin) amin= arr(i)
          if (arr(i).gt. amax) amax= arr(i)
         end if
100     continue
       end if
      end if
      return
      end



      subroutine cmnmx (arr,amin,amax)
      implicit none
      real arr(5), amin, amax
      integer nTage, iStUBGC, nIntv, iadj, imin,imax
      integer minTg,minJr, maxTg,maxJr


      nintv= nint(arr(3))
      if ((nintv .eq. 52).or.(nintv .eq. 13).or.(nintv .eq. 4)) then
       if (nintv .eq. 52) then          ! Wochen
        ntage=7
       else if (nintv .eq. 13) then     ! 28 Tagemonat
        ntage= 28
       else if (nintv .eq. 4) then      ! Quartal
        ntage=91
       end if
       call iubgc (nint(arr(4)),1, istubgc)    ! Start: Jahr=arr(4), Tag=1
       iadj= mod(istubgc,7)
       if (iadj .gt. 3) iadj=iadj-7
       imin= istubgc-iadj + nint(arr(5))*ntage ! Min= f(Startjahr,StartIntervall)
       imax= imin + nint(arr(2))*ntage

      else
       if (nintv .eq. 1) then ! Jahre
        mintg= 1
        maxtg= 1
        minjr= nint(arr(4))+1
        maxjr= nint(arr(4)+arr(2))
       else if ( nintv .eq. 12) then ! Monate
        call ymdyd (minjr,mintg, nint(arr(4)),nint(arr(5))+1,1)
        call ymdyd (maxjr,maxtg, nint(arr(4)),nint(arr(5)+arr(2)),1)
       else if ( nintv .eq. 365) then ! Tage
        minjr= nint(arr(4))
        mintg= nint(arr(5))
        maxjr= nint(arr(4))
        maxtg= nint(arr(5)+arr(2)) -1
       end if
       call iubgc (minjr,mintg, imin)
       call iubgc (maxjr,maxtg, imax)
      end if
      if (real(imax) .gt. amax) amax= real(imax)
      if (real(imin) .lt. amin) amin= real(imin)
      return
      end



C
C  Ticmarkoptimierung
C

      subroutine optim (ixy)
      implicit none
      integer ixy
      include 'G2dAG2.fd'

      if (cxytype(ixy) .eq. 2) cxylab(ixy)= 2
      if (cxylab(ixy) .eq. 2) cxylab(ixy)= cxytype(ixy)
      if (cxytype(ixy) .le. 2) then
       call loptim (ixy) ! Tic-Mark Optimierung fuer lineare und log. Daten
      else
       call coptim (ixy) ! Tic-Mark Optimierung fuer Kalenderdaten
      end if
      return
      end


      subroutine loptim (ixy)
      implicit none
      integer ixy ,i, labtyp, ntics, lsig, mtcs
      real dataint, amin,amax, aminor,amaxor, sigfac
      integer idataint
      integer mintic
      integer LINWDT, LINHGT
      real ROUNDD, ROUNDU
      include 'G2dAG2.fd'

      labtyp=abs( cxylab(ixy)) ! <0: Userlabel
      if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp

      amin= cxydmin(ixy)
      amax= cxydmax(ixy)
      ntics= abs(cxytics(ixy)) ! Anzahl >=1, 0= Flag fuer autoscale
      mintic= 0

      if (labtyp .eq. 2) then ! logarithmische Achsen
       amin= log10(max(amin,1./cinfin)) + 1.e-7  ! !> 0 => log10 definiert
       amax= log10(amax)
      end if

      aminor= amin
      amaxor= amax

      if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
       if (ixy.eq.1) then
        i= linwdt(8) ! 100 + LINWDT(3)
       else
        i= linhgt(3) ! 50 + LINHGT(3)
       end if
       ntics= (cxysmax(ixy) - cxysmin(ixy)) / i
       if (ntics .lt. 1) ntics= 1
      end if
      dataint= abs(amax-amin) / real(ntics)

310   continue ! repeat...
       if (labtyp .eq. 2) dataint= roundu(dataint,1.) ! logarithmische Achsen
       lsig= roundd(log10(dataint),1.) ! Anzahl signifikanter Nachkommastellen
       sigfac=10.**(lsig)
       if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
        if(labtyp .ne. 2) then ! nicht bei log. Achsen
         if ((dataint/sigfac) .le. 1.) then
          dataint= 1. * sigfac
          mintic= 10
         else if ((dataint/sigfac) .le. 2.) then
          dataint= 2. * sigfac
          mintic= 2
         else if ((dataint/sigfac) .le. 2.5) then
          dataint= 2.5 * sigfac
          mintic= 5
          lsig=lsig-1
         else if ((dataint/sigfac) .le. 5.) then
          dataint= 5. * sigfac
          mintic=  5
         else if ((dataint/sigfac) .le. 10.) then
          dataint= 10. * sigfac
          mintic= 10
          lsig=lsig+1
         else
          dataint= cinfin
          mintic= 0
         end if
        end if ! log. Achse
       else ! .not. neat
        lsig=lsig-2
       end if
       if (lsig .ge. 0) lsig=lsig+1
      if (cxyneat(ixy) .or. (labtyp .eq. 2) ) then ! ... until
       amin= roundd(amin+.01*sigfac,dataint) !  runde auf TicIntervall
       amax= roundu(amax-.01*sigfac,dataint) ! .01*sigfac= Genauigkeit Plot
       ntics= int(abs(amax-amin)/dataint+.0001)
       if(cxytics(ixy) .ne. 0) then ! until: ntics nicht vorbesetzt oder = vorbesetzt
        if(abs(cxytics(ixy)) .lt. ntics) then
         dataint= dataint * 1.1
         amin=aminor
         amax=amaxor
         goto 310 ! noch eine Iterationsschleife
        else if (abs(cxytics(ixy)) .gt. ntics) then
         ntics= abs(cxytics(ixy))
         amax= amin + real(ntics) * dataint
        end if ! abs(cxytics(ixy)) .eq. ntics: no action
       end if
      end if
      cxytics(ixy)= ntics
      if ((cxymtcs(ixy) .eq. 0) .and. (cxyden(ixy) .ge. 6)) then ! unbesetzt oder wenig TICS
       mtcs= mintic ! Bestimmung Minor TicMarcs
       if((mtcs .eq. 10) .or. (labtyp .eq. 2)) then
        if(cxyden(ixy) .lt. 9) mtcs=5
        if(cxyden(ixy) .lt. 7) mtcs=2
        if(labtyp .eq. 2) then ! log. Achsen
         idataint= nint(dataint)
         if (idataint .ne. 1) then ! mehrere Achsenintervalle
          i= 1
320       continue ! repeat...
           mtcs= idataint/i
          if ((mtcs*i .ne. idataint) .and. (i .lt. (idataint-1))) then ! ...until
           i= i+1
           goto 320
          else if (mtcs .gt. 10 ) then
           mtcs= 0  ! Failure
          end if
         else ! einzelne logarithmische Dekade
          if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 100* ntics) mtcs=-1 ! logarithm. Tics
          if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 20* linhgt(1)) mtcs=-2 ! Label
         end if
        end if
       end if
       cxymtcs(ixy)= mtcs
      end if

      cxylsig(ixy)= lsig
      cxyamin(ixy)= amin
      cxyamax(ixy)= amax
      if (labtyp .eq. 2) then ! logarithmische Achsen: Wiederherstellung der Originalwerte
       amax=10.**amax
       amin=10.**amin
      end if
      cxydmin(ixy)= amin
      cxydmax(ixy)= amax
      return
      end



      subroutine coptim (ixy)
      implicit none
      integer ixy , labtyp, ntics
      real dataint, amin,amax, aminor,amaxor
      integer LINWDT
      real ROUNDD, ROUNDU
      include 'G2dAG2.fd'

      if (cxytics(ixy) .eq. 1) cxytics(ixy)= 2 ! Minimum manuelle Ticwahl: 2
      labtyp=abs( cxylab(ixy)) ! <0: Userlabel
      if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp
      amin= cxydmin(ixy)
      amax= cxydmax(ixy)
      call calcon (amin,amax,labtyp,.true.) ! Konvertiere UBGC -> Labelzeiteinheit
      ntics= cxytics(ixy)
      aminor=amin
      amaxor=amax
      if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
       ntics= (cxysmax(ixy) - cxysmin(ixy)) / (25 + linwdt(1))
       if (ntics .lt. 2) ntics= 2
      end if
      dataint= abs(amax-amin) / real(ntics)

      if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
310    continue ! repeat...
        if (cxytics(ixy) .eq. 0) then ! keine manuelle Belegung erfolgt
         if (labtyp.eq.3) then ! Labeltyp: Tage
          if (dataint .le. 1.) then
           dataint= 1.
          else if (dataint .le. 7.) then
           dataint= 7.
          else if (dataint .le. 14.) then
           dataint= 14.
          else if (dataint .le. 28.) then
           dataint= 28.
          else if (dataint .le. 56.) then
           dataint= 56.
          else if (dataint .le. 128.) then
           dataint= 128.
          end if ! dataint > 128 -> unveraendert
         else if (labtyp.eq.4) then ! Labeltyp: Wochen
          if (dataint .le. 1.) then
           dataint= 1.
          else if (dataint .le. 2.) then
           dataint= 2.
          else if (dataint .le. 4.) then
           dataint= 4.
          else if (dataint .le. 8.) then
           dataint= 8.
          else if (dataint .le. 16.) then
           dataint= 16.
          else if (dataint .le. 26.) then
           dataint= 26.
          else if (dataint .le. 52.) then
           dataint= 52.
          else if (dataint .le. 104.) then
           dataint= 104.
          end if ! dataint -> unveraendert
         else if (labtyp.eq.5) then ! Labeltyp: Kalenderabschnitte
          if (dataint .le. 1.) then
           dataint= 1.
          else if (dataint .le. 2.) then
           dataint= 2.
          else if (dataint .le. 13.) then
           dataint= 13.
          else if (dataint .le. 26.) then
           dataint= 26.
          else if (dataint .le. 52.) then
           dataint= 52.
          end if ! dataint -> unveraendert
         else if (labtyp.eq.6) then ! Labeltyp: Monate
          if (dataint .le. 1.) then
           dataint= 1.
          else if (dataint .le. 2.) then
           dataint= 2.
          else if (dataint .le. 3.) then
           dataint= 3.
          else if (dataint .le. 4.) then
           dataint= 4.
          else if (dataint .le. 6.) then
           dataint= 6.
          else if (dataint .le. 12.) then
           dataint= 12.
          else if (dataint .le. 24.) then
           dataint= 24.
          else if (dataint .le. 36.) then
           dataint= 36.
          end if ! dataint -> unveraendert
         else if (labtyp.eq.7) then ! Labeltyp: Quartale
          if (dataint .le. 1.) then
           dataint= 1.
          else if (dataint .le. 2.) then
           dataint= 2.
          else if (dataint .le. 4.) then
           dataint= 4.
          else if (dataint .le. 8.) then
           dataint= 8.
          else if (dataint .le. 12.) then
           dataint= 12.
          else if (dataint .le. 16.) then
           dataint= 16.
          else if (dataint .le. 24.) then
           dataint= 24.
          end if ! dataint -> unveraendert
         else if (labtyp.eq.8) then ! Labeltyp: Jahre
          if (dataint .le. 1.) then
           dataint= 1.
          else if (dataint .le. 2.) then
           dataint= 2.
          else if (dataint .le. 5.) then
           dataint= 5.
          else if (dataint .le. 10.) then
           dataint= 10.
          else if (dataint .le. 20.) then
           dataint= 20.
          else if (dataint .le. 50.) then
           dataint= 50.
          else if (dataint .le. 100.) then
           dataint= 100.
          end if ! dataint -> unveraendert
         end if ! labtyp 3..8
        end if ! manuelle Vorbesetzung
        amin= roundd(amin,dataint) !  runde auf TicIntervall
        amax= roundu(amax,dataint)
        ntics= ifix(abs(amax-amin)/dataint+.0001)
        if (ntics .eq. 0) ntics = 2
       if(cxytics(ixy) .ne. 0) then ! until: ntics nicht oder = vorbesetzt
        if(abs(cxytics(ixy)) .lt. ntics) then ! Verringere Ticanzahl
         dataint= dataint * 1.1
         amin=aminor
         amax=amaxor
         goto 310 ! noch eine Iterationsschleife
        else if (abs(cxytics(ixy)) .gt. ntics) then ! Vergroessere Ticanzahl
```

```fortran
         ntics= abs(cxytics(ixy))
         amax= amin + real(ntics) * dataint
        end if ! abs(cxytics(ixy)) .eq. ntics: no action
       end if ! Ende der Schleife
      end if ! neat
      cxytics(ixy)= ntics
      cxylsig(ixy)= 0
      cxyamin(ixy)= amin
      cxyamax(ixy)= amax
      call calcon (amin,amax,labtyp,.false.) ! Labelzeiteinheit -> UBGC
      cxydmin(ixy)= amin
      cxydmax(ixy)= amax
      return
      end



C
C  Kalenderroutinen
C



      real function calpnt (arr,i)
      implicit none
      integer i
      real arr(5)
      integer iy,idays, itmp
      integer icltyp, istyr, istper, iubg1, iweek1, nodays
      save icltyp, istyr, istper, iubg1, iweek1, nodays

      if (i .eq. 1) then ! 1. Datenpunkt: Formatanalyse, Parameterberechnung
       istyr= nint(arr(4))
       istper= nint(arr(5))
       itmp= nint(arr(3)) ! Laenge Intervall in Tagen
       if (itmp .eq. 12) then ! Zeitintervall Monat
        icltyp= 2
       else if (itmp .eq. 365) then ! Zeitintervall Tage
        icltyp=3
        call iubgc (istyr,istper,iubg1)
       else if (itmp .eq. 52) then ! Zeitintervall Wochen
        icltyp= 4
        nodays= 7
       else if (itmp .eq. 13) then ! Zeitintervall 4 Wochen
        icltyp= 5
        nodays= 28
       else if (itmp .eq. 4) then  ! Zeitintervall Quartal
        icltyp= 6
        nodays= 91
       else ! Zeitintervall Jahre
        icltyp= 1
       end if
       if (icltyp .ge. 4) then
        call iubgc (istyr,1,iubg1)
        itmp= mod(iubg1+1,7)
        if(itmp .gt. 3) itmp= itmp-7
        iweek1= iubg1-itmp
        iubg1= iweek1+(istper-1)*nodays
       end if
      end if ! Ende Initialisierung, jetzt Berechnung

      if (icltyp .eq. 1) then ! Zeitintervall Jahr
       call iubgc (istyr+i,1,iubg1)
       calpnt= iubg1
      else if (icltyp .eq. 2) then ! Zeitintervall Monat
       call ymdyd (iy,idays,istyr,istper+i,1)
       call iubgc (iy,idays,iubg1)
       calpnt= iubg1 ! Zeitintervall Tage
      else if (icltyp .eq. 3) then
       calpnt= iubg1+i-1
      else ! Zeitintervall Wochen oder 4 Wochen
       calpnt= iweek1+(istper-1+i)*nodays
      end if
      return
      end



      subroutine calcon (amin,amax,labtyp,ubgc)
      implicit none
      real amin, amax
      integer labtyp
      logical ubgc
      integer iubg1, iubg2, iday1, iadj, id, month1,month2 , imin,imax
      real dimin, dimax
      integer iweek1
      real fnoday
      integer iy1,iy2, iy3,iy4, idays
      save iweek1, fnoday
      save iy1,iy2, iy3, iy4, idays

      real ROUNDD, ROUNDU

      if (labtyp .le. 3) return ! nicht Kalender, bzw.Tage: keine Transformation

      if (ubgc) then ! Konvertierung UBGC in Labeltype
       if ( (labtyp .eq. 4).or.(labtyp .eq. 5).or.(labtyp .eq. 7) ) then
        if (labtyp .eq. 4) fnoday= 7.
        if (labtyp .eq. 5) fnoday= 28.
        if (labtyp .eq. 7) fnoday= 91.
        iubg1=amin
        iubg2=amax
        call oubgc (iy1,idays,iubg1) ! Wochenanfang der 1.KW Startjahr
        iday1=iubg1-idays+1
        iadj=mod(iday1+1,7)
        if(iadj .gt. 3) iadj=iadj-7
        iweek1= iday1-iadj           ! Merken in iweek1
        dimin= roundd(real(iubg1-iweek1),fnoday)
        dimin= dimin/fnoday+1.
        call oubgc (iy2,idays,iubg2)
        dimax= roundu(real(iubg2-iweek1),fnoday)
        dimax= dimax/fnoday
       else if (labtyp .eq. 6) then
        call oubgc (iy1,idays,nint(amin))
        call ydymd (iy1,idays,iy3,month1,id)
        dimin= month1
        call oubgc (iy2,idays,nint(amax))
        call ydymd (iy2,idays,iy4,month2,id)
        dimax= (iy4-iy3)*12+month2
        if(id .gt. 1) dimax=dimax+1.
       else if (labtyp .eq. 8) then
        call oubgc (iy1,idays,nint(amin))
        dimin= iy1
        call oubgc(iy2,idays,nint(amax))
        dimax= iy2
        if(idays .gt. 1) dimax=dimax+1.
       end if
       amin= dimin-1.
       amax= dimax-1.
       return

      else ! Konvertierung Labeltype in UBGC
       amin=amin+1.
       amax=amax+1.
       if ((labtyp .eq. 4).or.(labtyp .eq. 5).or.(labtyp .eq. 7)) then
        amin= iweek1 + (nint(amin)-1) * nint(fnoday)
        amax= iweek1+(nint(amax)-1)*nint(fnoday)
       else if (labtyp .eq. 6)then
        iy4= iy3
        call ymdyd (iy1,idays,iy3,nint(amin),1)
        call iubgc (iy1,idays,imin)
        amin= imin
        call ymdyd (iy2,idays,iy4,nint(amax),1)
        call iubgc (iy2,idays,imax)
        amax= imax
       else if (labtyp .eq. 8) then
        call iubgc (nint(amin),1,imin)
        amin= imin
        call iubgc (nint(amax),1,imax)
        amax= imax
       end if
      endif
      return
      end


      subroutine ymdyd (iJulYrOut,iJulDayOut,
     1                               iGregYrIn,iGregMonIn,iGregDayIn)
      implicit none
      integer iJulYrOut,iJulDayOut, iGregYrIn,iGregMonIn,iGregDayIn
      integer iJulYrIn,iJulDayIn, iGregYrOut,iGregMonOut,iGregDayOut
      integer iMon, LEAP
      integer iDatTab(12)
      save idattab
      data idattab /0,31,59,90,120,151,181,212,243,273,304,334/

      ijulyrout= igregyrin
      imon= igregmonin
100   if (imon .lt. 1) then ! while iMon .not. in [1..12]
       imon= imon + 12
       ijulyrout= ijulyrout-1
       goto 100
      else if (imon .gt. 12) then
       imon= imon -12
       ijulyrout= ijulyrout+1
       goto 100
      end if
      ijuldayout= igregdayin + idattab(imon)
      if (imon .gt.2) ijuldayout= ijuldayout + leap(ijulyrout)
      return

C> entry subroutine YMDYD (iJulYrIn,iJulDayIn,iGregYrOut,iGregMonOut,iGregDayOut)
      entry ydymd(ijulyrin,ijuldayin,
     1                        igregyrout,igregmonout,igregdayout)

      igregdayout= ijuldayin
      igregyrout= ijulyrin
110   if (igregdayout .lt. 1) then ! while iGregDayOut .not. in [1..365(366)]
       igregyrout= igregyrout-1
       igregdayout= igregdayout + 365 + leap(igregyrout)
       goto 110
      else if (igregdayout .gt. 365+ leap(igregyrout)) then
       igregyrout= igregyrout+1
       igregdayout= igregdayout - 365 - leap(igregyrout)
       goto 110
      end if

      igregmonout= int( real(igregdayout)/29.5+1.)
      if (igregdayout .le. idattab(igregmonout)) then
       if ((igregmonout .le. 2) .or.
     1   (igregdayout.le.(idattab(igregmonout)+leap(igregyrout)))) then
        igregmonout= igregmonout-1
       end if
      end if
      igregdayout= igregdayout- idattab(igregmonout)
      if (igregmonout .gt. 2) igregdayout= igregdayout -leap(igregyrout)
      return
      end



      integer function  leap (iyear)
      implicit none
      integer iyear
      if (  (mod(iyear,4) .eq. 0) .and.
     1     ((mod(iyear,100).ne.0) .or. (mod(iyear,400).eq.0))  ) then
       leap= 1
      else
       leap= 0
      end if
      return
      end



      subroutine iubgc(iyear,iday, iubgcO)
      implicit none
      integer iyear,iday,iubgcO
      integer iYr1

      iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
      iubgco= 365* (iyear-1901) ! Verhinderung Overflow: Offset im Faktor
      iubgco= iubgco + int(iyr1/4) - int(iyr1/100) + int(iyr1/400)
      iubgco= iubgco + iday -460 ! Bezugsdatum 1.1.1901= 365*1901 + 460 Schalttage
      return
      end



      subroutine oubgc(iyear,iday,iubgcI)
      implicit none
      integer iyear,iday,iubgcI
      integer iYr1

      iyear= int( (real(iubgci) + 694325.99) / 365.2425 )
100   continue ! Schleife der evtl. Nachiteration
       iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
       iday= iubgci + 460 - 365*(iyear-1901)
       iday= iday + int(iyr1/100) - int(iyr1/4) - int(iyr1/400)
      if (iday .lt. 1) then ! Nachiteration?
       iyear= iyear-1
       goto 100
      end if
      return
      end



C
C  Zeichenroutinen
C
```

```

      subroutine frame
      implicit none
      include 'G2dAG2.fd'

      call movabs (cxysmax(1),cxysmin(2))
      call drwabs (cxysmax(1),cxysmax(2))
      call drwabs (cxysmin(1),cxysmax(2))
      call drwabs (cxysmin(1),cxysmin(2))
      call drwabs (cxysmax(1),cxysmin(2))
      return
      end



      subroutine dsplay (x,y)
      implicit none
      real x(5),y(5)

      call setwin
      call cplot (x,y)
      call grid
      call label (1)
      call label (2)
      return
      end



      subroutine cplot (x,y)
      implicit none
      real x(5),y(5)
      logical symbol
      integer i,i1, keyx, keyy, lines, linsav, icount, imax
      real xpoint(1), ypoint(1)
      real DATGET
      include 'G2dAG2.fd'

      call keyset (x,keyx)
      call keyset (y,keyy)
      if (keyx .eq. 1) then ! standard long
       imax= x(1)
      else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
       imax= x(2)
      else ! nonstandard
       imax= cnpts
      end if
      if (keyy .eq. 1) then ! standard long
       if (imax .lt. y(1)) imax= y(1)
      else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
       if (imax .lt. y(2)) imax= y(2)
      else ! nonstandard
       if (imax .lt. cnpts) imax= cnpts
      end if

      symbol= (csymbl .ne. 0) .and.(cline .ne.-2) .and.(cline .ne.-3)

      i= 1 ! Suche Startpunkt
100   continue ! repeat
       if (i .gt. imax) return ! kein Punkt zu zeichnen
       xpoint(1)= datget(x,i,keyx)
       ypoint(1)= datget(y,i,keyy)
      if ((xpoint(1) .ge. cinfin) .or. (ypoint(1) .ge. cinfin)) then ! while
       i= i+cstepl
       goto 100
      end if

      call movea (xpoint(1),ypoint(1))
      if (cline .eq. -4) call pointa (xpoint(1),ypoint(1))
      if (cline .lt. -10) call uline (xpoint(1),ypoint(1),1)
      if (cline .eq.-2 .or. cline .eq.-3) then
       call bar (xpoint(1),ypoint(1),cline)
      end if
      if (symbol) call bsyms (xpoint(1),ypoint(1),csymbl)

      if (cline .eq. -1) then
       lines= 2
      else if ((cline .eq. -2) .or. (cline .eq. -3)) then
       lines= 3
      else if (cline .eq. -4) then
       lines=4
      else if (cline .lt. -10) then
       lines=5
      else
       lines=1 ! bei cline = 0: dash ergibt durchgezogene Linie
      end if

      i1= i+cstepl
      if (i1 .ge. imax) return
      icount= csteps
      linsav= lines

      do 900 i=i1,imax,cstepl
       xpoint(1)= datget(x,i,keyx)
       ypoint(1)= datget(y,i,keyy)
       if ((xpoint(1) .ge. cinfin) .or. (ypoint(1) .ge. cinfin)) then
        if (i.gt.imax-cstepl) return ! Der letzte Punkt ist ungueltig -> done
        if ((cline .ne. -2) .and. (cline .ne. 3)) lines= 2
       else
        if (lines .eq. 1 ) then
         call dasha (xpoint(1),ypoint(1), cline) ! dashed or solid
        else if (lines .eq. 2 ) then
         call movea (xpoint(1),ypoint(1))
         lines=linsav ! restore after missing data
        else if (lines .eq. 3 ) then
         call bar (xpoint(1),ypoint(1),0)
        else if (lines .eq. 4 ) then
         call pointa (xpoint(1),ypoint(1))
        else
         call uline (xpoint(1),ypoint(1),i)
        end if
        if (symbol) then
         icount=icount-1
         if(icount .le. 0) then
          icount= csteps
          call bsyms (xpoint(1),ypoint(1),csymbl)
         end if
        end if
       end if
900   continue
      return
      end



      subroutine keyset (array,key)
      implicit none
      integer key
      integer npts
      real array(1)
      include 'G2dAG2.fd'

      if (cnpts .ne. 0) then       ! nonstandard array
       key= 5
      else
       npts= nint(array(1))
       if (npts .ge. 0) then       ! standard long
        key= 1
       else if (npts .eq. -1) then ! short
        key= 2
       else if (npts .eq. -2) then ! short calendar
        key= 3
       else                        ! short user
        key= 4
       end if
      end if
      return
      end



      real function datget (arr,i,key)
      implicit none
      integer i, key
      real calpnt, upoint
      real arr(5) ! Dimension 5 sonst GNU-Compilerwarnung bei dat= ...arr(5)...
      real dat, olddat
      save olddat

      if (key.eq.1) then ! standard long
       dat= arr(i+1)
      else if (key.eq.2) then ! standard short
       dat= arr(3) + arr(4)*real(i-1)
      else if (key.eq.3) then ! short calendar
       dat= calpnt(arr,i)
      else if (key.eq.4) then ! user
       dat= upoint(arr,i,olddat)
      else if (key.eq.5) then ! non standard
       dat= arr(i)
      endif
      olddat= dat
      datget= dat
      return
      end
```

```



C  Balkendiagramme

      subroutine bar (x,y,line)
      implicit none
      real x, y
      integer line
      integer key, ix,iy, ixl,iyl,ixh,iyh
      real xfac, yfac
      logical VerticalBar
      integer isymb, ihalf, lspace, minx,maxx,miny,maxy, ibegx,ibegy
      SAVE isymb, ihalf, lspace, minx,maxx,miny,maxy, ibegx,ibegy
      SAVE verticalbar
      include 'G2dAG2.fd'

      if (line .ne. 0) then ! Erster Aufruf -> Parameterbestimmung
       verticalbar= line .ne. -3
       isymb= csymbl
       ihalf= .5 * csizel
       lspace= csizes
       if (lspace .le. 1) lspace=20 ! Default: 20 Pixel Schraffur
       if (ihalf .lt. 2) ihalf=20 ! Default: 40 Pixel Balkenbreite
       if (cxysmin(1) .le. cxysmax(1)) then
        minx= cxysmin(1)
        maxx= cxysmax(1)
       else
        minx= cxysmax(1)
        maxx= cxysmin(1)
       end if
       if (cxysmin(2) .le. cxysmax(2)) then
        miny= cxysmin(2)
        maxy= cxysmax(2)
       else
        miny= cxysmax(2)
        maxy= cxysmin(2)
       end if

       call seetrn(xfac,yfac, key)
       if (key .eq. 2) then ! logarithmische Werte
        ibegx= cxysmin(1)
        ibegy= cxysmin(2)
       else
        call wincot (0.,0.,ibegx,ibegy)
       end if
      end if

      call wincot (x,y,ix,iy)
      if (verticalbar) then ! vertikale Balken
       iyl= min0(ibegy,iy)
       iyh= max0(ibegy,iy)
       ixl= min0(ix-ihalf,ix+ihalf)
       ixh= max0(ix-ihalf,ix+ihalf)
      else ! horizontale Balken
       iyl= min0(iy-ihalf,iy+ihalf)
       iyh= max0(iy-ihalf,iy+ihalf)
       ixl= min0(ibegx,ix)
       ixh= max0(ibegx,ix)
      end if
      ixl=max0(ixl,minx)
      ixh=min0(ixh,maxx)
      iyl=max0(iyl,miny)
      iyh=min0(iyh,maxy)
      if ((ixh-ixl .ge. 2) .and. (iyh-iyl .ge. 2)) then ! mindestens 2x2 Pxl
       call filbox(ixl,iyl,ixh,iyh,isymb,lspace)
      end if
      return
      end



      subroutine filbox (minx,miny,maxx,maxy,ishade,lspace)
      implicit none
      integer minx,miny,maxx,maxy,ishade,lspace
      integer iminx,imaxx,iminy,imaxy
      integer i, ishift, idely, iymax
      real ximin, ximax
      real savcom (60)

      iminx= min0(minx,maxx)        ! zeichne Rechteck
      iminy= min0(miny,maxy)
      imaxx= max0(minx,maxx)
      imaxy= max0(miny,maxy)

      call movabs (iminx,iminy)
      call drwabs (imaxx,iminy)
      call drwabs (imaxx,imaxy)
      call drwabs (iminx,imaxy)
      call drwabs (iminx,iminy)

      if ((ishade .le.0) .or. (ishade .gt. 15)) return ! ohne Schraffur

      ishift= ishade / 2
      if ((ishade-ishift*2) .ne. 0) then ! Bit0: horizontale Schraffur
       i= iminy
100    continue ! repeat...
        i= i+lspace
       if (i .lt. imaxy) then
        call movabs (iminx,i)
        call drwabs (imaxx,i)
        goto 100 ! ... until
       end if
      end if ! horizontale Schraffur gezeichnet

      if (mod(ishift,2) .ne. 0) then ! Bit1: vertikale Schraffur
       i= iminx
110    continue ! repeat
        i= i+lspace
       if(i .lt. imaxx) then
        call movabs (i,iminy)
        call drwabs (i,imaxy)
        goto 110
       end if ! vertikale Schraffur gezeichnet
      end if

      if (ishade .ge. 4) then ! diagonale Schraffuren
       ximin= real(iminx)
       ximax= real(imaxx)
       call svstat (savcom) ! verwende TCS-Clipping
       call lintrn
       call dwindo (ximin,ximax,real(iminy),real(imaxy))
       call twindo (iminx,imaxx,iminy,imaxy)

       if (ishade .ge. 8) then ! Bit3: diagonal fallend
        idely= iminx-imaxx
        iymax= imaxy+imaxx-iminx
        i= iminy+lspace
120     continue ! repeat ...
         call movea (ximin,real(i))
         call drawa (ximax,real(i+idely))
         i= i+lspace
        if (i .lt. iymax) goto 120 ! ... until
        ishift= ishade -8
       else
        ishift= ishade
       end if

       if (ishift .ge. 4) then ! Bit2: diagonal steigend
        idely= imaxx-iminx
        iymax= real(imaxy)
        i= iminy - idely + lspace
130     continue ! repeat...
         call movea (ximin,real(i))
         call drawa (ximax,real(i+idely))
         i= i+lspace
        if (i .lt. iymax) goto 130 ! ...until
       end if
       call restat (savcom)
      end if ! Diagonalen
      return
      end



C  Zeichnen von Symbolen

      subroutine bsyms (x,y,isym)
      implicit none
      real x,y
      integer isym
      include 'G2dAG2.fd'

      if (isym .ge. 0) then
       call symout (isym, csizes)
      else
       call users (x,y,isym)
      end if
      call movea (x,y)
      return
      end



      subroutine symout (isym,fac)
      implicit none
      integer isym
      real fac
      integer ix,iy, ihorz,ivert

      call seeloc (ix,iy)
      if (isym .gt. 127) then
       call softek (isym)
      else if (isym .ge. 33) then
       call csize (ihorz,ivert)
       ihorz= int( real(ihorz)*.3572)
       ivert= int( real(ivert)*.3182)
       call movrel (-ihorz,-ivert)
       call alfmod
       call toutpt (isym)
      else if (isym .le. 11) then
       call teksym (isym,fac)
      end if
      call movabs (ix,iy)
      return
      end



      subroutine teksym (isym,amult)
      implicit none
      integer isym
      real amult
      integer ihalf, ifull

      ihalf= nint(8.* amult)
      ifull=ihalf * 2
      if (isym .eq. 1) then ! Kreis
       call teksym1 (0, 360, 30, 8.*amult)
      else if (isym .eq. 2) then ! X
       call movrel (ihalf,ihalf)
       call drwrel (-ifull,-ifull)
       call movrel (0,ifull)
       call drwrel (ifull,-ifull)
      else if (isym .eq. 3) then ! Dreieck
       call teksym1 (90, 450, 120, 8.*amult)
      else if (isym .eq. 4) then ! Quadrat
       call teksym1 (45, 405, 90, 8.*amult)
      else if (isym .eq. 5) then ! Stern
       call teksym1 (90, 810, 144, 8.*amult)
      else if (isym .eq. 6) then ! Raute
       call teksym1 (90, 450, 90, 8.*amult)
      else if (isym .eq. 7) then ! vertikaler Balken
       call teksym1 (90, 270, 180, 8.*amult)
      else if (isym .eq. 8) then ! Kreuz
       call movrel (0,ihalf)
       call drwrel (0,-ifull)
       call movrel (-ihalf,ihalf)
       call drwrel (ifull,0)
      else if (isym .eq. 9) then ! Pfeil nach oben
       call drwrel (-2,-6)
       call drwrel (4,0)
       call drwrel (-2,6)
       call drwrel (0,-ifull)
      else if (isym .eq. 10) then ! Pfeil nach unten
       call drwrel (-2,6)
       call drwrel (4,0)
       call drwrel (-2,-6)
       call drwrel (0,ifull)
      else if (isym .eq. 11) then ! Durchstreichung
       call teksym1 (270, 630, 120, 8.*amult)
      end if
      return
      end



      subroutine teksym1 (istart, iend, incr, siz)
      implicit none
      integer istart, iend, incr
      real siz
      integer i, mx,my,mix,miy
      real b

      b= real(istart)*.01745
      mx= nint(siz*cos(b))
      my= nint(siz*sin(b))
      call movrel (mx,my)
      do 100 i= istart+incr, iend, incr
       b= real(i)*.01745
       mix= nint(siz*cos(b))
```

```
        miy= nint(siz*sin(b))
        call drwrel (mix-mx,miy-my)
        mx= mix
        my= miy
100    continue
       return
       end



C Netz und Ticmarks

       subroutine grid
       implicit none
       integer i, mlim
       real xyext,xyextm, tintvl,tmntvl
       include 'G2dAG2.fd'

       if (cxyfrm(2) .ne. 0) then ! Zeichnen der y-Achse
        i= min0(cxysmin(1),cxysmax(1)) + cxyloc(2)
        call movabs (i, cxysmax(2))
        call drwabs (i, cxysmin(2))
        if (cxybeg(2) .ne. cxyend(2)) then ! Zeichnen y-Ticmarks
         i= cxylab(2) ! Labeltyp
         if (i .eq. 1) i= cxytype(2) ! =1: Typ entsprechend Daten
         if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
          if(cxytics(2) .ne. 0) then
           tintvl= real(cxysmax(2)-cxysmin(2)) / real( cxytics(2))
          end if
          if (cxymtcs(2) .gt. 0) tmntvl= tintvl / real(cxymtcs(2))
          call movabs(cxybeg(2),cxysmin(2))
          call drwabs(cxyend(2),cxysmin(2))
          xyext= real(cxysmin(2))
          do 100, i=1,cxytics(2)
           if (cxymbeg(2) .ne. cxymend(2)) then ! Zeichnen Minor Ticmarks
            mlim= cxymtcs(2)-1
            xyextm= xyext
110         continue ! repeat...
            if (mlim.gt.0) then ! ...until mlim <= 0
             xyextm= xyextm+tmntvl
             call movabs (cxymbeg(2), nint(xyextm))
             call drwabs (cxymend(2), nint(xyextm))
             mlim=mlim-1
             goto 110
            else if (mlim. lt. 0) then
             call logtix (2,xyext,tintvl,cxymbeg(2),cxymend(2))
            end if
           end if
           xyext= xyext+tintvl
           call movabs (cxybeg(2), nint(xyext))
           call drwabs (cxyend(2), nint(xyext))
100       continue
         end if ! Labtyp=6: Monate
        end if ! Ende Zeichnen Ticmarks
       end if ! Ende Zeichnen der Achse

       if (cxyfrm(1) .ne. 0) then ! Zeichnen der x-Achse
        i= min0(cxysmin(2),cxysmax(2)) + cxyloc(1)
        call movabs (cxysmin(1), i)
        call drwabs (cxysmax(1), i)
        if (cxybeg(1) .ne. cxyend(1)) then ! Zeichnen y-Ticmarks
         i= cxylab(1) ! Labeltyp
         if (i .eq. 1) i= cxytype(1) ! =1: Typ entsprechend Daten
         if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
          if(cxytics(1) .ne. 0) then
           tintvl= real(cxysmax(1)-cxysmin(1)) / real( cxytics(1))
          end if
          if (cxymtcs(1) .gt. 0) tmntvl= tintvl / real(cxymtcs(1))
          call movabs(cxysmin(1), cxybeg(1))
          call drwabs(cxysmin(1), cxyend(1))
          xyext= real(cxysmin(1))
          do 120, i=1,cxytics(1)
           if (cxymbeg(1) .ne. cxymend(1)) then ! Zeichnen Minor Ticmarks
            mlim= cxymtcs(1)-1
            xyextm= xyext
130         continue ! repeat...
            if (mlim.gt.0) then ! ...until mlim <= 0
             xyextm= xyextm+tmntvl
             call movabs (nint(xyextm), cxymbeg(1))
             call drwabs (nint(xyextm), cxymend(1))
             mlim=mlim-1
             goto 130
            else if (mlim. lt. 0) then
             call logtix (1,xyext,tintvl,cxymbeg(1),cxymend(1))
            end if
           end if
           xyext= xyext+tintvl
           call movabs (nint(xyext), cxybeg(1))
           call drwabs (nint(xyext), cxyend(1))
 120      continue
         end if ! Labtyp=6: Monate
        end if ! Ende Zeichnen Ticmarks
       end if ! Ende Zeichnen der Achse
       return
       end



       subroutine logtix (nbase,start,tintvl,mstart,mend)
       implicit none
       integer nbase,mstart,mend
       real start, tintvl
       integer i, logtic, ihorz, ivert, idx,idy
       character*1 loglab
       include 'G2dAG2.fd'

       call csize (ihorz,ivert)
       do 100 i=2,9
        write (unit=loglab, fmt='(i1)') i ! Unicodefaehig durch Compilerfeature
        logtic= nint(log10(real(i))*tintvl + start)
        if (nbase .eq. 1) then ! x-Achse
         idx=  -ihorz/3
         if  (mstart .gt. mend) then
          idy= ivert
         else
          idy= -ivert
         end if
         call movabs (logtic,mend)
         call drwabs (logtic,mstart)
         if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
          call movrel (idx,idy)
          call toutstc (loglab)
         end if

        else if (nbase .eq. 2) then ! y-Achse
         if  (mstart .gt. mend) then
          idx= ihorz
         else
          idx= -ihorz
         end if
         idy=  -ivert / 3
         call movabs (mend,logtic)
         call drwabs (mstart,logtic)
        end if

        if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
         call movrel (idx,idy)
         call toutstc (loglab)
        end if
 100   continue
       return
       end



       subroutine tset (nbase)
       implicit none
       integer nbase
       integer IOTHER
       integer otherbase, near, nfar, newloc, nlen
       include 'G2dAG2.fd'

       otherbase= iother(nbase)
       near= min0(cxysmin(otherbase), cxysmax(otherbase))
       nfar= max0(cxysmin(otherbase), cxysmax(otherbase))
       newloc= near + cxyloc(nbase)
       if (cxyfrm(nbase) .ne. 1) then
        if (newloc .lt. ((nfar+near)/2)) then
         nlen= cxylen(nbase)
        else
         nlen= -cxylen(nbase)
         nfar= near
        end if
        call tset2 (newloc,nfar,nlen,cxyfrm(nbase),
      1                               cxybeg(nbase),cxyend(nbase))
       else
        cxybeg(nbase)= 0
        cxyend(nbase)= 0
       end if

       if ((cxymfrm(nbase) .ne. 1) .and. (cxymtcs(nbase) .ne. 0)) then
        nlen= nlen / 2
        call tset2 (newloc,nfar,nlen,cxymfrm(nbase),
      1                               cxymbeg(nbase),cxymend(nbase))
```

```
      else
       cxymbeg(nbase)= 0
       cxymend(nbase)= 0
      end if
      return
      end



      subroutine tset2 (newloc,nfar,nlen,nfrm,kstart,kend)
      implicit none
      integer newloc,nfar,nlen,nfrm,kstart,kend

      if (nfrm .eq. 3 .or. nfrm .eq. 6) then
       kstart= newloc
      else
       kstart=newloc-nlen
      end if
      if (kstart .lt. 0) then
       kstart= 0
      else if (kend .gt. 1023) then
       kstart= 1023
      end if

      if (nfrm .eq. 2) then
       kend= newloc
      else if (nfrm .eq. 5 .or. nfrm .eq. 6) then
       kend = nfar
      else
       kend=newloc+nlen
      end if
      if (kend .lt. 0) then
       kend= 0
      else if (kend .gt. 1023) then
       kend= 1023
      end if
      return
      end



      subroutine monpos (nbase,iy1,dpos, spos)
      implicit none
      integer nbase, iy1, spos
      integer iy,idays,iubgc1
      real dpos

      call ymdyd (iy,idays,iy1, nint(dpos)+1,1)
      call iubgc (iy,idays, iubgc1)
      call gline (nbase, real(iubgc1), spos)
      return
      end



      subroutine gline (nbase,datapt,spos)
      implicit none
      integer nbase, spos
      real datapt
      integer i
      include 'G2dAG2.fd'

      if (nbase .eq. 1) then ! x-Achsengrid
       call wincot (datapt,1., spos,i)
       if (iabs(cxyend(1)-cxybeg(1)) .ge. 2) then
        call movabs(spos,cxybeg(1))
        call drwabs(spos,cxyend(1))
       end if
      else ! y-Achsengrid
       call wincot (1.,datapt, i,spos)
       if (iabs(cxyend(2)-cxybeg(2)) .ge. 2) then
        call movabs(cxybeg(2),spos)
        call drwabs(cxyend(2),spos)
       end if
      end if
      return
      end



C Label

      subroutine label (nbase)
      implicit none
      integer nbase
      logical even, stag
      integer i, icv, igap, iquadrant, labtyp, ilim, iposflag, ioff, iy
      integer ispos,isintv, iyear
      integer level1, level2
      real fnum, fac, dpos, dintv
      character *(255) labstr
      integer IOTHER
      include 'G2dAG2.fd'

      labtyp= cxylab(nbase)
      if(labtyp .eq. 1) labtyp= cxytype(nbase) ! LabTyp=1: = dataType
      if (labtyp .eq. 0) return ! LabTyp=0: keine Label

      fac= 10.**(-cxyepon(nbase))

      dintv= real(cxystep(nbase)) / real(cxytics(nbase)) ! Zwischenergebnis
      isintv= nint(real(cxysmax(nbase)-cxysmin(nbase)) * dintv)
      dintv= (cxyamax(nbase)-cxyamin(nbase)) * dintv

      call csize (i,icv) ! nur icv = vertikale Hoehe benoetigt
      igap= icv / 3
      if (nbase.eq.1) igap= 2*igap
      if (iabs(cxysmax(iother(nbase))-cxysmin(iother(nbase)))
     1                                     .gt. 2* cxyloc(nbase)) then
       iquadrant= -1 ! untere Haelfte
      else
       iquadrant= +1
      end if
      level1= min0(cxysmax(iother(nbase)),cxysmin(iother(nbase)))
     1                               - (igap-icv/3 ) + cxyloc(nbase)
     2                        + isign(igap+cxylen(nbase),iquadrant)
      level2= level1 + isign(icv+igap, iquadrant)

      if (nbase .eq. 1) then ! Label links/zentriert/rechts?
       iposflag= 0 ! x-Achse: zentriert
      else
       iposflag= -iquadrant
      end if

      stag= cxystag(nbase) .eq. 2 ! Verwendung in Schleife
      even= .false.
      ilim= cxytics(nbase) + 1

      dpos= cxyamin(nbase)
      ispos= cxysmin(nbase)

      if (iabs(labtyp) .ge. 3 .and. iabs(labtyp) .le. 8) then ! Kalenderdaten
       call oubgc (iyear,i,ifix(cxydmin(nbase))) ! i: Tag nicht benoetigt
       dpos= dpos+dintv ! 1. Tic ungelabelt
       ispos= ispos+isintv
       ilim=ilim-1
       if (nbase .eq. 1) iposflag= 1 ! x-Achse Kalender: rechtsbuendig
      end if

      do 100 i=1,ilim, cxystep(nbase)
       if ((labtyp .le. 2) .or. (labtyp .ge. 8)) then
        fnum= dpos
       else ! Kalendertyp ohne Jahr
        if (labtyp.eq.3) then ! Tage
         fnum= 7.
        else if (labtyp.eq.4) then ! Wochen
         fnum= 52.
        else if (labtyp.eq.5) then ! Periods
         fnum= 13.
        else if (labtyp.eq.6) then ! Monate
         fnum= 12.
        else if (labtyp.eq.7) then ! Quartal
         fnum= 4.
        end if ! Jahr wird wie linear behandelt
        fnum= amod(dpos-1.,fnum)+1.
       end if

       if (labtyp .lt. 0) then
        call usesetc (fnum, cxywdth(nbase), nbase, labstr)
       else if ((labtyp .eq. 6) .OR. (labtyp .eq. 3)) then
        call alfsetc (fnum, labtyp, labstr)
        if (cxywdth(nbase) .lt. len(labstr)) then
         labstr(cxywdth(nbase)+1:cxywdth(nbase)+1)= char(0)
        end if
        if (labtyp .eq. 6) call monpos (nbase,iyear,dpos,ispos)
       else
        call numsetc (fnum*fac,cxywdth(nbase),nbase,labstr)
       end if
       call justerc (labstr, iposflag, ioff)

       if (nbase .eq. 1) then ! x-Achse
        iy= level1
        if(stag .and. even) iy= level2
        even= .not. even
```

```
          call notatec (ispos+ioff,iy, labstr)
         else ! y-Achse
          call notatec (level1+ioff,ispos-igap,labstr)
          end if
         dpos= dpos+dintv
         ispos= ispos+isintv
100     continue ! end do

        if ((labtyp .ne. 2) .and. (cxyetyp(2) .ge. 0)) then ! nicht logarithm.
         if (nbase .eq. 1) then ! x-Achse
          if (stag) level2= level2 + isign(icv+igap,iquadrant)
          i=(cxysmin(nbase)+cxysmax(nbase))/2.
          iy=level2
         else
          i= level1
          iy= max0(cxysmin(nbase),cxysmax(nbase)) +icv+igap
         end if
         call remlab (nbase,cxyloc(nbase),labtyp,i,iy)
        end if
        return
        end


        subroutine numsetc (fnum,iwidth,nbase, outstr)
        implicit none
        real fnum
        integer iwidth,nbase
        character outstr *(*)
        integer iexp
        include 'G2dAG2.fd'

        if (cxytype(nbase) .eq. 2) then
         if (fnum .gt. 0.) then
          iexp= fnum + .00005
         else if (fnum .lt. 0.) then
          iexp= fnum - .00005
         else
          iexp= 0
         end if
         call expoutc (nbase,iexp, outstr)
        else if ((cxytype(nbase).eq.1) .and. (cxydec(nbase).gt.0)) then
         call fformc (fnum,iwidth, cxydec(nbase), outstr)
        else
         call iformc (fnum,iwidth, outstr)
        end if
        return
        end


        subroutine iformc (fnum,iwidth, outstr)
        implicit none
        real fnum
        integer iwidth
        character outstr *(*)
        character fmtstr *(11)

        if (iwidth .le. 0) then ! iwidth=0: ohne Label
         outstr= char(0)
         return
        end if

        if (iwidth .gt. 99) goto 200 ! Errorhandler
        write (unit=fmtstr,fmt=100, err=200) iwidth
        if (len(outstr) .gt. iwidth) then
         write (unit= outstr, fmt=fmtstr, err=200) nint(fnum),0 ! 0: End of String
        else
         write (unit= outstr, fmt=fmtstr, err=200) nint(fnum) ! evtl. ohne EoS?
        end if

        return

200     continue ! Error Handler
        outstr= '???'
        if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
        return

100     format ('(SS,I' ,i2.2, ',A1)')
        end



        subroutine fformc (fnum,iwidth,idec, outstr)
        implicit none
        real fnum
        integer iwidth,idec
```

```
      character outstr *(*)
      integer nDgtM
      real fa
      include 'G2dAG2.fd'

      ndgtm= iwidth-idec
      if (fnum .ge. 0.) then
       ndgtm= ndgtm -1  ! Ziffern Mantisse
      else
       ndgtm= ndgtm-2   ! 1 Ziffer Vorzeichen
      end if
      fa= abs(fnum) ! Skalierung mindestens 2 signfikante Stellen: .1*abs(fnum)

      if ( ((fa .lt. 10./cinfin) .or. (fa .gt. .1**idec))
     1                        .and.(fa .lt. 10.**ndgtm)) then
       call fonlyc (fnum,iwidth,idec, outstr)
      else
       call eformc (fnum,iwidth,idec, outstr)
      end if
      return
      end



      subroutine fonlyc (fnum,iwidth,idec, outstr)
      implicit none
      real fnum
      integer iwidth,idec
      character outstr *(*)
      character fmtstr *(14)

      if (iwidth .le. 0) then ! iwidth=0: ohne Label
       outstr= char(0)
       return
      end if

      if ((idec .gt. iwidth-1) .or. (iwidth .gt. 99)) goto 200 ! Errorhandler
      write (unit=fmtstr,fmt=100, err=200) iwidth,idec
      if (len(outstr) .gt. iwidth) then
       write (unit= outstr, fmt=fmtstr, err=200) fnum,0 ! 0: End of String
      else
       write (unit= outstr, fmt=fmtstr, err=200) fnum ! evtl. ohne EoS?
      end if
      return

200   continue ! Error Handler
      outstr= '???'
      if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
      return

100   format ('(SS,F' ,i2.2,'.', i2.2,',A1)')
      end



      subroutine eformc (fnum,iwidth,idec, outstr)
      implicit none
      real fnum
      integer iwidth,idec
      character outstr *(*)
      integer iexpon
      character fmtstr *(18)

      if (iwidth .le. 0) then ! iwidth=0: ohne Label
       outstr= char(0)
       return
      end if

      call esplit (fnum,iwidth,idec,iexpon)
      if ((idec .gt. iwidth-7) .or. (iwidth .gt. 99)) goto 200 ! Errorhandler
      write (unit=fmtstr,fmt=100, err=200) iwidth-idec-6,iwidth,iwidth-7
      if (len(outstr) .gt. iwidth) then
       write (unit= outstr, fmt=fmtstr, err=200) fnum,0 ! 0: End of String
      else
       write (unit= outstr, fmt=fmtstr, err=200) fnum ! evtl. ohne EoS?
      end if
      return

200   continue ! Error Handler
      outstr= '???'
      if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
      return

100   format ('(SS,' ,i2.2,'P,E' ,i2.2,'.', i2.2,',A1)')
      end

```

```fortran

      subroutine esplit (fnum,iwidth,idec,iexpon)
      implicit none
      real fnum
      integer iwidth,idec,iexpon
      real fabs
      include 'G2dAG2.fd'

      fabs= abs(fnum)
      if (fabs .ge. 1.) then
       iexpon= ifix( alog10(fabs)+1.000005) - iwidth+idec+6 ! 6: Vorz.-Pkt-Exp(4)
      else if (fabs .ge. 10./cinfin) then
       iexpon= alog10(fabs)
      else
       iexpon= -alog10(cinfin)
      end if
      return
      end



      subroutine expoutc (nbase,iexp, outstr)
      implicit none
      integer nbase,iexp, i, iL, nexp
      character outstr *(*), tmpstr *(4)
      include 'G2dAG2.fd'

      il= len(outstr)
      nexp= abs(iexp)

      if ( (cxyetyp(nbase).eq.2) .and. (il.gt. 5)
     1             .and. (mod(nexp,3) .eq. 0)
     2             .and. (iexp.ge.1)  .and. (iexp.le.9) ) then ! MMMs
       do 20 i=3,nexp,3
        outstr(i/3:i/3)= 'M'
20     continue
       outstr(nexp/3+1:)= char(39) // 'S' // char(0)

      else if ( (cxyetyp(nbase).eq.3) .and. (il.gt.17)
     1                    .and. (iexp.ge.1) .and. (iexp.le.6)) then ! TENS
       if (nexp .eq. 1) then
        outstr= 'TENS' // char(0)
       else if (nexp .eq. 2) then
        outstr= 'HUNDREDS' // char(0)
       else if (nexp .eq. 3) then
        outstr= 'THOUSANDS' // char(0)
       else if (nexp .eq. 4) then
        outstr= 'TEN THOUSANDS' // char(0)
       else if (nexp .eq. 5) then
        outstr= 'HUNDRED THOUSANDS' // char(0)
       else if (nexp .eq. 6) then
        outstr= 'MILLIONS' // char(0)
       end if
      else if( (cxyetyp(nbase).eq.4) ! 10000
     1    .and. (iexp.ge.1)  .and. (iexp.le.9)
     2                   .and. (il.ge.nexp+2)) then
       do 30 i=2,nexp+1
        outstr(i:i)= '0'
 30    continue
       outstr(1:1)= '1'
       outstr(nexp+2:)= char(0)

      else if (il .gt. 7) then ! Default: Superscript EXP
       if (iexp .ne. 1) then
        if (nexp .lt. 10) then
         i=1
        else
         i=2
        end if
        if (iexp .lt. 0) then
         i= i+1
        end if
        call iformc (real(iexp), i, tmpstr)
       else
        tmpstr= char(0) ! 10 wird ohne Exponenten 1 ausgegeben
       end if
       if (iexp .ne. 0) then
        if (cxytype(nbase) .ne. 2) then
         outstr(1:1)= 'x'
         i= 2
        else
         i= 1
        end if
        outstr(i:)= '10' // char(1) ! Index UP
        outstr(i+3:)= tmpstr ! char(0) wird bei IFORMC angehaengt
       else
        outstr(1:)= '1' // char(0) ! 1 wird nicht als 10**0 ausgegeben
```

```
         end if
        else ! outstr zu kurz
         outstr= '???'
        end if

        return
        end



        subroutine alfsetc (fnum, labtyp, string)
        implicit none
        integer inum, labtyp
        real fnum
        character *(*) string

        inum= fnum + .001 ! truncate real to integer
        if (labtyp .eq. 3) then ! Tage
         if ((inum .eq. 0) .or. (inum .eq. 7)) then
          string= 'MONDAY' // char(0)
         else if (inum .eq. 1) then
          string= 'TUESDAY' // char(0)
         else if (inum .eq. 2) then
          string= 'WEDNESDAY' // char(0)
         else if (inum .eq. 3) then
          string= 'THURSDAY' // char(0)
         else if (inum .eq. 4) then
          string= 'FRIDAY' // char(0)
         else if (inum .eq. 5) then
          string= 'SATURDAY' // char(0)
         else if (inum .eq. 6) then
          string= 'SUNDAY' // char(0)
         end if
        else if (labtyp .eq. 6) then ! Monate
         if (inum .eq. 1) then
          string= 'JANUARY' // char(0)
         else if (inum .eq. 2) then
          string= 'FEBRUARY' // char(0)
         else if (inum .eq. 3) then
          string= 'MARCH' // char(0)
         else if (inum .eq. 4) then
          string= 'APRIL' // char(0)
         else if (inum .eq. 5) then
          string= 'MAY' // char(0)
         else if (inum .eq. 6) then
          string= 'JUNE' // char(0)
         else if (inum .eq. 7) then
          string= 'JULY' // char(0)
         else if (inum .eq. 8) then
          string= 'AUGUST' // char(0)
         else if (inum .eq. 9) then
          string= 'SEPTEMBER' // char(0)
         else if (inum .eq. 10) then
          string= 'OCTOBER' // char(0)
         else if (inum .eq. 11) then
          string= 'NOVEMBER' // char(0)
         else if (inum .eq. 12) then
          string= 'DECEMBER' // char(0)
         end if
        end if
        return
        end



        subroutine notatec (ix,iy, string)
        implicit none
        integer ix, iy
        character *(*) string
        integer i, iv, is
        integer ISTRINGLEN

        call csize(i,iv)          ! nur iv benoetigt
        call movabs(ix,iy)

        is= 1
        do 100 i=1, istringlen(string)
         if (string(i:i) .lt. char(31) ) then
          if (i.gt.is) call toutstc (string(is:i-is))
          if (string(i:i) .eq. char(1)) call movrel (0, iv/2)  ! Hochindex
          if (string(i:i) .eq. char(2)) call movrel (0, -iv/2) ! Index
          is= i+1
         end if
  100   continue
        if (is .le. istringlen(string)) call toutstc (string(is:))
        return
        end
```

```



      subroutine vlablc (string)
C
C  Sollte in das TCS verlagert werden, um vertikale Schrift zu erzeugen
C
      implicit none
      character string*(*)
      integer i, icy, ix,iy
      integer ISTRINGLEN

      if (istringlen(string) .le. 0) return
      call csize (i,icy)
      call seeloc (ix,iy)
      do 100 i=1,istringlen(string)
       iy= iy-icy
       if (iy .lt. 0) return
       call movabs (ix,iy)
       call toutpt (ichar(string(i:i)))
100   continue
      return
      end



      subroutine justerc (string, iPosFlag, iOff)
      implicit none
      integer iPosFlag, iOff
      character string*(*)
      integer i, iLen, nCtrl
      integer ISTRINGLEN, LINWDT

      ilen= istringlen(string)
      nctrl= 0     ! Zaehlen der Ctrlcharacter
      do 100 i=1, ilen
       if (string(i:i) .lt. char(31) ) nctrl= nctrl+1
100   continue

      if (iposflag .lt. 0) then ! linksbuendig
       ioff= 0
      else ! rechtsbuendig und zentriert
       ioff= -linwdt((ilen-nctrl)*8-2)/8       ! rechtsbuendig
       if (iposflag.eq.0) ioff= ioff / 2        ! zentriert
      end if

      return
      end



      subroutine width (nbase)
      implicit none
      integer nbase
      integer labtyp
      include 'G2dAG2.fd'

      labtyp= cxylab(nbase)
      if(labtyp .eq. 1) labtyp= cxytype(nbase) ! LabTyp=1: = dataType

      if ((cxywdth(nbase).ne.0) .and. (labtyp.ne.1)) return ! Manuelle Vorgabe nichtlinear

      if (labtyp.le.1) then ! lineare Achsen und anwenderdefinierte Label
       call lwidth (nbase)

      else if (labtyp .eq. 2) then ! logarithmische Achsen
       if (cxyetyp(nbase) .le. 1) then ! 10 mit Exponent
        cxywdth(nbase)= 6
       else if (cxyetyp(nbase) .eq. 2) then ! M, MM...
        cxywdth(nbase)= int(alog10(abs(cxydmax(nbase)))/3. ) + 6
       else if (cxyetyp(nbase) .eq. 3) then ! Ausgeschriebene Worte
        cxywdth(nbase)= 20
        cxystep(nbase)= 1
        cxystag(nbase)= 2
       else if (cxyetyp(nbase) .eq. 4) then ! 1 mit 0
        cxywdth(nbase)= max(abs(alog10(abs(cxydmin(nbase)))),
     1                       abs(alog10(abs(cxydmin(nbase)))) ) + 2
       end if

      else if (labtyp .gt. 2) then ! Kalenderachsen
       if ((labtyp .eq. 3) .or. (labtyp .eq. 6)) then ! Tage oder Monate
        cxywdth(nbase)= 9
       else
        cxywdth(nbase)= 4
       end if
      end if

      return
      end



      subroutine lwidth (nbase)
      implicit none
      integer nbase
      integer iadj, most, least, isign,iwidth, idelta, ndec, iexp
      real xmax
      real ROUNDD
      include 'G2dAG2.fd'

      iadj= 0
      xmax= amax1(abs(cxydmin(nbase)),abs(cxydmax(nbase)))
      if (xmax .gt. 1.) then
       most= int(alog10(xmax) + 1.00005) ! Position Most Significant Digit
       iadj= 1
      else if (xmax .eq. 1.) then
       most= 0
      else
       most= int(alog10(xmax) - 0.00005)
      end if

      ndec= cxydec(nbase)
      if (cxydec(nbase) .ne. 0) then ! Anzahl Dezimalstellen vorgegeben
       least= -ndec ! Entspricht Position LeastSignificant Digit
      else
       least= cxylsig(nbase)
      end if

      if (cxydmin(nbase) .lt. 0.) then
       isign=1    ! 1 Buchstabe Vorzeichen
      else
       isign=0
      end if

      if ((most .lt. 0) .or. (least .ge. 0)) then
       iwidth= max0(1,most)- min0(0,least) + isign
       if (most .lt. 0) iwidth= iwidth+1 ! 1 Dezimalpunkt
       if ((iwidth .gt. 5 ) .and. (cxyetyp(nbase) .ge. 0)) then
        if (cxyetyp(nbase).eq.2) then
         iexp= int( roundd(real(most-iadj),3.))
        else
         iexp= int( roundd(real(most-iadj),1.))
        end if
        iwidth= most-least+isign+ 2
        ndec= max0(0,iexp-least+iadj)
       else
        ndec= max(0,-least)
        iexp= 0
       end if
      else
       iexp= 0
       ndec= max(0,-least)
       iwidth= most-least+isign+1
       if (most .eq. 0) iwidth= iwidth+1 ! Einbezug fuehrende Null
      end if

      if ((cxywdth(nbase) .ne. 0).and.(cxywdth(nbase).lt. iwidth)) then
       idelta= iwidth - cxywdth(nbase) - ndec
       if ((ndec .gt. 0) .and. (idelta .lt. 1) ) then
        ndec= max0(0,-idelta)
        iwidth= cxywdth(nbase)
       else
        iexp= iexp+idelta
        if(ndec .gt. 0) iexp=iexp-1
        iwidth= cxywdth(nbase)
        ndec=0
       end if
      end if

      cxywdth(nbase)= iwidth
      cxydec(nbase)= ndec
      cxyepon(nbase)= iexp
      return
      end



      subroutine remlab (nbase,iloc,labtyp,ix,iy)
      implicit none
      integer nbase, iloc, labtyp, ix, iy
      integer iyear1,iday1, iyear2,iday2
      integer iyear,imon,iday, ioff, iposflag
      character label *(25)
      include 'G2dAG2.fd'

      if (iabs(labtyp) .eq. 1) then ! lineare Daten
       if (cxyepon(nbase) .eq. 0) return ! kein Exponent
       call expoutc (nbase,cxyepon(nbase), label)
      else ! Kalenderdaten
       if ((labtyp .ge. 4) .and. (labtyp.ne.6)) then ! Wochen, Quartale, Jahre
        ioff= 4 ! Überlappung der Jahre vermeiden
       else
        ioff= 0
       end if
       call oubgc (iyear1,iday1, nint(cxydmin(nbase))+ioff)
       call oubgc (iyear2,iday2, nint(cxydmax(nbase))-ioff)
       if (iday2 .le. 1) iyear2=iyear2-1
       iday2=iday2-1
       call ydymd(iyear1,iday1,iyear,imon,iday)

       if (iabs(labtyp).eq. 3) then
        call iformc (real(iday), 2, label(1:2))
        label(3:3)= ' ' ! 'dd '
        call alfsetc (real(imon), 6, label(4:6)) ! labtyp 6= Monate, Laenge 3
        label(7:7)= ' ' ! 'dd mmm '
        call iformc (real(iyear), 4, label(7:10)) ! 'dd mm yyyy'
        label(11:11)= char(0) ! evtl. Labelende
        if (iyear1 .lt. iyear2) then ! bei Bedarf Start und Endjahr
         label(11:11)= '-' ! 'dd mm yyyy-'
         call ydymd(iyear2,iday2,iyear,imon,iday)
         call iformc (real(iday), 2, label(12:13)) ! 'dd'
         label(14:14)= ' ' ! 'dd mm yyyy-dd '
         call alfsetc (real(imon), 6, label(15:17)) ! 'dd mmm'
         label(18:18)= ' ' ! 'dd mm yyyy-dd mmm '
         call iformc (real(iyear), 4, label(19:22)) ! 'dd mm yyyy-'
         label(23:23)= char(0)
        end if
       else
        call iformc (real(iyear), 4, label(1:4)) ! 'yyyy'
        label(5:5)= char(0)
        if (iyear1 .lt. iyear2) then ! bei Bedarf Start und Endjahr
         label(5:5)= '-' ! 'yyyy-'
         call iformc (real(iyear2), 4, label(6:9)) ! 'yyyy-yyyy'
         label(10:10)= char(0)
        end if
       end if
      end if

      if ((nbase.eq.1) .or. (iloc.eq.1)) then ! X-Achse oder y Zentriert
       iposflag= 0
      else
       iposflag= isign(1,1-iloc)
      end if
      call justerc (label, iposflag, ioff)
      call notatec (ix+ioff, iy,label)
      return
      end



      subroutine spread (nbase)
      implicit none
      integer nbase
      integer ih, labtyp, iwidth, iMaxWid
      integer LINWDT
      include 'G2dAG2.fd'

      if (cxystag(nbase) .ne. 1) return

      labtyp= cxylab(nbase)
      if ((labtyp .eq. 1) .or. (labtyp .eq. 0)) labtyp= cxytype(nbase)

100   continue ! outer loop
       if (nbase .eq. 1) then ! x-Achse
        iwidth= linwdt(cxywdth(nbase))
       else
        call csize(ih, iwidth)
       end if

       imaxwid= iabs(cxysmax(nbase)-cxysmin(nbase))- 2*iwidth
       imaxwid= imaxwid* cxystep(nbase)* cxystag(nbase) / cxytics(nbase)

       cxystep(nbase)= 1
       cxystag(nbase)= 1

       if (iwidth .lt. imaxwid) return ! exit loop

       if (nbase .eq. 1) then ! x-Achse
        cxystag(nbase)= 2
       else
        cxystep(nbase)= cxystep(nbase) + 1
```

```
         end if

 110     continue ! inner loop
          if(iwidth .lt. imaxwid) return ! exit loop
          if(cxystep(nbase) .gt. cxytics(nbase)) return ! exit loop
         if (labtyp .ne. 3 .and. labtyp .ne. 6) then ! cycle inner loop
          cxystep(nbase)= cxystep(nbase)+1
          goto 110
        else ! cycle outer loop
         if (cxywdth(nbase) .eq. 3) return
         cxywdth(nbase)=3
         goto 100
        end if ! cycle until force exit
        end



C
C  Tabellensuche und Rundungen
C

        real function findge (val,tab,in)
        implicit none
        integer in
        real val, tab(1)

 100    if (tab(in) .lt. val) goto 110 ! while
         in= in-1
         goto 100
 110    continue ! endwhile

 120    continue ! repeat
         in= in+1
        if (tab(in) .lt. val) goto 120 ! end repeat
        findge= tab(in)
        return
        end



        real function findle (val,tab,in)
        implicit none
        integer in
        real val, tab(1)
        real valeps

        valeps= val+ 1.e-7 ! Vergleich um 0 ermoeglichen (Rechengenauigkeit!)

 100    if (tab(in) .le. valeps) goto 110 ! while
         in= in-1
         goto 100
 110    continue ! endwhile

 120    continue ! repeat
         in= in+1
        if (tab(in) .lt. valeps) goto 120 ! end repeat
        findle= tab(in-1)
        return
        end



        integer function locge (ival,itab,iN)
        implicit none
        integer ival, itab(1), in

 100    if (itab(in) .lt. ival) goto 110 ! while
         in= in-1
         goto 100
 110    continue ! endwhile

 120    continue ! repeat
         in= in+1
        if (itab(in) .lt. ival) goto 120 ! end repeat
        locge= itab(in)
        return
        end



        integer function locle (ival,itab,iN)
        implicit none
        integer ival, itab(1), in

 100    if (itab(in) .le. ival) goto 110 ! while
         in= in-1
         goto 100
```

```
110   continue ! endwhile

120   continue ! repeat
       in= in+1
      if (itab(in) .le. ival) goto 120 ! end repeat
      locle= itab(in-1)
      return
      end



      real function roundd (value,finterval)
      implicit none
      real value,finterval
      integer ifrac
      real frac

      frac= value/finterval
      ifrac= int(frac)
      if (real(ifrac) .gt. frac) ifrac= ifrac-1 ! Abrunden bei frac neg.
      roundd = real(ifrac) * finterval
      if (roundd .gt. value) roundd= value
      return
      end



      real function roundu (value,finterval)
      implicit none
      real value,finterval
      integer ifrac
      real frac

      frac= value/finterval
      ifrac= int(frac)
      if (real(ifrac) .lt. frac) ifrac= ifrac+1 ! Aufrunden bei frac pos.
      roundu = real(ifrac) * finterval
      if (roundu .lt. value) roundu= value
      return
      end



C
C  Generelle Manipulationen der Commonvariablen
C
      subroutine savcom (Array)
      implicit none
      integer array(1)
      include 'G2dAG2.fd'

      integer i
      integer arr(1)
      equivalence(arr(1),cline)
      do 10 i=1,g2dag2l
       array(i)= arr(i)
10    continue
      return
      end



      subroutine rescom (Array)
      implicit none
      integer array(1)
      include 'G2dAG2.fd'

      integer i
      integer arr(1)
      equivalence(arr(1),cline)
      do 10 i=1,g2dag2l
       arr(i)= array(i)
10    continue
      return
      end



      integer function iother (ipar)
      implicit none
      integer ipar

      if (mod(ipar,2) .eq. 1) then ! ungerader Parameter=x-Achse
       iother= ipar+1
      else
       iother= ipar-1
      end if
      return
      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

```
C> \file       AG2Holerith.for
C> \version    2.2
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C> \~german
C> \brief   Graph2D: obsolete AG2 Routinen
C> \~english
C> \brief   Graph2D: deprecated AG2 routines
C> \~
C>
C> \~german
C>     Unterprogramme zur Behandlung von Holerithvariablen und direkter
C>     Manipulation des Commonblocks
C>
C> \~english
C>     Compatibility routines dealing with holerith characters
C>     and direct manipulation of common variables.
C>
C
C
C  Tektronix Advanced Graphics 2 - Version 2.x
C
C     Optionale Unterprogramme
C

C
C Stringfunktionen fuer Holerithvariablen
C

      subroutine notate (ix,iy,lenchr,iarray)
      implicit none
      integer ix,iy,lenchr, iarray(lenchr)
      integer i
      character *(255) buf

      do 100 i=1,lenchr
       buf(i:i)= char(iarray(i))
100   continue
      call notatec (ix,iy,buf(1:lenchr))
      return
      end



      subroutine alfset (fnum,kwidth,labtyp,ilabel)
      implicit none
      integer kwidth,labtyp, ilabel(kwidth)
      real fnum
      integer i, buflen
      character *(255) buf
      integer ISTRINGLEN

      call alfsetc (fnum, labtyp, buf)
      buflen= istringlen(buf)
      do 100 i=1,kwidth
       if (i .le. buflen) then
        ilabel(i)= ichar(buf(i:i))
       else
        ilabel(i)= ichar(' ')
       end if
100   continue
      return
      end



      subroutine numset (fnum,iwidth,nbase,ilabel,ifill)
      implicit none
      integer iwidth,nbase,ilabel(iwidth),ifill
      real fnum
      integer i, iLeadFill
      character *(255) buf
      integer ISTRINGLEN

      call numsetc (fnum,iwidth,nbase, buf)
      ileadfill= max(0,iwidth-istringlen(buf))
      do 100 i=1,iwidth
       ilabel(ileadfill+i)= ichar(buf(i:i))
100   continue
      i=1 ! iLabel ist rechtsjustiert!
      if (i.gt.ileadfill) goto 110 ! while
       ilabel(i)= ifill
       i= i+1
110   continue ! endwhile
      return
      end



      subroutine expout (nbase,iexp,ilabel,nchars,ifill)
      implicit none
      integer nbase,iexp, nchars, ilabel(nchars), ifill
      integer i, iLeadFill
      character *(255) buf
      integer ISTRINGLEN

      call expoutc (nbase,iexp, buf(1:nchars))
      ileadfill= max(0,nchars-istringlen(buf))
      do 100 i=1,nchars
       ilabel(ileadfill+i)= ichar(buf(i:i))
100   continue
      i=1 ! iLabel ist rechtsjustiert!
      if (i.gt.ileadfill) goto 110 ! while
       ilabel(i)= ifill
       i= i+1
110   continue ! endwhile
      return
      end



      subroutine hstrin (iString)
      implicit none
      integer iString(2)
      call anstr (istring(1),istring(2))
      return
      end



      subroutine hlabel (iLen, iString)
      implicit none
      integer iLen, iString(iLen)
      call anstr (ilen, istring)
      return
      end



      subroutine vstrin (iarray)
      implicit none
      integer iarray(2)
      call vlabel (iarray(1),iarray(2))
      return
      end



      subroutine vlabel (iLen,iString)
      implicit none
      integer iLen, iString(iLen)
      integer i
      character *(255) buf
      integer ISTRINGLEN
      do 100 i=1, ilen
       buf(i:i)= char(istring(i))
100   continue
      call vlablc (buf(:ilen))
      return
      end



      subroutine juster (iLen,iString,iposflag,ifill,lenchr, ioff)
      implicit none
      integer iLen,iString(iLen), iposflag,ifill, lenchr, ioff
      integer i
      character *(255) buf

      lenchr= 0
      do 100 i=1, ilen
       if ( (i .gt. 1) .or. (istring(i) .ne. ifill) ) then ! Ueberlese Startfillchars
        lenchr= lenchr+1
        buf(lenchr:lenchr)= char(abs(istring(i))) ! Tek Index -1,-2 -> char(1),char(2)
       end if
100   continue
      call justerc (buf, iposflag, ioff)
      return
      end



      subroutine eform (fnum,iwidth,idec,ilabel,ifill)
      implicit none
      integer iwidth,idec, ilabel(iwidth), ifill
      real fnum
      integer i
      character *(255) buf

      call eformc (fnum,iwidth,idec, buf)
      do 100 i=1,iwidth
       ilabel(i)= ichar(buf(i:i))
100   continue
      return
      end



      subroutine fform (fnum,iwidth,idec,ilabel,ifill)
      implicit none
      integer iwidth,idec, ilabel(255), ifill
      real fnum
      integer i
      character *(255) buf

      call fformc (fnum,iwidth,idec, buf)
      do 100 i=1,iwidth
       ilabel(i)= ichar(buf(i:i))
100   continue
      return
      end



      subroutine fonly (fnum,iwidth,idec,ilabel,ifill)
      implicit none
      integer iwidth,idec, ilabel(iwidth), ifill
      real fnum
      integer i
      character *(255) buf

      call fonlyc (fnum,iwidth,idec, buf)
      do 100 i=1,iwidth
       ilabel(i)= ichar(buf(i:i))
100   continue
      return
      end



      subroutine iform (fnum,iwidth,ilabel,ifill)
      implicit none
      integer iwidth,idec, ilabel(iwidth), ifill
      real fnum
      integer i
      character *(255) buf

      call iformc (fnum,iwidth,idec, buf)
      do 100 i=1,iwidth
       ilabel(i)= ichar(buf(i:i))
100   continue
      return
      end



C
C  Direkte Manipulation des Commonblocks
C

      integer function ibasec (iPar)
      implicit none
      integer ipar

      ibasec= -1-ipar
      return
      end



      integer function ibasex (ipar)
      implicit none
      integer ipar

      ibasex= 1 + 2*ipar
      return
      end



      integer function ibasey (ipar)
      implicit none
      integer ipar

      ibasey= 2 + 2*ipar
      return
      end



      real function comget (ipar)
      implicit none
      integer ipar
      include 'G2dAG2.fd'

      integer iarr(1), iarr2(1)
      real arr(1), arr2(1)
      equivalence(iarr(1),cline), (iarr2(1),cxyneat)
      equivalence(arr(1),cline), (arr2(1),cxyneat)

      if ((ipar.lt.0) .and. (ipar.ge. -9))then
       if ((ipar .eq. -4) .or. (ipar .le. -8)) then
        comget= arr(-ipar)
       else
        comget= real(iarr(-ipar))
       end if
      else if ((ipar.gt.0) .and. (ipar.le.56)) then
       if ((ipar.le.22) .or. ((ipar .ge. 27).and.(ipar.le.52))) then
        comget= real(iarr2(ipar))
       else
        comget= arr2(ipar)
       end if
      end if
      return
      end



      subroutine comset (iPar,val)
      implicit none
      integer iPar
      real val
      include 'G2dAG2.fd'

      integer iarr(1), iarr2(1)
      real arr(1), arr2(1)
      equivalence(iarr(1),cline), (iarr2(1),cxyneat)
      equivalence(arr(1),cline), (arr2(1),cxyneat)

      if ((ipar.lt.0) .and. (ipar.ge. -9))then
       if ((ipar.eq.-4) .or. (ipar .le. -8)) then
        arr(-ipar)= val
       else
        iarr(-ipar)= int(val)
       end if
      else if ((ipar.gt.0) .and. (ipar.le.56)) then
       if ((ipar.le.22) .or. ((ipar .ge. 27).and.(ipar.le.52))) then
        iarr2(ipar)= int(val)
       else
        arr2(ipar)= val
       end if
      end if
      return
      end



      subroutine comdmp
      implicit none
      integer i
      character *80 buf
      include 'G2dAG2.fd'

      call erase
      call home

      write (unit= buf,fmt=600, err=200) (cxyneat(i),i=1,2), cline
600   format (1x,' 0:  cxneat(1)=',l14,', (2)=',l14,',   cline=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=601, err=200) (cxyzero(i),i=1,2), csymbl
601   format (1x,' 1: cxyzero(1)=',l14,', (2)=',l14,',  csymbl=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=602, err=200) (cxyloc(i),i=1,2), csteps
602   format (1x,' 2:  cxyloc(1)=',i14,', (2)=',i14,',  csteps=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=603, err=200) (cxylab(i),i=1,2), cinfin
603   format (1x,' 3:  cxylab(1)=',i14,', (2)=',i14,',  cinfin=',e14.7)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=604, err=200) (cxyden(i),i=1,2), cnpts
604   format (1x,' 4:  cxyden(1)=',i14,', (2)=',i14,',   cnpts=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=605, err=200) (cxytics(i),i=1,2), cstepl
605   format (1x,' 5: cxytics(1)=',i14,', (2)=',i14,',  cstepl=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=606, err=200) (cxylen(i),i=1,2), cnumbr
606   format (1x,' 6:  cxylen(1)=',i14,', (2)=',i14,',  cnumbr=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=607, err=200) (cxyfrm(i),i=1,2), csizes
607   format (1x,' 7:  cxyfrm(1)=',i14,', (2)=',i14,',  csizes=',e14.7)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=608, err=200) (cxymtcs(i),i=1,2), csizel
608   format (1x,' 8: cxymtcs(1)=',i14,', (2)=',i14,',  csizel=',e14.7)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=609, err=200) (cxymfrm(i),i=1,2)
609   format (1x,' 9: cxymfrm(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=610, err=200) (cxydec(i),i=1,2)
610   format (1x,'10:  cxydec(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=611, err=200) (cxydmin(i),i=1,2)
611   format (1x,'11: cxydmin(1)=',e14.7,',  (2)=',e14.7)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=612, err=200) (cxydmax(i),i=1,2)
612   format (1x,'12: cxydmax(1)=',e14.7,',  (2)=',e14.7)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=613, err=200) (cxysmin(i),i=1,2)
613   format (1x,'13: cxysmin(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=614, err=200) (cxysmax(i),i=1,2)
614   format (1x,'14: cxysmax(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=615, err=200) (cxytype(i),i=1,2)
615   format (1x,'15: cxytype(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=616, err=200) (cxylsig(i),i=1,2)
616   format (1x,'16: cxylsig(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=617, err=200) (cxywdth(i),i=1,2)
617   format (1x,'17: cxywdth(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=618, err=200) (cxyepon(i),i=1,2)
618   format (1x,'18: cxyepon(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=619, err=200) (cxystep(i),i=1,2)
619   format (1x,'19: cxystep(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=620, err=200) (cxystag(i),i=1,2)
620   format (1x,'20: cxystag(1)=',i14,', (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=621, err=200) (cxyetyp(i),i=1,2)
621   format (1x,'21: cxyetyp(1)=',i14,',  (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=622, err=200) (cxybeg(i),i=1,2)
622   format (1x,'22:  cxybeg(1)=',i14,',  (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=623, err=200) (cxyend(i),i=1,2)
623   format (1x,'23:  cxyend(1)=',i14,',  (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=624, err=200) (cxymbeg(i),i=1,2)
624   format (1x,'24: cxymbeg(1)=',i14,',  (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=625, err=200) (cxymend(i),i=1,2)
625   format (1x,'25: cxymend(1)=',i14,',  (2)=',i14)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=626, err=200) (cxyamin(i),i=1,2)
626   format (1x,'26: cxyamin(1)=',e14.7,',  (2)=',e14.7)
      call toutstc (buf)
      call newlin
      write (unit= buf,fmt=627, err=200) (cxyamax(i),i=1,2)
627   format (1x,'27: cxyamax(1)=',e14.7,',  (2)=',e14.7)
      call toutstc (buf)

      call graphicerror (11,char(0))
      call erase

200   continue
      return
      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

```
C> \file    AG2uline.for
C> \brief   Graph2D: Dummy User Routine
C
C  Tektronix Advanced Graphics 2 - Version 2.0
C
C     User Subroutinen
C


      subroutine uline (x,y,i)
      return
      end

```

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

```
C> \file    AG2users.for
C> \brief   Graph2D: Dummy User Routine
C
C  Tektronix Advanced Graphics 2 - Version 2.0
C
C     User Subroutinen
C

      subroutine users (x,y,i)
      return
      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

```
C> \file    AG2UsrSoftek.for
C> \brief   Graph2D: Dummy User Routine
C
C  Tektronix Advanced Graphics 2 - Version 2.0
C
C     User Subroutinen
C

      subroutine softek (isym)
      return
      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

```
C> \file    Fgraph.fd
C> \brief   DOS Port: Declarations OW graph.lib
C>
C> \~german
C> Konstanten und Funktionen der Watcom DOS Graphik-Library. Ersatz für das zum
C> Microsoft Fortan-Compiler gehörende INCLUDE-File, abgeleitet aus dem
C> Watcom-Headerfile graph.fi.
C>
C> \~english
C> Functions and constants of the Watcom DOS Graphic Library. Substitution for
C> the INCLUDE-file of the Microsoft Fortran Compiler, derived from the
C> Watcom Headerfile graph.fi.
C>
C> \~
C> \author  Dr.-Ing. Klaus Friedewald
C>
C> \~german
C> \note
C> Der Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
C> von $ und _. Deswegen sind $notruncate und $notstrict überflüssig.
C>
C> \note
C> Hex-Zahlen werden nicht durch \#ff sondern durch 'ff'x dargestellt.
C>
C> \note
C> Die OpenWatcom Library graph.lib ist nicht Bestandteil von Graph2Ddos.lib
C> und muss bei den Linkoptionen der Hauptprogramme aufgeführt werden:
C> -libr graph.
C> \~english
C> \note
C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
C> contain $ and _. That for $notruncate und $notstrict are superfluous.
C>
C> \note
C> Hexadecimal numbers are represented by 'ff'x instead of \#ff.
C>
C> \note
C> The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to
C> be linked to the main programs:
C> -libr graph.
C> \~
C>
C> \cond

      structure/videoconfig/      ! structure for getvideoconfig
        integer*2 numxpixels
        integer*2 numypixels
        integer*2 numtextcols
        integer*2 numtextrows
        integer*2 numcolors
        integer*2 bitsperpixel
        integer*2 numvideopages
        integer*2 mode
        integer*2 adapter
        integer*2 monitor
        integer*2 memory
      end structure

      structure/xycoord/          ! structure for pixel position
        integer*2 xcoord
        integer*2 ycoord
      end structure

      structure/rccoord/          ! structure for text position
        integer*2 row
        integer*2 col
      end structure

C Videomodes

      integer*2, $MAXRESMODE, $MAXCOLORMODE, $DEFAULTMODE,$TEXTBW40,
     1           $TEXTC40,$TEXTBW80,$TEXTC80, $MRES4COLOR,$MRESNOCOLOR,
     2           $HRESBW,$TEXTMONO,$HERCMONO, $MRES16COLOR,$HRES16COLOR,
     3           $ERESNOCOLOR,$ERESCOLOR, $VRES2COLOR,$VRES16COLOR,
     4           $MRES256COLOR,$ORESCOLOR
      parameter($maxresmode   =-3)    ! graphics mode with highest resolution
      parameter($maxcolormode =-2)    ! graphics mode with most colors
      parameter($defaultmode  =-1)    ! restore screen to original mode
      parameter($textbw40      =0)    ! 40 x 25 text, 16 grey
      parameter($textc40       =1)    ! 40 x 25 text, 16/8 color
      parameter($textbw80      =2)    ! 80 x 25 text, 16 grey
      parameter($textc80       =3)    ! 80 x 25 text, 16/8 color
      parameter($mres4color    =4)    ! 320 x 200, 4 color
      parameter($mresnocolor   =5)    ! 320 x 200, 4 grey
      parameter($hresbw        =6)    ! 640 x 200, BW
      parameter($textmono      =7)    ! 80 x 25 text, BW
      parameter($hercmono      =8)    ! 720 x 348, BW for HGC
      parameter($mres16color   =13)   ! 320 x 200, 16 color
      parameter($hres16color   =14)   ! 640 x 200, 16 color
      parameter($eresnocolor   =15)   ! 640 x 350, BW
      parameter($erescolor     =16)   ! 640 x 350, 4 or 16 color
      parameter($vres2color    =17)   ! 640 x 480, BW
      parameter($vres16color   =18)   ! 640 x 480, 16 color
      parameter($mres256color  =19)   ! 320 x 200, 256 color
      parameter($orescolor     =64)   ! 640 x 400, 1 of 16 colors (Olivetti)

      integer*4 $MDPA,$CGA,$EGA,$MCGA,$VGA,$HGC,$OCGA,$OEGA,$OVGA
      parameter($mdpa     ='0001'x)    ! Monochrome Display Adapter (MDPA)
      parameter($cga      ='0002'x)    ! Color Graphics Adapter     (CGA)
      parameter($ega      ='0004'x)    ! Enhanced Graphics Adapter  (EGA)
      parameter($vga      ='0008'x)    ! Video Graphics Array       (VGA)
      parameter($mcga     ='0010'x)    ! MultiColor Graphics Array  (MCGA)
      parameter($hgc      ='0020'x)    ! Hercules Graphics Card     (HGC)
      parameter($ocga     ='0042'x)    ! Olivetti Color Graphics Adapter (OCGA)
      parameter($oega     ='0044'x)    ! Olivetti Enhanced Graphics Adapter (OEGA)
      parameter($ovga     ='0048'x)    ! Olivetti Video Graphics Array (OVGA)

      integer*4 $MONO,$COLOR,$ENHCOLOR,$ANALOGMONO,$ANALOGCOLOR,$ANALOG
      parameter($mono       ='0001'x)     ! Monochrome
      parameter($color      ='0002'x)     ! Color (or Enhanced emulating color)
      parameter($enhcolor   ='0004'x)     ! Enhanced Color
      parameter($analogmono ='0008'x)     ! Analog Monochrome only
      parameter($analogcolor='0010'x)     ! Analog Color only
      parameter($analog     ='0018'x)     ! Analog

C Plotting Action

      integer*2 $GBORDER,$GFILLINTERIOR,
     1          $GCLEARSCREEN, $GVIEWPORT,$GWINDOW

      parameter($gborder       =2)      ! draw outline only
      parameter($gfillinterior =3)      ! fill using current fill mask

      parameter($gclearscreen=0)
      parameter($gviewport   =1)
      parameter($gwindow     =2)

      integer*4 $GCURSOROFF,$GCURSORON,$GWRAPOFF,$GWRAPON
      parameter($gcursoroff=0)
      parameter($gcursoron =1)

      parameter($gwrapoff  =0)
      parameter($gwrapon   =1)

      integer*4 $GSCROLLUP, $GSCROLLDOWN
      parameter($gscrollup   =1)
      parameter($gscrolldown =-1)

      integer*4 $MAXTEXTROWS
      parameter($maxtextrows =-1)

      integer*4 $GPSET,$GPRESET,$GAND,$GOR,$GXOR
      parameter($gpset         =3)
      parameter($gpreset       =2)
      parameter($gand          =1)
      parameter($gor           =0)
      parameter($gxor          =4)

      integer*4 $BLACK,$BLUE,$GREEN,$CYAN,$RED,$MAGENTA,$BROWN,
     1          $WHITE,$GRAY, $LIGHTBLUE,$LIGHTGREEN,$LIGHTCYAN,
     2          $LIGHTRED,$LIGHTMAGENTA, $LIGHTYELLOW,$BRIGHTWHITE
      parameter($black         ='000000'x)
      parameter($blue          ='2a0000'x)
      parameter($green         ='002a00'x)
      parameter($cyan          ='2a2a00'x)
      parameter($red           ='00002a'x)
      parameter($magenta       ='2a002a'x)
      parameter($brown         ='00152a'x)
      parameter($white         ='2a2a2a'x)
      parameter($gray          ='151515'x)
      parameter($lightblue     ='3F1515'x)
      parameter($lightgreen    ='153f15'x)
      parameter($lightcyan     ='3f3f15'x)
      parameter($lightred      ='15153f'x)
      parameter($lightmagenta  ='3f153f'x)
      parameter($lightyellow   ='153f3f'x)
      parameter($brightwhite   ='3f3f3f'x)

      integer*4 $MODEFOFF,$MODEFOFFTOON,$MODEFOFFTOHI,$MODEFONTOOFF,
     1          $MODEFON,$MODEFONTOHI,$MODEFHITOOFF,$MODEFHITOON,
     2          $MODEFHI
      parameter($modefoff      =0)
      parameter($modefofftoon  =1)
      parameter($modefofftohi  =2)
      parameter($modefontooff  =3)
      parameter($modefon       =4)
      parameter($modefontohi   =5)
      parameter($modefhitooff  =6)
      parameter($modefhitoon   =7)
      parameter($modefhi       =8)

      integer*4 $MODE7OFF,$MODE7ON,$MODE7HI
      parameter($mode7off      =0)
      parameter($mode7on       =1)
      parameter($mode7hi       =2)

C external functions

      external setvideomode
      integer*2 setvideomode

      external setvideomoderows
      integer*2 setvideomoderows

      external setactivepage
      integer*2 setactivepage

      external setvisualpage
      integer*2 setvisualpage

      external getactivepage
      integer*2 getactivepage

      external getvisualpage
      integer*2 getvisualpage

      external getvideoconfig
      external setvieworg
      external getviewcoord
      external getphyscoord
      external setcliprgn
      external setviewport
      external clearscreen
      external moveto
      external getcurrentposition

      external lineto
      integer*2 lineto

      external rectangle
      integer*2 rectangle

      external ellipse
      integer*2 ellipse

      external arc
      integer*2 arc

      external pie
      integer*2 pie

      external setpixel
      integer*2 setpixel

      external getpixel
      integer*2 getpixel

      external floodfill
      integer*2 floodfill

      external setcolor
      integer*2 setcolor

      external getcolor
      integer*2 getcolor

      external setlinestyle

      external getlinestyle
      integer*2 getlinestyle

      external setfillmask
      external getfillmask

      external setbkcolor
      integer*4 setbkcolor

      external getbkcolor
      integer*4 getbkcolor

      external remappalette
      integer*4 remappalette

      external remapallpalette
      integer*2 remapallpalette

      external selectpalette
      integer*2 selectpalette

      external settextrows
      integer*2 settextrows

      external settextwindow
      external scrolltextwindow
      external outtext

      external wrapon
      integer*2 wrapon

      external displaycursor
      integer*2 displaycursor

      external settextcursor
      integer*2 settextcursor

      external gettextcursor
      integer*2 gettextcursor

      external settextposition
      external gettextposition

      external settextcolor
      integer*2 settextcolor

      external gettextcolor
      integer*2 gettextcolor

      external getimage
      external putimage

      external imagesize
      integer*4 imagesize



      structure/wxycoord/       ! window coordinates
        double precision wx
        double precision wy
      end structure

      external setwindow
      integer*2 setwindow

      external getwindowcoord
      external getviewcoord_w
      external getcurrentposition_w


      external arc_w
      integer*2 arc_w

      external ellipse_w
      integer*2 ellipse_w

      external floodfill_w
      integer*2 floodfill_w

      external getpixel_w
      integer*2 getpixel_w

      external lineto_w
      integer*2 lineto_w

      external moveto_w

      external pie_w
      integer*2 pie_w

      external rectangle_w
      integer*2 rectangle_w

      external setpixel_w
      integer*2 setpixel_w

      external getimage_w

      external imagesize_w
      integer*2 imagesize_w

      external putimage_w

      structure/fontinfo/
        integer*2 type          ! b0 set = vector,clear = bit map
        integer*2 ascent        ! pix dist from top to baseline
        integer*2 pixwidth      ! character width in pixels, 0=prop
        integer*2 pixheight     ! character height in pixels
        integer*2 avgwidth      ! average character width in pixels
        character*81 filename   ! file name including path
        character*32 facename   ! font name
      end structure


      integer*2 $NO_SPACE, $FIXED_SPACE, $PROP_SPACE
      parameter($no_space    = 0)
      parameter($fixed_space = 1)
      parameter($prop_space  = 2)

      integer*2 $NO_FONT_MAP, $VECTOR_MAP, $BIT_MAP
      parameter($no_font_map = 0)
      parameter($vector_map  = 1)
      parameter($bit_map     = 2)

      external registerfonts
      integer*2 registerfonts

      external unregisterfonts

      external setfont
      integer*2 setfont

      external getfontinfo
      integer*2 getfontinfo

      external outgtext

      external getgtextextent
      integer*2 getgtextextent
C
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

```
C> \file    Fgraph.fi
C> \brief   DOS Port: Interface OW graph.lib
C>
C> \~german
C> Interfacedeklaration der Watcom DOS Graphik-Library. Ersatz für das zum
C> Microsoft Fortran-Compiler gehörende INCLUDE-File, abgeleitet aus dem
C> Watcom-Headerfile graphapi.fi.
C>
C> \~english
C> Interface definition for the Watcom DOS Graphic Library. Substitutes
C> the INCLUDE-file of the Microsoft Fortran Compiler, derived from the
C> Watcom headerfile graphapi.fi.
C>
C> \~
C> \author  Dr.-Ing. Klaus Friedewald
C>
C> \~german
C> \note
C> Der Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
C> von $ und _. Deswegen sind $notruncate und $notstrict überflüssig.
C>
C> \note
C> Die OpenWatcom Library graph.lib ist nicht Bestandteil von Graph2Ddos.lib
C> und muss bei den Linkoptionen der Hauptprogramme aufgeführt werden:
C> -libr graph.
C> \~english
C> \note
C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
C> contain $ and _. That for $notruncate und $notstrict are superfluous.
C>
C> \note
C> The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to
C> be linked to the main programs:
C> -libr graph.
C> \~
C>


c$pragma aux arc "_arc_" parm (VALUE*2)

c$pragma aux arc_w "_arc_w_" parm (VALUE*8)

c$pragma aux clearscreen "_clearscreen_" parm (VALUE*2)

c$pragma aux displaycursor "_displaycursor_" parm (VALUE*2)

c$pragma aux ellipse  "_ellipse_" parm (VALUE*2)

c$pragma aux ellipse_w "_ellipse_w_" parm (VALUE*2, VALUE*8)

c$pragma aux floodfill  "_floodfill_" parm (VALUE*2)

c$pragma aux floodfill_w "_floodfill_w_" parm (VALUE*8, VALUE*8, VALUE*2)

c$pragma aux getactivepage  "_getactivepage_"

c$pragma aux getbkcolor  "_getbkcolor_"

c$pragma aux getcolor "_getcolor_"

c$pragma aux getcurrentposition "_getcurrentposition_" parm (REFERENCE FAR)

c$pragma aux getcurrentposition_w "_getcurrentposition_w_" parm (REFERENCE FAR)

c$pragma aux getfillmask "_getfillmask_" parm (REFERENCE FAR)

c$pragma aux getimage "_getimage_" parm (VALUE*2,VALUE*2,VALUE*2,VALUE*2, \
c REFERENCE FAR)

c$pragma aux getimage_w "_getimage_w_" parm (VALUE*8,VALUE*8,VALUE*8, \
c  VALUE*8,REFERENCE FAR)

c$pragma aux  getlinestyle "_getlinestyle_"

c$pragma aux getphyscoord "_getphyscoord_" parm (VALUE*2,VALUE*2, \
c  REFERENCE FAR)

c$pragma aux getpixel "_getpixel_" parm (VALUE*2)

c$pragma aux getpixel_w "_getpixel_w_" parm (VALUE*8)

c$pragma aux gettextcolor "_gettextcolor_"

c$pragma aux gettextcursor "_gettextcursor_"

```

```
c$pragma aux gettextposition "_gettextposition_" parm (REFERENCE FAR)

c$pragma aux getvideoconfig "_getvideoconfig_" parm (REFERENCE FAR)

c$pragma aux getviewcoord "_getviewcoord_" parm (VALUE*2,VALUE*2, \
c  REFERENCE FAR)

c$pragma aux getviewcoord_w "_getviewcoord_w_" parm (VALUE*8,VALUE*8, \
c  REFERENCE FAR)

c$pragma aux getvisualpage "_getvisualpage_"

c$pragma aux getwindowcoord "_getwindowcoord_" parm (VALUE*2,VALUE*2, \
c  REFERENCE FAR)

c$pragma aux imagesize "_imagesize_" parm (VALUE*2)

c$pragma aux imagesize_w "_imagesize_w_" parm (VALUE*8)

c$pragma aux lineto "_lineto_" parm (VALUE*2)

c$pragma aux lineto_w "_lineto_w_" parm (VALUE*8)

c$pragma aux moveto "_moveto_" parm (VALUE*2,VALUE*2,REFERENCE FAR)

c$pragma aux moveto_w "_moveto_w_" parm (VALUE*8,VALUE*8,REFERENCE FAR)

c$pragma aux _outtext "_outtext_" parm (DATA_REFERENCE FAR)

c$pragma aux pie "_pie_" parm (VALUE*2)

c$pragma aux pie_w "_pie_w_" parm (VALUE*2,VALUE*8)

c$pragma aux putimage "_putimage_" parm (VALUE*2,VALUE*2,REFERENCE FAR,VALUE*2)

c$pragma aux putimage_w "_putimage_w_" parm (VALUE*8,VALUE*8, \
c  REFERENCE FAR,VALUE*2)

c$pragma aux rectangle "_rectangle_" parm (VALUE*2)

c$pragma aux rectangle_w "_rectangle_w_" parm (VALUE*2,VALUE*8)

c$pragma aux remappalette "_remappalette_" parm (VALUE*2,VALUE*4)

c$pragma aux remapallpalette "_remapallpalette_" parm (VALUE*4)

c$pragma aux scrolltextwindow "_scrolltextwindow_" parm (VALUE*2)

c$pragma aux selectpalette "_selectpalette_" parm (VALUE*2)

c$pragma aux setactivepage "_setactivepage_" parm (VALUE*2)

c$pragma aux setbkcolor "_setbkcolor_" parm (VALUE*4)

c$pragma aux setcliprgn "_setcliprgn_" parm (VALUE*2)

c$pragma aux setcolor "_setcolor_" parm (VALUE*2)

c$pragma aux setfillmask "_setfillmask_" parm (REFERENCE FAR)

c$pragma aux setlinestyle "_setlinestyle_" parm (VALUE*2)

c$pragma aux setpixel "_setpixel_" parm (VALUE*2)

c$pragma aux setpixel_w"_setpixel_w_" parm (VALUE*8)

c$pragma aux settextcolor "_settextcolor_" parm (VALUE*2)

c$pragma aux settextcursor "_settextcursor_" parm (VALUE*2)

c$pragma aux settextposition "_settextposition_" parm (VALUE*2,VALUE*2, \
c  REFERENCE FAR)

c$pragma aux settextrows "_settextrows_" parm (VALUE*2)

c$pragma aux settextwindow "_settextwindow_" parm (VALUE*2)

c$pragma aux setvideomode "_setvideomode_" parm (VALUE*2)

c$pragma aux setvideomoderows "_setvideomoderows_" parm (VALUE*2)

c$pragma aux setvieworg "_setvieworg_" parm (VALUE*2, VALUE*2,REFERENCE FAR)

c$pragma aux setviewport "_setviewport_" parm (VALUE*2)

c$pragma aux setvisualpage "_setvisualpage_" parm (VALUE*2)

```

```
c$pragma aux setwindow "_setwindow_" parm (VALUE*2,VALUE*8)

c$pragma aux wrapon "_wrapon_" parm (VALUE*2)


c$pragma aux getfontinfo "_getfontinfo_" parm (REFERENCE FAR)

c$pragma aux getgtextextent "_getgtextextent_" parm (DATA_REFERENCE FAR)

c$pragma aux outgtext "_outgtext_" parm (DATA_REFERENCE FAR)

c$pragma aux registerfonts "_registerfonts_" parm (DATA_REFERENCE FAR)

c$pragma aux setfont "_setfont_" parm (DATA_REFERENCE FAR)

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

```
C> \file       G2dAG2.fd
C> \brief      Graph2D: AG2 Common Block G2dAG2
C> \version    2.0
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C
C  Da die folgende Definition kein Bestandteil eines Moduls
C  ist versagt der DOXYGEN-Parser bei der Kombination von
C  COMMON und integer. Workaraound: \\cond ... \\endcond
C> \cond

C Common Block G2dAG2, Version 2.0 für AG2
C     Die Funktion der Variablen entspricht dem Tektronix AG2 User-Manual,
C     jedoch sind die achsenbezogenen Variablen in einem Feld zusammenge-
C     fasst. Die x-Achse wird durch Index=1, y durch Index=2 beschrieben.
C
      integer     cline,csymbl,csteps ! ibase+ 0..2
      real        cinfin ! 3
      integer     cnpts,cstepl,cnumbr ! 4..6
      real        csizes,csizel ! 7,8

      logical     cxyneat(2),cxyzero(2) ! nbase+ 0, 1
      integer     cxyloc(2),cxylab(2),cxyden(2),cxytics(2) ! nbase+ 2..5
      integer     cxylen(2),cxyfrm(2),cxymtcs(2),cxymfrm(2),cxydec(2) ! 6..10
      real        cxydmin(2),cxydmax(2) ! 11,12
      integer     cxysmin(2),cxysmax(2),cxytype(2) ! 13..15
      integer     cxylsig(2),cxywdth(2),cxyepon(2) ! 16..18
      integer     cxystep(2),cxystag(2),cxyetyp(2) ! 19..21
      integer     cxybeg(2),cxyend(2),cxymbeg(2),cxymend(2) ! 22..25
      real        cxyamin(2),cxyamax(2) ! 26,27

      common /g2dag2/
C     & extent,cvectr,xvectr,yvectr,
C     & xtentc,xtentx,xtenty,
C
     & cline,csymbl,csteps,
     & cinfin,
     & cnpts,cstepl,cnumbr,csizes,csizel,
C
     & cxyneat,cxyzero,cxyloc,cxylab,cxyden,cxytics,
     & cxylen,cxyfrm,cxymtcs,cxymfrm,cxydec,
     & cxydmin,cxydmax,cxysmin,cxysmax,cxytype,
     & cxylsig,cxywdth,cxyepon,cxystep,cxystag,cxyetyp,
     & cxybeg,cxyend,cxymbeg,cxymend,cxyamin,cxyamax
C
C     & reserv(8)
      save /g2dag2/

      integer G2dAG2L         ! Benoetigt von SAVCOM, RESCOM
      parameter(g2dag21=65)  ! integer, real und logical gleich lang!
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'
```

```
      structure /bitmapfileheader/
            integer*2   DatKennung  ! = $4d42
            integer*4   DatSize     ! Bilddateigroesse in Byte
            integer*2   Reserved1
            integer*2   Reserved2
            integer*4   GraphDatDst ! Entfernung BITMAPFILEHEADER zu Graphikdaten (Byte)
      end structure
      structure /bitmapinfoheader/
            integer*4   BMpInfHdSiz ! Größe Bitmapinfoheader in Byte
            integer*4   PicWidth    ! Bildbreite Pixel, abgespeicherte Bytes durch 4 teilbar!
            integer*4   PicHeight   ! Bildhöhe in Pixel
            integer*2   iLayer      ! = 1
            integer*2   iBitPix     ! Bits per Pixel (1,4,8,24)
            integer*4   Kompr       ! Komprimierung =0(ohne),1(RLE8),2(RLE4)
            integer*4   PicSiz      ! Bildgroesse in Byte
            integer*4   HorPixDen   ! Horizontale Auflösung Pixel/ Meter
            integer*4   VerPixDen   ! Vertikale Auflösung Pixel/ Meter
            integer*4   iCol        ! Anzahl benutzte Farben
            integer*4   iVIPCol     ! Anzahl wichtige Farben =0(alle)
      end structure
      structure /rgbquad/
            integer*1   Blue
            integer*1   Green
            integer*1   Red
            integer*1   Reserved    ! =0
      end structure
      structure /fileheader/
            record /bitmapfileheader/  bfh
            record /bitmapinfoheader/  bih
            record /rgbquad/           palette(16)
      end structure

      record /fileheader/ filhead

      integer iWrtBuf
      parameter(iwrtbuf=650)
      integer*1 Buf(iWrtBuf)             ! > 2* (VGA-Auflösung/2)
      equivalence(buf,filhead)


      integer nByteRow
      integer iPtr, iPathlen
      integer*2 iHandle, ierr
      character*10 FilNam, Path*80

      call graphicerror (10,' ') ! Hardcopy in progress
c
c  Initialisierung Fileheader
c
      nbyterow=(kscrx+7-mod(kscrx-1,8))/2 ! Byte pro Zeile durch 4 teilbar
      if (2*nbyterow.gt.iwrtbuf) then
       call graphicerror (8, ' ') ! Hardcopy: Write Buffer Overflow
      end if

      filhead.bfh.datkennung= 19778 ! = 4d42h

      filhead.bfh.reserved1= 0
      filhead.bfh.reserved2= 0

      filhead.bfh.graphdatdst= 118 ! = 76h
      filhead.bfh.datsize=nbyterow*(kscry+1) + filhead.bfh.graphdatdst

      filhead.bih.bmpinfhdsiz= 40 ! = 28h
      filhead.bih.picwidth= kscrx+1
      filhead.bih.picheight= kscry+1

      filhead.bih.ilayer= 1
      filhead.bih.ibitpix=4         ! Auch bei Monochrom???
      filhead.bih.kompr= 0
      filhead.bih.picsiz= 0         ! nicht verwendet
      filhead.bih.horpixden= 0
      filhead.bih.verpixden= 0
      filhead.bih.icol= 0
      filhead.bih.ivipcol= 0

      filhead.palette(1).red= 0
      filhead.palette(1).green= 0
      filhead.palette(1).blue= 0

      filhead.palette(2).red= 0
      filhead.palette(2).green= 0
      filhead.palette(2).blue= 160

      filhead.palette(3).red= 0
      filhead.palette(3).green= 160
      filhead.palette(3).blue= 0

      filhead.palette(4).red= 0
      filhead.palette(4).green= 160
      filhead.palette(4).blue=160

      filhead.palette(5).red= 160
      filhead.palette(5).green= 0
      filhead.palette(5).blue= 0

      filhead.palette(6).red= 160
      filhead.palette(6).green= 0
      filhead.palette(6).blue= 160

      filhead.palette(7).red= 160
      filhead.palette(7).green= 80
      filhead.palette(7).blue= 0

      filhead.palette(8).red= 160
      filhead.palette(8).green= 160
      filhead.palette(8).blue= 160

      filhead.palette(9).red= 80
      filhead.palette(9).green= 80
      filhead.palette(9).blue= 80

      filhead.palette(10).red= 80
      filhead.palette(10).green= 80
      filhead.palette(10).blue= 240

      filhead.palette(11).red= 80
      filhead.palette(11).green= 240
      filhead.palette(11).blue= 80

      filhead.palette(12).red= 80
      filhead.palette(12).green= 240
      filhead.palette(12).blue= 240

      filhead.palette(13).red= 240
      filhead.palette(13).green= 80
      filhead.palette(13).blue= 80

      filhead.palette(14).red= 240
      filhead.palette(14).green= 80
      filhead.palette(14).blue= 240

      filhead.palette(15).red= 240
      filhead.palette(15).green= 240
      filhead.palette(15).blue= 80

      filhead.palette(16).red= 240
      filhead.palette(16).green= 240
      filhead.palette(16).blue= 240

      do 3 i=1,16
 3     filhead.palette(i).reserved= 0
c
c Create Filename and open
c
      path= 'SPL='//char(0)
      call getenv (path, len(path))
      ipathlen=istringlen(path)

      i=0
5     continue
       i= i+1
       write (filnam,fmt=300) i
       if (ipathlen.gt.0) then
        call openbytfil(ierr,ihandle,
     1                 path(:ipathlen)//'\'//filnam//char(0))
       else
        call openbytfil(ierr,ihandle, filnam//char(0))
       end if
      if (ierr.eq.80) goto 5  ! File exists - increase FilNam
      if (ierr.ne.0) call graphicerror (6, ' ') ! Hardcopy: Error during OPEN
c
c Zeilenweises Auslesen Bildschirmspeicher, Puffern und Fileausgabe
c
      iptr= filhead.bfh.graphdatdst +1

      do 20 iy=kscry,0,-1      ! oder 1?
       ix=0
 10    continue                     ! repeat
        buf(iptr)= ishl(getpixel(ix,iy),4)
        ix= ix+1
        if(ix.le.kscrx)buf(iptr)=buf(iptr).or.(getpixel(ix,iy).and.15)
        iptr= iptr+1
        ix=ix+1
       if (ix.le.kscrx) goto 10
```

```
          ix=ix                        ! Anzahl belegter Halfbytes
   15     if (ix.lt.2*nbyterow) then   ! do while
           buf(iptr)= 0
           iptr= iptr+1
           ix=ix+2
           goto 15
          end if                       ! end while
          call writebuf (ihandle, buf(1),iptr, 256)
   20    continue
c
c Empty Buffer and Close File
c
         call wrtbytfil (ierr, ihandle, buf(1), iptr)
         if (ierr.ne.0) call graphicerror (7, ' ') ! Hardcopy: Error during WRITE

         call closebytfil (ihandle)
         call statst (' ')
         return

  300    format ('HDC',i3.3,'.BMP')
         end



         subroutine writebuf (iHandle, Buf, iPtr, iWrite)
         integer*1 Buf(1)
         integer iPtr, iWrite
         integer*2 iHandle
         integer*2 iErr
   10    continue
          if (iptr.le.iwrite) return
          call wrtbytfil (ierr, ihandle, buf(1), iwrite)
          if (ierr.ne.0) call graphicerror (7, ' ') ! Hardcopy: Error during WRITE
          call lib_movc3 (iptr-iwrite,buf(iwrite+1), buf(1))
          iptr= iptr-iwrite
         goto 10
         end

```

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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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
```

```
      new= new1 // char(0)          ! => retten!

      temp= source(1:istringlen(source)) // char(0) ! evtl. Ueberlappung!
      destination= temp
      inext= index( destination(:istringlen(destination)),
     1                                        old(:istringlen(old)) )
      do while (inext.gt.0)
       if (inext.eq.1) then
        temp= destination
        if (new.eq.char(0)) then
         destination= temp(istringlen(old)+1:)
        else
         destination= new(:istringlen(new)) // temp(istringlen(old)+1:)
        end if
       else
        temp= destination(1:inext-1)
        templen= inext-1
        if (new.ne.char(0)) then
         temp= temp(1:templen)//new
         templen= templen+istringlen(new)
        end if
        if (inext+istringlen(old).lt.len(destination)) then
         temp= temp(1:templen)//destination(inext+istringlen(old):)
        end if
        destination= temp
       end if
       inext2= inext+istringlen(new)
       if (inext2.lt.len(destination)) then
        inext2= index(destination(inext2:), old(:istringlen(old)) )
       else
        inext2=0
       end if
       if (inext2.gt.0) then
        inext= inext+istringlen(new)+inext2-1
       else
        inext=0
       end if
      end do
      return
      end



      function istringlen (String)
C
C Ermittelt die Stringlänge bei durch char(0) abgeschlossenen STRINGs.
C Falls kein char(0) vorhanden ist, wird die Gesamtlänge übergeben.
C
      implicit none
      character *(*) string
      integer istringlen, i

      i= index(string,char(0))-1
      if (i.ge.0) then
       istringlen=i
      else
       istringlen= len(string)
      end if
      return
      end



      character*(*) function printstring (String)
C
C  Kopiert STRING in einen variabel langen PRINTSTRING. Hierdurch wird
C  der Ausdruck von Nullstrings (Fortran-Fehler!) vermieden.
C
      implicit none
      character string *(*)
      integer istringlen

      if (istringlen(string).gt.0) then
       printstring= string(1:istringlen(string))
      else
       printstring= ' '
      end if
      return
      end



      integer function itrimlen (string)
C
C  Bestimmt die Länge des Strings ohne angehängte Leerzeichen.
C  Bei Bedarf wird ein Char(0) angehaengt. Es darf in Ftn77 nie ein
C  Nullstring erzeugt werden, da sonst die RTL-Library abstuerzt. Deswegen
```

```
C  ist der kleinste erzeugte String ein Blank ' '.
C
      implicit none
      character *(*) string
      integer i, istringlen

      i=istringlen(string) +1

 10   continue
       i= i-1
       if (i.ge.1) then
        if (string(i:i).eq.' ') goto 10
      end if
      itrimlen=i
      if ((i.lt.len(string)).and.(len(string).gt.1)) then
       string(i+1:i+1)= char(0) ! .gt.1: Achtung, nie Nullstring erzeugen!
      end if
      return
      end
```

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

```
C> \file       TCS.for
C> \brief      TCS: Tektronix Plot 10 Emulation
C> \version    4.1
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C> \~german
C> Systemübergreifende TCS-Routinen
C> \~english
C> System independent subroutines
C> \~
C
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      26.07.23 Version 5.0:
C               Einheitliche Version CPM/DOS/Windows/SDL2/wX
C
C      27.11.20 Version 4.0:
C               Einheitliche Version CPM/DOS/Windows/SDL2
C
C      17.08.20 Version 3.2
C               Harmonisierung der Verwendung des Commonblocks TKTRNX
C               Variable KHOMEY wird jetzt (analog alter DOS-Version) verwendet.
C               Da KHOMEY nicht in der CP/M Version vorhanden ist, muss ab dieser
C               Version fuer eine Complilation unter CP/M die entsprechende Zeile
C               in der SUBROUTINE HOME geändert werden.
C
C      13.11.17 Version 3.1
C               Anpassung an OpenWatcom 2.0
C               Bugfix: Unterscheidung Aufrufe ueber windowsx.h (win16) und GDI (win32)
C                - SelectPen -> SelectObject
C                - DeletePen -> DeleteObject
C                - DeleteBrush -> DeleteObject
C                - GetStockBrush -> GetStockObject
C                - DeleteRgn -> DeleteObject
C                - SelectFont -> SelectObject
C                - DeleteFont -> DeleteObject
C
C      27.03.13 Version 3.0
C               Anpassung an Windows 7 und OpenWatcom 1.9
C               Anpassung an gfortran anstelle von g77 der GCC
C
C      22.12.05 Version 2.19
C               Elimination berechnetes GOTO in LOGTRN
C
C      18.10.05 Version 2.18
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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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)
      call rel2ab (x,y,xabs,yabs)
      call movea (xabs,yabs)
      return
      end



      subroutine pointr (X,Y)
      call rel2ab (x,y,xabs,yabs)
      call pointa (xabs,yabs)
      return
      end



      subroutine dashr (X,Y, iL)
      call rel2ab (x,y,xabs,yabs)
      call dasha (xabs,yabs, il)
      return
      end



      subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
      include 'Tktrnx.fd'
      call seeloc (ix,iy)
      call revcot (ix,iy,xabs,yabs)
      xabs= (( xrel*trcosf - yrel*trsinf)*trscal)+xabs
      yabs= (( xrel*trsinf + yrel*trcosf)*trscal)+yabs
      return
      end

C
C  Virtuelles Zeichnen, absolut
C

      subroutine drawa (X,Y)
      include 'Tktrnx.fd'
      call wincot (x,y,ix,iy)
      call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
      call drwabs (ix,iy)
      call swind1 (0,0,1023,780)
      return
      end



      subroutine movea (X,Y)
      include 'Tktrnx.fd'
      call wincot (x,y,ix,iy)
      call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
      call movabs (ix,iy)
      call swind1 (0,0,1023,780)
      return
      end



      subroutine pointa (X,Y)
      include 'Tktrnx.fd'
      call wincot (x,y,ix,iy)
      call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
      call pntabs (ix,iy)
      call swind1 (0,0,1023,780)
      return
      end



      subroutine dasha (X,Y, iL)
      include 'Tktrnx.fd'
      call wincot (x,y,ix,iy)
      call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
      call dshabs (ix,iy, il)
      call swind1 (0,0,1023,780)
      return
      end



      subroutine wincot (X,Y,IX,IY)
      include 'Tktrnx.fd'
      dx= x-tminvx
      dy= y-tminvy
      if ((xlog.lt.255.).and.(x.gt.0.)) dx= alog(x)-xlog
      if ((ylog.lt.255.).and.(y.gt.0.)) dy= alog(y)-ylog
      ix= ifix(dx*xfac+.5)+kminsx
      iy= ifix(dy*yfac+.5)+kminsy
      return
      end



      subroutine revcot (IX,IY,X,Y)
      include 'Tktrnx.fd'
      dx= float(ix-kminsx) / xfac
      dy= float(iy-kminsy) / yfac
      x= dx + tminvx
      y= dy + tminvy
      if (xlog.lt.255.) x= 2.718282**(dx+xlog)
      if (ylog.lt.255.) y= 2.718282**(dy+ylog)
      return
      end

C
C  Alphanumerische Ausgabe
C

      subroutine anstr (NChar, IStrin)
      dimension istrin(1)
      do 10 i=1,nchar
       call ancho (istrin(i))
10    continue
      return
      end



      subroutine ancho (ichar)
      include 'Tktrnx.fd'

      if (ichar.gt.31) goto 10
      if (ichar.eq.7) call bell
      if (ichar.eq.10) call linef
      if (ichar.eq.13) call cartn
      return

 10   call seeloc (ix,k)
      call csize (ixlen,k)
      if (ix.gt.krmrgn-ixlen) call newlin
      call toutpt (ichar)
      return
      end



      subroutine newlin
      call cartn
      call linef
      return
      end



      subroutine cartn
      include 'Tktrnx.fd'
      call seeloc (ix,iy)
      call movabs (klmrgn,iy)
      return
      end



      subroutine linef
      call seeloc (j,iy)
      call csize (j,iylen)
      if (iy.lt.iylen) call home
      call movrel (0,-iylen)
      return
      end



      subroutine baksp
      call csize (ix,iy)
      call movrel (-ix,0)
      return
      end



      subroutine newpag
      call erase
      call home
      return
      end



      function linhgt (Numlin)
      call csize (ix,iy)
      linhgt= numlin*iy
      return
      end



      function linwdt (NumChr)
      call csize (ix,iy)
      linwdt= numchr*ix
      return
      end

C
C  Initialisierungsroutinen
C

      subroutine lintrn
      include 'Tktrnx.fd'
      xlog= 255.
      ylog= 255.
      call rescal
      return
      end



      subroutine logtrn (IMODE)
      include 'Tktrnx.fd'
      call lintrn
      if ((imode .eq. 1) .or. (imode .eq. 3)) then
       xlog= 0.
      end if
      if ((imode .eq. 2) .or. (imode .eq. 3)) then
       ylog= 0.
      end if
      call rescal
      return
      end



      subroutine twindo (IX1,IX2,IY1,IY2)
      call swindo (ix1,ix2-ix1,iy1,iy2-iy1)
      return
      end



      subroutine swindo (IX,LX,IY,LY)
      include 'Tktrnx.fd'
      kminsx= ix
      kmaxsx= ix+lx
      kminsy= iy
      kmaxsy= iy+ly
      call rescal
      return
      end



      subroutine dwindo (X1,X2,Y1,Y2)
      call vwindo (x1,x2-x1,y1,y2-y1)
      return
      end



      subroutine vwindo (X,XL,Y,YL)
      include 'Tktrnx.fd'
      tminvx= x
      tmaxvx= x+xl
      tminvy= y
      tmaxvy= y+yl
      call rescal
      return
      end



      subroutine rescal
      include 'Tktrnx.fd'
      xfac= 0.
      yfac= 0.
      if ((tmaxvx.eq.tminvx) .or. (tmaxvy.eq.tminvy)) return
      dx= tmaxvx-tminvx
      dy= tmaxvy-tminvy
      if ((xlog.eq.255.).or.(amin1(tminvx,tmaxvx).le.0.)) goto 10
       xlog= alog(tminvx)
       dx= alog(tmaxvx)-xlog
10    if ((ylog.eq.255.).or.(amin1(tminvy,tmaxvy).le.0.)) goto 20
       ylog= alog(tminvy)
       dy= alog(tmaxvy)-ylog
20    xfac= float(kmaxsx-kminsx) / dx
      yfac= float(kmaxsy-kminsy) / dy
      return
      end



      subroutine rrotat (Grad)
      include 'Tktrnx.fd'
      trsinf= sin(grad/57.29578)
      trcosf= cos(grad/57.29578)
      return
      end



      subroutine rscale (Faktor)
      include 'Tktrnx.fd'
      trscal= faktor
      return
      end



      subroutine home
      include 'Tktrnx.fd'
C      call movabs(klmrgn,750) Fuer CP/M (kein khomey verfuegbar, -> !=750)
      call movabs(klmrgn,khomey)
      return
      end



      subroutine setmrg (Mlinks, Mrecht)
      include 'Tktrnx.fd'
      klmrgn= mlinks
      krmrgn= mrecht
      return
      end



      subroutine seetrm (IBaud,Iterm,ICSize,MaxScr)
      include 'Tktrnx.fd'
      ibaud= 0
      iterm= 1
      icsize= 1
      maxscr= 1023
      return
      end



      subroutine seetrn (xf,yf,key)
      include 'Tktrnx.fd'
      xf= xfac
      yf= yfac
      key= 1
      if ((xlog.lt.255.).or.(ylog.lt.255.)) key=2
      return
      end



      logical function genflg (ITEM)
      genflg= item.eq.0
      return
      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

```
; // DOXYGEN Dokumentation TCSdDOS.asm: als C-Programm möglich da ";" C-Leerbefehl entspricht
; /** \file TCSdDosa.asm \brief DOS Port: x86 Assembler Routinen \version 1.4
; \author  (C) 2022 Dr.-Ing. Klaus Friedewald
; \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3   */

; //! \brief Tastaturabfrage \param[out] [AX] Funktionsrückgabe ASCII
; (int) ktinput ()

; //! \brief Signalton
; (void) bell ()

; //! \brief Initialisierung Graphikmaus
; //! \param[out] iAvail Maus vorhanden
; //! \param[out] iButton Anzahl Tasten
; //! \param[in] iXmin, iXmax, iYmin, iYmax Zeichenfläche
; (void) GinCrsIn (bool iAvail,int iButton,int iXmin,int iXmax,int iYmin,int iYmax)

; //! \brief Abfrage Graphikmaus
; //! \param[out] ic Gedrueckte Taste
; //! \param[out] ix, iy  Cursorposition
; (void) GinCrs (int ic,int ix,int iy)

; //! \brief Reset Graphikmaus
; (void) GinCrsEx ()

; //! \brief Abfrage Enviromentvariable
; //! \param[in,out] Buf in=Variable out=Uebersetzung
; //! \param[in]   BufLen

; (void) GetEnv (char Buf, int BufLen)
; //! \brief Kopieren eines Feldes
; //! \param[in] iByte Anzahl verschiebender Bytes (0 zulässig)
; //! \param[in] Source zu kopierende Daten
; //! \param[out] Dest Zielfeld, kann auch Source überlappen

; (void) lib_movc3 (int iByte, char Source,char Dest)
; //! \brief Oeffnen eines Bytefiles
; //! \param[out] iErr Errorflag
; //! \param[out] iHandle Filehandle
; //! \param[in] FilNam Dateiname

; (void) OpenBytFil(int iErr,int iHandle,char FilNam)
; //! \brief WrtBytFil   Byteweises Schreiben ohne Steuerzeichen
; //! \param[out] iErr Errorflag
; //! \param[in] iHandle Filehandle
; //! \param[in] buf Daten
; //! \param[in] iWrite Anzahl zu schreibender Bytes

; (void) WrtBytFil (int iErr,int iHandle, char buf, int iWrite)
; //! \brief Schliesen eines Bytefiles
; //! \param[in] iHandle Filehandle

; (void) CloseBytFil (int iHandle)
; //! \cond
; ---------------------------   Changelog   ------------------------------
;
;  Version 1.2
;     25.10.01                Dr. Ing. K. Friedewald
;
;           ktinput:    Tastaturabfrage
;           bell:       Signalton
;           GinCrsIn:   Initialisierung Graphikmaus
;           GinCrs:     Abfrage Graphikmaus
;           GinCrsEx:   Wiederherstellen Graphikmaus
;
;           GetEnv:     Abfrage Enviromentvariable (C-Characterformat!)
;                        Input: Pufferfeld, Vorbesetzt mit Variablenname
;                               max. Länge Pufferfeld (einschliesslich char(0))
;                        Output:Pufferfeld, Übersetzter Wert
;
;           Lib_movC3   Kopieren eines Feldes
;                        Input: iByte, Anzahl verschiebender Bytes (0 zulässig)
;                               Source, zu kopierende Daten
;                        Output:Dest, Zielfeld, kann auch Source überlappen
;
;           OpenBytFil  Oeffnen eines Bytefiles
;                        Input: FilNam
;                        Output:iErr, iHandle
;
;           WrtBytFil   Byteweises Schreiben ohne Steuerzeichen
;                        Input: iHandle, Buf(*), iCount
;                        Output:iErr
;
;           CloseBytFil Schliesen eines Bytefiles
;                         Input: iHandle
```

```
;
;
;
;  Version 1.31
;     30.05.02                Dr. Ing. K. Friedewald
;
;           Anpassung an WATCOM-Assembler:
;            Auskommentieren der Microsoft-spezifischen Assemblerdirektiven
;            .no87, .list, title, subtitle, page
;           Bugfix: Fehlerhafte Parameterübergabe WRTBYTFIL:
;                    DS von Buf wurde überschrieben
;                    iErr jetzt übergeben (Programm: MOV, Deklaration:Offset)
;
;
;  Version 1.32
;     25.10.02                Dr. Ing. K. Friedewald
;
;           Bugfix: Schnell aufeinanderfolgende GINCRS-Aufrufe fehlerhaft
;                   Warten auf nicht gedrueckte Maustaste ergaenzt
;
;  Version 1.33
;     29.10.04                Dr. Ing. K. Friedewald
;
;           Anpassung an OpenWatcom-Linker 1.3: Großschreibung PUBLIC-Symbole
;
;  Version 1.4
;     04.12.20                Dr. Ing. K. Friedewald
;
;           Dokumentation durch DOXYGEN
;
;

;           title       'TCS Assembler Routinen'
            .8086
;            .no87
;            .list
            .model large

            public      KTINPUT     ; FORTRAN: integer*2 function ktinput ()

            public      BELL        ; FORTRAN: call bell ()

            public      GINCRS      ; FORTRAN: call gincrs (ic,ix,iy)
iC          equ    [BP] + 14        ; Integer*2 (Rückgabe 1,2: linke,rechte Maustaste sonst ASCII
iX          equ    [BP] + 10        ; Integer*2
iY          equ    [BP] + 6         ; Integer*2

            public      GINCRSIN    ; FORTRAN: call gincrsIn (iAvail, iButton, iX0,iX1,iY0,iY1)
iAvail      equ    [BP] + 26        ; Integer*2 oder Logical*2
iButton     equ    [BP] + 22        ; Integer*2
iX0         equ    [BP] + 18        ; Integer*2
iX1         equ    [BP] + 14        ; Integer*2
iY0         equ    [BP] + 10        ; Integer*2
iY1         equ    [BP] + 6         ; Integer*2

            public      GINCRSEX    ; FORTRAN: call GinCrsEx ()

            public      GETENV      ; FORTRAN: call GetEnv (CHARBUF, CharBufL)
CharBuf     equ    [BP] + 10        ; Vorbesetzt mit "NAME="//char(0)
CharBufL    equ    [BP] + 6

            public      OPENBYTFIL  ; FORTRAN: call OpenBytFil (iErr, iHandle, Filnam)
iErrO       equ    [BP] + 14
iHandleO    equ    [BP] + 10        ; integer*2 iHandle <> 0 falls o.k.
FilNam      equ    [BP] + 6         ; C-String

            public      WRTBYTFIL   ; FORTRAN: call WrtBytFil (iErr, iHandle, Buf, iCount)
iErr        equ    [BP] + 18
iHandle     equ    [BP] + 14        ; Integer*2
Buf         equ    [BP] + 10        ; byte array
iCount      equ    [BP] + 6         ; Integer*2

            public      CLOSEBYTFIL ; FORTRAN: call CloseBytFil (iHandle)
iHandleC    equ    [BP] + 6

            public      LIB_MOVC3_  ; FORTRAN: call Lib_MovC3_ (iByte, Source, Dest)
iByte       equ    [BP] + 14
Source      equ    [BP] + 10
Dest        equ    [BP] + 6

TCSdDosA_data segment public 'DATA'    ; obligatorischer Name für MS-Compiler


CrsDefHotX  equ    0                ; Definition Graphikmousecursor
CrsDefHotY  equ    0                ; Vorsicht, Cursor kann nicht über linke, obere Ecke geclippt
   werden!
CrsDef      dw     16 dup (0ffffh)  ; Screenmask (wird AND verküpft)
```

```
            dw    07c00h, 0c000h    ; Cursorform (wird XOR verknüpft)
            dw    0a000h, 09000h
            dw    08800h, 08400h
            dw    00200h, 00100h
            dw    00080h, 00000h
            dw    00000h, 00000h
            dw    00000h, 00000h
            dw    00000h, 00000h

TCSdDosA_data  ends

DGROUP      group TCSdDosA_data

;            subtitle    'TCS Basisfunktionen'
;            page

TcsdDosA_text segment public 'code'   ; obligatorischer Name für MS-Compiler

            assume CS:TcsdDosA_text, DS:DGROUP, SS:DGROUP

DOS         equ   021h              ; DOS-Interrupt
MOUSE       equ   033h              ; Mousedriver
VideoBIOS   equ   010h

;
; ********************
; *                  *
; * Function KTINPUT *
; *                  *
; ********************
;

ktinput     proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds

            mov   ah, 07h           ; DOS 7: Zeichen ohne Echo einlesen
            int   DOS
            mov   ah,0h

            pop   ds
            pop   bp
            ret

ktinput     endp
;
; *******************
; *                 *
; * Subroutine BELL *
; *                 *
; *******************
;
bell        proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds

            mov   ah, 0eh           ; Video-Bios: TTY Out
            mov   al, 07h           ; Bell
            mov   bh,0              ; Bildschirmnummer
            mov   bl,0              ; Grafik-Vordergrundfarbe
            int   VideoBIOS

            pop   ds
            pop   bp
            ret

bell        endp

;            subtitle    'Graphic Input Cursor'
;            page
;
; ***********************
; *                     *
; * Subroutine GINCRSIN *
; *                     *
; ***********************
;
ginCrsIn    proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds
            push  es

            mov   ax, 00h           ; FN : Reset Mouse
            int   MOUSE
            push  bx                ; Freimachen Indexregister
            lds   bx, iAvail        ; Adresse iAvail nach BX laden
            mov   [bx],ax           ; Wert AX nach iAvail
            lds   bx, iButton       ; Adresse iButton nach BX laden
            pop   ax
            mov   [bx],ax           ; Wert AX nach iButton

            mov   ax, 07h           ; FN : Setzen iXmin und iXmax
            lds   bx, iX0
            mov   cx,[bx]
            lds   bx, iX1
            mov   dx,[bx]
            int   MOUSE

            mov   ax, 08h           ; FN : Setzen iYmin und iYmax
            lds   bx, iY0
            mov   cx,[bx]
            lds   bx, iY1
            mov   dx,[bx]
            int   MOUSE

            mov   ax, 09h           ; FN : Definition Cursorform
            mov   bx, CrsDefHotX
            mov   cx, CrsDefHotY
            mov   dx, seg CrsDef    ; Mousedriver: Adressangabe über ES!
            mov   es, dx
            mov   dx, offset CrsDef
            int   MOUSE

            pop   es
            pop   ds
            pop   bp
            ret   24                 ; Parameteranzahl * 4 Bytes freigeben
gincrsIn    endp
;
; **********************
; *                    *
; * Subroutine GINCRSEX *
; *                    *
; **********************
;
ginCrsEx    proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds

            mov   ax, 00h           ; FN : Reset Mouse
            int   MOUSE

            pop   ds
            pop   bp
            ret   0                 ; Parameteranzahl * 4 Bytes freigeben
gincrsEx    endp
;
; ********************
; *                  *
; * Subroutine GINCRS *
; *                  *
; ********************
;
gincrs      proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds

            mov   ax, 01h           ; FN : Show Cursor
            int   MOUSE

WaitUp:     mov   ax, 03h           ; FN: Get Button Status
            int   MOUSE
            test  bx,bx             ; Taste noch gedrueckt?
            jnz   WaitUp            ; noch vom letzten mal -> Warte

KeyLoop:    mov   ax, 03h           ; FN : Get Button Status
            int   MOUSE             ; MouseDriver-Call
            test  bx,bx             ; Bit0 linke, Bit 1 rechte Maustaste
            jnz   ExitKeyLp         ; Taste gedrückt -> fertig

            mov   ah,06h            ; DOS 6: Zeichen ohne Warten einlesen
            mov   dl,0ffh
            int   DOS
            jz    KeyLoop           ; keine Keyboardtaste gedrückt -> weiter

            mov   ah,0h
            push  ax                  ; Terminator
            mov   ax, 03h             ; FN : Get Mouse Koordinaten
            int   MOUSE
            pop   bx                  ; Terminator ASCII

ExitKeyLp:  push  bx                  ; Terminator
            lds   bx, iX              ; Adresse iX nach BX laden
            mov   [bx],cx             ; CX: horizontale Mauskoordinate
            lds   bx, iY              ; Adresse iY nach BX laden
            mov   [bx],dx             ; DX: vertikale Mauskoordinate
            pop   ax                  ; Terminator
            lds   bx, iC              ; Adresse iC nach BX laden
            mov   [bx],ax             ; Übergabe in iC


            mov   ax, 02h             ; FN : Hide Cursor
            int   MOUSE

            pop   ds
            pop   bp
            ret   12                  ; Parameteranzahl * 4 Bytes freigeben
gincrs      endp

;            subtitle    'Get Enviroment'
;            page
;
; *********************
; *                   *
; * Subroutine GETENV *
; *                   *
; *********************
;
GetEnv      proc far

            push  bp
            mov   bp,sp               ; lokale Basis
            push  ds
            push  es
            push  di
            push  si
            pushf                     ; Rette Direction Flag!

            cld                       ; Stringsuche aufwärts
;
; Bestimmung Stringlänge Suchstring
;
            mov   cx, 0               ; Counter
            lds   si, CharBuf         ; Buffer = Suchstring
LenLoop:    mov   al,byte ptr ds:[si]; nächstes Zeichen
            or    al,al               ; Char(0) = Ende?
            jz    LenDone             ; ja
            inc   cx
            inc   si
            jmp   LenLoop

LenDone:    push  cx                  ; Länge des Suchstrings
;
; Get Enviroment
;
            mov   ah, 62h             ; DOS 62h: Get PSP
            int   DOS
            mov   es,bx               ; ES:00 jetzt auf PSP
            mov   bx,es:[2ch]         ; PSP Element 2c: Enviroment
            mov   es, bx
            xor   di,di               ; Jetzt: ES:DI auf 1. Eintrag Enviroment

SearchLoop: lds   si, CharBuf         ; Suchstring in DS:AX
            pop   cx                  ; Länge Suchstring
            push  cx
            repe  cmpsb               ; vergleichen mit Enviroment
            jz    Found
            xor   al,al               ; Ende Enviromenteintrag suchen
            mov   cx,-1
            repnz scasb
            cmp   byte ptr es:[di],0; letzter Eintrag?
            jnz   SearchLoop
            jmp   NotFound
;
; Abspeichern in den Puffer
;
NotFound:                             ; ES:DI auf Char(0)
Found:                                ; ES:DI auf Inhalt Enviromentvariable

            lds   bx, CharBufL        ; Parameter Bufferlänge
            mov   cx,[bx]             ; Counter = Bufferlänge

            lds   si, CharBuf       ; Zieladresse
StoreLoop:  mov   al,byte ptr es:[di]; nächstes Zeichen
            mov   byte ptr ds:[si],al; speichern
            or    al,al             ; Char(0) = Ende?
            jz    StoreDone         ; ja
            inc   di
            inc   si
            dec   cx
            jz    StoreDone         ; Bufferende erreicht
            jmp   StoreLoop

StoreDone:  pop   ax                ; Clear Stack, Suchstringlänge

            popf                    ; Restore Status
            pop   si
            pop   di
            pop   es
            pop   ds
            pop   bp
            ret   8

GetEnv      endp

;            subtitle    'Byte Files'
;            page
;
; ***********************
; *                     *
; * Function OpenBytFil *
; *                     *
; ***********************
;
OpenBytFil  proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds

            lds   dx,FilNam
            xor   cx,cx             ; Löschen Attribut -> unbeschränkter Zugriff
            mov   ah,05bh           ; Open New File
            int   DOS

            lds   bx, iHandleO      ; Adresse iButton nach BX laden
            mov   [bx],ax           ; FileHandle nach iHandle

            lds   bx, iErrO
            jc    ErrO              ; kein Carryflag -> iErr=0: i.O.
            xor   ax,ax             ; iErr=3: path not found, =4 too many open files
ErrO:       mov   [bx],ax           ; =5 access denied, =50h file exists

            pop   ds
            pop   bp
            ret   12                ; 12 = 3 Parameter

OpenBytFil  endp
;
;
; ***********************
; *                     *
; * Function WrtBytFil *
; *                     *
; ***********************
;

WrtBytFil   proc far

            push  bp
            mov   bp,sp             ; lokale Basis
            push  ds

            lds   bx,iCount
            mov   cx,[bx]
            jcxz  NoWrt             ; keine Bytes zu schreiben

            lds   bx,iHandle
            mov   bx,[bx]

            lds   dx,Buf            ; letzter Befehl vor DOS-call, DS auf Buf!

            mov   ah,040h           ; Write File
            int   DOS

            lds   bx,iCount
            mov   cx,[bx]
            xor   dx,dx             ; Clear Error-Flag
            cmp   ax,cx              ; Count IST < Count SOLL?
            jnl   WrtIO
            mov   dx,0ffffh          ; SET Error-Flag
WrtIO:      lds   bx,iErr            ; Store Error-Flag
            mov   [bx],dx

NoWrt:      pop   ds
            pop   bp
            ret   16                 ; 16 = 4 Parameter

WrtBytFil   endp
;
; ************************
; *                      *
; * Function CloseBytFil *
; *                      *
; ************************
;
CloseBytFil proc far

            push  bp
            mov   bp,sp              ; lokale Basis
            push  ds

            lds   bx,iHandleC
            mov   bx,[bx]
            mov   ah,03eh            ; Close File
            int   DOS

            pop   ds
            pop   bp
            ret   4                  ; 4 = 1 Parameter

CloseBytFil endp

;            subtitle    'lib$MoveC3'
;            page
;
; ************************
; *                      *
; * Subroutine lib_MovC3 *
; *                      *
; ************************
;
lib_movc3_  proc far

            push  bp
            mov   bp,sp              ; lokale Basis
            push  ds
            push  es
            push  di
            push  si
            pushf                    ; Rette Direction Flag!

;
; Kopieren des Strings
;

            lds   bx,iByte
            mov   cx,[bx]            ; Counter
            lds   si, Source         ; Buffer = Suchstring
            les   di, Dest

            cld                      ; aufwärts
            cmp   di,si
            jb    domove

            add   di,cx
            dec   di
            add   si,cx
            dec   si
            std                      ; abwärts

domove:     rep   movsb

            popf                     ; Restore Status
            pop   si
            pop   di
            pop   es
            pop   ds
            pop   bp
            ret   12

lib_movc3_  endp

TcsdDosA_text ends

```

```
            end
;
; //! \endcond

```

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

```
C> \file    TCSdDosa.fi
C> \brief   DOS Port: FORTRAN-Interface TCSdDOSa.asm
C>
C> \~german
C> Interfacedeklarationen fuer den Watcom Fortran-Compiler
C> \~english
C> Interface definitions for the Watcom Fortran Compiler
C> \~
C> \author  Dr.-Ing. Klaus Friedewald
C> \version 1.32
C> \date 06.02.2003
C> \~german
C> \note
C> Assemblerroutinen entsprechend Microsoft Procedure Call Standard
C>
C> \note
C> Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
C> von $ und _. Deswegen $notruncate und $notstrict ueberfluessig.
C>
C> \note
C> Hex-Zahlen werden nicht durch \#ff sondern durch \'ff\'x dargestellt
C> \~english
C> \note
C> Assemblerroutines are written according to the Microsoft Procedure Call Standard.
C>
C> \note
C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
C> contain $ and _. That for $notruncate und $notstrict are superfluous.
C>
C> \note
C> Hexadecimal numbers are represented by 'ff'x instead of \#ff.
C> \~
C>
C
C Interfacedeklarationen fuer den Watcom Fortran-Compiler
C Assemblerroutinen entsprechend Microsoft Procedure Call Standard
C
C
C   ktinput:    Tastaturabfrage [AX] dos7h
C   bell:       Signalton [ax,bx] video bios tty out
C   GinCrsIn:   Initialisierung Graphikmaus [ax,bx,cx,dx] int mouse
C   GinCrsEX:   Wiederherstellen Graphikmaus [ax] int mouse
C   GinCrs:     Abfrage Graphikmaus [ax,bx,cx,dx] int mouse
C
C   GetEnv:     Abfrage Enviroment (C-Characterformat!)[ax,bx,cx,dx] int dos
C
C   Lib_movC3_: Kopieren eines Feldes [ax,bx,cx]
C
C   OpenBytFil [ax,bx,cd,dx] dos
C   WrtBytFil [ax,bx,cd,dx] dos
C   CloseBytFil [ax,bx]
C   i.O.: kTinput, bell
C
C \cond

c$pragma aux kTinput value [ax] modify exact [ax]

c$pragma aux bell parm [] modify exact [ax bx]

c$pragma aux GetEnv parm reverse (DATA_REFERENCE FAR, REFERENCE FAR) []\
c  modify exact [ax bx cx dx]

c$pragma aux GinCrsIn parm reverse (REFERENCE FAR, reference far, \
c  reference far) [] modify exact [ax bx cx dx]

c$pragma aux GinCrs parm reverse (REFERENCE FAR) [] \
c  modify exact [ax bx cx dx]

c$pragma aux GinCrsEx modify exact [ax]

c$pragma aux lib_movC3_ parm reverse (REFERENCE FAR, DATA_REFERENCE FAR, \
c  DATA_REFERENCE FAR) [] modify exact [ax bx cx]

c$pragma aux OpenBytFil parm reverse (REFERENCE FAR, REFERENCE FAR, \
c  DATA_REFERENCE FAR) [] modify exact [ax bx cx dx]

c$pragma aux WrtBytFil parm reverse (REFERENCE FAR, REFERENCE FAR, \
c  DATA_REFERENCE FAR, REFERENCE FAR) [] modify exact [ax bx cx dx]

c$pragma aux CloseBytFil parm reverse (REFERENCE FAR) [] modify exact [ax bx]
C
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

```
C> \file       TCSdrDOS.for
C> \brief      DOS Port: High-Level Driver
C> \version    (2005, 45,2)
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C> \note \verbatim
C>    Erweiterungen gegenüber Tektronix:
C>     subroutine TOUTSTC (String): Ausgabe Fortran-String
C>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
C>     subroutine TXTCOL (iCol): Setzen Textfarbe
C>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
C>     subroutine DefaultColour: Wiederherstellung Defaultfarben
C> \endverbatim
C> \~english
C> \note \verbatim
C>    Extensions of the Tektronix TCS:
C>     subroutine TOUTSTC (String): Output Fortran-String
C>     subroutine LINCOL (iCol): Set line color (iCol=0..15)
C>     subroutine TXTCOL (iCol): Set text color
C>     subroutine BCKCOL (iCol): Set background color (visible after ERASE)
C>     subroutine DefaultColour: Reset default colors
C> \endverbatim
C> \~
C>
C
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     07.02.02 Version 1.2:
C           Implementierung multilinguale Fehlermeldungen
C
C     24.05.02 Version 1.3:
C           Umgebungsvariablen werten auch mit ";" getrennte Pfade aus
C           INCLUDE Interface TCSDOSA.FI zur Anpassung an den WATCOM-Compiler
C           TKTRNX.for: geänderte Common-Blocklänge aufgrund INTEGER*4
C                       bei WATCOM (MS: INTEGER*2)
C           DSHABS:     Ersatz Hex-Konstante durch Dezimalkonstante zur
C                       Erzielung Kompatibilität mit WATCOM-Compiler
C           INITT1:     Anpassung WATCOM-Compiler:
C                       - Apostrophe innerhalb von Strings durch 2 Apostrophe
C                       - Strings muessen mit CHAR(0) abgeschlossen werden
C                       BugFix: SETVIEWPORT erwartet INTEGER*2 statt REAL*4!
C           TOUTPT:     Anpassung WATCOM: auszugebender Character mit CHAR (0)
C           GraphicError: Format 900 ist bei den *.lng-Files streng zu be-
C                       folgen, d.h. "_12,Text" . Ausgabe OUTTEXT mit char(0).
C
C     23.07.02 Version 1.31:
C           Erweiterung: subroutine WINLBL (GraphicWinLbl, StatusWindLbl):
C                       Kompatibilität zu Windowsversion.
C           Eigenes Modul OUTTEXT zur Kompatibilität MS-WATCOM
C                       (Watcom benötigt char(0), dann MS Zeilenüberlauf)
C
C     24.08.02 Version 1.32:
C           ITALIC, ITALIR, DBLSIZ, NRMSIZ: Schriftarten Graphikausgabe.
C                       Kombination groß/kursiv nicht vorgesehen.
C                       Implementiert durch Fontfile GraphLib.FON
C                       (Quelle: Programm SOFTY und Arial Terminal TTF-Basis)
C           DSHABS:     Standardisierung Dash-Linestyles DOS-Windows:
C                       0: solid, 1: dotted, 2: dash-dotted, 3:dashed
C           DEFAULTCOLOUR: Bugfix Namensgebung, nicht DEFAULTCOLOURS
C
C     10.10.02 Version 1.33:
C           INITT:      Zur Vereinheitlichung DOS/Windows jetzt in diesem File
C           TCSLEV:     neu, zur Angleichung mit AG2LEV und Systemerkennung
C
C     19.10.02 Version 1.34 bzw. (2002,292,2)
C           Umbenennung TKTRNX.FOR in TKTRNX.FD zur Kompatibilität CP/M
C
C     25.10.02 Version (2002,298,2)
C           Entprellen Mousetaste bei GIN-Cursoreingabe
```

```
C
C     06.02.03 Version (2003, 37,2)
C           Vereinheitlichtes Interface lib$movc3 (Kompatibilitaet Windows)
C
C     12.01.04 Version (2004, 12,2)
C           INITT1:     Bugfix Endlosschleife bei fehlerhaftes Fontfile und
C                       Severity 5
C           GRAPHICERRORINIT: Defaultseverity 10 bei EXIT (FINITT, iErr=12)
C           Anmerkung: Die Subroutine GRAPHICERROR ruft sich bei Programm-
C                 abbruch über FINITT implizit selber rekursiv auf (nicht
C                 FORTRAN-konform!). Da jedoch keine lokalen Variablen ver-
C                 wendet werden, ist dies in der Regeln nicht kritisch.
C
C     25.10.04 Version (2004,299,2)
C           WINLBL:     Wertet jetzt den 3. Parameter (Initilisierungsfile)
C                       analog zur Windowsversion aus (einschliesslich Ueber-
C                       setzung '%:' und '.%'
C           LIB$MOVC3:  Umbenannt in LIB_MOVC3. Alte Assemblerroutine heisst
C                       jetzt LIB_MOVC3_.
C
C     15.02.05 Version (2005, 45,2)
C           GRAPHICERROR: Bugfix ErrSeverity=0 entspricht jetzt NO ACTION.
C

      include 'FGRAPH.FI'
      include 'TCSdDOSa.FI'



C
C  Ausgabe der Softwareversion
C
      subroutine tcslev(LEVEL)
      integer LEVEL(3)
      level(1)=2005     ! Aenderungsjahr
      level(2)=  45     ! Aenderungstag
      level(3)=   2     ! System= DOS

      return
      end



C
C  Bildschirm Verwaltung
C



      subroutine initt (iDummy)
      call lintrn
      call swindo (0,1023,0,780)
      call vwindo (0.,1023.,0.,780.)
      call rrotat (0.)
      call rscale (1.)
      call setmrg (0,1023)
      call initt1
      call home
      return
      end



      subroutine initt1
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      integer*2 iErr, iAvail, iButton, kScrX2, kScrY2
      integer iLen, iTrimLen, iParse

      character*80 cBuf, cBuf1*80
      record /videoconfig/ myscreen
      record /fontinfo/ myfont

      character *13 cFontFile            ! Graphikfontfile
      parameter(cfontfile='GRAPHLIB.FON'//char(0))

      character*5 cEnv                   ! Logischer Name für den Fontfilepfad
      parameter(cenv='LIB='//char(0))

      call graphicerrorinit

      ierr= setvideomode($maxresmode)

      if (ierr .eq. 0) then
       call graphicerror (2,' ') ! TCS-Initt: unknown graphic adapter
      end if

      call getvideoconfig (myscreen)
      kscrx= myscreen.numxpixels-1
      kscry= myscreen.numypixels-1-
     1   (myscreen.numypixels/myscreen.numtextrows)      ! Höhe Statuszeile

      call setviewport (0,0, kscrx, kscry)

      call settextwindow (myscreen.numtextrows,1,myscreen.numtextrows,
     1  myscreen.numtextcols)                            ! Statuszeile
      kstcol= myscreen.numtextcols - 1 ! Verhindere Scrollen durch -1

      if (registerfonts(cfontfile).lt.0) then
       cbuf= cenv                              ! Abfrage Enviroment
       call getenv (cbuf, len(cbuf))
       ilenpath= itrimlen(cbuf)
       iparse=1
   10  continue ! while
        if (iparse.le.ilenpath) then
         ilen= index(cbuf(iparse:ilenpath), ’;’)-1
         if (ilen.le.0) ilen=ilenpath-iparse+1
        else
         ilen= -1
        end if
        if ((ilen.lt.1).or.(iparse.gt.ilenpath)) then
         cbuf1= cenv    ! Notwendig zur Bildung des Substrings aus PARAMETER
         cbuf1=cbuf1(1:istringlen(cbuf1))//’:’//cfontfile
         call graphicerror (3,cbuf1(1:istringlen(cbuf1))) !openerror fontfile
         goto 15 ! ENDWHILE falls Errorseverity(3) < 10 (STOP)
        else
         cbuf1= cbuf(iparse:iparse+ilen-1)//’\’//cfontfile ! Chr0 in cFontFile
         call substitute (cbuf1,cbuf1, ’\\’, ’\’) ! kein doppelter Backslash!
        end if
       if (registerfonts(cbuf1(1:istringlen(cbuf1))).lt.0) then ! end while
        if (ilen.lt.ilenpath) then
         iparse= iparse+ilen+1
         goto 10        ! nächster Eintrag im Pfad
        else
         call graphicerror (3,cbuf1(1:istringlen(cbuf1)))
        end if
   15  end if
      end if

      call nrmsiz               ! Standardschrift: normalgroß, nicht kursiv

      kscrx2= kscrx             ! Konvertierung in int*2 durch WATCOM-Compiler
      kscry2= kscry
      call gincrsin (iavail, ibutton, 0, kscrx2, 0, kscry2)
      if (iavail.eq.-1) then
       imouse= ibutton
      else
       imouse= 0
      end if
      call defaultcolour
      call erase

      return
      end



      subroutine italic
C
C Verändern des Graphik-Fonts
C
      include ’FGRAPH.FD’
      include ’TKTRNX.FD’
      integer*2 iErr
      record /fontinfo/ myfont

      ierr= setfont(’t’’Italic’”//char(0))
      goto 10

      entry dblsiz
      ierr= setfont(’t’’Double’”//char(0))
      goto 10

      entry italir
      entry nrmsiz
      ierr= setfont(’t’’Normal’”//char(0))

 10   continue          ! identischer Code für ITALIC und ITALIR
      if (ierr.lt.0) then
       call graphicerror (4,’Normal/Italic/Double’) ! TCS-Initt: unknown font
      end if
      ierr= getfontinfo(myfont)
      khorsz= isign(irevscreenxcoord(int(myfont.pixwidth))
     1  - irevscreenxcoord(0),1)
```

```
      kversz= isign(irevscreenycoord(int(myfont.pixheight))
     1  - irevscreenycoord(0),1)
      khomey= 780-(1.1*kversz)
      return
      end



      subroutine graphicerrorinit
C
C  SUBROUTINE GraphicErrorInit, ENTRIES WinLbl, GraphicError
C  Internationalisierung der Fehlermeldungen
C
      implicit none
      include 'FGRAPH.FD'
      save errseverity, errmsg, filnam

      integer MaxErr
      parameter(maxerr=12)
      character *(*) Mssg
      character *(*) WinLblDummy, StatLblDummy, MessageFile
      integer iErr, i, iTrimLen,iStringLen, iErrSev
      integer iLenPath, iParse, iLen

      character*132 cEnv, FilNam, cBuf
      integer ErrSeverity (MaxErr)
      character*80 ErrMsg (MaxErr)
      data cenv,filnam /'LIB=','GRAPHLIB.LNG'/
      data errmsg/'GRAPHLIB %%% INITT: Incompatible message file - Press
     1 any key',
     2      'GRAPHLIB %%% INIT: Unknown graphic adapter',
     3      'GRAPHLIB %%% INIT: Error opening fontfile $$',
     4      'GRAPHLIB %%% INIT: Unknown font $$',
     5      'GRAPHLIB %%% INPUT: No mousedriver available, use keyboard'
     6     ,'GRAPHLIB %%% HARDCOPY: Error during OPEN',
     7      'GRAPHLIB %%% HARDCOPY: Error during WRITE',
     8      'GRAPHLIB %%% HARDCOPY: Internal error (buffer overflow)',
     9      '$$','Hardcopy in progress','Press any key to continue',
     2      'Press any key to exit program'/

      data errseverity /5,10,10,10, 1, 5, 5, 5, 1, 1, 5, 10/

      external iGetArg           ! Watcom Library-Funktion
      integer iGetArg

      cenv=cenv(1:itrimlen(cenv))//char(0)
      filnam= filnam(1:itrimlen(filnam))//char(0)

C
C  1.Priorität: Message-File durch WinLbl spezifiziert
C  2.Priorotät: GRAPHLIB.LNG im Arbeitsdirectory
C

      open (unit=9,form='FORMATTED', err=5, status='OLD', file=
     1                         filnam(1:istringlen(filnam)))
      goto 7      ! File gefunden -> Einlesen

C
C  3.Priorität: Message-File GRAPHLIB.LNG in LIB:
C

5     call getenv (cenv, len(cenv))
      ilenpath= itrimlen(cenv)
      iparse=1
   10 continue ! while
       if (iparse.le.ilenpath) then
        ilen= index(cenv(iparse:ilenpath), ';')-1
        if (ilen.le.0) ilen=ilenpath-iparse+1
       else
        goto 99          ! benutze Default
       end if
       if ((ilen.ge.1).and.(iparse.le.ilenpath)) then
        cbuf= cenv(iparse:iparse+ilen-1)//'\'//filnam ! Chr0 bereits in FilNam
        call substitute (cbuf,cbuf, '\\', '\') ! kein doppelter Backslash !
       end if
       open (unit=9,form='FORMATTED', err=6, status='OLD', file=
     1                         cbuf(1:istringlen(cbuf)))
      goto 7      ! File gefunden -> Einlesen
6     if (ilen.lt.ilenpath) then ! end while
       iparse= iparse+ilen+1
       goto 10  ! nächster Eintrag im Pfad
      else
       goto 99 ! kein File vorhanden - > benutze Default
      end if

7     do 20 i=1,maxerr
       read (unit=9, err=90, fmt=900) errseverity(i),errmsg(i)
```

```
20    continue

      close (unit=9)

99    return
C
C Ausgabe Fehlermeldung Messagefile
C
90    call outtext (errmsg(1)) ! Graphiksystem wurde noch nicht initialisiert!
      call tinput (i)
      return



      entry winlbl(winlbldummy, statlbldummy, messagefile)
C
C  Setzen des Messagefiles und Uebersetzung '%:' bzw. '.%'
C
      if (istringlen(messagefile).le.0) return
      filnam= messagefile
      i= igetarg(0, cbuf) ! Arg. 0: Programmname mit Directory
      if (i.gt.1) then
   30  continue ! repeat
        i= i-1
       if ((cbuf(i:i).ne.'\').and.(i.gt.1)) goto 30
       cbuf(i+1:i+1)= char(0)
       call substitute (filnam, filnam,'%:',cbuf)
      end if
      call substitute (filnam, filnam,'.%','.lng')
      return



      entry graphicerror(ierr,mssg)
C
C  Ausgabe der Fehlermeldung
C
      if (ierr.eq.99) then             ! Programmabbruch aus FINITT (2. Aufruf)
       if (errseverity(12).eq.10) then
        ierrsev= 99                            ! STOP
       else if (errseverity(12).eq.5) then
        ierrsev= 1                             ! TINPUT bereits durchgeführt
       else
        ierrsev= errseverity(12)
       end if
      else
       ierrsev= errseverity(ierr)
       if (ierrsev.gt.0) then
        call bell
        call substitute (errmsg(ierr),cbuf, '$$', mssg)
        call statst (cbuf)
       end if
      end if

      if (ierrsev.le.1) then               ! =1: Statusmeldung
       return
      else if (ierrsev.eq.99) then         
       stop                                ! =99: aus FINITT
      else
       call tinput (i)
       if (ierrsev.eq.5) then              ! =5: Warnung
        return
       else if (ierrsev.eq.10) then        ! =10: Abbruch
        if (ierr.ne.12) call finitt ()     ! Rekursion iErr=12 verhindern
       end if
      end if

      return
900   format (1x,i2,1x,a)
      end



      subroutine lincol (iCol)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      integer iColCode, iCol
      integer *2 iErr
      ilincol= icolcode(icol)
      ierr= setcolor(ilincol)
      return
      end



      subroutine txtcol (iCol)
      include 'TKTRNX.FD'
      integer iColCode, iCol
      itxtcol= icolcode(icol)
      return
      end



      subroutine bckcol (iCol)
      include 'TKTRNX.FD'
      integer iColCode, iCol
      ibckcol= icolcode(icol)
      return
      end



      Subroutine defaultcolour
      call bckcol (0)
      call lincol (1)
      call txtcol (1)
      return
      end


      integer function icolcode (iCol)
      include 'FGRAPH.FD'
      integer icoltab (15)    ! Anpassung Farbindex an VGA-Palette
      data icoltab/  15      ,12         ,10         ,11         ,9
C        iCol=       1        2           3           4           5
C        entspricht: weiss    rot         gruen       blau        lila
     1               ,14      ,7          ,13         ,4          ,2
C        iCol=       6        7           8           9           10
C        entspricht: gelb     grau        violett     mattrot     mattgruen
     2               ,1       ,3          ,6          ,8          ,5/
C        iCol=       11       12          13          14          15
C        entspricht: mattblau mattlila    orange      mattgrau    mattviolett
      if (icol.le.0) then
       icolcode= 0
      else if (icol.gt.15) then
       icolcode= icoltab(1)
      else
       icolcode= icoltab(icol)
      end if
      return
      end



      integer function iscreenxcoord (iX)
      include 'TKTRNX.FD'
      iscreenxcoord= (ix*kscrx)/1023
      return
      end



      integer function iscreenycoord (iY)
      include 'TKTRNX.FD'
      iscreenycoord= kscry-(kscry*iy)/780
      return
      end



      integer function irevscreenxcoord (iX)
      include 'TKTRNX.FD'
      irevscreenxcoord= (ix*1023)/kscrx
      return
      end



      integer function irevscreenycoord (iY)
      include 'TKTRNX.FD'
      irevscreenycoord= 780-(780*iy)/kscry
      return
      end



      subroutine erase
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      call clearscreen ($gclearscreen)
      ierr= setcolor(ibckcol)
      ierr= rectangle( $gfillinterior, 0, 0, kscrx, kscry)
      ierr= setcolor(ilincol)
      call movabs (kbeamx, kbeamy)      ! Cursorposition wiederherstellen
      return
      end



      subroutine finitt
      implicit none
      include 'FGRAPH.FD'
      integer*2 iErr
      call graphicerror (12,' ')        ! Press any key to exit program
      call unregisterfonts ()
      ierr= setvideomode($defaultmode)
      call gincrsex
      call graphicerror (99,' ')        ! Jetzt auch STOP möglich
      return
      end

C
C  Abspeichern Terminal Status Area
C

      subroutine svstat (Array)
      integer array(1)
      include 'TKTRNX.FD'
      integer arr(1)
      equivalence(arr(1),khomey)
      do 10 i=1,itktrnxl
10     array(i)= arr(i)
      return
      end



      subroutine restat (Array)
      integer array(1)
      include 'TKTRNX.FD'
      integer arr(1)
      equivalence(arr(1),khomey)
      do 10 i=1,itktrnxl
10     arr(i)= array(i)
      call movabs (kbeamx, kbeamy)
      return
      end


C
C  Absolute Zeichenbefehle
C

      subroutine movabs (ix,iy)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      record /xycoord/ oldxy
      integer iScreenXcoord, iScreenYcoord
      call moveto (iscreenxcoord(ix),iscreenycoord(iy), oldxy)
      kbeamx= ix
      kbeamy= iy
      return
      end



      subroutine pntabs (ix,iy)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      integer iScreenXcoord, iScreenYcoord
      integer oldPixel,ixs,iys
      record /xycoord/ oldxy
      ixs= iscreenxcoord(ix)
      iys= iscreenycoord(iy)
      call moveto (ixs,iys, oldxy)
      oldpixel= setpixel(ixs,iys)
      kbeamx= ix
      kbeamy= iy
      return
      end



      subroutine drwabs (ix,iy)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      integer iScreenXcoord, iScreenYcoord
      ierr= lineto(iscreenxcoord(ix), iscreenycoord(iy))
      kbeamx= ix
      kbeamy= iy
      return
      end


      subroutine dshabs (ix,iy, iMask)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      integer iScreenXcoord, iScreenYcoord
      integer*2 iErr
      if (imask.eq.0) then       ! solid line
       imask= 65535              ! 1111 1111 1111 1111
      else if (imask.eq.1) then ! dotted line
       imask= 43690              ! 1010 1010 1010 1010
      else if (imask.eq.2) then ! dash-dotted line
       imask= 58596              ! 1110 0100 1110 0100
      else if (imask.eq.3) then ! dashed line
       imask= 61680              ! 1111 0000 1111 0000
      end if
      call setlinestyle (imask)
      ierr= lineto(iscreenxcoord(ix), iscreenycoord(iy))
      call setlinestyle (65535) ! =#ffff, so zu WATCOM-Compiler kompatibel
      kbeamx= ix
      kbeamy= iy
      return
      end

C
C  Relative Zeichenbefehle
C

      subroutine movrel (iX, iY)
      include 'TKTRNX.FD'
      ixx= kbeamx + ix
      iyy= kbeamy + iy
      call movabs (ixx, iyy)
      return
      end


      subroutine pntrel (iX, iY)
      include 'TKTRNX.FD'
      ixx= kbeamx + ix
      iyy= kbeamy + iy
      call pntabs (ixx, iyy)
      return
      end


      subroutine drwrel (iX, iY)
      include 'TKTRNX.FD'
      ixx= kbeamx + ix
      iyy= kbeamy + iy
      call drwabs (ixx, iyy)
      return
      end


      subroutine dshrel (iX, iY, iMask)
      include 'TKTRNX.FD'
      ixx= kbeamx + ix
      iyy= kbeamy + iy
      call dshabs (ixx, iyy, imask)
      return
      end

C
C   Ersatz SEELOC der CP/M-Version, SEELOC1 unnötig
C

      subroutine seeloc (IX,IY)
      include 'TKTRNX.FD'
      ix= kbeamx
      iy= kbeamy
      return
      end


      Subroutine swind1 (ix1,iy1, ix2,iy2)
      include 'FGRAPH.FD'
      integer iScreenXcoord, iScreenYcoord
      call setcliprgn (iscreenxcoord(ix1),iscreenycoord(iy1),
     1                     iscreenxcoord(ix2),iscreenycoord(iy2))
```

```
      return
      end



      Subroutine alpha
      implicit none
      include 'FGRAPH.FD'
      integer*2 iErr
      ierr= setvideomode($defaultmode)
      return
      end

C
C  Textausgabe
C

      subroutine csize (Ixlen,iylen)
      include 'TKTRNX.FD'
      ixlen= khorsz
      iylen= kversz
      return
      end



      subroutine toutpt (iChr)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      record /xycoord/ oldxy
      integer iScreenXcoord, iScreenYcoord
      integer*2 iErr
      call moveto (iscreenxcoord(kbeamx),iscreenycoord(kbeamy+kversz)
     1       , oldxy)
      ierr= setcolor(itxtcol)
      call outgtext (char(ichr)//char(0))
      ierr= setcolor(ilincol)
      kbeamx= kbeamx+khorsz
      call moveto (iscreenxcoord(kbeamx), iscreenycoord(kbeamy), oldxy)
      return
      end



      subroutine toutst (nChr, iChrArr)
      integer iChrArr (1)
      if (nchr.eq.0) return
      do 10 i=1,nchr
10     call toutpt (ichrarr(i))
      return
      end



      subroutine toutstc (String)
      character *(*) String
      do 10 i=1,istringlen(string)
 10    call toutpt (ichar(string(i:i)))
      return
      end



      subroutine statst (String)
      include 'FGRAPH.FD'
      include 'TKTRNX.FD'
      record /rccoord/ s
      character *(*) String
      character *80 Buf
      buf= string(1:istringlen(string)) ! Mit Blanks auf 80 Zeichen aufgefüllt
      call settextposition (1,1,s)
      call outtext (buf(1:min(80,kstcol)))
      return
      end

C
C  Eingabe
C

      subroutine tinput (iChr)
      integer *2 kTinput
      ichr= ktinput()  ! Konversion Integer*2 nach *4 durch Compiler
      return
      end


      subroutine dcursr (IC,IX,IY)
      include 'TKTRNX.FD'
      integer ic, ix, iy
      integer*2 ic2, ix2, iy2
      if (imouse.ne.0) then
       call gincrs (ic2,ix2,iy2)
       ix= ix2                    ! Watcom: Konvertierung int*2 in int*4
       iy= iy2
       ic= ic2
      else
       call graphicerror (5, ' ') ! No Mousedriver available, use Keyboard
       call tinput (ic)
       ix= 0
       iy= 0
      end if
      ix= irevscreenxcoord(ix)
      iy= irevscreenycoord(iy)
      return
      end

C
C  Interface lib$movc3 (Anpassung Parameterübergabe durch "TcsDDosA.FI"
C
      subroutine lib_movc3 (iLen, sou, dst)
      integer iLen
      character *(*) sou,dst
      call lib_movc3_ (ilen, sou, dst)
      return
      end

C
C>  Entry Dummyroutinen
C
      subroutine  anmode
C> AlfMod
      entry       alfmod
C> pClipt
      entry       pclipt
C> ioWait
      entry       iowait
      return
      end



      logical function winselect (iDummy)
      winselect= .false.
      return
      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

```
C> \file TKTRNX.fd
C> \brief   DOS Port: TCS Common Block TKTRNX
C> \version 1.0
C> \author  Dr.-Ing. Klaus Friedewald
C> \~german
C> Common Block TKTRNX, Version für DOS und INTEGER*4 Variablen (WATCOM-Compiler)
C> \~english
C> Common Block TKTRNX, version for DOS and INTEGER*4 variables (WATCOM-Compiler)
C> \~german
C> \note
C> Da die folgende Definition kein Bestandteil eines Moduls
C> ist, versagt der DOXYGEN-Parser bei der Kombination von
C> COMMON und integer. Workaround: \\cond ... \\endcond
C> \~english
C> Because the following declaration not beeing part of a module, DOXYGEN could
C> not interpret the combinattion COMMON / INTEGER.
C> Workaround: \\cond ... \\endcond
C> \~
C> \cond
C>
C Common Block TKTRNX, Version für DOS und INTEGER*4 Variablen (WATCOM-Compiler)
C
      COMMON /tktrnx/
c           kbaudr,kerror,kgrafl,
     1 khomey,
c           kkmode,
     2 khorsz,kversz,
c           kitalc,ksizef,
     3 klmrgn,krmrgn, kscrx,kscry,
c           ktblsz,khorzt(10),kvertt(10),
     4 kbeamx,kbeamy,
c           kmovef,kpchar(4),kdasht,
     5 kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
c      trealx,trealy,timagx,timagy,
     6 trcosf,trsinf,trscal
     u ,xfac,yfac,xlog,ylog,kstcol,
     u ilincol, ibckcol, itxtcol, imouse
      SAVE /tktrnx/

      integer iTktrnxL
      parameter(itktrnxl=29) ! +11)

c Neue Variablen:
c     kScrX, kScrY: Zeichenfläche in Pixeln
c           Unterer Bildschirmrand für eine Statuszeile freigehalten
c     kBeamX, kBeamY: Aktuelle Strahlposition im (1024/780) Koordinatensystem
c     kStCol: Maximale Zeichenzahl in der Statuszeile
c     iLinCol, iBckCol, iTxtCol: Farbindices
c     iMouse: Anzahl der Maustasten. iMouse=0: keine Maus vorhanden
c
c Achtung:
c       Anpassung Parameters iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
c     Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
c
C
C> \endcond
```

Index