

Graph2D Library --- DOS ---

Generated by Doxygen 1.8.19

1 Graph2D / Plot10 & AG II- DOS Port	1
2 File Index	3
2.1 File List	3
3 File Documentation	5
3.1 AG2.for File Reference	5
3.1.1 Detailed Description	7
3.1.2 Function/Subroutine Documentation	8
3.1.2.1 ag2lev()	8
3.1.2.2 alfsetc()	8
3.1.2.3 bar()	8
3.1.2.4 binitt()	8
3.1.2.5 bsyms()	8
3.1.2.6 calcon()	9
3.1.2.7 calpnt()	9
3.1.2.8 check()	9
3.1.2.9 cmnmx()	9
3.1.2.10 coptim()	9
3.1.2.11 cplot()	10
3.1.2.12 datget()	10
3.1.2.13 dinitx()	10
3.1.2.14 dinity()	10
3.1.2.15 dlimx()	10
3.1.2.16 dlimy()	11
3.1.2.17 dsplay()	11
3.1.2.18 eformc()	11
3.1.2.19 esplit()	11
3.1.2.20 expoutc()	11
3.1.2.21 fformc()	12
3.1.2.22 filbox()	12
3.1.2.23 findge()	12
3.1.2.24 findle()	12
3.1.2.25 fonlyc()	13
3.1.2.26 frame()	13
3.1.2.27 gline()	13
3.1.2.28 grid()	13
3.1.2.29 hbarst()	13
3.1.2.30 iformc()	14
3.1.2.31 infin()	14
3.1.2.32 iothcr()	14
3.1.2.33 iubgc()	14
3.1.2.34 justerc()	14

3.1.2.35 <code>keyset()</code>	15
3.1.2.36 <code>label()</code>	15
3.1.2.37 <code>leap()</code>	15
3.1.2.38 <code>line()</code>	15
3.1.2.39 <code>locge()</code>	15
3.1.2.40 <code>locle()</code>	16
3.1.2.41 <code>logtix()</code>	16
3.1.2.42 <code>loptim()</code>	16
3.1.2.43 <code>lwidth()</code>	16
3.1.2.44 <code>mnmx()</code>	16
3.1.2.45 <code>monpos()</code>	17
3.1.2.46 <code>notatec()</code>	17
3.1.2.47 <code>npts()</code>	17
3.1.2.48 <code>numsetc()</code>	17
3.1.2.49 <code>optim()</code>	17
3.1.2.50 <code>oubgc()</code>	18
3.1.2.51 <code>place()</code>	18
3.1.2.52 <code>remlab()</code>	18
3.1.2.53 <code>rescom()</code>	18
3.1.2.54 <code>rgchek()</code>	18
3.1.2.55 <code>roundd()</code>	19
3.1.2.56 <code>roundu()</code>	19
3.1.2.57 <code>savcom()</code>	19
3.1.2.58 <code>setwin()</code>	19
3.1.2.59 <code>size()</code>	19
3.1.2.60 <code>sizes()</code>	20
3.1.2.61 <code>slimx()</code>	20
3.1.2.62 <code>slimy()</code>	20
3.1.2.63 <code>spread()</code>	20
3.1.2.64 <code>stepl()</code>	20
3.1.2.65 <code>steps()</code>	21
3.1.2.66 <code>syml()</code>	21
3.1.2.67 <code>symout()</code>	21
3.1.2.68 <code>teksym()</code>	21
3.1.2.69 <code>teksym1()</code>	21
3.1.2.70 <code>tset()</code>	22
3.1.2.71 <code>tset2()</code>	22
3.1.2.72 <code>typck()</code>	22
3.1.2.73 <code>vbarst()</code>	22
3.1.2.74 <code>vlablc()</code>	22
3.1.2.75 <code>width()</code>	23
3.1.2.76 <code>xden()</code>	23

3.1.2.77 xetyp()	23
3.1.2.78 xfrm()	23
3.1.2.79 xlab()	23
3.1.2.80 xlen()	23
3.1.2.81 xloc()	24
3.1.2.82 xloctp()	24
3.1.2.83 xmfrm()	24
3.1.2.84 xmtcs()	24
3.1.2.85 xneat()	24
3.1.2.86 xtics()	24
3.1.2.87 xtype()	25
3.1.2.88 xwidth()	25
3.1.2.89 xzero()	25
3.1.2.90 yden()	25
3.1.2.91 yetyp()	25
3.1.2.92 yfrm()	25
3.1.2.93 ylab()	26
3.1.2.94 ylen()	26
3.1.2.95 yloc()	26
3.1.2.96 ylocrt()	26
3.1.2.97 ymdyd()	26
3.1.2.98 ymfrm()	27
3.1.2.99 ymtcs()	27
3.1.2.100 yneat()	27
3.1.2.101 ytics()	27
3.1.2.102 ytype()	27
3.1.2.103 ywidth()	27
3.1.2.104 yzero()	28
3.2 AG2.for	28
3.3 AG2Holerith.for File Reference	63
3.3.1 Detailed Description	64
3.3.2 Function/Subroutine Documentation	64
3.3.2.1 alfset()	64
3.3.2.2 comdmp()	64
3.3.2.3 comget()	65
3.3.2.4 comset()	65
3.3.2.5 eform()	65
3.3.2.6 expout()	65
3.3.2.7 fform()	65
3.3.2.8 fonly()	66
3.3.2.9 hlabel()	66
3.3.2.10 hstrin()	66

3.3.2.11 ibasec()	66
3.3.2.12 ibasex()	66
3.3.2.13 ibasey()	67
3.3.2.14 iform()	67
3.3.2.15 juster()	67
3.3.2.16 notate()	67
3.3.2.17 numset()	68
3.3.2.18 vlabel()	68
3.3.2.19 vstrin()	68
3.4 AG2Holerith.for	68
3.5 AG2uline.for File Reference	73
3.5.1 Detailed Description	74
3.5.2 Function/Subroutine Documentation	74
3.5.2.1 uline()	74
3.6 AG2uline.for	74
3.7 AG2umnmx.for File Reference	74
3.7.1 Detailed Description	74
3.7.2 Function/Subroutine Documentation	75
3.7.2.1 umnmx()	75
3.8 AG2umnmx.for	75
3.9 AG2upoint.for File Reference	75
3.9.1 Detailed Description	75
3.9.2 Function/Subroutine Documentation	75
3.9.2.1 upoint()	76
3.10 AG2upoint.for	76
3.11 AG2users.for File Reference	76
3.11.1 Detailed Description	76
3.11.2 Function/Subroutine Documentation	76
3.11.2.1 users()	76
3.12 AG2users.for	77
3.13 AG2useset.for File Reference	77
3.13.1 Detailed Description	77
3.13.2 Function/Subroutine Documentation	77
3.13.2.1 useset()	77
3.14 AG2useset.for	77
3.15 AG2usesetC.for File Reference	78
3.15.1 Detailed Description	78
3.15.2 Function/Subroutine Documentation	78
3.15.2.1 usesetc()	78
3.16 AG2usesetC.for	78
3.17 AG2UsrSoftek.for File Reference	79
3.17.1 Detailed Description	79

3.17.2 Function/Subroutine Documentation	79
3.17.2.1 <code>softek()</code>	79
3.18 <code>AG2UsrSoftek.for</code>	79
3.19 <code>Fgraph.fd</code> File Reference	79
3.19.1 Detailed Description	80
3.20 <code>Fgraph.fd</code>	80
3.21 <code>Fgraph.fi</code> File Reference	85
3.21.1 Detailed Description	85
3.22 <code>Fgraph.fi</code>	85
3.23 <code>G2dAG2.fd</code> File Reference	87
3.23.1 Detailed Description	87
3.24 <code>G2dAG2.fd</code>	88
3.25 <code>GetHDC.for</code> File Reference	88
3.25.1 Detailed Description	89
3.25.2 Function/Subroutine Documentation	89
3.25.2.1 <code>gethdc()</code>	89
3.26 <code>GetHDC.for</code>	89
3.27 <code>hdcopy.for</code> File Reference	91
3.27.1 Detailed Description	91
3.27.2 Function/Subroutine Documentation	91
3.27.2.1 <code>hdcopy()</code>	92
3.27.2.2 <code>writebuf()</code>	92
3.28 <code>hdcopy.for</code>	92
3.29 <code>Mainpage.dox</code> File Reference	95
3.30 <code>outtext.for</code> File Reference	95
3.30.1 Detailed Description	95
3.30.2 Function/Subroutine Documentation	96
3.30.2.1 <code>outtext()</code>	96
3.31 <code>outtext.for</code>	96
3.32 <code>Strings.for</code> File Reference	96
3.32.1 Detailed Description	97
3.32.2 Function/Subroutine Documentation	97
3.32.2.1 <code>istringlen()</code>	97
3.32.2.2 <code>itrimlen()</code>	97
3.32.2.3 <code>printstring()</code>	97
3.32.2.4 <code>substitute()</code>	98
3.33 <code>Strings.for</code>	98
3.34 <code>TCS.for</code> File Reference	100
3.34.1 Detailed Description	101
3.34.2 Function/Subroutine Documentation	101
3.34.2.1 <code>ancho()</code>	101
3.34.2.2 <code>anstr()</code>	101

3.34.2.3 baksp()	101
3.34.2.4 cartn()	102
3.34.2.5 dasha()	102
3.34.2.6 dashr()	102
3.34.2.7 drawa()	102
3.34.2.8 drawr()	102
3.34.2.9 dwindo()	103
3.34.2.10 genflg()	103
3.34.2.11 home()	103
3.34.2.12 linef()	103
3.34.2.13 linhgt()	103
3.34.2.14 lintrn()	104
3.34.2.15 linwdt()	104
3.34.2.16 logtrn()	104
3.34.2.17 movea()	104
3.34.2.18 mover()	104
3.34.2.19 newlin()	105
3.34.2.20 newpag()	105
3.34.2.21 pointa()	105
3.34.2.22 pointr()	105
3.34.2.23 rel2ab()	105
3.34.2.24 rescal()	106
3.34.2.25 revcot()	106
3.34.2.26 rrotat()	106
3.34.2.27 rscale()	106
3.34.2.28 seetrm()	106
3.34.2.29 seetrn()	107
3.34.2.30 setmrg()	107
3.34.2.31 swindo()	107
3.34.2.32 twindo()	107
3.34.2.33 vcursr()	107
3.34.2.34 vwindo()	108
3.34.2.35 wincot()	108
3.35 TCS.for	108
3.36 TCSdDosa.asm File Reference	114
3.36.1 Detailed Description	115
3.36.2 Function Documentation	115
3.36.2.1 bell()	115
3.36.2.2 CloseBytFil()	115
3.36.2.3 GetEnv()	116
3.36.2.4 GinCrs()	116
3.36.2.5 GinCrsEx()	116

3.36.2.6 GinCrslIn()	117
3.36.2.7 kinput()	117
3.36.2.8 lib_movc3()	117
3.36.2.9 OpenBytFil()	118
3.36.2.10 WrtBytFil()	118
3.37 TCSdDosa.asm	118
3.38 TCSdDosa.fi File Reference	125
3.38.1 Detailed Description	125
3.39 TCSdDosa.fi	126
3.40 TCSdrDOS.for File Reference	127
3.40.1 Detailed Description	128
3.40.2 Function/Subroutine Documentation	128
3.40.2.1 alpha()	129
3.40.2.2 anmode()	129
3.40.2.3 bckcol()	129
3.40.2.4 csize()	129
3.40.2.5 dcursr()	129
3.40.2.6 defaultcolour()	130
3.40.2.7 drwabs()	130
3.40.2.8 drwrel()	130
3.40.2.9 dshabs()	130
3.40.2.10 dshrel()	130
3.40.2.11 erase()	131
3.40.2.12 finitt()	131
3.40.2.13 graphicerrorinit()	131
3.40.2.14 icolcode()	131
3.40.2.15 initt()	131
3.40.2.16 initt1()	131
3.40.2.17 irevscreenxcoord()	132
3.40.2.18 irevscreenycoord()	132
3.40.2.19 iscreenxcoord()	132
3.40.2.20 iscreenycoord()	132
3.40.2.21 italic()	132
3.40.2.22 lib_movc3()	133
3.40.2.23 lincol()	133
3.40.2.24 movabs()	133
3.40.2.25 movrel()	133
3.40.2.26 pntabs()	133
3.40.2.27 pntrel()	134
3.40.2.28 restat()	134
3.40.2.29 seeloc()	134
3.40.2.30 statst()	134

3.40.2.31 svstat()	134
3.40.2.32 swind1()	135
3.40.2.33 tcslev()	135
3.40.2.34 tinput()	135
3.40.2.35 toutpt()	135
3.40.2.36 toutst()	135
3.40.2.37 toutstc()	136
3.40.2.38 txtcol()	136
3.41 TCSdrDOS.for	136
3.42 TKTRNX.fd File Reference	145
3.42.1 Detailed Description	145
3.43 TKTRNX.fd	146
Index	147

Chapter 1

Graph2D / Plot10 & AG II- DOS Port

Graphic Driver for DOS

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

How to build the library:

Copy the sources into the /build subdirectory by invoking "\$getfiles.bat DOS" and use the Watcom-Workspace files.

Using the library:

After building the library and linking it to the applications, the main characteristics could be changed by the following files:

graphlib.fon: Fontfile for the graphic text

graphlib.lng: Translations of the messages

Hardcopies are generated as standard *.bmp-files.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

AG2.for	Graph2D: Tektronix Advanced Graphing II Emulation	5
AG2Holerith.for	Graph2D: deprecated AG2 routines	63
AG2uline.for	Graph2D: Dummy User Routine	73
AG2umnm.x.for	Graph2D: Dummy User Routine	74
AG2upoint.for	Graph2D: Dummy User Routine	75
AG2users.for	Graph2D: Dummy User Routine	76
AG2useset.for	Graph2D: Dummy User Routine	77
AG2usesetC.for	Graph2D: Dummy User Routine	78
AG2UsrSoftek.for	Graph2D: Dummy User Routine	79
Fgraph.fd	DOS Port: Declarations OW graph.lib	79
Fgraph.fi	DOS Port: Interface OW graph.lib	85
G2dAG2.fd	Graph2D: AG2 Common Block G2dAG2	87
GetHDC.for	Utility: Restore Hardcopies	88
hdcopy.for	DOS Port: Hardcopy	91
outtext.for	DOS Port: alphanumeric output to the graphic screen	95
Strings.for	TCS: String functions	96
TCS.for	TCS: Tektronix Plot 10 Emulation	100
TCSdDosa.asm	DOS Port: x86 Assembler Routinen	114

TCSdDosa.fi	
DOS Port: FORTRAN-Interface TCSdDOSa.asm	125
TCSdrDOS.for	
DOS Port: High-Level Driver	127
TKTRNX.fd	
DOS Port: TCS Common Block TKTRNX	145

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)
- 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

(2022,284, 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 user routines
- G2dAG2.fd:        commonblock
```

Definition in file [AG2.for](#).

3.1.2 Function/Subroutine Documentation

3.1.2.1 `ag2lev()`

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

Definition at line [94](#) of file [AG2.for](#).

3.1.2.2 `alfsetc()`

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

Definition at line [2564](#) of file [AG2.for](#).

3.1.2.3 `bar()`

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

Definition at line [1689](#) of file [AG2.for](#).

3.1.2.4 `binitt()`

```
subroutine binitt
```

Definition at line [714](#) of file [AG2.for](#).

3.1.2.5 `bsyms()`

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

Definition at line [1841](#) of file [AG2.for](#).

3.1.2.6 calcon()

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

Definition at line [1326](#) of file [AG2.for](#).

3.1.2.7 calpnt()

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

Definition at line [1271](#) of file [AG2.for](#).

3.1.2.8 check()

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

Definition at line [798](#) of file [AG2.for](#).

3.1.2.9 cmnmx()

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

Definition at line [920](#) of file [AG2.for](#).

3.1.2.10 coptim()

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

Definition at line [1115](#) of file [AG2.for](#).

3.1.2.11 cplot()

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

Definition at line [1539](#) of file [AG2.for](#).

3.1.2.12 datget()

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

Definition at line [1661](#) of file [AG2.for](#).

3.1.2.13 dinitx()

```
subroutine dinitx
```

Definition at line [644](#) of file [AG2.for](#).

3.1.2.14 dinity()

```
subroutine dinity
```

Definition at line [658](#) of file [AG2.for](#).

3.1.2.15 dlimx()

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

Definition at line [464](#) of file [AG2.for](#).

3.1.2.16 dlimy()

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

Definition at line 476 of file [AG2.for](#).

3.1.2.17 dsplay()

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

Definition at line 1525 of file [AG2.for](#).

3.1.2.18 eformc()

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

Definition at line 2435 of file [AG2.for](#).

3.1.2.19 esplit()

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

Definition at line 2468 of file [AG2.for](#).

3.1.2.20 expoutc()

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

Definition at line 2488 of file [AG2.for](#).

3.1.2.21 fformc()

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

Definition at line [2376](#) of file [AG2.for](#).

3.1.2.22 filbox()

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

Definition at line [1756](#) of file [AG2.for](#).

3.1.2.23 findge()

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

Definition at line [2923](#) of file [AG2.for](#).

3.1.2.24 findle()

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

Definition at line [2942](#) of file [AG2.for](#).

3.1.2.25 fonlyc()

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

Definition at line [2404](#) of file [AG2.for](#).

3.1.2.26 frame()

```
subroutine frame
```

Definition at line [1511](#) of file [AG2.for](#).

3.1.2.27 gline()

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

Definition at line [2174](#) of file [AG2.for](#).

3.1.2.28 grid()

```
subroutine grid
```

Definition at line [1957](#) of file [AG2.for](#).

3.1.2.29 hbarst()

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

Definition at line [672](#) of file [AG2.for](#).

3.1.2.30 iformc()

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

Definition at line [2344](#) of file [AG2.for](#).

3.1.2.31 infin()

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

Definition at line [142](#) of file [AG2.for](#).

3.1.2.32 iother()

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

Definition at line [3067](#) of file [AG2.for](#).

3.1.2.33 iubgc()

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

Definition at line [1474](#) of file [AG2.for](#).

3.1.2.34 justerc()

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

Definition at line [2667](#) of file [AG2.for](#).

3.1.2.35 keyset()

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

Definition at line 1635 of file [AG2.for](#).

3.1.2.36 label()

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

Definition at line 2201 of file [AG2.for](#).

3.1.2.37 leap()

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

Definition at line 1460 of file [AG2.for](#).

3.1.2.38 line()

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

Definition at line 109 of file [AG2.for](#).

3.1.2.39 locge()

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

Definition at line 2964 of file [AG2.for](#).

3.1.2.40 locle()

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

Definition at line 2982 of file [AG2.for](#).

3.1.2.41 logtix()

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

Definition at line 2043 of file [AG2.for](#).

3.1.2.42 loptim()

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

Definition at line 988 of file [AG2.for](#).

3.1.2.43 lwidth()

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

Definition at line 2733 of file [AG2.for](#).

3.1.2.44 mnmx()

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

Definition at line 881 of file [AG2.for](#).

3.1.2.45 monpos()

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

Definition at line 2160 of file [AG2.for](#).

3.1.2.46 notatec()

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

Definition at line 2619 of file [AG2.for](#).

3.1.2.47 npts()

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

Definition at line 155 of file [AG2.for](#).

3.1.2.48 numsetc()

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

Definition at line 2317 of file [AG2.for](#).

3.1.2.49 optim()

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

Definition at line 971 of file [AG2.for](#).

3.1.2.50 oubgc()

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

Definition at line [1488](#) of file [AG2.for](#).

3.1.2.51 place()

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

Definition at line [512](#) of file [AG2.for](#).

3.1.2.52 remlab()

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

Definition at line [2808](#) of file [AG2.for](#).

3.1.2.53 rescom()

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

Definition at line [3051](#) of file [AG2.for](#).

3.1.2.54 rgchek()

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

Definition at line [854](#) of file [AG2.for](#).

3.1.2.55 roundd()

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

Definition at line 3000 of file [AG2.for](#).

3.1.2.56 roundu()

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

Definition at line 3016 of file [AG2.for](#).

3.1.2.57 savcom()

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

Definition at line 3035 of file [AG2.for](#).

3.1.2.58 setwin()

```
subroutine setwin
```

Definition at line 622 of file [AG2.for](#).

3.1.2.59 sizel()

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

Definition at line 188 of file [AG2.for](#).

3.1.2.60 sizes()

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

Definition at line 177 of file [AG2.for](#).

3.1.2.61 slimx()

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

Definition at line 488 of file [AG2.for](#).

3.1.2.62 slimy()

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

Definition at line 500 of file [AG2.for](#).

3.1.2.63 spread()

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

Definition at line 2871 of file [AG2.for](#).

3.1.2.64 stepl()

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

Definition at line 166 of file [AG2.for](#).

3.1.2.65 steps()

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

Definition at line 131 of file [AG2.for](#).

3.1.2.66 symb1()

```
subroutine symb1 (  
    integer ipar )
```

Definition at line 120 of file [AG2.for](#).

3.1.2.67 symout()

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

Definition at line 1858 of file [AG2.for](#).

3.1.2.68 teksym()

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

Definition at line 1883 of file [AG2.for](#).

3.1.2.69 teksym1()

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

Definition at line 1931 of file [AG2.for](#).

3.1.2.70 tset()

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

Definition at line [2090](#) of file [AG2.for](#).

3.1.2.71 tset2()

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

Definition at line [2128](#) of file [AG2.for](#).

3.1.2.72 typck()

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

Definition at line [823](#) of file [AG2.for](#).

3.1.2.73 vbarst()

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

Definition at line [692](#) of file [AG2.for](#).

3.1.2.74 vlablc()

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

Definition at line [2644](#) of file [AG2.for](#).

3.1.2.75 width()

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

Definition at line [2692](#) of file [AG2.for](#).

3.1.2.76 xden()

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

Definition at line [312](#) of file [AG2.for](#).

3.1.2.77 xetyp()

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

Definition at line [596](#) of file [AG2.for](#).

3.1.2.78 xfrm()

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

Definition at line [390](#) of file [AG2.for](#).

3.1.2.79 xlab()

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

Definition at line [290](#) of file [AG2.for](#).

3.1.2.80 xlen()

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

Definition at line [364](#) of file [AG2.for](#).

3.1.2.81 xloc()

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

Definition at line 246 of file [AG2.for](#).

3.1.2.82 xloctp()

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

Definition at line 268 of file [AG2.for](#).

3.1.2.83 xmfrm()

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

Definition at line 438 of file [AG2.for](#).

3.1.2.84 xmtcs()

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

Definition at line 416 of file [AG2.for](#).

3.1.2.85 xneat()

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

Definition at line 202 of file [AG2.for](#).

3.1.2.86 xtics()

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

Definition at line 342 of file [AG2.for](#).

3.1.2.87 xtype()

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

Definition at line [544](#) of file [AG2.for](#).

3.1.2.88 xwidth()

```
subroutine xwidth (  
    integer ipar )
```

Definition at line [570](#) of file [AG2.for](#).

3.1.2.89 xzero()

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

Definition at line [224](#) of file [AG2.for](#).

3.1.2.90 yden()

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

Definition at line [327](#) of file [AG2.for](#).

3.1.2.91 yetyp()

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

Definition at line [609](#) of file [AG2.for](#).

3.1.2.92 yfrm()

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

Definition at line [403](#) of file [AG2.for](#).

3.1.2.93 ylab()

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

Definition at line [301](#) of file [AG2.for](#).

3.1.2.94 ylen()

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

Definition at line [377](#) of file [AG2.for](#).

3.1.2.95 yloc()

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

Definition at line [257](#) of file [AG2.for](#).

3.1.2.96 ylocrt()

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

Definition at line [279](#) of file [AG2.for](#).

3.1.2.97 ymdyd()

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

Definition at line [1405](#) of file [AG2.for](#).

3.1.2.98 ymfrm()

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

Definition at line 451 of file [AG2.for](#).

3.1.2.99 ymtcs()

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

Definition at line 427 of file [AG2.for](#).

3.1.2.100 yneat()

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

Definition at line 213 of file [AG2.for](#).

3.1.2.101 ytics()

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

Definition at line 353 of file [AG2.for](#).

3.1.2.102 ytype()

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

Definition at line 557 of file [AG2.for](#).

3.1.2.103 ywdth()

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

Definition at line 583 of file [AG2.for](#).

3.1.2.104 yzero()

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

Definition at line 235 of file [AG2.for](#).

3.2 AG2.for

```
00001 C> \file      AG2.for
00002 C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
00003 C> \version    (2022,284, x)
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> Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
00009 C> \note
00010 C> Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00011 C> SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00012 C>
00013 C> \~english
00014 C> Layer 2: scientific 2-D graphic subroutines
00015 C> \note
00016 C> The control character for exponent (originally -1) is now SOH=char(1)
00017 C> and for index (originally -2) STX=char(2).
00018 C>
00019 C> \~
00020 C> \note \verbatim
00021 C> Package:
00022 C> - AG2.for:      chart plotting routines
00023 C> - AG2Holerith.for: deprecated routines
00024 C> - AG2USR.for:   default user routines
00025 C> - G2dAG2.fd:    commonblock
00026 C> \endverbatim
00027 C
00028 C
00029 C Tektronix Advanced Graphics 2 - Version 2.x
00030 C
00031 C
00032 C Neuer Code in Fortran 77. Die Verwendung der im Manual dokumentierten
00033 C Unterprogramme bleibt unverändert, die direkte Manipulation von
00034 C Variablen des zugrundeliegenden Commonblockes ist jedoch nicht mehr
00035 C empfehlenswert. IBASEX (iPar) und IBASEY(iPar) mit ipar <>0,
00036 C IBASEC, COMGET und COMSET sollten in neuen Programmen nicht verwendet
00037 C werden.
00038 C
00039 C Die Zwischenspeicherung der Statusvariablen ueber
00040 C SAVCOM und RESCOM
00041 C und die Achsensteuerung ueber
00042 C IBASEX(0), IBASEY(0) und IOTHER
00043 C werden weiterhin unterstuetzt.
00044 C
00045 C Die Implementation der Unterprogramme COMGET und COMSET setzt die gleiche
00046 C Laenge von REAL und INTEGER-Variablen voraus.
00047 C
00048 C Da Holerithvariablen von modernen Compilern uneinheitlich unterstuetzt
00049 C werden (4Habcd entweder als gepackte Integervariable oder als Character-
00050 C variable interpretiert), wurden die folgenden Routinen angepasst:
00051 C - subroutine PLACE (Lit): Lit wird nur noch als Ordnungszahl (1..13)
00052 C und nicht mehr alternativ als Literal ('STD', 'UPH') interpretiert.
00053 C
00054 C subroutine LEAP (iyear): Die Schaltjahrkorrektur erfolgt nicht mehr
00055 C als SUBROUTINE ueber einen Common-Block, sondern direkt als
00056 C integer function LEAP (iyear) != 1: Schaltjahr, sonst 0
00057 C
00058 C Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00059 C SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00060 C
00061 C Intern erfolgt die Stringverarbeitung ueber Charaktervariablen als
00062 C nullterminierte C-Strings.
00063 C
00064 C Der User-API wurden die folgenden Unterprogramme als Charaktervarianten
00065 C der Original-Holerithroutinen hinzugefuegt:
00066 C - subroutine NUMSETC (fnum,nbase, outstr,fillstr)
00067 C - subroutine FONLYC (fnum,iwidth,idec, outstr,fillstr)
00068 C - subroutine EFORMC (fnum,iwidth,idec, outstr,fillstr)
00069 C - subroutine EXPOUTC (nbase,iexp, outstr,fillstr)
00070 C - subroutine ALFSETC (fnum,iwidth,labtyp,outstr)
00071 C - subroutine NOTATEC (IX,IY,LENCHR,IARRAY)
```

```

00072 C      - subroutine JUSTERC
00073 C
00074 C      - subroutine USESETC (fnum, iwidth, nbase, labstr)
00075 C
00076 C      subroutine MONPOS (nbase,iyl,dpos, spos) ! spos ist INTEGER
00077 C      subroutine GLINE (nbase,datapt,spos) ! spos ist INTEGER
00078 C
00079 C      Der Code ab Version 2.0 wird nicht mehr fuer CP/M entwickelt. Letzte
00080 C      unter CP/M compilierbare Version: (2006, 013, 1)
00081 C
00082 C      Zugehoerige Module:
00083 C      - AG2.FOR:      Basisfunktionen
00084 C      - AG2Holerith:  Veraltete Unterprogramme zur Wahrung der Kompatibilitaet
00085 C                    (Unterstuetzung Holerithvariablen und vektorisierter Zu-
00086 C                    griff auf den Commonblock)
00087 C      - AG2USR.FOR:   Userroutinen
00088 C      - G2dAG2.fd:    Commonblockdefinition
00089 C
00090 C
00091 C
00092 C      Ausgabe der Softwareversion
00093 C
00094 C      subroutine ag2lev (ilevel)
00095 C      implicit none
00096 C      integer ilevel(3)
00097 C
00098 C      call tcslev (ilevel) ! level(3)= System aus TCS
00099 C      ilevel(1)=2022      ! Aenderungsjahr
00100 C      ilevel(2)= 284      ! Aenderungstag
00101 C      return
00102 C      end
00103 C
00104 C
00105 C
00106 C
00107 C      Setzen allgemeiner Commonvariablen
00108 C
00109 C      subroutine line (ipar)
00110 C      implicit none
00111 C      integer ipar
00112 C      include 'G2dAG2.fd'
00113 C
00114 C      cline= ipar
00115 C      return
00116 C      end
00117 C
00118 C
00119 C
00120 C      subroutine symb1 (ipar)
00121 C      implicit none
00122 C      integer ipar
00123 C      include 'G2dAG2.fd'
00124 C
00125 C      csymb1= ipar
00126 C      return
00127 C      end
00128 C
00129 C
00130 C
00131 C      subroutine steps (ipar)
00132 C      implicit none
00133 C      integer ipar
00134 C      include 'G2dAG2.fd'
00135 C
00136 C      csteps= ipar
00137 C      return
00138 C      end
00139 C
00140 C
00141 C
00142 C      subroutine infin (par)
00143 C      implicit none
00144 C      real par
00145 C      include 'G2dAG2.fd'
00146 C
00147 C      if (par .gt. 0.) then
00148 C        cinfin= par
00149 C      end if
00150 C      return
00151 C      end
00152 C
00153 C
00154 C
00155 C      subroutine npts (ipar)
00156 C      implicit none
00157 C      integer ipar
00158 C      include 'G2dAG2.fd'

```

```
00159
00160     cnpts= ipar
00161     return
00162 end
00163
00164
00165
00166     subroutine step1 (ipar)
00167     implicit none
00168     integer ipar
00169     include 'G2dAG2.fd'
00170
00171     cstep1= ipar
00172     return
00173 end
00174
00175
00176
00177     subroutine sizes (par)
00178     implicit none
00179     real par
00180     include 'G2dAG2.fd'
00181
00182     csizes= par
00183     return
00184 end
00185
00186
00187
00188     subroutine sizel (par)
00189     implicit none
00190     real par
00191     include 'G2dAG2.fd'
00192
00193     csizel= par
00194     return
00195 end
00196
00197
00198
00199 C
00200 C   Setzen der achsenbezogenen Commonvariablen
00201 C
00202     subroutine xneat (ipar)
00203     implicit none
00204     integer ipar
00205     include 'G2dAG2.fd'
00206
00207     cxyneat(1) = ipar .ne. 0
00208     return
00209 end
00210
00211
00212
00213     subroutine yneat (ipar)
00214     implicit none
00215     integer ipar
00216     include 'G2dAG2.fd'
00217
00218     cxyneat(2) = ipar .ne. 0
00219     return
00220 end
00221
00222
00223
00224     subroutine xzero (ipar)
00225     implicit none
00226     integer ipar
00227     include 'G2dAG2.fd'
00228
00229     cxyzzero(1) = ipar .ne. 0
00230     return
00231 end
00232
00233
00234
00235     subroutine yzero (ipar)
00236     implicit none
00237     integer ipar
00238     include 'G2dAG2.fd'
00239
00240     cxyzzero(2) = ipar .ne. 0
00241     return
00242 end
00243
00244
00245
```



```

00246      subroutine xloc (ipar)
00247      implicit none
00248      integer ipar
00249      include 'G2dAG2.fd'
00250
00251      cxyloc(1)= ipar
00252      return
00253      end
00254
00255
00256
00257      subroutine yloc (ipar)
00258      implicit none
00259      integer ipar
00260      include 'G2dAG2.fd'
00261
00262      cxyloc(2)= ipar
00263      return
00264      end
00265
00266
00267
00268      subroutine xloctp (ipar)
00269      implicit none
00270      integer ipar
00271      include 'G2dAG2.fd'
00272
00273      cxyloc(1)= ipar+abs(cxysmax(2)-cxysmin(2))
00274      return
00275      end
00276
00277
00278
00279      subroutine ylocrt (ipar)
00280      implicit none
00281      integer ipar
00282      include 'G2dAG2.fd'
00283
00284      cxyloc(2)= ipar + abs(cxysmax(1)-cxysmin(1))
00285      return
00286      end
00287
00288
00289
00290      subroutine xlab (ipar)
00291      implicit none
00292      integer ipar
00293      include 'G2dAG2.fd'
00294
00295      cxylab(1)= ipar
00296      return
00297      end
00298
00299
00300
00301      subroutine ylab (ipar)
00302      implicit none
00303      integer ipar
00304      include 'G2dAG2.fd'
00305
00306      cxylab(2)= ipar
00307      return
00308      end
00309
00310
00311
00312      subroutine xden (ipar)
00313      implicit none
00314      integer ipar
00315      include 'G2dAG2.fd'
00316
00317      if ((ipar .ge. 0) .and. (ipar .le. 10)) then
00318        cxyden(1)= ipar
00319        cxytics(1)= 0
00320        cxymtcs(1)= 0
00321      end if
00322      return
00323      end
00324
00325
00326
00327      subroutine yden (ipar)
00328      implicit none
00329      integer ipar
00330      include 'G2dAG2.fd'
00331
00332      if ((ipar .ge. 0) .and. (ipar .le. 10)) then

```

```

00333      cxyden(2)= ipar
00334      cxytics(2)= 0
00335      cxymtcs(2)= 0
00336      end if
00337      return
00338      end
00339
00340
00341
00342      subroutine xtics (ipar)
00343      implicit none
00344      integer ipar
00345      include 'G2dAG2.fd'
00346
00347      cxytics(1)= abs(ipar)
00348      return
00349      end
00350
00351
00352
00353      subroutine ytics (ipar)
00354      implicit none
00355      integer ipar
00356      include 'G2dAG2.fd'
00357
00358      cxytics(2)= abs(ipar)
00359      return
00360      end
00361
00362
00363
00364      subroutine xlen (ipar)
00365      implicit none
00366      integer ipar
00367      include 'G2dAG2.fd'
00368
00369      if (ipar .ge. 0) then
00370          cxylen(1)= ipar
00371      end if
00372      return
00373      end
00374
00375
00376
00377      subroutine ylen (ipar)
00378      implicit none
00379      integer ipar
00380      include 'G2dAG2.fd'
00381
00382      if (ipar .ge. 0) then
00383          cxylen(2)= ipar
00384      end if
00385      return
00386      end
00387
00388
00389
00390      subroutine xfrm (ipar)
00391      implicit none
00392      integer ipar
00393      include 'G2dAG2.fd'
00394
00395      if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00396          cxyfrm(1)= ipar
00397      end if
00398      return
00399      end
00400
00401
00402
00403      subroutine yfrm (ipar)
00404      implicit none
00405      integer ipar
00406      include 'G2dAG2.fd'
00407
00408      if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00409          cxyfrm(2)= ipar
00410      end if
00411      return
00412      end
00413
00414
00415
00416      subroutine xmtcs (ipar)
00417      implicit none
00418      integer ipar
00419      include 'G2dAG2.fd'

```

```

00420
00421     cxymtcs(1)= abs(ipar)
00422     return
00423 end
00424
00425
00426
00427     subroutine ymtcs (ipar)
00428     implicit none
00429     integer ipar
00430     include 'G2dAG2.fd'
00431
00432     cxymtcs(2)= abs(ipar)
00433     return
00434 end
00435
00436
00437
00438     subroutine xmfrm (ipar)
00439     implicit none
00440     integer ipar
00441     include 'G2dAG2.fd'
00442
00443     if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00444         cxyxmfrm(1)= ipar
00445     end if
00446     return
00447 end
00448
00449
00450
00451     subroutine ymfrm (ipar)
00452     implicit none
00453     integer ipar
00454     include 'G2dAG2.fd'
00455
00456     if ((ipar .ge. 0) .and. (ipar .le. 6)) then
00457         cxyymfrm(2)= ipar
00458     end if
00459     return
00460 end
00461
00462
00463
00464     subroutine dlimx (xmin,xmax)
00465     implicit none
00466     real xmin,xmax
00467     include 'G2dAG2.fd'
00468
00469     cxydmin(1)= xmin
00470     cxydmax(1)= xmax
00471     return
00472 end
00473
00474
00475
00476     subroutine dlimy (ymin,ymax)
00477     implicit none
00478     real ymin,ymax
00479     include 'G2dAG2.fd'
00480
00481     cxydmin(2)= ymin
00482     cxydmax(2)= ymax
00483     return
00484 end
00485
00486
00487
00488     subroutine slimx (ixmin,ixmax)
00489     implicit none
00490     integer ixmin,ixmax
00491     include 'G2dAG2.fd'
00492
00493     cxysmin(1)= ixmin
00494     cxysmax(1)= ixmax
00495     return
00496 end
00497
00498
00499
00500     subroutine slimy (iymin,iymax)
00501     implicit none
00502     integer iymin,iymax
00503     include 'G2dAG2.fd'
00504
00505     cxysmin(2)= iymin
00506     cxysmax(2)= iymax

```

```

00507      return
00508  end
00509
00510
00511
00512  subroutine place (ipar)
00513    implicit none
00514    include 'G2dAG2.fd'
00515    integer ipar
00516
00517    integer postab (4,13)      ! Koordinaten des Zeichenbereiches
00518    data postab /150,900, 125,700,
00519    2          150,850, 525,700,
00520    3          150,850, 150,325,
00521    4          150,450, 525,700,
00522    5          650,950, 525,700,
00523    6          150,450, 150,325,
00524    7          650,950, 150,325,
00525    8          150,325, 525,700,
00526    9          475,650, 525,700,
00527    a          800,975, 525,700,
00528    1          150,325, 150,325,
00529    2          475,650, 150,325,
00530    3          800,975, 150,325/
00531    save postab
00532
00533    if ((ipar .ge. 1) .and. (ipar.le.13)) then
00534      cxysmin(1)= postab(1,ipar)
00535      cxysmax(1)= postab(2,ipar)
00536      cxysmin(2)= postab(3,ipar)
00537      cxysmax(2)= postab(4,ipar)
00538    end if
00539    return
00540  end
00541
00542
00543
00544  subroutine xtype (ipar)
00545    implicit none
00546    integer ipar
00547    include 'G2dAG2.fd'
00548
00549    if ((ipar .ge. 1) .and. (ipar .le. 8)) then
00550      cxytype(1)= ipar
00551    end if
00552    return
00553  end
00554
00555
00556
00557  subroutine ytype (ipar)
00558    implicit none
00559    integer ipar
00560    include 'G2dAG2.fd'
00561
00562    if ((ipar .ge. 1) .and. (ipar .le. 8)) then
00563      cxytype(2)= ipar
00564    end if
00565    return
00566  end
00567
00568
00569
00570  subroutine xwidth (ipar)
00571    implicit none
00572    integer ipar
00573    include 'G2dAG2.fd'
00574
00575    if (ipar .ge. 0) then
00576      cxywidth(1)= ipar
00577    end if
00578    return
00579  end
00580
00581
00582
00583  subroutine ywidth (ipar)
00584    implicit none
00585    integer ipar
00586    include 'G2dAG2.fd'
00587
00588    if (ipar .ge. 0) then
00589      cxywidth(2)= ipar
00590    end if
00591    return
00592  end
00593

```

```

00594
00595
00596     subroutine xetyp (ipar)
00597     implicit none
00598     integer ipar
00599     include 'G2dAG2.fd'
00600
00601     if ((ipar .ge. 0) .and. (ipar .le. 4)) then
00602         cxyetyp(1)= ipar
00603     end if
00604     return
00605 end
00606
00607
00608
00609     subroutine yetyp (ipar)
00610     implicit none
00611     integer ipar
00612     include 'G2dAG2.fd'
00613
00614     if ((ipar .ge. 0) .and. (ipar .le. 4)) then
00615         cxyetyp(2)= ipar
00616     end if
00617     return
00618 end
00619
00620
00621
00622     subroutine setwin
00623     implicit none
00624     include 'G2dAG2.fd'
00625
00626     call twindo (cxysmin(1),cxysmax(1), cxysmin(2),cxysmax(2))
00627     call dwindo (cxydmin(1),cxydmax(1), cxydmin(2),cxydmax(2))
00628     if (cxytype(1) .eq. 2) then
00629         if (cxytype(2) .eq. 2) then
00630             call logtrn (3)
00631         else
00632             call logtrn (1)
00633         end if
00634     else if (cxytype(2) .eq. 2) then
00635         call logtrn (2)
00636     else
00637         call lintrn
00638     end if
00639     return
00640 end
00641
00642
00643
00644     subroutine dinitx
00645     implicit none
00646     include 'G2dAG2.fd'
00647
00648     cxydmin(1)= 0.           ! Datenbereich
00649     cxydmax(1)= 0.
00650     cxywidth(1)= 0          ! Dezimalstellen
00651     cxydec(1)= 0            ! Dezimalstellen
00652     cxyepon(1)= 0           ! Exponent Label
00653     return
00654 end
00655
00656
00657
00658     subroutine dinity
00659     implicit none
00660     include 'G2dAG2.fd'
00661
00662     cxydmin(2)= 0.           ! Datenbereich
00663     cxydmax(2)= 0.
00664     cxywidth(2)= 0          ! Dezimalstellen
00665     cxydec(2)= 0            ! Dezimalstellen
00666     cxyepon(2)= 0           ! Exponent Label
00667     return
00668 end
00669
00670
00671
00672     subroutine hbarst (ishade,iwbar,idbar)
00673     implicit none
00674     integer ishade,iwbar,idbar
00675     include 'G2dAG2.fd'
00676
00677     cline= -3
00678     if ((ishade .ge. 0).and. (ishade .le. 15)) csymb1= ishade
00679     csizes= real(idbar)
00680     csizel= real(iwbar)

```

```

00681
00682     if (cxyfrm(2) .eq. 5) then
00683         cxyfrm(2)= 2
00684     else if (cxyfrm(2) .eq. 6) then
00685         cxyfrm(2)= 1
00686     end if
00687     return
00688 end
00689
00690
00691
00692 subroutine vbarst (ishade,iwbar,idbar)
00693 implicit none
00694 integer ishade,iwbar,idbar
00695 include 'G2dAG2.fd'
00696
00697 cline= -2
00698 if ((ishade .ge. 0) .and. (ishade .le. 15)) csymb1= ishade
00699 csizes= real(idbar)
00700 csizel= real(iwbar)
00701 if (cxyfrm(1) .eq. 5) then
00702     cxyfrm(1)= 2
00703 else if (cxyfrm(1) .eq. 6) then
00704     cxyfrm(1)= 1
00705 end if
00706 return
00707 end
00708
00709
00710
00711 C
00712 C Berechnung der Commonvariablen
00713 C
00714 subroutine binitt
00715 implicit none
00716 integer ih
00717 include 'G2dAG2.fd'
00718
00719 cline= 0
00720 csymb1= 0
00721 csteps= 1
00722 cinfin= 1.e30
00723 cnpts= 0
00724 cstepl= 1
00725 cnumbr= 0
00726 csizes= 1.
00727 csizel= 1.
00728
00729 cxyneat(1)= .true.
00730 cxyneat(2)= .true.
00731 cxyzero(1)= .true.
00732 cxyzero(2)= .true.
00733 cxyloc(1)= 0
00734 cxyloc(2)= 0
00735 cxylab(1)= 1
00736 cxylab(2)= 1
00737 cxyden(1)= 8
00738 cxyden(2)= 8
00739 cxytics(2)= 0
00740 cxytics(2)= 0
00741
00742 call csize (ih,cxylen(1))
00743 cxylen(2)= cxylen(1)
00744
00745 cxyfrm(1)= 5
00746 cxyfrm(2)= 5
00747 cxymtcs(1)= 0
00748 cxymtcs(2)= 0
00749 cxymfrm(1)= 2
00750 cxymfrm(2)= 2
00751 cxydec(1)= 0
00752 cxydec(2)= 0
00753 cxydmin(1)= 0.
00754 cxydmin(2)= 0.
00755 cxydmax(1)= 0.
00756 cxydmax(2)= 0.
00757
00758 cxysmin(1)= 150
00759 cxysmin(2)= 125
00760 cxysmax(1)= 900
00761 cxysmax(2)= 700
00762
00763 cxytype(1)= 1
00764 cxytype(2)= 1
00765 cxylsig(1)= 0
00766 cxylsig(2)= 0
00767 cxywidth(1)= 0

```

```

00768      cxywidth(2)= 0
00769      cxyepon(1)= 0
00770      cxyepon(2)= 0
00771      cxystep(1)= 1
00772      cxystep(2)= 1
00773      cxystag(1)= 1
00774      cxystag(2)= 1
00775      cxyetyp(1)= 0
00776      cxyetyp(2)= 0
00777      cxybeg(1)= 0
00778      cxybeg(2)= 0
00779      cxyend(1)= 0
00780      cxyend(2)= 0
00781      cxymbeg(1)= 0
00782      cxymbeg(2)= 0
00783      cxymend(1)= 0
00784      cxymend(2)= 0
00785      cxyamin(1)= 0.
00786      cxyamin(2)= 0.
00787      cxyamax(1)= 0.
00788      cxyamax(2)= 0.
00789      return
00790      end
00791
00792
00793
00794 C
00795 C  Datenanalyse
00796 C
00797
00798      subroutine check (x,y)
00799      implicit none
00800      real x(5),y(5)
00801      include 'G2dAG2.fd'
00802
00803      external SPREAD ! External wg. Namenskonflikt FTN90-Intrinsic
00804
00805      call typck (1,x)
00806      call rgchek(1,x)
00807      call optim (1)
00808      call width (1)
00809      if (cxystag(1) .eq. 1) call spread (1)
00810      call tset (1)
00811
00812      call typck (2,y)
00813      call rgchek(2,y)
00814      call optim(2)
00815      call width(2)
00816      if (cxystag(2) .eq. 1) call spread (2)
00817      call tset (2)
00818      return
00819      end
00820
00821
00822
00823      subroutine typck (ixy, arr)
00824      implicit none
00825      integer ixy
00826      real arr(5)
00827      integer i
00828      include 'G2dAG2.fd'
00829
00830      if ((cxytype(ixy) .lt. 3) .or. (nint(arr(1)) .lt. -1 )) then
00831      if ((cnpts .ne. 0) .or. (nint(arr(1)) .ne. -2) ) return
00832      i= nint(arr(3))
00833      if ( i .eq. 1) then
00834      cxytype(ixy)= 8
00835      else if ( i .eq. 4) then
00836      cxytype(ixy)= 7
00837      else if ( i .eq. 12) then
00838      cxytype(ixy)= 6
00839      else if ( i .eq. 13) then
00840      cxytype(ixy)= 5
00841      else if ( i .eq. 52) then
00842      cxytype(ixy)= 4
00843      else if ( i .eq. 365) then
00844      cxytype(ixy)= 3
00845      end if
00846      else
00847      cxytype(ixy)= 1
00848      end if
00849      return
00850      end
00851
00852
00853
00854      subroutine rgchek (ixy,arr)

```

```

00855      implicit none
00856      integer ixy
00857      real arr(5)
00858      real amin, amax
00859      include 'G2dAG2.fd'
00860
00861      if (cxydmax(ixy) .eq. cxydmin(ixy)) then ! Bereich schon bestimmt?
00862      if (cxyzzero(ixy)) then ! Nullpunktunterdrueckung?
00863      amin= cinfin
00864      else
00865      amin= 0.
00866      end if
00867      amax= -amin
00868      call mnmx (arr, amin, amax)
00869      if (amax .eq. amin) then
00870      amin= amin - 0.5
00871      amax= amax + 0.5
00872      end if
00873      cxydmin(ixy)= amin
00874      cxydmax(ixy)= amax
00875      end if
00876      return
00877      end
00878
00879
00880
00881      subroutine mnmx (arr,amin,amax)
00882      implicit none
00883      real arr(5), amin,amax, aminmax
00884      integer i, itype, nstart,nlim
00885      include 'G2dAG2.fd'
00886
00887      if (cnpts .eq. 0) then                                     ! Tek Standard-Format
00888      nlim= nint(arr(1)) + 1
00889      nstart= 2
00890      else
00891      nlim= cnpts
00892      nstart= 1
00893      end if
00894      if ((arr(1) .lt. 0.) .and. (cnpts .eq. 0)) then ! Kurzformate
00895      itype= abs(arr(1))
00896      if (itype .eq. 1) then
00897      aminmax= arr(3) + (arr(2)-1.) * arr(4)
00898      amin= aminl(arr(3),aminmax,amin)
00899      amax= amaxl(arr(3),aminmax,amax)
00900      else if (itype .eq. 2) then
00901      call cmnmx (arr,amin,amax)
00902      else
00903      call umnmx (arr,amin,amax)
00904      end if
00905      else                                     ! Langformate
00906      if (nstart .le. nlim) then
00907      do 100 i= nstart, nlim
00908      if (arr(i) .lt. cinfin) then
00909      if (arr(i).lt. amin) amin= arr(i)
00910      if (arr(i).gt. amax) amax= arr(i)
00911      end if
00912 100    continue
00913      end if
00914      end if
00915      return
00916      end
00917
00918
00919
00920      subroutine cmnmx (arr,amin,amax)
00921      implicit none
00922      real arr(5), amin, amax
00923      integer nTage, iStUBGC, nIntv, iadj, imin,imax
00924      integer minTg,minJr, maxTg,maxJr
00925
00926
00927      nintv= nint(arr(3))
00928      if ((nintv .eq. 52).or.(nintv .eq. 13).or.(nintv .eq. 4)) then
00929      if (nintv .eq. 52) then                                ! Wochen
00930      ntage=7
00931      else if (nintv .eq. 13) then                            ! 28 Tagemonat
00932      ntage= 28
00933      else if (nintv .eq. 4) then                             ! Quartal
00934      ntage=91
00935      end if
00936      call iubgc (nint(arr(4)),1, istubgc) ! Start: Jahr=arr(4), Tag=1
00937      iadj= mod(istubgc,7)
00938      if (iadj .gt. 3) iadj=iadj-7
00939      imin= istubgc-iadj + nint(arr(5))*ntage ! Min= f(Startjahr,StartIntervall)
00940      imax= imin + nint(arr(2))*ntage
00941

```



```

00942     else
00943         if (nintv .eq. 1) then ! Jahre
00944             mintg= 1
00945             maxtg= 1
00946             minjr= nint(arr(4))+1
00947             maxjr= nint(arr(4)+arr(2))
00948         else if ( nintv .eq. 12) then ! Monate
00949             call ymdyd (minjr,mintg, nint(arr(4)),nint(arr(5))+1,1)
00950             call ymdyd (maxjr,maxtg, nint(arr(4)),nint(arr(5)+arr(2)),1)
00951         else if ( nintv .eq. 365) then ! Tage
00952             minjr= nint(arr(4))
00953             mintg= nint(arr(5))
00954             maxjr= nint(arr(4))
00955             maxtg= nint(arr(5)+arr(2)) -1
00956         end if
00957         call iubgc (minjr,mintg, imin)
00958         call iubgc (maxjr,maxtg, imax)
00959     end if
00960     if (real(imax) .gt. amax) amax= real(imax)
00961     if (real(imin) .lt. amin) amin= real(imin)
00962     return
00963 end
00964
00965
00966
00967 C
00968 C Ticmarkoptimierung
00969 C
00970
00971 subroutine optim (ixy)
00972     implicit none
00973     integer ixy
00974     include 'G2dAG2.fd'
00975
00976     if (cxytype(ixy) .eq. 2) cxylab(ixy)= 2
00977     if (cxylab(ixy) .eq. 2) cxylab(ixy)= cxytype(ixy)
00978     if (cxytype(ixy) .le. 2) then
00979         call loptim (ixy) ! Tic-Mark Optimierung fuer lineare und log. Daten
00980     else
00981         call coptim (ixy) ! Tic-Mark Optimierung fuer Kalenderdaten
00982     end if
00983     return
00984 end
00985
00986
00987
00988 subroutine loptim (ixy)
00989     implicit none
00990     integer ixy ,i, labtyp, ntics, lsig, mtcs
00991     real dataint, amin,amax, aminor,amaxor, sigfac
00992     integer idataint
00993     integer mintic
00994     integer LINWDT, LINHGT
00995     real ROUND, ROUNDU
00996     include 'G2dAG2.fd'
00997
00998     labtyp=abs( cxylab(ixy)) ! <0: Userlabel
00999     if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp
01000
01001     amin= cxydmin(ixy)
01002     amax= cxydmax(ixy)
01003     ntics= abs(cxytics(ixy)) ! Anzahl >=1, 0= Flag fuer autoscale
01004     mintic= 0
01005
01006     if (labtyp .eq. 2) then ! logarithmische Achsen
01007         amin= log10(max(amin,1./cinf)) + 1.e-7 ! > 0 => log10 definiert
01008         amax= log10(amax)
01009     end if
01010
01011     aminor= amin
01012     amaxor= amax
01013
01014     if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
01015         if (ixy.eq.1) then
01016             i= linwdt(8) ! 100 + LINWDT(3)
01017         else
01018             i= linhgt(3) ! 50 + LINHGT(3)
01019         end if
01020         ntics= (cxysmax(ixy) - cxysmin(ixy)) / i
01021         if (ntics .lt. 1) ntics= 1
01022     end if
01023     dataint= abs(amax-amin) / real(ntics)
01024
01025 310 continue ! repeat...
01026     if (labtyp .eq. 2) dataint= roundu(dataint,1.) ! logarithmische Achsen
01027     lsig= roundd(log10(dataint),1.) ! Anzahl signifikanter Nachkommastellen
01028     sigfac=10.**(lsig)

```

```

01029      if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
01030      if (labtyp .ne. 2) then ! nicht bei log. Achsen
01031          if ((dataint/sigfac) .le. 1.) then
01032              dataint= 1. * sigfac
01033              mintic= 10
01034          else if ((dataint/sigfac) .le. 2.) then
01035              dataint= 2. * sigfac
01036              mintic= 2
01037          else if ((dataint/sigfac) .le. 2.5) then
01038              dataint= 2.5 * sigfac
01039              mintic= 5
01040              lsig=lsig-1
01041          else if ((dataint/sigfac) .le. 5.) then
01042              dataint= 5. * sigfac
01043              mintic= 5
01044          else if ((dataint/sigfac) .le. 10.) then
01045              dataint= 10. * sigfac
01046              mintic= 10
01047              lsig=lsig+1
01048          else
01049              dataint= cinfin
01050              mintic= 0
01051          end if
01052      end if ! log. Achse
01053      else ! .not. neat
01054          lsig=lsig-2
01055      end if
01056      if (lsig .ge. 0) lsig=lsig+1
01057      if (cxyneat(ixy) .or. (labtyp .eq. 2) ) then ! ... until
01058          amin= roundd(amin+.01*sigfac,dataint) !   runde auf TicIntervall
01059          amax= roundu(amax-.01*sigfac,dataint) ! .01*sigfac= Genauigkeit Plot
01060          ntics= int(abs(amax-amin)/dataint+.0001)
01061          if (cxytics(ixy) .ne. 0) then ! until: ntics nicht vorbesetzt oder = vorbesetzt
01062              if (abs(cxytics(ixy)) .lt. ntics) then
01063                  dataint= dataint * 1.1
01064                  amin=aminor
01065                  amax=amaxor
01066                  goto 310 ! noch eine Iterationsschleife
01067              else if (abs(cxytics(ixy)) .gt. ntics) then
01068                  ntics= abs(cxytics(ixy))
01069                  amax= amin + real(ntics) * dataint
01070              end if ! abs(cxytics(ixy)) .eq. ntics: no action
01071          end if
01072      end if
01073      cxytics(ixy)= ntics
01074
01075      if ((cxymtcs(ixy) .eq. 0) .and. (cxyden(ixy) .ge. 6)) then ! unbesetzt oder wenig TICS
01076          mtcs= mintic ! Bestimmung Minor TicMarcs
01077          if (mtcs .eq. 10) .or. (labtyp .eq. 2)) then
01078              if (cxyden(ixy) .lt. 9) mtcs=5
01079              if (cxyden(ixy) .lt. 7) mtcs=2
01080          if (labtyp .eq. 2) then ! log. Achsen
01081              idataint= nint(dataint)
01082              if (idataint .ne. 1) then ! mehrere Achsenintervalle
01083                  i= 1
01084          320          continue ! repeat...
01085                      mtcs= idataint/i
01086                      if ((mtcs*i .ne. idataint) .and. (i .lt. (idataint-1))) then ! ...until
01087                          i= i+1
01088                          goto 320
01089                      else if (mtcs .gt. 10 ) then
01090                          mtcs= 0 ! Failure
01091                      end if
01092              else ! einzelne logarithmische Dekade
01093                  if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 100* ntics) mtcs=-1 ! logarithm. Tics
01094                  if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 20* linhgt(1)) mtcs=-2 ! Label
01095              end if
01096          end if
01097      end if
01098      cxymtcs(ixy)= mtcs
01099      end if
01100
01101      cxylsig(ixy)= lsig
01102      cxyamin(ixy)= amin
01103      cxyamax(ixy)= amax
01104      if (labtyp .eq. 2) then ! logarithmische Achsen: Wiederherstellung der Originalwerte
01105          amax=10.**amax
01106          amin=10.**amin
01107      end if
01108      cxydmin(ixy)= amin
01109      cxydmax(ixy)= amax
01110      return
01111      end
01112
01113
01114
01115      subroutine coptim (ixy)

```

```

01116      implicit none
01117      integer ixy , labtyp, ntics
01118      real dataint, amin,amax, aminor,amaxor
01119      integer LINWDT
01120      real ROUND, ROUNDU
01121      include 'G2dAG2.fd'
01122
01123      if (cxytics(ixy) .eq. 1) cxytics(ixy)= 2 ! Minimum manuelle Ticwahl: 2
01124      labtyp=abs( cxylab(ixy)) ! <0: Userlabel
01125      if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp
01126      amin= cxydmin(ixy)
01127      amax= cxydmax(ixy)
01128      call calcon (amin,amax,labtyp,.true.) ! Konvertiere UBGC -> Labelzeiteinheit
01129      ntics= cxytics(ixy)
01130      aminor=amin
01131      amaxor=amax
01132      if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
01133        ntics= (cxysmax(ixy) - cxysmin(ixy)) / (25 + linwdt(1))
01134        if (ntics .lt. 2) ntics= 2
01135      end if
01136      dataint= abs(amax-amin) / real(ntics)
01137
01138      if (cxyneat(ixy)) then ! Achsentheilung aus Tabelle
01139 310    continue ! repeat...
01140        if (cxytics(ixy) .eq. 0) then ! keine manuelle Belegung erfolgt
01141          if (labtyp.eq.3) then ! Labeltyp: Tage
01142            if (dataint .le. 1.) then
01143              dataint= 1.
01144            else if (dataint .le. 7.) then
01145              dataint= 7.
01146            else if (dataint .le. 14.) then
01147              dataint= 14.
01148            else if (dataint .le. 28.) then
01149              dataint= 28.
01150            else if (dataint .le. 56.) then
01151              dataint= 56.
01152            else if (dataint .le. 128.) then
01153              dataint= 128.
01154            end if ! dataint > 128 -> unveraendert
01155          else if (labtyp.eq.4) then ! Labeltyp: Wochen
01156            if (dataint .le. 1.) then
01157              dataint= 1.
01158            else if (dataint .le. 2.) then
01159              dataint= 2.
01160            else if (dataint .le. 4.) then
01161              dataint= 4.
01162            else if (dataint .le. 8.) then
01163              dataint= 8.
01164            else if (dataint .le. 16.) then
01165              dataint= 16.
01166            else if (dataint .le. 26.) then
01167              dataint= 26.
01168            else if (dataint .le. 52.) then
01169              dataint= 52.
01170            else if (dataint .le. 104.) then
01171              dataint= 104.
01172            end if ! dataint -> unveraendert
01173          else if (labtyp.eq.5) then ! Labeltyp: Kalenderabschnitte
01174            if (dataint .le. 1.) then
01175              dataint= 1.
01176            else if (dataint .le. 2.) then
01177              dataint= 2.
01178            else if (dataint .le. 13.) then
01179              dataint= 13.
01180            else if (dataint .le. 26.) then
01181              dataint= 26.
01182            else if (dataint .le. 52.) then
01183              dataint= 52.
01184            end if ! dataint -> unveraendert
01185          else if (labtyp.eq.6) then ! Labeltyp: Monate
01186            if (dataint .le. 1.) then
01187              dataint= 1.
01188            else if (dataint .le. 2.) then
01189              dataint= 2.
01190            else if (dataint .le. 3.) then
01191              dataint= 3.
01192            else if (dataint .le. 4.) then
01193              dataint= 4.
01194            else if (dataint .le. 6.) then
01195              dataint= 6.
01196            else if (dataint .le. 12.) then
01197              dataint= 12.
01198            else if (dataint .le. 24.) then
01199              dataint= 24.
01200            else if (dataint .le. 36.) then
01201              dataint= 36.
01202            end if ! dataint -> unveraendert

```

```

01203     else if (labtyp.eq.7) then ! Labeltyp: Quartale
01204         if (dataint .le. 1.) then
01205             dataint= 1.
01206         else if (dataint .le. 2.) then
01207             dataint= 2.
01208         else if (dataint .le. 4.) then
01209             dataint= 4.
01210         else if (dataint .le. 8.) then
01211             dataint= 8.
01212         else if (dataint .le. 12.) then
01213             dataint= 12.
01214         else if (dataint .le. 16.) then
01215             dataint= 16.
01216         else if (dataint .le. 24.) then
01217             dataint= 24.
01218         end if ! dataint -> unveraendert
01219     else if (labtyp.eq.8) then ! Labeltyp: Jahre
01220         if (dataint .le. 1.) then
01221             dataint= 1.
01222         else if (dataint .le. 2.) then
01223             dataint= 2.
01224         else if (dataint .le. 5.) then
01225             dataint= 5.
01226         else if (dataint .le. 10.) then
01227             dataint= 10.
01228         else if (dataint .le. 20.) then
01229             dataint= 20.
01230         else if (dataint .le. 50.) then
01231             dataint= 50.
01232         else if (dataint .le. 100.) then
01233             dataint= 100.
01234         end if ! dataint -> unveraendert
01235     end if ! labtyp 3..8
01236 end if ! manuelle Vorbesetzung
01237 amin= roundd(amin,dataint) ! runde auf TicIntervall
01238 amax= roundu(amax,dataint)
01239 ntics= ifix(abs(amax-amin)/dataint+.0001)
01240 if (ntics .eq. 0) ntics = 2
01241 if(cxytics(ixy) .ne. 0) then ! until: ntics nicht oder = vorbesetzt
01242     if(abs(cxytics(ixy)) .lt. ntics) then ! Verringere Ticanzahl
01243         dataint= dataint * 1.1
01244         amin=aminor
01245         amax=amaxor
01246         goto 310 ! noch eine Iterationsschleife
01247     else if (abs(cxytics(ixy)) .gt. ntics) then ! Vergroessere Ticanzahl
01248         ntics= abs(cxytics(ixy))
01249         amax= amin + real(ntics) * dataint
01250     end if ! abs(cxytics(ixy)) .eq. ntics: no action
01251 end if ! Ende der Schleife
01252 end if ! neat
01253 cxytics(ixy)= ntics
01254 cxylsig(ixy)= 0
01255 cxyamin(ixy)= amin
01256 cxyamax(ixy)= amax
01257 call calcon (amin,amax,labtyp,.false.) ! Labelzeiteinheit -> UBGC
01258 cxydmin(ixy)= amin
01259 cxydmax(ixy)= amax
01260 return
01261 end
01262
01263
01264
01265 C
01266 C Kalenderroutinen
01267 C
01268
01269
01270
01271 real function calpnt (arr,i)
01272 implicit none
01273 integer i
01274 real arr(5)
01275 integer iy,idays, itmp
01276 integer icltyp, istyr, istper, iubgl, iweekl, nodays
01277 save icltyp, istyr, istper, iubgl, iweekl, nodays
01278
01279 if (i .eq. 1) then ! 1. Datenpunkt: Formatanalyse, Parameterberechnung
01280     istyr= nint(arr(4))
01281     istper= nint(arr(5))
01282     itmp= nint(arr(3)) ! Laenge Intervall in Tagen
01283     if (itmp .eq. 12) then ! Zeitintervall Monat
01284         icltyp= 2
01285     else if (itmp .eq. 365) then ! Zeitintervall Tage
01286         icltyp=3
01287     call iubgc (istyr,istper,iubgl)
01288     else if (itmp .eq. 52) then ! Zeitintervall Wochen
01289         icltyp= 4

```

```

01290      nodays= 7
01291      else if (itmp .eq. 13) then ! Zeitintervall 4 Wochen
01292      icltyp= 5
01293      nodays= 28
01294      else if (itmp .eq. 4) then ! Zeitintervall Quartal
01295      icltyp= 6
01296      nodays= 91
01297      else ! Zeitintervall Jahre
01298      icltyp= 1
01299      end if
01300      if (icltyp .ge. 4) then
01301      call iubgc (istyr,1,iubg1)
01302      itmp= mod(iubg1+1,7)
01303      if(itmp .gt. 3) itmp= itmp-7
01304      iweek1= iubg1-itmp
01305      iubg1= iweek1+(istper-1)*nodays
01306      end if
01307      end if ! Ende Initialisierung, jetzt Berechnung
01308
01309      if (icltyp .eq. 1) then ! Zeitintervall Jahr
01310      call iubgc (istyr+i,1,iubg1)
01311      calpnt= iubg1
01312      else if (icltyp .eq. 2) then ! Zeitintervall Monat
01313      call ymdyd (iy,idades,istyr,istper+i,1)
01314      call iubgc (iy,idades,iubg1)
01315      calpnt= iubg1 ! Zeitintervall Tage
01316      else if (icltyp .eq. 3) then
01317      calpnt= iubg1+i-1
01318      else ! Zeitintervall Wochen oder 4 Wochen
01319      calpnt= iweek1+(istper-1+i)*nodays
01320      end if
01321      return
01322      end
01323
01324
01325
01326      subroutine calcon (amin,amax,labtyp,ubgc)
01327      implicit none
01328      real amin, amax
01329      integer labtyp
01330      logical ubgc
01331      integer iubg1, iubg2, iday1, iadj, id, month1,month2 , imin,imax
01332      real dimin, dimax
01333      integer iweek1
01334      real fnoday
01335      integer iy1,iy2, iy3,iy4, idays
01336      save iweek1, fnoday
01337      save iy1,iy2, iy3, iy4, idays
01338
01339      real ROUND, ROUNDU
01340
01341      if (labtyp .le. 3) return ! nicht Kalender, bzw.Tage: keine Transformation
01342
01343      if (ubgc) then ! Konvertierung UBGC in Labeltype
01344      if ( (labtyp .eq. 4).or.(labtyp .eq. 5).or.(labtyp .eq. 7) ) then
01345      if (labtyp .eq. 4) fnoday= 7.
01346      if (labtyp .eq. 5) fnoday= 28.
01347      if (labtyp .eq. 7) fnoday= 91.
01348      iubg1=amin
01349      iubg2=amax
01350      call iubgc (iy1,idades,iubg1) ! Wochenanfang der 1.KW Startjahr
01351      iday1=iubg1-idades+1
01352      iadj=mod(iday1+1,7)
01353      if(iadj .gt. 3) iadj=iadj-7
01354      iweek1= iday1-iadj ! Merken in iweek1
01355      dimin= roundd(real(iubg1-iweek1),fnoday)
01356      dimin= dimin/fnoday+1.
01357      call iubgc (iy2,idades,iubg2)
01358      dimax= roundu(real(iubg2-iweek1),fnoday)
01359      dimax= dimax/fnoday
01360      else if (labtyp .eq. 6) then
01361      call iubgc (iy1,idades,nint(amin))
01362      call ydymd (iy1,idades,iy3,month1,id)
01363      dimin= month1
01364      call iubgc (iy2,idades,nint(amax))
01365      call ydymd (iy2,idades,iy4,month2,id)
01366      dimax= (iy4-iy3)*12+month2
01367      if(id .gt. 1) dimax=dimax+1.
01368      else if (labtyp .eq. 8) then
01369      call iubgc (iy1,idades,nint(amin))
01370      dimin= iy1
01371      call iubgc(iy2,idades,nint(amax))
01372      dimax= iy2
01373      if(idays .gt. 1) dimax=dimax+1.
01374      end if
01375      amin= dimin-1.
01376      amax= dimax-1.

```

```

01377         return
01378
01379     else ! Konvertierung Labeltype in UBGC
01380         amin=amin+1.
01381         amax=amax+1.
01382         if ((labtyp .eq. 4).or.(labtyp .eq. 5).or.(labtyp .eq. 7)) then
01383             amin= iweek1 + (nint(amin)-1) * nint(fnoday)
01384             amax= iweek1+(nint(amax)-1)*nint(fnoday)
01385         else if (labtyp .eq. 6)then
01386             iy4= iy3
01387             call ymdyd (iy1,idays,iy3,nint(amin),1)
01388             call iubgc (iy1,idays,imin)
01389             amin= imin
01390             call ymdyd (iy2,idays,iy4,nint(amax),1)
01391             call iubgc (iy2,idays,imax)
01392             amax= imax
01393         else if (labtyp .eq. 8) then
01394             call iubgc (nint(amin),1,imin)
01395             amin= imin
01396             call iubgc (nint(amax),1,imax)
01397             amax= imax
01398         end if
01399     endif
01400     return
01401 end
01402
01403
01404
01405 subroutine ymdyd (iJulYrOut,iJulDayOut,
01406 1 iGregYrIn,iGregMonIn,iGregDayIn)
01407     implicit none
01408     integer iJulYrOut,iJulDayOut, iGregYrIn,iGregMonIn,iGregDayIn
01409     integer iJulYrIn,iJulDayIn, iGregYrOut,iGregMonOut,iGregDayOut
01410     integer iMon, LEAP
01411     integer iDatTab(12)
01412     save idattab
01413     data idattab /0,31,59,90,120,151,181,212,243,273,304,334/
01414
01415     ijulyrout= igregyrin
01416     imon= igregmonin
01417 100 if (imon .lt. 1) then ! while iMon .not. in [1..12]
01418         imon= imon + 12
01419         ijulyrout= ijulyrout-1
01420         goto 100
01421     else if (imon .gt. 12) then
01422         imon= imon -12
01423         ijulyrout= ijulyrout+1
01424         goto 100
01425     end if
01426     ijuldayout= igregdayin + idattab(imon)
01427     if (imon .gt.2) ijuldayout= ijuldayout + leap(ijulyrout)
01428     return
01429
01430
01431     entry ydynd(ijulyrin,ijuldayin,
01432 1 igregyrout,igregmonout,igregdayout)
01433
01434     igregdayout= ijuldayin
01435     igregyrout= ijulyrin
01436 110 if (igregdayout .lt. 1) then ! while iGregDayOut .not. in [1..365(366)]
01437         igregyrout= igregyrout-1
01438         igregdayout= igregdayout + 365 + leap(igregyrout)
01439         goto 110
01440     else if (igregdayout .gt. 365+ leap(igregyrout)) then
01441         igregyrout= igregyrout+1
01442         igregdayout= igregdayout - 365 - leap(igregyrout)
01443         goto 110
01444     end if
01445
01446     igregmonout= int( real(igregdayout)/29.5+1.)
01447     if (igregdayout .le. idattab(igregmonout)) then
01448         if ((igregmonout .le. 2) .or.
01449 1 (igregdayout.le.(idattab(igregmonout)+leap(igregyrout)))) then
01450         igregmonout= igregmonout-1
01451     end if
01452     end if
01453     igregdayout= igregdayout- idattab(igregmonout)
01454     if (igregmonout .gt. 2) igregdayout= igregdayout -leap(igregyrout)
01455     return
01456 end
01457
01458
01459
01460 integer function leap (iyear)
01461     implicit none
01462     integer iyear
01463     if ( (mod(iyear,4) .eq. 0) .and.

```

```

01464      1      ((mod(iyear,100).ne.0) .or. (mod(iyear,400).eq.0)) ) then
01465          leap= 1
01466      else
01467          leap= 0
01468      end if
01469      return
01470  end
01471
01472
01473
01474  subroutine iubgc(iyear,iday, iubgcO)
01475      implicit none
01476      integer iyear,iday,iubgcO
01477      integer iYr1
01478
01479      iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
01480      iubgcO= 365* (iyear-1901) ! Verhinderung Overflow: Offset im Faktor
01481      iubgcO= iubgcO + int(iyr1/4) - int(iyr1/100) + int(iyr1/400)
01482      iubgcO= iubgcO + iday -460 ! Bezugsdatum 1.1.1901= 365*1901 + 460 Schalttage
01483      return
01484  end
01485
01486
01487
01488  subroutine oubgc(iyear,iday,iubgcI)
01489      implicit none
01490      integer iyear,iday,iubgcI
01491      integer iYr1
01492
01493      iyear= int( (real(iubgci) + 694325.99) / 365.2425 )
01494  100 continue ! Schleife der evtl. Nachiteration
01495      iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
01496      iday= iubgci + 460 - 365*(iyear-1901)
01497      iday= iday + int(iyr1/100) - int(iyr1/4) - int(iyr1/400)
01498      if (iday .lt. 1) then ! Nachiteration?
01499          iyear= iyear-1
01500          goto 100
01501      end if
01502      return
01503  end
01504
01505
01506
01507 C
01508 C Zeichenroutinen
01509 C
01510
01511  subroutine frame
01512      implicit none
01513      include 'G2dAG2.fd'
01514
01515      call movabs (cxysmax(1),cxysmin(2))
01516      call drwabs (cxysmax(1),cxysmax(2))
01517      call drwabs (cxysmin(1),cxysmax(2))
01518      call drwabs (cxysmin(1),cxysmin(2))
01519      call drwabs (cxysmax(1),cxysmin(2))
01520      return
01521  end
01522
01523
01524
01525  subroutine dsplay (x,y)
01526      implicit none
01527      real x(5),y(5)
01528
01529      call setwin
01530      call cplot (x,y)
01531      call grid
01532      call label (1)
01533      call label (2)
01534      return
01535  end
01536
01537
01538
01539  subroutine cplot (x,y)
01540      implicit none
01541      real x(5),y(5)
01542      logical symbol
01543      integer i,il, keyx, keyy, lines, linsav, icount, imax
01544      real xpoint(1), ypoint(1)
01545      real DATGET
01546      include 'G2dAG2.fd'
01547
01548      call keyset (x,keyx)
01549      call keyset (y,keyy)
01550      if (keyx .eq. 1) then ! standard long

```

```

01551         imax= x(1)
01552     else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
01553         imax= x(2)
01554     else ! nonstandard
01555         imax= cnpts
01556     end if
01557     if (keyy .eq. 1) then ! standard long
01558         if (imax .lt. y(1)) imax= y(1)
01559     else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
01560         if (imax .lt. y(2)) imax= y(2)
01561     else ! nonstandard
01562         if (imax .lt. cnpts) imax= cnpts
01563     end if
01564
01565     symbol= (csymb1 .ne. 0) .and. (cline .ne.-2) .and. (cline .ne.-3)
01566
01567     i= 1 ! Suche Startpunkt
01568 100 continue ! repeat
01569     if (i .gt. imax) return ! kein Punkt zu zeichnen
01570     xpoint(1)= datget(x,i,keyx)
01571     ypoint(1)= datget(y,i,keyy)
01572     if ((xpoint(1) .ge. cfinf) .or. (ypoint(1) .ge. cfinf)) then ! while
01573         i= i+cstepl
01574         goto 100
01575     end if
01576
01577     call movea (xpoint(1),ypoint(1))
01578     if (cline .eq. -4) call pointa (xpoint(1),ypoint(1))
01579     if (cline .lt. -10) call uline (xpoint(1),ypoint(1),1)
01580     if (cline .eq.-2 .or. cline .eq.-3) then
01581         call bar (xpoint(1),ypoint(1),cline)
01582     end if
01583     if (symbol) call bsyms (xpoint(1),ypoint(1),csymb1)
01584
01585     if (cline .eq. -1) then
01586         lines= 2
01587     else if ((cline .eq. -2) .or. (cline .eq. -3)) then
01588         lines= 3
01589     else if (cline .eq. -4) then
01590         lines=4
01591     else if (cline .lt. -10) then
01592         lines=5
01593     else
01594         lines=1 ! bei cline = 0: dash ergibt durchgezogene Linie
01595     end if
01596
01597     il= i+cstepl
01598     if (il .ge. imax) return
01599     icount= csteps
01600     linsav= lines
01601
01602     do 900 i=il,imax,cstepl
01603         xpoint(1)= datget(x,i,keyx)
01604         ypoint(1)= datget(y,i,keyy)
01605         if ((xpoint(1) .ge. cfinf) .or. (ypoint(1) .ge. cfinf)) then
01606             if (i.gt.imax-cstepl) return ! Der letzte Punkt ist ungueltig -> done
01607             if ((cline .ne. -2) .and. (cline .ne. 3)) lines= 2
01608         else
01609             if (lines .eq. 1 ) then
01610                 call dasha (xpoint(1),ypoint(1), cline) ! dashed or solid
01611             else if (lines .eq. 2 ) then
01612                 call movea (xpoint(1),ypoint(1))
01613                 lines=linsav ! restore after missing data
01614             else if (lines .eq. 3 ) then
01615                 call bar (xpoint(1),ypoint(1),0)
01616             else if (lines .eq. 4 ) then
01617                 call pointa (xpoint(1),ypoint(1))
01618             else
01619                 call uline (xpoint(1),ypoint(1),i)
01620             end if
01621             if (symbol) then
01622                 icount=icount-1
01623                 if(icount .le. 0) then
01624                     icount= csteps
01625                     call bsyms (xpoint(1),ypoint(1),csymb1)
01626                 end if
01627             end if
01628         end if
01629 900 continue
01630     return
01631 end
01632
01633
01634
01635 subroutine keyset (array,key)
01636 implicit none
01637 integer key

```



```

01638     integer npts
01639     real array(1)
01640     include 'G2dAG2.fd'
01641
01642     if (cnpts .ne. 0) then      ! nonstandard array
01643         key= 5
01644     else
01645         npts= nint(array(1))
01646         if (npts .ge. 0) then    ! standard long
01647             key= 1
01648         else if (npts .eq. -1) then ! short
01649             key= 2
01650         else if (npts .eq. -2) then ! short calendar
01651             key= 3
01652         else                      ! short user
01653             key= 4
01654         end if
01655     end if
01656     return
01657 end
01658
01659
01660
01661 real function datget (arr,i,key)
01662 implicit none
01663 integer i, key
01664 real calpnt, upoint
01665 real arr(5) ! Dimension 5 sonst GNU-Compilerwarnung bei dat= ...arr(5)...
01666 real dat, olddat
01667 save olddat
01668
01669 if (key.eq.1) then ! standard long
01670     dat= arr(i+1)
01671 else if (key.eq.2) then ! standard short
01672     dat= arr(3) + arr(4)*real(i-1)
01673 else if (key.eq.3) then ! short calendar
01674     dat= calpnt(arr,i)
01675 else if (key.eq.4) then ! user
01676     dat= upoint(arr,i,olddat)
01677 else if (key.eq.5) then ! non standard
01678     dat= arr(i)
01679 endif
01680 olddat= dat
01681 datget= dat
01682 return
01683 end
01684
01685
01686
01687 C Balkendiagramme
01688
01689 subroutine bar (x,y,line)
01690 implicit none
01691 real x, y
01692 integer line
01693 integer key, ix,iy, ixl,iyl,ixh,iyh
01694 real xfac, yfac
01695 logical VerticalBar
01696 integer isymb, ihalf, lspace, minx,maxx,miny,maxy, ibegx,ibegy
01697 SAVE isymb, ihalf, lspace, minx,maxx,miny,maxy, ibegx,ibegy
01698 SAVE verticalbar
01699 include 'G2dAG2.fd'
01700
01701 if (line .ne. 0) then ! Erster Aufruf -> Parameterbestimmung
01702     verticalbar= line .ne. -3
01703     isymb= csymb1
01704     ihalf= .5 * csizel
01705     lspace= csizes
01706     if (lspace .le. 1) lspace=20 ! Default: 20 Pixel Schraffur
01707     if (ihalf .lt. 2) ihalf=20 ! Default: 40 Pixel Balkenbreite
01708     if (cxysmin(1) .le. cxysmax(1)) then
01709         minx= cxysmin(1)
01710         maxx= cxysmax(1)
01711     else
01712         minx= cxysmax(1)
01713         maxx= cxysmin(1)
01714     end if
01715     if (cxysmin(2) .le. cxysmax(2)) then
01716         miny= cxysmin(2)
01717         maxy= cxysmax(2)
01718     else
01719         miny= cxysmax(2)
01720         maxy= cxysmin(2)
01721     end if
01722
01723     call seetrn(xfac,yfac, key)
01724     if (key .eq. 2) then ! logarithmische Werte

```

```

01725         ibegx= cxysmin(1)
01726         ibegy= cxysmin(2)
01727     else
01728         call wincot (0.,0.,ibegx,ibegy)
01729     end if
01730 end if
01731
01732 call wincot (x,y,ix,iy)
01733 if (verticalbar) then ! vertikale Balken
01734     iyl= min0(ibegy,iy)
01735     iyh= max0(ibegy,iy)
01736     ixl= min0(ix-ihalf,ix+ihalf)
01737     ixh= max0(ix-ihalf,ix+ihalf)
01738 else ! horizontale Balken
01739     iyl= min0(iy-ihalf,iy+ihalf)
01740     iyh= max0(iy-ihalf,iy+ihalf)
01741     ixl= min0(ibegx,ix)
01742     ixh= max0(ibegx,ix)
01743 end if
01744 ixl=max0(ixl,minx)
01745 ixh=min0(ixh,maxx)
01746 iyl=max0(iyl,miny)
01747 iyh=min0(iyh,maxy)
01748 if ((ixh-ixl .ge. 2) .and. (iyh-iyl .ge. 2)) then ! mindestens 2x2 Pxl
01749     call filbox(ixl,iyl,ixh,iyh,isymb,lspace)
01750 end if
01751 return
01752 end
01753
01754
01755
01756 subroutine filbox (minx,miny,maxx,maxy,ishade,lspace)
01757 implicit none
01758 integer minx,miny,maxx,maxy,ishade,lspace
01759 integer iminx,imaxx,iminy,imaxy
01760 integer i, ishift, idely, iymax
01761 real xmin, xmax
01762 real savcom (60)
01763
01764 iminx= min0(minx,maxx)      ! zeichne Rechteck
01765 iminy= min0(miny,maxy)
01766 imaxx= max0(minx,maxx)
01767 imaxy= max0(miny,maxy)
01768
01769 call movabs (iminx,iminy)
01770 call drwabs (imaxx,iminy)
01771 call drwabs (imaxx,imaxy)
01772 call drwabs (iminx,imaxy)
01773 call drwabs (iminx,iminy)
01774
01775 if ((ishade .le.0) .or. (ishade .gt. 15)) return ! ohne Schraffur
01776
01777 ishift= ishade / 2
01778 if ((ishade-ishift*2) .ne. 0) then ! Bit0: horizontale Schraffur
01779     i= iminy
01780 100 continue ! repeat...
01781     i= i+lspace
01782     if (i .lt. imaxy) then
01783         call movabs (iminx,i)
01784         call drwabs (imaxx,i)
01785         goto 100 ! ... until
01786     end if
01787 end if ! horizontale Schraffur gezeichnet
01788
01789 if (mod(ishift,2) .ne. 0) then ! Bit1: vertikale Schraffur
01790     i= iminx
01791 110 continue ! repeat
01792     i= i+lspace
01793     if(i .lt. imaxx) then
01794         call movabs (i,iminy)
01795         call drwabs (i,imaxy)
01796         goto 110
01797     end if ! vertikale Schraffur gezeichnet
01798 end if
01799
01800 if (ishade .ge. 4) then ! diagonale Schraffuren
01801     xmin= real(iminx)
01802     xmax= real(imaxx)
01803     call svstat (savcom) ! verwende TCS-Clipping
01804     call lintrn
01805     call dwindo (xmin,xmax,real(iminy),real(imaxy))
01806     call twindo (iminx,imaxx,iminy,imaxy)
01807
01808 if (ishade .ge. 8) then ! Bit3: diagonal fallend
01809     idely= iminx-imaxx
01810     iymax= imaxy+imaxx-iminx
01811     i= iminy+lspace

```

```

01812 120      continue ! repeat ...
01813          call movea (ximin,real(i))
01814          call drawa (ximax,real(i+idely))
01815          i= i+lspace
01816          if (i .lt. iymax) goto 120 ! ... until
01817          ishift= ishade -8
01818      else
01819          ishift= ishade
01820      end if
01821
01822      if (ishift .ge. 4) then ! Bit2: diagonal steigend
01823          idely= imaxx-iminx
01824          iymax= real(imaxy)
01825          i= iminy - idely + lspace
01826 130      continue ! repeat...
01827          call movea (ximin,real(i))
01828          call drawa (ximax,real(i+idely))
01829          i= i+lspace
01830          if (i .lt. iymax) goto 130 ! ...until
01831      end if
01832      call restat (savcom)
01833  end if ! Diagonalen
01834      return
01835  end
01836
01837
01838
01839 C Zeichnen von Symbolen
01840
01841      subroutine bsyms (x,y,isym)
01842      implicit none
01843      real x,y
01844      integer isym
01845      include 'G2dAG2.fd'
01846
01847      if (isym .ge. 0) then
01848          call symout (isym, csizes)
01849      else
01850          call users (x,y,isym)
01851      end if
01852      call movea (x,y)
01853      return
01854  end
01855
01856
01857
01858      subroutine symout (isym,fac)
01859      implicit none
01860      integer isym
01861      real fac
01862      integer ix,iy, ihorz,ivert
01863
01864      call seeloc (ix,iy)
01865      if (isym .gt. 127) then
01866          call softek (isym)
01867      else if (isym .ge. 33) then
01868          call csize (ihorz,ivert)
01869          ihorz= int( real(ihorz)*.3572)
01870          ivert= int( real(ivert)*.3182)
01871          call movrel (-ihorz,-ivert)
01872          call alfmod
01873          call toutpt (isym)
01874      else if (isym .le. 11) then
01875          call teksym (isym,fac)
01876      end if
01877      call movabs (ix,iy)
01878      return
01879  end
01880
01881
01882
01883      subroutine teksym (isym,amult)
01884      implicit none
01885      integer isym
01886      real amult
01887      integer ihalf, ifull
01888
01889      ihalf= nint(8.* amult)
01890      ifull=ihalf * 2
01891      if (isym .eq. 1) then ! Kreis
01892          call teksyml (0, 360, 30, 8.*amult)
01893      else if (isym .eq. 2) then ! X
01894          call movrel (ihalf,ihalf)
01895          call drwrel (-ifull,-ifull)
01896          call movrel (0,ifull)
01897          call drwrel (ifull,-ifull)
01898      else if (isym .eq. 3) then ! Dreieck

```

```

01899      call teksyml (90, 450, 120, 8.*amult)
01900  else if (isym .eq. 4) then ! Quadrat
01901      call teksyml (45, 405, 90, 8.*amult)
01902  else if (isym .eq. 5) then ! Stern
01903      call teksyml (90, 810, 144, 8.*amult)
01904  else if (isym .eq. 6) then ! Raute
01905      call teksyml (90, 450, 90, 8.*amult)
01906  else if (isym .eq. 7) then ! vertikaler Balken
01907      call teksyml (90, 270, 180, 8.*amult)
01908  else if (isym .eq. 8) then ! Kreuz
01909      call movrel (0,ihalf)
01910      call drwrel (0,-ifull)
01911      call movrel (-ihalf,ihalf)
01912      call drwrel (ifull,0)
01913  else if (isym .eq. 9) then ! Pfeil nach oben
01914      call drwrel (-2,-6)
01915      call drwrel (4,0)
01916      call drwrel (-2,6)
01917      call drwrel (0,-ifull)
01918  else if (isym .eq. 10) then ! Pfeil nach unten
01919      call drwrel (-2,6)
01920      call drwrel (4,0)
01921      call drwrel (-2,-6)
01922      call drwrel (0,ifull)
01923  else if (isym .eq. 11) then ! Durchstreichung
01924      call teksyml (270, 630, 120, 8.*amult)
01925  end if
01926  return
01927  end
01928
01929
01930
01931  subroutine teksyml (istart, iend, incr, siz)
01932  implicit none
01933  integer istart, iend, incr
01934  real siz
01935  integer i, mx,my,mix,miy
01936  real b
01937
01938  b= real(istart)*.01745
01939  mx= nint(siz*cos(b))
01940  my= nint(siz*sin(b))
01941  call movrel (mx,my)
01942  do 100 i= istart+incr, iend, incr
01943      b= real(i)*.01745
01944      mix= nint(siz*cos(b))
01945      miy= nint(siz*sin(b))
01946      call drwrel (mix-mx,miy-my)
01947      mx= mix
01948      my= miy
01949  100 continue
01950  return
01951  end
01952
01953
01954
01955  C Netz und Ticmarks
01956
01957  subroutine grid
01958  implicit none
01959  integer i, mlim
01960  real xyext,xyextm, tintvl,tmntvl
01961  include 'G2dAG2.fd'
01962
01963  if (cxyfrm(2) .ne. 0) then ! Zeichnen der y-Achse
01964      i= min0(cxysmin(1),cxysmax(1)) + cxyloc(2)
01965      call movabs (i, cxysmax(2))
01966      call drwabs (i, cxysmin(2))
01967  if (cxybeg(2) .ne. cxyend(2)) then ! Zeichnen y-Ticmarks
01968      i= cxylab(2) ! Labeltyp
01969      if (i .eq. 1) i= cxytype(2) ! =1: Typ entsprechend Daten
01970      if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
01971          if(cxytics(2) .ne. 0) then
01972              tintvl= real(cxysmax(2)-cxysmin(2)) / real( cxytics(2))
01973          end if
01974          if (cxymtcs(2) .gt. 0) tmntvl= tintvl / real(cxymtcs(2))
01975          call movabs(cxybeg(2),cxysmin(2))
01976          call drwabs(cxyend(2),cxysmin(2))
01977          xyext= real(cxysmin(2))
01978          do 100, i=1,cxytics(2)
01979              if (cxymbeg(2) .ne. cxymend(2)) then ! Zeichnen Minor Ticmarks
01980                  mlim= cxymtcs(2)-1
01981                  xyextm= xyext
01982  110 continue ! repeat...
01983                  if (mlim.gt.0) then ! ...until mlim <= 0
01984                      xyextm= xyextm+tmntvl
01985                      call movabs (cxymbeg(2), nint(xyextm))

```

```

01986         call drwabs (cxymend(2), nint(xyextm))
01987         mlim=mlim-1
01988         goto 110
01989     else if (mlim.lt. 0) then
01990         call logtix (2,xyext,tintvl,cxybeg(2),cxymend(2))
01991     end if
01992 end if
01993 xyext= xyext+tintvl
01994 call movabs (cxybeg(2), nint(xyext))
01995 call drwabs (cxyend(2), nint(xyext))
01996 100 continue
01997 end if ! Labtyp=6: Monate
01998 end if ! Ende Zeichnen Ticmarks
01999 end if ! Ende Zeichnen der Achse
02000
02001 if (cxyfrm(1) .ne. 0) then ! Zeichnen der x-Achse
02002     i= min0(cxysmin(2),cxysmax(2)) + cxyloc(1)
02003     call movabs (cxysmin(1), i)
02004     call drwabs (cxysmax(1), i)
02005     if (cxybeg(1) .ne. cxyend(1)) then ! Zeichnen y-Ticmarks
02006         i= cxylab(1) ! Labeltyp
02007         if (i.eq. 1) i= cxytype(1) ! =1: Typ entsprechend Daten
02008         if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
02009             if(cxytics(1) .ne. 0) then
02010                 tintvl= real(cxysmax(1)-cxysmin(1)) / real( cxytics(1))
02011             end if
02012             if (cxymtcs(1) .gt. 0) tmntvl= tintvl / real(cxymtcs(1))
02013             call movabs(cxysmin(1), cxybeg(1))
02014             call drwabs(cxysmin(1), cxyend(1))
02015             xyext= real(cxysmin(1))
02016             do 120, i=1,cxytics(1)
02017                 if (cxymbeg(1) .ne. cxymend(1)) then ! Zeichnen Minor Ticmarks
02018                     mlim= cxymtcs(1)-1
02019                     xyextm= xyext
02020 130 continue ! repeat...
02021                     if (mlim.gt.0) then ! ...until mlim <= 0
02022                         xyextm= xyextm+tmntvl
02023                         call movabs (nint(xyextm), cxymbeg(1))
02024                         call drwabs (nint(xyextm), cxymend(1))
02025                         mlim=mlim-1
02026                         goto 130
02027                     else if (mlim.lt. 0) then
02028                         call logtix (1,xyext,tintvl,cxybeg(1),cxymend(1))
02029                     end if
02030                 end if
02031                 xyext= xyext+tintvl
02032                 call movabs (nint(xyext), cxybeg(1))
02033                 call drwabs (nint(xyext), cxyend(1))
02034 120 continue
02035             end if ! Labtyp=6: Monate
02036         end if ! Ende Zeichnen Ticmarks
02037     end if ! Ende Zeichnen der Achse
02038     return
02039 end
02040
02041
02042
02043 subroutine logtix (nbase,start,tintvl,mstart,mend)
02044 implicit none
02045 integer nbase,mstart,mend
02046 real start, tintvl
02047 integer i, logtic, ihorz, ivert, idx,idy
02048 character*1 loglab
02049 include 'G2dAG2.fd'
02050
02051 call csize (ihorz,ivert)
02052 do 100 i=2,9
02053     write (unit=loglab, fmt='(i1)') i ! Unicodfaehig durch Compilerfeature
02054     logtic= nint(log10(real(i))*tintvl + start)
02055     if (nbase .eq. 1) then ! x-Achse
02056         idx= -ihorz/3
02057         if (mstart .gt. mend) then
02058             idy= ivert
02059         else
02060             idy= -ivert
02061         end if
02062         call movabs (logtic,mend)
02063         call drwabs (logtic,mstart)
02064         if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
02065             call movrel (idx,idy)
02066             call toutstc (loglab)
02067         end if
02068     else if (nbase .eq. 2) then ! y-Achse
02069         if (mstart .gt. mend) then
02070             idx= ihorz
02071         else

```

```

02073         idx= -ihorz
02074     end if
02075     idy= -ivert / 3
02076     call movabs (mend,logtic)
02077     call drwabs (mstart,logtic)
02078 end if
02079
02080     if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
02081         call movrel (idx,idy)
02082         call toutstc (loglab)
02083     end if
02084 100 continue
02085 return
02086 end
02087
02088
02089
02090 subroutine tset (nbase)
02091 implicit none
02092 integer nbase
02093 integer IOTHER
02094 integer otherbase, near, nfar, newloc, nlen
02095 include 'G2dAG2.fd'
02096
02097 otherbase= iother(nbase)
02098 near= min0(cxysmin(otherbase), cxysmax(otherbase))
02099 nfar= max0(cxysmin(otherbase), cxysmax(otherbase))
02100 newloc= near + cxyloc(nbase)
02101 if (cxyfrm(nbase) .ne. 1) then
02102     if (newloc .lt. ((nfar+near)/2)) then
02103         nlen= cxylen(nbase)
02104     else
02105         nlen= -cxylen(nbase)
02106         nfar= near
02107     end if
02108     call tset2 (newloc,nfar,nlen,cxyfrm(nbase),
02109 1 cxybeg(nbase),cxyend(nbase))
02110 else
02111     cxybeg(nbase)= 0
02112     cxyend(nbase)= 0
02113 end if
02114
02115 if ((cxymfrm(nbase) .ne. 1) .and. (cxymtcs(nbase) .ne. 0)) then
02116     nlen= nlen / 2
02117     call tset2 (newloc,nfar,nlen,cxymfrm(nbase),
02118 1 cxymbeg(nbase),cxymend(nbase))
02119 else
02120     cxymbeg(nbase)= 0
02121     cxymend(nbase)= 0
02122 end if
02123 return
02124 end
02125
02126
02127
02128 subroutine tset2 (newloc,nfar,nlen,nfrm,kstart,kend)
02129 implicit none
02130 integer newloc,nfar,nlen,nfrm,kstart,kend
02131
02132 if (nfrm .eq. 3 .or. nfrm .eq. 6) then
02133     kstart= newloc
02134 else
02135     kstart=newloc-nlen
02136 end if
02137 if (kstart .lt. 0) then
02138     kstart= 0
02139 else if (kend .gt. 1023) then
02140     kstart= 1023
02141 end if
02142
02143 if (nfrm .eq. 2) then
02144     kend= newloc
02145 else if (nfrm .eq. 5 .or. nfrm .eq. 6) then
02146     kend= nfar
02147 else
02148     kend=newloc+nlen
02149 end if
02150 if (kend .lt. 0) then
02151     kend= 0
02152 else if (kend .gt. 1023) then
02153     kend= 1023
02154 end if
02155 return
02156 end
02157
02158
02159

```

```

02160      subroutine monpos (nbase,iy1,dpos, spos)
02161      implicit none
02162      integer nbase, iy1, spos
02163      integer iy, idays, iubgc1
02164      real dpos
02165
02166      call ymdyd (iy, idays, iy1, nint(dpos)+1, 1)
02167      call iubgc (iy, idays, iubgc1)
02168      call gline (nbase, real(iubgc1), spos)
02169      return
02170      end
02171
02172
02173
02174      subroutine gline (nbase, datapt, spos)
02175      implicit none
02176      integer nbase, spos
02177      real datapt
02178      integer i
02179      include 'G2dAG2.fd'
02180
02181      if (nbase .eq. 1) then ! x-Achsengrid
02182          call wincot (datapt, 1., spos, i)
02183          if (iabs(cxyend(1)-cxybeg(1)) .ge. 2) then
02184              call movabs(spos, cxybeg(1))
02185              call drwabs(spos, cxyend(1))
02186          end if
02187      else ! y-Achsengrid
02188          call wincot (1., datapt, i, spos)
02189          if (iabs(cxyend(2)-cxybeg(2)) .ge. 2) then
02190              call movabs(cxybeg(2), spos)
02191              call drwabs(cxyend(2), spos)
02192          end if
02193      end if
02194      return
02195      end
02196
02197
02198
02199      C Label
02200
02201      subroutine label (nbase)
02202      implicit none
02203      integer nbase
02204      logical even, stag
02205      integer i, icv, igap, iquadrant, labtyp, ilim, iposflag, ioff, iy
02206      integer ispos, isintv, iyear
02207      integer level1, level2
02208      real fnum, fac, dpos, dintv
02209      character *(255) labstr
02210      integer IOTHER
02211      include 'G2dAG2.fd'
02212
02213      labtyp= cxylab(nbase)
02214      if(labtyp .eq. 1) labtyp= cxytype(nbase) ! LabTyp=1: = dataType
02215      if (labtyp .eq. 0) return ! LabTyp=0: keine Label
02216
02217      fac= 10.**(-cxyepon(nbase))
02218
02219      dintv= real(cxystep(nbase)) / real(cxytics(nbase)) ! Zwischenergebnis
02220      isintv= nint(real(cxysmax(nbase)-cxysmin(nbase)) * dintv)
02221      dintv= (cxyamax(nbase)-cxyamin(nbase)) * dintv
02222
02223      call csize (i, icv) ! nur icv = vertikale Hoehe benoetigt
02224      igap= icv / 3
02225      if (nbase.eq.1) igap= 2*igap
02226      if (iabs(cxysmax(iother(nbase))-cxysmin(iother(nbase)))
02227      1 .gt. 2* cxyloc(nbase)) then
02228          iquadrant= -1 ! untere Haelfte
02229      else
02230          iquadrant= +1
02231      end if
02232      level1= min0(cxysmax(iother(nbase)), cxysmin(iother(nbase)))
02233      1 - (igap-icv/3 ) + cxyloc(nbase)
02234      2 + isign(igap+cxylen(nbase), iquadrant)
02235      level2= level1 + isign(icv+igap, iquadrant)
02236
02237      if (nbase .eq. 1) then ! Label links/zentriert/rechts?
02238          iposflag= 0 ! x-Achse: zentriert
02239      else
02240          iposflag= -iquadrant
02241      end if
02242
02243      stag= cxystag(nbase) .eq. 2 ! Verwendung in Schleife
02244      even= .false.
02245      ilim= cxytics(nbase) + 1
02246

```

```

02247     dpos= cxyamin(nbase)
02248     ispos= cxysmin(nbase)
02249
02250     if (iabs(labtyp) .ge. 3 .and. iabs(labtyp) .le. 8) then ! Kalenderdaten
02251       call oubgc (iyear,i,ifix(cxydmin(nbase))) ! i: Tag nicht benoetigt
02252       dpos= dpos+dintv ! 1. Tic ungelabelt
02253       ispos= ispos+isintv
02254       ilim=ilim-1
02255       if (nbase .eq. 1) iposflag= 1 ! x-Achse Kalender: rechtsbuendig
02256     end if
02257
02258     do 100 i=1,ilim, cxystep(nbase)
02259       if ((labtyp .le. 2) .or. (labtyp .ge. 8)) then
02260         fnum= dpos
02261       else ! Kalendertyp ohne Jahr
02262         if (labtyp.eq.3) then ! Tage
02263           fnum= 7.
02264         else if (labtyp.eq.4) then ! Wochen
02265           fnum= 52.
02266         else if (labtyp.eq.5) then ! Periods
02267           fnum= 13.
02268         else if (labtyp.eq.6) then ! Monate
02269           fnum= 12.
02270         else if (labtyp.eq.7) then ! Quartal
02271           fnum= 4.
02272         end if ! Jahr wird wie linear behandelt
02273         fnum= amod(dpos-1.,fnum)+1.
02274       end if
02275
02276       if (labtyp .lt. 0) then
02277         call usesetc (fnum, cxywdth(nbase), nbase, labstr)
02278       else if ((labtyp .eq. 6) .OR. (labtyp .eq. 3)) then
02279         call alfsetc (fnum, labtyp, labstr)
02280         if (cxywdth(nbase) .lt. len(labstr)) then
02281           labstr(cxywdth(nbase)+1:cxywdth(nbase)+1)= char(0)
02282         end if
02283         if (labtyp .eq. 6) call monpos (nbase,iyear,dpos,ispos)
02284       else
02285         call numsetc (fnum*fac,cxywdth(nbase),nbase,labstr)
02286       end if
02287       call justerc (labstr, iposflag, ioff)
02288
02289       if (nbase .eq. 1) then ! x-Achse
02290         iy= level1
02291         if(stag .and. even) iy= level2
02292         even= .not. even
02293         call notatec (ispos+ioff,iy, labstr)
02294       else ! y-Achse
02295         call notatec (level1+ioff,ispos-igap,labstr)
02296       end if
02297       dpos= dpos+dintv
02298       ispos= ispos+isintv
02299 100 continue ! end do
02300
02301     if ((labtyp .ne. 2) .and. (cxyetyp(2) .ge. 0)) then ! nicht logarithm.
02302       if (nbase .eq. 1) then ! x-Achse
02303         if (stag) level2= level2 + isign(icv+igap,iquadrant)
02304         i=(cxysmin(nbase)+cxysmax(nbase))/2.
02305         iy=level2
02306       else
02307         i= level1
02308         iy= max0(cxysmin(nbase),cxysmax(nbase)) +icv+igap
02309       end if
02310       call remlab (nbase,cxyloc(nbase),labtyp,i,iy)
02311     end if
02312     return
02313   end
02314
02315
02316
02317   subroutine numsetc (fnum,iwidth,nbase, outstr)
02318   implicit none
02319   real fnum
02320   integer iwidth,nbase
02321   character outstr *(*)
02322   integer iexp
02323   include 'G2dAG2.fd'
02324
02325   if (cxytype(nbase) .eq. 2) then
02326     if (fnum .gt. 0.) then
02327       iexp= fnum + .00005
02328     else if (fnum .lt. 0.) then
02329       iexp= fnum - .00005
02330     else
02331       iexp= 0
02332     end if
02333     call expoutc (nbase,iexp, outstr)

```



```

02334     else if ((cxytype(nbase).eq.1) .and. (cxydec(nbase).gt.0)) then
02335         call fformc (fnum,iwidth, cxydec(nbase), outstr)
02336     else
02337         call iformc (fnum,iwidth, outstr)
02338     end if
02339     return
02340 end
02341
02342
02343
02344 subroutine iformc (fnum,iwidth, outstr)
02345 implicit none
02346 real fnum
02347 integer iwidth
02348 character outstr *(*)
02349 character fmtstr *(11)
02350
02351 if (iwidth .le. 0) then ! iwidth=0: ohne Label
02352     outstr= char(0)
02353     return
02354 end if
02355
02356 if (iwidth .gt. 99) goto 200 ! ErrorHandler
02357 write (unit=fmtstr,fmt=100, err=200) iwidth
02358 if (len(outstr) .gt. iwidth) then
02359     write (unit= outstr, fmt=fmtstr, err=200) nint(fnum),0 ! 0: End of String
02360 else
02361     write (unit= outstr, fmt=fmtstr, err=200) nint(fnum) ! evtl. ohne EoS?
02362 end if
02363
02364 return
02365
02366 200 continue ! Error Handler
02367 outstr= '???'
02368 if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
02369 return
02370
02371 100 format ('(SS,I' ,i2.2, ',A1)')
02372 end
02373
02374
02375
02376 subroutine fformc (fnum,iwidth,idec, outstr)
02377 implicit none
02378 real fnum
02379 integer iwidth,idec
02380 character outstr *(*)
02381 integer ndgtM
02382 real fa
02383 include 'G2dAG2.fd'
02384
02385 ndgtm= iwidth-idec
02386 if (fnum .ge. 0.) then
02387     ndgtm= ndgtm -1 ! Ziffern Mantis
02388 else
02389     ndgtm= ndgtm-2 ! 1 Ziffer Vorzeichen
02390 end if
02391 fa= abs(fnum) ! Skalierung mindestens 2 signifikante Stellen: .1*abs(fnum)
02392
02393 if ( ((fa .lt. 10./cinf) .or. (fa .gt. .1**idec))
02394 1 .and. (fa .lt. 10.**ndgtm)) then
02395     call fonlyc (fnum,iwidth,idec, outstr)
02396 else
02397     call eformc (fnum,iwidth,idec, outstr)
02398 end if
02399 return
02400 end
02401
02402
02403
02404 subroutine fonlyc (fnum,iwidth,idec, outstr)
02405 implicit none
02406 real fnum
02407 integer iwidth,idec
02408 character outstr *(*)
02409 character fmtstr *(14)
02410
02411 if (iwidth .le. 0) then ! iwidth=0: ohne Label
02412     outstr= char(0)
02413     return
02414 end if
02415
02416 if ((idec .gt. iwidth-1) .or. (iwidth .gt. 99)) goto 200 ! ErrorHandler
02417 write (unit=fmtstr,fmt=100, err=200) iwidth,idec
02418 if (len(outstr) .gt. iwidth) then
02419     write (unit= outstr, fmt=fmtstr, err=200) fnum,0 ! 0: End of String
02420 else

```

```

02421     write (unit= outstr, fmt=fmtstr, err=200) fnum ! evtl. ohne EoS?
02422 end if
02423 return
02424
02425 200 continue ! Error Handler
02426 outstr= '???'
02427 if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
02428 return
02429
02430 100 format ('(SS,F' ,i2.2,'.', i2.2,',A1)')
02431 end
02432
02433
02434
02435 subroutine eformc (fnum,iwidth,idec, outstr)
02436 implicit none
02437 real fnum
02438 integer iwidth,idec
02439 character outstr *(*)
02440 integer iexpon
02441 character fmtstr *(18)
02442
02443 if (iwidth .le. 0) then ! iwidth=0: ohne Label
02444   outstr= char(0)
02445   return
02446 end if
02447
02448 call esplit (fnum,iwidth,idec,iexpon)
02449 if ((idec .gt. iwidth-7) .or. (iwidth .gt. 99)) goto 200 ! ErrorHandler
02450 write (unit=fmtstr,fmt=100, err=200) iwidth-idec-6,iwidth,iwidth-7
02451 if (len(outstr) .gt. iwidth) then
02452   write (unit= outstr, fmt=fmtstr, err=200) fnum,0 ! 0: End of String
02453 else
02454   write (unit= outstr, fmt=fmtstr, err=200) fnum ! evtl. ohne EoS?
02455 end if
02456 return
02457
02458 200 continue ! Error Handler
02459 outstr= '???'
02460 if (iwidth.lt.len(outstr)) outstr(iwidth+1:iwidth+1)= char(0)
02461 return
02462
02463 100 format ('(SS,' ,i2.2,'P,E' ,i2.2,'.', i2.2,',A1)')
02464 end
02465
02466
02467
02468 subroutine esplit (fnum,iwidth,idec,iexpon)
02469 implicit none
02470 real fnum
02471 integer iwidth,idec,iexpon
02472 real fabs
02473 include 'G2dAG2.fd'
02474
02475 fabs= abs(fnum)
02476 if (fabs .ge. 1.) then
02477   iexpon= ifix( alog10(fabs)+1.000005) - iwidth+idec+6 ! 6: Vorz.-Pkt-Exp(4)
02478 else if (fabs .ge. 10./cinf) then
02479   iexpon= alog10(fabs)
02480 else
02481   iexpon= -alog10(cinf)
02482 end if
02483 return
02484 end
02485
02486
02487
02488 subroutine expoutc (nbase,iexp, outstr)
02489 implicit none
02490 integer nbase,iexp, i, iL, nexp
02491 character outstr *(*), tmpstr *(4)
02492 include 'G2dAG2.fd'
02493
02494 iL= len(outstr)
02495 nexp= abs(iexp)
02496
02497 if ( (cxyetyp(nbase).eq.2) .and. (iL.gt. 5)
02498 1 .and. (mod(nexp,3) .eq. 0)
02499 2 .and. (iexp.ge.1) .and. (iexp.le.9) ) then ! MMMs
02500   do 20 i=3,nexp,3
02501     outstr(i/3:i/3)= 'M'
02502 20 continue
02503   outstr(nexp/3+1:)= char(39) // 'S' // char(0)
02504
02505 else if ( (cxyetyp(nbase).eq.3) .and. (iL.gt.17)
02506 1 .and. (iexp.ge.1) .and. (iexp.le.6)) then ! TENS
02507   if (nexp .eq. 1) then

```

```

02508         outstr= 'TENS' // char(0)
02509     else if (nexp .eq. 2) then
02510         outstr= 'HUNDREDS' // char(0)
02511     else if (nexp .eq. 3) then
02512         outstr= 'THOUSANDS' // char(0)
02513     else if (nexp .eq. 4) then
02514         outstr= 'TEN THOUSANDS' // char(0)
02515     else if (nexp .eq. 5) then
02516         outstr= 'HUNDRED THOUSANDS' // char(0)
02517     else if (nexp .eq. 6) then
02518         outstr= 'MILLIONS' // char(0)
02519     end if
02520 else if ( (cxytyp(nbase).eq.4) ! 10000
02521 1     .and. (iexp.ge.1) .and. (iexp.le.9)
02522 2     .and. (il.ge.nexp+2)) then
02523     do 30 i=2,nexp+1
02524         outstr(i:i)= '0'
02525 30     continue
02526     outstr(1:1)= '1'
02527     outstr(nexp+2:)= char(0)
02528
02529 else if (il .gt. 7) then ! Default: Superscript EXP
02530     if (iexp .ne. 1) then
02531         if (nexp .lt. 10) then
02532             i=1
02533         else
02534             i=2
02535         end if
02536         if (iexp .lt. 0) then
02537             i= i+1
02538         end if
02539         call iformc (real(iexp), i, tmpstr)
02540     else
02541         tmpstr= char(0) ! 10 wird ohne Exponenten 1 ausgegeben
02542     end if
02543     if (iexp .ne. 0) then
02544         if (cxytype(nbase) .ne. 2) then
02545             outstr(1:1)= 'x'
02546             i= 2
02547         else
02548             i= 1
02549         end if
02550         outstr(i:)= '10' // char(1) ! Index UP
02551         outstr(i+3:)= tmpstr ! char(0) wird bei IFORMC angehaengt
02552     else
02553         outstr(1:)= '1' // char(0) ! 1 wird nicht als 10**0 ausgegeben
02554     end if
02555 else ! outstr zu kurz
02556     outstr= '???'
02557 end if
02558
02559 return
02560 end
02561
02562
02563
02564 subroutine alfsetc (fnum, labtyp, string)
02565 implicit none
02566 integer inum, labtyp
02567 real fnum
02568 character *(*) string
02569
02570 inum= fnum + .001 ! truncate real to integer
02571 if (labtyp .eq. 3) then ! Tage
02572     if ((inum .eq. 0) .or. (inum .eq. 7)) then
02573         string= 'MONDAY' // char(0)
02574     else if (inum .eq. 1) then
02575         string= 'TUESDAY' // char(0)
02576     else if (inum .eq. 2) then
02577         string= 'WEDNESDAY' // char(0)
02578     else if (inum .eq. 3) then
02579         string= 'THURSDAY' // char(0)
02580     else if (inum .eq. 4) then
02581         string= 'FRIDAY' // char(0)
02582     else if (inum .eq. 5) then
02583         string= 'SATURDAY' // char(0)
02584     else if (inum .eq. 6) then
02585         string= 'SUNDAY' // char(0)
02586     end if
02587 else if (labtyp .eq. 6) then ! Monate
02588     if (inum .eq. 1) then
02589         string= 'JANUARY' // char(0)
02590     else if (inum .eq. 2) then
02591         string= 'FEBRUARY' // char(0)
02592     else if (inum .eq. 3) then
02593         string= 'MARCH' // char(0)
02594     else if (inum .eq. 4) then

```

```

02595         string= 'APRIL' // char(0)
02596     else if (inum .eq. 5) then
02597         string= 'MAY' // char(0)
02598     else if (inum .eq. 6) then
02599         string= 'JUNE' // char(0)
02600     else if (inum .eq. 7) then
02601         string= 'JULY' // char(0)
02602     else if (inum .eq. 8) then
02603         string= 'AUGUST' // char(0)
02604     else if (inum .eq. 9) then
02605         string= 'SEPTEMBER' // char(0)
02606     else if (inum .eq. 10) then
02607         string= 'OCTOBER' // char(0)
02608     else if (inum .eq. 11) then
02609         string= 'NOVEMBER' // char(0)
02610     else if (inum .eq. 12) then
02611         string= 'DECEMBER' // char(0)
02612     end if
02613 end if
02614 return
02615 end
02616
02617
02618
02619 subroutine notatec (ix,iy, string)
02620 implicit none
02621 integer ix, iy
02622 character *(*) string
02623 integer i, iv, is
02624 integer ISTRINGLEN
02625
02626 call csize(i,iv)          ! nur iv benoetigt
02627 call movabs(ix,iy)
02628
02629 is= 1
02630 do 100 i=1, istringlen(string)
02631     if (string(i:i) .lt. char(31) ) then
02632         if (i.gt.is) call toutstc (string(is:i-is))
02633         if (string(i:i) .eq. char(1)) call movrel (0, iv/2) ! Hochindex
02634         if (string(i:i) .eq. char(2)) call movrel (0, -iv/2) ! Index
02635         is= i+1
02636     end if
02637 100 continue
02638 if (is .le. istringlen(string)) call toutstc (string(is:))
02639 return
02640 end
02641
02642
02643
02644 subroutine vlablc (string)
02645 C
02646 C Sollte in das TCS verlagert werden, um vertikale Schrift zu erzeugen
02647 C
02648 implicit none
02649 character string*(*)
02650 integer i, icy, ix,iy
02651 integer ISTRINGLEN
02652
02653 if (istringlen(string) .le. 0) return
02654 call csize (i,icy)
02655 call seeloc (ix,iy)
02656 do 100 i=1, istringlen(string)
02657     iy= iy-icy
02658     if (iy .lt. 0) return
02659     call movabs (ix,iy)
02660     call toutpt (ichar(string(i:i)))
02661 100 continue
02662 return
02663 end
02664
02665
02666
02667 subroutine justerc (string, iPosFlag, iOff)
02668 implicit none
02669 integer iPosFlag, iOff
02670 character string*(*)
02671 integer i, ilen, nCtrl
02672 integer ISTRINGLEN, LINWDT
02673
02674 ilen= istringlen(string)
02675 nctrl= 0 ! Zaehlen der Ctrlcharacter
02676 do 100 i=1, ilen
02677     if (string(i:i) .lt. char(31) ) nctrl= nctrl+1
02678 100 continue
02679
02680 if (iposflag .lt. 0) then ! linksbuendig
02681     ioff= 0

```

```

02682     else ! rechtsbuendig und zentriert
02683         ioff= -linwdt((ilen-nctrl)*8-2)/8          ! rechtsbuendig
02684         if (iposflag.eq.0) ioff= ioff / 2          ! zentriert
02685     end if
02686
02687     return
02688 end
02689
02690
02691
02692 subroutine width (nbase)
02693     implicit none
02694     integer nbase
02695     integer labtyp
02696     include 'G2dAG2.fd'
02697
02698     labtyp= cxylab(nbase)
02699     if(labtyp .eq. 1) labtyp= cxytype(nbase) ! LabTyp=1: = dataType
02700
02701     if ((cxywdth(nbase).ne.0) .and. (labtyp.ne.1)) return ! Manuelle Vorgabe nichtlinear
02702
02703     if (labtyp.le.1) then ! lineare Achsen und anwenderdefinierte Label
02704         call lwidth (nbase)
02705     else if (labtyp .eq. 2) then ! logarithmische Achsen
02706         if (cxyetyp(nbase) .le. 1) then ! 10 mit Exponent
02707             cxywdth(nbase)= 6
02708         else if (cxyetyp(nbase) .eq. 2) then ! M, MM...
02709             cxywdth(nbase)= int(alog10(abs(cxydmax(nbase)))/3. ) + 6
02710         else if (cxyetyp(nbase) .eq. 3) then ! Ausgeschriebene Worte
02711             cxywdth(nbase)= 20
02712             cxystep(nbase)= 1
02713             cxystag(nbase)= 2
02714         else if (cxyetyp(nbase) .eq. 4) then ! 1 mit 0
02715             cxywdth(nbase)= max(abs(alog10(abs(cxydmin(nbase)))),
02716 1             abs(alog10(abs(cxydmin(nbase)))) ) + 2
02717         end if
02718     else if (labtyp .gt. 2) then ! Kalenderachsen
02719         if ((labtyp .eq. 3) .or. (labtyp .eq. 6)) then ! Tage oder Monate
02720             cxywdth(nbase)= 9
02721         else
02722             cxywdth(nbase)= 4
02723         end if
02724     end if
02725 end if
02726
02727 return
02728 end
02729
02730
02731
02732
02733 subroutine lwidth (nbase)
02734     implicit none
02735     integer nbase
02736     integer iadj, most, least, isign,iwidth, idelta, ndec, iexp
02737     real xmax
02738     real ROUND
02739     include 'G2dAG2.fd'
02740
02741     iadj= 0
02742     xmax= amax1(abs(cxydmin(nbase)),abs(cxydmax(nbase)))
02743     if (xmax .gt. 1.) then
02744         most= int(alog10(xmax) + 1.00005) ! Position Most Significant Digit
02745         iadj= 1
02746     else if (xmax .eq. 1.) then
02747         most= 0
02748     else
02749         most= int(alog10(xmax) - 0.00005)
02750     end if
02751
02752     ndec= cxydec(nbase)
02753     if (cxydec(nbase) .ne. 0) then ! Anzahl Dezimalstellen vorgegeben
02754         least= -ndec ! Entspricht Position LeastSignificant Digit
02755     else
02756         least= cxylsig(nbase)
02757     end if
02758
02759     if (cxydmin(nbase) .lt. 0.) then
02760         isign=1 ! 1 Buchstabe Vorzeichen
02761     else
02762         isign=0
02763     end if
02764
02765     if ((most .lt. 0) .or. (least .ge. 0)) then
02766         iwidth= max0(1,most)- min0(0,least) + isign
02767         if (most .lt. 0) iwidth= iwidth+1 ! 1 Dezimalpunkt
02768         if ((iwidth .gt. 5 ) .and. (cxyetyp(nbase) .ge. 0)) then

```

```

02769         if (cxyetyp(nbase).eq.2) then
02770             iexp= int( roundd(real(most-iadj),3.))
02771         else
02772             iexp= int( roundd(real(most-iadj),1.))
02773         end if
02774         iwidth= most-least+isign+ 2
02775         ndec= max0(0,iexp-least+iadj)
02776     else
02777         ndec= max(0,-least)
02778         iexp= 0
02779     end if
02780 else
02781     iexp= 0
02782     ndec= max(0,-least)
02783     iwidth= most-least+isign+1
02784     if (most .eq. 0) iwidth= iwidth+1 ! Einbezug fuehrende Null
02785 end if
02786
02787 if ((cxywdth(nbase) .ne. 0).and.(cxywdth(nbase).lt. iwidth)) then
02788     idelta= iwidth - cxywdth(nbase) - ndec
02789     if ((ndec .gt. 0) .and. (idelta .lt. 1) ) then
02790         ndec= max0(0,-idelta)
02791         iwidth= cxywdth(nbase)
02792     else
02793         iexp= iexp+idelta
02794         if(ndec .gt. 0) iexp=iexp-1
02795         iwidth= cxywdth(nbase)
02796         ndec=0
02797     end if
02798 end if
02799
02800 cxywdth(nbase)= iwidth
02801 cxydec(nbase)= ndec
02802 cxyepon(nbase)= iexp
02803 return
02804 end
02805
02806
02807
02808 subroutine remlab (nbase,iloc,labtyp,ix,iy)
02809 implicit none
02810 integer nbase, iloc, labtyp, ix, iy
02811 integer iyear1,iday1, iyear2,iday2
02812 integer iyear,imon,iday, ioff, iposflag
02813 character label *(25)
02814 include 'G2dAG2.fd'
02815
02816 if (iabs(labtyp) .eq. 1) then ! lineare Daten
02817     if (cxyepon(nbase) .eq. 0) return ! kein Exponent
02818     call expoutc (nbase,cxyepon(nbase), label)
02819 else ! Kalenderdaten
02820     if ((labtyp .ge. 4) .and. (labtyp.ne.6)) then ! Wochen, Quartale, Jahre
02821         ioff= 4 ! Überlappung der Jahre vermeiden
02822     else
02823         ioff= 0
02824     end if
02825     call oubgc (iyear1,iday1, nint(cxydmin(nbase))+ioff)
02826     call oubgc (iyear2,iday2, nint(cxydmax(nbase))-ioff)
02827     if (iday2 .le. 1) iyear2=iyear2-1
02828     iday2=iday2-1
02829     call ydynd(iyear1,iday1,iyear,imon,iday)
02830
02831     if (iabs(labtyp).eq. 3) then
02832         call iformc (real(iday), 2, label(1:2))
02833         label(3:3)= ' ' ! 'dd '
02834         call alfsetc (real(imon), 6, label(4:6)) ! labtyp 6= Monate, Laenge 3
02835         label(7:7)= ' ' ! 'dd mmm '
02836         call iformc (real(iyear), 4, label(7:10)) ! 'dd mm yyyy'
02837         label(11:11)= char(0) ! evtl. Labelende
02838         if (iyear1 .lt. iyear2) then ! bei Bedarf Start und Endjahr
02839             label(11:11)= '-' ! 'dd mm yyyy-'
02840             call ydynd(iyear2,iday2,iyear,imon,iday)
02841             call iformc (real(iday), 2, label(12:13)) ! 'dd'
02842             label(14:14)= ' ' ! 'dd mm yyyy-dd '
02843             call alfsetc (real(imon), 6, label(15:17)) ! 'dd mmm'
02844             label(18:18)= ' ' ! 'dd mm yyyy-dd mmm '
02845             call iformc (real(iyear), 4, label(19:22)) ! 'dd mm yyyy-'
02846             label(23:23)= char(0)
02847         end if
02848     else
02849         call iformc (real(iyear), 4, label(1:4)) ! 'yyyy'
02850         label(5:5)= char(0)
02851         if (iyear1 .lt. iyear2) then ! bei Bedarf Start und Endjahr
02852             label(5:5)= '-' ! 'yyyy-'
02853             call iformc (real(iyear2), 4, label(6:9)) ! 'yyyy-yyyy'
02854             label(10:10)= char(0)
02855         end if

```

```

02856         end if
02857     end if
02858
02859     if ((nbase.eq.1) .or. (iloc.eq.1)) then ! X-Achse oder y Zentriert
02860         iposflag= 0
02861     else
02862         iposflag= isign(1,1-iloc)
02863     end if
02864     call justerc (label, iposflag, ioff)
02865     call notatec (ix+ioff, iy,label)
02866     return
02867 end
02868
02869
02870
02871 subroutine spread (nbase)
02872 implicit none
02873 integer nbase
02874 integer ih, labtyp, iwidth, iMaxWid
02875 integer LINWDT
02876 include 'G2dAG2.fd'
02877
02878 if (cxystag(nbase) .ne. 1) return
02879
02880 labtyp= cxylab(nbase)
02881 if ((labtyp .eq. 1) .or. (labtyp .eq. 0)) labtyp= cxytype(nbase)
02882
02883 100 continue ! outer loop
02884     if (nbase .eq. 1) then ! x-Achse
02885         iwidth= linwdt(cxywdth(nbase))
02886     else
02887         call csize(ih, iwidth)
02888     end if
02889
02890     imaxwid= iabs(cxysmax(nbase)-cxysmin(nbase))- 2*iwidth
02891     imaxwid= imaxwid* cxystep(nbase)* cxystag(nbase) / cxytics(nbase)
02892
02893     cxystep(nbase)= 1
02894     cxystag(nbase)= 1
02895
02896     if (iwidth .lt. imaxwid) return ! exit loop
02897
02898     if (nbase .eq. 1) then ! x-Achse
02899         cxystag(nbase)= 2
02900     else
02901         cxystep(nbase)= cxystep(nbase) + 1
02902     end if
02903
02904 110 continue ! inner loop
02905     if(iwidth .lt. imaxwid) return ! exit loop
02906     if(cxystep(nbase) .gt. cxytics(nbase)) return ! exit loop
02907     if (labtyp .ne. 3 .and. labtyp .ne. 6) then ! cycle inner loop
02908         cxystep(nbase)= cxystep(nbase)+1
02909         goto 110
02910     else ! cycle outer loop
02911         if (cxywdth(nbase) .eq. 3) return
02912         cxywdth(nbase)=3
02913         goto 100
02914     end if ! cycle until force exit
02915 end
02916
02917
02918
02919 C
02920 C  Tabellensuche und Rundungen
02921 C
02922
02923 real function findge (val,tab,in)
02924 implicit none
02925 integer in
02926 real val, tab(1)
02927
02928 100 if (tab(in) .lt. val) goto 110 ! while
02929     in= in-1
02930     goto 100
02931 110 continue ! endwhile
02932
02933 120 continue ! repeat
02934     in= in+1
02935     if (tab(in) .lt. val) goto 120 ! end repeat
02936     findge= tab(in)
02937     return
02938 end
02939
02940
02941
02942 real function findle (val,tab,in)

```

```

02943      implicit none
02944      integer in
02945      real val, tab(1)
02946      real valeps
02947
02948      valeps= val+ 1.e-7 ! Vergleich um 0 ermöglichen (Rechengenauigkeit!)
02949
02950 100    if (tab(in) .le. valeps) goto 110 ! while
02951        in= in-1
02952        goto 100
02953 110    continue ! endwhile
02954
02955 120    continue ! repeat
02956        in= in+1
02957        if (tab(in) .lt. valeps) goto 120 ! end repeat
02958        findle= tab(in-1)
02959        return
02960    end
02961
02962
02963
02964      integer function locge (ival,itab,in)
02965      implicit none
02966      integer ival, itab(1), in
02967
02968 100    if (itab(in) .lt. ival) goto 110 ! while
02969        in= in-1
02970        goto 100
02971 110    continue ! endwhile
02972
02973 120    continue ! repeat
02974        in= in+1
02975        if (itab(in) .lt. ival) goto 120 ! end repeat
02976        locge= itab(in)
02977        return
02978    end
02979
02980
02981
02982      integer function locle (ival,itab,in)
02983      implicit none
02984      integer ival, itab(1), in
02985
02986 100    if (itab(in) .le. ival) goto 110 ! while
02987        in= in-1
02988        goto 100
02989 110    continue ! endwhile
02990
02991 120    continue ! repeat
02992        in= in+1
02993        if (itab(in) .le. ival) goto 120 ! end repeat
02994        locle= itab(in-1)
02995        return
02996    end
02997
02998
02999
03000      real function roundd (value,finterval)
03001      implicit none
03002      real value,finterval
03003      integer ifrac
03004      real frac
03005
03006      frac= value/finterval
03007      ifrac= int(frac)
03008      if (real(ifrac) .gt. frac) ifrac= ifrac-1 ! Abrunden bei frac neg.
03009      roundd = real(ifrac) * finterval
03010      if (roundd .gt. value) roundd= value
03011      return
03012    end
03013
03014
03015
03016      real function roundu (value,finterval)
03017      implicit none
03018      real value,finterval
03019      integer ifrac
03020      real frac
03021
03022      frac= value/finterval
03023      ifrac= int(frac)
03024      if (real(ifrac) .lt. frac) ifrac= ifrac+1 ! Aufrunden bei frac pos.
03025      roundu = real(ifrac) * finterval
03026      if (roundu .lt. value) roundu= value
03027      return
03028    end
03029

```



```

03030
03031
03032 C
03033 C  Generelle Manipulationen der Commonvariablen
03034 C
03035     subroutine savcom (Array)
03036     implicit none
03037     integer array(1)
03038     include 'G2dAG2.fd'
03039
03040     integer i
03041     integer arr(1)
03042     equivalence(arr(1),cline)
03043     do 10 i=1,g2dag21
03044         array(i)= arr(i)
03045 10    continue
03046     return
03047     end
03048
03049
03050
03051     subroutine rescom (Array)
03052     implicit none
03053     integer array(1)
03054     include 'G2dAG2.fd'
03055
03056     integer i
03057     integer arr(1)
03058     equivalence(arr(1),cline)
03059     do 10 i=1,g2dag21
03060         arr(i)= array(i)
03061 10    continue
03062     return
03063     end
03064
03065
03066
03067     integer function iother (ipar)
03068     implicit none
03069     integer ipar
03070
03071     if (mod(ipar,2) .eq. 1) then ! ungerader Parameter=x-Achse
03072         iother= ipar+1
03073     else
03074         iother= ipar-1
03075     end if
03076     return
03077     end

```

3.3 AG2Holerith.for File Reference

Graph2D: deprecated AG2 routines.

Functions/Subroutines

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

- integer function [ibasex](#) (iPar)
- integer function [ibasey](#) (iPar)
- real function [comget](#) (iPar)
- subroutine [comset](#) (iPar, val)
- subroutine [comdmp](#)

3.3.1 Detailed Description

Graph2D: deprecated AG2 routines.

Version

2.2

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

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

Definition in file [AG2Holerith.for](#).

3.3.2 Function/Subroutine Documentation

3.3.2.1 [alfset\(\)](#)

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

Definition at line [45](#) of file [AG2Holerith.for](#).

3.3.2.2 [comdmp\(\)](#)

```
subroutine comdmp
```

Definition at line [328](#) of file [AG2Holerith.for](#).

3.3.2.3 comget()

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

Definition at line 271 of file [AG2Holerith.for](#).

3.3.2.4 comset()

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

Definition at line 299 of file [AG2Holerith.for](#).

3.3.2.5 eform()

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

Definition at line 173 of file [AG2Holerith.for](#).

3.3.2.6 expout()

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

Definition at line 90 of file [AG2Holerith.for](#).

3.3.2.7 fform()

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

Definition at line 189 of file [AG2Holerith.for](#).

3.3.2.8 fonly()

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

Definition at line 205 of file [AG2Holerith.for](#).

3.3.2.9 hlabel()

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

Definition at line 121 of file [AG2Holerith.for](#).

3.3.2.10 hstrin()

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

Definition at line 112 of file [AG2Holerith.for](#).

3.3.2.11 ibasec()

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

Definition at line 241 of file [AG2Holerith.for](#).

3.3.2.12 ibasex()

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

Definition at line 251 of file [AG2Holerith.for](#).

3.3.2.13 ibasey()

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

Definition at line 261 of file [AG2Holerith.for](#).

3.3.2.14 iform()

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

Definition at line 221 of file [AG2Holerith.for](#).

3.3.2.15 juster()

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

Definition at line 154 of file [AG2Holerith.for](#).

3.3.2.16 notate()

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

Definition at line 30 of file [AG2Holerith.for](#).

3.3.2.17 numset()

```

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

```

Definition at line 67 of file [AG2Holerith.for](#).

3.3.2.18 vlabel()

```

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

```

Definition at line 139 of file [AG2Holerith.for](#).

3.3.2.19 vstrin()

```

subroutine vstrin (
    integer, dimension(2) iarray )

```

Definition at line 130 of file [AG2Holerith.for](#).

3.4 AG2Holerith.for

```

00001 C> \file      AG2Holerith.for
00002 C> \version   2.2
00003 C> \author    (C) 2022 Dr.-Ing. Klaus Friedewald
00004 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00005 C> \~german
00006 C> \brief     Graph2D: obsolete AG2 Routinen
00007 C> \~english
00008 C> \brief     Graph2D: deprecated AG2 routines
00009 C> \~
00010 C>
00011 C> \~german
00012 C>     Unterprogramme zur Behandlung von Holerithvariablen und direkter
00013 C>     Manipulation des Commonblocks
00014 C>
00015 C> \~english
00016 C>     Compatibility routines dealing with holerith characters
00017 C>     and direct manipulation of common variables.
00018 C>
00019 C
00020 C
00021 C Tektronix Advanced Graphics 2 - Version 2.x
00022 C
00023 C     Optionale Unterprogramme
00024 C
00025 C
00026 C
00027 C Stringfunktionen fuer Holerithvariablen
00028 C
00029 C
00030     subroutine notate (ix,iy,lenchr,iarray)
00031     implicit none

```

```

00032      integer ix,iy,lenchr, iarray(lenchr)
00033      integer i
00034      character *(255) buf
00035
00036      do 100 i=1,lenchr
00037          buf(i:i)= char(iarray(i))
00038 100  continue
00039      call notatec (ix,iy,buf(1:lenchr))
00040      return
00041  end
00042
00043
00044
00045      subroutine alfset (fnum,kwidth,labtyp,ilabel)
00046      implicit none
00047      integer kwidth,labtyp, ilabel(kwidth)
00048      real fnum
00049      integer i, buflen
00050      character *(255) buf
00051      integer ISTRINGLEN
00052
00053      call alfsetc (fnum, labtyp, buf)
00054      buflen= istringlen(buf)
00055      do 100 i=1,kwidth
00056          if (i .le. buflen) then
00057              ilabel(i)= ichar(buf(i:i))
00058          else
00059              ilabel(i)= ichar(' ')
00060          end if
00061 100  continue
00062      return
00063  end
00064
00065
00066
00067      subroutine numset (fnum,iwidth,nbase,ilabel,ifill)
00068      implicit none
00069      integer iwidth,nbase,ilabel(iwidth),ifill
00070      real fnum
00071      integer i, iLeadFill
00072      character *(255) buf
00073      integer ISTRINGLEN
00074
00075      call numsetc (fnum,iwidth,nbase, buf)
00076      ileadfill= max(0,iwidth-istringlen(buf))
00077      do 100 i=1,iwidth
00078          ilabel(ileadfill+i)= ichar(buf(i:i))
00079 100  continue
00080      i=1 ! iLabel ist rechtsjustiert!
00081      if (i.gt.ileadfill) goto 110 ! while
00082          ilabel(i)= ifill
00083          i= i+1
00084 110  continue ! endwhile
00085      return
00086  end
00087
00088
00089
00090      subroutine expout (nbase,iexp,ilabel,nchars,ifill)
00091      implicit none
00092      integer nbase,iexp, nchars, ilabel(nchars), ifill
00093      integer i, iLeadFill
00094      character *(255) buf
00095      integer ISTRINGLEN
00096
00097      call expoutc (nbase,iexp, buf(1:nchars))
00098      ileadfill= max(0,nchars-istringlen(buf))
00099      do 100 i=1,nchars
00100          ilabel(ileadfill+i)= ichar(buf(i:i))
00101 100  continue
00102      i=1 ! iLabel ist rechtsjustiert!
00103      if (i.gt.ileadfill) goto 110 ! while
00104          ilabel(i)= ifill
00105          i= i+1
00106 110  continue ! endwhile
00107      return
00108  end
00109
00110
00111
00112      subroutine hstrin (iString)
00113      implicit none
00114      integer iString(2)
00115      call anstr (istring(1),istring(2))
00116      return
00117  end
00118

```

```

00119
00120
00121     subroutine hlabel (iLen, iString)
00122     implicit none
00123     integer iLen, iString(iLen)
00124     call anstr (ilen, istring)
00125     return
00126     end
00127
00128
00129
00130     subroutine vstrin (iarray)
00131     implicit none
00132     integer iarray(2)
00133     call vlabel (iarray(1),iarray(2))
00134     return
00135     end
00136
00137
00138
00139     subroutine vlabel (iLen,iString)
00140     implicit none
00141     integer iLen, iString(iLen)
00142     integer i
00143     character *(255) buf
00144     integer ISTRINGLEN
00145     do 100 i=1, ilen
00146         buf(i:i)= char(istring(i))
00147 100 continue
00148     call vlabelc (buf(:ilen))
00149     return
00150     end
00151
00152
00153
00154     subroutine juster (iLen,iString,iposflag,ifill,lenchr, ioff)
00155     implicit none
00156     integer iLen,iString(iLen), iposflag,ifill, lenchr, ioff
00157     integer i
00158     character *(255) buf
00159
00160     lenchr= 0
00161     do 100 i=1, ilen
00162         if ( (i .gt. 1) .or. (istring(i) .ne. ifill) ) then ! Ueberlese Startfillchars
00163             lenchr= lenchr+1
00164             buf(lenchr:lenchr)= char(abs(istring(i))) ! Tek Index -1,-2 -> char(1),char(2)
00165         end if
00166 100 continue
00167     call justerc (buf, iposflag, ioff)
00168     return
00169     end
00170
00171
00172
00173     subroutine eform (fnum,iwidth,idec,ilabel,ifill)
00174     implicit none
00175     integer iwidth,idec, ilabel(iwidth), ifill
00176     real fnum
00177     integer i
00178     character *(255) buf
00179
00180     call eformc (fnum,iwidth,idec, buf)
00181     do 100 i=1,iwidth
00182         ilabel(i)= ichar(buf(i:i))
00183 100 continue
00184     return
00185     end
00186
00187
00188
00189     subroutine fform (fnum,iwidth,idec,ilabel,ifill)
00190     implicit none
00191     integer iwidth,idec, ilabel(255), ifill
00192     real fnum
00193     integer i
00194     character *(255) buf
00195
00196     call fformc (fnum,iwidth,idec, buf)
00197     do 100 i=1,iwidth
00198         ilabel(i)= ichar(buf(i:i))
00199 100 continue
00200     return
00201     end
00202
00203
00204
00205     subroutine fonly (fnum,iwidth,idec,ilabel,ifill)

```



```

00206      implicit none
00207      integer iwidth,idec, ilabel(iwidth), ifill
00208      real fnum
00209      integer i
00210      character *(255) buf
00211
00212      call fonlyc (fnum,iwidth,idec, buf)
00213      do 100 i=1,iwidth
00214         ilabel(i)= ichar(buf(i:i))
00215 100    continue
00216      return
00217      end
00218
00219
00220
00221      subroutine iform (fnum,iwidth,ilabel,ifill)
00222      implicit none
00223      integer iwidth,idec, ilabel(iwidth), ifill
00224      real fnum
00225      integer i
00226      character *(255) buf
00227
00228      call iformc (fnum,iwidth,idec, buf)
00229      do 100 i=1,iwidth
00230         ilabel(i)= ichar(buf(i:i))
00231 100    continue
00232      return
00233      end
00234
00235
00236
00237 C
00238 C   Direkte Manipulation des Commonblocks
00239 C
00240
00241      integer function ibasec (iPar)
00242      implicit none
00243      integer ipar
00244
00245      ibasec= -1-ipar
00246      return
00247      end
00248
00249
00250
00251      integer function ibasex (ipar)
00252      implicit none
00253      integer ipar
00254
00255      ibasex= 1 + 2*ipar
00256      return
00257      end
00258
00259
00260
00261      integer function ibasey (ipar)
00262      implicit none
00263      integer ipar
00264
00265      ibasey= 2 + 2*ipar
00266      return
00267      end
00268
00269
00270
00271      real function comget (ipar)
00272      implicit none
00273      integer ipar
00274      include 'G2dAG2.fd'
00275
00276      integer iarr(1), iarr2(1)
00277      real arr(1), arr2(1)
00278      equivalence(iarr(1),cline), (iarr2(1),cxyneat)
00279      equivalence(arr(1),cline), (arr2(1),cxyneat)
00280
00281      if ((ipar.lt.0) .and. (ipar.ge. -9))then
00282         if ((ipar .eq. -4) .or. (ipar .le. -8)) then
00283            comget= arr(-ipar)
00284         else
00285            comget= real(iarr(-ipar))
00286         end if
00287      else if ((ipar.gt.0) .and. (ipar.le.56)) then
00288         if ((ipar.le.22) .or. ((ipar .ge. 27).and.(ipar.le.52))) then
00289            comget= real(iarr2(ipar))
00290         else
00291            comget= arr2(ipar)
00292         end if

```

```

00293     end if
00294     return
00295 end
00296
00297
00298
00299 subroutine comset (iPar,val)
00300 implicit none
00301 integer iPar
00302 real val
00303 include 'G2dAG2.fd'
00304
00305 integer iarr(1), iarr2(1)
00306 real arr(1), arr2(1)
00307 equivalence(iarr(1),cline), (iarr2(1),cxyneat)
00308 equivalence(arr(1),cline), (arr2(1),cxyneat)
00309
00310 if ((ipar.lt.0) .and. (ipar.ge. -9))then
00311   if ((ipar.eq.-4) .or. (ipar.le. -8)) then
00312     arr(-ipar)= val
00313   else
00314     iarr(-ipar)= int(val)
00315   end if
00316 else if ((ipar.gt.0) .and. (ipar.le.56)) then
00317   if ((ipar.le.22) .or. ((ipar.ge. 27).and.(ipar.le.52))) then
00318     iarr2(ipar)= int(val)
00319   else
00320     arr2(ipar)= val
00321   end if
00322 end if
00323 return
00324 end
00325
00326
00327
00328 subroutine comdmp
00329 implicit none
00330 integer i
00331 character *80 buf
00332 include 'G2dAG2.fd'
00333
00334 call erase
00335 call home
00336
00337 write (unit= buf,fmt=600, err=200) (cxyneat(i),i=1,2), cline
00338 600 format (1x,' 0: cxneat(1)=' ,i14,' , (2)=' ,i14,' , cline=' ,i14)
00339 call toutstc (buf)
00340 call newlin
00341 write (unit= buf,fmt=601, err=200) (cxyzero(i),i=1,2), csymb1
00342 601 format (1x,' 1: cxyzero(1)=' ,i14,' , (2)=' ,i14,' , csymb1=' ,i14)
00343 call toutstc (buf)
00344 call newlin
00345 write (unit= buf,fmt=602, err=200) (cxyloc(i),i=1,2), csteps
00346 602 format (1x,' 2: cxyloc(1)=' ,i14,' , (2)=' ,i14,' , csteps=' ,i14)
00347 call toutstc (buf)
00348 call newlin
00349 write (unit= buf,fmt=603, err=200) (cxylab(i),i=1,2), cinfin
00350 603 format (1x,' 3: cxylab(1)=' ,i14,' , (2)=' ,i14,' , cinfin=' ,e14.7)
00351 call toutstc (buf)
00352 call newlin
00353 write (unit= buf,fmt=604, err=200) (cxyden(i),i=1,2), cnpts
00354 604 format (1x,' 4: cxyden(1)=' ,i14,' , (2)=' ,i14,' , cnpts=' ,i14)
00355 call toutstc (buf)
00356 call newlin
00357 write (unit= buf,fmt=605, err=200) (cxytics(i),i=1,2), cstepl
00358 605 format (1x,' 5: cxytics(1)=' ,i14,' , (2)=' ,i14,' , cstepl=' ,i14)
00359 call toutstc (buf)
00360 call newlin
00361 write (unit= buf,fmt=606, err=200) (cxylen(i),i=1,2), cnumbr
00362 606 format (1x,' 6: cxylen(1)=' ,i14,' , (2)=' ,i14,' , cnumbr=' ,i14)
00363 call toutstc (buf)
00364 call newlin
00365 write (unit= buf,fmt=607, err=200) (cxyfrm(i),i=1,2), csizes
00366 607 format (1x,' 7: cxyfrm(1)=' ,i14,' , (2)=' ,i14,' , csizes=' ,e14.7)
00367 call toutstc (buf)
00368 call newlin
00369 write (unit= buf,fmt=608, err=200) (cxymtcs(i),i=1,2), csizel
00370 608 format (1x,' 8: cxymtcs(1)=' ,i14,' , (2)=' ,i14,' , csizel=' ,e14.7)
00371 call toutstc (buf)
00372 call newlin
00373 write (unit= buf,fmt=609, err=200) (cxymfrm(i),i=1,2)
00374 609 format (1x,' 9: cxymfrm(1)=' ,i14,' , (2)=' ,i14)
00375 call toutstc (buf)
00376 call newlin
00377 write (unit= buf,fmt=610, err=200) (cxydec(i),i=1,2)
00378 610 format (1x,'10: cxydec(1)=' ,i14,' , (2)=' ,i14)
00379 call toutstc (buf)

```

```

00380      call newlin
00381      write (unit= buf,fmt=611, err=200) (cxydmin(i),i=1,2)
00382 611 format (1x,'11: cxydmin(1)=' ,e14.7,' , (2)=' ,e14.7)
00383      call toutstc (buf)
00384      call newlin
00385      write (unit= buf,fmt=612, err=200) (cxydmax(i),i=1,2)
00386 612 format (1x,'12: cxydmax(1)=' ,e14.7,' , (2)=' ,e14.7)
00387      call toutstc (buf)
00388      call newlin
00389      write (unit= buf,fmt=613, err=200) (cxysmin(i),i=1,2)
00390 613 format (1x,'13: cxysmin(1)=' ,i14,' , (2)=' ,i14)
00391      call toutstc (buf)
00392      call newlin
00393      write (unit= buf,fmt=614, err=200) (cxysmax(i),i=1,2)
00394 614 format (1x,'14: cxysmax(1)=' ,i14,' , (2)=' ,i14)
00395      call toutstc (buf)
00396      call newlin
00397      write (unit= buf,fmt=615, err=200) (cxytype(i),i=1,2)
00398 615 format (1x,'15: cxytype(1)=' ,i14,' , (2)=' ,i14)
00399      call toutstc (buf)
00400      call newlin
00401      write (unit= buf,fmt=616, err=200) (cxylsig(i),i=1,2)
00402 616 format (1x,'16: cxylsig(1)=' ,i14,' , (2)=' ,i14)
00403      call toutstc (buf)
00404      call newlin
00405      write (unit= buf,fmt=617, err=200) (cxywdth(i),i=1,2)
00406 617 format (1x,'17: cxywdth(1)=' ,i14,' , (2)=' ,i14)
00407      call toutstc (buf)
00408      call newlin
00409      write (unit= buf,fmt=618, err=200) (cxyepon(i),i=1,2)
00410 618 format (1x,'18: cxyepon(1)=' ,i14,' , (2)=' ,i14)
00411      call toutstc (buf)
00412      call newlin
00413      write (unit= buf,fmt=619, err=200) (cxystep(i),i=1,2)
00414 619 format (1x,'19: cxystep(1)=' ,i14,' , (2)=' ,i14)
00415      call toutstc (buf)
00416      call newlin
00417      write (unit= buf,fmt=620, err=200) (cxystag(i),i=1,2)
00418 620 format (1x,'20: cxystag(1)=' ,i14,' , (2)=' ,i14)
00419      call toutstc (buf)
00420      call newlin
00421      write (unit= buf,fmt=621, err=200) (cxyetyp(i),i=1,2)
00422 621 format (1x,'21: cxyetyp(1)=' ,i14,' , (2)=' ,i14)
00423      call toutstc (buf)
00424      call newlin
00425      write (unit= buf,fmt=622, err=200) (cxybeg(i),i=1,2)
00426 622 format (1x,'22: cxybeg(1)=' ,i14,' , (2)=' ,i14)
00427      call toutstc (buf)
00428      call newlin
00429      write (unit= buf,fmt=623, err=200) (cxyend(i),i=1,2)
00430 623 format (1x,'23: cxyend(1)=' ,i14,' , (2)=' ,i14)
00431      call toutstc (buf)
00432      call newlin
00433      write (unit= buf,fmt=624, err=200) (cxymbeg(i),i=1,2)
00434 624 format (1x,'24: cxymbeg(1)=' ,i14,' , (2)=' ,i14)
00435      call toutstc (buf)
00436      call newlin
00437      write (unit= buf,fmt=625, err=200) (cxymend(i),i=1,2)
00438 625 format (1x,'25: cxymend(1)=' ,i14,' , (2)=' ,i14)
00439      call toutstc (buf)
00440      call newlin
00441      write (unit= buf,fmt=626, err=200) (cxyamin(i),i=1,2)
00442 626 format (1x,'26: cxyamin(1)=' ,e14.7,' , (2)=' ,e14.7)
00443      call toutstc (buf)
00444      call newlin
00445      write (unit= buf,fmt=627, err=200) (cxyamax(i),i=1,2)
00446 627 format (1x,'27: cxyamax(1)=' ,e14.7,' , (2)=' ,e14.7)
00447      call toutstc (buf)
00448
00449      call graphicerror (11,char(0))
00450      call erase
00451
00452 200 continue
00453      return
00454      end

```

3.5 AG2uline.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine [uline](#) (x, y, i)

3.5.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2uline.for](#).

3.5.2 Function/Subroutine Documentation

3.5.2.1 [uline\(\)](#)

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

Definition at line 10 of file [AG2uline.for](#).

3.6 AG2uline.for

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

3.7 AG2umnmx.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine [umnmx](#) (array, amin, amax)

3.7.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2umnmx.for](#).

3.7.2 Function/Subroutine Documentation

3.7.2.1 umnmx()

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

Definition at line 9 of file [AG2umnmx.for](#).

3.8 AG2umnmx.for

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

3.9 AG2upoint.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- real function [upoint](#) (arr, ii, oldone)

3.9.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2upoint.for](#).

3.9.2 Function/Subroutine Documentation

3.9.2.1 upoint()

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

Definition at line 9 of file [AG2upoint.for](#).

3.10 AG2upoint.for

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

3.11 AG2users.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine [users](#) (x, y, i)

3.11.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2users.for](#).

3.11.2 Function/Subroutine Documentation

3.11.2.1 users()

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

Definition at line 9 of file [AG2users.for](#).

3.12 AG2users.for

```

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

```

3.13 AG2useset.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine [useset](#) (fnum, iwidth, nbase, labeli)

3.13.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2useset.for](#).

3.13.2 Function/Subroutine Documentation

3.13.2.1 useset()

```

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

```

Definition at line 9 of file [AG2useset.for](#).

3.14 AG2useset.for

```

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

```

3.15 AG2usesetC.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine [usesetc](#) (fnum, iwidth, nbase, labstr)

3.15.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2usesetC.for](#).

3.15.2 Function/Subroutine Documentation

3.15.2.1 usesetc()

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

Definition at line 9 of file [AG2usesetC.for](#).

3.16 AG2usesetC.for

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


3.17 AG2UsrSoftek.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine [softek](#) (isym)

3.17.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file [AG2UsrSoftek.for](#).

3.17.2 Function/Subroutine Documentation

3.17.2.1 [softek\(\)](#)

```
subroutine softek (  
    isym )
```

Definition at line 9 of file [AG2UsrSoftek.for](#).

3.18 AG2UsrSoftek.for

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

3.19 Fgraph.fd File Reference

DOS Port: Declarations OW graph.lib.

3.19.1 Detailed Description

DOS Port: Declarations OW graph.lib.

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

Author

Dr.-Ing. Klaus Friedewald

Note

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

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

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

Definition in file [Fgraph.fd](#).

3.20 Fgraph.fd

```

00001 C> \file      Fgraph.fd
00002 C> \brief    DOS Port: Declarations OW graph.lib
00003 C>
00004 C> \~german
00005 C> Konstanten und Funktionen der Watcom DOS Graphik-Library. Ersatz für das zum
00006 C> Microsoft Fortran-Compiler gehörende INCLUDE-File, abgeleitet aus dem
00007 C> Watcom-Headerfile graph.fi.
00008 C>
00009 C> \~english
00010 C> Functions and constants of the Watcom DOS Graphic Library. Substitution for
00011 C> the INCLUDE-file of the Microsoft Fortran Compiler, derived from the
00012 C> Watcom Headerfile graph.fi.
00013 C>
00014 C> \~
00015 C> \author    Dr.-Ing. Klaus Friedewald
00016 C>
00017 C> \~german
00018 C> \note
00019 C> Der Watcom Compiler erlaubt 32 Zeichen lange Variablenamen unter Verwendung
00020 C> von $ und _. Deswegen sind $nottruncate und $notstrict überflüssig.
00021 C>
00022 C> \note
00023 C> Hex-Zahlen werden nicht durch \#ff sondern durch 'ff'x dargestellt.
00024 C>
00025 C> \note
00026 C> Die OpenWatcom Library graph.lib ist nicht Bestandteil von Graph2Ddos.lib
00027 C> und muss bei den Linkoptionen der Hauptprogramme aufgeführt werden:
00028 C> -libr graph.
00029 C> \~english
00030 C> \note
00031 C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
00032 C> contain $ and _. That for $nottruncate und $notstrict are superfluous.
00033 C>
00034 C> \note
00035 C> Hexadecimal numbers are represented by 'ff'x instead of \#ff.
00036 C>
00037 C> \note
00038 C> The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to
00039 C> be linked to the main programs:
00040 C> -libr graph.
00041 C> \~
00042 C>
00043 C> \cond
00044
00045     structure/videoconfig/      ! structure for getvideoconfig
00046     integer*2 numxpixels

```

```

00047         integer*2 numypixels
00048         integer*2 numtextcols
00049         integer*2 numtextrows
00050         integer*2 numcolors
00051         integer*2 bitsperpixel
00052         integer*2 numvideopages
00053         integer*2 mode
00054         integer*2 adapter
00055         integer*2 monitor
00056         integer*2 memory
00057     end structure
00058
00059     structure/xycoord/           ! structure for pixel position
00060         integer*2 xcoord
00061         integer*2 ycoord
00062     end structure
00063
00064     structure/rccoord/           ! structure for text position
00065         integer*2 row
00066         integer*2 col
00067     end structure
00068
00069 C Videomodes
00070
00071     integer*2, $MAXRESMODE, $MAXCOLORMODE, $DEFAULTMODE,$TEXTBW40,
00072     1      $TEXTC40,$TEXTBW80,$TEXTC80, $MRES4COLOR,$MRESNOCOLOR,
00073     2      $HRESBW,$TEXTMONO,$HERCMONO, $MRES16COLOR,$HRES16COLOR,
00074     3      $ERESNOCOLOR,$ERESCOLOR, $VRES2COLOR,$VRES16COLOR,
00075     4      $MRES256COLOR,$ORESOLOR
00076     parameter($maxresmode ==-3)      ! graphics mode with highest resolution
00077     parameter($maxcolormode ==-2)    ! graphics mode with most colors
00078     parameter($defaultmode ==-1)     ! restore screen to original mode
00079     parameter($textbw40 =0)          ! 40 x 25 text, 16 grey
00080     parameter($textc40 =1)           ! 40 x 25 text, 16/8 color
00081     parameter($textbw80 =2)          ! 80 x 25 text, 16 grey
00082     parameter($textc80 =3)           ! 80 x 25 text, 16/8 color
00083     parameter($mres4color =4)        ! 320 x 200, 4 color
00084     parameter($mresnocolor =5)       ! 320 x 200, 4 grey
00085     parameter($hresbw =6)            ! 640 x 200, BW
00086     parameter($textmono =7)          ! 80 x 25 text, BW
00087     parameter($hercmmono =8)         ! 720 x 348, BW for HGC
00088     parameter($mres16color =13)      ! 320 x 200, 16 color
00089     parameter($hres16color =14)      ! 640 x 200, 16 color
00090     parameter($eresnocolor =15)      ! 640 x 350, BW
00091     parameter($erescolor =16)        ! 640 x 350, 4 or 16 color
00092     parameter($vres2color =17)       ! 640 x 480, BW
00093     parameter($vres16color =18)      ! 640 x 480, 16 color
00094     parameter($mres256color =19)     ! 320 x 200, 256 color
00095     parameter($orescolor =64)        ! 640 x 400, 1 of 16 colors (Olivetti)
00096
00097     integer*4 $MDPA,$CGA,$EGA,$MCGA,$VGA,$HGC,$OCGA,$OEGA,$OVGA
00098     parameter($mdpa ='0001'x)        ! Monochrome Display Adapter (MDPA)
00099     parameter($cga ='0002'x)         ! Color Graphics Adapter (CGA)
00100     parameter($ega ='0004'x)         ! Enhanced Graphics Adapter (EGA)
00101     parameter($vga ='0008'x)         ! Video Graphics Array (VGA)
00102     parameter($mcca ='0010'x)        ! MultiColor Graphics Array (MCGA)
00103     parameter($hgc ='0020'x)         ! Hercules Graphics Card (HGC)
00104     parameter($ocga ='0042'x)        ! Olivetti Color Graphics Adapter (OCGA)
00105     parameter($oega ='0044'x)        ! Olivetti Enhanced Graphics Adapter (OEGA)
00106     parameter($ovga ='0048'x)        ! Olivetti Video Graphics Array (OVGA)
00107
00108     integer*4 $MONO,$COLOR,$ENHCOLOR,$ANALOGMONO,$ANALOGCOLOR,$ANALOG
00109     parameter($mono ='0001'x)        ! Monochrome
00110     parameter($color ='0002'x)       ! Color (or Enhanced emulating color)
00111     parameter($enhcolor ='0004'x)    ! Enhanced Color
00112     parameter($analogmono ='0008'x)  ! Analog Monochrome only
00113     parameter($analogcolor ='0010'x) ! Analog Color only
00114     parameter($analog ='0018'x)      ! Analog
00115
00116 C Plotting Action
00117
00118     integer*2 $GBORDER,$GFILLINTERIOR,
00119     1      $GCLEARSCREEN, $GVIEWPORT,$GWINDOW
00120
00121     parameter($gborder =2)            ! draw outline only
00122     parameter($gfillinterior =3)      ! fill using current fill mask
00123
00124     parameter($gclearscreen=0)
00125     parameter($gviewport =1)
00126     parameter($gwindow =2)
00127
00128     integer*4 $GCURSOROFF,$GCURSORON,$GWRAPOFF,$GWRAPON
00129     parameter($gcursoroff=0)
00130     parameter($gcursoron =1)
00131
00132     parameter($gwrapoff =0)
00133     parameter($gwrapon =1)

```

```

00134
00135     integer*4 $GSCROLLUP, $GSCROLLOWDOWN
00136     parameter($gscrollup =1)
00137     parameter($gscrolldown =-1)
00138
00139     integer*4 $MAXTEXTROWS
00140     parameter($maxtextrows =-1)
00141
00142     integer*4 $GPSET,$GPRESET,$GAND,$GOR,$GXOR
00143     parameter($gpset =3)
00144     parameter($gpreset =2)
00145     parameter($gand =1)
00146     parameter($gor =0)
00147     parameter($gxor =4)
00148
00149     integer*4 $BLACK,$BLUE,$GREEN,$CYAN,$RED,$MAGENTA,$BROWN,
00150     1 $WHITE,$GRAY, $LIGHTBLUE,$LIGHTGREEN,$LIGHTCYAN,
00151     2 $LIGHTRED,$LIGHTMAGENTA, $LIGHTYELLOW,$BRIGHTWHITE
00152     parameter($black = '000000'x)
00153     parameter($blue = '2a0000'x)
00154     parameter($green = '002a00'x)
00155     parameter($cyan = '2a2a00'x)
00156     parameter($red = '00002a'x)
00157     parameter($magenta = '2a002a'x)
00158     parameter($brown = '00152a'x)
00159     parameter($white = '2a2a2a'x)
00160     parameter($gray = '151515'x)
00161     parameter($lightblue = '3f1515'x)
00162     parameter($lightgreen = '153f15'x)
00163     parameter($lightcyan = '3f3f15'x)
00164     parameter($lightred = '15153f'x)
00165     parameter($lightmagenta = '3f153f'x)
00166     parameter($lightyellow = '153f3f'x)
00167     parameter($brightwhite = '3f3f3f'x)
00168
00169     integer*4 $MODEFOFF,$MODEFOFFTOON,$MODEFOFFTOHI,$MODEFONTOOFF,
00170     1 $MODEFON,$MODEFONTOHI,$MODEFHITOOFF,$MODEFHITON,
00171     2 $MODEFHI
00172     parameter($modefoff =0)
00173     parameter($modefofftoon =1)
00174     parameter($modefofftohi =2)
00175     parameter($modefontooff =3)
00176     parameter($modefon =4)
00177     parameter($modefontohi =5)
00178     parameter($modefhitooff =6)
00179     parameter($modefhitoon =7)
00180     parameter($modefhi =8)
00181
00182     integer*4 $MODE7OFF,$MODE7ON,$MODE7HI
00183     parameter($mode7off =0)
00184     parameter($mode7on =1)
00185     parameter($mode7hi =2)
00186
00187 C external functions
00188
00189     external setvideomode
00190     integer*2 setvideomode
00191
00192     external setvideomoderows
00193     integer*2 setvideomoderows
00194
00195     external setactivepage
00196     integer*2 setactivepage
00197
00198     external setvisualpage
00199     integer*2 setvisualpage
00200
00201     external getactivepage
00202     integer*2 getactivepage
00203
00204     external getvisualpage
00205     integer*2 getvisualpage
00206
00207     external getvideoconfig
00208     external setvieworg
00209     external getviewcoord
00210     external getphyscoord
00211     external setcliprgn
00212     external setviewport
00213     external clearscreen
00214     external moveto
00215     external getcurrentposition
00216
00217     external lineto
00218     integer*2 lineto
00219
00220     external rectangle

```

```

00221     integer*2 rectangle
00222
00223     external ellipse
00224     integer*2 ellipse
00225
00226     external arc
00227     integer*2 arc
00228
00229     external pie
00230     integer*2 pie
00231
00232     external setpixel
00233     integer*2 setpixel
00234
00235     external getpixel
00236     integer*2 getpixel
00237
00238     external floodfill
00239     integer*2 floodfill
00240
00241     external setcolor
00242     integer*2 setcolor
00243
00244     external getcolor
00245     integer*2 getcolor
00246
00247     external setlinestyle
00248
00249     external getlinestyle
00250     integer*2 getlinestyle
00251
00252     external setfillmask
00253     external getfillmask
00254
00255     external setbkcolor
00256     integer*4 setbkcolor
00257
00258     external getbkcolor
00259     integer*4 getbkcolor
00260
00261     external remappalette
00262     integer*4 remappalette
00263
00264     external remapallpalette
00265     integer*2 remapallpalette
00266
00267     external selectpalette
00268     integer*2 selectpalette
00269
00270     external settextrrows
00271     integer*2 settextrrows
00272
00273     external settextrwindow
00274     external scrolltextwindow
00275     external outtext
00276
00277     external wrapon
00278     integer*2 wrapon
00279
00280     external displaycursor
00281     integer*2 displaycursor
00282
00283     external settextrcursor
00284     integer*2 settextrcursor
00285
00286     external gettextcursor
00287     integer*2 gettextcursor
00288
00289     external settextrposition
00290     external gettextposition
00291
00292     external settextrcolor
00293     integer*2 settextrcolor
00294
00295     external gettextcolor
00296     integer*2 gettextcolor
00297
00298     external getimage
00299     external putimage
00300
00301     external imagesize
00302     integer*4 imagesize
00303
00304
00305
00306     structure/wxycoord/      ! window coordinates
00307     double precision wx

```

```

00308         double precision wy
00309     end structure
00310
00311     external setwindow
00312     integer*2 setwindow
00313
00314     external getwindowcoord
00315     external getviewcoord_w
00316     external getcurrentposition_w
00317
00318
00319     external arc_w
00320     integer*2 arc_w
00321
00322     external ellipse_w
00323     integer*2 ellipse_w
00324
00325     external floodfill_w
00326     integer*2 floodfill_w
00327
00328     external getpixel_w
00329     integer*2 getpixel_w
00330
00331     external lineto_w
00332     integer*2 lineto_w
00333
00334     external moveto_w
00335
00336     external pie_w
00337     integer*2 pie_w
00338
00339     external rectangle_w
00340     integer*2 rectangle_w
00341
00342     external setpixel_w
00343     integer*2 setpixel_w
00344
00345     external getimage_w
00346
00347     external imagesize_w
00348     integer*2 imagesize_w
00349
00350     external putimage_w
00351
00352     structure/fontinfo/
00353         integer*2 type           ! b0 set = vector,clear = bit map
00354         integer*2 ascent        ! pix dist from top to baseline
00355         integer*2 pixwidth      ! character width in pixels, 0=prop
00356         integer*2 pixheight     ! character height in pixels
00357         integer*2 avgwidth      ! average character width in pixels
00358         character*81 filename    ! file name including path
00359         character*32 facename   ! font name
00360     end structure
00361
00362
00363     integer*2 $NO_SPACE, $FIXED_SPACE, $PROP_SPACE
00364     parameter($no_space = 0)
00365     parameter($fixed_space = 1)
00366     parameter($prop_space = 2)
00367
00368     integer*2 $NO_FONT_MAP, $VECTOR_MAP, $BIT_MAP
00369     parameter($no_font_map = 0)
00370     parameter($vector_map = 1)
00371     parameter($bit_map = 2)
00372
00373     external registerfonts
00374     integer*2 registerfonts
00375
00376     external unregisterfonts
00377
00378     external setfont
00379     integer*2 setfont
00380
00381     external getfontinfo
00382     integer*2 getfontinfo
00383
00384     external outgtext
00385
00386     external getgtextextent
00387     integer*2 getgtextextent
00388 C
00389 C> \endcond

```

3.21 Fgraph.fi File Reference

DOS Port: Interface OW graph.lib.

3.21.1 Detailed Description

DOS Port: Interface OW graph.lib.

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

Author

Dr.-Ing. Klaus Friedewald

Note

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

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

Definition in file [Fgraph.fi](#).

3.22 Fgraph.fi

```
00001 C> \file      Fgraph.fi
00002 C> \brief    DOS Port: Interface OW graph.lib
00003 C>
00004 C> \~german
00005 C> Interfacedeklaration der Watcom DOS Graphik-Library. Ersatz für das zum
00006 C> Microsoft Fortran-Compiler gehörende INCLUDE-File, abgeleitet aus dem
00007 C> Watcom-Headerfile graphapi.fi.
00008 C>
00009 C> \~english
00010 C> Interface definition for the Watcom DOS Graphic Library. Substitutes
00011 C> the INCLUDE-file of the Microsoft Fortran Compiler, derived from the
00012 C> Watcom headerfile graphapi.fi.
00013 C>
00014 C> \~
00015 C> \author    Dr.-Ing. Klaus Friedewald
00016 C>
00017 C> \~german
00018 C> \note
00019 C> Der Watcom Compiler erlaubt 32 Zeichen lange Variablennamen unter Verwendung
00020 C> von $ und _. Deswegen sind $nottruncate und $notstrict überflüssig.
00021 C>
00022 C> \note
00023 C> Die OpenWatcom Library graph.lib ist nicht Bestandteil von Graph2Ddos.lib
00024 C> und muss bei den Linkoptionen der Hauptprogramme aufgeführt werden:
00025 C> -libr graph.
00026 C> \~english
00027 C> \note
00028 C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
00029 C> contain $ and _. That for $nottruncate und $notstrict are superfluous.
00030 C>
00031 C> \note
00032 C> The Watcom library graph.lib ist not included in Graph2Ddos.lib and has to
00033 C> be linked to the main programs:
00034 C> -libr graph.
00035 C> \~
00036 C>
00037
00038
00039 c$pragma aux arc "_arc_" parm (VALUE*2)
```

```
00040
00041 c$pragma aux arc_w "_arc_w_" parm (VALUE*8)
00042
00043 c$pragma aux clearscreen "_clearscreen_" parm (VALUE*2)
00044
00045 c$pragma aux displaycursor "_displaycursor_" parm (VALUE*2)
00046
00047 c$pragma aux ellipse "_ellipse_" parm (VALUE*2)
00048
00049 c$pragma aux ellipse_w "_ellipse_w_" parm (VALUE*2, VALUE*8)
00050
00051 c$pragma aux floodfill "_floodfill_" parm (VALUE*2)
00052
00053 c$pragma aux floodfill_w "_floodfill_w_" parm (VALUE*8, VALUE*8, VALUE*2)
00054
00055 c$pragma aux getactivepage "_getactivepage_"
00056
00057 c$pragma aux getbkcolor "_getbkcolor_"
00058
00059 c$pragma aux getcolor "_getcolor_"
00060
00061 c$pragma aux getcurrentposition "_getcurrentposition_" parm (REFERENCE FAR)
00062
00063 c$pragma aux getcurrentposition_w "_getcurrentposition_w_" parm (REFERENCE FAR)
00064
00065 c$pragma aux getfillmask "_getfillmask_" parm (REFERENCE FAR)
00066
00067 c$pragma aux getimage "_getimage_" parm (VALUE*2,VALUE*2,VALUE*2,VALUE*2, \
00068 c REFERENCE FAR)
00069
00070 c$pragma aux getimage_w "_getimage_w_" parm (VALUE*8,VALUE*8,VALUE*8, \
00071 c VALUE*8,REFERENCE FAR)
00072
00073 c$pragma aux getlinestyle "_getlinestyle_"
00074
00075 c$pragma aux getphyscoord "_getphyscoord_" parm (VALUE*2,VALUE*2, \
00076 c REFERENCE FAR)
00077
00078 c$pragma aux getpixel "_getpixel_" parm (VALUE*2)
00079
00080 c$pragma aux getpixel_w "_getpixel_w_" parm (VALUE*8)
00081
00082 c$pragma aux gettextcolor "_gettextcolor_"
00083
00084 c$pragma aux gettextcursor "_gettextcursor_"
00085
00086 c$pragma aux gettextposition "_gettextposition_" parm (REFERENCE FAR)
00087
00088 c$pragma aux getvideoconfig "_getvideoconfig_" parm (REFERENCE FAR)
00089
00090 c$pragma aux getviewcoord "_getviewcoord_" parm (VALUE*2,VALUE*2, \
00091 c REFERENCE FAR)
00092
00093 c$pragma aux getviewcoord_w "_getviewcoord_w_" parm (VALUE*8,VALUE*8, \
00094 c REFERENCE FAR)
00095
00096 c$pragma aux getvisualpage "_getvisualpage_"
00097
00098 c$pragma aux getwindowcoord "_getwindowcoord_" parm (VALUE*2,VALUE*2, \
00099 c REFERENCE FAR)
00100
00101 c$pragma aux imagesize "_imagesize_" parm (VALUE*2)
00102
00103 c$pragma aux imagesize_w "_imagesize_w_" parm (VALUE*8)
00104
00105 c$pragma aux lineto "_lineto_" parm (VALUE*2)
00106
00107 c$pragma aux lineto_w "_lineto_w_" parm (VALUE*8)
00108
00109 c$pragma aux moveto "_moveto_" parm (VALUE*2,VALUE*2,REFERENCE FAR)
00110
00111 c$pragma aux moveto_w "_moveto_w_" parm (VALUE*8,VALUE*8,REFERENCE FAR)
00112
00113 c$pragma aux _outtext "_outtext_" parm (DATA_REFERENCE FAR)
00114
00115 c$pragma aux pie "_pie_" parm (VALUE*2)
00116
00117 c$pragma aux pie_w "_pie_w_" parm (VALUE*2,VALUE*8)
00118
00119 c$pragma aux putimage "_putimage_" parm (VALUE*2,VALUE*2,REFERENCE FAR,VALUE*2)
00120
00121 c$pragma aux putimage_w "_putimage_w_" parm (VALUE*8,VALUE*8, \
00122 c REFERENCE FAR,VALUE*2)
00123
00124 c$pragma aux rectangle "_rectangle_" parm (VALUE*2)
00125
00126 c$pragma aux rectangle_w "_rectangle_w_" parm (VALUE*2,VALUE*8)
```



```

00127
00128 c$pragma aux remappalette "_remappalette_" parm (VALUE*2,VALUE*4)
00129
00130 c$pragma aux remapallpalette "_remapallpalette_" parm (VALUE*4)
00131
00132 c$pragma aux scrolltextwindow "_scrolltextwindow_" parm (VALUE*2)
00133
00134 c$pragma aux selectpalette "_selectpalette_" parm (VALUE*2)
00135
00136 c$pragma aux setactivepage "_setactivepage_" parm (VALUE*2)
00137
00138 c$pragma aux setbkcolor "_setbkcolor_" parm (VALUE*4)
00139
00140 c$pragma aux setcliprgn "_setcliprgn_" parm (VALUE*2)
00141
00142 c$pragma aux setcolor "_setcolor_" parm (VALUE*2)
00143
00144 c$pragma aux setfillmask "_setfillmask_" parm (REFERENCE FAR)
00145
00146 c$pragma aux setlinestyle "_setlinestyle_" parm (VALUE*2)
00147
00148 c$pragma aux setpixel "_setpixel_" parm (VALUE*2)
00149
00150 c$pragma aux setpixel_w "_setpixel_w_" parm (VALUE*8)
00151
00152 c$pragma aux settextcolor "_settextcolor_" parm (VALUE*2)
00153
00154 c$pragma aux settextcursor "_settextcursor_" parm (VALUE*2)
00155
00156 c$pragma aux settextposition "_settextposition_" parm (VALUE*2,VALUE*2, \
00157 c REFERENCE FAR)
00158
00159 c$pragma aux settextrows "_settextrows_" parm (VALUE*2)
00160
00161 c$pragma aux settextwindow "_settextwindow_" parm (VALUE*2)
00162
00163 c$pragma aux setvideomode "_setvideomode_" parm (VALUE*2)
00164
00165 c$pragma aux setvideomoderows "_setvideomoderows_" parm (VALUE*2)
00166
00167 c$pragma aux setvieworg "_setvieworg_" parm (VALUE*2, VALUE*2,REFERENCE FAR)
00168
00169 c$pragma aux setviewport "_setviewport_" parm (VALUE*2)
00170
00171 c$pragma aux setvisualpage "_setvisualpage_" parm (VALUE*2)
00172
00173 c$pragma aux setwindow "_setwindow_" parm (VALUE*2,VALUE*8)
00174
00175 c$pragma aux wrapon "_wrapon_" parm (VALUE*2)
00176
00177
00178 c$pragma aux getfontinfo "_getfontinfo_" parm (REFERENCE FAR)
00179
00180 c$pragma aux getgtextextent "_getgtextextent_" parm (DATA_REFERENCE FAR)
00181
00182 c$pragma aux outgtext "_outgtext_" parm (DATA_REFERENCE FAR)
00183
00184 c$pragma aux registerfonts "_registerfonts_" parm (DATA_REFERENCE FAR)
00185
00186 c$pragma aux setfont "_setfont_" parm (DATA_REFERENCE FAR)
00187
00188 c$pragma aux unregisterfonts "_unregisterfonts_"

```

3.23 G2dAG2.fd File Reference

Graph2D: AG2 Common Block G2dAG2.

3.23.1 Detailed Description

Graph2D: AG2 Common Block G2dAG2.

Version

2.0

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Definition in file [G2dAG2.fd](#).

3.24 G2dAG2.fd

```

00001 C> \file      G2dAG2.fd
00002 C> \brief     Graph2D: AG2 Common Block G2dAG2
00003 C> \version    2.0
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C
00007 C Da die folgende Definition kein Bestandteil eines Moduls
00008 C ist versagt der DOXYGEN-Parser bei der Kombination von
00009 C COMMON und integer. Workaround: \\cond ... \\endcond
00010 C> \cond
00011
00012 C Common Block G2dAG2, Version 2.0 für AG2
00013 C Die Funktion der Variablen entspricht dem Tektronix AG2 User-Manual,
00014 C jedoch sind die achsenbezogenen Variablen in einem Feld zusammenge-
00015 C fasst. Die x-Achse wird durch Index=1, y durch Index=2 beschrieben.
00016 C
00017 integer      cline,csymb1,csteps ! ibase+ 0..2
00018 real         cfinfin ! 3
00019 integer      cnpts,cstepl,cnumbr ! 4..6
00020 real         csizes,csizel ! 7,8
00021
00022 logical      cxyneat(2),cxyzero(2) ! nbase+ 0, 1
00023 integer      cxyloc(2),cxylab(2),cxyden(2),cxytics(2) ! nbase+ 2..5
00024 integer      cxylon(2),cxyfrm(2),cxymtcs(2),cxymfrm(2),cxydec(2) ! 6..10
00025 real         cxydmin(2),cxydmax(2) ! 11,12
00026 integer      cxysmin(2),cxysmax(2),cxytype(2) ! 13..15
00027 integer      cxylsig(2),cxywidth(2),cxyepon(2) ! 16..18
00028 integer      cxystep(2),cxystag(2),cxyetyp(2) ! 19..21
00029 integer      cxybeg(2),cxyend(2),cxymbeg(2),cxymend(2) ! 22..25
00030 real         cxyamin(2),cxyamax(2) ! 26,27
00031
00032 common /g2dag2/
00033 C & extent,cvectr,xvectr,yvectr,
00034 C & xtentc,xtentx,xtenty,
00035 C
00036 & cline,csymb1,csteps,
00037 & cfinfin,
00038 & cnpts,cstepl,cnumbr,csizes,csizel,
00039 C
00040 & cxyneat,cxyzero,cxyloc,cxylab,cxyden,cxytics,
00041 & cxylon,cxyfrm,cxymtcs,cxymfrm,cxydec,
00042 & cxydmin,cxydmax,cxysmin,cxysmax,cxytype,
00043 & cxylsig,cxywidth,cxyepon,cxystep,cxystag,cxyetyp,
00044 & cxybeg,cxyend,cxymbeg,cxymend,cxyamin,cxyamax
00045 C
00046 C & reserv(8)
00047 save /g2dag2/
00048
00049 integer G2dAG2L ! Benoetigt von SAVCOM, RESCOM
00050 parameter(g2dag2l=65) ! integer, real und logical gleich lang!
00051 C> \endcond

```

3.25 GetHDC.for File Reference

Utility: Restore Hardcopies.

Functions/Subroutines

- logical function [gethdc](#) (Filnam)

3.25.1 Detailed Description

Utility: Restore Hardcopies.

Version

1.1

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Read and plot hardcopies

Temporary input unit: 41. If already used, an other channel will be searched.

Definition in file [GetHDC.for](#).

3.25.2 Function/Subroutine Documentation

3.25.2.1 gethdc()

```
logical function gethdc (  
    character *(*) Filnam )
```

Parameters

<i>FilNam</i>	Hardcopyfie
---------------	-------------

Returns

(optional) .true. -> Error

Definition at line 15 of file [GetHDC.for](#).

3.26 GetHDC.for

```
00001 C> \file      GetHDC.for  
00002 C> \brief      Utility: Restore Hardcopies  
00003 C> \version     1.1  
00004 C> \author      (C) 2023 Dr.-Ing. Klaus Friedewald  
00005 C> \copyright   GNU LESSER GENERAL PUBLIC LICENSE Version 3  
00006 C> \~german  
00007 C> Einlesen und Zeichnen von Hardcopydateien\n
```

```

00008 C> Verwendete temporaeres Ein/Ausgabeunit: 41. Falls bereits belegt, wird ein freier Kanal gesucht
00009 C> \~english
00010 C> Read and plot hardcopies\n
00011 C> Temporary input unit: 41. If already used, an other channel will be searched.
00012 C> \~
00013 C
00014
00015     logical function gethdc (Filnam)
00016 C> \param FilNam: Hardcopyfie
00017 C> \result (optional) .true. -> Error
00018     implicit none
00019     integer tcs_mesagelen, iunit
00020     parameter(tcs_mesagelen=132)
00021     character *(*) filnam
00022     logical iunitused
00023     character *(TCS_MESSAGELEN+1) txtstring
00024
00025     integer ios, idash, iprntlen, iactlen
00026     integer action, i1, i2
00027
00028     iunit= 40
00029     gethdc= .true.
00030
00031 5     continue ! repeat
00032         iunit= iunit+1
00033         inquire (unit=iunit, opened= iunitused)
00034         if (iunitused) goto 5
00035
00036         open (iunit,file=filnam,status='old',iostat=ios,form='formatted')
00037         if (ios.ne.0) then
00038             call graphicerror (6, ' ')
00039             return
00040         end if
00041
00042 10    continue ! repeat
00043         read (iunit, fmt='(i2,1x,i4,1x,i3)', iostat=ios)action, i1, i2
00044         if (ios.gt.0) then ! Error, not EOF
00045             call graphicerror (8, ' ')
00046             return
00047         end if
00048         if (action.eq.1) then ! XACTION_INITT
00049             call defaultcolour()
00050             call erase ()
00051         else if (action.eq.2) then ! XACTION_ERASE
00052             call erase ()
00053         else if (action.eq.3) then ! XACTION_MOVABS
00054             call movabs (i1,i2)
00055         else if (action.eq.4) then ! XACTION_DRWABS
00056             call drwabs (i1,i2)
00057         else if (action.eq.5) then ! XACTION_DSHSTYLE
00058             idash= i1
00059         else if (action.eq.6) then ! XACTION_DSHABS
00060             call dshabs (i1,i2,idash)
00061         else if (action.eq.7) then ! XACTION_PNTABS
00062             call pntabs (i1,i2)
00063         else if (action.eq.8) then ! XACTION_GTEXT
00064             iprntlen= i1
00065             if (iprntlen.gt.tcs_mesagelen) iprntlen= tcs_mesagelen
00066             txtstring(1:1)= char(i2)
00067             if (iprntlen.eq.1) then
00068                 txtstring= txtstring(1:1) // char(0)
00069                 call toutstc (txtstring)
00070             else
00071                 iactlen= 1
00072             end if
00073         else if (action.eq.9) then ! XACTION_ASCII
00074             if (iactlen.lt.iprntlen) then
00075                 iactlen= iactlen+1
00076                 txtstring(iactlen:iactlen)= char(i1)
00077             end if
00078             if (iactlen.lt.iprntlen) then
00079                 iactlen= iactlen+1
00080                 txtstring(iactlen:iactlen)= char(i2)
00081             end if
00082             if (iactlen.ge.iprntlen) then
00083                 txtstring(iactlen+1:iactlen+1) = char(0)
00084                 call toutstc (txtstring)
00085             end if
00086         else if (action.eq.10) then ! XACTION_BCKCOL
00087             call bckcol(i1)
00088         else if (action.eq.11) then ! XACTION_LINCOL
00089             call lincol (i1)
00090         else if (action.eq.12) then ! XACTION_TXTCOL
00091             call txtcol (i1)
00092         else if (action.eq.13) then ! XACTION_FONTATTR
00093             if (i1.eq.0) call italir()
00094             if (i1.eq.1) call italic()

```

```
00095         if (i2.eq.0) call nrmsiz()
00096         if (i2.eq.1) call dblsiz()
00097         else if (action.eq.14) then ! XACTION_NOOP
00098             continue
00099         else ! unknown
00100             continue
00101         end if
00102         if (ios.eq.0) goto 10 ! until EOF
00103
00104         close (iunit)
00105         gethdc= .false.
00106         return
00107     end
```

3.27 hdcopy.for File Reference

DOS Port: Hardcopy.

Functions/Subroutines

- subroutine [hdcopy](#)
- subroutine [writebuf](#) (iHandle, Buf, iPtr, iWrite)

3.27.1 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](#).

3.27.2 Function/Subroutine Documentation


```

00052         integer*4 PicWidth      ! Bildbreite Pixel, abgespeicherte Bytes durch 4 teilbar!
00053         integer*4 PicHeight     ! Bildhöhe in Pixel
00054         integer*2 iLayer        ! = 1
00055         integer*2 iBitPix       ! Bits per Pixel (1,4,8,24)
00056         integer*4 Kompr         ! Komprimierung =0 (ohne), 1 (RLE8), 2 (RLE4)
00057         integer*4 PicSiz        ! Bildgroesse in Byte
00058         integer*4 HorPixDen     ! Horizontale Auflösung Pixel/ Meter
00059         integer*4 VerPixDen     ! Vertikale Auflösung Pixel/ Meter
00060         integer*4 iCol          ! Anzahl benutzte Farben
00061         integer*4 iVIPCol       ! Anzahl wichtige Farben =0 (alle)
00062     end structure
00063     structure /rgbquad/
00064         integer*1 Blue
00065         integer*1 Green
00066         integer*1 Red
00067         integer*1 Reserved      ! =0
00068     end structure
00069     structure /fileheader/
00070         record /bitmapfileheader/ bfh
00071         record /bitmapinfoheader/ bih
00072         record /rgbquad/          palette(16)
00073     end structure
00074
00075     record /fileheader/ filhead
00076
00077     integer iWrtBuf
00078     parameter(iwrtbuf=650)
00079     integer*1 Buf(iWrtBuf)          ! > 2* (VGA-Auflösung/2)
00080     equivalence(buf,filhead)
00081
00082
00083     integer nByteRow
00084     integer iPtr, iPathlen
00085     integer*2 iHandle, ierr
00086     character*10 FilNam, Path*80
00087
00088     call graphicerror (10,' ') ! Hardcopy in progress
00089 c
00090 c Initialisierung Fileheader
00091 c
00092     nbyterow=(kscrx+7-mod(kscrx-1,8))/2 ! Byte pro Zeile durch 4 teilbar
00093     if (2*nbyterow.gt.iwrtbuf) then
00094         call graphicerror (8, ' ') ! Hardcopy: Write Buffer Overflow
00095     end if
00096
00097     filhead.bfh.datkennung= 19778 ! = 4d42h
00098
00099     filhead.bfh.reserved1= 0
00100     filhead.bfh.reserved2= 0
00101
00102     filhead.bfh.graphdatdst= 118 ! = 76h
00103     filhead.bfh.datsize=nbyterow*(kscry+1) + filhead.bfh.graphdatdst
00104
00105     filhead.bih.bmpinfhdsiz= 40 ! = 28h
00106     filhead.bih.picwidth= kscrx+1
00107     filhead.bih.picheight= kscry+1
00108
00109     filhead.bih.ilayer= 1
00110     filhead.bih.ibitpix=4          ! Auch bei Monochrom???
00111     filhead.bih.kompr= 0
00112     filhead.bih.picsiz= 0          ! nicht verwendet
00113     filhead.bih.horpixden= 0
00114     filhead.bih.verpixden= 0
00115     filhead.bih.icol= 0
00116     filhead.bih.ivipcol= 0
00117
00118     filhead.palette(1).red= 0
00119     filhead.palette(1).green= 0
00120     filhead.palette(1).blue= 0
00121
00122     filhead.palette(2).red= 0
00123     filhead.palette(2).green= 0
00124     filhead.palette(2).blue= 160
00125
00126     filhead.palette(3).red= 0
00127     filhead.palette(3).green= 160
00128     filhead.palette(3).blue= 0
00129
00130     filhead.palette(4).red= 0
00131     filhead.palette(4).green= 160
00132     filhead.palette(4).blue=160
00133
00134     filhead.palette(5).red= 160
00135     filhead.palette(5).green= 0
00136     filhead.palette(5).blue= 0
00137
00138     filhead.palette(6).red= 160

```

```

00139     filhead.palette(6).green= 0
00140     filhead.palette(6).blue= 160
00141
00142     filhead.palette(7).red= 160
00143     filhead.palette(7).green= 80
00144     filhead.palette(7).blue= 0
00145
00146     filhead.palette(8).red= 160
00147     filhead.palette(8).green= 160
00148     filhead.palette(8).blue= 160
00149
00150     filhead.palette(9).red= 80
00151     filhead.palette(9).green= 80
00152     filhead.palette(9).blue= 80
00153
00154     filhead.palette(10).red= 80
00155     filhead.palette(10).green= 80
00156     filhead.palette(10).blue= 240
00157
00158     filhead.palette(11).red= 80
00159     filhead.palette(11).green= 240
00160     filhead.palette(11).blue= 80
00161
00162     filhead.palette(12).red= 80
00163     filhead.palette(12).green= 240
00164     filhead.palette(12).blue= 240
00165
00166     filhead.palette(13).red= 240
00167     filhead.palette(13).green= 80
00168     filhead.palette(13).blue= 80
00169
00170     filhead.palette(14).red= 240
00171     filhead.palette(14).green= 80
00172     filhead.palette(14).blue= 240
00173
00174     filhead.palette(15).red= 240
00175     filhead.palette(15).green= 240
00176     filhead.palette(15).blue= 80
00177
00178     filhead.palette(16).red= 240
00179     filhead.palette(16).green= 240
00180     filhead.palette(16).blue= 240
00181
00182     do 3 i=1,16
00183 3       filhead.palette(i).reserved= 0
00184 c
00185 c Create Filename and open
00186 c
00187     path= 'SPL='//char(0)
00188     call getenv (path, len(path))
00189     ipathlen=istringlen(path)
00190
00191     i=0
00192 5     continue
00193     i= i+1
00194     write (filnam,fmt=300) i
00195     if (ipathlen.gt.0) then
00196         call openbytfil(ierr,ihandle,
00197 1         path(:ipathlen)//'\ '//filnam//char(0))
00198     else
00199         call openbytfil(ierr,ihandle, filnam//char(0))
00200     end if
00201     if (ierr.eq.80) goto 5 ! File exists - increase FilNam
00202     if (ierr.ne.0) call graphicerror (6, ' ') ! Hardcopy: Error during OPEN
00203 c
00204 c Zeilenweises Auslesen Bildschirmspeicher, Puffern und Fileausgabe
00205 c
00206     iptr= filhead.bfh.graphdatdst +1
00207
00208     do 20 iy=kscry,0,-1 ! oder 1?
00209     ix=0
00210 10    continue ! repeat
00211         buf(iptr)= ishl(getpixel(ix,iy),4)
00212         ix= ix+1
00213         if (ix.le.kscrx)buf(iptr)=buf(iptr).or. (getpixel(ix,iy).and.15)
00214         iptr= iptr+1
00215         ix=ix+1
00216         if (ix.le.kscrx) goto 10
00217         ix=ix ! Anzahl belegter Halfbytes
00218 15    if (ix.lt.2*nbyterow) then ! do while
00219             buf(iptr)= 0
00220             iptr= iptr+1
00221             ix=ix+2
00222             goto 15
00223         end if ! end while
00224         call writebuf (ihandle, buf(1),iptr, 256)
00225 20    continue

```



```

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

```

3.29 Mainpage.dox File Reference

3.30 outtext.for File Reference

DOS Port: alphanumeric output to the graphic screen.

Functions/Subroutines

- subroutine [outtext](#) (text)

3.30.1 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](#).

3.32.1 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](#).

3.32.2 Function/Subroutine Documentation

3.32.2.1 istringlen()

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

Definition at line [94](#) of file [Strings.for](#).

3.32.2.2 itrimlen()

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

Definition at line [133](#) of file [Strings.for](#).

3.32.2.3 printstring()

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

Definition at line [114](#) of file [Strings.for](#).

3.32.2.4 substitute()

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

Definition at line 30 of file [Strings.for](#).

3.33 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 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
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 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
00029 C
00030 C      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 C      implicit none
00039 C      integer iNext, iNext2, TempLen
00040 C      integer iStringLen
00041 C      character *(*) Source, Destination, Old1, New1
00042 C      character*255 temp, old, new
00043 C
00044 C      if (istringlen(old1).le.0) return
00045 C      if (istringlen(source) .le. 0) then
00046 C          destination= char(0)
00047 C          return
00048 C      end if
00049 C
00050 C      old= old1 // char(0)          ! old evtl. = Destination
00051 C      new= new1 // char(0)          ! => retten!
00052 C
00053 C      temp= source(1:istringlen(source)) // char(0) ! evtl. Ueberlappung!
00054 C      destination= temp
00055 C      inext= index( destination(:istringlen(destination)),
00056 C      1                                old(:istringlen(old)) )
00057 C      do while (inext.gt.0)
00058 C          if (inext.eq.1) then
00059 C              temp= destination
00060 C              if (new.eq.char(0)) then
00061 C                  destination= temp(istringlen(old)+1:)
00062 C              else
00063 C                  destination= new(:istringlen(new)) // temp(istringlen(old)+1:)
00064 C              end if
00065 C          else
00066 C              temp= destination(1:inext-1)
```

```

00067         templen= inext-1
00068         if (new.ne.char(0)) then
00069             temp= temp(1:templen)//new
00070             templen= templen+istringlen(new)
00071         end if
00072         if (inext+istringlen(old).lt.len(destination)) then
00073             temp= temp(1:templen)//destination(inext+istringlen(old):)
00074         end if
00075         destination= temp
00076     end if
00077     inext2= inext+istringlen(new)
00078     if (inext2.lt.len(destination)) then
00079         inext2= index(destination(inext2:), old(:istringlen(old)) )
00080     else
00081         inext2=0
00082     end if
00083     if (inext2.gt.0) then
00084         inext= inext+istringlen(new)+inext2-1
00085     else
00086         inext=0
00087     end if
00088 end do
00089 return
00090 end
00091
00092
00093
00094 function istringlen (String)
00095 C
00096 C Ermittelt die Stringlänge bei durch char(0) abgeschlossenen STRINGS.
00097 C Falls kein char(0) vorhanden ist, wird die Gesamtlänge übergeben.
00098 C
00099     implicit none
00100     character *(*) string
00101     integer istringlen, i
00102
00103     i= index(string,char(0))-1
00104     if (i.ge.0) then
00105         istringlen=i
00106     else
00107         istringlen= len(string)
00108     end if
00109     return
00110 end
00111
00112
00113
00114 character*(*) function printstring (String)
00115 C
00116 C Kopiert STRING in einen variabel langen PRINTSTRING. Hierdurch wird
00117 C der Ausdruck von Nullstrings (Fortran-Fehler!) vermieden.
00118 C
00119     implicit none
00120     character string *(*)
00121     integer istringlen
00122
00123     if (istringlen(string).gt.0) then
00124         printstring= string(1:istringlen(string))
00125     else
00126         printstring= ' '
00127     end if
00128     return
00129 end
00130
00131
00132
00133 integer function itrimlen (string)
00134 C
00135 C Bestimmt die Länge des Strings ohne angehängte Leerzeichen.
00136 C Bei Bedarf wird ein Char(0) angehaengt. Es darf in Ftn77 nie ein
00137 C Nullstring erzeugt werden, da sonst die RTL-Library abstuerzt. Deswegen
00138 C ist der kleinste erzeugte String ein Blank ' '.
00139 C
00140     implicit none
00141     character *(*) string
00142     integer i, istringlen
00143
00144     i=istringlen(string) +1
00145
00146 10 continue
00147     i= i-1
00148     if (i.ge.1) then
00149         if (string(i:i).eq.' ') goto 10
00150     end if
00151     itrimlen=i
00152     if ((i.lt.len(string)).and.(len(string).gt.1)) then
00153         string(i+1:i+1)= char(0) ! .gt.1: Achtung, nie Nullstring erzeugen!

```

```
00154         end if
00155         return
00156     end
00157
```

3.34 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)

3.34.1 Detailed Description

TCS: Tektronix Plot 10 Emulation.

Version

4.0

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

System independent subroutines

Definition in file [TCS.for](#).

3.34.2 Function/Subroutine Documentation

3.34.2.1 ancho()

```
subroutine ancho (
    ichar )
```

Definition at line [315](#) of file [TCS.for](#).

3.34.2.2 anstr()

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

Definition at line [305](#) of file [TCS.for](#).

3.34.2.3 baksp()

```
subroutine baksp
```

Definition at line [360](#) of file [TCS.for](#).

3.34.2.4 `cartn()`

```
subroutine cartn
```

Definition at line 341 of file [TCS.for](#).

3.34.2.5 `dasha()`

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

Definition at line 266 of file [TCS.for](#).

3.34.2.6 `dashr()`

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

Definition at line 212 of file [TCS.for](#).

3.34.2.7 `drawa()`

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

Definition at line 233 of file [TCS.for](#).

3.34.2.8 `drawr()`

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

Definition at line 188 of file [TCS.for](#).

3.34.2.9 dwindo()

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

Definition at line 438 of file [TCS.for](#).

3.34.2.10 genflg()

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

Definition at line 534 of file [TCS.for](#).

3.34.2.11 home()

```
subroutine home
```

Definition at line 494 of file [TCS.for](#).

3.34.2.12 linef()

```
subroutine linef
```

Definition at line 350 of file [TCS.for](#).

3.34.2.13 linhgt()

```
function linhgt (  
    Numlin )
```

Definition at line 376 of file [TCS.for](#).

3.34.2.14 lintrn()

```
subroutine lintrn
```

Definition at line [394](#) of file [TCS.for](#).

3.34.2.15 linwdt()

```
function linwdt (  
    NumChr )
```

Definition at line [384](#) of file [TCS.for](#).

3.34.2.16 logtrn()

```
subroutine logtrn (  
    IMODE )
```

Definition at line [404](#) of file [TCS.for](#).

3.34.2.17 movea()

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

Definition at line [244](#) of file [TCS.for](#).

3.34.2.18 mover()

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

Definition at line [196](#) of file [TCS.for](#).

3.34.2.19 newlin()

```
subroutine newlin
```

Definition at line [333](#) of file [TCS.for](#).

3.34.2.20 newpag()

```
subroutine newpag
```

Definition at line [368](#) of file [TCS.for](#).

3.34.2.21 pointa()

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

Definition at line [255](#) of file [TCS.for](#).

3.34.2.22 pointr()

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

Definition at line [204](#) of file [TCS.for](#).

3.34.2.23 rel2ab()

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

Definition at line [220](#) of file [TCS.for](#).

3.34.2.24 rescal()

```
subroutine rescal
```

Definition at line 457 of file [TCS.for](#).

3.34.2.25 revcot()

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

Definition at line 290 of file [TCS.for](#).

3.34.2.26 rrotat()

```
subroutine rrotat (  
    Grad )
```

Definition at line 477 of file [TCS.for](#).

3.34.2.27 rscale()

```
subroutine rscale (  
    Faktor )
```

Definition at line 486 of file [TCS.for](#).

3.34.2.28 seetrm()

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

Definition at line 512 of file [TCS.for](#).

3.34.2.29 seetrn()

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

Definition at line [523](#) of file [TCS.for](#).

3.34.2.30 setmrg()

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

Definition at line [503](#) of file [TCS.for](#).

3.34.2.31 swindo()

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

Definition at line [426](#) of file [TCS.for](#).

3.34.2.32 twindo()

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

Definition at line [419](#) of file [TCS.for](#).

3.34.2.33 vcursr()

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

Definition at line [178](#) of file [TCS.for](#).

[illegible]


```

00221      include 'Tktrnx.fd'
00222      call seeloc (ix,iy)
00223      call revcot (ix,iy,xabs,yabs)
00224      xabs= (( xrel*trcosf - yrel*trsinf)*trscal)+xabs
00225      yabs= (( xrel*trsinf + yrel*trcosf)*trscal)+yabs
00226      return
00227      end
00228
00229 C
00230 C Virtuelles Zeichnen, absolut
00231 C
00232
00233      subroutine drawa (X,Y)
00234      include 'Tktrnx.fd'
00235      call wincot (x,y,ix,iy)
00236      call swindl (kminsx,kminsy,kmaxsx,kmaxsy)
00237      call drwabs (ix,iy)
00238      call swindl (0,0,1023,780)
00239      return
00240      end
00241
00242
00243
00244      subroutine movea (X,Y)
00245      include 'Tktrnx.fd'
00246      call wincot (x,y,ix,iy)
00247      call swindl (kminsx,kminsy,kmaxsx,kmaxsy)
00248      call movabs (ix,iy)
00249      call swindl (0,0,1023,780)
00250      return
00251      end
00252
00253
00254
00255      subroutine pointa (X,Y)
00256      include 'Tktrnx.fd'
00257      call wincot (x,y,ix,iy)
00258      call swindl (kminsx,kminsy,kmaxsx,kmaxsy)
00259      call pntabs (ix,iy)
00260      call swindl (0,0,1023,780)
00261      return
00262      end
00263
00264
00265
00266      subroutine dasha (X,Y, iL)
00267      include 'Tktrnx.fd'
00268      call wincot (x,y,ix,iy)
00269      call swindl (kminsx,kminsy,kmaxsx,kmaxsy)
00270      call dshabs (ix,iy, iL)
00271      call swindl (0,0,1023,780)
00272      return
00273      end
00274
00275
00276
00277      subroutine wincot (X,Y,IX,IY)
00278      include 'Tktrnx.fd'
00279      dx= x-tminvx
00280      dy= y-tminvy
00281      if ((xlog.lt.255.).and.(x.gt.0.)) dx= alog(x)-xlog
00282      if ((ylog.lt.255.).and.(y.gt.0.)) dy= alog(y)-ylog
00283      ix= ifix(dx*xfac+.5)+kminsx
00284      iy= ifix(dy*yfac+.5)+kminsy
00285      return
00286      end
00287
00288
00289
00290      subroutine revcot (IX,IY,X,Y)
00291      include 'Tktrnx.fd'
00292      dx= float(ix-kminsx) / xfac
00293      dy= float(iy-kminsy) / yfac
00294      x= dx + tminvx
00295      y= dy + tminvy
00296      if (xlog.lt.255.) x= 2.718282**(dx+xlog)
00297      if (ylog.lt.255.) y= 2.718282**(dy+ylog)
00298      return
00299      end
00300
00301 C
00302 C Alphanumerische Ausgabe
00303 C
00304
00305      subroutine anstr (NChar, IStrin)
00306      dimension istrin(1)
00307      do 10 i=1,nchar

```

```

00308      call ancho (istrin(i))
00309 10  continue
00310      return
00311  end
00312
00313
00314
00315      subroutine ancho (ichar)
00316      include 'Tktrnx.fd'
00317
00318      if (ichar.gt.31) goto 10
00319      if (ichar.eq.7) call bell
00320      if (ichar.eq.10) call linef
00321      if (ichar.eq.13) call cartn
00322      return
00323
00324 10  call seeloc (ix,k)
00325      call csize (ixlen,k)
00326      if (ix.gt.krmrgn-ixlen) call newlin
00327      call toutpt (ichar)
00328      return
00329  end
00330
00331
00332
00333      subroutine newlin
00334      call cartn
00335      call linef
00336      return
00337  end
00338
00339
00340
00341      subroutine cartn
00342      include 'Tktrnx.fd'
00343      call seeloc (ix,iy)
00344      call movabs (klmrgn,iy)
00345      return
00346  end
00347
00348
00349
00350      subroutine linef
00351      call seeloc (j,iy)
00352      call csize (j,iylen)
00353      if (iy.lt.iylen) call home
00354      call movrel (0,-iylen)
00355      return
00356  end
00357
00358
00359
00360      subroutine baksp
00361      call csize (ix,iy)
00362      call movrel (-ix,0)
00363      return
00364  end
00365
00366
00367
00368      subroutine newpag
00369      call erase
00370      call home
00371      return
00372  end
00373
00374
00375
00376      function linhgt (Numlin)
00377      call csize (ix,iy)
00378      linhgt= numlin*iy
00379      return
00380  end
00381
00382
00383
00384      function linwdt (NumChr)
00385      call csize (ix,iy)
00386      linwdt= numchr*ix
00387      return
00388  end
00389
00390 C
00391 C Initialisierungsroutinen
00392 C
00393
00394      subroutine lintrn

```

```

00395     include 'Tktrnx.fd'
00396     xlog= 255.
00397     ylog= 255.
00398     call rescal
00399     return
00400 end
00401
00402
00403
00404     subroutine logtrn (IMODE)
00405     include 'Tktrnx.fd'
00406     call lintrn
00407     if ((imode .eq. 1) .or. (imode .eq. 3)) then
00408         xlog= 0.
00409     end if
00410     if ((imode .eq. 2) .or. (imode .eq. 3)) then
00411         ylog= 0.
00412     end if
00413     call rescal
00414     return
00415 end
00416
00417
00418
00419     subroutine twindo (IX1,IX2,IY1,IY2)
00420     call swindo (ix1,ix2-ix1,iy1,iy2-iy1)
00421     return
00422 end
00423
00424
00425
00426     subroutine swindo (IX,LX,IY,LY)
00427     include 'Tktrnx.fd'
00428     kminsx= ix
00429     kmaxsx= ix+lx
00430     kminsy= iy
00431     kmaxsy= iy+ly
00432     call rescal
00433     return
00434 end
00435
00436
00437
00438     subroutine dwindo (X1,X2,Y1,Y2)
00439     call vwindo (x1,x2-x1,y1,y2-y1)
00440     return
00441 end
00442
00443
00444
00445     subroutine vwindo (X,XL,Y,YL)
00446     include 'Tktrnx.fd'
00447     tminvx= x
00448     tmaxvx= x+xl
00449     tminvy= y
00450     tmaxvy= y+yl
00451     call rescal
00452     return
00453 end
00454
00455
00456
00457     subroutine rescal
00458     include 'Tktrnx.fd'
00459     xfac= 0.
00460     yfac= 0.
00461     if ((tmaxvx.eq.tminvx) .or. (tmaxvy.eq.tminvy)) return
00462     dx= tmaxvx-tminvx
00463     dy= tmaxvy-tminvy
00464     if ((xlog.eq.255.) .or. (amin1(tminvx,tmaxvx).le.0.)) goto 10
00465     xlog= alog(tminvx)
00466     dx= alog(tmaxvx)-xlog
00467 10    if ((ylog.eq.255.) .or. (amin1(tminvy,tmaxvy).le.0.)) goto 20
00468     ylog= alog(tminvy)
00469     dy= alog(tmaxvy)-ylog
00470 20    xfac= float(kmaxsx-kminsx) / dx
00471     yfac= float(kmaxsy-kminsy) / dy
00472     return
00473 end
00474
00475
00476
00477     subroutine rrotat (Grad)
00478     include 'Tktrnx.fd'
00479     trsinf= sin(grad/57.29578)
00480     trcosf= cos(grad/57.29578)
00481     return

```

```

00482     end
00483
00484
00485
00486     subroutine rscale (Faktor)
00487     include 'Tktrnx.fd'
00488     trscal= faktor
00489     return
00490     end
00491
00492
00493
00494     subroutine home
00495     include 'Tktrnx.fd'
00496 C     call movabs(klmggn,750) Fuer CP/M (kein khomey verfuegbar, -> !=750)
00497     call movabs(klmggn,khomey)
00498     return
00499     end
00500
00501
00502
00503     subroutine setmrg (Mlinks, Mrecht)
00504     include 'Tktrnx.fd'
00505     klmggn= mlinks
00506     krmrgn= mrecht
00507     return
00508     end
00509
00510
00511
00512     subroutine seetrm (IBaud, Iterm, ICSIZE,MaxScr)
00513     include 'Tktrnx.fd'
00514     ibaud= 0
00515     iterm= 1
00516     icsize= 1
00517     maxscr= 1023
00518     return
00519     end
00520
00521
00522
00523     subroutine seetrm (xf,yf,key)
00524     include 'Tktrnx.fd'
00525     xf= xfac
00526     yf= yfac
00527     key= 1
00528     if ((xlog.lt.255.).or.(ylog.lt.255.)) key=2
00529     return
00530     end
00531
00532
00533
00534     logical function genflg (ITEM)
00535     genflg= item.eq.0
00536     return
00537     end
00538

```

3.36 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.

3.36.1 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](#).

3.36.2 Function Documentation

3.36.2.1 bell()

```
void bell ( )
```

Signalton.

3.36.2.2 CloseBytFil()

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

Schliesen eines Bytefiles.

Parameters

in	<i>iHandle</i>	Filehandle
----	----------------	------------

3.36.2.3 GetEnv()

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

Abfrage Enviromentvariable

Parameters

in, out	<i>Buf</i>	in=Variable out=Uebersetzung
in	<i>BufLen</i>	

3.36.2.4 GinCrs()

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

Abfrage Graphikmaus.

Parameters

out	<i>ic</i>	Gedruckte Taste
out	<i>ix,iy</i>	Cursorposition

3.36.2.5 GinCrsEx()

```
void GinCrsEx ( )
```

Reset Graphikmaus.

3.36.2.6 GinCrsIn()

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

Initialisierung Graphikmaus.

Parameters

out	<i>iAvail</i>	Maus vorhanden
out	<i>iButton</i>	Anzahl Tasten
in	<i>iXmin,iXmax,iYmin,iYmax</i>	Zeichenfläche

3.36.2.7 kinput()

```
int kinput ( )
```

Tastaturabfrage.

Parameters

out	 <i>AX</i>	Funktionsrückgabe ASCII
-----	--	----------------------------

3.36.2.8 lib_movc3()

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

Kopieren eines Feldes

Parameters

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

3.36.2.9 OpenBytFil()

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

Oeffnen eines Bytefiles.

Parameters

out	<i>iErr</i>	Errorflag
out	<i>iHandle</i>	Filehandle
in	<i>FilNam</i>	Dateiname

3.36.2.10 WrtBytFil()

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

WrtBytFil Byteweises Schreiben ohne Steuerzeichen.

Parameters

out	<i>iErr</i>	Errorflag
in	<i>iHandle</i>	Filehandle
in	<i>buf</i>	Daten
in	<i>iWrite</i>	Anzahl zu schreibender Bytes

3.37 TCSdDosa.asm

```
00001 ; // DOXYGEN Dokumentation TCSdDOS.asm: als C-Programm möglich da ";" C-Leerbefehl entspricht
00002 ; /** \file TCSdDosa.asm \brief DOS Port: x86 Assembler Routinen \version 1.4
00003 ; \author (C) 2022 Dr.-Ing. Klaus Friedewald
00004 ; \copyright GNU LESSER GENERAL PUBLIC LICENSE Version 3 */
00005
00006 ; //! \brief Tastaturabfrage \param[out] [AX] Funktionsrückgabe ASCII
00007 ; (int) ktinput ()
00008
00009 ; //! \brief Signalton
00010 ; (void) bell ()
00011
00012 ; //! \brief Initialisierung Graphikmaus
00013 ; //! \param[out] iAvail Maus vorhanden
00014 ; //! \param[out] iButton Anzahl Tasten
00015 ; //! \param[in] iXmin, iXmax, iYmin, iYmax Zeichenfläche
00016 ; (void) GinCrsIn (bool iAvail,int iButton,int iXmin,int iXmax,int iYmin,int iYmax)
00017
00018 ; //! \brief Abfrage Graphikmaus
00019 ; //! \param[out] ic Gedruckte Taste
00020 ; //! \param[out] ix, iy Cursorposition
00021 ; (void) GinCrs (int ic,int ix,int iy)
00022
```



```

00023 ; /// \brief Reset Graphikmaus
00024 ; (void) GinCrsEx ()
00025
00026 ; /// \brief Abfrage Enviromentvariable
00027 ; /// \param[in,out] Buf in=Variable out=Uebersetzung
00028 ; /// \param[in] BufLen
00029
00030 ; (void) GetEnv (char Buf, int BufLen)
00031 ; /// \brief Kopieren eines Feldes
00032 ; /// \param[in] iByte Anzahl verschiebender Bytes (0 zulässig)
00033 ; /// \param[in] Source zu kopierende Daten
00034 ; /// \param[out] Dest Zielfeld, kann auch Source überlappen
00035
00036 ; (void) lib_movc3 (int iByte, char Source, char Dest)
00037 ; /// \brief Oeffnen eines Bytefiles
00038 ; /// \param[out] iErr Errorflag
00039 ; /// \param[out] iHandle Filehandle
00040 ; /// \param[in] FilNam Dateiname
00041
00042 ; (void) OpenBytFil (int iErr, int iHandle, char FilNam)
00043 ; /// \brief WrtBytFil Byteweises Schreiben ohne Steuerzeichen
00044 ; /// \param[out] iErr Errorflag
00045 ; /// \param[in] iHandle Filehandle
00046 ; /// \param[in] buf Daten
00047 ; /// \param[in] iWrite Anzahl zu schreibender Bytes
00048
00049 ; (void) WrtBytFil (int iErr, int iHandle, char buf, int iWrite)
00050 ; /// \brief Schliesen eines Bytefiles
00051 ; /// \param[in] iHandle Filehandle
00052
00053 ; (void) CloseBytFil (int iHandle)
00054 ; /// \cond
00055 ; ----- Changelog -----
00056 ;
00057 ; Version 1.2
00058 ; 25.10.01 Dr. Ing. K. Friedewald
00059 ;
00060 ; ktinput: Tastaturabfrage
00061 ; bell: Signalton
00062 ; GinCrsIn: Initialisierung Graphikmaus
00063 ; GinCrs: Abfrage Graphikmaus
00064 ; GinCrsEx: Wiederherstellen Graphikmaus
00065 ;
00066 ; GetEnv: Abfrage Enviromentvariable (C-Characterformat!)
00067 ; Input: Pufferfeld, Vorbesetzt mit Variablenname
00068 ; max. Länge Pufferfeld (einschliesslich char(0))
00069 ; Output: Pufferfeld, Übersetzter Wert
00070 ;
00071 ; Lib_movC3 Kopieren eines Feldes
00072 ; Input: iByte, Anzahl verschiebender Bytes (0 zulässig)
00073 ; Source, zu kopierende Daten
00074 ; Output: Dest, Zielfeld, kann auch Source überlappen
00075 ;
00076 ; OpenBytFil Oeffnen eines Bytefiles
00077 ; Input: FilNam
00078 ; Output: iErr, iHandle
00079 ;
00080 ; WrtBytFil Byteweises Schreiben ohne Steuerzeichen
00081 ; Input: iHandle, Buf(*), iCount
00082 ; Output: iErr
00083 ;
00084 ; CloseBytFil Schliesen eines Bytefiles
00085 ; Input: iHandle
00086 ;
00087 ;
00088 ;
00089 ; Version 1.31
00090 ; 30.05.02 Dr. Ing. K. Friedewald
00091 ;
00092 ; Anpassung an WATCOM-Assembler:
00093 ; Auskommentieren der Microsoft-spezifischen Assemblerdirektiven
00094 ; .no87, .list, title, subtitle, page
00095 ; Bugfix: Fehlerhafte Parameterübergabe WRTBYTFIL:
00096 ; DS von Buf wurde überschrieben
00097 ; iErr jetzt übergeben (Programm: MOV, Deklaration: Offset)
00098 ;
00099 ;
00100 ; Version 1.32
00101 ; 25.10.02 Dr. Ing. K. Friedewald
00102 ;
00103 ; Bugfix: Schnell aufeinanderfolgende GINCRS-Aufrufe fehlerhaft
00104 ; Warten auf nicht gedruckte Maustaste ergaenzt
00105 ;
00106 ; Version 1.33
00107 ; 29.10.04 Dr. Ing. K. Friedewald
00108 ;
00109 ; Anpassung an OpenWatcom-Linker 1.3: Großschreibung PUBLIC-Symbole

```

```

00110 ;
00111 ; Version 1.4
00112 ; 04.12.20 Dr. Ing. K. Friedewald
00113 ;
00114 ; Dokumentation durch DOXYGEN
00115 ;
00116 ;
00117 ;
00118 ; title 'TCS Assembler Routinen'
00119 ; .8086
00120 ; .no87
00121 ; .list
00122 ; .model large
00123 ;
00124 public KTINPUT ; FORTRAN: integer*2 function ktinput ()
00125 ;
00126 public BELL ; FORTRAN: call bell ()
00127 ;
00128 public GINCRS ; FORTRAN: call gincrs (ic,ix,iy)
00129 ic equ [BP] + 14 ; Integer*2 (Rückgabe 1,2: linke,rechte Maustaste sonst ASCII
00130 ix equ [BP] + 10 ; Integer*2
00131 iy equ [BP] + 6 ; Integer*2
00132 ;
00133 public GINCRSIN ; FORTRAN: call gincrsIn (iAvail, iButton, ix0,ix1,iY0,iY1)
00134 iAvail equ [BP] + 26 ; Integer*2 oder Logical*2
00135 iButton equ [BP] + 22 ; Integer*2
00136 ix0 equ [BP] + 18 ; Integer*2
00137 ix1 equ [BP] + 14 ; Integer*2
00138 iy0 equ [BP] + 10 ; Integer*2
00139 iy1 equ [BP] + 6 ; Integer*2
00140 ;
00141 public GINCRSEX ; FORTRAN: call GinCrsEx ()
00142 ;
00143 public GETENV ; FORTRAN: call GetEnv (CHARBUF, CharBufL)
00144 CharBuf equ [BP] + 10 ; Vorbesetzt mit "NAME="//char(0)
00145 CharBufL equ [BP] + 6
00146 ;
00147 public OPENBYTFIL ; FORTRAN: call OpenBytFil (iErr, iHandle, Filnam)
00148 iErrO equ [BP] + 14
00149 iHandleO equ [BP] + 10 ; integer*2 iHandle <> 0 falls o.k.
00150 FilNam equ [BP] + 6 ; C-String
00151 ;
00152 public WRTBYTFIL ; FORTRAN: call WrtBytFil (iErr, iHandle, Buf, iCount)
00153 iErr equ [BP] + 18
00154 iHandle equ [BP] + 14 ; Integer*2
00155 Buf equ [BP] + 10 ; byte array
00156 iCount equ [BP] + 6 ; Integer*2
00157 ;
00158 public CLOSEBYTFIL ; FORTRAN: call CloseBytFil (iHandle)
00159 iHandleC equ [BP] + 6
00160 ;
00161 public LIB_MOVC3_ ; FORTRAN: call Lib_MovC3_ (iByte, Source, Dest)
00162 iByte equ [BP] + 14
00163 Source equ [BP] + 10
00164 Dest equ [BP] + 6
00165 ;
00166 TCSdDosA_data segment public 'DATA' ; obligatorischer Name für MS-Compiler
00167 ;
00168 ;
00169 CrsDefHotX equ 0 ; Definition Graphikmousecursor
00170 CrsDefHotY equ 0 ; Vorsicht, Cursor kann nicht über linke, obere Ecke geclickt
00171 ; werden!
00171 CrsDef dw 16 dup (0ffffh) ; Screenmask (wird AND verknüpft)
00172 dw 07c00h, 0c000h ; Cursorform (wird XOR verknüpft)
00173 dw 0a000h, 09000h
00174 dw 08800h, 08400h
00175 dw 00200h, 00100h
00176 dw 00080h, 00000h
00177 dw 00000h, 00000h
00178 dw 00000h, 00000h
00179 dw 00000h, 00000h
00180 ;
00181 TCSdDosA_data ends
00182 ;
00183 DGROUP group TCSdDosA_data
00184 ;
00185 ; subtitle 'TCS Basisfunktionen'
00186 ; page
00187 ;
00188 TcsdDosA_text segment public 'code' ; obligatorischer Name für MS-Compiler
00189 ;
00190 assume CS:TcsdDosA_text, DS:DGROUP, SS:DGROUP
00191 ;
00192 DOS equ 021h ; DOS-Interrupt
00193 MOUSE equ 033h ; Mousedriver
00194 VideoBIOS equ 010h
00195 ;

```

```

00196 ;
00197 ; *****
00198 ; *
00199 ; * Function KTINPUT *
00200 ; *
00201 ; *****
00202 ;
00203
00204 ktinput    proc far
00205
00206             push    bp
00207             mov     bp,sp                ; lokale Basis
00208             push    ds
00209
00210             mov     ah, 07h              ; DOS 7: Zeichen ohne Echo einlesen
00211             int     DOS
00212             mov     ah,0h
00213
00214             pop     ds
00215             pop     bp
00216             ret
00217
00218 ktinput    endp
00219 ;
00220 ; *****
00221 ; *
00222 ; * Subroutine BELL *
00223 ; *
00224 ; *****
00225 ;
00226 bell      proc far
00227
00228             push    bp
00229             mov     bp,sp                ; lokale Basis
00230             push    ds
00231
00232             mov     ah, 0eh              ; Video-Bios: TTY Out
00233             mov     al, 07h              ; Bell
00234             mov     bh,0                 ; Bildschirmnummer
00235             mov     bl,0                 ; Grafik-Vordergrundfarbe
00236             int     VideoBIOS
00237
00238             pop     ds
00239             pop     bp
00240             ret
00241
00242 bell      endp
00243
00244 ;             subtitle    'Graphic Input Cursor'
00245 ;             page
00246 ;
00247 ; *****
00248 ; *
00249 ; * Subroutine GINCRSIN *
00250 ; *
00251 ; *****
00252 ;
00253 ginCrsIn   proc far
00254
00255             push    bp
00256             mov     bp,sp                ; lokale Basis
00257             push    ds
00258             push    es
00259
00260             mov     ax, 00h              ; FN : Reset Mouse
00261             int     MOUSE
00262             push    bx                    ; Freimachen Indexregister
00263             lds     bx, iAvail            ; Adresse iAvail nach BX laden
00264             mov     [bx],ax              ; Wert AX nach iAvail
00265             lds     bx, iButton           ; Adresse iButton nach BX laden
00266             pop     ax
00267             mov     [bx],ax              ; Wert AX nach iButton
00268
00269             mov     ax, 07h              ; FN : Setzen iXmin und iXmax
00270             lds     bx, iX0
00271             mov     cx,[bx]
00272             lds     bx, iX1
00273             mov     dx,[bx]
00274             int     MOUSE
00275
00276             mov     ax, 08h              ; FN : Setzen iYmin und iYmax
00277             lds     bx, iY0
00278             mov     cx,[bx]
00279             lds     bx, iY1
00280             mov     dx,[bx]
00281             int     MOUSE
00282

```

```

00283      mov     ax, 09h           ; FN : Definition Cursorform
00284      mov     bx, CrsDefHotX
00285      mov     cx, CrsDefHotY
00286      mov     dx, seg CrsDef     ; Mousedriver: Adressangabe über ES!
00287      mov     es, dx
00288      mov     dx, offset CrsDef
00289      int     MOUSE
00290
00291      pop     es
00292      pop     ds
00293      pop     bp
00294      ret     24                 ; Parameteranzahl * 4 Bytes freigeben
00295 gincrsIn  endp
00296 ;
00297 ; *****
00298 ; *
00299 ; * Subroutine GINCRSEX *
00300 ; *
00301 ; *****
00302 ;
00303 ginCrsEx  proc far
00304
00305      push    bp
00306      mov     bp,sp              ; lokale Basis
00307      push    ds
00308
00309      mov     ax, 00h           ; FN : Reset Mouse
00310      int     MOUSE
00311
00312      pop     ds
00313      pop     bp
00314      ret     0                 ; Parameteranzahl * 4 Bytes freigeben
00315 gincrsEx  endp
00316 ;
00317 ; *****
00318 ; *
00319 ; * Subroutine GINCRS *
00320 ; *
00321 ; *****
00322 ;
00323 gincrs    proc far
00324
00325      push    bp
00326      mov     bp,sp              ; lokale Basis
00327      push    ds
00328
00329      mov     ax, 01h           ; FN : Show Cursor
00330      int     MOUSE
00331
00332 WaitUp:   mov     ax, 03h       ; FN: Get Button Status
00333      int     MOUSE
00334      test    bx,bx              ; Taste noch gedrueckt?
00335      jnz     WaitUp             ; noch vom letzten mal -> Warte
00336
00337 KeyLoop:  mov     ax, 03h       ; FN : Get Button Status
00338      int     MOUSE              ; MouseDriver-Call
00339      test    bx,bx              ; Bit0 linke, Bit 1 rechte Maustaste
00340      jnz     ExitKeyLp          ; Taste gedrückt -> fertig
00341
00342      mov     ah,06h             ; DOS 6: Zeichen ohne Warten einlesen
00343      mov     dl,0ffh
00344      int     DOS
00345      jz      KeyLoop            ; keine Keyboardtaste gedrückt -> weiter
00346
00347      mov     ah,0h
00348      push    ax                 ; Terminator
00349      mov     ax, 03h           ; FN : Get Mouse Koordinaten
00350      int     MOUSE
00351      pop     bx                 ; Terminator ASCII
00352
00353 ExitKeyLp: push    bx           ; Terminator
00354      lds     bx, iX              ; Adresse iX nach BX laden
00355      mov     [bx],cx             ; CX: horizontale Mauskoordinate
00356      lds     bx, iY              ; Adresse iY nach BX laden
00357      mov     [bx],dx             ; DX: vertikale Mauskoordinate
00358      pop     ax                 ; Terminator
00359      lds     bx, iC              ; Adresse iC nach BX laden
00360      mov     [bx],ax             ; Übergabe in iC
00361
00362
00363      mov     ax, 02h           ; FN : Hide Cursor
00364      int     MOUSE
00365
00366      pop     ds
00367      pop     bp
00368      ret     12                 ; Parameteranzahl * 4 Bytes freigeben
00369 gincrs    endp

```

```

00370
00371 ;           subtitle   'Get Enviroment'
00372 ;           page
00373 ;
00374 ; *****
00375 ; *
00376 ; * Subroutine GETENV *
00377 ; *
00378 ; *****
00379 ;
00380 GetEnv      proc far
00381
00382             push  bp
00383             mov   bp,sp           ; lokale Basis
00384             push  ds
00385             push  es
00386             push  di
00387             push  si
00388             pushf                ; Rette Direction Flag!
00389
00390             cld                  ; Stringsuche aufwärts
00391 ;
00392 ; Bestimmung Stringlänge Suchstring
00393 ;
00394             mov   cx, 0           ; Counter
00395             lds   si, CharBuf     ; Buffer = Suchstring
00396 LenLoop:    mov   al,byte ptr ds:[si]; nächstes Zeichen
00397             or    al,al           ; Char(0) = Ende?
00398             jz    LenDone         ; ja
00399             inc   cx
00400             inc   si
00401             jmp   LenLoop
00402
00403 LenDone:    push  cx             ; Länge des Suchstrings
00404 ;
00405 ; Get Enviroment
00406 ;
00407             mov   ah, 62h         ; DOS 62h: Get PSP
00408             int   DOS
00409             mov   es,bx           ; ES:00 jetzt auf PSP
00410             mov   bx,es:[2ch]     ; PSP Element 2c: Enviroment
00411             mov   es, bx
00412             xor   di,di           ; Jetzt: ES:DI auf 1. Eintrag Enviroment
00413
00414 SearchLoop: lds   si, CharBuf     ; Suchstring in DS:AX
00415             pop   cx             ; Länge Suchstring
00416             push  cx
00417             repe cmpsb            ; vergleichen mit Enviroment
00418             jz    Found          ;
00419             xor   al,al           ; Ende Enviromenteintrag suchen
00420             mov   cx,-1
00421             repnz scasb
00422             cmp   byte ptr es:[di],0; letzter Eintrag?
00423             jnz   SearchLoop
00424             jmp   NotFound
00425 ;
00426 ; Abspeichern in den Puffer
00427 ;
00428 NotFound:   ; ES:DI auf Char(0)
00429 Found:      ; ES:DI auf Inhalt Enviromentvariable
00430
00431             lds   bx, CharBufL    ; Parameter Bufferlänge
00432             mov   cx,[bx]         ; Counter = Bufferlänge
00433
00434             lds   si, CharBuf     ; Zieladresse
00435 StoreLoop:  mov   al,byte ptr es:[di]; nächstes Zeichen
00436             mov   byte ptr ds:[si],al; speichern
00437             or    al,al           ; Char(0) = Ende?
00438             jz    StoreDone       ; ja
00439             inc   di
00440             inc   si
00441             dec   cx
00442             jz    StoreDone       ; Bufferende erreicht
00443             jmp   StoreLoop
00444
00445 StoreDone:  pop   ax             ; Clear Stack, Suchstringlänge
00446
00447             popf                ; Restore Status
00448             pop   si
00449             pop   di
00450             pop   es
00451             pop   ds
00452             pop   bp
00453             ret   8
00454
00455 GetEnv      endp
00456

```

```

00457 ;           subtitle   'Byte Files'
00458 ;           page
00459 ;
00460 ; *****
00461 ; *
00462 ; * Function OpenBytFil *
00463 ; *
00464 ; *****
00465 ;
00466 OpenBytFil proc far
00467
00468         push  bp
00469         mov   bp,sp           ; lokale Basis
00470         push  ds
00471
00472         lds   dx,FilNam
00473         xor   cx,cx           ; Löschen Attribut -> unbeschränkter Zugriff
00474         mov   ah,05bh         ; Open New File
00475         int   DOS
00476
00477         lds   bx, iHandle0     ; Adresse iButton nach BX laden
00478         mov   [bx],ax          ; FileHandle nach iHandle
00479
00480         lds   bx, iErr0
00481         jc    ErrO             ; kein Carryflag -> iErr=0: i.O.
00482         xor   ax,ax            ; iErr=3: path not found, =4 too many open files
00483 ErrO:    mov   [bx],ax         ; =5 access denied, =50h file exists
00484
00485         pop   ds
00486         pop   bp
00487         ret    12              ; 12 = 3 Parameter
00488
00489 OpenBytFil endp
00490 ;
00491 ;
00492 ; *****
00493 ; *
00494 ; * Function WrtBytFil *
00495 ; *
00496 ; *****
00497 ;
00498
00499 WrtBytFil  proc far
00500
00501         push  bp
00502         mov   bp,sp           ; lokale Basis
00503         push  ds
00504
00505         lds   bx,iCount
00506         mov   cx,[bx]
00507         jcxz  NoWrt           ; keine Bytes zu schreiben
00508
00509         lds   bx,iHandle
00510         mov   bx,[bx]
00511
00512         lds   dx,Buf          ; letzter Befehl vor DOS-call, DS auf Buf!
00513
00514         mov   ah,040h         ; Write File
00515         int   DOS
00516
00517         lds   bx,iCount
00518         mov   cx,[bx]
00519         xor   dx,dx            ; Clear Error-Flag
00520         cmp   ax,cx            ; Count IST < Count SOLL?
00521         jnl   WrtIO
00522         mov   dx,0ffffh        ; SET Error-Flag
00523 WrtIO:    lds   bx,iErr        ; Store Error-Flag
00524         mov   [bx],dx
00525
00526 NoWrt:    pop   ds
00527         pop   bp
00528         ret    16              ; 16 = 4 Parameter
00529
00530 WrtBytFil  endp
00531 ;
00532 ; *****
00533 ; *
00534 ; * Function CloseBytFil *
00535 ; *
00536 ; *****
00537 ;
00538 CloseBytFil proc far
00539
00540         push  bp
00541         mov   bp,sp           ; lokale Basis
00542         push  ds
00543

```

```

00544         lds    bx,iHandleC
00545         mov     bx,[bx]
00546         mov     ah,03eh           ; Close File
00547         int     DOS
00548
00549         pop     ds
00550         pop     bp
00551         ret     4                 ; 4 = 1 Parameter
00552
00553 CloseBytFil endp
00554
00555 ;             subtitle    'lib$MoveC3'
00556 ;             page
00557 ;
00558 ; *****
00559 ; *
00560 ; * Subroutine lib_MovC3 *
00561 ; *
00562 ; *****
00563 ;
00564 lib_movc3_  proc far
00565
00566         push    bp
00567         mov     bp,sp           ; lokale Basis
00568         push    ds
00569         push    es
00570         push    di
00571         push    si
00572         pushf                 ; Rette Direction Flag!
00573
00574 ;
00575 ; Kopieren des Strings
00576 ;
00577
00578         lds     bx,iByte
00579         mov     cx,[bx]         ; Counter
00580         lds     si, Source      ; Buffer = Suchstring
00581         les     di, Dest
00582
00583         cld                     ; aufwärts
00584         cmp     di,si
00585         jnb     domove
00586
00587         add     di,cx
00588         dec     di
00589         add     si,cx
00590         dec     si
00591         std                     ; abwärts
00592
00593 domove:   rep     movsb
00594
00595         popf                 ; Restore Status
00596         pop     si
00597         pop     di
00598         pop     es
00599         pop     ds
00600         pop     bp
00601         ret     12
00602
00603 lib_movc3_ endp
00604
00605 TcsdDosA_text ends
00606
00607         end
00608 ;
00609 ; ///! \endcond
00610

```

3.38 TCSdDosa.fi File Reference

DOS Port: FORTRAN-Interface TCSdDOSa.asm.

3.38.1 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

Assembler routines 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 \$nottruncate und \$notstrict are superfluous.

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

Definition in file [TCSdDosa.fi](#).

3.39 TCSdDosa.fi

```

00001 C> \file      TCSdDosa.fi
00002 C> \brief    DOS Port: FORTRAN-Interface TCSdDOSa.asm
00003 C>
00004 C> \~german
00005 C> Interfacedeclarationen fuer den Watcom Fortran-Compiler
00006 C> \~english
00007 C> Interface definitions for the Watcom Fortran Compiler
00008 C> \~
00009 C> \author   Dr.-Ing. Klaus Friedewald
00010 C> \version 1.32
00011 C> \date    06.02.2003
00012 C> \~german
00013 C> \note
00014 C> Assembler routines entsprechend Microsoft Procedure Call Standard
00015 C>
00016 C> \note
00017 C> Watcom Compiler erlaubt 32 Zeichen lange Variablenamen unter Verwendung
00018 C> von $ und _. Deswegen $nottruncate und $notstrict ueberfluessig.
00019 C>
00020 C> \note
00021 C> Hex-Zahlen werden nicht durch \#ff sondern durch \'ff\'x dargestellt
00022 C> \~english
00023 C> \note
00024 C> Assembler routines are written according to the Microsoft Procedure Call Standard.
00025 C>
00026 C> \note
00027 C> Watcom-FTN77 variable names are allowed to be 32 characters long and may
00028 C> contain $ and _. That for $nottruncate und $notstrict are superfluous.
00029 C>
00030 C> \note
00031 C> Hexadecimal numbers are represented by \'ff\'x instead of \#ff.
00032 C> \~
00033 C>
00034 C
00035 C Interfacedeclarationen fuer den Watcom Fortran-Compiler
00036 C Assembler routines entsprechend Microsoft Procedure Call Standard
00037 C
00038 C
00039 C kinput:    Tastaturabfrage [AX] dos7h
00040 C bell:      Signalton [ax,bx] video bios tty out
00041 C GinCrsIn:  Initialisierung Graphikmaus [ax,bx,cx,dx] int mouse
00042 C GinCrsEX:  Wiederherstellen Graphikmaus [ax] int mouse
00043 C GinCrs:    Abfrage Graphikmaus [ax,bx,cx,dx] int mouse
00044 C
00045 C GetEnv:     Abfrage Enviroment (C-Characterformat!) [ax,bx,cx,dx] int dos
00046 C
00047 C Lib_movC3_: Kopieren eines Feldes [ax,bx,cx]

```



```

00048 C
00049 C   OpenBytFil [ax,bx,cd,dx] dos
00050 C   WrtBytFil [ax,bx,cd,dx] dos
00051 C   CloseBytFil [ax,bx]
00052 C   i.O.: kTinput, bell
00053 C
00054 C \cond
00055
00056 c$pragma aux kTinput value [ax] modify exact [ax]
00057
00058 c$pragma aux bell parm [] modify exact [ax bx]
00059
00060 c$pragma aux GetEnv parm reverse (DATA_REFERENCE FAR, REFERENCE FAR) [] \
00061 c   modify exact [ax bx cx dx]
00062
00063 c$pragma aux GinCrsIn parm reverse (REFERENCE FAR, reference far, \
00064 c   reference far) [] modify exact [ax bx cx dx]
00065
00066 c$pragma aux GinCrs parm reverse (REFERENCE FAR) [] \
00067 c   modify exact [ax bx cx dx]
00068
00069 c$pragma aux GinCrsEx modify exact [ax]
00070
00071 c$pragma aux lib_movC3_ parm reverse (REFERENCE FAR, DATA_REFERENCE FAR, \
00072 c   DATA_REFERENCE FAR) [] modify exact [ax bx cx]
00073
00074 c$pragma aux OpenBytFil parm reverse (REFERENCE FAR, REFERENCE FAR, \
00075 c   DATA_REFERENCE FAR) [] modify exact [ax bx cx dx]
00076
00077 c$pragma aux WrtBytFil parm reverse (REFERENCE FAR, REFERENCE FAR, \
00078 c   DATA_REFERENCE FAR, REFERENCE FAR) [] modify exact [ax bx cx dx]
00079
00080 c$pragma aux CloseBytFil parm reverse (REFERENCE FAR) [] modify exact [ax bx]
00081 C
00082 C \endcond

```

3.40 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](#) (lxlen, 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](#)

3.40.1 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](#).

3.40.2 Function/Subroutine Documentation

3.40.2.1 alpha()

```
subroutine alpha
```

Definition at line [686](#) of file [TCSdrDOS.for](#).

3.40.2.2 anmode()

```
subroutine anmode
```

Definition at line [801](#) of file [TCSdrDOS.for](#).

3.40.2.3 bckcol()

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

Definition at line [427](#) of file [TCSdrDOS.for](#).

3.40.2.4 csize()

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

Definition at line [698](#) of file [TCSdrDOS.for](#).

3.40.2.5 dcursr()

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

Definition at line [767](#) of file [TCSdrDOS.for](#).

3.40.2.6 defaultcolour()

```
subroutine defaultcolour
```

Definition at line [436](#) of file [TCSdrDOS.for](#).

3.40.2.7 drwabs()

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

Definition at line [587](#) of file [TCSdrDOS.for](#).

3.40.2.8 drwrel()

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

Definition at line [645](#) of file [TCSdrDOS.for](#).

3.40.2.9 dshabs()

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

Definition at line [599](#) of file [TCSdrDOS.for](#).

3.40.2.10 dshrel()

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

Definition at line [655](#) of file [TCSdrDOS.for](#).

3.40.2.11 erase()

```
subroutine erase
```

Definition at line [500](#) of file [TCSdrDOS.for](#).

3.40.2.12 finitt()

```
subroutine finitt
```

Definition at line [513](#) of file [TCSdrDOS.for](#).

3.40.2.13 graphicerrorinit()

```
subroutine graphicerrorinit
```

Definition at line [254](#) of file [TCSdrDOS.for](#).

3.40.2.14 icolcode()

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

Definition at line [444](#) of file [TCSdrDOS.for](#).

3.40.2.15 initt()

```
subroutine initt (  
    iDummy )
```

Definition at line [121](#) of file [TCSdrDOS.for](#).

3.40.2.16 initt1()

```
subroutine initt1
```

Definition at line [135](#) of file [TCSdrDOS.for](#).

3.40.2.17 irevscreenxcoord()

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

Definition at line 484 of file [TCSdrDOS.for](#).

3.40.2.18 irevscreenycoord()

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

Definition at line 492 of file [TCSdrDOS.for](#).

3.40.2.19 iscreenxcoord()

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

Definition at line 468 of file [TCSdrDOS.for](#).

3.40.2.20 iscreenycoord()

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

Definition at line 476 of file [TCSdrDOS.for](#).

3.40.2.21 italic()

```
subroutine italic
```

Definition at line 219 of file [TCSdrDOS.for](#).

3.40.2.22 lib_movc3()

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

Definition at line 790 of file [TCSdrDOS.for](#).

3.40.2.23 lincol()

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

Definition at line 406 of file [TCSdrDOS.for](#).

3.40.2.24 movabs()

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

Definition at line 557 of file [TCSdrDOS.for](#).

3.40.2.25 movrel()

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

Definition at line 625 of file [TCSdrDOS.for](#).

3.40.2.26 pntabs()

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

Definition at line 570 of file [TCSdrDOS.for](#).

3.40.2.27 pntrel()

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

Definition at line 635 of file [TCSdrDOS.for](#).

3.40.2.28 restat()

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

Definition at line 541 of file [TCSdrDOS.for](#).

3.40.2.29 seeloc()

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

Definition at line 667 of file [TCSdrDOS.for](#).

3.40.2.30 statst()

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

Definition at line 744 of file [TCSdrDOS.for](#).

3.40.2.31 svstat()

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

Definition at line 529 of file [TCSdrDOS.for](#).

3.40.2.32 swind1()

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

Definition at line 676 of file [TCSdrDOS.for](#).

3.40.2.33 tcslev()

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

Definition at line 104 of file [TCSdrDOS.for](#).

3.40.2.34 tinput()

```
subroutine tinput (  
    iChr )
```

Definition at line 760 of file [TCSdrDOS.for](#).

3.40.2.35 toutpt()

```
subroutine toutpt (  
    iChr )
```

Definition at line 707 of file [TCSdrDOS.for](#).

3.40.2.36 toutst()

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

Definition at line 725 of file [TCSdrDOS.for](#).


```

00056 C      Kombination groß/kursiv nicht vorgesehen.
00057 C      Implementiert durch Fontfile GraphLib.FON
00058 C      (Quelle: Programm SOFTY und Arial Terminal TTF-Basis)
00059 C      DSHABS:      Standardisierung Dash-Linestyles DOS-Windows:
00060 C      0: solid, 1: dotted, 2: dash-dotted, 3:dashed
00061 C      DEFAULTCOLOUR: Bugfix Namensgebung, nicht DEFAULTCOLOURS
00062 C
00063 C      10.10.02 Version 1.33:
00064 C      INITT:      Zur Vereinheitlichung DOS/Windows jetzt in diesem File
00065 C      TCSLEV:      neu, zur Angleichung mit AG2LEV und Systemerkennung
00066 C
00067 C      19.10.02 Version 1.34 bzw. (2002,292,2)
00068 C      Umbenennung TKTRNX.FOR in TKTRNX.FD zur Kompatibilität CP/M
00069 C
00070 C      25.10.02 Version (2002,298,2)
00071 C      Entprellen Mousetaste bei GIN-Cursoreingabe
00072 C
00073 C      06.02.03 Version (2003, 37,2)
00074 C      Vereinheitlichtes Interface lib$movc3 (Kompatibilitaet Windows)
00075 C
00076 C      12.01.04 Version (2004, 12,2)
00077 C      INITT1:      Bugfix Endlosschleife bei fehlerhaftes Fontfile und
00078 C      Severity 5
00079 C      GRAPHICERRORINIT: Defaultseverity 10 bei EXIT (FINITT, iErr=12)
00080 C      Anmerkung: Die Subroutine GRAPHICERROR ruft sich bei Programm-
00081 C      abbruch über FINITT implizit selber rekursiv auf (nicht
00082 C      FORTRAN-konform!). Da jedoch keine lokalen Variablen ver-
00083 C      wendet werden, ist dies in der Regeln nicht kritisch.
00084 C
00085 C      25.10.04 Version (2004,299,2)
00086 C      WINLBL:      Wertet jetzt den 3. Parameter (Initilisierungsfile)
00087 C      analog zur Windowsversion aus (einschliesslich Ueber-
00088 C      setzung '%:' und '%')
00089 C      LIB$MOVVC3: Umbenannt in LIB_MOVC3. Alte Assembleroutine heisst
00090 C      jetzt LIB_MOVC3_.
00091 C
00092 C      15.02.05 Version (2005, 45,2)
00093 C      GRAPHICERROR: Bugfix ErrSeverity=0 entspricht jetzt NO ACTION.
00094 C
00095 C
00096 C      include 'FGRAPH.FI'
00097 C      include 'TCSdDOSa.FI'
00098 C
00099 C
00100 C
00101 C
00102 C      Ausgabe der Softwareversion
00103 C
00104 C      subroutine tcslev(LEVEL)
00105 C      integer LEVEL(3)
00106 C      level(1)=2005      ! Aenderungsjahr
00107 C      level(2)= 45      ! Aenderungstag
00108 C      level(3)= 2      ! System= DOS
00109 C
00110 C      return
00111 C      end
00112 C
00113 C
00114 C
00115 C
00116 C      Bildschirm Verwaltung
00117 C
00118 C
00119 C
00120 C
00121 C      subroutine initt (iDummy)
00122 C      call lintrn
00123 C      call swindo (0,1023,0,780)
00124 C      call vwindo (0.,1023.,0.,780.)
00125 C      call rrotat (0.)
00126 C      call rscale (1.)
00127 C      call setmrg (0,1023)
00128 C      call initt1
00129 C      call home
00130 C      return
00131 C      end
00132 C
00133 C
00134 C
00135 C      subroutine initt1
00136 C      include 'FGRAPH.FD'
00137 C      include 'TKTRNX.FD'
00138 C      integer*2 iErr, iAvail, iButton, kScrX2, kScrY2
00139 C      integer iLen, iTrimLen, iParse
00140 C
00141 C      character*80 cBuf, cBuf1*80
00142 C      record /videoconfig/ myscreen

```

```

00143      record /fontinfo/ myfont
00144
00145      character *13 cFontFile          ! Graphikfontfile
00146      parameter(cfontfile='GRAPHLIB.FON'//char(0))
00147
00148      character*5 cEnv                 ! Logischer Name für den Fontfilepfad
00149      parameter(cenv='LIB='//char(0))
00150
00151      call graphicerrorinit
00152
00153      ierr= setvideomode($maxresmode)
00154
00155      if (ierr .eq. 0) then
00156        call graphicerror (2,' ') ! TCS-Initt: unknown graphic adapter
00157      end if
00158
00159      call getvideoconfig (myscreen)
00160      kscrx= myscreen.numxpixels-1
00161      kscry= myscreen.numypixels-1-
00162      1 (myscreen.numypixels/myscreen.numtextrows)      ! Höhe Statuszeile
00163
00164      call setviewport (0,0, kscrx, kscry)
00165
00166      call settetxwindow (myscreen.numtextrows,1,myscreen.numtextrows,
00167      1 myscreen.numtextcols)      ! Statuszeile
00168      kstcol= myscreen.numtextcols - 1 ! Verhindere Scrollen durch -1
00169
00170      if (registerfonts(cfontfile).lt.0) then
00171        cbuf= cenv                      ! Abfrage Enviroment
00172        call getenv (cbuf, len(cbuf))
00173        ilenpath= itrmlen(cbuf)
00174        iparse=1
00175      10 continue ! while
00176        if (iparse.le.ilenpath) then
00177          ilen= index(cbuf(iparse:ilenpath), ';')-1
00178          if (ilen.le.0) ilen=ilenpath-iparse+1
00179        else
00180          ilen= -1
00181        end if
00182        if ((ilen.lt.1).or.(iparse.gt.ilenpath)) then
00183          cbufl= cenv ! Notwendig zur Bildung des Substrings aus PARAMETER
00184          cbufl=cbufl(1:istringlen(cbufl))//': '//cfontfile
00185          call graphicerror (3,cbufl(1:istringlen(cbufl))) !openerror fontfile
00186          goto 15 ! ENDWHILE falls Errorseverity(3) < 10 (STOP)
00187        else
00188          cbufl= cbuf(iparse:iparse+ilen-1)//'\ '//cfontfile ! Chr0 in cFontFile
00189          call substitute (cbufl,cbufl, '\\', '\') ! kein doppelter Backslash!
00190        end if
00191        if (registerfonts(cbufl(1:istringlen(cbufl))).lt.0) then ! end while
00192          if (ilen.lt.ilenpath) then
00193            iparse= iparse+ilen+1
00194            goto 10 ! nächster Eintrag im Pfad
00195          else
00196            call graphicerror (3,cbufl(1:istringlen(cbufl)))
00197          end if
00198      15 end if
00199      end if
00200
00201      call nrmsiz                      ! Standardschrift: normalgroß, nicht kursiv
00202
00203      kscrx2= kscrx                    ! Konvertierung in int*2 durch WATCOM-Compiler
00204      kscry2= kscry
00205      call gincrsin (iavail, ibutton, 0, kscrx2, 0, kscry2)
00206      if (iavail.eq.-1) then
00207        imouse= ibutton
00208      else
00209        imouse= 0
00210      end if
00211      call defaultcolour
00212      call erase
00213
00214      return
00215      end
00216
00217
00218
00219      subroutine italic
00220 C
00221 C Verändern des Graphik-Fonts
00222 C
00223      include 'FGRAPH.FD'
00224      include 'TKTRNX.FD'
00225      integer*2 iErr
00226      record /fontinfo/ myfont
00227
00228      ierr= setfont('t' 'Italic' "//char(0))
00229      goto 10

```

```

00230
00231     entry dblsiz
00232     ierr= setfont('t''Double'"/char(0))
00233     goto 10
00234
00235     entry italir
00236     entry nrmsiz
00237     ierr= setfont('t''Normal'"/char(0))
00238
00239 10  continue          ! identischer Code für ITALIC und ITALIR
00240     if (ierr.lt.0) then
00241         call graphicerror (4,'Normal/Italic/Double') ! TCS-Initt: unknown font
00242     end if
00243     ierr= getfontinfo(myfont)
00244     khorsz= isign(irevscreenxcoord(int(myfont.pixwidth))
00245 1 - irevscreenxcoord(0),1)
00246     kversz= isign(irevscreenycoord(int(myfont.pixheight))
00247 1 - irevscreenycoord(0),1)
00248     khomey= 780-(1.1*kversz)
00249     return
00250 end
00251
00252
00253
00254 subroutine graphicerrorinit
00255 C
00256 C SUBROUTINE GraphicErrorInit, ENTRIES WinLbl, GraphicError
00257 C Internationalisierung der Fehlermeldungen
00258 C
00259     implicit none
00260     include 'FGRAPH.FD'
00261     save errseverity, errmsg, filnam
00262
00263     integer MaxErr
00264     parameter(maxerr=12)
00265     character *(*) Mssg
00266     character *(*) WinLblDummy, StatLblDummy, MessageFile
00267     integer iErr, i, iTrimLen,iStringLen, iErrSev
00268     integer iLenPath, iParse, iLen
00269
00270     character*132 cEnv, FilNam, cBuf
00271     integer ErrSeverity (MaxErr)
00272     character*80 ErrMsg (MaxErr)
00273     data cenv,filnam /'LIB=','GRAPHLIB.LNG'/
00274     data errmsg/'GRAPHLIB %%% INITT: Incompatible message file - Press
00275 1 any key',
00276 2 'GRAPHLIB %%% INITT: Unknown graphic adapter',
00277 3 'GRAPHLIB %%% INITT: Error opening fontfile $$',
00278 4 'GRAPHLIB %%% INITT: Unknown font $$',
00279 5 'GRAPHLIB %%% INPUT: No mousedriver available, use keyboard'
00280 6 'GRAPHLIB %%% HARDCOPY: Error during OPEN',
00281 7 'GRAPHLIB %%% HARDCOPY: Error during WRITE',
00282 8 'GRAPHLIB %%% HARDCOPY: Internal error (buffer overflow)',
00283 9 '$$', 'Hardcopy in progress', 'Press any key to continue',
00284 2 'Press any key to exit program'/
00285
00286     data errseverity /5,10,10,10, 1, 5, 5, 5, 1, 1, 5, 10/
00287
00288     external iGetArg          ! Watcom Library-Funktion
00289     integer iGetArg
00290
00291     cenv=cenv(1:itrimlen(cenv))/char(0)
00292     filnam= filnam(1:itrimlen(filnam))/char(0)
00293
00294 C
00295 C 1.Priorität: Message-File durch WinLbl spezifiziert
00296 C 2.Priorität: GRAPHLIB.LNG im Arbeitsdirectory
00297 C
00298
00299     open (unit=9,form='FORMATTED', err=5, status='OLD', file=
00300 1          filnam(1:istringlen(filnam)))
00301     goto 7          ! File gefunden -> Einlesen
00302
00303 C
00304 C 3.Priorität: Message-File GRAPHLIB.LNG in LIB:
00305 C
00306
00307 5     call getenv (cenv, len(cenv))
00308     ilenpath= itrimlen(cenv)
00309     iparse=1
00310 10  continue ! while
00311     if (iparse.le.ilenpath) then
00312         ilen= index(cenv(iparse:ilenpath), ';')-1
00313         if (ilen.le.0) ilen=ilenpath-iparse+1
00314     else
00315         goto 99          ! benutze Default
00316     end if

```

```

00317         if ((ilen.ge.1).and.(iparse.le.ilenpath)) then
00318             cbuf= cenv(iparse:iparse+ilen-1)//'\ '//filnam ! Chr0 bereits in FilNam
00319             call substitute (cbuf,cbuf, '\\', '\') ! kein doppelter Backslash !
00320         end if
00321         open (unit=9,form='FORMATTED', err=6, status='OLD', file=
00322             1          cbuf(1:istringlen(cbuf)))
00323         goto 7          ! File gefunden -> Einlesen
00324 6       if (ilen.lt.ilenpath) then ! end while
00325             iparse= iparse+ilen+1
00326             goto 10      ! nächster Eintrag im Pfad
00327         else
00328             goto 99 ! kein File vorhanden - > benutze Default
00329         end if
00330
00331 7       do 20 i=1,maxerr
00332             read (unit=9, err=90, fmt=900) errseverity(i),errmsg(i)
00333 20      continue
00334
00335         close (unit=9)
00336
00337 99      return
00338 C
00339 C Ausgabe Fehlermeldung Messagefile
00340 C
00341 90      call outtext (errmsg(1)) ! Graphiksystem wurde noch nicht initialisiert!
00342         call tinput (i)
00343         return
00344
00345
00346
00347         entry winlbl(winlbldummy, statlbldummy, messagefile)
00348 C
00349 C Setzen des Messagefiles und Uebersetzung '%:' bzw. '%.'
00350 C
00351         if (istringlen(messagefile).le.0) return
00352         filnam= messagefile
00353         i= igetarg(0, cbuf) ! Arg. 0: Programmname mit Directory
00354         if (i.gt.1) then
00355 30      continue ! repeat
00356             i= i-1
00357             if ((cbuf(i:i).ne.'\'').and.(i.gt.1)) goto 30
00358             cbuf(i+1:i+1)= char(0)
00359             call substitute (filnam, filnam,'%:',cbuf)
00360         end if
00361         call substitute (filnam, filnam,'%','%.lng')
00362         return
00363
00364
00365
00366         entry graphicerror(ierr,mssg)
00367 C
00368 C Ausgabe der Fehlermeldung
00369 C
00370         if (ierr.eq.99) then                ! Programmabbruch aus FINITT (2. Aufruf)
00371             if (errseverity(12).eq.10) then
00372                 ierrsev= 99                ! STOP
00373             else if (errseverity(12).eq.5) then
00374                 ierrsev= 1                ! TINPUT bereits durchgeführt
00375             else
00376                 ierrsev= errseverity(12)
00377             end if
00378         else
00379             ierrsev= errseverity(ierr)
00380             if (ierrsev.gt.0) then
00381                 call bell
00382                 call substitute (errmsg(ierr),cbuf, '$$', mssg)
00383                 call statst (cbuf)
00384             end if
00385         end if
00386
00387         if (ierrsev.le.1) then                ! =1: Statusmeldung
00388             return
00389         else if (ierrsev.eq.99) then
00390             stop                            ! =99: aus FINITT
00391         else
00392             call tinput (i)
00393             if (ierrsev.eq.5) then            ! =5: Warnung
00394                 return
00395             else if (ierrsev.eq.10) then
00396                 if (ierr.ne.12) call finitt () ! Rekursion iErr=12 verhindern
00397             end if
00398         end if
00399
00400         return
00401 900     format (1x,i2,1x,a)
00402         end
00403

```

```

00404
00405
00406     subroutine lincol (iCol)
00407     include 'FGRAPH.FD'
00408     include 'TKTRNX.FD'
00409     integer iColCode, iCol
00410     integer *2 iErr
00411     ilincol= icolcode(iCol)
00412     ierr= setcolor(ilincol)
00413     return
00414     end
00415
00416
00417
00418     subroutine txtcol (iCol)
00419     include 'TKTRNX.FD'
00420     integer iColCode, iCol
00421     itxtcol= icolcode(iCol)
00422     return
00423     end
00424
00425
00426
00427     subroutine bckcol (iCol)
00428     include 'TKTRNX.FD'
00429     integer iColCode, iCol
00430     ibckcol= icolcode(iCol)
00431     return
00432     end
00433
00434
00435
00436     Subroutine defaultcolour
00437     call bckcol (0)
00438     call lincol (1)
00439     call txtcol (1)
00440     return
00441     end
00442
00443
00444     integer function icolcode (iCol)
00445     include 'FGRAPH.FD'
00446     integer icoltab (15)      ! Anpassung Farbindex an VGA-Palette
00447     data icoltab/ 15      ,12      ,10      ,11      ,9
00448 C      iCol=      1      2      3      4      5
00449 C      entspricht: weiss  rot      gruen   blau   lila
00450     1      iCol=      ,14      ,7      ,13      ,4      ,2
00451 C      entspricht: gelb   grau    violett mattrot mattgruen
00452 C      2      iCol=      ,1      ,3      ,6      ,8      ,5/
00453 C      entspricht: mattblau mattlila orange  mattgrau mattviolett
00454 C      iCol=      11      12      13      14      15
00455 C      entspricht: mattblau mattlila orange  mattgrau mattviolett
00456     if (icol.le.0) then
00457         icolcode= 0
00458     else if (icol.gt.15) then
00459         icolcode= icoltab(1)
00460     else
00461         icolcode= icoltab(icol)
00462     end if
00463     return
00464     end
00465
00466
00467
00468     integer function iscreenxcoord (iX)
00469     include 'TKTRNX.FD'
00470     iscreenxcoord= (ix*kscrx)/1023
00471     return
00472     end
00473
00474
00475
00476     integer function iscreenycoord (iY)
00477     include 'TKTRNX.FD'
00478     iscreenycoord= kscry- (kscry*iY)/780
00479     return
00480     end
00481
00482
00483
00484     integer function irevscreenxcoord (iX)
00485     include 'TKTRNX.FD'
00486     irevscreenxcoord= (ix*1023)/kscrx
00487     return
00488     end
00489
00490

```

```

00491
00492     integer function irevscreenycoord (iy)
00493     include 'TKTRNX.FD'
00494     irevscreenycoord= 780-(780*iy)/kscry
00495     return
00496     end
00497
00498
00499
00500     subroutine erase
00501     include 'FGRAPH.FD'
00502     include 'TKTRNX.FD'
00503     call clearscreen ($gclearscreen)
00504     ierr= setcolor(ibckcol)
00505     ierr= rectangle( $gfillinterior, 0, 0, kscrx, kscry)
00506     ierr= setcolor(ilincol)
00507     call movabs (kbeamx, kbeamy)      ! Cursorposition wiederherstellen
00508     return
00509     end
00510
00511
00512
00513     subroutine finitt
00514     implicit none
00515     include 'FGRAPH.FD'
00516     integer*2 ierr
00517     call graphicerror (12,' ')      ! Press any key to exit program
00518     call unregisterfonts ()
00519     ierr= setvideomode($defaultmode)
00520     call gincrsex
00521     call graphicerror (99,' ')      ! Jetzt auch STOP möglich
00522     return
00523     end
00524
00525 C
00526 C   Abspeichern Terminal Status Area
00527 C
00528
00529     subroutine svstat (Array)
00530     integer array(1)
00531     include 'TKTRNX.FD'
00532     integer arr(1)
00533     equivalence(arr(1),khomey)
00534     do 10 i=1,itktrnx1
00535 10    array(i)= arr(i)
00536     return
00537     end
00538
00539
00540
00541     subroutine restat (Array)
00542     integer array(1)
00543     include 'TKTRNX.FD'
00544     integer arr(1)
00545     equivalence(arr(1),khomey)
00546     do 10 i=1,itktrnx1
00547 10    arr(i)= array(i)
00548     call movabs (kbeamx, kbeamy)
00549     return
00550     end
00551
00552
00553 C
00554 C   Absolute Zeichenbefehle
00555 C
00556
00557     subroutine movabs (ix,iy)
00558     include 'FGRAPH.FD'
00559     include 'TKTRNX.FD'
00560     record /xycoord/ oldxy
00561     integer iscreenXcoord, iscreenYcoord
00562     call moveto (iscreenxcoord(ix),iscreenycoord(iy), oldxy)
00563     kbeamx= ix
00564     kbeamy= iy
00565     return
00566     end
00567
00568
00569
00570     subroutine pntabs (ix,iy)
00571     include 'FGRAPH.FD'
00572     include 'TKTRNX.FD'
00573     integer iscreenXcoord, iscreenYcoord
00574     integer oldPixel,ixs,iys
00575     record /xycoord/ oldxy
00576     ixs= iscreenxcoord(ix)
00577     iys= iscreenycoord(iy)

```



```

00578      call moveto (ixs,iys, oldxy)
00579      oldpixel= setpixel(ixs,iys)
00580      kbeamx= ix
00581      kbeamy= iy
00582      return
00583      end
00584
00585
00586
00587      subroutine drwabs (ix,iy)
00588      include 'FGRAPH.FD'
00589      include 'TKTRNX.FD'
00590      integer iScreenXcoord, iScreenYcoord
00591      ierr= lineto(iscreenxcoord(ix), iscreenycoord(iy))
00592      kbeamx= ix
00593      kbeamy= iy
00594      return
00595      end
00596
00597
00598
00599      subroutine dshabs (ix,iy, iMask)
00600      include 'FGRAPH.FD'
00601      include 'TKTRNX.FD'
00602      integer iScreenXcoord, iScreenYcoord
00603      integer*2 ierr
00604      if (imask.eq.0) then          ! solid line
00605          imask= 65535              ! 1111 1111 1111 1111
00606      else if (imask.eq.1) then ! dotted line
00607          imask= 43690              ! 1010 1010 1010 1010
00608      else if (imask.eq.2) then ! dash-dotted line
00609          imask= 58596              ! 1110 0100 1110 0100
00610      else if (imask.eq.3) then ! dashed line
00611          imask= 61680              ! 1111 0000 1111 0000
00612      end if
00613      call setlinestyle (imask)
00614      ierr= lineto(iscreenxcoord(ix), iscreenycoord(iy))
00615      call setlinestyle (65535) ! =#ffff, so zu WATCOM-Compiler kompatibel
00616      kbeamx= ix
00617      kbeamy= iy
00618      return
00619      end
00620
00621 C
00622 C Relative Zeichenbefehle
00623 C
00624
00625      subroutine movrel (iX, iY)
00626      include 'TKTRNX.FD'
00627      ix= kbeamx + ix
00628      iy= kbeamy + iy
00629      call movabs (ix, iy)
00630      return
00631      end
00632
00633
00634
00635      subroutine pntrel (iX, iY)
00636      include 'TKTRNX.FD'
00637      ix= kbeamx + ix
00638      iy= kbeamy + iy
00639      call pntabs (ix, iy)
00640      return
00641      end
00642
00643
00644
00645      subroutine drwrel (iX, iY)
00646      include 'TKTRNX.FD'
00647      ix= kbeamx + ix
00648      iy= kbeamy + iy
00649      call drwabs (ix, iy)
00650      return
00651      end
00652
00653
00654
00655      subroutine dshrel (iX, iY, iMask)
00656      include 'TKTRNX.FD'
00657      ix= kbeamx + ix
00658      iy= kbeamy + iy
00659      call dshabs (ix, iy, imask)
00660      return
00661      end
00662
00663 C
00664 C Ersatz SEELOC der CP/M-Version, SEELOC1 unnötig

```

```

00665 C
00666
00667     subroutine seeloc (IX,IY)
00668     include 'TKTRNX.FD'
00669     ix= kbeamx
00670     iy= kbeamy
00671     return
00672     end
00673
00674
00675
00676     Subroutine swindl (ix1,iy1, ix2,iy2)
00677     include 'FGRAPH.FD'
00678     integer iScreenXcoord, iScreenYcoord
00679     call setcliprgn (iscreenxcoord(ix1),iscreenycoord(iy1),
00680 1          iscreenxcoord(ix2),iscreenycoord(iy2))
00681     return
00682     end
00683
00684
00685
00686     Subroutine alpha
00687     implicit none
00688     include 'FGRAPH.FD'
00689     integer*2 iErr
00690     ierr= setvideomode($defaultmode)
00691     return
00692     end
00693
00694 C
00695 C Textausgabe
00696 C
00697
00698     subroutine csize (Ixlen,iylen)
00699     include 'TKTRNX.FD'
00700     ixlen= khorsz
00701     iylen= kversz
00702     return
00703     end
00704
00705
00706
00707     subroutine toutpt (iChr)
00708     include 'FGRAPH.FD'
00709     include 'TKTRNX.FD'
00710     record /xycoord/ oldxy
00711     integer iScreenXcoord, iScreenYcoord
00712     integer*2 iErr
00713     call moveto (iscreenxcoord(kbeamx),iscreenycoord(kbeamy+kversz)
00714 1          , oldxy)
00715     ierr= setcolor(itxtcol)
00716     call outgtext (char(ichr)//char(0))
00717     ierr= setcolor(ilincol)
00718     kbeamx= kbeamx+khorsz
00719     call moveto (iscreenxcoord(kbeamx), iscreenycoord(kbeamy), oldxy)
00720     return
00721     end
00722
00723
00724
00725     subroutine toutst (nChr, iChrArr)
00726     integer iChrArr (1)
00727     if (nchr.eq.0) return
00728     do 10 i=1,nchr
00729 10    call toutpt (ichrarr(i))
00730     return
00731     end
00732
00733
00734
00735     subroutine toutstc (String)
00736     character *(*) String
00737     do 10 i=1,istringlen(string)
00738 10    call toutpt (ichar(string(i:i)))
00739     return
00740     end
00741
00742
00743
00744     subroutine statst (String)
00745     include 'FGRAPH.FD'
00746     include 'TKTRNX.FD'
00747     record /rccoord/ s
00748     character *(*) String
00749     character *80 Buf
00750     buf= string(1:istringlen(string)) ! Mit Blanks auf 80 Zeichen aufgefüllt
00751     call setttextposition (1,1,s)

```

```

00752      call outtext (buf(1:min(80,kstcol)))
00753      return
00754      end
00755
00756 C
00757 C   Eingabe
00758 C
00759
00760      subroutine tinput (iChr)
00761      integer *2 kTinput
00762      ichr= ktinput() ! Konversion Integer*2 nach *4 durch Compiler
00763      return
00764      end
00765
00766
00767      subroutine dcursr (IC,IX,IY)
00768      include 'TKTRNX.FD'
00769      integer ic, ix, iy
00770      integer*2 ic2, ix2, iy2
00771      if (imouse.ne.0) then
00772          call gincrs (ic2,ix2,iy2)
00773          ix= ix2
00774          iy= iy2
00775          ic= ic2
00776      else
00777          call graphicerror (5, ' ') ! No Mousedriver available, use Keyboard
00778          call tinput (ic)
00779          ix= 0
00780          iy= 0
00781      end if
00782      ix= irevscreenxcoord(ix)
00783      iy= irevscreenycoord(iy)
00784      return
00785      end
00786
00787 C
00788 C   Interface lib$movc3 (Anpassung Parameterübergabe durch "TcsDDosA.FI"
00789 C
00790      subroutine lib_movc3 (iLen, sou, dst)
00791      integer iLen
00792      character *(*) sou,dst
00793      call lib_movc3_ (iLen, sou, dst)
00794      return
00795      end
00796
00797 C
00798 C   Dummyroutinen
00799 C
00800
00801      subroutine anmode
00802      entry      alfmod
00803      entry      pclipt
00804      entry      iowait
00805      return
00806      end

```

3.42 TKTRNX.fd File Reference

DOS Port: TCS Common Block TKTRNX.

3.42.1 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](#).

3.43 TKTRNX.fd

```

00001 C> \file TKTRNX.fd
00002 C> \brief   DOS Port: TCS Common Block TKTRNX
00003 C> \version 1.0
00004 C> \author  Dr.-Ing. Klaus Friedewald
00005 C> \~german
00006 C> Common Block TKTRNX, Version für DOS und INTEGER*4 Variablen (WATCOM-Compiler)
00007 C> \~english
00008 C> Common Block TKTRNX, version for DOS and INTEGER*4 variables (WATCOM-Compiler)
00009 C> \~german
00010 C> \note
00011 C> Da die folgende Definition kein Bestandteil eines Moduls
00012 C> ist, versagt der DOXYGEN-Parser bei der Kombination von
00013 C> COMMON und integer. Workaround: \\cond ... \\endcond
00014 C> \~english
00015 C> Because the following declaration not beeing part of a module, DOXYGEN could
00016 C> not interpret the combination COMMON / INTEGER.
00017 C> Workaround: \\cond ... \\endcond
00018 C> \~
00019 C> \cond
00020 C>
00021 C Common Block TKTRNX, Version für DOS und INTEGER*4 Variablen (WATCOM-Compiler)
00022 C
00023     COMMON /tktrnx/
00024 c         kbaudr,kerror,kgraf1,
00025     1 khomey,
00026 c         kkmode,
00027     2 khorsz,kversz,
00028 c         kitalc,ksizef,
00029     3 klmrgn,kmrgn, kscrx,kscry,
00030 c         ktblsz,khorzt(10),kvertt(10),
00031     4 kbeamx,kbeamy,
00032 c         kmovef,kpchar(4),kdasht,
00033     5 kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
00034 c         trealx,trealy,timagx,timagy,
00035     6 trcosf,trsinf,trscal
00036     u ,xfac,yfac,xlog,ylog,kstcol,
00037     u ilincol, ibckcol, itxtcol, imouse
00038     SAVE /tktrnx/
00039
00040     integer iTktrnxL
00041     parameter(itktrnxL=29) ! +11)
00042
00043 c Neue Variablen:
00044 c     kScrX, kScrY: Zeichenfläche in Pixeln
00045 c         Unterer Bildschirmrand für eine Statuszeile freigehalten
00046 c     kBeamX, kBeamY: Aktuelle Strahlposition im (1024/780) Koordinatensystem
00047 c     kStCol: Maximale Zeichenzahl in der Statuszeile
00048 c     iLinCol, iBckCol, iTxtCol: Farbindices
00049 c     iMouse: Anzahl der Maustasten. iMouse=0: keine Maus vorhanden
00050 c
00051 c Achtung:
00052 c     Anpassung Parameters iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
00053 c     Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
00054 c
00055 C
00056 C> \endcond

```

Index

AG2.for, [5](#)

ag2lev, [8](#)

alfsetc, [8](#)

bar, [8](#)

binitt, [8](#)

bsyms, [8](#)

calcon, [8](#)

calpnt, [9](#)

check, [9](#)

cmnmx, [9](#)

coptim, [9](#)

cplot, [9](#)

datget, [10](#)

dinitx, [10](#)

dinity, [10](#)

dlimx, [10](#)

dlimy, [10](#)

dsplay, [11](#)

eformc, [11](#)

esplit, [11](#)

expoutc, [11](#)

fformc, [11](#)

filbox, [12](#)

findge, [12](#)

findle, [12](#)

fonlyc, [12](#)

frame, [13](#)

gline, [13](#)

grid, [13](#)

hbarst, [13](#)

iformc, [13](#)

infin, [14](#)

iother, [14](#)

iubgc, [14](#)

justerc, [14](#)

keyset, [14](#)

label, [15](#)

leap, [15](#)

line, [15](#)

locge, [15](#)

locle, [15](#)

logtix, [16](#)

loptim, [16](#)

lwidth, [16](#)

mnmx, [16](#)

monpos, [16](#)

notatec, [17](#)

npts, [17](#)

numsetc, [17](#)

optim, [17](#)

oubgc, [17](#)

place, [18](#)

remlab, [18](#)

rescom, [18](#)

rgchek, [18](#)

roundd, [18](#)

roundu, [19](#)

savcom, [19](#)

setwin, [19](#)

sizel, [19](#)

sizes, [19](#)

slimx, [20](#)

slimy, [20](#)

spread, [20](#)

stepl, [20](#)

steps, [20](#)

symbl, [21](#)

symout, [21](#)

teksym, [21](#)

teksym1, [21](#)

tset, [21](#)

tset2, [22](#)

typck, [22](#)

vbarst, [22](#)

vlablc, [22](#)

width, [22](#)

xden, [23](#)

xetyp, [23](#)

xfrm, [23](#)

xlab, [23](#)

xlen, [23](#)

xloc, [23](#)

xloctp, [24](#)

xmfrm, [24](#)

xmtcs, [24](#)

xneat, [24](#)

xtics, [24](#)

xtype, [24](#)

xwidth, [25](#)

xzero, [25](#)

yden, [25](#)

yetyp, [25](#)

yfrm, [25](#)

ylab, [25](#)

ylen, [26](#)

yloc, [26](#)

ylocrt, [26](#)

ymdyd, [26](#)

- ymfrm, [26](#)
- ymtcs, [27](#)
- yneat, [27](#)
- ytics, [27](#)
- ytype, [27](#)
- ywdth, [27](#)
- yzero, [27](#)
- AG2Holerith.for, [63](#)
 - alfset, [64](#)
 - comdmp, [64](#)
 - comget, [64](#)
 - comset, [65](#)
 - eform, [65](#)
 - expout, [65](#)
 - fform, [65](#)
 - fonly, [65](#)
 - hlabel, [66](#)
 - hstrin, [66](#)
 - ibasec, [66](#)
 - ibasex, [66](#)
 - ibasey, [66](#)
 - iform, [67](#)
 - juster, [67](#)
 - notate, [67](#)
 - numset, [67](#)
 - vlabel, [68](#)
 - vstrin, [68](#)
- ag2lev
 - AG2.for, [8](#)
- AG2uline.for, [73](#)
 - uline, [74](#)
- AG2umnmix.for, [74](#)
 - umnmix, [75](#)
- AG2upoint.for, [75](#)
 - upoint, [75](#)
- AG2users.for, [76](#)
 - users, [76](#)
- AG2useset.for, [77](#)
 - useset, [77](#)
- AG2usesetC.for, [78](#)
 - usesetc, [78](#)
- AG2UstrSoftek.for, [79](#)
 - softek, [79](#)
- alfset
 - AG2Holerith.for, [64](#)
- alfsetc
 - AG2.for, [8](#)
- alpha
 - TCSdrDOS.for, [128](#)
- ancho
 - TCS.for, [101](#)
- anmode
 - TCSdrDOS.for, [129](#)
- anstr
 - TCS.for, [101](#)
- baksp
 - TCS.for, [101](#)
- bar
 - AG2.for, [8](#)
- bckcol
 - TCSdrDOS.for, [129](#)
- bell
 - TCSdDosa.asm, [115](#)
- binitt
 - AG2.for, [8](#)
- bsyms
 - AG2.for, [8](#)
- calcon
 - AG2.for, [8](#)
- calpnt
 - AG2.for, [9](#)
- cartn
 - TCS.for, [101](#)
- check
 - AG2.for, [9](#)
- CloseBytFil
 - TCSdDosa.asm, [115](#)
- cmnmix
 - AG2.for, [9](#)
- comdmp
 - AG2Holerith.for, [64](#)
- comget
 - AG2Holerith.for, [64](#)
- comset
 - AG2Holerith.for, [65](#)
- coptim
 - AG2.for, [9](#)
- cplot
 - AG2.for, [9](#)
- csize
 - TCSdrDOS.for, [129](#)
- dasha
 - TCS.for, [102](#)
- dashr
 - TCS.for, [102](#)
- datget
 - AG2.for, [10](#)
- dcursr
 - TCSdrDOS.for, [129](#)
- defaultcolour
 - TCSdrDOS.for, [129](#)
- dinitx
 - AG2.for, [10](#)
- dinity
 - AG2.for, [10](#)
- dlimx
 - AG2.for, [10](#)
- dlimy
 - AG2.for, [10](#)
- drawa
 - TCS.for, [102](#)
- drawr
 - TCS.for, [102](#)
- drwabs
 - TCSdrDOS.for, [130](#)

- drwrel
 - TCSdrDOS.for, [130](#)
- dshabs
 - TCSdrDOS.for, [130](#)
- dshrel
 - TCSdrDOS.for, [130](#)
- dsplay
 - AG2.for, [11](#)
- dwindo
 - TCS.for, [102](#)
- eform
 - AG2Holerith.for, [65](#)
- eformc
 - AG2.for, [11](#)
- erase
 - TCSdrDOS.for, [130](#)
- esplit
 - AG2.for, [11](#)
- expout
 - AG2Holerith.for, [65](#)
- expoutc
 - AG2.for, [11](#)
- fform
 - AG2Holerith.for, [65](#)
- fformc
 - AG2.for, [11](#)
- Fgraph.fd, [79](#)
- Fgraph.fi, [85](#)
- filbox
 - AG2.for, [12](#)
- findge
 - AG2.for, [12](#)
- findle
 - AG2.for, [12](#)
- finitt
 - TCSdrDOS.for, [131](#)
- fonly
 - AG2Holerith.for, [65](#)
- fonlyc
 - AG2.for, [12](#)
- frame
 - AG2.for, [13](#)
- G2dAG2.fd, [87](#)
- genflg
 - TCS.for, [103](#)
- GetEnv
 - TCSdDosa.asm, [116](#)
- gethdc
 - GetHDC.for, [89](#)
- GetHDC.for, [88](#)
- gethdc, [89](#)
- GinCrs
 - TCSdDosa.asm, [116](#)
- GinCrsEx
 - TCSdDosa.asm, [116](#)
- GinCrsIn
 - TCSdDosa.asm, [116](#)
- gline
 - AG2.for, [13](#)
- graphicerrorinit
 - TCSdrDOS.for, [131](#)
- grid
 - AG2.for, [13](#)
- hbarst
 - AG2.for, [13](#)
- hdcopy
 - hdcopy.for, [91](#)
- hdcopy.for, [91](#)
- hdcopy, [91](#)
- writebuf, [92](#)
- hlabel
 - AG2Holerith.for, [66](#)
- home
 - TCS.for, [103](#)
- hstrin
 - AG2Holerith.for, [66](#)
- ibasec
 - AG2Holerith.for, [66](#)
- ibasex
 - AG2Holerith.for, [66](#)
- ibasey
 - AG2Holerith.for, [66](#)
- icolcode
 - TCSdrDOS.for, [131](#)
- iform
 - AG2Holerith.for, [67](#)
- iformc
 - AG2.for, [13](#)
- infin
 - AG2.for, [14](#)
- initt
 - TCSdrDOS.for, [131](#)
- initt1
 - TCSdrDOS.for, [131](#)
- iother
 - AG2.for, [14](#)
- irevscreenxcoord
 - TCSdrDOS.for, [131](#)
- irevscreenycoord
 - TCSdrDOS.for, [132](#)
- iscreenxcoord
 - TCSdrDOS.for, [132](#)
- iscreenycoord
 - TCSdrDOS.for, [132](#)
- istringlen
 - Strings.for, [97](#)
- italic
 - TCSdrDOS.for, [132](#)
- itrimlen
 - Strings.for, [97](#)
- iubgc
 - AG2.for, [14](#)

juster
 AG2Holerith.for, 67
justerc
 AG2.for, 14
keyset
 AG2.for, 14
kinput
 TCSdDosa.asm, 117
label
 AG2.for, 15
leap
 AG2.for, 15
lib_movc3
 TCSdDosa.asm, 117
 TCSdrDOS.for, 132
lincol
 TCSdrDOS.for, 133
line
 AG2.for, 15
linef
 TCS.for, 103
linhgt
 TCS.for, 103
lintrn
 TCS.for, 103
linwdt
 TCS.for, 104
locge
 AG2.for, 15
locle
 AG2.for, 15
logtix
 AG2.for, 16
logtrn
 TCS.for, 104
loptim
 AG2.for, 16
lwidth
 AG2.for, 16
Mainpage.dox, 95
mmmx
 AG2.for, 16
monpos
 AG2.for, 16
movabs
 TCSdrDOS.for, 133
movea
 TCS.for, 104
mover
 TCS.for, 104
movrel
 TCSdrDOS.for, 133
newlin
 TCS.for, 104
newpag
 TCS.for, 105
notate
 AG2Holerith.for, 67
notatec
 AG2.for, 17
npts
 AG2.for, 17
numset
 AG2Holerith.for, 67
numsetc
 AG2.for, 17
OpenBytFil
 TCSdDosa.asm, 117
optim
 AG2.for, 17
oubgc
 AG2.for, 17
outtext
 outtext.for, 96
outtext.for, 95
 outtext, 96
place
 AG2.for, 18
pntabs
 TCSdrDOS.for, 133
pntrel
 TCSdrDOS.for, 133
pointa
 TCS.for, 105
pointr
 TCS.for, 105
printstring
 Strings.for, 97
rel2ab
 TCS.for, 105
remlab
 AG2.for, 18
rescal
 TCS.for, 105
rescom
 AG2.for, 18
restat
 TCSdrDOS.for, 134
revcot
 TCS.for, 106
rgchek
 AG2.for, 18
roundd
 AG2.for, 18
roundu
 AG2.for, 19
rrotat
 TCS.for, 106
rscale
 TCS.for, 106

- savcom
 - AG2.for, [19](#)
- seeloc
 - TCSdrDOS.for, [134](#)
- seetrm
 - TCS.for, [106](#)
- seetrn
 - TCS.for, [106](#)
- setmrg
 - TCS.for, [107](#)
- setwin
 - AG2.for, [19](#)
- sizel
 - AG2.for, [19](#)
- sizes
 - AG2.for, [19](#)
- slimx
 - AG2.for, [20](#)
- slimy
 - AG2.for, [20](#)
- softek
 - AG2UsrSoftek.for, [79](#)
- spread
 - AG2.for, [20](#)
- statst
 - TCSdrDOS.for, [134](#)
- stepl
 - AG2.for, [20](#)
- steps
 - AG2.for, [20](#)
- Strings.for, [96](#)
 - istringlen, [97](#)
 - itrimlen, [97](#)
 - printstring, [97](#)
 - substitute, [97](#)
- substitute
 - Strings.for, [97](#)
- svstat
 - TCSdrDOS.for, [134](#)
- swind1
 - TCSdrDOS.for, [134](#)
- swindo
 - TCS.for, [107](#)
- syml
 - AG2.for, [21](#)
- symout
 - AG2.for, [21](#)
- TCS.for, [100](#)
 - ancho, [101](#)
 - anstr, [101](#)
 - baksp, [101](#)
 - cartn, [101](#)
 - dasha, [102](#)
 - dashr, [102](#)
 - drawa, [102](#)
 - drawr, [102](#)
 - dwindo, [102](#)
 - genflg, [103](#)
 - home, [103](#)
 - linef, [103](#)
 - linhgt, [103](#)
 - lintrn, [103](#)
 - linwdt, [104](#)
 - logtrn, [104](#)
 - movea, [104](#)
 - mover, [104](#)
 - newlin, [104](#)
 - newpag, [105](#)
 - pointa, [105](#)
 - pointr, [105](#)
 - rel2ab, [105](#)
 - rescal, [105](#)
 - revcot, [106](#)
 - rrotat, [106](#)
 - rscale, [106](#)
 - seetrm, [106](#)
 - seetrn, [106](#)
 - setmrg, [107](#)
 - swindo, [107](#)
 - twindo, [107](#)
 - vcursr, [107](#)
 - vwindo, [107](#)
 - wincot, [108](#)
- TCSdDosa.asm, [114](#)
 - bell, [115](#)
 - CloseBytFil, [115](#)
 - GetEnv, [116](#)
 - GinCrs, [116](#)
 - GinCrsEx, [116](#)
 - GinCrsIn, [116](#)
 - ktinput, [117](#)
 - lib_movc3, [117](#)
 - OpenBytFil, [117](#)
 - WrtBytFil, [118](#)
- TCSdDosa.fi, [125](#)
- TCSdrDOS.for, [127](#)
 - alpha, [128](#)
 - anmode, [129](#)
 - bckcol, [129](#)
 - csize, [129](#)
 - dcursr, [129](#)
 - defaultcolour, [129](#)
 - drwabs, [130](#)
 - drwrel, [130](#)
 - dshabs, [130](#)
 - dshrel, [130](#)
 - erase, [130](#)
 - finitt, [131](#)
 - graphicerrorinit, [131](#)
 - icolcode, [131](#)
 - initt, [131](#)
 - initt1, [131](#)
 - irevscreenxcoord, [131](#)
 - irevscreenycoord, [132](#)
 - iscreenxcoord, [132](#)
 - iscreenycoord, [132](#)

- italic, 132
- lib_movc3, 132
- lincol, 133
- movabs, 133
- movrel, 133
- pntabs, 133
- pntrel, 133
- restat, 134
- seeloc, 134
- statst, 134
- svstat, 134
- swind1, 134
- tcslev, 135
- tinput, 135
- toutpt, 135
- toutst, 135
- toutstc, 135
- txtcol, 136
- tcslev
 - TCSdrDOS.for, 135
- teksym
 - AG2.for, 21
- teksym1
 - AG2.for, 21
- tinput
 - TCSdrDOS.for, 135
- TKTRNX.fd, 145
- toutpt
 - TCSdrDOS.for, 135
- toutst
 - TCSdrDOS.for, 135
- toutstc
 - TCSdrDOS.for, 135
- tset
 - AG2.for, 21
- tset2
 - AG2.for, 22
- twindo
 - TCS.for, 107
- txtcol
 - TCSdrDOS.for, 136
- typck
 - AG2.for, 22
- uline
 - AG2uline.for, 74
- umnmx
 - AG2umnmx.for, 75
- upoint
 - AG2upoint.for, 75
- users
 - AG2users.for, 76
- useset
 - AG2useset.for, 77
- usesetc
 - AG2usesetc.for, 78
- vbarst
 - AG2.for, 22
- vcursr
 - TCS.for, 107
- vlabel
 - AG2Holerith.for, 68
- vlablc
 - AG2.for, 22
- vstrin
 - AG2Holerith.for, 68
- vwindo
 - TCS.for, 107
- width
 - AG2.for, 22
- wincot
 - TCS.for, 108
- writebuf
 - hdcopy.for, 92
- WrtBytFil
 - TCSdDosa.asm, 118
- xden
 - AG2.for, 23
- xetyp
 - AG2.for, 23
- xfrm
 - AG2.for, 23
- xlab
 - AG2.for, 23
- xlen
 - AG2.for, 23
- xloc
 - AG2.for, 23
- xloctp
 - AG2.for, 24
- xmfrm
 - AG2.for, 24
- xmtcs
 - AG2.for, 24
- xneat
 - AG2.for, 24
- xtics
 - AG2.for, 24
- xtype
 - AG2.for, 24
- xwdth
 - AG2.for, 25
- xzero
 - AG2.for, 25
- yden
 - AG2.for, 25
- yetyp
 - AG2.for, 25
- yfrm
 - AG2.for, 25
- ylab
 - AG2.for, 25
- ylen
 - AG2.for, 26

yloc
 AG2.for, [26](#)
ylocrt
 AG2.for, [26](#)
ymdyd
 AG2.for, [26](#)
ymfrm
 AG2.for, [26](#)
ymtcs
 AG2.for, [27](#)
yneat
 AG2.for, [27](#)
ytics
 AG2.for, [27](#)
ytype
 AG2.for, [27](#)
ywdth
 AG2.for, [27](#)
yzero
 AG2.for, [27](#)