Graph2D Library --- Windows ---

Generated by Doxygen 1.8.19

Chapter 1

Plot10 & Advanced Graphing II

Graph2D is written entirely in FTN77 and ANSI C90. Initially it was developed using the Open Watcom compiler. Now the MINGW-GCC is used additionally to allow linking against applications written in modern Fortran.

1.0.0.1 How to build the library:

Copy the sources into the /build subdirectory by running "$$getfiles.bat win32 (win16, gnu32, gnu64...)" and then use the workspace files.

1.0.0.2 Using the library:

After building the library and linking it to an application, the main features can be changed by the following files:

- Initialization: by calling the WINLBL subroutine, editing the registry or by ∗.ini/∗.xml files
- Icons: by linking to a resource or using ∗.ini-files

1.0.0.3 Hardcopies

By default ∗.wmf hardcopies are used, but other formats can be configured before compiling the package.

Chapter 2

Compiler setup and foreign libraries

2.0.1 Setting up the IDE

2.0.1.1 Open source libraries

Building and storing of the binaries in /OpenContent/binaries/... is only necessary once, and only when using a new compiler.

sglib is a macro library, no compilation is required:

- Copy the file "sglib.h" into the /include directories.
- Copy the file "index.html" -> TekLib\OpenContent\docs\sglib

2.0.1.2 OpenWatcom for Windows 16bit and 32bit

2.0.1.2.1 Basic configuration of the IDE Create the directory C:\UsrProg\Watcom and then "Run as Administrator" open-watcom-2_0-c-win-x64.exe and open-watcom-2_0-f77-win-x64.exe with the following options

- 16bit compiler: All
- 32bit compiler: All
- Target: DOS, Win16, Win NT
- Host: Win 64
- Toolkit: All

2.0.1.2.2 Build the miniXML library:

- Unzip mxml-x.y.zip to \build
- Copy OpenContent\MiniXMLlib\OpenWatcom∗.∗ to \build
- Build the static version with mxml1.wpj and the DLL-version with mxml1d.wpj
- Copy from \build:
  mxml.h -> TekLib\OpenContent\binaries\Watcom mxml1.lib
  !!! Caution, DLL is only of limited use: Erroneous file operations "Unable to read XML file with default callback." !!!
  mxml1d.lib, mxml1d.dll ->TekLib\OpenContent\binaries\Watcom\lib
- Copy the documentation from \build\doc:
  mxml.html, mxml-cover.png -> TekLib\OpenContent\docs\Mini-XML

2.0.1.3 MingGW (TDM and CodeBlocks) for Windows 32bit and 64bit

2.0.1.3.1 Basic configuration of the IDE Install both TDM toolchains, for 32-bit and for 64-bit (e.g. in C:\Usr↩ Prog\TDM-GCC-64 and C:\UsrProg\TDM-GCC-32). Then edit the following entries in CodeBlocks at Settings -> Compiler:

- GNU GCC Compiler:
  "Compiler Settings" -> "Compiler Flags" General\Target 64bit [-m64]
  "Toolchain executables": C:\UsrProg\TDM-GCC-64
- GNU Fortran Compiler:
  "Compiler Settings" -> "Other Compiler options": -m64
  "Toolchain executables" : C:\UsrProg\TDM-GCC-64

To build 32bit programs the global GCC settings must be changed accordingly. The 32bit settings define new compilers and can now be distinguished from the 64bit versions when used within the 32bit workspaces.

2.0.1.3.2 Building the miniXML library MiniXML: The compilation uses an MSYS terminal, seperately for 32- and 64-bit.

- Unzip mxml-x.y.zip
- $ cd /home/mxml-x.y
- $ ./configure –help
- For 32bit: $ ./configure –build=mingw32
  For 64bit: $ ./configure –build=mingw64
- Edit makefile and add the following flags:
  LIBS = -lpthread -lssp
- $ make
- $ make test

- $ exit
- Copy (in MS Windows):
  mxml.h -> TekLib\OpenContent\binaries\gcc libmxml.a, (libmxml1.a, mxml1.dll) ->TekLib\Open↩
  Content\binaries\gcc\lib
- Copy the documentation:
  mxml.html, mxml-cover.png -> TekLib\OpenContent\docs\Mini-XML

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Data Type Documentation

5.1 TKTRNXcommonBlock Struct Reference

`#include <TKTRNX.h>`

Public Attributes

- FTNINT khomey
- FTNINT khorsz
- FTNINT kversz
- FTNINT kitalc
- FTNINT ksizef
- FTNINT klmrgn
- FTNINT krmrgn
- FTNINT kBeamX
- FTNINT kBeamY
- FTNINT kminsx
- FTNINT kminsy
- FTNINT kmaxsx
- FTNINT kmaxsy
- FTNREAL tminvx
- FTNREAL tminvy
- FTNREAL tmaxvx
- FTNREAL tmaxvy
- FTNREAL trcosf
- FTNREAL trsinf
- FTNREAL trscal
- FTNREAL xfac
- FTNREAL yfac
- FTNREAL xlog
- FTNREAL ylog
- FTNINT kStCol
- FTNINT iLinCol
- FTNINT iBckCol
- FTNINT iTxtCol
- FTNINT iMouse

5.1.1 Detailed Description

Definition at line 24 of file TKTRNX.h.

5.1.2 Member Data Documentation

5.1.2.1 iBckCol

```
FTNINT TKTRNXcommonBlock::iBckCol
```

Definition at line 45 of file TKTRNX.h.

5.1.2.2 iLinCol

```
FTNINT TKTRNXcommonBlock::iLinCol
```

Definition at line 45 of file TKTRNX.h.

5.1.2.3 iMouse

```
FTNINT TKTRNXcommonBlock::iMouse
```

Definition at line 45 of file TKTRNX.h.

5.1.2.4 iTxtCol

```
FTNINT TKTRNXcommonBlock::iTxtCol
```

Definition at line 45 of file TKTRNX.h.

5.1.2.5 kBeamX

```
FTNINT TKTRNXcommonBlock::kBeamX
```

Definition at line 34 of file TKTRNX.h.

5.1.2.6 kBeamY

```
FTNINT TKTRNXcommonBlock::kBeamY
```

Definition at line 34 of file TKTRNX.h.

5.1.2.7 khomey

```
FTNINT TKTRNXcommonBlock::khomey
```

Definition at line 27 of file TKTRNX.h.

5.1.2.8 khorsz

```
FTNINT TKTRNXcommonBlock::khorsz
```

Definition at line 29 of file TKTRNX.h.

5.1.2.9 kitalc

```
FTNINT TKTRNXcommonBlock::kitalc
```

Definition at line 30 of file TKTRNX.h.

5.1.2.10 klmrgn

```
FTNINT TKTRNXcommonBlock::klmrgn
```

Definition at line 31 of file TKTRNX.h.

5.1.2.11 kmaxsx

```
FTNINT TKTRNXcommonBlock::kmaxsx
```

Definition at line 36 of file TKTRNX.h.

5.1.2.12 kmaxsy

```
FTNINT TKTRNXcommonBlock::kmaxsy
```

Definition at line 36 of file TKTRNX.h.

5.1.2.13 kminsx

```
FTNINT TKTRNXcommonBlock::kminsx
```

Definition at line 36 of file TKTRNX.h.

5.1.2.14 kminsy

```
FTNINT TKTRNXcommonBlock::kminsy
```

Definition at line 36 of file TKTRNX.h.

5.1.2.15 krmrgn

```
FTNINT TKTRNXcommonBlock::krmrgn
```

Definition at line 31 of file TKTRNX.h.

5.1.2.16 ksizef

```
FTNINT TKTRNXcommonBlock::ksizef
```

Definition at line 30 of file TKTRNX.h.

5.1.2.17 kStCol

```
FTNINT TKTRNXcommonBlock::kStCol
```

Definition at line 44 of file TKTRNX.h.

5.1.2.18 kversz

```
FTNINT TKTRNXcommonBlock::kversz
```

Definition at line 29 of file TKTRNX.h.

5.1.2.19 tmaxvx

```
FTNREAL TKTRNXcommonBlock::tmaxvx
```

Definition at line 39 of file TKTRNX.h.

5.1.2.20 tmaxvy

```
FTNREAL TKTRNXcommonBlock::tmaxvy
```

Definition at line 39 of file TKTRNX.h.

5.1.2.21 tminvx

```
FTNREAL TKTRNXcommonBlock::tminvx
```

Definition at line 39 of file TKTRNX.h.

5.1.2.22 tminvy

```
FTNREAL TKTRNXcommonBlock::tminvy
```

Definition at line 39 of file TKTRNX.h.

5.1.2.23 trcosf

```
FTNREAL TKTRNXcommonBlock::trcosf
```

Definition at line 41 of file TKTRNX.h.

5.1.2.24 trscal

```
FTNREAL TKTRNXcommonBlock::trscal
```

Definition at line 41 of file TKTRNX.h.

5.1.2.25 trsinf

```
FTNREAL TKTRNXcommonBlock::trsinf
```

Definition at line 41 of file TKTRNX.h.

5.1.2.26 xfac

```
FTNREAL TKTRNXcommonBlock::xfac
```

Definition at line 42 of file TKTRNX.h.

5.1.2.27 xlog

```
FTNREAL TKTRNXcommonBlock::xlog
```

Definition at line 42 of file TKTRNX.h.

5.1.2.28 yfac

```
FTNREAL TKTRNXcommonBlock::yfac
```

Definition at line 42 of file TKTRNX.h.

5.1.2.29 ylog

```
FTNREAL TKTRNXcommonBlock::ylog
```

Definition at line 42 of file TKTRNX.h.

The documentation for this struct was generated from the following file:

- TKTRNX.h

Chapter 6

File Documentation

6.1 AG2.for File Reference

Graph2D: Tektronix Advanced Graphing II Emulation.

Functions/Subroutines

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

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

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

6.1.1 Detailed Description

Graph2D: Tektronix Advanced Graphing II Emulation.

Version

(2025,347, x)

Author

(C) 2025 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Layer 2: scientific 2-D graphic subroutines

Note

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

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

Definition in file AG2.for.

6.1.2 Function/Subroutine Documentation

6.1.2.1 ag2infin()

```
real function ag2infin
```

Definition at line 155 of file AG2.for.

6.1.2.2 ag2lev()

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

Definition at line 94 of file AG2.for.

6.1.2.3 alfsetc()

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

Definition at line 2574 of file AG2.for.

6.1.2.4 bar()

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

Definition at line 1698 of file AG2.for.

6.1.2.5 binitt()

```
subroutine binitt
```

Definition at line 724 of file AG2.for.

6.1.2.6 bsyms()

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

Definition at line 1850 of file AG2.for.

6.1.2.7 calcon()

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

Definition at line 1336 of file AG2.for.

6.1.2.8 calpnt()

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

Definition at line 1281 of file AG2.for.

6.1.2.9 check()

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

Definition at line 808 of file AG2.for.

6.1.2.10 cmnmx()

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

Definition at line 930 of file AG2.for.

6.1.2.11 coptim()

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

Definition at line 1125 of file AG2.for.

6.1.2.12 cplot()

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

Definition at line 1548 of file AG2.for.

6.1.2.13 datget()

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

Definition at line 1670 of file AG2.for.

6.1.2.14 dinitx()

```
subroutine dinitx
```

Definition at line 654 of file AG2.for.

6.1.2.15 dinity()

```
subroutine dinity
```

Definition at line 668 of file AG2.for.

6.1.2.16 dlimx()

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

Definition at line 474 of file AG2.for.

6.1.2.17 dlimy()

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

Definition at line 486 of file AG2.for.

6.1.2.18 dsplay()

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

Definition at line 1534 of file AG2.for.

6.1.2.19 eformc()

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

Definition at line 2445 of file AG2.for.

6.1.2.20 esplit()

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

Definition at line 2478 of file AG2.for.

6.1.2.21 expoutc()

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

Definition at line 2498 of file AG2.for.

6.1.2.22 fformc()

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

Definition at line 2385 of file AG2.for.

6.1.2.23 filbox()

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

Definition at line 1765 of file AG2.for.

6.1.2.24 findge()

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

Definition at line 2933 of file AG2.for.

6.1.2.25 findle()

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

Definition at line 2952 of file AG2.for.

6.1.2.26 fonlyc()

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

Definition at line 2414 of file AG2.for.

6.1.2.27 frame()

```
subroutine frame
```

Definition at line 1520 of file AG2.for.

6.1.2.28 gline()

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

Definition at line 2183 of file AG2.for.

6.1.2.29 grid()

```
subroutine grid
```

Definition at line 1966 of file AG2.for.

6.1.2.30 hbarst()

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

Definition at line 682 of file AG2.for.

6.1.2.31 iformc()

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

Definition at line 2353 of file AG2.for.

6.1.2.32 infin()

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

Definition at line 142 of file AG2.for.

6.1.2.33 iother()

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

Definition at line 3077 of file AG2.for.

6.1.2.34 iubgc()

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

Definition at line 1483 of file AG2.for.

6.1.2.35 justerc()

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

Definition at line 2677 of file AG2.for.

6.1.2.36 keyset()

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

Definition at line 1644 of file AG2.for.

6.1.2.37 label()

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

Definition at line 2210 of file AG2.for.

6.1.2.38 leap()

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

Definition at line 1469 of file AG2.for.

6.1.2.39 line()

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

Definition at line 109 of file AG2.for.

6.1.2.40 locge()

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

Definition at line 2974 of file AG2.for.

6.1.2.41 locle()

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

Definition at line 2992 of file AG2.for.

6.1.2.42 logtix()

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

Definition at line 2052 of file AG2.for.

6.1.2.43 loptim()

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

Definition at line 998 of file AG2.for.

6.1.2.44 lwidth()

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

Definition at line 2743 of file AG2.for.

6.1.2.45 mnmx()

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

Definition at line 891 of file AG2.for.

6.1.2.46 monpos()

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

Definition at line 2169 of file AG2.for.

6.1.2.47 notatec()

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

Definition at line 2629 of file AG2.for.

6.1.2.48 npts()

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

Definition at line 165 of file AG2.for.

6.1.2.49 numsetc()

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

Definition at line 2326 of file AG2.for.

6.1.2.50 optim()

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

Definition at line 981 of file AG2.for.

6.1.2.51 oubgc()

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

Definition at line 1497 of file AG2.for.

6.1.2.52 place()

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

Definition at line 522 of file AG2.for.

6.1.2.53 remlab()

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

Definition at line 2818 of file AG2.for.

6.1.2.54 rescom()

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

Definition at line 3061 of file AG2.for.

6.1.2.55 rgchek()

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

Definition at line 864 of file AG2.for.

6.1.2.56 roundd()

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

Definition at line 3010 of file AG2.for.

6.1.2.57 roundu()

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

Definition at line 3026 of file AG2.for.

6.1.2.58 savcom()

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

Definition at line 3045 of file AG2.for.

6.1.2.59 setwin()

```
subroutine setwin
```

Definition at line 632 of file AG2.for.

6.1.2.60 sizel()

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

Definition at line 198 of file AG2.for.

6.1.2.61 sizes()

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

Definition at line 187 of file AG2.for.

6.1.2.62 slimx()

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

Definition at line 498 of file AG2.for.

6.1.2.63 slimy()

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

Definition at line 510 of file AG2.for.

6.1.2.64 spread()

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

Definition at line 2881 of file AG2.for.

6.1.2.65 stepl()

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

Definition at line 176 of file AG2.for.

6.1.2.66 steps()

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

Definition at line 131 of file AG2.for.

6.1.2.67 symbl()

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

Definition at line 120 of file AG2.for.

6.1.2.68 symout()

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

Definition at line 1867 of file AG2.for.

6.1.2.69 teksym()

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

Definition at line 1892 of file AG2.for.

6.1.2.70 teksym1()

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

Definition at line 1940 of file AG2.for.

6.1.2.71 tset()

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

Definition at line 2099 of file AG2.for.

6.1.2.72 tset2()

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

Definition at line 2137 of file AG2.for.

6.1.2.73 typck()

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

Definition at line 833 of file AG2.for.

6.1.2.74 vbarst()

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

Definition at line 702 of file AG2.for.

6.1.2.75 vlablc()

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

Definition at line 2654 of file AG2.for.

6.1.2.76 width()

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

Definition at line 2702 of file AG2.for.

6.1.2.77 xden()

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

Definition at line 322 of file AG2.for.

6.1.2.78 xetyp()

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

Definition at line 606 of file AG2.for.

6.1.2.79 xfrm()

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

Definition at line 400 of file AG2.for.

6.1.2.80 xlab()

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

Definition at line 300 of file AG2.for.

6.1.2.81 xlen()

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

Definition at line 374 of file AG2.for.

6.1.2.82 xloc()

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

Definition at line 256 of file AG2.for.

6.1.2.83 xloctp()

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

Definition at line 278 of file AG2.for.

6.1.2.84 xmfrm()

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

Definition at line 448 of file AG2.for.

6.1.2.85 xmtcs()

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

Definition at line 426 of file AG2.for.

6.1.2.86 xneat()

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

Definition at line 212 of file AG2.for.

6.1.2.87 xtics()

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

Definition at line 352 of file AG2.for.

6.1.2.88 xtype()

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

Definition at line 554 of file AG2.for.

6.1.2.89 xwdth()

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

Definition at line 580 of file AG2.for.

6.1.2.90 xzero()

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

Definition at line 234 of file AG2.for.

6.1.2.91 yden()

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

Definition at line 337 of file AG2.for.

6.1.2.92 yetyp()

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

Definition at line 619 of file AG2.for.

6.1.2.93 yfrm()

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

Definition at line 413 of file AG2.for.

6.1.2.94 ylab()

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

Definition at line 311 of file AG2.for.

6.1.2.95 ylen()

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

Definition at line 387 of file AG2.for.

6.1.2.96 yloc()

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

Definition at line 267 of file AG2.for.

6.1.2.97 ylocrt()

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

Definition at line 289 of file AG2.for.

6.1.2.98 ymdyd()

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

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

Definition at line 1414 of file AG2.for.

6.1.2.99 ymfrm()

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

Definition at line 461 of file AG2.for.

6.1.2.100 ymtcs()

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

Definition at line 437 of file AG2.for.

6.1.2.101 yneat()

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

Definition at line 223 of file AG2.for.

6.1.2.102 ytics()

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

Definition at line 363 of file AG2.for.

6.1.2.103 ytype()

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

Definition at line 567 of file AG2.for.

6.1.2.104 ywdth()

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

Definition at line 593 of file AG2.for.

6.1.2.105 yzero()

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

Definition at line 245 of file AG2.for.

6.2 AG2.for

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.3 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

6.3.1 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.

6.3.2 Function/Subroutine Documentation

6.3.2.1 alfset()

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

Definition at line 45 of file AG2Holerith.for.

6.3.2.2 comdmp()

```
subroutine comdmp
```

Definition at line 328 of file AG2Holerith.for.

6.3.2.3 comget()

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

Definition at line 271 of file AG2Holerith.for.

6.3.2.4 comset()

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

Definition at line 299 of file AG2Holerith.for.

6.3.2.5 eform()

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

Definition at line 173 of file AG2Holerith.for.

6.3.2.6 expout()

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

Definition at line 90 of file AG2Holerith.for.

6.3.2.7 fform()

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

Definition at line 189 of file AG2Holerith.for.

6.3.2.8 fonly()

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

Definition at line 205 of file AG2Holerith.for.

6.3.2.9 hlabel()

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

Definition at line 121 of file AG2Holerith.for.

6.3.2.10 hstrin()

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

Definition at line 112 of file AG2Holerith.for.

6.3.2.11 ibasec()

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

Definition at line 241 of file AG2Holerith.for.

6.3.2.12 ibasex()

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

Definition at line 251 of file AG2Holerith.for.

6.3.2.13 ibasey()

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

Definition at line 261 of file AG2Holerith.for.

6.3.2.14 iform()

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

Definition at line 221 of file AG2Holerith.for.

6.3.2.15 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.

6.3.2.16 notate()

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

Definition at line 30 of file AG2Holerith.for.

6.3.2.17 numset()

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

Definition at line 67 of file AG2Holerith.for.

6.3.2.18 vlabel()

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

Definition at line 139 of file AG2Holerith.for.

6.3.2.19 vstrin()

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

Definition at line 130 of file AG2Holerith.for.

6.4 AG2Holerith.for

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

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

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

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

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

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

6.5 AG2uline.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine uline (x, y, i)

6.5.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2uline.for.

6.5.2 Function/Subroutine Documentation

6.5.2.1 uline()

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

Definition at line 10 of file AG2uline.for.

6.6 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
```

6.7 AG2umnmx.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine umnmx (array, amin, amax)

6.7.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2umnmx.for.

6.7.2 Function/Subroutine Documentation

6.7.2.1 umnmx()

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

Definition at line 9 of file AG2umnmx.for.

6.8 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
```

6.9 AG2upoint.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- real function upoint (arr, ii, oldone)

6.9.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2upoint.for.

6.9.2 Function/Subroutine Documentation

6.9.2.1 upoint()

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

Definition at line 9 of file AG2upoint.for.

6.10 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
```

6.11 AG2users.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine users (x, y, i)

6.11.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2users.for.

6.11.2 Function/Subroutine Documentation

6.11.2.1 users()

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

Definition at line 9 of file AG2users.for.

6.12 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
```

6.13 AG2useset.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

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

6.13.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2useset.for.

6.13.2 Function/Subroutine Documentation

6.13.2.1 useset()

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

Definition at line 9 of file AG2useset.for.

6.14 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
```

6.15 AG2usesetC.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

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

6.15.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2usesetC.for.

6.15.2 Function/Subroutine Documentation

6.15.2.1 usesetc()

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

Definition at line 9 of file AG2usesetC.for.

6.16 AG2usesetC.for

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

6.17 AG2UsrSoftek.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine softek (isym)

6.17.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2UsrSoftek.for.

6.17.2 Function/Subroutine Documentation

6.17.2.1 softek()

```
subroutine softek (
              isym )
```

Definition at line 9 of file AG2UsrSoftek.for.

6.18 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
```

6.19 CreateMainWindow.c File Reference

MS Windows Port: Init FTN77 Main

```
#include <windows.h>
#include <tchar.h>
#include "TCSdWINc.h"
```

Macros

- #define WIN32_LEAN_AND_MEAN
- #define WINMAIN_ICON _T("WinMainIcon")
- #define WINMAIN_DEFWINCLASS _T("WinMainFTN77")

Functions

- void CreateMainWindow_IfNecessary (HINSTANCE *hMainProgInst, HWND *hMainProgWindow, LPTSTR szWinName)

6.19.1 Detailed Description

MS Windows Port: Init FTN77 Main

Version

1.2

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Only if necessary: creates a main window

Note

The calling Fortranprogram has to allocate appropriate variables to receive pointers, q.v. TCSinitt.for

Definition in file CreateMainWindow.c.

6.19.2 Macro Definition Documentation

6.19.2.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```
Definition at line 25 of file CreateMainWindow.c.

6.19.2.2 WINMAIN_DEFWINCLASS

```
#define WINMAIN_DEFWINCLASS _T("WinMainFTN77")
```
Definition at line 36 of file CreateMainWindow.c.

6.19.2.3 WINMAIN_ICON

```
#define WINMAIN_ICON _T("WinMainIcon")
```
Definition at line 35 of file CreateMainWindow.c.

6.19.3 Function Documentation

6.19.3.1 CreateMainWindow_IfNecessary()

```
void CreateMainWindow_IfNecessary (
            HINSTANCE * hMainProgInst,
            HWND * hMainProgWindow,
            LPTSTR szWinName )
```
In case that the compiler has not created a window for the main program, this subroutine creates and shows a new main window. The class will be named according to the constant WINMAIN_DEFWINCLASS.
The window icon can be defined as WinMainIcon by a resource file.

Parameters

in	*hMainProgInst*	Main instance
in,out	*hMainProgWindow*	Main window
in	*szWinName*	Window name in case a main window does not exist

Definition at line 70 of file CreateMainWindow.c.

6.20 CreateMainWindow.c

```
00001 /** ****************************************************************************
00002 \file       CreateMainWindow.c
00003 \brief      MS Windows Port: Init FTN77 Main
00004 \version    1.2
00005 \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          Erzeugt nur bei Bedarf ein Fenster für das Hauptprogramm
00009 \note
00010          Die Pointervariablen muessen vom aufrufenden Fortranprogramm
00011          ausreichend groß dimensioniert werden, s. TCSinitt.for
00012 \~english
00013          Only if necessary: creates a main window
00014 \note
00015          The calling Fortranprogram has to allocate appropriate variables
00016          to receive pointers, q.v. TCSinitt.for
00017 \~
00018
00019 ******************************************************************************* */
00020
```

```
00021 #if defined(__WATCOMC__) && defined(__WINDOWS__)
00022  #define NULL 0           // nur win16: Ueberlagern #define NULL ( (void *) 0)
00023 #endif                    // aus aus stddef.h, string.h...
00024
00025 #define WIN32_LEAN_AND_MEAN
00026 #include <windows.h>
00027
00028 #include <tchar.h>
00029 #include "TCSdWINc.h"   // Unterstuetzung 16/32bit Kompatibilitaet
00030
00031 #if defined(__WATCOMC__) && defined(__SW_BW)
00032  #include <wdefwin.h>   // Compilerswitch -bw: Watcom Default Window System
00033 #endif
00034
00035 #define WINMAIN_ICON         _T("WinMainIcon")
00036 #define WINMAIN_DEFWINCLASS  _T("WinMainFTN77")
00037
00038 /** ****************************************************************************
00039
00040 \~german
00041 \brief Initialisierung der FTN77 Hauptprogramme
00042
00043   Unterprogramm zur Initialisierung von Windows. Erzeugt und zeigt(!) ein
00044   Fenster für das Hauptprogramm, falls noch keine Windows-Initialisierung
00045   anderweitig (z.B. durch den Compiler) vorgenommen wurde. Die Klasse wird
00046   entsprechend der Konstante WINMAIN_DEFWINCLASS benannt.
00047
00048   Das Icon kann über ein Resourcefile als WinMainIcon definiert werden.
00049
00050 \param[in] hMainProgInst Instanz des Hauptprogrammes
00051 \param[in,out] hMainProgWindow Fenster des Hauptprogrammes
00052 \param[in] szWinName Fenstername des evtl. erzeugten Fensters
00053 \~english
00054
00055   In case that the compiler has not created a window for the main program,
00056   this subroutine creates and shows a new main window. The class will be
00057   named according to the constant WINMAIN_DEFWINCLASS.
00058
00059   The window icon can be defined as WinMainIcon by a resource file.
00060
00061 \param[in] hMainProgInst Main instance
00062 \param[in,out] hMainProgWindow Main window
00063 \param[in] szWinName Window name in case a main window does not exist
00064 \~
00065
00066
00067 ******************************************************************************* */
00068
00069
00070 void CreateMainWindow_IfNecessary (HINSTANCE * hMainProgInst,
00071                                    HWND * hMainProgWindow, LPTSTR szWinName)
00072
00073 {
00074
00075 TCHAR           szClassName [] = WINMAIN_DEFWINCLASS; /* Class Name */
00076 static WNDCLASS wincl;       /* SAVE Data structure for the windowclass */
00077 #if defined(__WIN32__) || defined(_WIN32)
00078  DWORD           ErrorCode;
00079  LPVOID          lpMsgBuf;
00080 #endif
00081
00082
00083     if (*hMainProgWindow == NULL ) { // Hauptprogramm ohne (bekanntes) Fenster
00084
00085      /* Create MainWindow */
00086
00087      wincl.hInstance = *hMainProgInst;
00088      wincl.lpszClassName = szClassName;
00089      wincl.lpfnWndProc = DefWindowProc;      /* keine eigene Windowsroutine */
00090      wincl.style =  CS_DBLCLKS;              /* Catch double-clicks */
00091
00092      wincl.hIcon = LoadIcon (*hMainProgInst, WINMAIN_ICON);
00093      wincl.hCursor = NULL;
00094      wincl.lpszMenuName = NULL;    // No menu
00095      wincl.cbClsExtra = 0;         // No extra bytes after the window class
00096      wincl.cbWndExtra = 0;         // structure or the window instance
00097      wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;
00098
00099      /* Register the window class. Fail: most probable UNICODE on win98 */
00100      if (!RegisterClass (&wincl)) {
00101       #if defined(__WIN32__) || defined(_WIN32)
00102        ErrorCode= GetLastError(); // win32-Funktion
00103 //     if (ErrorCode == ERROR_CLASS_ALREADY_EXISTS) {
00104 //      Hier bei Bedarf Fehlerbehandlung einführen
00105 //     } else {
00106        FormatMessage(
00107         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
```

```
00108           NULL,
00109           ErrorCode,
00110           MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
00111           (LPTSTR) &lpMsgBuf,
00112           0,
00113           NULL
00114          );
00115          MessageBox (NULL, lpMsgBuf,_T("Error in CreateMainWindow"), MB_ICONSTOP);
00116          LocalFree( lpMsgBuf ); // Free the buffer
00117 //      } // Ende der Fehlerbehandlung
00118        #else // rudimentaere Fehlerbehandlung 16bit Windows
00119         MessageBox (NULL, _T("Window Class not registered"),
00120                           _T("Error in CreateMainWindow"), MB_ICONSTOP);
00121        #endif
00122        return;
00123       }
00124
00125       /* The class is registered, let's create the program */
00126       *hMainProgWindow = CreateWindow (
00127         szClassName,                 // Classname
00128         szWinName,                   // Title Text
00129         WS_POPUPWINDOW | WS_DISABLED, // disabled -> Prozessverwaisung verhindern
00130         CW_USEDEFAULT,               // Windows decides the position
00131         CW_USEDEFAULT,               // of the Window
00132         0,                           // The programs width
00133         0,                           // and height in pixels
00134         HWND_DESKTOP,                // Parent: desktop
00135         NULL,                        // No menu
00136         *hMainProgInst,              // Program Instance handler
00137         NULL                         // No Window Creation data
00138       );
00139       ShowWindow (*hMainProgWindow, SW_SHOW);
00140      } else {    // Mainwindow bereits vorhanden
00141       #if defined(__WATCOMC__) && defined(__SW_BW)
00142        _dwSetAppTitle (szWinName);    // Fenstername Watcom Default Window
00143       #endif
00144      }
00145 }
00146
```

6.21 G2dAG2.fd File Reference

Graph2D: AG2 Common Block G2dAG2.

6.21.1 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.

6.22 G2dAG2.fd

```
00001 C> \file       G2dAG2.fd
00002 C> \brief      Graph2D: AG2 Common Block G2dAG2
00003 C> \version    2.0
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C
00007 C  Da die folgende Definition kein Bestandteil eines Moduls
00008 C  ist versagt der DOXYGEN-Parser bei der Kombination von
00009 C  COMMON und integer. Workaraound: \\cond ... \\endcond
00010 C> \cond
00011
```

```
00012 C Common Block G2dAG2, Version 2.0 für AG2
00013 C     Die Funktion der Variablen entspricht dem Tektronix AG2 User-Manual,
00014 C     jedoch sind die achsenbezogenen Variablen in einem Feld zusammenge-
00015 C     fasst. Die x-Achse wird durch Index=1, y durch Index=2 beschrieben.
00016 C
00017       integer     cline,csymbl,csteps ! ibase+ 0..2
00018       real        cinfin ! 3
00019       integer     cnpts,cstepl,cnumbr ! 4..6
00020       real        csizes,csizel ! 7,8
00021
00022       logical     cxyneat(2),cxyzero(2) ! nbase+ 0, 1
00023       integer     cxyloc(2),cxylab(2),cxyden(2),cxytics(2) ! nbase+ 2..5
00024       integer     cxylen(2),cxyfrm(2),cxymtcs(2),cxymfrm(2),cxydec(2) ! 6..10
00025       real        cxydmin(2),cxydmax(2) ! 11,12
00026       integer     cxysmin(2),cxysmax(2),cxytype(2) ! 13..15
00027       integer     cxylsig(2),cxywdth(2),cxyepon(2) ! 16..18
00028       integer     cxystep(2),cxystag(2),cxyetyp(2) ! 19..21
00029       integer     cxybeg(2),cxyend(2),cxymbeg(2),cxymend(2) ! 22..25
00030       real        cxyamin(2),cxyamax(2) ! 26,27
00031
00032       common /g2dag2/
00033 C     & extent,cvectr,xvectr,yvectr,
00034 C     & xtentc,xtentx,xtenty,
00035 C
00036      & cline,csymbl,csteps,
00037      & cinfin,
00038      & cnpts,cstepl,cnumbr,csizes,csizel,
00039 C
00040      & cxyneat,cxyzero,cxyloc,cxylab,cxyden,cxytics,
00041      & cxylen,cxyfrm,cxymtcs,cxymfrm,cxydec,
00042      & cxydmin,cxydmax,cxysmin,cxysmax,cxytype,
00043      & cxylsig,cxywdth,cxyepon,cxystep,cxystag,cxyetyp,
00044      & cxybeg,cxyend,cxymbeg,cxymend,cxyamin,cxyamax
00045 C
00046 C     & reserv(8)
00047       save /g2dag2/
00048
00049       integer G2dAG2L         ! Benoetigt von SAVCOM, RESCOM
00050       parameter(g2dag2l=65)  ! integer, real und logical gleich lang!
00051 C> \endcond
```

6.23 GetHDC.for File Reference

Restore Hardcopies.

Functions/Subroutines

- logical function gethdc (Filnam)

6.23.1 Detailed Description

Restore Hardcopies.

Version

1.2

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.

6.23.2 Function/Subroutine Documentation

6.23.2.1 gethdc()

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

Parameters

FilNam	Hardcopyfie

Returns

(optional) .true. -> Error

Definition at line 15 of file GetHDC.for.

6.24 GetHDC.for

```
00001 C> \file       GetHDC.for
00002 C> \brief      Restore Hardcopies
00003 C> \version    1.2
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       include 'Tktrnx.fd'
00019       integer tcs_messagelen, iunit
00020       parameter(tcs_messagelen=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_messagelen) iprntlen= tcs_messagelen
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 if (action.eq.15) then ! XACTION_CLIP
00100           if (i1.eq.0) then ! clipping not active
00101             kminsx= 0
00102             kminsy= 0
00103             kmaxsx= 1023 ! TEK_XMAX
00104             kmaxsy= 780 ! TEK_YMAX
00105             call swind1(kminsx,kminsy,kmaxsx,kmaxsy) ! Set bool ClippingNotActive
00106           end if
00107         else if (action.eq.16) then ! XACTION_CLIP1
00108           kminsx= i1
00109           kminsy= i2
00110           call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
00111         else if (action.eq.17) then ! XACTION_CLIP2
00112           kmaxsx= i1
00113           kmaxsy= i2
00114           call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
00115         else ! unknown
00116           continue
00117         end if
00118       if (ios.eq.0) goto 10 ! until EOF
00119
00120       close (iunit)
00121       gethdc= .false.
00122       return
00123       end
```

6.25 GetMainInstance.c File Reference

MS Windows Port: Get Main Window and Instance.

```
#include <windows.h>
#include <tchar.h>
```

Macros

- #define WIN32_LEAN_AND_MEAN

Functions

- void GetMainInstAndWin (HINSTANCE *hMainProgInst, HWND *hMainProgWindow)

  Determination of instance and window of FTN77 main programs.

- void SaveMainInstAndWin (HINSTANCE *hMainProgInst, HWND *hMainProgWindow)

  Update the global variables containing instance and window of main.

6.25.1 Detailed Description

MS Windows Port: Get Main Window and Instance.

Version

1.5

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Get Instance and Window of the FTN77 Main Program
Definition in file GetMainInstance.c.

6.25.2 Macro Definition Documentation

6.25.2.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

Definition at line 22 of file GetMainInstance.c.

6.25.3 Function Documentation

6.25.3.1 GetMainInstAndWin()

```
void GetMainInstAndWin (
            HINSTANCE * hMainProgInst,
            HWND * hMainProgWindow )
```

Determination of instance and window of FTN77 main programs.
This routine has to be linked to the main program under all circumstances. In case of beeing part of a DLL, the instance handle of the DLL would be returned! The routine is fortran-callable.

Parameters

out	*hMainProgInst*	instance of main
out	*hMainProgWindow*	window of main

Definition at line 118 of file GetMainInstance.c.

6.25.3.2 SaveMainInstAndWin()

```
void SaveMainInstAndWin (
            HINSTANCE * hMainProgInst,
            HWND * hMainProgWindow )
```

Update the global variables containing instance and window of main.
Necessary after invoking CreateMainWindow_IfNecessary, where a new window handle could be created. The creation of a new window could be done by a DLL-based routine.

Parameters

in	*hMainProgInst*	instance of main
in	*hMainProgWindow*	window of main

Definition at line 182 of file GetMainInstance.c.

6.26 GetMainInstance.c

```
00001 /** ****************************************************************************
00002 \file       GetMainInstance.c
00003 \brief      MS Windows Port: Get Main Window and Instance
00004 \version    1.5
00005 \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          Ermittlung Instanz und Fenster der FTN77 Hauptprogramme
00009 \~english
00010          Get Instance and Window of the FTN77 Main Program
00011 \~
00012
00013 ***************************************************************************** */
00014
00015
00016 #if defined(__WATCOMC__) && defined(__WINDOWS__)
00017  #define NULL 0           // nur win16: Ueberlagern #define NULL ( (void *) 0)
00018 #endif                    // aus aus stddef.h, string.h...
00019
00020
00021
00022 #define WIN32_LEAN_AND_MEAN
00023 #include <windows.h>
00024 #include <tchar.h>
00025
00026
00027
00028 /*
00029 -------------------------- Externe Bezüge -----------------------------
00030 */
00031
00032 #ifdef __WATCOMC__      // Bis 11.0c: WATCOM Fortran Default Window System 10.0
00033  #if (__WATCOMC__ == 1100) // Source OpenWatcom 0.8, bld\clib\defwin\c bzw. \h
00034   extern HWND _MainWindow; // winglob.c, wmain.c, winmain.c, win.h
00035   #define EXTERN_WINDOW _MainWindow
00036   #undef EXTERN_INSTANCE
00037  #elif (__WATCOMC__ >= 1200)                // Open Watcom 1.0 bis 1.9:
00038   #if (!defined(__WIN32__) && !defined(_WIN32))         // 16bit-Windows
00039    #ifndef __SW_BW
00040     #error 16bit Windows requieres Default Window System, use the /bw switch
00041    #else
00042     extern HWND _MainWindow;     // Open Watcom Default Window System 1.0
00043     #define EXTERN_WINDOW _MainWindow
00044     #undef EXTERN_INSTANCE
00045    #endif
00046   #else              // 32bit-Windows: Default Window System deaktiviert
00047    #if defined (__SW_BW)
00048     #pragma message ("OpenWatcom >=1.0: Default Window System disabled!")
00049     #undefine __SW_BW
00050    #endif
00051    HWND _TCSMainWindow= NULL;
00052    #define EXTERN_WINDOW _TCSMainWindow
00053    #undef EXTERN_INSTANCE
00054   #endif
00055   #if (__WATCOMC__ > 1300)
00056    #pragma message ("New Compiler. Check if _MainWindow is defined")
00057    #pragma message (" (in bld\clib\defwin\c\winglob.c to compile for win16)")
00058    #pragma message (" Status V2.0 (__WATCOMC__ = 1300): unmodified since 3 years")
00059   #endif
00060  #else
00061   #pragma message ("Untested Compiler.") // Alte kommerzielle Compilerversionen
00062   HWND _TCSMainWindow= NULL;    // Ohne Default Window System?
00063   #define EXTERN_WINDOW _TCSMainWindow
00064   #undef EXTERN_INSTANCE
00065  #endif
00066  #pragma aux GetMainInstAndWin "^";     // fuer DLL: Fenster muss im Haupt-
00067  #pragma aux SaveMainInstAndWin "^";    // programm gespeichert werden
00068 #endif
00069
00070 #ifdef __GNUC__              // MinGW und GNU:
00071  #if __GNUC__<4 // bis GCC 4.0 Verwendung von g77, ab 4.0 gfortran
00072   extern HINSTANCE _MainInst; // Symbole werden durch das (selbstgeschriebene)
```

```
00073   extern HWND _MainWindow;    // WinMain.c erzeugt und belegt
00074  #else // gfortran: Init WinMain durch Constructor, nicht libfrtbegin
00075   static HINSTANCE _MainInst; // Falls von mehreren Bibliotheken(TekLib,ProcInp)
00076   static HWND _MainWindow;    // verwendet wird nur 1 Instanz gelinkt
00077  #endif
00078  #define EXTERN_INSTANCE _MainInst
00079  #define EXTERN_WINDOW _MainWindow
00080  #define GetMainInstAndWin getmaininstandwin_
00081  #define SaveMainInstAndWin savemaininstandwin_
00082 #endif
00083
00084 #ifdef _MSC_VER         // Microsoft Visual Cpp 6.0, ungeprueft da ohne FTN
00085  extern HINSTANCE hInst;
00086  #define EXTERN_INSTANCE hInst
00087  #define EXTERN_WINDOW HWND_DESKTOP
00088 #endif
00089
00090
00091
00092 /** ****************************************************************************
00093
00094  \~german
00095  \brief  Ermittlung Instanz und Fenster der FTN77 Hauptprogramme
00096
00097   Es muss in jedem Fall zu dem Hauptprogramm gelinkt werden und darf sich
00098   nicht in einer DLL befinden, da sonst die Instanz der DLL ermittelt wird!
00099   Das Unterprogramm ist von Fortran aufrufbar.
00100
00101   \param[out] hMainProgInst Instanz des Hauptprogrammes
00102   \param[out] hMainProgWindow Fenster des Hauptprogrammes
00103         Ermittlung Instanz und Fenster der FTN77 Hauptprogramme
00104  \~english
00105  \brief  Determination of instance and window of FTN77 main programs
00106
00107   This routine has to be linked to the main program under all circumstances.
00108   In case of beeing part of a DLL, the instance handle of the DLL would be returned!
00109   The routine is fortran-callable.
00110
00111   \param[out] hMainProgInst instance of main
00112   \param[out] hMainProgWindow window of main
00113  \~
00114
00115 ******************************************************************************** **/
00116
00117
00118 void GetMainInstAndWin (HINSTANCE * hMainProgInst, HWND * hMainProgWindow)
00119
00120 {
00121     #if defined EXTERN_WINDOW
00122      *hMainProgWindow= EXTERN_WINDOW;
00123     #else
00124      *hMainProgWindow= NULL;  // wird bei Bedarf spaeter erzeugt
00125     #endif
00126
00127     #if defined EXTERN_INSTANCE
00128      *hMainProgInst= EXTERN_INSTANCE;
00129     #else
00130      *hMainProgInst= NULL;
00131     #endif
00132
00133     if (*hMainProgInst == NULL) {
00134      #if defined EXTERN_WINDOW
00135       if (EXTERN_WINDOW != NULL ) { // Hauptprogramm besitzt (bekanntes) Fenster
00136        #if defined __WATCOMC__      // Watcom Default Window System 16/32 bit
00137         #if (!defined(__WIN32__) && !defined(_WIN32))
00138          *hMainProgInst= (HINSTANCE)GetWindowWord(EXTERN_WINDOW, GWW_HINSTANCE);
00139         #else                        // Watcom ohne 64bit Windows
00140          *hMainProgInst= (HINSTANCE)GetWindowLong(EXTERN_WINDOW, GWL_HINSTANCE);
00141         #endif
00142        #else                        // alle anderen Compiler ohne 16bit Windows
00143         #if (!defined(_WIN64))      // 32 bit
00144          *hMainProgInst= (HINSTANCE)GetWindowLong(EXTERN_WINDOW, GWL_HINSTANCE);
00145         #else                        // 64 bit
00146         *hMainProgInst= (HINSTANCE)GetWindowLongPtr(EXTERN_WINDOW, GWLP_HINSTANCE);
00147         #endif
00148        #endif
00149       } else { // kein offenes Fenster, z.B. Watcom-Consolenanwendung
00150        *hMainProgInst= GetModuleHandle (NULL);
00151       }
00152      #else      // kein Fenster ermittelbar
00153       *hMainProgInst= GetModuleHandle (NULL);
00154      #endif
00155     }
00156 }
00157
00158 /** ****************************************************************************
00159
```

```
00160  \~german
00161  \brief Aktualisierung globalen Speichervariablen Hauptinstanz und Hauptfenster.
00162
00163  Notwendig nach Aufruf von CreateMainWindow_IfNecessary, da dort evtl. ein neues
00164  Fensterhandle erzeugt wird. Da sich das Unterprogramm im Modul des Hauptprogrammes
00165  befindet, kann das Erzeugen des Fensters auch durch eine DLL erfolgen.
00166
00167  \param[in] hMainProgInst Instanzenhandle
00168  \param[in] hMainProgWindow Fensterhandle
00169  \~english
00170  \brief  Update the global variables containing instance and window of main
00171
00172   Necessary after invoking CreateMainWindow_IfNecessary, where a new window handle
00173   could be created. The creation of a new window could be done by a DLL-based routine.
00174
00175   \param[in] hMainProgInst instance of main
00176   \param[in] hMainProgWindow window of main
00177  \~
00178
00179 ********************************************************************************* **/
00180
00181
00182 void SaveMainInstAndWin (HINSTANCE * hMainProgInst, HWND * hMainProgWindow)
00183
00184 {
00185     #if defined EXTERN_INSTANCE
00186      EXTERN_INSTANCE= *hMainProgInst;
00187     #endif
00188
00189     #if defined EXTERN_WINDOW
00190      EXTERN_WINDOW= *hMainProgWindow;
00191     #endif
00192 }
```

6.27 Mainpage.dox File Reference

6.28 PlotHDC.for File Reference

Utility: Plot Journalfiles.

Functions/Subroutines

- program plothdc

6.28.1 Detailed Description

Utility: Plot Journalfiles.

Version

1.0-GCC

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Utility to draw journal-hardcopies from SDL2 and wX programs. With cut/paste they could be used by other MS-win programs. Program parameters are optained by calling gfortran extensions.

Note

```
Invoke by:
 $> plothdc FileName
```

Definition in file PlotHDC.for.

6.28.2 Function/Subroutine Documentation

6.28.2.1 plothdc()

```
program plothdc
```

Definition at line 26 of file PlotHDC.for.

6.29 PlotHDC.for

```
00001 C> \file       PlotHDC.for
00002 C> \brief      Utility: Plot Journalfiles
00003 C> \version    1.0-GCC
00004 C> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C> Hilfsprogramm zur Anzeige von Journal-Hardcopies von SDL2 und wX-Programmen.
00009 C> Diese koennen dann ueber Cut/Paste in andere Windowsprogramme uebernommen werden.
00010 C> Die Abfrage der Programmparameter erfolgt durch gfortran spezifische Erweiterungen.
00011 C> \note \verbatim
00012 C>    Aufruf durch:
00013 C>     $> plothdc FileName
00014 C> \endverbatim
00015 C>
00016 C> \~english
00017 C> Utility to draw journal-hardcopies from SDL2 and wX programs.
00018 C> With cut/paste they could be used by other MS-win programs.
00019 C> Program parameters are optained by calling gfortran extensions.
00020 C> \note \verbatim
00021 C>    Invoke by:
00022 C>     $> plothdc FileName
00023 C> \endverbatim
00024 C> \~
00025 C>
00026       program plothdc
00027       implicit none
00028       integer itrimlen
00029       integer ipar
00030       character * 128 filnam
00031
00032       call initt (0)
00033       ipar = iargc() ! Version for GCC compiler
00034       call getarg(1,filnam)
00035
00036       if (ipar.gt.0) then
00037         call gethdc (filnam(1:itrimlen(filnam))//char(0))
00038       else
00039         call graphicerror (9, 'Please invoke by: PlotHDC FileName')
00040       end if
00041       call finitt
00042       end
```

6.30 Strings.for File Reference

TCS: String functions.

Functions/Subroutines

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

6.30.1 Detailed Description

TCS: String functions.

Version

1.26

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Fortran utility functions for string processing
Definition in file Strings.for.

6.30.2 Function/Subroutine Documentation

6.30.2.1 istringlen()

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

Definition at line 94 of file Strings.for.

6.30.2.2 itrimlen()

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

Definition at line 133 of file Strings.for.

6.30.2.3 printstring()

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

Definition at line 114 of file Strings.for.

6.30.2.4 substitute()

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

Definition at line 30 of file Strings.for.

6.31 Strings.for

```
00001 C> \file       Strings.for
00002 C> \brief      TCS: String functions
00003 C> \version    1.26
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Hilfsfunktionen zur Fortran Stringverarbeitung
00008 C> \~english
00009 C> Fortran utility functions for string processing
00010 C> \~
00011 C>
00012 C
00013 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
00014 C
00015 C  Unterprogramme zur Behandlung von Fortran-Strings.
00016 C  Die Stringenden werden entweder durch CHAR(0) markiert oder
00017 C  ueber die Deklaration ermittelt.
00018 C
00019 C     9.11.88     K. Friedewald
00020 C
00021 C  Ergaenzungen:
00022 C     iTrimLen
00023 C
00024 C     7.12.01     K. Friedewald
00025 C
00026 C  Version: 1.26
00027 C
00028 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
00029
00030       subroutine substitute (Source, Destination, Old1, New1)
00031 C
00032 C  Durchsucht SOURCE nach den Substrings OLD, ersetzt sie durch NEW
00033 C  und uebergibt das Ergebniss in DESTINATION. Wenn New=CHAR(0), werden
00034 C  die vorkommenden OLD nur geloescht.
00035 C
00036 C  Stringenden koennen durch CHAR(0) markiert werden.
00037 C
00038       implicit none
00039       integer iNext, iNext2, TempLen
00040       integer iStringLen
00041       character *(*) Source, Destination, Old1, New1
00042       character*255 temp, old, new
00043
00044       if (istringlen(old1).le.0) return
00045       if (istringlen(source) .le. 0) then
00046        destination= char(0)
00047        return
00048       end if
00049
00050       old= old1 // char(0)          ! old evtl. = Destination
00051       new= new1 // char(0)          ! => retten!
00052
00053       temp= source(1:istringlen(source)) // char(0) ! evtl. Ueberlappung!
00054       destination= temp
00055       inext= index( destination(:istringlen(destination)),
00056      1                                       old(:istringlen(old)) )
00057       do while (inext.gt.0)
00058        if (inext.eq.1) then
00059         temp= destination
00060         if (new.eq.char(0)) then
00061          destination= temp(istringlen(old)+1:)
00062         else
00063          destination= new(:istringlen(new)) // temp(istringlen(old)+1:)
00064         end if
00065        else
00066         temp= destination(1:inext-1)
00067         templen= inext-1
00068         if (new.ne.char(0)) then
00069          temp= temp(1:templen)//new
00070          templen= templen+istringlen(new)
00071         end if
00072         if (inext+istringlen(old).lt.len(destination)) then
00073          temp= temp(1:templen)//destination(inext+istringlen(old):)
00074         end if
00075         destination= temp
00076        end if
00077        inext2= inext+istringlen(new)
00078        if (inext2.lt.len(destination)) then
00079         inext2= index(destination(inext2:), old(:istringlen(old)) )
00080        else
00081         inext2=0
00082        end if
00083        if (inext2.gt.0) then
00084         inext= inext+istringlen(new)+inext2-1
00085        else
00086         inext=0
00087        end if
00088       end do
00089       return
00090       end
00091
00092
00093
00094       function istringlen (String)
00095 C
00096 C Ermittelt die Stringlänge bei durch char(0) abgeschlossenen STRINGs.
00097 C Falls kein char(0) vorhanden ist, wird die Gesamtlänge übergeben.
00098 C
00099       implicit none
00100       character *(*) string
```

```
00101       integer istringlen, i
00102
00103       i= index(string,char(0))-1
00104       if (i.ge.0) then
00105        istringlen=i
00106       else
00107        istringlen= len(string)
00108       end if
00109       return
00110       end
00111
00112
00113
00114       character*(*) function printstring (String)
00115 C
00116 C  Kopiert STRING in einen variabel langen PRINTSTRING. Hierdurch wird
00117 C  der Ausdruck von Nullstrings (Fortran-Fehler!) vermieden.
00118 C
00119       implicit none
00120       character string *(*)
00121       integer istringlen
00122
00123       if (istringlen(string).gt.0) then
00124        printstring= string(1:istringlen(string))
00125       else
00126        printstring= ' '
00127       end if
00128       return
00129       end
00130
00131
00132
00133       integer function itrimlen (string)
00134 C
00135 C  Bestimmt die Länge des Strings ohne angehängte Leerzeichen.
00136 C  Bei Bedarf wird ein Char(0) angehaengt. Es darf in Ftn77 nie ein
00137 C  Nullstring erzeugt werden, da sonst die RTL-Library abstuerzt. Deswegen
00138 C  ist der kleinste erzeugte String ein Blank ' '.
00139 C
00140       implicit none
00141       character *(*) string
00142       integer i, istringlen
00143
00144       i=istringlen(string) +1
00145
00146  10   continue
00147        i= i-1
00148        if (i.ge.1) then
00149         if (string(i:i).eq.' ') goto 10
00150       end if
00151       itrimlen=i
00152       if ((i.lt.len(string)).and.(len(string).gt.1)) then
00153        string(i+1:i+1)= char(0) ! .gt.1: Achtung, nie Nullstring erzeugen!
00154       end if
00155       return
00156       end
00157
```

6.32 TCS.for File Reference

TCS: Tektronix Plot 10 Emulation.

Functions/Subroutines

- subroutine vcursr (IC, X, Y)
- subroutine drawr (X, Y)
- subroutine mover (X, Y)
- subroutine pointr (X, Y)
- subroutine dashr (X, Y, iL)
- subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
- subroutine drawa (X, Y)
- subroutine movea (X, Y)
- subroutine pointa (X, Y)
- subroutine dasha (X, Y, iL)
- subroutine wincot (X, Y, IX, IY)

- subroutine revcot (IX, IY, X, Y)
- subroutine anstr (NChar, IStrin)
- subroutine ancho (ichar)
- subroutine newlin
- subroutine cartn
- subroutine linef
- subroutine baksp
- subroutine newpag
- function linhgt (Numlin)
- function linwdt (NumChr)
- subroutine lintrn
- subroutine logtrn (IMODE)
- subroutine twindo (IX1, IX2, IY1, IY2)
- subroutine swindo (IX, LX, IY, LY)
- subroutine dwindo (X1, X2, Y1, Y2)
- subroutine vwindo (X, XL, Y, YL)
- subroutine rescal
- subroutine rrotat (Grad)
- subroutine rscale (Faktor)
- subroutine home
- subroutine setmrg (Mlinks, Mrecht)
- subroutine seetrm (IBaud, Iterm, ICSize, MaxScr)
- subroutine seetrn (xf, yf, key)
- logical function genflg (ITEM)

6.32.1 Detailed Description

TCS: Tektronix Plot 10 Emulation.

Version

4.1

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

System independent subroutines
Definition in file TCS.for.

6.32.2 Function/Subroutine Documentation

6.32.2.1 ancho()

```
subroutine ancho (
              ichar )
```

Definition at line 339 of file TCS.for.

6.32.2.2 anstr()

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

Definition at line 329 of file TCS.for.

6.32.2.3 baksp()

```
subroutine baksp
```

Definition at line 384 of file TCS.for.

6.32.2.4 cartn()

```
subroutine cartn
```

Definition at line 365 of file TCS.for.

6.32.2.5 dasha()

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

Definition at line 290 of file TCS.for.

6.32.2.6 dashr()

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

Definition at line 236 of file TCS.for.

6.32.2.7 drawa()

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

Definition at line 257 of file TCS.for.

6.32.2.8 drawr()

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

Definition at line 212 of file TCS.for.

6.32.2.9 dwindo()

```
subroutine dwindo (
              X1,
              X2,
```

```
                Y1,
                Y2 )
```

Definition at line 462 of file TCS.for.

6.32.2.10 genflg()

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

Definition at line 558 of file TCS.for.

6.32.2.11 home()

```
subroutine home
```

Definition at line 518 of file TCS.for.

6.32.2.12 linef()

```
subroutine linef
```

Definition at line 374 of file TCS.for.

6.32.2.13 linhgt()

```
function linhgt (
                Numlin )
```

Definition at line 400 of file TCS.for.

6.32.2.14 lintrn()

```
subroutine lintrn
```

Definition at line 418 of file TCS.for.

6.32.2.15 linwdt()

```
function linwdt (
                NumChr )
```

Definition at line 408 of file TCS.for.

6.32.2.16 logtrn()

```
subroutine logtrn (
                IMODE )
```

Definition at line 428 of file TCS.for.

6.32.2.17 movea()

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

Definition at line 268 of file TCS.for.

6.32.2.18 mover()

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

Definition at line 220 of file TCS.for.

6.32.2.19 newlin()

```
subroutine newlin
```

Definition at line 357 of file TCS.for.

6.32.2.20 newpag()

```
subroutine newpag
```

Definition at line 392 of file TCS.for.

6.32.2.21 pointa()

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

Definition at line 279 of file TCS.for.

6.32.2.22 pointr()

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

Definition at line 228 of file TCS.for.

6.32.2.23 rel2ab()

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

Definition at line 244 of file TCS.for.

6.32.2.24 rescal()

```
subroutine rescal
```

Definition at line 481 of file TCS.for.

6.32.2.25 revcot()

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

Definition at line 314 of file TCS.for.

6.32.2.26 rrotat()

```
subroutine rrotat (
              Grad )
```

Definition at line 501 of file TCS.for.

6.32.2.27 rscale()

```
subroutine rscale (
              Faktor )
```

Definition at line 510 of file TCS.for.

6.32.2.28 seetrm()

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

Definition at line 536 of file TCS.for.

6.32.2.29 seetrn()

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

Definition at line 547 of file TCS.for.

6.32.2.30 setmrg()

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

Definition at line 527 of file TCS.for.

6.32.2.31 swindo()

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

Definition at line 450 of file TCS.for.

6.32.2.32 twindo()

```
subroutine twindo (
              IX1,
              IX2,
```

```
              IY1,
              IY2 )
```

Definition at line 443 of file TCS.for.

6.32.2.33 vcursr()

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

Definition at line 202 of file TCS.for.

6.32.2.34 vwindo()

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

Definition at line 469 of file TCS.for.

6.32.2.35 wincot()

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

Definition at line 301 of file TCS.for.

6.33 TCS.for

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

```
C                - DeleteFont -> DeleteObject
C
C      27.03.13 Version 3.0
C               Anpassung an Windows 7 und OpenWatcom 1.9
C               Anpassung an gfortran anstelle von g77 der GCC
C
C      22.12.05 Version 2.19
C               Elimination berechnetes GOTO in LOGTRN
C
C      18.10.05 Version 2.18
C               Anpassung der Windowsversionen zur gemeinsamen Verwendung SDL2:
C                 TCSdrWIN.for
C                 TCSdWINc.h
C                 - Überfuehrung der Deklaration aus TCSdWIN.c nach *.h:
C                   GraphicError und CreateMainWindow_IfNecessary
C                 - Definition der Fehlernummern als Konstante statt enum
C               Abhaengigkeit Watcom-Defaultwindowsystem eliminiert
C               - TCSdWINc.c: Kein Abbruch bei OpenWatcom > 1.3 und
C                 definiertem Symbol trace_calls
C
C      26.10.04 Version 2.17
C               Bugfix Windows-System: Größe und Defaultposition des Status-
C                fensters wird bei der Erzeugung berechnet -> 1. RESTORE nach
C                Verkleinern des Graphikfensters entspricht dem vorherigen
C                Bild. 2. Angleichung des Verhaltens von 16- und 32bit Windows
C               Bei Definition des Symbols STAT_WINDOW_PRIVATE erhält das
C                Statusfenster einen privaten Devicekontext.
C               Zusammenfuehrung Initialisierung der Windows-Library und
C                Windows-DLL -> zusaetzliche Sourcefiles
C                TCSinitt.for, CreateMainWindow.c, GetMainInstance.c
C
C      23.06.04 Version 2.16:
C               Anpassungen an GNU-Compiler fuer Win32. Zusätzliches Sourcefile
C                fuer die GNU-Version: WinMain.c
C               CSIZE in Windows-Version: Korrektur Rundungsfehler
C
C      08.06.04 Version 2.15:
C               Umbenennung lib$movc3 in lib_movc3 (entsprechend ANSI-Fortran)
C               Modul STRINGS.FOR: Version 1.24
C
C      27.06.03 Version 2.14:
C               Verarbeitung Steuerzeichen in ANCHO
C
C      21.10.02 Version 2.13:
C               Einheitliche Version CPM/DOS/Windows
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Grundversion fuer C128 / Version 1.0:
C
C      Zugehoerige Module:
C              TKTRNX.FOR    Common-Block TKTRNX
C              TCSBASIC.ASM  Low-Level Routinen in Bank 0, C128 spezifisch
C              TCSDRIVR.ASM  Treiber fuer TCSBASIC
C              TCSGIN.ASM    Treiber des Gin-Cursors
C
C      20.4.88         Dr.-Ing. K. Friedewald
C                      4000 Duesseldorf 1
C                      Gerresheimerstr. 84
C
C      21.10.02 Version 2.13:
C               Vereinheitlichung CPM/DOS/Windowsversion
C               Zusätzliches Modul: TCSdrCPM.FOR: früher Teil von TCS.FOR
C               Ausschließliche Verwendung von durch grosses "C" eingeleiteten
C                Kommentaren zur Kompatibilität mit FORTRAN 4
C               Umbenennung des Includefiles in Tktrnx.fd. So kann unter CP/M
C                das als Teil des Filenamens interpretierte "'" der INCLUDE-
C                Anweisung entsprechend der 8.3 Filenamen umgesetzt werden.
C               Implementierung Unterprogramm TCSLEV
C               Bugfix: Kommentar in Tktrnx.fd wurde falsch gekennzeichnet
C                       (c statt C) -> SVSTAT und RESTAT fehlerhaft, da nicht
C                       erkannte Kommentare zusaetzliche Variablen erzeugten.
C
C      TBD: Implementierung vertikale Auflösung von 400 Pixeln
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassung an DOS:
C
C      Aenderungen gegenueber CP/M-Version:
C               SEELOC, DCURSR, SVSTAT, RESTAT, CSIZE in TCSdrDOS.FOR
C      Bugfix:  DASHA, DASHR - Korrektur Parameterliste
C               SEETRM - ibaud statt ibaudr
C
C      Zugehoerige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TCSdrDOS.FOR  Bildschirmtreiber
```

```
C               TCSdDOSa.ASM  Betriebssystemspezifische Low-Level Routinen
C               HDCOPY.FOR    Hardcopyroutine
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C               OUTTEXT.FOR   nur für WATCOM-Compiler
C
C      25.10.01 Version 2.00:  Dr.-Ing. K. Friedewald
C
C      07.02.02 Version 2.10:
C               Implementierung multilinguale Fehlermeldungen
C
C      11.10.02 Version 2.12:
C               Vereinheitlichung DOS/Windowsversion
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an Microsoft-Windows:
C
C      Aenderungen gegenueber DOS-Version:
C               INITT befinden sich jetzt in TCSdrWIN.FOR bzw. TCSinitt.FOR
C
C      Zugehoerige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TKTRNX.h      Common-Block TKTRNX für Zugriff durch C
C               TCSdrWIN.FOR  Bildschirmtreiber
C               TCSdWINc.c    Windowspezifische API-Routinen
C               TCSdWINc.h    Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C
C      27.10.01 Version 2.11: Dr.-Ing. K. Friedewald
C
C      11.10.02 Version 2.12:
C               Vereinheitlichung DOS/Windowsversion
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an SDL2:
C
C      Aenderungen gegenueber Windows-Version:
C               Fehlerausgabe in den Windows-Debug-Channel (bzw. *ix Fehlerkanal)
C               Statusfenster analog DOS nur einzeilig ohne Scrollmöglichkeit
C
C      Zugehoerige Module:
C               TKTRNX.FOR    identisch mit Windows-Version
C               TKTRNX.h      identisch mit Windows-Version
C               TCSdrSDL.FOR  SDL2-spezifische API-Routinen
C               TCSdSDLc.c    SDL2-spezifische API-Routinen
C               TCSdSDLc.h    Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR   identisch mit Windows-Version
C
C      27.11.20 Version 4.00: Dr.-Ing. K. Friedewald
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an WXwidgets:
C
C      Aenderungen gegenueber SDL2-Version:
C               Fehlerausgabe in den wxLogStatus
C               Statusfenster durch initt1() konfigurierbar
C
C      Zugehoerige Module:
C               TKTRNX.FOR     identisch mit Windows-Version
C               TKTRNX.hpp     identisch mit Windows-Version
C               TCSdrWXfor.f08 WX-spezifische API-Routinen
C               TCSdrWXcpp.cpp WX-spezifische API-Routinen
C               TCSdrWXcpp.hpp Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR    identisch mit Windows-Version
C               Graph2D.f08    Interfacemodul Anwenderprogramme ab Fortran 2003
C               graph2d.h      Header fuer C/Cpp Anwenderprogramme
C
C      26.07.23 Version 5.00: Dr.-Ing. K. Friedewald
C



C
C Graphic Input
C

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

C
C  Virtuelle Graphik, relativ
```

```
00210 C
00211
00212       subroutine drawr (X,Y)
00213       call rel2ab (x,y,xabs,yabs)
00214       call drawa (xabs,yabs)
00215       return
00216       end
00217
00218
00219
00220       subroutine mover (X,Y)
00221       call rel2ab (x,y,xabs,yabs)
00222       call movea (xabs,yabs)
00223       return
00224       end
00225
00226
00227
00228       subroutine pointr (X,Y)
00229       call rel2ab (x,y,xabs,yabs)
00230       call pointa (xabs,yabs)
00231       return
00232       end
00233
00234
00235
00236       subroutine dashr (X,Y, iL)
00237       call rel2ab (x,y,xabs,yabs)
00238       call dasha (xabs,yabs, il)
00239       return
00240       end
00241
00242
00243
00244       subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
00245       include 'Tktrnx.fd'
00246       call seeloc (ix,iy)
00247       call revcot (ix,iy,xabs,yabs)
00248       xabs= (( xrel*trcosf - yrel*trsinf)*trscal)+xabs
00249       yabs= (( xrel*trsinf + yrel*trcosf)*trscal)+yabs
00250       return
00251       end
00252
00253 C
00254 C  Virtuelles Zeichnen, absolut
00255 C
00256
00257       subroutine drawa (X,Y)
00258       include 'Tktrnx.fd'
00259       call wincot (x,y,ix,iy)
00260       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00261       call drwabs (ix,iy)
00262       call swind1 (0,0,1023,780)
00263       return
00264       end
00265
00266
00267
00268       subroutine movea (X,Y)
00269       include 'Tktrnx.fd'
00270       call wincot (x,y,ix,iy)
00271       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00272       call movabs (ix,iy)
00273       call swind1 (0,0,1023,780)
00274       return
00275       end
00276
00277
00278
00279       subroutine pointa (X,Y)
00280       include 'Tktrnx.fd'
00281       call wincot (x,y,ix,iy)
00282       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00283       call pntabs (ix,iy)
00284       call swind1 (0,0,1023,780)
00285       return
00286       end
00287
00288
00289
00290       subroutine dasha (X,Y, iL)
00291       include 'Tktrnx.fd'
00292       call wincot (x,y,ix,iy)
00293       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00294       call dshabs (ix,iy, il)
00295       call swind1 (0,0,1023,780)
00296       return
```

```
00297       end
00298
00299
00300
00301       subroutine wincot (X,Y,IX,IY)
00302       include 'Tktrnx.fd'
00303       dx= x-tminvx
00304       dy= y-tminvy
00305       if ((xlog.lt.255.).and.(x.gt.0.)) dx= alog(x)-xlog
00306       if ((ylog.lt.255.).and.(y.gt.0.)) dy= alog(y)-ylog
00307       ix= ifix(dx*xfac+.5)+kminsx
00308       iy= ifix(dy*yfac+.5)+kminsy
00309       return
00310       end
00311
00312
00313
00314       subroutine revcot (IX,IY,X,Y)
00315       include 'Tktrnx.fd'
00316       dx= float(ix-kminsx) / xfac
00317       dy= float(iy-kminsy) / yfac
00318       x= dx + tminvx
00319       y= dy + tminvy
00320       if (xlog.lt.255.) x= 2.718282**(dx+xlog)
00321       if (ylog.lt.255.) y= 2.718282**(dy+ylog)
00322       return
00323       end
00324
00325 C
00326 C  Alphanumerische Ausgabe
00327 C
00328
00329       subroutine anstr (NChar, IStrin)
00330       dimension istrin(1)
00331       do 10 i=1,nchar
00332        call ancho (istrin(i))
00333 10    continue
00334       return
00335       end
00336
00337
00338
00339       subroutine ancho (ichar)
00340       include 'Tktrnx.fd'
00341
00342       if (ichar.gt.31) goto 10
00343       if (ichar.eq.7) call bell
00344       if (ichar.eq.10) call linef
00345       if (ichar.eq.13) call cartn
00346       return
00347
00348  10   call seeloc (ix,k)
00349       call csize (ixlen,k)
00350       if (ix.gt.krmrgn-ixlen) call newlin
00351       call toutpt (ichar)
00352       return
00353       end
00354
00355
00356
00357       subroutine newlin
00358       call cartn
00359       call linef
00360       return
00361       end
00362
00363
00364
00365       subroutine cartn
00366       include 'Tktrnx.fd'
00367       call seeloc (ix,iy)
00368       call movabs (klmrgn,iy)
00369       return
00370       end
00371
00372
00373
00374       subroutine linef
00375       call seeloc (j,iy)
00376       call csize (j,iylen)
00377       if (iy.lt.iylen) call home
00378       call movrel (0,-iylen)
00379       return
00380       end
00381
00382
00383
```

```
00384       subroutine baksp
00385       call csize (ix,iy)
00386       call movrel (-ix,0)
00387       return
00388       end
00389
00390
00391
00392       subroutine newpag
00393       call erase
00394       call home
00395       return
00396       end
00397
00398
00399
00400       function linhgt (Numlin)
00401       call csize (ix,iy)
00402       linhgt= numlin*iy
00403       return
00404       end
00405
00406
00407
00408       function linwdt (NumChr)
00409       call csize (ix,iy)
00410       linwdt= numchr*ix
00411       return
00412       end
00413
00414 C
00415 C  Initialisierungsroutinen
00416 C
00417
00418       subroutine lintrn
00419       include 'Tktrnx.fd'
00420       xlog= 255.
00421       ylog= 255.
00422       call rescal
00423       return
00424       end
00425
00426
00427
00428       subroutine logtrn (IMODE)
00429       include 'Tktrnx.fd'
00430       call lintrn
00431       if ((imode .eq. 1) .or. (imode .eq. 3)) then
00432        xlog= 0.
00433       end if
00434       if ((imode .eq. 2) .or. (imode .eq. 3)) then
00435        ylog= 0.
00436       end if
00437       call rescal
00438       return
00439       end
00440
00441
00442
00443       subroutine twindo (IX1,IX2,IY1,IY2)
00444       call swindo (ix1,ix2-ix1,iy1,iy2-iy1)
00445       return
00446       end
00447
00448
00449
00450       subroutine swindo (IX,LX,IY,LY)
00451       include 'Tktrnx.fd'
00452       kminsx= ix
00453       kmaxsx= ix+lx
00454       kminsy= iy
00455       kmaxsy= iy+ly
00456       call rescal
00457       return
00458       end
00459
00460
00461
00462       subroutine dwindo (X1,X2,Y1,Y2)
00463       call vwindo (x1,x2-x1,y1,y2-y1)
00464       return
00465       end
00466
00467
00468
00469       subroutine vwindo (X,XL,Y,YL)
00470       include 'Tktrnx.fd'
```

```
00471       tminvx= x
00472       tmaxvx= x+xl
00473       tminvy= y
00474       tmaxvy= y+yl
00475       call rescal
00476       return
00477       end
00478
00479
00480
00481       subroutine rescal
00482       include 'Tktrnx.fd'
00483       xfac= 0.
00484       yfac= 0.
00485       if ((tmaxvx.eq.tminvx) .or. (tmaxvy.eq.tminvy)) return
00486       dx= tmaxvx-tminvx
00487       dy= tmaxvy-tminvy
00488       if ((xlog.eq.255.).or.(amin1(tminvx,tmaxvx).le.0.)) goto 10
00489        xlog= alog(tminvx)
00490        dx= alog(tmaxvx)-xlog
00491 10    if ((ylog.eq.255.).or.(amin1(tminvy,tmaxvy).le.0.)) goto 20
00492        ylog= alog(tminvy)
00493        dy= alog(tmaxvy)-ylog
00494 20    xfac= float(kmaxsx-kminsx) / dx
00495       yfac= float(kmaxsy-kminsy) / dy
00496       return
00497       end
00498
00499
00500
00501       subroutine rrotat (Grad)
00502       include 'Tktrnx.fd'
00503       trsinf= sin(grad/57.29578)
00504       trcosf= cos(grad/57.29578)
00505       return
00506       end
00507
00508
00509
00510       subroutine rscale (Faktor)
00511       include 'Tktrnx.fd'
00512       trscal= faktor
00513       return
00514       end
00515
00516
00517
00518       subroutine home
00519       include 'Tktrnx.fd'
00520 C      call movabs(klmrgn,750) Fuer CP/M (kein khomey verfuegbar, -> !=750)
00521       call movabs(klmrgn,khomey)
00522       return
00523       end
00524
00525
00526
00527       subroutine setmrg (Mlinks, Mrecht)
00528       include 'Tktrnx.fd'
00529       klmrgn= mlinks
00530       krmrgn= mrecht
00531       return
00532       end
00533
00534
00535
00536       subroutine seetrm (IBaud,Iterm,ICSize,MaxScr)
00537       include 'Tktrnx.fd'
00538       ibaud= 0
00539       iterm= 1
00540       icsize= 1
00541       maxscr= 1023
00542       return
00543       end
00544
00545
00546
00547       subroutine seetrn (xf,yf,key)
00548       include 'Tktrnx.fd'
00549       xf= xfac
00550       yf= yfac
00551       key= 1
00552       if ((xlog.lt.255.).or.(ylog.lt.255.)) key=2
00553       return
00554       end
00555
00556
00557
```

```
00558       logical function genflg (ITEM)
00559       genflg= item.eq.0
00560       return
00561       end
```

6.34 TCSdrWIN.for File Reference

MS Windows Port: High-Level Driver.

Functions/Subroutines

- subroutine tcslev (LEVEL)
- subroutine svstat (Array)
- subroutine restat (Array)
- subroutine movrel (iX, iY)
- subroutine pntrel (iX, iY)
- subroutine drwrel (iX, iY)
- subroutine dshrel (iX, iY, iMask)
- subroutine seeloc (IX, IY)
- subroutine toutpt (iChr)
- subroutine toutst (nChr, iChrArr)
- subroutine toutstc (String)
- subroutine statst (String)
- subroutine anmode

  Entry Dummyroutinen.
- logical function winselect (iDummy)

6.34.1 Detailed Description

MS Windows Port: High-Level Driver.

Version

(2022, 88,x)

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

MS Windows specific subroutines

Note

```
Supplement to Tektronix:
 subroutine TOUTSTC (String): Print Fortran-String
 subroutine LINCOL (iCol): Set line color (iCol=0..15)
 subroutine TXTCOL (iCol): Set text color
 subroutine BCKCOL (iCol): Set background color (shows after ERASE)
 subroutine DefaultColour: Reset default colors
```

Definition in file TCSdrWIN.for.

6.34.2 Function/Subroutine Documentation

6.34.2.1 anmode()

```
subroutine anmode
```

Entry Dummyroutinen.
AlfMod
pClipt
ioWait
alpha
Definition at line 269 of file TCSdrWIN.for.

6.34.2.2 drwrel()

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

Definition at line 191 of file TCSdrWIN.for.

6.34.2.3 dshrel()

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

Definition at line 201 of file TCSdrWIN.for.

6.34.2.4 movrel()

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

Definition at line 171 of file TCSdrWIN.for.

6.34.2.5 pntrel()

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

Definition at line 181 of file TCSdrWIN.for.

6.34.2.6 restat()

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

Definition at line 153 of file TCSdrWIN.for.

6.34.2.7 seeloc()

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

Definition at line 213 of file TCSdrWIN.for.

6.34.2.8 statst()

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

Definition at line 255 of file TCSdrWIN.for.

6.34.2.9 svstat()

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

Definition at line 140 of file TCSdrWIN.for.

6.34.2.10 tcslev()

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

Definition at line 123 of file TCSdrWIN.for.

6.34.2.11 toutpt()

```
subroutine toutpt (
              iChr )
```

Definition at line 228 of file TCSdrWIN.for.

6.34.2.12 toutst()

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

Definition at line 236 of file TCSdrWIN.for.

6.34.2.13 toutstc()

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

Definition at line 247 of file TCSdrWIN.for.

6.34.2.14 winselect()

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

Definition at line 283 of file TCSdrWIN.for.

6.35 TCSdrWIN.for

```
00001 C> \file       TCSdrWIN.for
00002 C> \brief      MS Windows Port: High-Level Driver
00003 C> \version    (2022, 88,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> MS Windows-spezifische TCS-Routinen
00009 C> \note \verbatim
00010 C>    Erweiterungen gegenüber Tektronix:
00011 C>     subroutine TOUTSTC (String): Ausgabe Fortran-String
00012 C>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
```

```
00013 C>     subroutine TXTCOL (iCol): Setzen Textfarbe
00014 C>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00015 C>     subroutine DefaultColour: Wiederherstellung Defaultfarben
00016 C> \endverbatim
00017 C>
00018 C>
00019 C> \~english
00020 C> MS Windows specific subroutines
00021 C> \note \verbatim
00022 C>    Supplement to Tektronix:
00023 C>     subroutine TOUTSTC (String): Print Fortran-String
00024 C>     subroutine LINCOL (iCol): Set line color (iCol=0..15)
00025 C>     subroutine TXTCOL (iCol): Set text color
00026 C>     subroutine BCKCOL (iCol): Set background color (shows after ERASE)
00027 C>     subroutine DefaultColour: Reset default colors
00028 C> \endverbatim
00029 C> \~
00030 C>
00031 C
00032 C
00033 C  TCS Graphik Grundfunktionen für Windows
00034 C
00035 C     Version 1.95 bzw. (2022,88,x)
00036 C     - Anpassung 64bit Windows 10 und kleinere Bugfixes
00037 C
00038 C     Version 1.94 bzw. (2021,123,x)
00039 C     - Ergaenzung englische Dokumentation
00040 C
00041 C     Version 1.93 bzw. (2020,332,x)
00042 C     - Fehlerbehandlung analog SDL-Version
00043 C
00044 C     Version 1.92 bzw. (2020,230,x)
00045 C     - Harmonisierung Commonblock TKTRNX
00046 C     - Verwendung von khorsz, kversz, khomey in Abhängigkeit vom Zeichensatz
00047 C
00048 C     Version 1.91 bzw. (2017,317,x)
00049 C     - Bugfix
00050 C
00051 C     Version 1.9
00052 C     - Anpassung Windows7
00053 C
00054 C     Version 1.8 bzw. (2008,134,x)
00055 C     - Hardcopy fuer Journal=3 in Form von Postscriptfiles. TBD.
00056 C     - Ergaenzung Journal=3: Implementation Schriftarten.
00057 C     - DRWABS bei Journal=3: Der Endpunkt wird erst beim Neuzeichnen ge-
00058 C       setzt, im Journal steht nur die Linie mit Endpunkt. Vorteil: UNIX
00059 C       muss den Endpunkt so nicht zweimal setzen.
00060 C     - Fehlermeldungen der Listenverwaltung fuer Journal=3 erfolgen durch
00061 C       GraphError bzw. Unterprogramm TCSJouListError.
00062 C     - Bugfix TCSdWINc.h: Eintrag von TCSLEV3 in C++ Klassendefinition.
00063 C     - Bugfix OUTGTEXT: Prüfung auf freien Platz erfolgt mit gesamtem String.
00064 C
00065 C     Version 1.7 bzw. (2005,291,x)
00066 C     - Einfuehrung des Windows-unabhaengigen Journals zur Vorbereitung
00067 C       der X11-Version. Wahl des Journaltyps (Metafile oder Liste) durch
00068 C       bedingte Kompilation, gesteuert von der Konstante JOURNALTYP
00069 C       im File TCSdWINc.c
00070 C     - Bugfix GraphicError: ErrSeverity=0 entspricht jetzt NO ACTION.
00071 C     - Das System wird nicht mehr durch Fortran-Pragmas in TCSLEV, sondern
00072 C       durch das neue Unterprogramm TCSLEV3 in TCSdWINc.c ermittelt.
00073 C
00074 C     Version 1.6 bzw. (2004,302,x)
00075 C     - Auslagern der Subroutine INITT in ein eigenes File. So kann sicher-
00076 C       gestellt werden, dass sich INITT stets im *.exe des Hauptprogrammes
00077 C       und nicht in einer DLL befindet und eine Ermittlung der Programm-
00078 C       instanz und nicht der DLL-Instanz erfolgt.
00079 C     - Sources der LIB- und DLL-Version zusammengefasst
00080 C
00081 C     Version 1.5 bzw. (2004,167,x)
00082 C     - Anpassung TCSLEV: 5= Alternative Win32-Version für GCC
00083 C
00084 C     Version 1.4 bzw. (2004, 22,x)
00085 C     - Bugfix OUTGTEXT: Bei c-Strings auch char(0) als Stringende erkennen
00086 C     - Bugfix INITT1: Wiederherstellung Charakterdefinitionsblock nach
00087 C       Erzeugung des Statusfensterfonts -> Buchstabengroesse bei ITALIC,
00088 C       ITALIR, DBLSIZ, NRMSIZ wird jetzt richtig gesetzt.
00089 C     - Verschieben und Scrollen Statusfenster auch bei Eingabe möglich
00090 C
00091 C     Version 1.3 bzw. (2003, 78,x)
00092 C     - Falls die eigene Applikation in einem anderen Fenster aktiv ist, setzt
00093 C       TINPUT den Fokus wieder in dieses Fenster zurück
00094 C     - Icon für das Graphikfenster
00095 C     - Instanzermittlung ueber Programmnamen fuer die DLL-Version
00096 C
00097 C     Version 1.2 bzw. (2003, 36,x)
00098 C     - Ergänzung lib$movc3 zur Kompatibilität DOS
00099 C     - Verwirrendes Bildschirmverhalten bei sehr langsamen Rechnern nach Erase
```

```
00100 C       -> Einfügen UpdateWindow
00101 C
00102 C     Version 1.1 bzw. (2002,292,x)
00103 C     - Umbenennung TKTRNX.FOR in TKTRNX.FD zur Kompatibilität CP/M
00104 C
00105 C     Version 1.0
00106 C     - Erweiterungen gegenüber Tektronix:
00107 C           subroutine TOUTSTC (String): Ausgabe Fortran-String
00108 C           subroutine STATST (String) : Ausgabe String in Statusfenster
00109 C           subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
00110 C           subroutine TXTCOL (iCol): Setzen Textfarbe
00111 C           subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00112 C           subroutine DefaultColour: Wiederherstellung Defaultfarben
00113 C
00114 C
00115 C     27.09.02         Dr.-Ing. K. Friedewald
00116 C
00117
00118
00119
00120 C
00121 C  Ausgabe der Softwareversion
00122 C
00123       subroutine tcslev(LEVEL)
00124       integer LEVEL(3)
00125       level(1)=2022     ! Aenderungsjahr
00126       level(2)=  88     ! Aenderungstag
00127 C Kennzeichnung des Systems, wird im systemabhaengigem Code gesetzt
00128 C     3=Watcom && MS-Win16  4=Watcom && MS-Win32  5=GNU-Win32  7=GNU-Win64
00129       call tcslev3 (level(3))
00130
00131       return
00132       end
00133
00134
00135
00136 C
00137 C  Abspeichern Terminal Status Area (wie DOS)
00138 C
00139
00140       subroutine svstat (Array)
00141       integer array(1)
00142       include 'TKTRNX.FD'
00143       integer arr(1)
00144       equivalence(arr(1),khomey)
00145       do 10 i=1,itktrnxl
00146        array(i)= arr(i)
00147 10    continue
00148       return
00149       end
00150
00151
00152
00153       subroutine restat (Array)
00154       integer array(1)
00155       include 'TKTRNX.FD'
00156       integer arr(1)
00157       equivalence(arr(1),khomey)
00158       do 10 i=1,itktrnxl
00159        arr(i)= array(i)
00160 10    continue
00161       call movabs (kbeamx, kbeamy)
00162       return
00163       end
00164
00165
00166
00167 C
00168 C  Relative Zeichenbefehle (wie DOS)
00169 C
00170
00171       subroutine movrel (iX, iY)
00172       include 'TKTRNX.FD'
00173       ixx= kbeamx + ix
00174       iyy= kbeamy + iy
00175       call movabs (ixx, iyy)
00176       return
00177       end
00178
00179
00180
00181       subroutine pntrel (iX, iY)
00182       include 'TKTRNX.FD'
00183       ixx= kbeamx + ix
00184       iyy= kbeamy + iy
00185       call pntabs (ixx, iyy)
00186       return
```

```
00187       end
00188
00189
00190
00191       subroutine drwrel (iX, iY)
00192       include 'TKTRNX.FD'
00193       ixx= kbeamx + ix
00194       iyy= kbeamy + iy
00195       call drwabs (ixx, iyy)
00196       return
00197       end
00198
00199
00200
00201       subroutine dshrel (iX, iY, iMask)
00202       include 'TKTRNX.FD'
00203       ixx= kbeamx + ix
00204       iyy= kbeamy + iy
00205       call dshabs (ixx, iyy, imask)
00206       return
00207       end
00208
00209 C
00210 C   Ersatz SEELOC der CP/M-Version, SEELOC1 unnötig (wie DOS)
00211 C
00212
00213       subroutine seeloc (IX,IY)
00214       include 'TKTRNX.FD'
00215       ix= kbeamx
00216       iy= kbeamy
00217       return
00218       end
00219
00220
00221
00222 C
00223 C  Textausgabe, geändert zu DOS-Version
00224 C
00225
00226
00227
00228       subroutine toutpt (iChr)
00229       include 'TKTRNX.FD'
00230       call outgtext (char(ichr))
00231       return
00232       end
00233
00234
00235
00236       subroutine toutst (nChr, iChrArr)
00237       integer iChrArr (1)
00238       if (nchr.eq.0) return
00239       do 10 i=1,nchr
00240        call toutpt (ichrarr(i))
00241 10    continue
00242       return
00243       end
00244
00245
00246
00247       subroutine toutstc (String)
00248       character *(*) String
00249       call outgtext (string)
00250       return
00251       end
00252
00253
00254
00255       subroutine statst (String)
00256       character *(*) String
00257       call outtext (string)
00258       return
00259       end
00260
00261
00262
00263
00264 C
00265 C>  Entry Dummyroutinen
00266 C         (WINLBL keine Dummyroutine, ALPHA zusätzlich)
00267 C
00268
00269       subroutine    anmode
00270 C> AlfMod
00271       entry         alfmod
00272 C> pClipt
00273       entry         pclipt
```

```
00274 C> ioWait
00275       entry         iowait
00276 C> alpha
00277       entry         alpha
00278       return
00279       end
00280
00281
00282
00283       logical function winselect (iDummy)
00284       winselect= .false.
00285       return
00286       end
00287
```

6.36 TCSdWINc.c File Reference

MS Windows Port: Low-Level Driver.

```
#include <windows.h>
#include <windowsx.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <tchar.h>
#include "TCSdWINc.h"
#include "TKTRNX.h"
```

Macros

- #define JOURNALTYP 1
- #define INIFILEXT _TEXT(".INI")
- #define WIN32_LEAN_AND_MEAN
- #define MAX_PENSTYLE_INDEX 3
- #define MAX_COLOR_INDEX 15
- #define TMPSTRLEN TCS_WINDOW_NAMELEN
- #define TMPSTRLREN TCS_WINDOW_NAMELEN

Typedefs

- typedef TCHAR StatLine[STAT_MAXCOLUMNS+1]
- typedef TCHAR ErrMsg[STAT_MAXCOLUMNS]

Functions

- void CreateMainWindow_IfNecessary (HINSTANCE *hMainProgInst, HWND *hMainProgWindow, LPTSTR szWinName)
- void TCSGraphicError (int iErr, const char *msg)
- bool PointInWindow (FTNINT ix1, FTNINT iy1)
- bool ClipLineStart (FTNINT ix1, FTNINT iy1, FTNINT ix2, FTNINT iy2, FTNINT *isx, FTNINT *isy)
- void TCSWndProc_OnPaint (HWND hWindow)
- void TCSWndProc_OnSize (HWND hWindow, UINT message, WPARAM width, LPARAM height)
- void TCSWndProc_OnRbuttondown (HWND hWindow, BOOL DoubleClick, int MouseX, int MouseY, UINT ShftCtrlKeyMask)
- bool TCSWndProc_OnErasebkgnd (HWND hWindow, HDC hDC)
- bool TCSWndProc_OnCopyClipboard ()
- LRESULT CALLBACK EXPORT16 TCSWndProc (HWND hWindow, UINT Message, WPARAM wParam, L↩ PARAM lParam)
- void TCSstatWndProc_OnPaint (HWND hWindow)
- void TCSstatWndProc_OnKillfocus (HWND hWindow, HWND hNewWindow)

- void TCSstatWndProc_OnGetminmaxinfo (HWND hWindow, MINMAXINFO FAR *lpMinMaxInfo)
- void TCSstatWndProc_OnVScroll (HWND hWindow, HWND hNewWindow, WPARAM wParam, LPARAM lParam)
- LRESULT CALLBACK EXPORT16 TCSstatWndProc (HWND hWindow, UINT Message, WPARAM wParam, LPARAM lParam)
- void tcslev3 (FTNINT *SysLev)
- void PresetProgPar ()
- void CustomizeProgPar ()
- void winlbl (FTNSTRPAR *PloWinNam, FTNSTRPAR *StatWinNam, FTNSTRPAR *IniFilNam FTNSTRP↩AR_TAIL(IniFilNam))
- void initt1 (HINSTANCE *hParentInstance, HWND *hParentWindow)
- void finitt ()
- void swind1 (FTNINT *ix1, FTNINT *iy1, FTNINT *ix2, FTNINT *iy2)
- void erase (void)
- void movabs (FTNINT *ix, FTNINT *iy)
- void drwabs (FTNINT *ix, FTNINT *iy)
- void dshabs (FTNINT *ix, FTNINT *iy, FTNINT *iMask)
- void pntabs (FTNINT *ix, FTNINT *iy)
- void bckcol (FTNINT *iCol)
- void lincol (FTNINT *iCol)
- void txtcol (FTNINT *iCol)
- void DefaultColour (void)
- void outgtext (FTNSTRPAR *ftn_string FTNSTRPAR_TAIL(ftn_string))
- void italic (void)
- void italir (void)
- void dblsiz (void)
- void nrmsiz (void)
- void csize (FTNINT *ix, FTNINT *iy)
- void tinput (FTNINT *ic)
- void dcursr (FTNINT *ic, FTNINT *ix, FTNINT *iy)
- void bell (void)
- void outtext (FTNSTRPAR *ftn_string FTNSTRPAR_TAIL(ftn_string))
- void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string, FTNINT *iL FTNSTRPAR_TAIL(ftn_string))
- void hdcopy (void)
- void lib_movc3 (FTNINT *len, FTNSTRPAR *sou, FTNSTRPAR *dst FTNSTRPAR_TAIL(sou) FTNSTRP↩AR_TAIL(dst))

Variables

- static RECT TCSrect = {0,0, HiRes(TEK_XMAX),HiRes(TEK_YMAX)}
- static bool TCSinitialized = false
- static bool ClippingNotActive = true
- static bool TCSStatWindowAutomatic = true
- static HINSTANCE hTCSInst = NULL
- static HWND hTCSWindow = NULL
- static HWND hTCSstatWindow = NULL
- static HWND hOwnerWindow = NULL
- static HDC hTCSWindowDC
- static HDC hTCSMetaFileDC
- static LOGFONT TCSFontdefinition
- static HFONT hTCSFont
- static HFONT hTCSSysFont
- static HPEN hTCSPen
- static HCURSOR hGinCurs
- static HCURSOR hMouseCurs

- static TCHAR szTCSWindowName [TCS_WINDOW_NAMELEN] = ""
- static TCHAR szTCSstatWindowName [TCS_WINDOW_NAMELEN] = ""
- static TCHAR szTCSMainWindowName [TCS_WINDOW_NAMELEN] = TCS_MAINWINDOW_NAME
- static TCHAR szTCSIniFile [TCS_FILE_NAMELEN] = TCS_INIFILE_NAME INIFILEXT
- static TCHAR szTCSIconFile [TCS_FILE_NAMELEN] = TCS_ICONFILE_NAME
- static TCHAR szTCSMenuCopyText [TCS_MENUENTRY_LEN] = TCS_INIDEF_COPMEN
- static TCHAR szTCSHardcopyFile [TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME
- static TCHAR szTCSGraphicFont [TCS_FILE_NAMELEN] = TCS_INIDEF_FONT
- static TCHAR szTCSSysFont [TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT
- static TCHAR szTCSsect0 [TCS_FILE_NAMELEN] = TCS_INISECT0
- static StatLine TCSstatTextBuf [STAT_MAXROWS]
- static int TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX
- static int TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY
- static int TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX
- static int TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY
- static int TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX
- static int TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY
- static int TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX
- static int TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY
- static int TCSstatScrollY
- static int TCSstatOrgY
- static int TCSstatCursorPosY
- static int TCSstatRow
- static int TextLineHeight
- static int TCSCharHeight
- static int TCSBackgroundColour
- static int TCSDefaultLinCol = TCS_INIDEF_LINCOL
- static int TCSDefaultTxtCol = TCS_INIDEF_TXTCOL
- static int TCSDefaultBckCol = TCS_INIDEF_BCKCOL
- static int iHardcopyCount =1
- static POINT TCSGinCurPos = { TEK_XMAX / 2, TEK_YMAX / 2}
- static ErrMsg szTCSErrorMsg [(int) MSG_MAXERRNO+1]
- static int TCSErrorLev [(int) MSG_MAXERRNO+1]
- static DWORD dwPenStyle [ ]
- static DWORD dwColorTable [ ]

6.36.1 Detailed Description

MS Windows Port: Low-Level Driver.

Version

1.97

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

system-specific subroutines of the teklib-library
Definition in file TCSdWINc.c.

6.36.2 Macro Definition Documentation

6.36.2.1 INIFILEXT

```
#define INIFILEXT _TEXT(".INI")
```
Definition at line 231 of file TCSdWINc.c.

6.36.2.2 JOURNALTYP

```
#define JOURNALTYP 1
```
Definition at line 218 of file TCSdWINc.c.

6.36.2.3 MAX_COLOR_INDEX

```
#define MAX_COLOR_INDEX 15
```
Definition at line 509 of file TCSdWINc.c.

6.36.2.4 MAX_PENSTYLE_INDEX

```
#define MAX_PENSTYLE_INDEX 3
```
Definition at line 486 of file TCSdWINc.c.

6.36.2.5 TMPSTRLEN

```
#define TMPSTRLEN TCS_WINDOW_NAMELEN
```

6.36.2.6 TMPSTRLREN

```
#define TMPSTRLREN TCS_WINDOW_NAMELEN
```

6.36.2.7 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```
Definition at line 257 of file TCSdWINc.c.

6.36.3 Typedef Documentation

6.36.3.1 ErrMsg

```
typedef TCHAR ErrMsg[STAT_MAXCOLUMNS]
```
Definition at line 428 of file TCSdWINc.c.

6.36.3.2 StatLine

```
typedef TCHAR StatLine[STAT_MAXCOLUMNS+1]
```
Definition at line 400 of file TCSdWINc.c.

6.36.4 Function Documentation

6.36.4.1 bckcol()

```
void bckcol (
            FTNINT * iCol )
```

Definition at line 2925 of file TCSdWINc.c.

6.36.4.2 bell()

```
void bell (
            void  )
```

Definition at line 3638 of file TCSdWINc.c.

6.36.4.3 ClipLineStart()

```
bool ClipLineStart (
            FTNINT ix1,
            FTNINT iy1,
            FTNINT ix2,
            FTNINT iy2,
            FTNINT * isx,
            FTNINT * isy )
```

Definition at line 730 of file TCSdWINc.c.

6.36.4.4 CreateMainWindow_IfNecessary()

```
void CreateMainWindow_IfNecessary (
            HINSTANCE * hMainProgInst,
            HWND * hMainProgWindow,
            LPTSTR szWinName )
```

In case that the compiler has not created a window for the main program, this subroutine creates and shows a new main window. The class will be named according to the constant WINMAIN_DEFWINCLASS.
The window icon can be defined as WinMainIcon by a resource file.

Parameters

in	*hMainProgInst*	Main instance
in,out	*hMainProgWindow*	Main window
in	*szWinName*	Window name in case a main window does not exist

Definition at line 70 of file CreateMainWindow.c.

6.36.4.5 csize()

```
void csize (
            FTNINT * ix,
            FTNINT * iy )
```

Definition at line 3292 of file TCSdWINc.c.

6.36.4.6 CustomizeProgPar()

```
void CustomizeProgPar ( )
```

Definition at line 1744 of file TCSdWINc.c.

6.36.4.7 dblsiz()

```
void dblsiz (
            void  )
```

Definition at line 3212 of file TCSdWINc.c.

6.36.4.8 dcursr()

```
void dcursr (
            FTNINT * ic,
            FTNINT * ix,
            FTNINT * iy )
```

Definition at line 3477 of file TCSdWINc.c.

6.36.4.9 DefaultColour()

```
void DefaultColour (
            void  )
```

Definition at line 3011 of file TCSdWINc.c.

6.36.4.10 drwabs()

```
void drwabs (
            FTNINT * ix,
            FTNINT * iy )
```

Definition at line 2747 of file TCSdWINc.c.

6.36.4.11 dshabs()

```
void dshabs (
            FTNINT * ix,
            FTNINT * iy,
            FTNINT * iMask )
```

Definition at line 2801 of file TCSdWINc.c.

6.36.4.12 erase()

```
void erase (
            void  )
```

Definition at line 2595 of file TCSdWINc.c.

6.36.4.13 finitt()

```
void finitt ( )
```

Definition at line 2520 of file TCSdWINc.c.

6.36.4.14 GraphicError()

```
void GraphicError (
            FTNINT * iErr,
            FTNSTRPAR * ftn_string,
            FTNINT *iL  FTNSTRPAR_TAILftn_string )
```

Definition at line 3676 of file TCSdWINc.c.

6.36.4.15 hdcopy()

```
void hdcopy (
             void  )
```

Definition at line 3690 of file TCSdWINc.c.

6.36.4.16 initt1()

```
void initt1 (
             HINSTANCE * hParentInstance,
             HWND * hParentWindow )
```

Definition at line 1942 of file TCSdWINc.c.

6.36.4.17 italic()

```
void italic (
             void  )
```

Definition at line 3136 of file TCSdWINc.c.

6.36.4.18 italir()

```
void italir (
             void  )
```

Definition at line 3174 of file TCSdWINc.c.

6.36.4.19 lib_movc3()

```
void lib_movc3 (
             FTNINT * len,
             FTNSTRPAR * sou,
             FTNSTRPAR *dst  FTNSTRPAR_TAILsou) FTNSTRPAR_TAIL(dst )
```

Definition at line 3921 of file TCSdWINc.c.

6.36.4.20 lincol()

```
void lincol (
             FTNINT * iCol )
```

Definition at line 2946 of file TCSdWINc.c.

6.36.4.21 movabs()

```
void movabs (
             FTNINT * ix,
             FTNINT * iy )
```

Definition at line 2719 of file TCSdWINc.c.

6.36.4.22 nrmsiz()

```
void nrmsiz (
             void  )
```

Definition at line 3252 of file TCSdWINc.c.

6.36.4.23 outgtext()

```
void outgtext (
            FTNSTRPAR *ftn_string  FTNSTRPAR_TAILftn_string )
```

Definition at line 3030 of file TCSdWINc.c.

6.36.4.24 outtext()

```
void outtext (
            FTNSTRPAR *ftn_string  FTNSTRPAR_TAILftn_string )
```

Definition at line 3646 of file TCSdWINc.c.

6.36.4.25 pntabs()

```
void pntabs (
            FTNINT * ix,
            FTNINT * iy )
```

Definition at line 2896 of file TCSdWINc.c.

6.36.4.26 PointInWindow()

```
bool PointInWindow (
            FTNINT ix1,
            FTNINT iy1 )
```

Definition at line 721 of file TCSdWINc.c.

6.36.4.27 PresetProgPar()

```
void PresetProgPar ( )
```

Definition at line 1715 of file TCSdWINc.c.

6.36.4.28 swind1()

```
void swind1 (
            FTNINT * ix1,
            FTNINT * iy1,
            FTNINT * ix2,
            FTNINT * iy2 )
```

Definition at line 2586 of file TCSdWINc.c.

6.36.4.29 TCSGraphicError()

```
void TCSGraphicError (
            int iErr,
            const char * msg )
```

Definition at line 519 of file TCSdWINc.c.

6.36.4.30 tcslev3()

```
void tcslev3 (
            FTNINT * SysLev )
```

Definition at line 1678 of file TCSdWINc.c.

6.36.4.31 TCSstatWndProc()

```
LRESULT CALLBACK EXPORT16 TCSstatWndProc (
            HWND hWindow,
            UINT Message,
            WPARAM wParam,
            LPARAM lParam )
```

Definition at line 1656 of file TCSdWINc.c.

6.36.4.32 TCSstatWndProc_OnGetminmaxinfo()

```
void TCSstatWndProc_OnGetminmaxinfo (
            HWND hWindow,
            MINMAXINFO FAR * lpMinMaxInfo )
```

Definition at line 1597 of file TCSdWINc.c.

6.36.4.33 TCSstatWndProc_OnKillfocus()

```
void TCSstatWndProc_OnKillfocus (
            HWND hWindow,
            HWND hNewWindow )
```

Definition at line 1590 of file TCSdWINc.c.

6.36.4.34 TCSstatWndProc_OnPaint()

```
void TCSstatWndProc_OnPaint (
            HWND hWindow )
```

Definition at line 1569 of file TCSdWINc.c.

6.36.4.35 TCSstatWndProc_OnVScroll()

```
void TCSstatWndProc_OnVScroll (
            HWND hWindow,
            HWND hNewWindow,
            WPARAM wParam,
            LPARAM lParam )
```

Definition at line 1620 of file TCSdWINc.c.

6.36.4.36 TCSWndProc()

```
LRESULT CALLBACK EXPORT16 TCSWndProc (
            HWND hWindow,
            UINT Message,
            WPARAM wParam,
            LPARAM lParam )
```

Definition at line 1530 of file TCSdWINc.c.

6.36.4.37 TCSWndProc_OnCopyClipboard()

```
bool TCSWndProc_OnCopyClipboard ( )
```

Definition at line 1410 of file TCSdWINc.c.

6.36.4.38 TCSWndProc_OnErasebkgnd()

```
bool TCSWndProc_OnErasebkgnd (
            HWND hWindow,
            HDC hDC )
```

Definition at line 1389 of file TCSdWINc.c.

6.36.4.39 TCSWndProc_OnPaint()

```
void TCSWndProc_OnPaint (
            HWND hWindow )
```

Definition at line 1119 of file TCSdWINc.c.

6.36.4.40 TCSWndProc_OnRbuttondown()

```
void TCSWndProc_OnRbuttondown (
            HWND hWindow,
            BOOL DoubleClick,
            int MouseX,
            int MouseY,
            UINT ShftCtrlKeyMask )
```

Definition at line 1380 of file TCSdWINc.c.

6.36.4.41 TCSWndProc_OnSize()

```
void TCSWndProc_OnSize (
            HWND hWindow,
            UINT message,
            WPARAM width,
            LPARAM height )
```

Definition at line 1364 of file TCSdWINc.c.

6.36.4.42 tinput()

```
void tinput (
            FTNINT * ic )
```

Definition at line 3346 of file TCSdWINc.c.

6.36.4.43 txtcol()

```
void txtcol (
            FTNINT * iCol )
```

Definition at line 2988 of file TCSdWINc.c.

6.36.4.44 winlbl()

```
void winlbl (
            FTNSTRPAR * PloWinNam,
```

```
            FTNSTRPAR * StatWinNam,
            FTNSTRPAR *IniFilNam  FTNSTRPAR_TAILIniFilNam )
```

Definition at line 1835 of file TCSdWINc.c.

6.36.5 Variable Documentation

6.36.5.1 ClippingNotActive

```
bool ClippingNotActive = true  [static]
```

Definition at line 350 of file TCSdWINc.c.

6.36.5.2 dwColorTable

```
DWORD dwColorTable[]  [static]
```

Initial value:

```
= {
                          RGB (240,240,240),
                          RGB (  0,  0,  0),
                          RGB (240, 80, 80),
                          RGB ( 80,240, 80),
                          RGB ( 80,240,240),
                          RGB ( 80, 80,240),
                          RGB (240,240, 80),
                          RGB (160,160,160),
                          RGB (240, 80,240),
                          RGB (160,  0,  0),
                          RGB (  0,160,  0),
                          RGB (  0,  0,160),
                          RGB (  0,160,160),
                          RGB (160, 80,  0),
                          RGB ( 80, 80, 80),
                          RGB (160,  0,160)
                         }
```

Definition at line 491 of file TCSdWINc.c.

6.36.5.3 dwPenStyle

```
DWORD dwPenStyle[]  [static]
```

Initial value:

```
= {
                           PS_SOLID,
                           PS_DOT,
                           PS_DASHDOT,
                           PS_DASH
                          }
```

Definition at line 480 of file TCSdWINc.c.

6.36.5.4 hGinCurs

```
HCURSOR hGinCurs  [static]
```

Definition at line 385 of file TCSdWINc.c.

6.36.5.5 hMouseCurs

```
HCURSOR hMouseCurs  [static]
```

Definition at line 386 of file TCSdWINc.c.

6.36.5.6 hOwnerWindow

```
HWND hOwnerWindow = NULL  [static]
```

Definition at line 357 of file TCSdWINc.c.

6.36.5.7 hTCSFont

```
HFONT hTCSFont  [static]
```
Definition at line 380 of file TCSdWINc.c.

6.36.5.8 hTCSInst

```
HINSTANCE hTCSInst = NULL  [static]
```
Definition at line 353 of file TCSdWINc.c.

6.36.5.9 hTCSMetaFileDC

```
HDC hTCSMetaFileDC  [static]
```
Definition at line 362 of file TCSdWINc.c.

6.36.5.10 hTCSPen

```
HPEN hTCSPen  [static]
```
Definition at line 383 of file TCSdWINc.c.

6.36.5.11 hTCSstatWindow

```
HWND hTCSstatWindow = NULL  [static]
```
Definition at line 356 of file TCSdWINc.c.

6.36.5.12 hTCSSysFont

```
HFONT hTCSSysFont  [static]
```
Definition at line 381 of file TCSdWINc.c.

6.36.5.13 hTCSWindow

```
HWND hTCSWindow = NULL  [static]
```
Definition at line 355 of file TCSdWINc.c.

6.36.5.14 hTCSWindowDC

```
HDC hTCSWindowDC  [static]
```
Definition at line 359 of file TCSdWINc.c.

6.36.5.15 iHardcopyCount

```
int iHardcopyCount =1  [static]
```
Definition at line 421 of file TCSdWINc.c.

6.36.5.16 szTCSErrorMsg

```
ErrMsg szTCSErrorMsg[(int) MSG_MAXERRNO+1]  [static]
```

Initial value:

```
=
                {_T("Element 0 unused"),_T("DOS"),_T("DOS"),_T("DOS"),
                 _T("DOS"),_T("DOS"),
                TCS_INIDEF_HDCOPN,
                TCS_INIDEF_HDCWRT,
                TCS_INIDEF_HDCINT,
                TCS_INIDEF_USR,
                TCS_INIDEF_HDCACT,
                TCS_INIDEF_USRWRN,
                TCS_INIDEF_EXIT,
                TCS_INIDEF_COPMEM,
                TCS_INIDEF_COPLCK,
                TCS_INIDEF_JOUCREATE,
                TCS_INIDEF_JOUENTRY,
                TCS_INIDEF_JOUADD,
                TCS_INIDEF_JOUCLR,
                TCS_INIDEF_JOUUNKWN,
                TCS_INIDEF_XMLPARSER,
                TCS_INIDEF_XMLOPEN,
                _T("SDL"),
                TCS_INIDEF_USR2,
                TCS_INIDEF_INI2,
                _T("Maxerr only for internal Use") }
```

Definition at line 429 of file TCSdWINc.c.

6.36.5.17 szTCSGraphicFont

```
TCHAR szTCSGraphicFont[TCS_FILE_NAMELEN] = TCS_INIDEF_FONT  [static]
```

Definition at line 395 of file TCSdWINc.c.

6.36.5.18 szTCSHardcopyFile

```
TCHAR szTCSHardcopyFile[TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME  [static]
```

Definition at line 394 of file TCSdWINc.c.

6.36.5.19 szTCSIconFile

```
TCHAR szTCSIconFile[TCS_FILE_NAMELEN] = TCS_ICONFILE_NAME  [static]
```

Definition at line 392 of file TCSdWINc.c.

6.36.5.20 szTCSIniFile

```
TCHAR szTCSIniFile[TCS_FILE_NAMELEN] = TCS_INIFILE_NAME INIFILEXT  [static]
```

Definition at line 391 of file TCSdWINc.c.

6.36.5.21 szTCSMainWindowName

```
TCHAR szTCSMainWindowName[TCS_WINDOW_NAMELEN] = TCS_MAINWINDOW_NAME  [static]
```

Definition at line 390 of file TCSdWINc.c.

6.36.5.22 szTCSMenuCopyText

```
TCHAR szTCSMenuCopyText[TCS_MENUENTRY_LEN] = TCS_INIDEF_COPMEN  [static]
```

Definition at line 393 of file TCSdWINc.c.

6.36.5.23 szTCSsect0

```
TCHAR szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0  [static]
```
Definition at line 397 of file TCSdWINc.c.

6.36.5.24 szTCSstatWindowName

```
TCHAR szTCSstatWindowName[TCS_WINDOW_NAMELEN] = ""  [static]
```
Definition at line 389 of file TCSdWINc.c.

6.36.5.25 szTCSSysFont

```
TCHAR szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT  [static]
```
Definition at line 396 of file TCSdWINc.c.

6.36.5.26 szTCSWindowName

```
TCHAR szTCSWindowName[TCS_WINDOW_NAMELEN] = ""  [static]
```
Definition at line 388 of file TCSdWINc.c.

6.36.5.27 TCSBackgroundColour

```
int TCSBackgroundColour  [static]
```
Definition at line 417 of file TCSdWINc.c.

6.36.5.28 TCSCharHeight

```
int TCSCharHeight  [static]
```
Definition at line 416 of file TCSdWINc.c.

6.36.5.29 TCSDefaultBckCol

```
int TCSDefaultBckCol = TCS_INIDEF_BCKCOL  [static]
```
Definition at line 420 of file TCSdWINc.c.

6.36.5.30 TCSDefaultLinCol

```
int TCSDefaultLinCol = TCS_INIDEF_LINCOL  [static]
```
Definition at line 418 of file TCSdWINc.c.

6.36.5.31 TCSDefaultTxtCol

```
int TCSDefaultTxtCol = TCS_INIDEF_TXTCOL  [static]
```
Definition at line 419 of file TCSdWINc.c.

6.36.5.32 TCSErrorLev

```
int TCSErrorLev[(int) MSG_MAXERRNO+1]  [static]
```
Initial value:
```
=
                {10,10,10,10,10,10,
                TCS_INIDEF_HDCOPNL,
```

```
                TCS_INIDEF_HDCWRTL,
                TCS_INIDEF_HDCINTL,
                TCS_INIDEF_USRL,
                TCS_INIDEF_HDCACTL,
                TCS_INIDEF_USRWRNL,
                TCS_INIDEF_EXITL,
                TCS_INIDEF_COPMEML,
                TCS_INIDEF_COPLCKL,
                TCS_INIDEF_JOUCREATEL,
                TCS_INIDEF_JOUENTRYL,
                TCS_INIDEF_JOUADDL,
                TCS_INIDEF_JOUCLRL,
                TCS_INIDEF_JOUUNKWNL,
                TCS_INIDEF_XMLPARSERL,
                TCS_INIDEF_XMLOPENL,
                10,
                TCS_INIDEF_USR2L,
                TCS_INIDEF_INI2L,
                10}
```

Definition at line 453 of file TCSdWINc.c.

6.36.5.33 TCSFontdefinition

```
LOGFONT TCSFontdefinition  [static]
```

Definition at line 378 of file TCSdWINc.c.

6.36.5.34 TCSGinCurPos

```
POINT TCSGinCurPos = { TEK_XMAX / 2, TEK_YMAX / 2}  [static]
```

Definition at line 423 of file TCSdWINc.c.

6.36.5.35 TCSinitialized

```
bool TCSinitialized = false  [static]
```

Definition at line 349 of file TCSdWINc.c.

6.36.5.36 TCSrect

```
RECT TCSrect = {0,0, HiRes(TEK_XMAX),HiRes(TEK_YMAX)}  [static]
```

Definition at line 347 of file TCSdWINc.c.

6.36.5.37 TCSstatCursorPosY

```
int TCSstatCursorPosY  [static]
```

Definition at line 413 of file TCSdWINc.c.

6.36.5.38 TCSstatOrgY

```
int TCSstatOrgY  [static]
```

Definition at line 412 of file TCSdWINc.c.

6.36.5.39 TCSstatRow

```
int TCSstatRow  [static]
```

Definition at line 414 of file TCSdWINc.c.

6.36.5.40 TCSstatScrollY

```
int TCSstatScrollY  [static]
```
Definition at line 411 of file TCSdWINc.c.

6.36.5.41 TCSstatTextBuf

```
StatLine TCSstatTextBuf[STAT_MAXROWS]  [static]
```
Definition at line 401 of file TCSdWINc.c.

6.36.5.42 TCSStatWindowAutomatic

```
bool TCSStatWindowAutomatic = true  [static]
```
Definition at line 351 of file TCSdWINc.c.

6.36.5.43 TCSstatWindowIniXrelpos

```
int TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX  [static]
```
Definition at line 407 of file TCSdWINc.c.

6.36.5.44 TCSstatWindowIniXrelsiz

```
int TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX  [static]
```
Definition at line 409 of file TCSdWINc.c.

6.36.5.45 TCSstatWindowIniYrelpos

```
int TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY  [static]
```
Definition at line 408 of file TCSdWINc.c.

6.36.5.46 TCSstatWindowIniYrelsiz

```
int TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY  [static]
```
Definition at line 410 of file TCSdWINc.c.

6.36.5.47 TCSwindowIniXrelpos

```
int TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX  [static]
```
Definition at line 403 of file TCSdWINc.c.

6.36.5.48 TCSwindowIniXrelsiz

```
int TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX  [static]
```
Definition at line 405 of file TCSdWINc.c.

6.36.5.49 TCSwindowIniYrelpos

```
int TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY  [static]
```
Definition at line 404 of file TCSdWINc.c.

6.36.5.50 TCSwindowIniYrelsiz

```
int TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY  [static]
```

Definition at line 406 of file TCSdWINc.c.

6.36.5.51 TextLineHeight

```
int TextLineHeight  [static]
```

Definition at line 415 of file TCSdWINc.c.

6.37 TCSdWINc.c

```
00001 /** ****************************************************************************
00002 \file       TCSdWINc.c
00003 \brief      MS Windows Port: Low-Level Driver
00004 \version    1.97
00005 \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          Systemnahe Graphikroutinen für das Tektronix Graphiksystem
00009 \~english
00010          system-specific subroutines of the teklib-library
00011 \~
00012 **************************************************************************** */
00013
00014 /*
00015         Anmerkungen:
00016           1. Die Systemmeldungen erfolgen in einem eigenen, im Regelfall
00017              unsichtbaren, Fenster. Durch Drücken der rechten Maustaste
00018              im Graphikfenster kann es sichtbar gemacht werden, durch
00019              Setzen des Fokus auf das Graphikfenster verschwindet es wieder.
00020              Bei aktiviertem GIN-Cursor kann die Umschaltung über der Titel-
00021              zeile erfolgen.
00022           2. Die Art der Protokollierung zum Neuzeichnen eines Fensters wird
00023              durch die Konstante JOURNALTYP gesteuert:
00024              --- JOURNALTYP 1 ---
00025              Die Zeichenbefehle werden mithilfe eines Metafiles im Speicher
00026              aufgezeichnet. Das Abspielen eines Metafiles in ein anderes führt
00027              bei Windows bis 3.0 einschließlich zum Systemabsturz! Ab Windows
00028              3.1 aufwärts ist das Problem behoben. Mögliche Abhilfe bei Windows
00029              3.0: Verwendung von Festplatten-basierten Metafiles.
00030              (lt. MS-SDK Dokumentation).
00031              --- JOURNALTYP 2: ---
00032              Anstelle eines Windows-Metafiles (*.wmf) wird ein extended
00033              Metafile (*.emf) verwendet. Funktion wurde im Hinblick auf das
00034              64bit-Windows entwickelt, für 32bit Windows entsteht im Vergleich
00035              zum Journaltyp 1 lediglich ein Performancenachteil.
00036              Anmerkung: MS-WORD besitzt Filter sowohl für *.wmf als auch *.emf
00037                         Dateien. Jedoch ist der *.emf-Filter bis WORD 2000 SP1
00038                         fehlerhaft (Buchstaben des stehen evtl. auf dem Kopf)
00039                         In Windows XP wird nach jedem Neuskalieren das *.emf
00040                         Metafile immer größer. Hierdurch dauert das Neuzeich-
00041                         nen unakzeptabel lange. Dieses Problem tritt bei
00042                         Windows 2000 nicht auf
00043                         -> JOURNALFILE 1 bei 32-bit Windows Default.
00044              --- JOURNALTYP 3: ---
00045              Die Zeichenbefehle werden in einer Liste aufgezeichnet. Ein
00046              einzelner Befehl hat den Aufbau
00047              struct xaction_typ {
00048                         FTNINT action
00049                         FTNINT i1
00050                         FTNINT i2
00051                                 } XACTION;
00052              Die TCS-Befehle im einzelnen:
00053                     erase ()
00054                      XACTION.action= XACTION_ERASE;
00055                     movabs (ix,iy)
00056                      XACTION.action= XACTION_MOVABS;
00057                      XACTION.i1= ix;
00058                      XACTION.i2= ix;
00059                     drwabs (ix.iy)
00060                      XACTION.action= XACTION_DRWABS;
00061                      XACTION.i1= ix;
00062                      XACTION.i2= ix;
00063                     dshabs (ix,iy,iDash)
00064                      XACTION.action= XACTION_DSHSTYLE;
00065                      XACTION.i1= iDash;
00066                      XACTION.action= XACTION_DSHABS;
00067                      XACTION.i1= ix;
00068                      XACTION.i2= ix;
```

```
00069                   pntabs (ix,iy)
00070                    XACTION.action= XACTION_PNTABS;
00071                    XACTION.i1= ix;
00072                    XACTION.i2= ix;
00073                   outgtext (string) - Graphiktext
00074                    XACTION.action= XACTION_GTEXT;
00075                    XACTION.i1= iChar;
00076                    XACTION.i2= iASCII_1;
00077                    XACTION.action= XACTION_ASCII;
00078                    XACTION.i1= iASCII_2;
00079                    XACTION.i2= iASCII_3;
00080                    ...
00081                    XACTION.action= XACTION_ASCII;
00082                    XACTION.i1= iASCII_iChar;
00083                   italic ()
00084                    XACTION.action= XACTION_FONTATTR;
00085                    XACTION.i1= 1; // Attribut 1
00086                    XACTION.i2= 1; // true
00087                   italir ()
00088                    XACTION.action= XACTION_FONTATTR;
00089                    XACTION.i1= 1; // Attribut 1
00090                    XACTION.i2= 0; // false
00091                   dblsiz ()
00092                    XACTION.action= XACTION_FONTATTR;
00093                    XACTION.i1= 2; // Attribut 2
00094                    XACTION.i2= 1; // true
00095                   nrmsiz ()
00096                    XACTION.action= XACTION_FONTATTR;
00097                    XACTION.i1= 2; // Attribut 2
00098                    XACTION.i2= 0; // false
00099
00100                   bckcol (iCol) - keine Zeichenarbeit, nur Commonblock
00101                   lincol (iCol)
00102                   txtcol (iCol)
00103                   DefaultColour () - keine Zeichenarbeit, nur Commonblock
00104
00105            3. Clipping: Windows erwartet die Angabe der Clipping-region in
00106               Devicekoordinaten, daher wird die Clipping-Region bei Vergrößern
00107               und Verzerren des Fensters nicht angepasst. Abhilfe: Implementa-
00108               tion einer eigen Clippingroutine, gesteuert über den Tektronix-
00109               Commonblock. Die (funktionierende) Definition der Clippingregion
00110               bei Ausgabe in die Zwischenablage wird so überflüssig.
00111            4. Linestyle in der Regel nur durchgezogen (wird auch durch LINCOL
00112               zurückgesetzt) -> Merken nicht nötig. Die aktuelle Farbe muß
00113               jedoch für DASH gemerkt werden!!!
00114            5. Übergabe der Windows-Instanz:
00115               A.  Subroutine INITT (iDummy) ruft GetMainInstAndWin auf und
00116                   speichert Instanz und Windowhandle durch SaveMainInstAndWin.
00117               B.  Übergabe des Instanz-Handlers als Parameter von INITT1 (hInst)
00118                   Der Aufruf von INITT1 kann auch mehrmals erfolgen, d.h. möglich
00119                   ist ein Aufruf von INITT1 durch ein C-Hauptprogramm und ein
00120                   erneuter INITT1-Aufruf durch FORTRAN-Unterprogramm. Hier gilt
00121                   dann der erste Aufruf, also die durch C übergebene Instanz.
00122               C.  Zur Vereinfachung der Programmentwicklung mit MS-Visual C++
00123                   wird bei INITT1(0) und Kompilierung durch den MS-Compiler
00124                   die Standardvariable hInst des Visual Studio verwendet.
00125            6. Initialisierung erfolgt in dem File GRAPH2D.INI
00126               Default: im Windows-Directory (c:\WINNT)
00127            7. Abweichend zur DOS-Version entspricht der Farbindex 0 weiss
00128               (Hintergrund) und der Index 1 schwarz.
00129            8. Bei Kompilierung als Konsolenanwendung oder als Window-Anwendung
00130               ohne Default-Windowsystem Fehler möglich. Debuggen durch
00131               Definition von "extended_error_handling".
00132               Ursache: fehlendes Fenster für das Hauptprogramm, Fehler ist
00133               jetzt behoben.
00134            9. Bei Watcom-Compiler den C-Teil ohne Optimierung compilieren!!!
00135           10. Getestete Compiler: WATCOM 11.0c, OpenWatcom 1.0 - 2.0.
00136               Bei neuen Compilern erst mit #define trace_calls übersetzen.
00137               Prüfen, ob __MainWindow definiert!
00138           11. Anpassungen an GNU-Compiler. Anstelle des Watcom-Defaultwindow-
00139               systems wird die eigene Routine WinMain.c verwendet.
00140           12. Auf Wunsch kann das Statusfenster einen privaten Device-Kontext
00141               erhalten: Definition des Symbols STAT_WINDOW_PRIVATE
00142           13. Bei mehreren Fenstern des Hauptprogrammes kann durch <Alt><F6>
00143               zwischen den einzelnen Fenstern umgeschaltet werden.
00144           14. Fuer die 16bit-Version ist das Watcom Default Window System
00145               notwendig. Bei 32bit ist ab der OpenWatcom Version 1.0 das
00146               Defaultsystem deaktiviert.
00147           15. Skalierung des Tektronix-Bildschirmkoordinatensystems (1023/780)
00148               ist bei Bildschirmen höherer Auflösung nicht ausreichend. Falls
00149               Anzahl der Bildschirmpixel in x-Richtung größer als 1024*Pixfac
00150               ist, hinterläßt der Rahmen eines über das Graphikfenster gezogenes
00151               Fensters horizontale und vertikale dünne Linien, die nach Mini-
00152               mierung und Neuzeichnen des Graphikfensters verschwinden.
00153               Vorsicht: PixFac *1024 darf bis einschließlich Windows95 nicht
00154               den 2-Byte int  Zahlenbereich (-32768...+32767) überschreiten!!!
00155               Bei PixFac=100 kann derzeit kein Refresh des Bildschirms durchge-
```

```
00156              fuehrt werden, nach erstem Zeichnen der Linie ((0,0)->(1023,780))
00157              erfolgt kein Neuzeichnen. Nicht nur einzige (?!) Ursache ist die
00158              Verwendung der 16bit GDI Befehle um METAFILE.
00159              Falls PixFac nicht definiert wird, erfolgt keine zusaetzliche
00160              Koordinatentransformation -> Performancegewinn bei alten Systemen.
00161          16. Im Falle von JOURNALTYP=3 darf der Fehler JOUUNKWN nur als
00162              Warnung definiert werden (G2dJouEntryUnknwnL= 1), da sonst inner-
00163              halb von TINPUT ein rekursiver Aufruf von TCSWndProc_OnPaint
00164              ueber GraphicError erfolgt!
00165              Dieser Punkt ist ab Version 1.93 mit der Verlagerung der Routine
00166              GraphicError in den c-Teil behoben.
00167          17. Die Defaultwerte des *.ini-Files müssen fuer die Initialisierung
00168              durch die Registry und/oder XML-Files auch bei der Variablen-
00169              definition angegeben werden, da GetPrivateProfileString nicht
00170              mehr in jedem Fall aufgerufen wird und somit Variablen evtl.
00171              nicht mehr vorbelegt sein koennen.
00172          18. Die Steuerung der Initialisierungmethode erfolgt ueber die File-
00173              extension des Initialisierungfiles.
00174              *.INI: Windows Initialisierungsfile
00175              *.REG: 32bit-Windows Registry
00176              *.XML: XML-Dateien
00177              Der Default (steuerbar durch das Extensiontoken .%) wird durch
00178               #define INIFILEXT _TEXT(".REG")         // win32: Registry
00179              bestimmt.
00180              Durch die Definition der Konstanten REGSUPPORT bzw. XMLSUPPORT
00181              wird der entsprechende Programmteil eingebunden.
00182          19. Aufgrund eines Bugs in der 32-bit Version von win7 darf eine
00183              Tastaturabfrage nicht ohne Filter efolgen, also nicht
00184               GetMessage (&msg, NULL, 0, 0);
00185              sondern
00186               GetMessage (&msg, NULL, WM_NULL, WM_USER);
00187              oder
00188               GetMessage (&msg, hWIND, 0, 0);
00189              Die früheren Versionen bis XP und auch die 64bit Version von Win7
00190              sind hiervon nicht betroffen.
00191          20. XML-Dateien verwenden i.d.R. UTF-8 Codierungen, deswegen erfolgt
00192              bei _UNICODE keine Einbindung des XML-Parsers.
00193          21. Journalfile Typ 3: Die verwendete Listenbibliothek verträgt sich
00194              nicht mit den Makros LoRes und HiRes. Deswegen darf dann PixFac
00195              nicht definiert werden.
00196
00197 */
00198
00199
00200 // #define UNICODE   // fuer Windows-Headerfiles -> jedoch Watcom FTN77 nicht
00201 // #define _UNICODE  // fuer C-Runtime Headerfiles  UNICODEfähig !?!
00202
00203
00204 /*
00205 -------------------- Konfiguration des Zielystems ----------------------
00206 */
00207
00208 // #define PixFac  30                // s. Kommentar 15, 21
00209 // #define STAT_WINDOW_PRIVATE       // s. Kommentar 12
00210 // #define REGSUPPORT                // s. Kommentar 18
00211 // #define XMLSUPPORT                // s. Kommentar 18
00212 // #define INIFILEXT _TEXT(".XML")   // s. Kommentar 18
00213 // #define JOURNALTYP 3              // s. Kommentar 2, 21
00214
00215 #if !defined(JOURNALTYP) // Defaultwerte, falls nicht oben definiert
00216  #if !defined(__WIN32__) && !defined(_WIN32)
00217   /* Defaultvorgabe 16bit: langsame CPU, Aufloesung <= 1024x780 Pxl */
00218   #define JOURNALTYP 1       // s. Kommentar 2, nur *.wmf implementiert
00219   #undef PixFac              // s. Kommentar 15, LoRes
00220   #undef STAT_WINDOW_PRIVATE // s. Kommentar 12
00221  #else
00222   // Default 32bit: kein extended Metafile, Auflösung <= 30*1024 x 30*780 Pxl
00223   #define JOURNALTYP 1       // *.emf hoeherer Overhead -> unnoetig
00224   #define PixFac  30         // Koordinatentransformation hochauflösende CRT's
00225   #undef STAT_WINDOW_PRIVATE // s. Kommentar 12
00226  #endif
00227 #endif
00228
00229 #if !defined(INIFILEXT)
00230  #if !defined(__WIN32__) && !defined(_WIN32)
00231   #define INIFILEXT _TEXT(".INI") // s. Kommentar 18, win16: *.ini Dateien
00232   #undef REGSUPPORT              // Keine vollwertige Registry, nur win.ini
00233   #undef XMLSUPPORT              // Programmgroesse verringern
00234  #else
00235   #define INIFILEXT _TEXT(".REG") // win32: Registry
00236   #define REGSUPPORT
00237   #if (defined(__WIN64__) || defined(_WIN64))
00238    #define XMLSUPPORT
00239   #else
00240    #undef XMLSUPPORT
00241   #endif
00242  #endif
```

```
00243 #endif
00244
00245 #if (JOURNALTYP == 3)
00246  #undef PixFac                  // s. Kommentar 21
00247 #endif
00248
00249 #if defined(UNICODE) || defined(_UNICODE)
00250  #undef XMLSUPPORT              // s. Kommentar 20
00251 #endif
00252
00253 /*
00254 -------------------- Headerfiles ----------------------------------------
00255 */
00256
00257 #define WIN32_LEAN_AND_MEAN
00258 #include <windows.h>   // Muss unbedingt vor den Standard C-Headern stehen, da
00259 #include <windowsx.h>  // hier NULL fuer 16bit Windows als 0 definiert wird
00260
00261 #include <stdlib.h>
00262 #include <string.h>
00263 #include <stdio.h>
00264 #include <tchar.h>     // Public Domain ueber MINGW-Package, nicht nur MS
00265
00266 #if defined(__WATCOMC__) && defined(__SW_BW)
00267  #include <wdefwin.h>   // Compilerswitch -bw: Watcom Default Window System
00268 #endif
00269
00270 #ifdef XMLSUPPORT
00271  #include  "mxml.h"
00272 #endif
00273
00274 #if (JOURNALTYP == 3)
00275  #include "sglib.h"
00276 #endif
00277
00278 #include "TCSdWINc.h"
00279 #include "TKTRNX.h"
00280
00281 /*
00282 -------------------- Debug Compiler Switches ---------------------------
00283 */
00284
00285 // #define extended_error_handling
00286 #if !defined(__WIN32__) && !defined(_WIN32)
00287   #undef extended_error_handling
00288 #endif
00289
00290 // #define trace_calls
00291 /* Debug-Messageboxen / Compilermessages, nach include definieren! */
00292
00293 #ifdef trace_calls
00294
00295  #ifdef __WATCOMC__
00296   #if (__WATCOMC__ == 1100)
00297    #pragma message ( "Symbol __WATCOMC__ defined to 1100 (Version 11.0c)")
00298   #elif (__WATCOMC__ >= 1200)
00299    #pragma message ( "Symbol __WATCOMC__ defined (OpenWatcom Version >= 1.0)")
00300   #else
00301    /* Andere Versionen noch nicht getestet! */
00302    #pragma message ( "Untested Version: Symbol __WATCOMC__ defined to :")
00303    #pragma message (__WATCOMC__) // Erzwingen Fehler zur Erweiterung
00304   #endif
00305   #if !defined(__WIN32__) && !defined(_WIN32)
00306    #pragma message ( "16 bit Windows" )
00307   #else
00308    #pragma message ( "32 bit Windows" )
00309   #endif
00310  #endif
00311
00312  #ifdef _MSC_VER
00313   #pragma message ( "Symbol _MSC_VER defined" )
00314   #if !defined(__WIN32__) && !defined(_WIN32)
00315    #pragma message ( "16 bit Windows" )
00316   #else
00317    #pragma message ( "32 bit Windows" )
00318   #endif
00319  #endif
00320
00321  #ifdef __GNUC__
00322   #warning "GNU-Compiler"
00323   #if !defined(__WIN32__) && !defined(_WIN32)
00324    #warning "16 bit Windows"
00325   #elif !defined(__WIN64__) && !defined(_WIN64)
00326    #warning "32 bit Windows"
00327   #else
00328    #warning "64 bit Windows"
00329   #endif
```

```
00330  #endif
00331
00332 #endif
00333
00334 /*
00335 -------------- Compilerunabhaengige externe Bezüge ---------------------
00336 */
00337
00338
00339 extern void CreateMainWindow_IfNecessary (HINSTANCE * hMainProgInst,
00340                                   HWND * hMainProgWindow, LPTSTR szWinName);
00341
00342
00343 /*
00344 ------------------------- Globale Variablen ----------------------------
00345 */
00346
00347 static  RECT    TCSrect = {0,0, HiRes(TEK_XMAX),HiRes(TEK_YMAX)}; // Plotbereich
00348
00349 static  bool    TCSinitialized = false,
00350                 ClippingNotActive = true,
00351                 TCSStatWindowAutomatic = true;
00352
00353 static  HINSTANCE hTCSInst = NULL;
00354
00355 static  HWND    hTCSWindow = NULL,
00356                 hTCSstatWindow = NULL,
00357                 hOwnerWindow = NULL;
00358
00359 static  HDC     hTCSWindowDC;     // privater DC, gilt ganze Fensterlebensdauer
00360
00361 #if (JOURNALTYP == 1)
00362  static  HDC    hTCSMetaFileDC;   // Metafile als Recorder für WM_PAINT
00363 #elif (JOURNALTYP == 2)
00364  static  HDC    hTCSMetaFileDC;   // extended Metafile als Recorder WM_PAINT
00365 #elif (JOURNALTYP == 3)
00366  struct xJournalEntry_typ {struct xJournalEntry_typ * previous;
00367                            struct xJournalEntry_typ * next;
00368                            FTNINT action; FTNINT i1; FTNINT i2;};
00369  static struct xJournalEntry_typ* hTCSJournal = NULL;
00370                                   // Journal zum Neuzeichnen des Fensters
00371 #endif
00372
00373 #ifdef STAT_WINDOW_PRIVATE
00374  static HDC     hTCSstatWindowDC;
00375 #endif
00376
00377
00378 static  LOGFONT TCSFontdefinition;
00379
00380 static  HFONT   hTCSFont,
00381                 hTCSSysFont;
00382
00383 static  HPEN    hTCSPen;
00384
00385 static  HCURSOR hGinCurs,
00386                 hMouseCurs;
00387
00388 static  TCHAR   szTCSWindowName[TCS_WINDOW_NAMELEN] = "", // Default TCS_WINDOW_NAME erst in ??
       gesetzt
00389                 szTCSstatWindowName[TCS_WINDOW_NAMELEN] = "", // TCS_STATWINDOW_NAME,
00390                 szTCSMainWindowName[TCS_WINDOW_NAMELEN] = TCS_MAINWINDOW_NAME,
00391                 szTCSIniFile[TCS_FILE_NAMELEN] = TCS_INIFILE_NAME INIFILEXT,
00392                 szTCSIconFile[TCS_FILE_NAMELEN] = TCS_ICONFILE_NAME,
00393                 szTCSMenuCopyText[TCS_MENUENTRY_LEN] = TCS_INIDEF_COPMEN,
00394                 szTCSHardcopyFile[TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME,
00395                 szTCSGraphicFont[TCS_FILE_NAMELEN] = TCS_INIDEF_FONT,
00396                 szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT,
00397                 szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0;
00398
00399
00400 typedef TCHAR   StatLine[STAT_MAXCOLUMNS+1];
00401 static  StatLine TCSstatTextBuf[STAT_MAXROWS];
00402
00403 static  int     TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX, // rel. Bildschirmpos.
00404                 TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY, // bei Init in %
00405                 TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX,
00406                 TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY,
00407                 TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX, // dito
00408                 TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY, // Statusfenster
00409                 TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX,
00410                 TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY,
00411                 TCSstatScrollY, // Position des sichtbaren Scrollbereichs
00412                 TCSstatOrgY,    // Ursprung des log. Koordinatensystems
00413                 TCSstatCursorPosY,
00414                 TCSstatRow,
00415                 TextLineHeight,
```

```
00416                 TCSCharHeight,
00417                 TCSBackgroundColour,
00418                 TCSDefaultLinCol = TCS_INIDEF_LINCOL,
00419                 TCSDefaultTxtCol = TCS_INIDEF_TXTCOL,
00420                 TCSDefaultBckCol = TCS_INIDEF_BCKCOL,
00421                 iHardcopyCount =1;  // Zähler zur Erzeugung Filenamen
00422
00423 static  POINT   TCSGinCurPos = { TEK_XMAX / 2, TEK_YMAX / 2};
00424
00425
00426 /* Zuordnung Fehlernummern zu Meldungen,  */
00427
00428 typedef TCHAR   ErrMsg[STAT_MAXCOLUMNS];
00429 static  ErrMsg  szTCSErrorMsg[(int) MSG_MAXERRNO+1] =
00430                 {_T("Element 0 unused"),_T("DOS"),_T("DOS"),_T("DOS"),
00431                  _T("DOS"),_T("DOS"),    // Errno 0..5
00432                 TCS_INIDEF_HDCOPN,      // Errno 6
00433                 TCS_INIDEF_HDCWRT,      // Errno 7
00434                 TCS_INIDEF_HDCINT,      // Errno 8
00435                 TCS_INIDEF_USR,         // Errno 9
00436                 TCS_INIDEF_HDCACT,      // Errno 10
00437                 TCS_INIDEF_USRWRN,      // Errno 11
00438                 TCS_INIDEF_EXIT,        // Errno 12
00439                 TCS_INIDEF_COPMEM,      // Errno 13
00440                 TCS_INIDEF_COPLCK,      // Errno 14
00441                 TCS_INIDEF_JOUCREATE,   // Errno 15
00442                 TCS_INIDEF_JOUENTRY,    // Errno 16
00443                 TCS_INIDEF_JOUADD,      // Errno 17
00444                 TCS_INIDEF_JOUCLR,      // Errno 18
00445                 TCS_INIDEF_JOUUNKWN,    // Errno 19
00446                 TCS_INIDEF_XMLPARSER,   // Errno 20
00447                 TCS_INIDEF_XMLOPEN,     // Errno 21
00448                 _T("SDL"),
00449                 TCS_INIDEF_USR2,        // Errno 23
00450                 TCS_INIDEF_INI2,        // Errno 24
00451                 _T("Maxerr only for internal Use") };
00452
00453 static  int     TCSErrorLev[(int) MSG_MAXERRNO+1] =
00454                 {10,10,10,10,10,10,
00455                 TCS_INIDEF_HDCOPNL,     // Errno 6
00456                 TCS_INIDEF_HDCWRTL,     // Errno 7
00457                 TCS_INIDEF_HDCINTL,     // Errno 8
00458                 TCS_INIDEF_USRL,        // Errno 9
00459                 TCS_INIDEF_HDCACTL,     // Errno 10
00460                 TCS_INIDEF_USRWRNL,     // Errno 11
00461                 TCS_INIDEF_EXITL,       // Errno 12
00462                 TCS_INIDEF_COPMEML,     // Errno 13
00463                 TCS_INIDEF_COPLCKL,     // Errno 14
00464                 TCS_INIDEF_JOUCREATEL,  // Errno 15
00465                 TCS_INIDEF_JOUENTRYL,   // Errno 16
00466                 TCS_INIDEF_JOUADDL,     // Errno 17
00467                 TCS_INIDEF_JOUCLRL,     // Errno 18
00468                 TCS_INIDEF_JOUUNKWNL,   // Errno 19
00469                 TCS_INIDEF_XMLPARSERL,  // Errno 20
00470                 TCS_INIDEF_XMLOPENL,    // Errno 21
00471                 10,
00472                 TCS_INIDEF_USR2L,       // Errno 23
00473                 TCS_INIDEF_INI2L,       // Errno 24
00474                 10};
00475
00476
00477
00478 /* Zuordnung der Linienarten zu Liniennummern */
00479
00480 static  DWORD dwPenStyle[] = {
00481                             PS_SOLID,   /* iMask= 0 */
00482                             PS_DOT,     /* iMask= 1 */
00483                             PS_DASHDOT, /* iMask= 2 */
00484                             PS_DASH     /* iMask= 3 */
00485                            };
00486 #define MAX_PENSTYLE_INDEX 3
00487
00488
00489 /* Zuordnung der Farbennummern zur VGA-Palette */
00490
00491 static  DWORD dwColorTable[] = {
00492                             RGB (240,240,240), /* iCol= 00: weiss (DOS: 01) */
00493                             RGB (  0,  0,  0), /* iCol= 01: schwarz(DOS:00) */
00494                             RGB (240, 80, 80), /* iCol= 02: rot             */
00495                             RGB ( 80,240, 80), /* iCol= 03: gruen           */
00496                             RGB ( 80,240,240), /* iCol= 04: blau            */
00497                             RGB ( 80, 80,240), /* iCol= 05: lila            */
00498                             RGB (240,240, 80), /* iCol= 06: gelb            */
00499                             RGB (160,160,160), /* iCol= 07: grau            */
00500                             RGB (240, 80,240), /* iCol= 08: violett         */
00501                             RGB (160,  0,  0), /* iCol= 09: mattrot         */
00502                             RGB (  0,160,  0), /* iCol= 10: mattgruen       */
```

```
00503                             RGB (  0,  0,160), /* iCol= 11: mattblau        */
00504                             RGB (  0,160,160), /* iCol= 12: mattlila        */
00505                             RGB (160, 80,  0), /* iCol= 13: orange          */
00506                             RGB ( 80, 80, 80), /* iCol= 14: mattgrau        */
00507                             RGB (160,  0,160)  /* iCol= 15: mattviolett     */
00508                            };
00509 #define MAX_COLOR_INDEX 15
00510
00511
00512
00513 /*
00514 ----------------------- Globale Unterprogramme -------------------------
00515 */
00516
00517
00518
00519 void TCSGraphicError (int iErr, const char* msg)
00520 {
00521 char cBuf[TCS_MESSAGELEN];
00522 FTNINT i; // Dummyparameter
00523 FTNSTRDESC  ftnstrg;
00524
00525     snprintf( cBuf, TCS_MESSAGELEN, szTCSErrorMsg[iErr], msg );
00526     if ((iErr == WRN_JOUUNKWN) || // Rekursion von TCSWndProc_OnPaint vermeiden
00527         (iErr == ERR_XMLOPEN)          ) { // System noch nicht initialisiert
00528      MessageBox (NULL, _T(cBuf), szTCSWindowName, MB_ICONINFORMATION);
00529     } else { // ab jetzt mit bell, outtext...
00530      InvalidateRect (hTCSWindow, NULL, true); /* ,ClientArea, EraseFlag */
00531      UpdateWindow (hTCSWindow); /* Notwendig bei OnPaint mit Journaltyp=3 */
00532      bell (); // -> MessgageBeep / winuser.h, ohne Initialisierung verwendbar
00533      ftnstrg.addr= cBuf; ftnstrg.len= strlen (cBuf);
00534      outtext (CALLFTNSTRA(ftnstrg) CALLFTNSTRL(ftnstrg));
00535      if (TCSErrorLev[iErr] >1) {
00536       if (TCSErrorLev[iErr] < 10) {
00537        if (TCSErrorLev[iErr] == 5) {
00538         tinput (&i); // Press Any Key
00539        }
00540        if (TCSErrorLev[iErr]==8) {
00541         MessageBox (NULL, _T(cBuf), szTCSWindowName, MB_ICONINFORMATION);
00542        }
00543       } else {
00544        if (TCSErrorLev[iErr] == 10) {
00545         tinput (&i); // Press Any Key
00546        }
00547        if (TCSErrorLev[iErr]==12) {
00548          MessageBox (NULL, _T(cBuf), szTCSWindowName, MB_ICONSTOP);
00549        }
00550        if (iErr != ERR_EXIT) { // Error-Level von finitt durch XML veraenderbar
00551          TCSErrorLev[ERR_EXIT] = 10; // Hier: Fehler mit Programmabbruch
00552          finitt ();                  // Erzwungenes Beenden durch finitt
00553        }
00554       }
00555      }
00556     }
00557 }
00558
00559
00560
00561 // ------------- Unterprogramme fuer die Event Handler  ----------------
00562
00563
00564
00565
00566 // ------------- Unterprogramme für die Userroutinen -------------------
00567
00568
00569 #if defined(REGSUPPORT)
00570  void StoreIni (TCHAR * szSection, TCHAR * szField, TCHAR * szValue)
00571  {
00572
00573     if (_tcsicmp (szSection,TCS_INISECT1) == 0 ) { // Section1: Names ---------
00574      if (_tcsicmp (szField,TCS_INIVAR_WINNAM) == 0 ) {
00575       if (_tcslen(szTCSWindowName)==0) _tcsncpy(szTCSWindowName,
00576                                                 szValue,TCS_WINDOW_NAMELEN-1);
00577      } else if (_tcsicmp (szField,TCS_INIVAR_STATNAM) == 0 ) {
00578       if (_tcslen(szTCSstatWindowName)==0) _tcsncpy(szTCSstatWindowName,
00579                                                   szValue,TCS_WINDOW_NAMELEN-1);
00580      } else if (_tcsicmp (szField,TCS_INIVAR_MAINWINNAM) == 0 ) {
00581       _tcsncpy(szTCSMainWindowName, szValue,TCS_WINDOW_NAMELEN-1);
00582      } else if (_tcsicmp (szField,TCS_INIVAR_HDCNAM) == 0 ) {
00583       _tcsncpy(szTCSHardcopyFile, szValue,TCS_FILE_NAMELEN-1);
00584      }
00585
00586     } else if (_tcsicmp (szSection,TCS_INISECT2) == 0 ) { // Section2: Layout -
00587      if (_tcsicmp (szField,TCS_INIVAR_COPMEN) == 0 ) {
00588       _tcsncpy(szTCSMenuCopyText, szValue,TCS_MENUENTRY_LEN-1);
00589      } else if (_tcsicmp (szField,TCS_INIVAR_FONT) == 0 ) {
```

```
00590       _tcsncpy(szTCSGraphicFont, szValue,TCS_FILE_NAMELEN-1);
00591      } else if (_tcsicmp (szField,TCS_INIVAR_SYSFONT) == 0 ) {
00592       _tcsncpy(szTCSSysFont, szValue,TCS_FILE_NAMELEN-1);
00593      } else if (_tcsicmp (szField,TCS_INIVAR_ICONNAM) == 0 ) {
00594       _tcsncpy(szTCSIconFile, szValue,TCS_FILE_NAMELEN-1);
00595
00596      } else if (_tcsicmp (szField,TCS_INIVAR_WINPOSX) == 0 ) {
00597       TCSwindowIniXrelpos= * (int*) szValue;
00598      } else if (_tcsicmp (szField,TCS_INIVAR_WINPOSY) == 0 ) {
00599       TCSwindowIniYrelpos= * (int*) szValue;
00600      } else if (_tcsicmp (szField,TCS_INIVAR_WINSIZX) == 0 ) {
00601       TCSwindowIniXrelsiz= * (int*) szValue;
00602      } else if (_tcsicmp (szField,TCS_INIVAR_WINSIZY) == 0 ) {
00603       TCSwindowIniYrelsiz= * (int*) szValue;
00604
00605      } else if (_tcsicmp (szField,TCS_INIVAR_STATPOSX) == 0 ) {
00606       TCSstatWindowIniXrelpos= * (int*) szValue;
00607      } else if (_tcsicmp (szField,TCS_INIVAR_STATPOSY) == 0 ) {
00608       TCSstatWindowIniYrelpos= * (int*) szValue;
00609      } else if (_tcsicmp (szField,TCS_INIVAR_STATSIZX) == 0 ) {
00610       TCSstatWindowIniXrelsiz= * (int*) szValue;
00611      } else if (_tcsicmp (szField,TCS_INIVAR_STATSIZY) == 0 ) {
00612       TCSstatWindowIniYrelsiz= * (int*) szValue;
00613
00614      } else if (_tcsicmp (szField,TCS_INIVAR_LINCOL) == 0 ) {
00615       TCSDefaultLinCol= * (int*) szValue;
00616      } else if (_tcsicmp (szField,TCS_INIVAR_TXTCOL) == 0 ) {
00617       TCSDefaultTxtCol= * (int*) szValue;
00618      } else if (_tcsicmp (szField,TCS_INIVAR_BCKCOL) == 0 ) {
00619       TCSDefaultBckCol= * (int*) szValue;
00620      }
00621
00622     } else if (_tcsicmp (szSection,TCS_INISECT3) == 0 ) { // Section3: Messages
00623      if (_tcsicmp (szField,TCS_INIVAR_HDCOPN) == 0 ) {
00624       _tcsncpy(szTCSErrorMsg[WRN_HDCFILOPN], szValue,STAT_MAXCOLUMNS-1);
00625      } else if (_tcsicmp (szField,TCS_INIVAR_HDCOPNL) == 0 ) {
00626       TCSErrorLev[WRN_HDCFILOPN]= * (int*) szValue;
00627
00628      } else if (_tcsicmp (szField,TCS_INIVAR_HDCWRT) == 0 ) {
00629       _tcsncpy(szTCSErrorMsg[WRN_HDCFILWRT], szValue,STAT_MAXCOLUMNS-1);
00630      } else if (_tcsicmp (szField,TCS_INIVAR_HDCWRTL) == 0 ) {
00631       TCSErrorLev[WRN_HDCFILWRT]= * (int*) szValue;
00632
00633      } else if (_tcsicmp (szField,TCS_INIVAR_HDCINT) == 0 ) {
00634       _tcsncpy(szTCSErrorMsg[WRN_HDCINTERN], szValue,STAT_MAXCOLUMNS-1);
00635      } else if (_tcsicmp (szField,TCS_INIVAR_HDCINTL) == 0 ) {
00636       TCSErrorLev[WRN_HDCINTERN]= * (int*) szValue;
00637
00638      } else if (_tcsicmp (szField,TCS_INIVAR_USR) == 0 ) {
00639       _tcsncpy(szTCSErrorMsg[MSG_USR], szValue,STAT_MAXCOLUMNS-1);
00640      } else if (_tcsicmp (szField,TCS_INIVAR_USRL) == 0 ) {
00641       TCSErrorLev[MSG_USR]= * (int*) szValue;
00642
00643      } else if (_tcsicmp (szField,TCS_INIVAR_HDCACT) == 0 ) {
00644       _tcsncpy(szTCSErrorMsg[MSG_HDCACT], szValue,STAT_MAXCOLUMNS-1);
00645      } else if (_tcsicmp (szField,TCS_INIVAR_HDCACTL) == 0 ) {
00646       TCSErrorLev[MSG_HDCACT]= * (int*) szValue;
00647
00648      } else if (_tcsicmp (szField,TCS_INIVAR_USRWRN) == 0 ) {
00649       _tcsncpy(szTCSErrorMsg[WRN_USRPRESSANY], szValue,STAT_MAXCOLUMNS-1);
00650      } else if (_tcsicmp (szField,TCS_INIVAR_USRWRNL) == 0 ) {
00651       TCSErrorLev[WRN_USRPRESSANY]= * (int*) szValue;
00652
00653      } else if (_tcsicmp (szField,TCS_INIVAR_EXIT) == 0 ) {
00654       _tcsncpy(szTCSErrorMsg[ERR_EXIT], szValue,STAT_MAXCOLUMNS-1);
00655      } else if (_tcsicmp (szField,TCS_INIVAR_EXITL) == 0 ) {
00656       TCSErrorLev[ERR_EXIT]= * (int*) szValue;
00657
00658      } else if (_tcsicmp (szField,TCS_INIVAR_COPMEM) == 0 ) {
00659       _tcsncpy(szTCSErrorMsg[WRN_COPYNOMEM], szValue,STAT_MAXCOLUMNS-1);
00660      } else if (_tcsicmp (szField,TCS_INIVAR_COPMEML) == 0 ) {
00661       TCSErrorLev[WRN_COPYNOMEM]= * (int*) szValue;
00662
00663      } else if (_tcsicmp (szField,TCS_INIVAR_COPLCK) == 0 ) {
00664       _tcsncpy(szTCSErrorMsg[WRN_COPYLOCK], szValue,STAT_MAXCOLUMNS-1);
00665      } else if (_tcsicmp (szField,TCS_INIVAR_COPLCKL) == 0 ) {
00666       TCSErrorLev[WRN_COPYLOCK]= * (int*) szValue;
00667
00668      } else if (_tcsicmp (szField,TCS_INIVAR_JOUCREATE) == 0 ) {
00669       _tcsncpy(szTCSErrorMsg[WRN_JOUCREATE], szValue,STAT_MAXCOLUMNS-1);
00670      } else if (_tcsicmp (szField,TCS_INIVAR_JOUCREATEL) == 0 ) {
00671       TCSErrorLev[WRN_JOUCREATE]= * (int*) szValue;
00672
00673      } else if (_tcsicmp (szField,TCS_INIVAR_JOUENTRY) == 0 ) {
00674       _tcsncpy(szTCSErrorMsg[WRN_JOUENTRY], szValue,STAT_MAXCOLUMNS-1);
00675      } else if (_tcsicmp (szField,TCS_INIVAR_JOUENTRYL) == 0 ) {
00676       TCSErrorLev[WRN_JOUENTRY]= * (int*) szValue;
```

```
00677
00678       } else if (_tcsicmp (szField,TCS_INIVAR_JOUADD) == 0 ) {
00679        _tcsncpy(szTCSErrorMsg[WRN_JOUADD], szValue,STAT_MAXCOLUMNS-1);
00680       } else if (_tcsicmp (szField,TCS_INIVAR_JOUADDL) == 0 ) {
00681        TCSErrorLev[WRN_JOUADD]= * (int*) szValue;
00682
00683       } else if (_tcsicmp (szField,TCS_INIVAR_JOUCLR) == 0 ) {
00684        _tcsncpy(szTCSErrorMsg[WRN_JOUCLR], szValue,STAT_MAXCOLUMNS-1);
00685       } else if (_tcsicmp (szField,TCS_INIVAR_JOUCLRL) == 0 ) {
00686        TCSErrorLev[WRN_JOUCLR]= * (int*) szValue;
00687
00688       } else if (_tcsicmp (szField,TCS_INIVAR_JOUUNKWN) == 0 ) {
00689        _tcsncpy(szTCSErrorMsg[WRN_JOUUNKWN], szValue,STAT_MAXCOLUMNS-1);
00690       } else if (_tcsicmp (szField,TCS_INIVAR_JOUUNKWNL) == 0 ) {
00691        TCSErrorLev[WRN_JOUUNKWN]= * (int*) szValue;
00692
00693       } else if (_tcsicmp (szField,TCS_INIVAR_XMLPARSER) == 0 ) {
00694        _tcsncpy(szTCSErrorMsg[ERR_XMLPARSER], szValue,STAT_MAXCOLUMNS-1);
00695       } else if (_tcsicmp (szField,TCS_INIVAR_XMLPARSERL) == 0 ) {
00696        TCSErrorLev[ERR_XMLPARSER]= * (int*) szValue;
00697
00698       } else if (_tcsicmp (szField,ERR_XMLOPEN) == 0 ) {
00699        _tcsncpy(szTCSErrorMsg[ERR_XMLOPEN], szValue,STAT_MAXCOLUMNS-1);
00700       } else if (_tcsicmp (szField,TCS_INIVAR_XMLOPENL) == 0 ) {
00701        TCSErrorLev[ERR_XMLOPEN]= * (int*) szValue;
00702
00703       } else if (_tcsicmp (szField,TCS_INIVAR_USR2) == 0 ) {
00704        _tcsncpy(szTCSErrorMsg[MSG_USR2], szValue,STAT_MAXCOLUMNS-1);
00705       } else if (_tcsicmp (szField,TCS_INIVAR_USR2L) == 0 ) {
00706        TCSErrorLev[MSG_USR2]= * (int*) szValue;
00707
00708       } else if (_tcsicmp (szField,TCS_INIVAR_INI2) == 0 ) {
00709        _tcsncpy(szTCSErrorMsg[WRN_INI2], szValue,STAT_MAXCOLUMNS-1);
00710       } else if (_tcsicmp (szField,TCS_INIVAR_INI2L) == 0 ) {
00711        TCSErrorLev[WRN_INI2]= * (int*) szValue;
00712
00713       }
00714
00715     } // End case section
00716
00717  }
00718 #endif
00719
00720
00721 bool PointInWindow (FTNINT ix1, FTNINT iy1)
00722 {
00723     if (ClippingNotActive ) return true;
00724     return ( (TKTRNX.kminsx <= ix1) && (TKTRNX.kmaxsx >= ix1) &&
00725                     (TKTRNX.kminsy <= iy1) && (TKTRNX.kmaxsy >= iy1));
00726 }
00727
00728
00729
00730 bool ClipLineStart (FTNINT ix1, FTNINT iy1, FTNINT ix2, FTNINT iy2,
00731                                                 FTNINT *isx, FTNINT *isy)
00732 /* ClipLineStart=true: isx,isy Startpunkt; =false: Linie nicht zeichnen */
00733 {
00734     if (ClippingNotActive) {
00735      *isx= ix1; *isy= iy1;
00736      return true;
00737     }
00738
00739     if (ix1 < TKTRNX.kminsx) { /* Start links vom Fenster */
00740      if (ix2 < TKTRNX.kminsx) return false;
00741      *isy= iy1+((TKTRNX.kminsx-ix1) * (iy2-iy1)) / (ix2-ix1);
00742      if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
00743       *isx= TKTRNX.kminsx;
00744       return true;
00745      }
00746      if (iy1 == iy2) return false;
00747      if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
00748       *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00749       *isy= TKTRNX.kminsy;
00750      } else {
00751       *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00752       *isy= TKTRNX.kmaxsy;
00753      }
00754      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00755      return true;
00756
00757     } else if (ix1 > TKTRNX.kmaxsx) { /* Start rechts vom Fenster */
00758      if (ix2 > TKTRNX.kmaxsx) return false;
00759      *isy= iy1+((TKTRNX.kmaxsx-ix1) * (iy2-iy1)) / (ix2-ix1);
00760      if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
00761       *isx= TKTRNX.kmaxsx;
00762       return true;
00763      }
```

```
00764       if (iy1 == iy2) return false;
00765       if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
00766        *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00767        *isy= TKTRNX.kmaxsy;
00768       } else {
00769        *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00770        *isy= TKTRNX.kminsy;
00771       }
00772       if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00773       return true;
00774
00775      } else if (iy1 < TKTRNX.kminsy) { /* Start unter dem Fenster */
00776       if (iy2 < TKTRNX.kminsy) return false;
00777       *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00778       if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00779       *isy= TKTRNX.kminsy;
00780       return true;
00781
00782      } else if (iy1 > TKTRNX.kmaxsy) { /* Start ueber dem Fenster */
00783       if (iy2 > TKTRNX.kmaxsy) return false;
00784       *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00785       if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00786       *isy= TKTRNX.kmaxsy;
00787       return true;
00788
00789      }
00790      *isx= ix1;                          /* Startpunkt liegt im Fenster */
00791      *isy= iy1;
00792      return true;
00793 }
00794
00795
00796
00797 /*
00798 ------------------- Event Handler zum Parsen von XML-Dateien ----------
00799 */
00800
00801 #if defined(XMLSUPPORT)
00802
00803 void sax_callback (mxml_node_t *node, mxml_sax_event_t event, void *usr)
00804 {
00805 char * StorePtr;
00806
00807     switch (event) {
00808      case MXML_SAX_ELEMENT_OPEN: {
00809       switch (*(int*)usr ) {
00810        case -1: { // Statemachine: noch keine aktive Sektion
00811         if (strcmp(mxmlGetElement(node),szTCSsect0) == 0) {
00812          *(int*)usr= 0;  // Parsing active
00813          mxmlElementSetAttr (node,"typ","none");
00814         }
00815         break;
00816        }
00817        case 0: {
00818         if ((strcmp(mxmlGetElement(node),TCS_INISECT1) == 0)  ) {
00819          *(int*)usr= 1; // State: TCS_INISECT1
00820         } else if ((strcmp(mxmlGetElement(node),TCS_INISECT2) == 0)  ) {
00821          *(int*)usr= 2; // State: TCS_INISECT2
00822         } else if ((strcmp(mxmlGetElement(node),TCS_INISECT3) == 0)  ) {
00823          *(int*)usr= 3; // State: TCS_INISECT3
00824         }
00825         mxmlElementSetAttr (node,"typ","none");
00826         break;
00827        }
00828
00829        case 1: { // Section = Names
00830         if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINNAM) == 0)  ) {
00831          mxmlElementSetAttr (node,"typ","opaque");
00832          mxmlElementSetAttrf(node,"store","%p",&szTCSWindowName);
00833         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATNAM) == 0)  ) {
00834          mxmlElementSetAttr (node,"typ","opaque");
00835          mxmlElementSetAttrf(node,"store","%p",&szTCSstatWindowName);
00836         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_MAINWINNAM) == 0)  ) {
00837          mxmlElementSetAttr (node,"typ","opaque");
00838          mxmlElementSetAttrf(node,"store","%p",&szTCSMainWindowName);
00839         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCNAM) == 0)  ) {
00840          mxmlElementSetAttr (node,"typ","opaque");
00841          mxmlElementSetAttrf(node,"store","%p",&szTCSHardcopyFile);
00842         }
00843         break;
00844        }
00845
00846        case 2: { // Section = Layout
00847         if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPMEN) == 0)  ) {
00848          mxmlElementSetAttr (node,"typ","opaque");
00849          mxmlElementSetAttrf(node,"store","%p",&szTCSMenuCopyText);
00850         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_FONT) == 0)  ) {
```

```
00851           mxmlElementSetAttr (node,"typ","opaque");
00852           mxmlElementSetAttrf(node,"store","%p",&szTCSGraphicFont);
00853          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_SYSFONT) == 0)  ) {
00854           mxmlElementSetAttr (node,"typ","opaque");
00855           mxmlElementSetAttrf(node,"store","%p",&szTCSSysFont);
00856          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_ICONNAM) == 0)  ) {
00857           mxmlElementSetAttr (node,"typ","opaque");
00858           mxmlElementSetAttrf(node,"store","%p",&szTCSIconFile);
00859
00860          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSX) == 0)  ) {
00861           mxmlElementSetAttr (node,"typ","integer");
00862           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelpos);
00863          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSY) == 0)  ) {
00864           mxmlElementSetAttr (node,"typ","integer");
00865           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelpos);
00866          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZX) == 0)  ) {
00867           mxmlElementSetAttr (node,"typ","integer");
00868           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelsiz);
00869          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZY) == 0)  ) {
00870           mxmlElementSetAttr (node,"typ","integer");
00871           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelsiz);
00872
00873          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSX) == 0)  ) {
00874           mxmlElementSetAttr (node,"typ","integer");
00875           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelpos);
00876          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSY) == 0)  ) {
00877           mxmlElementSetAttr (node,"typ","integer");
00878           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelpos);
00879          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZX) == 0)  ) {
00880           mxmlElementSetAttr (node,"typ","integer");
00881           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelsiz);
00882          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZY) == 0)  ) {
00883           mxmlElementSetAttr (node,"typ","integer");
00884           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelsiz);
00885
00886          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_LINCOL) == 0)  ) {
00887           mxmlElementSetAttr (node,"typ","integer");
00888           mxmlElementSetAttrf(node,"store","%p",&TCSDefaultLinCol);
00889          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_TXTCOL) == 0)  ) {
00890           mxmlElementSetAttr (node,"typ","integer");
00891           mxmlElementSetAttrf(node,"store","%p",&TCSDefaultTxtCol);
00892          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_BCKCOL) == 0)  ) {
00893           mxmlElementSetAttr (node,"typ","integer");
00894           mxmlElementSetAttrf(node,"store","%p",&TCSDefaultBckCol);
00895          }
00896          break;
00897         }
00898
00899         case 3: { // Section = Messages
00900          if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPN) == 0)  ) {
00901           mxmlElementSetAttr (node,"typ","opaque");
00902           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILOPN]);
00903          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPNL) == 0)  ) {
00904           mxmlElementSetAttr (node,"typ","integer");
00905           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILOPN]);
00906
00907          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRT) == 0)  ) {
00908           mxmlElementSetAttr (node,"typ","opaque");
00909           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILWRT]);
00910          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRTL) == 0)  ) {
00911           mxmlElementSetAttr (node,"typ","integer");
00912           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILWRT]);
00913
00914          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINT) == 0)  ) {
00915           mxmlElementSetAttr (node,"typ","opaque");
00916           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCINTERN]);
00917          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINTL) == 0)  ) {
00918           mxmlElementSetAttr (node,"typ","integer");
00919           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCINTERN]);
00920
00921          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR) == 0)  ) {
00922           mxmlElementSetAttr (node,"typ","opaque");
00923           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR]);
00924          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRL) == 0)  ) {
00925           mxmlElementSetAttr (node,"typ","integer");
00926           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR]);
00927
00928          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACT) == 0)  ) {
00929           mxmlElementSetAttr (node,"typ","opaque");
00930           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_HDCACT]);
00931          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACTL) == 0)  ) {
00932           mxmlElementSetAttr (node,"typ","integer");
00933           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_HDCACT]);
00934
00935          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRN) == 0)  ) {
00936           mxmlElementSetAttr (node,"typ","opaque");
00937           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_USRPRESSANY]);
```

```
00938         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRNL) == 0)  ) {
00939          mxmlElementSetAttr (node,"typ","integer");
00940          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_USRPRESSANY]);
00941
00942         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXIT) == 0)  ) {
00943          mxmlElementSetAttr (node,"typ","opaque");
00944          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_EXIT]);
00945         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXITL) == 0)  ) {
00946          mxmlElementSetAttr (node,"typ","integer");
00947          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_EXIT]);
00948
00949         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPMEM) == 0)  ) {
00950          mxmlElementSetAttr (node,"typ","opaque");
00951          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_COPYNOMEM]);
00952         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPMEML) == 0)  ) {
00953          mxmlElementSetAttr (node,"typ","integer");
00954          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_COPYNOMEM]);
00955
00956         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPLCK) == 0)  ) {
00957          mxmlElementSetAttr (node,"typ","opaque");
00958          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_COPYLOCK]);
00959         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPLCKL) == 0)  ) {
00960          mxmlElementSetAttr (node,"typ","integer");
00961          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_COPYLOCK]);
00962
00963         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATE) == 0)  ) {
00964          mxmlElementSetAttr (node,"typ","opaque");
00965          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCREATE]);
00966         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATEL) == 0)  ) {
00967          mxmlElementSetAttr (node,"typ","integer");
00968          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCREATE]);
00969
00970         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRY) == 0)  ) {
00971          mxmlElementSetAttr (node,"typ","opaque");
00972          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUENTRY]);
00973         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRYL) == 0)  ) {
00974          mxmlElementSetAttr (node,"typ","integer");
00975          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUENTRY]);
00976
00977         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADD) == 0)  ) {
00978          mxmlElementSetAttr (node,"typ","opaque");
00979          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUADD]);
00980         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADDL) == 0)  ) {
00981          mxmlElementSetAttr (node,"typ","integer");
00982          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUADD]);
00983
00984         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLR) == 0)  ) {
00985          mxmlElementSetAttr (node,"typ","opaque");
00986          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCLR]);
00987         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLRL) == 0)  ) {
00988          mxmlElementSetAttr (node,"typ","integer");
00989          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCLR]);
00990
00991         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWN) == 0)  ) {
00992          mxmlElementSetAttr (node,"typ","opaque");
00993          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUUNKWN]);
00994         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWNL) == 0)  ) {
00995          mxmlElementSetAttr (node,"typ","integer");
00996          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUUNKWN]);
00997
00998         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSER) == 0)  ) {
00999          mxmlElementSetAttr (node,"typ","opaque");
01000          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLPARSER]);
01001         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSERL) == 0)  ) {
01002          mxmlElementSetAttr (node,"typ","integer");
01003          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLPARSER]);
01004
01005         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPEN) == 0)  ) {
01006          mxmlElementSetAttr (node,"typ","opaque");
01007          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLOPEN]);
01008         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPENL) == 0)  ) {
01009          mxmlElementSetAttr (node,"typ","integer");
01010          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLOPEN]);
01011
01012         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2) == 0)  ) {
01013          mxmlElementSetAttr (node,"typ","opaque");
01014          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR2]);
01015         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2L) == 0)  ) {
01016          mxmlElementSetAttr (node,"typ","integer");
01017          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR2]);
01018
01019         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2) == 0)  ) {
01020          mxmlElementSetAttr (node,"typ","opaque");
01021          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_INI2]);
01022         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2L) == 0)  ) {
01023          mxmlElementSetAttr (node,"typ","integer");
01024          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_INI2]);
```

```
01025
01026          }
01027          break;
01028         }
01029
01030        }
01031        break;
01032       }
01033
01034       case MXML_SAX_DATA: {
01035        switch (mxmlGetType(node)) {
01036         case MXML_INTEGER: {
01037          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01038          (*(int*)StorePtr)= mxmlGetInteger(node);
01039          break;
01040         }
01041         case MXML_REAL: {
01042          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01043          (*(float*)StorePtr)= mxmlGetReal(node);
01044          break;
01045         }
01046         case MXML_TEXT: {
01047          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01048          strcpy (StorePtr, mxmlGetText(node, NULL));
01049          break;
01050         }
01051         case MXML_OPAQUE: {
01052          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01053          strcpy (StorePtr, mxmlGetOpaque(node));
01054          break;
01055         }
01056        }
01057        break;
01058       }
01059
01060       case MXML_SAX_ELEMENT_CLOSE: {
01061        if ((*(int*)usr==0) && (strcmp(mxmlGetElement(node),szTCSsect0)==0)) {
01062         *(int*)usr= -1; // State: idle
01063        } else if (
01064               ((*(int*)usr==1) && (strcmp(mxmlGetElement(node),TCS_INISECT1)==0))
01065            || ((*(int*)usr==2) && (strcmp(mxmlGetElement(node),TCS_INISECT2)==0))
01066            || ((*(int*)usr==3) && (strcmp(mxmlGetElement(node),TCS_INISECT3)==0))
01067           ) {
01068         *(int*)usr= 0; // State: Parsing active
01069        }
01070        break;
01071       }
01072      }
01073 }
01074
01075
01076 /* ------------------------------------------------------------------------ */
01077
01078
01079 mxml_type_t      sax_type_callback(mxml_node_t  *node)
01080 {
01081 const char *type;
01082
01083     if ((type = mxmlElementGetAttr(node, "typ")) == NULL) type = "none";
01084     if (!strcmp(type, "integer"))
01085      return (MXML_INTEGER);
01086     else if (!strcmp(type, "opaque") || !strcmp(type, "pre"))
01087      return (MXML_OPAQUE);
01088     else if (!strcmp(type, "real"))
01089      return (MXML_REAL);
01090     else if (!strcmp(type, "text"))
01091      return (MXML_TEXT);
01092     else
01093      return (MXML_IGNORE);
01094 }
01095
01096 /* ------------------------------------------------------------------------ */
01097
01098
01099 mxml_error_cb_t sax_error_callback (char *mssg)
01100 {
01101     TCSGraphicError (ERR_XMLPARSER, mssg);
01102     return;
01103 }
01104
01105 /* ------------------------------------------------------------------------ */
01106
01107 #endif   // Ende XML-Unterstützung
01108
01109
01110
01111
```

```
01112 /*
01113 ------------------- Event Handler Graphikfenster -----------------------
01114 */
01115
01116
01117
01118
01119 void TCSWndProc_OnPaint (HWND hWindow)
01120 {
01121 PAINTSTRUCT ps;
01122 #if (JOURNALTYP == 1)
01123  HMETAFILE hmf;
01124  HDC hTCSMetaFileDC1;
01125 #elif (JOURNALTYP == 2)
01126  HENHMETAFILE hmf;
01127  ENHMETAHEADER emh ;
01128  HDC hTCSMetaFileDC1;
01129  RECT   crtrect;
01130 #elif (JOURNALTYP == 3)
01131  struct xJournalEntry_typ    * xJournalEntry;
01132  HPEN   hPenDash, hPenOld;
01133  HFONT  hOldFont;
01134  int    iMaskIndex;
01135  int    iGraphTextLen, iGraphTextLenAkt;
01136  TCHAR  GraphTextBuf[STAT_MAXCOLUMNS+1];
01137 #endif
01138
01139
01140     BeginPaint (hWindow, &ps);
01141
01142 #if (JOURNALTYP == 1)
01143     hmf = CloseMetaFile (hTCSMetaFileDC);
01144     PlayMetaFile (hTCSWindowDC, hmf);           /* Wiederherstellung Anzeige */
01145
01146     hTCSMetaFileDC1  = CreateMetaFile (NULL);   /* 16bit Windows Metafile */
01147     PlayMetaFile (hTCSMetaFileDC1, hmf);        /* für neues Journalfile */
01148     DeleteMetaFile (hmf);                       /* alter Status Bildschirm */
01149     hTCSMetaFileDC = hTCSMetaFileDC1;           /* bereit zum Weiterzeichnen */
01150
01151 #elif (JOURNALTYP == 2)
01152     hmf = CloseEnhMetaFile (hTCSMetaFileDC);
01153     GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
01154     GetClientRect(hTCSWindow, &crtrect); // Zeichenbereich CRT in Pixeln
01155
01156     SetViewportExtEx (hTCSWindowDC, crtrect.right-crtrect.left,
01157                       crtrect.bottom-crtrect.top, NULL); // Zeichne EMF 1:1
01158     SetViewportOrgEx (hTCSWindowDC, crtrect.left, crtrect.bottom, NULL);
01159     SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
01160     SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);
01161
01162     PlayEnhMetaFile (hTCSWindowDC, hmf, &TCSrect); // Wiederherstellung Anzeige
01163
01164     SetViewportExtEx (hTCSWindowDC, crtrect.right-crtrect.left,
01165                       crtrect.top-crtrect.bottom, NULL); // Skaliere auf TEK
01166     SetViewportOrgEx (hTCSWindowDC, crtrect.left, crtrect.top, NULL);
01167     SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
01168     SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);
01169
01170
01171     hTCSMetaFileDC1  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
01172                       _T("TCS for Windows\0Journalfile created by OnPaint\0"));
01173
01174     SetMapMode (hTCSMetaFileDC1, MM_ANISOTROPIC);
01175     SetViewportExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
01176     SetViewportOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);
01177     SetWindowExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
01178     SetWindowOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);
01179
01180     PlayEnhMetaFile (hTCSMetaFileDC1, hmf, &TCSrect); // neues Journal
01181
01182     DeleteEnhMetaFile (hmf);                    // Bildschirminhalt restauriert
01183     hTCSMetaFileDC = hTCSMetaFileDC1;           // bereit zum Weiterzeichnen
01184     SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
01185     SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
01186     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
01187     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
01188
01189     #if !defined(__WIN32__) && !defined(_WIN32)
01190      SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
01191     #else
01192      SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
01193     #endif
01194     SetBkMode (hTCSMetaFileDC, TRANSPARENT );   // Metafile weitergegeben !
01195     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); // CP
01196     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
01197     #if !defined(__WIN32__) && !defined(_WIN32)
01198      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
```

```
01199     #else
01200      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
01201     #endif
01202
01203 #elif (JOURNALTYP == 3)
01204 //      if (hTCSJournal != NULL) {
01205     SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, hTCSJournal, previous, next, xJournalEntry)
01206     while (xJournalEntry != NULL) {
01207      switch (xJournalEntry->action) {
01208       case XACTION_INITT: {
01209        TKTRNX.iLinCol= TCSDefaultLinCol;
01210        TKTRNX.iTxtCol= TCSDefaultTxtCol;
01211        TKTRNX.iBckCol= TCSDefaultBckCol;
01212        initt2(); // HOME, Font, Scale...
01213       } // weiter mit Erase
01214       case XACTION_ERASE: {
01215        SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
01216        SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);
01217        SetBkMode (hTCSWindowDC, TRANSPARENT );
01218        SetTextAlign (hTCSWindowDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
01219        SetTextColor (hTCSWindowDC, dwColorTable[TKTRNX.iTxtCol]);
01220        #if !defined(__WIN32__) && !defined(_WIN32)
01221         SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
01222        #else
01223         SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
01224        #endif
01225        break;
01226       }
01227       case XACTION_MOVABS: {
01228        MoveToEx (hTCSWindowDC, HiRes(xJournalEntry->i1),
01229                                              HiRes(xJournalEntry->i2), NULL);
01230        TKTRNX.kBeamX= xJournalEntry->i1;
01231        TKTRNX.kBeamY= xJournalEntry->i2;
01232        break;
01233       }
01234       case XACTION_DRWABS: {
01235        LineTo (hTCSWindowDC, HiRes(xJournalEntry->i1),
01236                    HiRes(xJournalEntry->i2) ); // Endpunkt nicht mitgezeichnet!
01237        SetPixel (hTCSWindowDC,HiRes(xJournalEntry->i1),
01238                       HiRes(xJournalEntry->i2), dwColorTable[TKTRNX.iLinCol]);
01239        TKTRNX.kBeamX= xJournalEntry->i1;
01240        TKTRNX.kBeamY= xJournalEntry->i2;
01241        break;
01242       }
01243       case XACTION_DSHSTYLE: {
01244        iMaskIndex= xJournalEntry->i1;
01245        break;
01246       }
01247       case XACTION_DSHABS: {
01248        hPenDash= CreatePen (dwPenStyle[iMaskIndex], 0,
01249                                              dwColorTable[TKTRNX.iLinCol]);
01250        #if !defined(__WIN32__) && !defined(_WIN32)
01251         SelectPen (hTCSWindowDC, hPenDash); // 16bit: Makro aus windowsx.h
01252        #else
01253         SelectObject (hTCSWindowDC, hPenDash); // 32bit: GDI Standardaufruf
01254        #endif
01255        LineTo (hTCSWindowDC, HiRes(xJournalEntry->i1),
01256                                              HiRes(xJournalEntry->i2) );
01257        #if !defined(__WIN32__) && !defined(_WIN32)
01258         SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
01259         DeletePen (hPenDash);
01260        #else
01261         SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
01262         DeleteObject (hPenDash);
01263        #endif
01264        TKTRNX.kBeamX= xJournalEntry->i1;
01265        TKTRNX.kBeamY= xJournalEntry->i2;
01266        break;
01267       }
01268       case XACTION_PNTABS: {
01269        SetPixel (hTCSWindowDC,HiRes(xJournalEntry->i1),
01270                    HiRes(xJournalEntry->i2), dwColorTable[TKTRNX.iLinCol]);
01271        TKTRNX.kBeamX= xJournalEntry->i1;
01272        TKTRNX.kBeamY= xJournalEntry->i2;
01273        break;
01274       }
01275       case XACTION_BCKCOL: {
01276        TKTRNX.iBckCol= xJournalEntry->i1;
01277        break;
01278       }
01279       case XACTION_LINCOL: {
01280        hTCSPen= CreatePen (PS_SOLID, 0, dwColorTable[xJournalEntry->i1]);
01281        #if !defined(__WIN32__) && !defined(_WIN32)
01282         hPenOld= SelectPen (hTCSWindowDC, hTCSPen);// 16bit: Makro aus windowsx.h
01283         DeletePen (hPenOld);
01284        #else
01285         hPenOld= SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
```

```
01286         DeleteObject (hPenOld);
01287        #endif
01288        TKTRNX.iLinCol= xJournalEntry->i1;
01289        break;
01290       }
01291       case XACTION_TXTCOL: {
01292        SetTextColor (hTCSWindowDC, dwColorTable[xJournalEntry->i1]);
01293        TKTRNX.iTxtCol= xJournalEntry->i1;
01294        break;
01295       }
01296       case XACTION_FONTATTR: {
01297        TKTRNX.kitalc= xJournalEntry->i1;
01298        TCSFontdefinition.lfItalic= (TKTRNX.kitalc > 0);
01299        hTCSFont= CreateFontIndirect (&TCSFontdefinition);
01300        #if !defined(__WIN32__) && !defined(_WIN32)
01301         hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
01302         DeleteFont (hOldFont);
01303        #else
01304         hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
01305         DeleteObject (hOldFont);
01306        #endif
01307
01308        if (TKTRNX.ksizef != xJournalEntry->i2) {
01309         TKTRNX.ksizef= xJournalEntry->i2;
01310         TCSFontdefinition.lfHeight= (1+TKTRNX.ksizef)*TCSCharHeight;
01311         TCSFontdefinition.lfWidth= 0;
01312         hTCSFont= CreateFontIndirect (&TCSFontdefinition);
01313         #if !defined(__WIN32__) && !defined(_WIN32)
01314          hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
01315          DeleteFont (hOldFont);
01316         #else
01317          hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
01318          DeleteObject (hOldFont);
01319         #endif
01320         TKTRNX.khomey = TEK_YMAX - 1.5f*(1+TKTRNX.ksizef)*TCS_REL_CHR_HEIGHT;
01321        }
01322        break;
01323       }
01324       case XACTION_GTEXT: {
01325        iGraphTextLenAkt= 0;
01326        iGraphTextLen= (int) xJournalEntry->i1;
01327        if (iGraphTextLen > STAT_MAXCOLUMNS) iGraphTextLen= STAT_MAXCOLUMNS;
01328        if (iGraphTextLen == 0) break;
01329        GraphTextBuf[iGraphTextLenAkt++]= (TCHAR) xJournalEntry->i2;
01330        if (iGraphTextLen == 1) {
01331         GraphTextBuf[iGraphTextLenAkt]= (FTNCHAR) 0;
01332         TextOut (hTCSWindowDC, 0,0,GraphTextBuf, iGraphTextLen);
01333        }
01334        break;
01335       }
01336       case XACTION_ASCII: {
01337        if (iGraphTextLenAkt < iGraphTextLen) {
01338         GraphTextBuf[iGraphTextLenAkt++]= (TCHAR) xJournalEntry->i1;
01339         if (iGraphTextLenAkt < iGraphTextLen)
01340          GraphTextBuf[iGraphTextLenAkt++]= (TCHAR) xJournalEntry->i2;
01341         if (iGraphTextLenAkt >= iGraphTextLen)
01342          TextOut (hTCSWindowDC, 0,0,GraphTextBuf, iGraphTextLen);
01343        }
01344        break;
01345       }
01346       case XACTION_NOOP: {
01347        break;
01348       }
01349       default: {
01350        TCSGraphicError (WRN_JOUUNKWN,"");
01351        break;
01352       }
01353      }
01354      xJournalEntry= xJournalEntry -> previous;
01355     }
01356 //      }
01357 #endif
01358
01359     EndPaint( hWindow, &ps );
01360 }
01361
01362
01363
01364 void TCSWndProc_OnSize (HWND hWindow, UINT message, WPARAM width, LPARAM height)
01365 {
01366     switch (message) {
01367      case SIZE_MINIMIZED:   /* Minimierung -> keine Aktion notwendig */
01368       break;
01369      case SIZE_RESTORED:    /*(Erst- oder Neu)Skalierung des Fensters */
01370      case SIZE_MAXIMIZED:       /* sichtbar: 0<=ix<=1023 / 0<=iy<=780 */
01371       SetMapMode (hTCSWindowDC, MM_ANISOTROPIC);
01372       SetViewportExtEx (hTCSWindowDC, width, -height, NULL);
```

```
01373       SetViewportOrgEx (hTCSWindowDC, 0, 0, NULL);
01374       /* Bei erneuter Änderung des Viewport geht die Auflösung verloren! */
01375     }
01376 }
01377
01378
01379
01380 void TCSWndProc_OnRbuttondown (HWND hWindow, BOOL DoubleClick, int MouseX,
01381                                      int MouseY, UINT ShftCtrlKeyMask)
01382 {
01383     ShowWindow (hTCSstatWindow, SW_SHOW);
01384     UpdateWindow(hTCSstatWindow);
01385 }
01386
01387
01388
01389 bool TCSWndProc_OnErasebkgnd (HWND hWindow, HDC hDC)
01390 {
01391 RECT ClientArea;
01392 HBRUSH hBack;
01393
01394     GetClientRect (hWindow, &ClientArea);
01395     DPtoLP (hDC, (LPPOINT)&ClientArea.left,2);
01396
01397     hBack= CreateSolidBrush (dwColorTable[TCSBackgroundColour]);
01398     FillRect(hTCSWindowDC, &ClientArea, hBack);
01399     #if !defined(__WIN32__) && !defined(_WIN32)
01400      DeleteBrush (hBack);
01401     #else
01402      DeleteObject (hBack);
01403     #endif
01404
01405     return false;
01406 }
01407
01408
01409
01410 bool TCSWndProc_OnCopyClipboard ()
01411 {
01412 #if (JOURNALTYP == 1)
01413  FTNINT iErr;
01414  HMETAFILE hmf;
01415  HDC hTCSNewMetaFileDC;
01416  HGLOBAL hGlobalMem;
01417  LPMETAFILEPICT lpMfp;
01418  HRGN    hWindowRegion;
01419 #elif (JOURNALTYP == 2)
01420  FTNINT iErr;
01421  HENHMETAFILE hmf, hmf1;
01422  ENHMETAHEADER emh ;
01423  HDC hTCSMetaFileDC1;
01424 #endif
01425
01426
01427 #if (JOURNALTYP == 1)
01428     hmf = CloseMetaFile (hTCSMetaFileDC);       /* Metafile für WM_PAINT */
01429
01430     hGlobalMem= GlobalAlloc(GMEM_MOVEABLE | GMEM_SHARE, sizeof(METAFILEPICT));
01431     if (hGlobalMem == NULL) {
01432      iErr= WRN_COPYNOMEM;
01433      TCSGraphicError (iErr,"");
01434      return false;                      /* Error: OutOfMemory -> ret */
01435     }
01436     lpMfp= (LPMETAFILEPICT) GlobalLock (hGlobalMem);
01437
01438     lpMfp->mm= MM_ANISOTROPIC;
01439     lpMfp->xExt= 0;             /* Keine Defaultgröße vorgeben */
01440     lpMfp->yExt= 0;             /* sonst in MM_HIMETRIC Device-Einheiten! */
01441
01442     hTCSNewMetaFileDC  = CreateMetaFile (NULL);
01443
01444     ScaleViewportExtEx (hTCSNewMetaFileDC, 1,1,-1,1,NULL);   // für Clipboard
01445
01446     hWindowRegion= CreateRectRgn(TCSrect.left, TCSrect.top, TCSrect.right,TCSrect.bottom); //
      rechts,oben
01447     SelectClipRgn (hTCSNewMetaFileDC, hWindowRegion); // nicht eingeschlossen
01448     #if !defined(__WIN32__) && !defined(_WIN32)
01449      DeleteRgn (hWindowRegion); // Resource freigeben
01450     #else
01451      DeleteObject (hWindowRegion);
01452     #endif
01453
01454     PlayMetaFile (hTCSNewMetaFileDC, hmf);
01455
01456     lpMfp->hMF= CloseMetaFile (hTCSNewMetaFileDC);
01457
01458     GlobalUnlock(hGlobalMem);
```

```
01459
01460     hTCSNewMetaFileDC  = CreateMetaFile (NULL); /* 16bit Windows Metafile */
01461     PlayMetaFile (hTCSNewMetaFileDC, hmf);      /* für neues Journalfile */
01462     DeleteMetaFile (hmf);                       /* alter Status Bildschirm */
01463     hTCSMetaFileDC = hTCSNewMetaFileDC;         /* bereit Weiterzeichnen */
01464
01465     if (!OpenClipboard (hTCSWindow)) {          /* Error: Clipboard locked */
01466      GlobalFree (hGlobalMem);
01467      iErr= WRN_COPYLOCK;
01468      TCSGraphicError (iErr,"");
01469      return false;
01470     }
01471     EmptyClipboard ();
01472     SetClipboardData (CF_METAFILEPICT, hGlobalMem);
01473     CloseClipboard ();  /* Jetzt GlobalFree() NICHT mehr aufrufen */
01474
01475 #elif (JOURNALTYP == 2)
01476     hmf = CloseEnhMetaFile (hTCSMetaFileDC);    /* Metafile für WM_PAINT */
01477     hmf1 = CopyEnhMetaFile (hmf, NULL) ;
01478     if (!OpenClipboard (hTCSWindow)) {          /* Error: Clipboard locked */
01479      iErr= WRN_COPYLOCK;
01480      TCSGraphicError (iErr,"");
01481      return false;
01482     }
01483     EmptyClipboard () ;
01484     SetClipboardData (CF_ENHMETAFILE, hmf1) ;
01485     CloseClipboard () ;
01486
01487     GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
01488     hTCSMetaFileDC1  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
01489                 _T("TCS for Windows\0Journalfile created by CopyClipboard\0"));
01490     SetMapMode (hTCSMetaFileDC1, MM_ANISOTROPIC);
01491     SetViewportExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
01492     SetViewportOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);
01493     SetWindowExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
01494     SetWindowOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);
01495
01496     SetBkMode (hTCSMetaFileDC, TRANSPARENT );
01497     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
01498
01499     PlayEnhMetaFile (hTCSMetaFileDC1, hmf, &TCSrect); // neues Journal
01500
01501     DeleteEnhMetaFile (hmf);                      // alter Status Bildschirm
01502     hTCSMetaFileDC = hTCSMetaFileDC1;             // bereit zum Weiterzeichnen
01503
01504     SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
01505     SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
01506     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
01507     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
01508
01509     #if !defined(__WIN32__) && !defined(_WIN32)
01510      SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
01511     #else
01512      SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
01513     #endif
01514     SetBkMode (hTCSMetaFileDC, TRANSPARENT );   // Metafile weitergegeben !
01515     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); // CP
01516     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
01517     #if !defined(__WIN32__) && !defined(_WIN32)
01518      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
01519     #else
01520      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
01521     #endif
01522
01523 #endif
01524
01525     return true;
01526 }
01527
01528
01529
01530 LRESULT CALLBACK EXPORT16 TCSWndProc(HWND hWindow, UINT Message,
01531                              WPARAM wParam, LPARAM lParam)
01532 {
01533     switch( Message ) {
01534      HANDLE_MSG(hWindow, WM_PAINT, TCSWndProc_OnPaint);
01535      HANDLE_MSG(hWindow, WM_RBUTTONDOWN, TCSWndProc_OnRbuttondown);
01536      HANDLE_MSG(hWindow, WM_SIZE, TCSWndProc_OnSize);
01537      HANDLE_MSG(hWindow, WM_ERASEBKGND, TCSWndProc_OnErasebkgnd);
01538      case WM_SYSCOMMAND:
01539       if (wParam == TCS_WM_COPY) {
01540        #ifdef trace_calls
01541         MessageBox(NULL, "WM_SYSCOMMAND (TCS_WM_COPY)",
01542                         "Internal Information GRAPH2D - TCSwindowProc",
01543                          MB_OK | MB_ICONINFORMATION);
01544        #endif
01545        TCSWndProc_OnCopyClipboard ();
```

```
01546        break;
01547       } else {
01548        return DefWindowProc( hWindow, Message, wParam, lParam );
01549       }
01550      case WM_CLOSE: // Schliessen des Graphikfensters nicht zulassen! Meldung
01551       break;        // kann trotz Menuesperre über <ALT><F4> erzeugt werden
01552      case WM_ACTIVATEAPP: // Neuzeichnen wg. Fensterminimierung fremde Appl.
01553       UpdateWindow (hWindow);
01554       return 0;
01555      default:
01556       return DefWindowProc( hWindow, Message, wParam, lParam );
01557     }
01558     return 0;
01559 }
01560
01561
01562
01563 /*
01564 ------------------- Event Handler Statusfenster ------------------------
01565 */
01566
01567
01568
01569 void TCSstatWndProc_OnPaint (HWND hWindow)
01570 {
01571 int i;
01572 PAINTSTRUCT ps;
01573
01574     BeginPaint (hWindow, &ps);
01575     #if !defined(__WIN32__) && !defined(_WIN32)
01576      SelectFont (ps.hdc, hTCSSysFont);      // Aktuellen Zeichenstatus an
01577     #else
01578      SelectObject (ps.hdc, hTCSSysFont);      // Aktuellen Zeichenstatus an
01579     #endif
01580     SetMapMode (ps.hdc, MM_TEXT);
01581     SetWindowOrgEx (ps.hdc, 0,TCSstatOrgY*TextLineHeight, NULL);
01582     for (i=0; i <= TCSstatRow; i++ )
01583      TextOut (ps.hdc, 0, i*TextLineHeight, TCSstatTextBuf[i],
01584                                          _tcslen (TCSstatTextBuf[i]));
01585     EndPaint( hWindow, &ps );
01586 }
01587
01588
01589
01590 void TCSstatWndProc_OnKillfocus (HWND hWindow, HWND hNewWindow)
01591 {
01592     if (TCSStatWindowAutomatic) ShowWindow (hWindow, SW_HIDE);
01593 }
01594
01595
01596
01597 void TCSstatWndProc_OnGetminmaxinfo (HWND hWindow, MINMAXINFO FAR* lpMinMaxInfo)
01598 /* Beschränkung User-erzeugbare Fenstergröße */
01599 {
01600     lpMinMaxInfo -> ptMaxSize.x = GetSystemMetrics (SM_CXMAXIMIZED);
01601     lpMinMaxInfo -> ptMaxSize.y = (int) (TCS_REL_CHR_SPACE*TextLineHeight) +
01602                                   STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
01603     lpMinMaxInfo -> ptMaxPosition.x = 0;
01604     #if !defined(__WIN32__) && !defined(_WIN32)
01605      lpMinMaxInfo -> ptMaxPosition.y = GetSystemMetrics (SM_CYFULLSCREEN) -
01606                                   STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
01607     #else
01608      lpMinMaxInfo -> ptMaxPosition.y = GetSystemMetrics (SM_CYMAXIMIZED) -
01609                                   (lpMinMaxInfo -> ptMaxSize.y);
01610     #endif
01611     lpMinMaxInfo -> ptMinTrackSize.x = GetSystemMetrics (SM_CXMINTRACK);
01612     lpMinMaxInfo -> ptMinTrackSize.y = GetSystemMetrics (SM_CYMINTRACK);
01613     lpMinMaxInfo -> ptMaxTrackSize.x = GetSystemMetrics (SM_CXMAXIMIZED);
01614     lpMinMaxInfo -> ptMaxTrackSize.y = STAT_ADDLINES*TextLineHeight+
01615                                   (lpMinMaxInfo -> ptMaxSize.y);
01616 }
01617
01618
01619
01620 void TCSstatWndProc_OnVScroll (HWND hWindow, HWND hNewWindow, WPARAM wParam,
01621                                                               LPARAM lParam)
01622 {
01623     switch (wParam) {
01624      case SB_LINEUP:
01625       TCSstatScrollY --;
01626       if (TCSstatScrollY < 0) TCSstatScrollY=0;
01627       break;
01628      case SB_LINEDOWN:
01629       TCSstatScrollY ++;
01630       if (TCSstatScrollY >= STAT_MAXROWS) TCSstatScrollY=STAT_MAXROWS-1;
01631       break;
01632      case SB_PAGEUP:
```

```
01633       TCSstatScrollY -= STAT_PAGESIZ;
01634       if (TCSstatScrollY < 0) TCSstatScrollY=0;
01635       break;
01636      case SB_PAGEDOWN:
01637       TCSstatScrollY += STAT_PAGESIZ;
01638       if (TCSstatScrollY >= STAT_MAXROWS) TCSstatScrollY=STAT_MAXROWS-1;
01639       break;
01640      case SB_THUMBPOSITION:
01641       TCSstatScrollY= (int) lParam;
01642       if (TCSstatScrollY < 0) TCSstatScrollY=0;
01643       if (TCSstatScrollY >= STAT_MAXROWS) TCSstatScrollY=STAT_MAXROWS-1;
01644       InvalidateRect (hWindow, NULL, true); /* ,ClientArea, EraseFlag */
01645       UpdateWindow (hWindow);               /* zwingend notwendig für Win16 */
01646       break;
01647     }
01648     ScrollWindow (hWindow, 0, (TCSstatOrgY-TCSstatScrollY)*TextLineHeight,
01649                                                                 NULL, NULL);
01650     SetScrollPos (hWindow, SB_VERT, TCSstatScrollY, true);
01651     TCSstatOrgY= TCSstatScrollY;
01652 }
01653
01654
01655
01656 LRESULT CALLBACK EXPORT16 TCSstatWndProc(HWND hWindow, UINT Message,
01657                             WPARAM wParam, LPARAM lParam)
01658 {
01659     switch( Message ) {
01660      HANDLE_MSG(hWindow, WM_PAINT, TCSstatWndProc_OnPaint);
01661      HANDLE_MSG(hWindow, WM_KILLFOCUS, TCSstatWndProc_OnKillfocus);
01662      HANDLE_MSG(hWindow, WM_GETMINMAXINFO, TCSstatWndProc_OnGetminmaxinfo);
01663      HANDLE_MSG(hWindow, WM_VSCROLL, TCSstatWndProc_OnVScroll);
01664      default:
01665       return DefWindowProc( hWindow, Message, wParam, lParam );
01666     }
01667     return 0;
01668 }
01669
01670
01671
01672 /*
01673 ---------------------- Userroutinen: Initialisierung -------------------
01674 */
01675
01676
01677
01678 extern void tcslev3 (FTNINT *SysLev)
01679
01680 {
01681     *SysLev= TCSLEV3SYS;
01682 }
01683
01684
01685
01686 #ifdef XMLSUPPORT
01687
01688 void XMLreadProgPar (const char * filname)
01689 {
01690 int ParserState;
01691 FILE *fp;
01692 mxml_node_t *tree;
01693
01694     fp = fopen(filname, "r");
01695     if (fp == NULL) {
01696      TCSGraphicError (ERR_XMLOPEN, filname);
01697     } else {
01698       ParserState= -1; // State= idle
01699       mxmlSetErrorCallback ((mxml_error_cb_t)sax_error_callback);
01700       tree = mxmlSAXLoadFile(NULL, fp, sax_type_callback, sax_callback, &ParserState);
01701       fclose(fp);
01702     }
01703 }
01704
01705 #endif    // Ende XML-Unterstützung
01706
01707
01708
01709 /*
01710 Defaultwerte sind bereits durch Compiler initialisiert worden. Hier werden nur
01711 die Parameter wiederhergestellt, die fuer einen erneuten Aufruf von initt nach
01712 finitt sinnvoll sind.
01713 */
01714
01715 void PresetProgPar ()
01716 {
01717     TCSDefaultLinCol= TCS_INIDEF_LINCOL;
01718     TCSDefaultTxtCol= TCS_INIDEF_TXTCOL;
01719     TCSDefaultBckCol= TCS_INIDEF_BCKCOL;
```

```
01720
01721     TCSwindowIniXrelpos= TCS_INIDEF_WINPOSX;
01722     TCSwindowIniYrelpos= TCS_INIDEF_WINPOSY;
01723     TCSwindowIniXrelsiz= TCS_INIDEF_WINSIZX;
01724     TCSwindowIniYrelsiz= TCS_INIDEF_WINSIZY;
01725
01726     TCSstatWindowIniXrelpos= TCS_INIDEF_STATPOSX;
01727     TCSstatWindowIniYrelpos= TCS_INIDEF_STATPOSY;
01728     TCSstatWindowIniXrelsiz= TCS_INIDEF_STATSIZX;
01729     TCSstatWindowIniYrelsiz= TCS_INIDEF_STATSIZY;
01730
01731     // Fensternamen werden nur durch winlbl vorher veraendert
01732
01733     // Hardcopyname und Zaehlerstand bleibt!
01734
01735     // Fehlermeldungen werden bei der Variablendefinition durch den Compiler initialisiert
01736 }
01737
01738
01739
01740 /*
01741 Anpassung der Dateinamen an die Laufzeitumgebung
01742 */
01743
01744 void CustomizeProgPar ()
01745 {
01746 // Absicherung der Definition der Programmparameter
01747 #if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN)
01748  #define TMPSTRLEN TCS_FILE_NAMELEN
01749 #else
01750  #define TMPSTRLEN TCS_WINDOW_NAMELEN
01751 #endif
01752
01753 int         iL;
01754 char        szTmpString[TMPSTRLEN];
01755 FTNSTRDESC  ftn_WorkString, o, n;
01756
01757 szTmpString[0]= '\0';
01758 n.addr= szTmpString; // Token bei Fonts werden geloescht
01759 n.len= TMPSTRLEN;
01760
01761 #ifdef XMLSUPPORT // Angabe von Dateinamen fuer Fonts bei Windows nicht moeglich
01762     o.addr= PROGDIRTOKEN; // Token %: loeschen
01763     o.len= strlen (o.addr);
01764     ftn_WorkString.len= TCS_FILE_NAMELEN; // Font Graphikfenster
01765     ftn_WorkString.addr= szTCSGraphicFont;
01766     o.addr= PROGDIRTOKEN; // Substring %: loeschen
01767     o.len= strlen (o.addr);
01768     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01769                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01770                 CALLFTNSTRL(ftn_WorkString)
01771                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01772
01773     ftn_WorkString.addr= szTCSSysFont; // Font Statusfenster
01774     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01775                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01776                 CALLFTNSTRL(ftn_WorkString)
01777                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01778
01779
01780     o.addr= INIFILEXTTOKEN; // Token .% loeschen
01781     o.len= strlen (o.addr); // Font Statusfenster
01782     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01783                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01784                 CALLFTNSTRL(ftn_WorkString)
01785                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01786
01787     ftn_WorkString.addr= szTCSGraphicFont; // Font Graphikfenster
01788     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01789                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01790                 CALLFTNSTRL(ftn_WorkString)
01791                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01792 #endif // Ende XML-Unterstützung, in *.INI und Registry keine Verwendung Token
01793
01794     if (strlen(szTCSWindowName) == 0) { // '/0' durch WINLBL -> Default
01795         strncpy(szTCSWindowName, TCS_WINDOW_NAME, TCS_WINDOW_NAMELEN);
01796     }
01797     if (strlen(szTCSstatWindowName) == 0) {
01798         strncpy(szTCSstatWindowName, TCS_STATWINDOW_NAME, TCS_WINDOW_NAMELEN);
01799     }
01800
01801     o.addr= PROGDIRTOKEN; // Substring %: vollstaendiger Programmname
01802     o.len= strlen (o.addr);
01803     #if !defined(__WIN32__) && !defined(_WIN32)   /* nicht bei DLL möglich */
01804      #if defined __WATCOMC__
01805       iL= 0;            /* Argument 0= Voller Programmname mit Directory */
01806       iL= igetarg ((FTNINT *) &iL, &n);
```

```
01807      #else
01808       #error "Kompilation für 16bit Windows nur mit Watcom-Compiler möglich"
01809      #endif
01810     #else   /* alternativ nur Win32: hInst=NULL: prozesserzeugende Instanz */
01811      iL= GetModuleFileName(NULL, n.addr, n.len);
01812     #endif
01813     if (iL <= 0) {
01814      n.addr[0]= (FTNCHAR) 0; /* kein Programmnamen bekannt */
01815     }
01816     ftn_WorkString.len= TCS_WINDOW_NAMELEN; // Ersatz %: im Graphikfenster
01817     ftn_WorkString.addr= szTCSWindowName;
01818     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01819                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01820                 CALLFTNSTRL(ftn_WorkString)
01821                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01822    ftn_WorkString.addr= szTCSstatWindowName; // Ersatz %: im Statusfenster
01823    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01824                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01825                 CALLFTNSTRL(ftn_WorkString)
01826                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01827
01828 // Absicherung TMPSTRLEN nicht mehr benoetigt
01829 #undef TMPSTRLEN
01830 }
01831
01832
01833
01834
01835 extern void winlbl (FTNSTRPAR * PloWinNam, FTNSTRPAR * StatWinNam,
01836                                            FTNSTRPAR *IniFilNam
01837                                            FTNSTRPAR_TAIL(PloWinNam)
01838                                            FTNSTRPAR_TAIL(StatWinNam)
01839                                            FTNSTRPAR_TAIL(IniFilNam)      )
01840
01841 {
01842
01843 #if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN)
01844  #define TMPSTRLREN TCS_FILE_NAMELEN
01845 #else
01846  #define TMPSTRLREN TCS_WINDOW_NAMELEN
01847 #endif
01848
01849 FTNCHARLEN  i, iL;
01850 FTNCHAR     szTmpString[TMPSTRLREN], szTmpString1[TMPSTRLREN];
01851 FTNCHAR *   iAt;
01852 FTNSTRDESC  o, n, ftn_WorkString;
01853
01854
01855     iL= min(FTNSTRPARL(PloWinNam), TMPSTRLREN-1);    // Name des Grahikfensters
01856     _tcsncpy(szTmpString, FTNSTRPARA(PloWinNam),iL);
01857     szTmpString[iL]= (FTNCHAR) 0; // Fortranstring evtl. ohne \0
01858     iL= min (_tcslen (szTmpString), TCS_WINDOW_NAMELEN-1);
01859     if (iL > 0) {
01860      _tcsncpy( szTCSWindowName, szTmpString, iL);
01861      szTCSWindowName[iL]= (FTNCHAR) 0;
01862     }
01863
01864     iL= min(FTNSTRPARL(StatWinNam), TMPSTRLREN-1);   // Name des Statusfensters
01865     _tcsncpy(szTmpString, FTNSTRPARA(StatWinNam), iL);
01866     szTmpString[iL]= (FTNCHAR) 0; // Fortranstring evtl. ohne \0
01867     iL= min (_tcslen (szTmpString), TCS_WINDOW_NAMELEN-1);
01868     if (iL > 0) {
01869      _tcsncpy( szTCSstatWindowName, szTmpString, iL);
01870      szTCSstatWindowName[iL]= (FTNCHAR) 0;
01871     }
01872
01873     iL= min(FTNSTRPARL(IniFilNam), TMPSTRLREN-1); // Name Initialisierungsdatei
01874     _tcsncpy(szTmpString, FTNSTRPARA(IniFilNam), iL);
01875     szTmpString[iL]= (FTNCHAR) 0; // Fortranstring evtl. ohne \0
01876
01877     iL= min (_tcslen (szTmpString), TCS_FILE_NAMELEN-1);
01878
01879     if (iL > 0) {
01880      _tcsncpy( szTCSIniFile, szTmpString, iL);
01881      szTCSIniFile[iL]= (FTNCHAR) 0;
01882
01883      iAt= _tcsstr (szTCSIniFile, _T("@")); // Section Level0?
01884      if (iAt != 0) {
01885       _tcsncpy(szTCSsect0, &iAt[1], iL); // Abspeichern
01886       iAt[0]= (FTNCHAR) 0; // Abschneiden von @Section0 in szTCSIniFile
01887      }
01888
01889      ftn_WorkString.len= TCS_FILE_NAMELEN;
01890      ftn_WorkString.addr= szTCSIniFile;
01891
01892      n.len= _tcslen (INIFILEXT);
01893      n.addr= INIFILEXT;
```

```
01894       o.len= _tcslen (INIFILEXTTOKEN);
01895       o.addr= INIFILEXTTOKEN;
01896       SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01897                   CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01898                   CALLFTNSTRL(ftn_WorkString)
01899                   CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01900
01901       n.len= TCS_FILE_NAMELEN;
01902       n.addr= (FTNCHAR *) &szTmpString1;
01903       o.len= _tcslen (PROGDIRTOKEN);
01904       o.addr= PROGDIRTOKEN;
01905
01906       _tcsncpy (szTmpString1, szTCSIniFile, TCS_FILE_NAMELEN);
01907       _tcsrev (szTmpString1); // Abfrage Ende des Strings, Extension rueckwaerts!
01908
01909       if (_tcsnicmp (szTmpString1, _T("GER."),4) == 0) { // Filename endet .REG?
01910        n.addr[0]= (FTNCHAR) 0; /* keine Directory sinnvoll -> Token loeschen */
01911       } else {
01912        #if !defined(__WIN32__) && !defined(_WIN32)   /* nicht bei DLL möglich */
01913         #if defined __WATCOMC__
01914          iL= 0;            /* Argument 0= Voller Programmname mit Directory */
01915          iL= igetarg ((FTNINT *) &iL, &n);
01916         #else
01917          #error "Kompilation für 16bit Windows nur mit Watcom-Compiler möglich"
01918         #endif
01919        #else   /* alternativ nur Win32: hInst=NULL: prozesserzeugende Instanz */
01920         iL= GetModuleFileName(NULL, n.addr, n.len);
01921        #endif
01922        if (iL>0) {
01923         for (i=iL-1;(n.addr[i]!= (FTNCHAR) '\\' ) || (i==0); i--);
01924         i++;
01925         if (i < n.len) n.addr[i]= (FTNCHAR) 0; /* jetzt: Programmname entfernt */
01926        } else {
01927         n.addr[0]= (FTNCHAR) 0; /* keine Directory bekannt */
01928        }
01929       }
01930       SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01931                   CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01932                   CALLFTNSTRL(ftn_WorkString)
01933                   CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01934
01935     }
01936
01937 #undef TMPSTRLREN
01938 }
01939
01940
01941
01942 extern void initt1 (HINSTANCE *hParentInstance, HWND *hParentWindow)
01943 {
01944 int         nCmdShow, iX,iY, iSizeX, iSizeY;
01945 DWORD       FirstShow;
01946 WNDCLASS    TCSWndClass;
01947 HMENU       SysMenu;
01948 TCHAR       szTmpString[TCS_FILE_NAMELEN];
01949 TEXTMETRIC  lpTM;
01950
01951 #if defined(__WIN32__) || defined(_WIN32) || defined (REGSUPPORT)
01952  DWORD       retValue;
01953  LPVOID      lpMsgBuf;
01954 #endif
01955
01956 #if defined(REGSUPPORT)
01957  HKEY hSysrootKey, hRootKey,hSectionKey;
01958  TCHAR szRootKey[TCS_FILE_NAMELEN]= _T("Software\\"); // +IniFilename ohne Ext.
01959  TCHAR szSectionKey[TCS_FILE_NAMELEN];
01960  TCHAR szTmpString2[TCS_FILE_NAMELEN];
01961  DWORD dwSectionKeyLen;
01962  DWORD TmpStringLen, TmpStringLen2;
01963  DWORD i, j;
01964  DWORD retValue2;
01965 #endif
01966
01967 #if (JOURNALTYP == 2)
01968  RECT   screenrect;
01969  int iWidthMM, iHeightMM, iWidthPixel, iHeightPixel;
01970 #elif (JOURNALTYP == 3)
01971  struct xJournalEntry_typ * xJournalEntry;
01972 #endif
01973
01974
01975     if (TCSinitialized) return;   /* Bereits initialisiert */
01976     TCSinitialized= true;
01977
01978     PresetProgPar (); // Nach 2.Aufruf: nur Farben keine Namen wiederherstellen
01979
01980     if ( _tcslen (szTCSIniFile) <= 4) { // Extension muss angegeben werden!
```

```
01981       _tcsncpy (szTCSIniFile, _T("TooShortInitfilename"), TCS_FILE_NAMELEN);
01982     }
01983
01984     _tcsncpy (szTmpString, szTCSIniFile, TCS_FILE_NAMELEN);
01985     _tcsrev (szTmpString); // Abfrage Ende des Strings, Extension rueckwaerts!
01986
01987     /*
01988         Falls Extension des Ini-Files .XML: XML-Parser
01989     */
01990 #if defined(XMLSUPPORT)
01991     if (_tcsnicmp (szTmpString, _T("LMX."),4) == 0) { // Filename endet .XML?
01992       XMLreadProgPar (szTCSIniFile);
01993     } else  // endif Initialisierung ueber *.xml
01994 #endif
01995
01996
01997     /*
01998         Falls Extension des Ini-Files .REG: Auswertung der Registry
01999     */
02000 #if defined(REGSUPPORT)
02001     if (_tcsnicmp (szTmpString, _T("GER."),4) == 0) { // Filename endet .REG?
02002       _tcsncat (szRootKey, szTCSIniFile, _tcslen (szTCSIniFile)-4);
02003       for (hSysrootKey= HKEY_LOCAL_MACHINE; hSysrootKey!= NULL; ) {
02004         if (!RegOpenKeyEx( hSysrootKey, szRootKey, 0, KEY_READ, &hRootKey)) {
02005           szSectionKey[0]= (FTNCHAR) 0; // 1. Durchlauf ohne Section
02006           for (i = 0, retValue= false; !retValue; i++) {
02007             if (!RegOpenKeyEx( hRootKey, szSectionKey, 0, KEY_READ, &hSectionKey)) {
02008               for (j = 0, retValue2 = false; !retValue2; j++) {
02009                 TmpStringLen= TCS_FILE_NAMELEN;          // Codewort
02010                 TmpStringLen2= TCS_FILE_NAMELEN;         // Wert des Codewortes
02011                 retValue2= RegEnumValue(hSectionKey, j, szTmpString, &TmpStringLen,
02012                                         NULL, NULL, (LPBYTE) szTmpString2, &TmpStringLen2);
02013                 if (!retValue2) StoreIni (szSectionKey,szTmpString, szTmpString2);
02014               }
02015               RegCloseKey(hSectionKey);
02016             }
02017             dwSectionKeyLen= TCS_FILE_NAMELEN;
02018             retValue= RegEnumKeyEx(hRootKey, i, szSectionKey, &dwSectionKeyLen,
02019                                                          NULL,  NULL, NULL, NULL);
02020           }
02021           RegCloseKey(hRootKey);
02022         }
02023         if (hSysrootKey == HKEY_LOCAL_MACHINE) {
02024           hSysrootKey= HKEY_CURRENT_USER;
02025         } else if (hSysrootKey == HKEY_CURRENT_USER) {
02026           hSysrootKey= NULL;
02027         }
02028       } // 2x: HKEY_LOCAL_MACHINE, HKEY_CURRENT_USER (ueberschreibt LOCAL_MACH.)
02029     } else  // endif Registryinitialisierung
02030 #endif
02031
02032     /*
02033         Falls Extension des Ini-Files .INI: Auswertung der Initialisierungsdatei
02034     */
02035
02036     if (_tcsnicmp (szTmpString, _T("INI."),4) == 0) { // Filename endet .INI?
02037       if (_tcslen(szTCSWindowName)==0)
02038         GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_WINNAM,
02039         TCS_WINDOW_NAME, szTCSWindowName, TCS_WINDOW_NAMELEN, szTCSIniFile);
02040       if (_tcslen(szTCSstatWindowName)==0)
02041         GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_STATNAM,
02042         TCS_STATWINDOW_NAME,szTCSstatWindowName,TCS_WINDOW_NAMELEN,szTCSIniFile);
02043
02044       GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_MAINWINNAM,
02045         TCS_MAINWINDOW_NAME,szTCSMainWindowName,TCS_WINDOW_NAMELEN,szTCSIniFile);
02046
02047       GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_HDCNAM, TCS_HDCFILE_NAME,
02048                               szTCSHardcopyFile,TCS_FILE_NAMELEN,szTCSIniFile);
02049
02050
02051       GetPrivateProfileString (TCS_INISECT2,TCS_INIVAR_COPMEN,TCS_INIDEF_COPMEN,
02052                               szTCSMenuCopyText, STAT_MAXCOLUMNS, szTCSIniFile);
02053       GetPrivateProfileString (TCS_INISECT2,TCS_INIVAR_FONT,TCS_INIDEF_FONT,
02054                               szTCSGraphicFont, TCS_FILE_NAMELEN, szTCSIniFile);
02055       GetPrivateProfileString (TCS_INISECT2,TCS_INIVAR_SYSFONT,TCS_INIDEF_SYSFONT,
02056                               szTCSSysFont, TCS_FILE_NAMELEN, szTCSIniFile);
02057       GetPrivateProfileString(TCS_INISECT2,TCS_INIVAR_ICONNAM, TCS_ICONFILE_NAME,
02058                               szTCSIconFile,TCS_FILE_NAMELEN,szTCSIniFile);
02059
02060       TCSwindowIniXrelpos= GetPrivateProfileInt (TCS_INISECT2,
02061                           TCS_INIVAR_WINPOSX, TCS_INIDEF_WINPOSX, szTCSIniFile);
02062       TCSwindowIniYrelpos= GetPrivateProfileInt (TCS_INISECT2,
02063                           TCS_INIVAR_WINPOSY, TCS_INIDEF_WINPOSY, szTCSIniFile);
02064       TCSwindowIniXrelsiz= GetPrivateProfileInt (TCS_INISECT2,
02065                           TCS_INIVAR_WINSIZX, TCS_INIDEF_WINSIZX, szTCSIniFile);
02066       TCSwindowIniYrelsiz= GetPrivateProfileInt (TCS_INISECT2,
02067                           TCS_INIVAR_WINSIZY, TCS_INIDEF_WINSIZY, szTCSIniFile);
```

```
02068
02069       TCSstatWindowIniXrelpos= GetPrivateProfileInt (TCS_INISECT2,
02070                           TCS_INIVAR_STATPOSX, TCS_INIDEF_STATPOSX, szTCSIniFile);
02071       TCSstatWindowIniYrelpos= GetPrivateProfileInt (TCS_INISECT2,
02072                           TCS_INIVAR_STATPOSY, TCS_INIDEF_STATPOSY, szTCSIniFile);
02073       TCSstatWindowIniXrelsiz= GetPrivateProfileInt (TCS_INISECT2,
02074                           TCS_INIVAR_STATSIZX, TCS_INIDEF_STATSIZX, szTCSIniFile);
02075       TCSstatWindowIniYrelsiz= GetPrivateProfileInt (TCS_INISECT2,
02076                           TCS_INIVAR_STATSIZY, TCS_INIDEF_STATSIZY, szTCSIniFile);
02077
02078       TCSDefaultLinCol= GetPrivateProfileInt (TCS_INISECT2,
02079                           TCS_INIVAR_LINCOL,TCS_INIDEF_LINCOL, szTCSIniFile);
02080       TCSDefaultTxtCol= GetPrivateProfileInt (TCS_INISECT2,
02081                           TCS_INIVAR_TXTCOL,TCS_INIDEF_TXTCOL, szTCSIniFile);
02082       TCSDefaultBckCol= GetPrivateProfileInt (TCS_INISECT2,
02083                           TCS_INIVAR_BCKCOL,TCS_INIDEF_BCKCOL, szTCSIniFile);
02084
02085
02086       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_HDCOPN,TCS_INIDEF_HDCOPN,
02087                     szTCSErrorMsg[WRN_HDCFILOPN], STAT_MAXCOLUMNS, szTCSIniFile);
02088       TCSErrorLev[WRN_HDCFILOPN]= GetPrivateProfileInt (TCS_INISECT3,
02089                           TCS_INIVAR_HDCOPNL,TCS_INIDEF_HDCOPNL, szTCSIniFile);
02090
02091       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_HDCWRT,TCS_INIDEF_HDCWRT,
02092                     szTCSErrorMsg[WRN_HDCFILWRT], STAT_MAXCOLUMNS, szTCSIniFile);
02093       TCSErrorLev[WRN_HDCFILWRT]= GetPrivateProfileInt (TCS_INISECT3,
02094                           TCS_INIVAR_HDCWRTL,TCS_INIDEF_HDCWRTL, szTCSIniFile);
02095
02096       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_HDCINT,TCS_INIDEF_HDCINT,
02097                     szTCSErrorMsg[WRN_HDCINTERN], STAT_MAXCOLUMNS, szTCSIniFile);
02098       TCSErrorLev[WRN_HDCFILWRT]= GetPrivateProfileInt (TCS_INISECT3,
02099                           TCS_INIVAR_HDCINTL,TCS_INIDEF_HDCINTL, szTCSIniFile);
02100
02101       GetPrivateProfileString (TCS_INISECT3, TCS_INIVAR_USR,TCS_INIDEF_USR,
02102                           szTCSErrorMsg[MSG_USR], STAT_MAXCOLUMNS, szTCSIniFile);
02103       TCSErrorLev[MSG_USR]= GetPrivateProfileInt (TCS_INISECT3, TCS_INIVAR_USRL,
02104                           TCS_INIDEF_USRL, szTCSIniFile);
02105
02106       GetPrivateProfileString (TCS_INISECT3, TCS_INIVAR_HDCACT,TCS_INIDEF_HDCACT,
02107                           szTCSErrorMsg[MSG_HDCACT], STAT_MAXCOLUMNS, szTCSIniFile);
02108       TCSErrorLev[MSG_HDCACT]= GetPrivateProfileInt (TCS_INISECT3,
02109                           TCS_INIVAR_HDCACTL,TCS_INIDEF_HDCACTL, szTCSIniFile);
02110
02111       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_USRWRN,TCS_INIDEF_USRWRN,
02112                     szTCSErrorMsg[WRN_USRPRESSANY],STAT_MAXCOLUMNS,szTCSIniFile);
02113       TCSErrorLev[WRN_USRPRESSANY]= GetPrivateProfileInt (TCS_INISECT3,
02114                           TCS_INIVAR_USRWRNL,TCS_INIDEF_USRWRNL, szTCSIniFile);
02115
02116       GetPrivateProfileString (TCS_INISECT3, TCS_INIVAR_EXIT,TCS_INIDEF_EXIT,
02117                           szTCSErrorMsg[ERR_EXIT], STAT_MAXCOLUMNS, szTCSIniFile);
02118       TCSErrorLev[ERR_EXIT]= GetPrivateProfileInt (TCS_INISECT3,
02119                           TCS_INIVAR_EXITL,TCS_INIDEF_EXITL, szTCSIniFile);
02120
02121       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_COPMEM,TCS_INIDEF_COPMEM,
02122                     szTCSErrorMsg[WRN_COPYNOMEM], STAT_MAXCOLUMNS, szTCSIniFile);
02123       TCSErrorLev[WRN_COPYNOMEM]= GetPrivateProfileInt (TCS_INISECT3,
02124                           TCS_INIVAR_COPMEML,TCS_INIDEF_COPMEML, szTCSIniFile);
02125
02126       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_COPLCK,TCS_INIDEF_COPLCK,
02127                      szTCSErrorMsg[WRN_COPYLOCK], STAT_MAXCOLUMNS, szTCSIniFile);
02128       TCSErrorLev[WRN_COPYLOCK]= GetPrivateProfileInt (TCS_INISECT3,
02129                           TCS_INIVAR_COPLCKL,TCS_INIDEF_COPLCKL, szTCSIniFile);
02130
02131       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUCREATE,TCS_INIDEF_JOUCREATE,
02132                      szTCSErrorMsg[WRN_JOUCREATE], STAT_MAXCOLUMNS, szTCSIniFile);
02133       TCSErrorLev[WRN_JOUCREATE]= GetPrivateProfileInt (TCS_INISECT3,
02134                           TCS_INIVAR_JOUCREATEL,TCS_INIDEF_JOUCREATEL, szTCSIniFile);
02135
02136       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUENTRY,TCS_INIDEF_JOUENTRY,
02137                     szTCSErrorMsg[WRN_JOUENTRY], STAT_MAXCOLUMNS, szTCSIniFile);
02138       TCSErrorLev[WRN_JOUENTRY]= GetPrivateProfileInt (TCS_INISECT3,
02139                           TCS_INIVAR_JOUENTRYL,TCS_INIDEF_JOUENTRYL, szTCSIniFile);
02140
02141       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUADD,TCS_INIDEF_JOUADD,
02142                     szTCSErrorMsg[WRN_JOUADD], STAT_MAXCOLUMNS, szTCSIniFile);
02143       TCSErrorLev[WRN_JOUADD]= GetPrivateProfileInt (TCS_INISECT3,
02144                           TCS_INIVAR_JOUADDL,TCS_INIDEF_JOUADDL, szTCSIniFile);
02145
02146       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUCLR,TCS_INIDEF_JOUCLR,
02147                     szTCSErrorMsg[WRN_JOUCLR], STAT_MAXCOLUMNS, szTCSIniFile);
02148       TCSErrorLev[WRN_JOUCLR]= GetPrivateProfileInt (TCS_INISECT3,
02149                           TCS_INIVAR_JOUCLRL,TCS_INIDEF_JOUCLRL, szTCSIniFile);
02150
02151       GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUUNKWN,TCS_INIDEF_JOUUNKWN,
02152                     szTCSErrorMsg[WRN_JOUUNKWN], STAT_MAXCOLUMNS, szTCSIniFile);
02153       TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
02154                           TCS_INIVAR_JOUUNKWNL,TCS_INIDEF_JOUUNKWNL, szTCSIniFile);
```

```
02155
02156
02157      GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_XMLPARSER,TCS_INIDEF_XMLPARSER,
02158                    szTCSErrorMsg[ERR_XMLPARSER], STAT_MAXCOLUMNS, szTCSIniFile);
02159      TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
02160                      TCS_INIVAR_XMLPARSERL,TCS_INIDEF_XMLPARSERL, szTCSIniFile);
02161
02162      GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_XMLOPEN,TCS_INIDEF_XMLOPEN,
02163                    szTCSErrorMsg[ERR_XMLOPEN], STAT_MAXCOLUMNS, szTCSIniFile);
02164      TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
02165                      TCS_INIVAR_XMLOPENL,TCS_INIDEF_XMLOPENL, szTCSIniFile);
02166
02167      GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_USR2,TCS_INIDEF_USR2,
02168                    szTCSErrorMsg[MSG_USR2], STAT_MAXCOLUMNS, szTCSIniFile);
02169      TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
02170                      TCS_INIVAR_USR2L,TCS_INIDEF_USR2L, szTCSIniFile);
02171
02172      GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_INI2,TCS_INIDEF_INI2,
02173                    szTCSErrorMsg[WRN_INI2], STAT_MAXCOLUMNS, szTCSIniFile);
02174      TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
02175                      TCS_INIVAR_INI2L,TCS_INIDEF_INI2L, szTCSIniFile);
02176
02177     } // endif Initialisierung ueber *.ini
02178
02179
02180     CustomizeProgPar (); // Ersatz %: durch Programmverzeichnis
02181
02182     /*
02183     Übernahme der durch den Nutzer angepassten Initialisierungsdaten
02184     */
02185
02186     TKTRNX.iLinCol= TCSDefaultLinCol;
02187     TKTRNX.iTxtCol= TCSDefaultTxtCol;
02188     TKTRNX.iBckCol= TCSDefaultBckCol;
02189
02190     /*
02191         Ermittlung der Instanz des Processes
02192     */
02193
02194     hTCSInst= *hParentInstance; // In Hauptprogramm durch INITT ermittelt
02195     hOwnerWindow= *hParentWindow;
02196
02197     if (_tcscmp(szTCSMainWindowName,_T("%:")) == 0) {
02198      _tcsncpy( szTCSMainWindowName,GetCommandLine(), STAT_MAXCOLUMNS);
02199     }
02200
02201     CreateMainWindow_IfNecessary (&hTCSInst,&hOwnerWindow,szTCSMainWindowName);
02202
02203     *hParentWindow= hOwnerWindow;   // Publizieren evtl. neues Handle DLL->Main
02204
02205     /*
02206         Ermittlung allgemeiner systemspezifischer Parameter
02207     */
02208
02209     TextLineHeight= GetSystemMetrics (SM_CYMENU); /* Höhe Menüeintrag */
02210     TCSCharHeight= (int)(TCS_REL_CHR_HEIGHT* (float)(HiRes(TextLineHeight)));
02211
02212     TCSBackgroundColour= TKTRNX.iBckCol;
02213
02214     TKTRNX.kStCol = STAT_MAXCOLUMNS;
02215     TKTRNX.iMouse = 3; /* werden z.Zt. bei DCURSR () ausgewertet */
02216
02217     /*
02218         Erzeugung des Graphikfensters
02219     */
02220
02221     TCSWndClass.style          = CS_OWNDC | CS_HREDRAW | CS_VREDRAW;
02222     TCSWndClass.lpfnWndProc    = TCSWndProc;
02223     TCSWndClass.cbClsExtra     = 0;
02224     TCSWndClass.cbWndExtra     = 0;
02225     TCSWndClass.hInstance      = hTCSInst;
02226
02227     #if (defined(__WIN32__) || defined(_WIN32))
02228      if (_tcslen (szTCSIconFile) != 0) {
02229       TCSWndClass.hIcon         = LoadImage (NULL, szTCSIconFile,
02230                                       IMAGE_ICON,0,0,LR_LOADFROMFILE);
02231      } else {
02232       TCSWndClass.hIcon         = LoadIcon (hTCSInst, TCS_WINDOW_ICON);
02233                                 /* Falls Icon nicht definiert->LoadIcon=NULL */
02234      }
02235     #else
02236      TCSWndClass.hIcon         = LoadIcon (hTCSInst, TCS_WINDOW_ICON);
02237     #endif
02238
02239     TCSWndClass.hCursor        = LoadCursor(NULL, IDC_ARROW);
02240     TCSWndClass.hbrBackground  = NULL; /* Erase-Handler, Brush unnötig */
02241     TCSWndClass.lpszMenuName   = NULL;
```

```
02242     TCSWndClass.lpszClassName  = TCS_WINDOWCLASS;
02243
02244      /* Register the window class. Fail: most probable UNICODE on win98 */
02245     if (!RegisterClass (&TCSWndClass)) {
02246      #if defined(__WIN32__) || defined(_WIN32)
02247       retValue= GetLastError(); // win32-Funktion
02248 //    if (retValue == ERROR_CLASS_ALREADY_EXISTS) {
02249 //     Hier bei Bedarf Fehlerbehandlung einführen
02250 //    } else {
02251       FormatMessage(
02252         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
02253         NULL,
02254         retValue,
02255         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
02256         (LPTSTR) &lpMsgBuf,
02257         0,
02258         NULL
02259       );
02260       MessageBox (NULL, lpMsgBuf, szTCSWindowName, MB_ICONSTOP);
02261       LocalFree( lpMsgBuf ); // Free the buffer
02262 //    } // Ende der Fehlerbehandlung
02263      #else // rudimentaere Fehlerbehandlung 16bit Windows
02264       MessageBox (NULL, _T("Window Class not registered"),
02265                                     szTCSWindowName, MB_ICONSTOP);
02266      #endif
02267      return;
02268     }
02269
02270     if ((TCSwindowIniXrelsiz < 100) || (TCSwindowIniYrelsiz < 100) ) {
02271      nCmdShow= SW_SHOWNORMAL; /* Achtung, int = 2Byte bei WIN16!!! */
02272      iX= (int) ( ( (long int) TCSwindowIniXrelpos *
02273                    (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
02274      iY= (int) ( ( (long int) TCSwindowIniYrelpos *
02275                    (long int) GetSystemMetrics (SM_CYMAXIMIZED)) / 100);
02276      iSizeX= (int) ( ( (long int) TCSwindowIniXrelsiz *
02277                    (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
02278      iSizeY= (int) ( ( (long int) TCSwindowIniYrelsiz *
02279                    (long int) GetSystemMetrics (SM_CYMAXIMIZED)) / 100);
02280     } else {
02281      nCmdShow= SW_SHOWMAXIMIZED;
02282      iX= 0;
02283      iY= 0;
02284      iSizeX= GetSystemMetrics (SM_CXMAXIMIZED);
02285      iSizeY= GetSystemMetrics (SM_CYMAXIMIZED);
02286     }
02287
02288     hTCSWindow = CreateWindow(TCS_WINDOWCLASS, szTCSWindowName,
02289                         WS_OVERLAPPEDWINDOW,
02290                         iX, iY,
02291                         iSizeX, iSizeY,
02292                         hOwnerWindow,
02293                         (HMENU) NULL,
02294                         (HINSTANCE) hTCSInst, (LPSTR) NULL);
02295
02296     if (hTCSWindow == NULL) return;
02297
02298     hTCSWindowDC = GetDC (hTCSWindow);
02299
02300     SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
02301     SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);
02302
02303 #if (JOURNALTYP == 1)
02304     hTCSMetaFileDC = CreateMetaFile (NULL); /* Memory-based 16bit Metafile */
02305     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
02306     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
02307     MoveToEx (hTCSMetaFileDC, 0, 0, NULL);  /* Cursorposition Neuzeichnen */
02308
02309 #elif (JOURNALTYP == 2)
02310     iWidthMM = GetDeviceCaps(hTCSWindowDC, HORZSIZE); // Bildschirmgroesse(mm)
02311     iHeightMM = GetDeviceCaps(hTCSWindowDC, VERTSIZE);
02312     iWidthPixel = GetDeviceCaps(hTCSWindowDC, HORZRES); // Bildschirm (Pixel)
02313     iHeightPixel = GetDeviceCaps(hTCSWindowDC, VERTRES);
02314
02315     screenrect.left= (TCSrect.left *iWidthMM *100)/iWidthPixel; // in .01 mm
02316     screenrect.top= (TCSrect.top *iHeightMM *100)/iHeightPixel;
02317     screenrect.right= (TCSrect.right *iWidthMM *100)/iWidthPixel; // right > left!
02318     screenrect.bottom= (TCSrect.bottom *iHeightMM *100)/iHeightPixel; // bottom > top!
02319
02320     hTCSMetaFileDC = CreateEnhMetaFile (hTCSWindowDC, NULL, &screenrect,
02321             _T("TCS for Windows\0Journalfile created by INITT\0" ));
02322
02323     SetMapMode (hTCSMetaFileDC, MM_ANISOTROPIC);
02324     SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
02325     SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
02326
02327     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
02328     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
```

```
02329
02330     MoveToEx (hTCSMetaFileDC, 0, 0, NULL);  /* Cursorposition Neuzeichnen */
02331 #endif
02332
02333     ShowWindow (hTCSWindow, nCmdShow);      /* Skalierung Viewport  */
02334     UpdateWindow(hTCSWindow);               /* in TCSWndProc_OnSize */
02335
02336     SysMenu = GetSystemMenu (hTCSWindow, FALSE); /* Systemmenu: kein Close */
02337     DeleteMenu (SysMenu, 6, MF_BYPOSITION);
02338     AppendMenu (SysMenu,MF_STRING,TCS_WM_COPY,szTCSMenuCopyText);  /* Copy */
02339
02340     TCSFontdefinition.lfHeight= TCSCharHeight; /* Höhe, Breite */
02341     TCSFontdefinition.lfWidth= 0;
02342     TCSFontdefinition.lfEscapement= 0; /* lfEscapement=lfOrientation */
02343     TCSFontdefinition.lfOrientation= 0;
02344     TCSFontdefinition.lfWeight= FW_NORMAL; /* Strichstärke */
02345     TCSFontdefinition.lfItalic= false;
02346     TCSFontdefinition.lfUnderline= false;
02347     TCSFontdefinition.lfStrikeOut= false;
02348     TCSFontdefinition.lfCharSet= ANSI_CHARSET;
02349     TCSFontdefinition.lfOutPrecision= OUT_TT_ONLY_PRECIS;
02350     TCSFontdefinition.lfClipPrecision= CLIP_DEFAULT_PRECIS;
02351     TCSFontdefinition.lfQuality= DRAFT_QUALITY;
02352     TCSFontdefinition.lfPitchAndFamily= FF_MODERN | FIXED_PITCH;
02353     _tcscpy (TCSFontdefinition.lfFaceName, szTCSGraphicFont);
02354                         /* Bevorzugter Font, keine Proportionalschrift!!! */
02355
02356     hTCSFont= CreateFontIndirect (&TCSFontdefinition);
02357     #if !defined(__WIN32__) && !defined(_WIN32)
02358      SelectFont (hTCSWindowDC, hTCSFont);      // Aktuellen Zeichenstatus an
02359     #else
02360      SelectObject (hTCSWindowDC, hTCSFont);      // Aktuellen Zeichenstatus an
02361     #endif
02362     SetTextColor (hTCSWindowDC, dwColorTable[TKTRNX.iTxtCol]);
02363
02364     GetTextMetrics (hTCSWindowDC, &lpTM);
02365     TKTRNX.kitalc= 0;
02366     TKTRNX.ksizef= 0;
02367     TKTRNX.khorsz= (FTNINT) ((float)LoRes((float)lpTM.tmAveCharWidth *TEK_XMAX/iSizeX) + 0.25f);
02368     TKTRNX.kversz= (FTNINT) ((float)LoRes((float)lpTM.tmHeight *TEK_YMAX/iSizeY)  + 0.25f);
02369
02370     SetBkMode (hTCSWindowDC, TRANSPARENT );   /* Attribut statisch, durch */
02371     SetTextAlign (hTCSWindowDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); /* Ort: */
02372
02373     hTCSPen= CreatePen (PS_SOLID, 0, dwColorTable[TKTRNX.iLinCol]);
02374     #if !defined(__WIN32__) && !defined(_WIN32)
02375      SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
02376     #else
02377      SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
02378     #endif
02379
02380     hGinCurs=LoadCursor(NULL, IDC_CROSS);
02381     hMouseCurs=LoadCursor(NULL, IDC_ARROW);
02382
02383 #if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
02384     #if !defined(__WIN32__) && !defined(_WIN32)
02385      SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
02386     #else
02387      SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
02388     #endif
02389     SetBkMode (hTCSMetaFileDC, TRANSPARENT );
02390     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
02391     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
02392     #if !defined(__WIN32__) && !defined(_WIN32)
02393      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
02394     #else
02395      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
02396     #endif
02397
02398 #elif (JOURNALTYP == 3)
02399     hTCSJournal= NULL;
02400     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02401     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");
02402
02403     xJournalEntry->action=  XACTION_NOOP; // Erkennung Listenanfang: Wurzelelement ohne Funktion
02404     xJournalEntry->i1= 0;
02405     xJournalEntry->i2= 0;
02406     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02407
02408     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02409     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02410     xJournalEntry->action=  XACTION_INITT;
02411     xJournalEntry->i1= 0;
02412     xJournalEntry->i2= 0;
02413     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02414 #endif
02415
```

```
02416     /*
02417         Erzeugung des Statusfensters
02418     */
02419
02420     TCSWndClass.style          = CS_HREDRAW | CS_VREDRAW;  //  CS_OWNDC |
02421     TCSWndClass.lpfnWndProc    = TCSstatWndProc;
02422     TCSWndClass.hInstance      = hTCSInst;
02423     TCSWndClass.hIcon          = NULL;
02424     TCSWndClass.hCursor        = LoadCursor(NULL, IDC_ARROW);
02425     #if !defined(__WIN32__) && !defined(_WIN32)
02426      TCSWndClass.hbrBackground  = (HBRUSH) GetStockBrush(WHITE_BRUSH);
02427     #else
02428      TCSWndClass.hbrBackground  = GetStockObject(WHITE_BRUSH);
02429     #endif
02430     TCSWndClass.lpszMenuName   = NULL;
02431     TCSWndClass.lpszClassName  = TCS_STAT_WINDOWCLASS;
02432
02433     if (!RegisterClass (&TCSWndClass)) {
02434      #if defined(__WIN32__) || defined(_WIN32)
02435       retValue= GetLastError(); // win32-Funktion
02436 //    if (retValue == ERROR_CLASS_ALREADY_EXISTS) {
02437 //     Hier bei Bedarf Fehlerbehandlung einführen
02438 //    } else {
02439       FormatMessage(
02440         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
02441         NULL,
02442         retValue,
02443         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
02444         (LPTSTR) &lpMsgBuf,
02445         0,
02446         NULL
02447       );
02448       MessageBox (NULL, lpMsgBuf, szTCSWindowName, MB_ICONSTOP);
02449       LocalFree( lpMsgBuf ); // Free the buffer
02450 //    } // Ende der Fehlerbehandlung
02451      #else // rudimentaere Fehlerbehandlung 16bit Windows
02452       MessageBox (NULL, _T("Window Class not registered"),
02453                                    szTCSWindowName, MB_ICONSTOP);
02454      #endif
02455      return;
02456     }
02457
02458     if ((TCSstatWindowIniXrelsiz < 100) || (TCSstatWindowIniYrelsiz < 100) ) {
02459      FirstShow= WS_OVERLAPPED | WS_SIZEBOX | WS_VSCROLL; // WIN16: int*2 !
02460      iX= (int) ( ( (long int) TCSstatWindowIniXrelpos *
02461                    (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
02462      iY= (int) ( ( (long int) TCSstatWindowIniYrelpos *
02463                    (long int) GetSystemMetrics (SM_CYMAXIMIZED)) / 100);
02464      iSizeX= (int) ( ( (long int) TCSstatWindowIniXrelsiz *
02465                        (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
02466      iSizeY= (int) ( ( (long int)  TCSstatWindowIniYrelsiz *
02467                        (long int) GetSystemMetrics (SM_CYMAXIMIZED) ) / 100);
02468     } else {
02469      FirstShow= WS_OVERLAPPED | WS_SIZEBOX | WS_VSCROLL | WS_MAXIMIZE;
02470      iX= 0;
02471      iY = GetSystemMetrics (SM_CYMAXIMIZED) -
02472                     #if defined(__WIN32__) || defined(_WIN32)
02473                                  (int) (TCS_REL_CHR_SPACE*TextLineHeight) -
02474                     #endif
02475                                STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
02476      iSizeX= GetSystemMetrics (SM_CXMAXIMIZED);
02477      iSizeY= (int) (TCS_REL_CHR_SPACE*TextLineHeight) +
02478                                STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
02479     }
02480
02481     hTCSstatWindow = CreateWindow(TCS_STAT_WINDOWCLASS, szTCSstatWindowName,
02482                         FirstShow,
02483                         iX, iY,
02484                         iSizeX, iSizeY,
02485                         (HWND) hTCSWindow, (HMENU) NULL,
02486                         (HINSTANCE) hTCSInst, (LPSTR) NULL);
02487
02488     if (hTCSstatWindow == NULL) return;
02489
02490     #ifdef STAT_WINDOW_PRIVATE
02491      hTCSstatWindowDC = GetDC (hTCSstatWindow);
02492     #endif
02493
02494     TCSFontdefinition.lfHeight= TextLineHeight; /* Buchstabenhöhe */
02495     _tcscpy (TCSFontdefinition.lfFaceName, szTCSSysFont);
02496                         /* Bevorzugter Font, keine Proportionalschrift!!! */
02497     hTCSSysFont= CreateFontIndirect (&TCSFontdefinition);
02498
02499     TCSFontdefinition.lfHeight= TCSCharHeight; /* Wiederherstellung Graphikzeichensatz */
02500     _tcscpy (TCSFontdefinition.lfFaceName, szTCSGraphicFont);
02501
02502
```

```
02503     TCSStatWindowAutomatic = true;
02504     TCSstatCursorPosY= 0;
02505     TCSstatScrollY= 0;
02506     TCSstatRow= -1;
02507     TCSstatOrgY= TCSstatScrollY;
02508     SetScrollRange (hTCSstatWindow, SB_VERT, 0,STAT_MAXROWS-1, true);
02509     SetScrollPos (hTCSstatWindow, SB_VERT, TCSstatScrollY, true);
02510 
02511     ShowWindow (hTCSstatWindow, SW_HIDE);
02512 
02513     ClippingNotActive= true;
02514 
02515     return;
02516 }
02517 
02518 
02519 
02520 extern void finitt ()
02521 {
02522 // FTNINT iErr;
02523 #if (JOURNALTYP == 1)
02524  HMETAFILE hmf;
02525 #elif (JOURNALTYP == 2)
02526  HENHMETAFILE hmf;
02527 #elif (JOURNALTYP == 3)
02528  struct xJournalEntry_typ * xJournalEntry;
02529 #endif
02530 
02531 
02532     if (!TCSinitialized) return; /* Graphiksystem nicht initialisiert */
02533 
02534     TCSGraphicError (ERR_EXIT,"");  /* TCSinitialized verhindert Rekursion*/
02535 
02536     TCSinitialized= false;           /* Ab jetzt nicht mehr funktionsfähig */
02537 
02538     ReleaseDC (hTCSWindow, hTCSWindowDC);
02539     DestroyWindow (hTCSWindow);
02540     UnregisterClass (TCS_WINDOWCLASS, hTCSInst);
02541 
02542 #if (JOURNALTYP == 1)
02543     hmf = CloseMetaFile (hTCSMetaFileDC);
02544     DeleteMetaFile (hmf);
02545 #elif (JOURNALTYP == 2)
02546     hmf = CloseEnhMetaFile (hTCSMetaFileDC);
02547     DeleteEnhMetaFile (hmf);
02548 #elif (JOURNALTYP == 3)
02549     SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, hTCSJournal,
02550            xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
02551     hTCSJournal= NULL;
02552 #endif
02553 
02554     #ifdef STAT_WINDOW_PRIVATE
02555      ReleaseDC (hTCSstatWindow, hTCSstatWindowDC);
02556     #endif
02557     DestroyWindow (hTCSstatWindow);
02558     UnregisterClass (TCS_STAT_WINDOWCLASS, hTCSInst);
02559 
02560     #if !defined(__WIN32__) && !defined(_WIN32)
02561      DeleteFont (hTCSFont);
02562      DeleteFont (hTCSSysFont);
02563      DeletePen (hTCSPen);
02564     #else
02565      DeleteObject (hTCSFont);
02566      DeleteObject (hTCSSysFont);
02567      DeleteObject (hTCSPen);
02568     #endif
02569 
02570     #if defined(__WATCOMC__) && defined(__SW_BW)
02571      _dwShutDown();         // Shutdown Watcom Default Window System
02572     #endif
02573 
02574     if (TCSErrorLev[ERR_EXIT] >= 10) exit (EXIT_SUCCESS); // Programmende
02575     return; // Bei Fehlerlevel <10 zurück zum Hauptprogramm
02576 }
02577 
02578 
02579 
02580 /*
02581 ---------------------- Userroutinen: Zeichnen -------------------
02582 */
02583 
02584 
02585 
02586 extern void swind1 (FTNINT *ix1,FTNINT *iy1,FTNINT *ix2,FTNINT *iy2)
02587 {
02588     ClippingNotActive = (*ix1==0) && (*iy1==0) &&
02589                                      (*ix2==TEK_XMAX) && (*iy2==TEK_YMAX);
```

```
02590     /* Berechnung BOOL zur Wahrung der Programmstruktur der DOS-Version */
02591 }
02592
02593
02594
02595 extern void erase (void)
02596 {
02597 #if (JOURNALTYP == 1)
02598  HMETAFILE hmf;
02599  HRGN       hWindowRegion;
02600  HBRUSH     hBack;
02601 #elif (JOURNALTYP == 2)
02602  HENHMETAFILE   hmf;
02603  ENHMETAHEADER emh ;
02604 #elif (JOURNALTYP == 3)
02605  struct xJournalEntry_typ    * xJournalEntry;
02606 #endif
02607
02608 #if (JOURNALTYP == 1)
02609      hmf = CloseMetaFile (hTCSMetaFileDC);   /* Cursor, Farben unverändert! */
02610      DeleteMetaFile (hmf);                   /* alter Status Bildschirm */
02611      hTCSMetaFileDC  = CreateMetaFile (NULL);/* für neues Journalfile */
02612      SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
02613      SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
02614
02615      hBack= CreateSolidBrush (dwColorTable[TKTRNX.iBckCol]);
02616      hWindowRegion= CreateRectRgn (TCSrect.left, TCSrect.top, TCSrect.right,TCSrect.bottom); //
     rechts,oben
02617      FillRgn (hTCSMetaFileDC, hWindowRegion, hBack);         // nicht eingeschlossen
02618      #if !defined(__WIN32__) && !defined(_WIN32)
02619       DeleteBrush (hBack);
02620       DeleteRgn (hWindowRegion);                  /* Resourcen freigeben */
02621       SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
02622      #else
02623       DeleteObject (hBack);
02624       DeleteObject (hWindowRegion);
02625       SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
02626      #endif
02627
02628      SetBkMode (hTCSMetaFileDC, TRANSPARENT );
02629      SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
02630      SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
02631      #if !defined(__WIN32__) && !defined(_WIN32)
02632       SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
02633      #else
02634       SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
02635      #endif
02636
02637      MoveToEx (hTCSMetaFileDC, HiRes(TKTRNX.kBeamX), HiRes(TKTRNX.kBeamY), NULL);
02638
02639 #elif (JOURNALTYP == 2)
02640      hmf = CloseEnhMetaFile (hTCSMetaFileDC);
02641      GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
02642      DeleteEnhMetaFile (hmf);                      // alter Status Bildschirm
02643
02644      hTCSMetaFileDC  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
02645                        _T("TCS for Windows\0Journalfile created by Erase\0\0"));
02646
02647      SetMapMode (hTCSMetaFileDC, MM_ANISOTROPIC);
02648      SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
02649      SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
02650      SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
02651      SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
02652
02653      #if !defined(__WIN32__) && !defined(_WIN32)
02654       SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
02655      #else
02656       SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
02657      #endif
02658      SetBkMode (hTCSMetaFileDC, TRANSPARENT );
02659      SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
02660      SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
02661      #if !defined(__WIN32__) && !defined(_WIN32)
02662       SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
02663      #else
02664       SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
02665      #endif
02666
02667      MoveToEx (hTCSMetaFileDC, HiRes(TKTRNX.kBeamX),HiRes(TKTRNX.kBeamY), NULL);
02668
02669 #elif (JOURNALTYP == 3)
02670      SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, hTCSJournal,
02671           xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
02672      hTCSJournal= NULL;
02673
02674      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02675      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
```

```
02676      xJournalEntry->action=  XACTION_NOOP;
02677      xJournalEntry->i1= 0;
02678      xJournalEntry->i2= 0;
02679      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02680
02681      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02682      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
02683      xJournalEntry->action=  XACTION_LINCOL;
02684      xJournalEntry->i1= TKTRNX.iLinCol;
02685      xJournalEntry->i2= 0;
02686      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02687
02688      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02689      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
02690      xJournalEntry->action=  XACTION_TXTCOL;
02691      xJournalEntry->i1= TKTRNX.iTxtCol;
02692      xJournalEntry->i2= 0;
02693      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02694
02695      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02696      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
02697      xJournalEntry->action=  XACTION_BCKCOL;
02698      xJournalEntry->i1= TKTRNX.iBckCol;
02699      xJournalEntry->i2= 0;
02700      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02701
02702      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02703      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02704      xJournalEntry->action=  XACTION_ERASE;
02705      xJournalEntry->i1= 0;
02706      xJournalEntry->i2= 0;
02707      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02708 #endif
02709
02710     TCSBackgroundColour=TKTRNX.iBckCol; /* Jetzt in ERASE-Handler wirksam */
02711
02712     InvalidateRect (hTCSWindow, NULL, true); /* ,ClientArea, EraseFlag */
02713     UpdateWindow (hTCSWindow); /* 16bit Rechner: gegen Irritation Anwender */
02714
02715 }
02716
02717
02718
02719 extern void movabs (FTNINT *ix,FTNINT *iy)
02720 {
02721 int ixx, iyy; /* Erzwingt Typangleichung Windows-GDI / Fortran */
02722
02723 #if (JOURNALTYP == 3)
02724  struct xJournalEntry_typ    * xJournalEntry;
02725 #endif
02726
02727     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
02728     if (PointInWindow (*ix, *iy)) {
02729      ixx= HiRes(*ix); iyy= HiRes(*iy);
02730      MoveToEx (hTCSWindowDC, ixx, iyy, NULL);
02731
02732 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02733      MoveToEx (hTCSMetaFileDC, ixx, iyy, NULL);
02734 #elif (JOURNALTYP == 3)
02735      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02736      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02737      xJournalEntry->action=  XACTION_MOVABS;
02738      xJournalEntry->i1= *ix;
02739      xJournalEntry->i2= *iy;
02740      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02741 #endif
02742     }
02743 }
02744
02745
02746
02747 extern void drwabs (FTNINT *ix,FTNINT *iy)
02748 {
02749 FTNINT iXClip, iYClip;
02750 int ixx, iyy;
02751
02752 #if (JOURNALTYP == 3)
02753  struct xJournalEntry_typ    * xJournalEntry;
02754 #endif
02755
02756     if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
02757      ixx= HiRes(iXClip); iyy= HiRes(iYClip);
02758      MoveToEx (hTCSWindowDC, ixx,iyy, NULL);
02759 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02760      MoveToEx (hTCSMetaFileDC, ixx,iyy, NULL);
02761 #elif (JOURNALTYP == 3)
02762      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
```

```
02763       if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02764       xJournalEntry->action=  XACTION_MOVABS;
02765       xJournalEntry->i1= iXClip;
02766       xJournalEntry->i2= iYClip;
02767       SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02768 #endif
02769
02770       ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip,&iYClip);
02771       ixx= HiRes(iXClip); iyy= HiRes(iYClip);    /* geclippter Endpunkt */
02772       LineTo (hTCSWindowDC, ixx,iyy);      /* Endpunkt nicht mitgezeichnet! */
02773       SetPixel (hTCSWindowDC,ixx,iyy,dwColorTable[TKTRNX.iLinCol]);
02774
02775 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02776       LineTo (hTCSMetaFileDC, ixx,iyy);
02777       SetPixel (hTCSMetaFileDC,ixx,iyy, dwColorTable[TKTRNX.iLinCol]);
02778 #elif (JOURNALTYP == 3)
02779       xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02780       if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02781       xJournalEntry->action=  XACTION_DRWABS;
02782       xJournalEntry->i1= iXClip;
02783       xJournalEntry->i2= iYClip;
02784       SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02785
02786       xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02787       if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
02788       xJournalEntry->action=  XACTION_MOVABS;
02789       xJournalEntry->i1= *ix;
02790       xJournalEntry->i2= *iy;
02791       SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02792 #endif
02793
02794     }
02795     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
02796
02797 }
02798
02799
02800
02801 extern void dshabs (FTNINT *ix,FTNINT *iy, FTNINT *iMask)
02802 {
02803 HPEN     hPenDash;
02804 FTNINT iXClip, iYClip;
02805 int      iMaskIndex, ixx, iyy;
02806
02807 #if (JOURNALTYP == 3)
02808  struct xJournalEntry_typ    * xJournalEntry;
02809 #endif
02810
02811     if (*iMask < 0) {     /* Verhindern eines Access-Errors bei Integermaskenübergabe */
02812       iMaskIndex= 0;
02813     } else if (*iMask > MAX_PENSTYLE_INDEX) {
02814       iMaskIndex= 1;       /* Style: dotted */
02815     } else {
02816       iMaskIndex= *iMask;
02817     }
02818
02819     if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
02820       ixx= HiRes(iXClip); iyy= HiRes(iYClip);
02821       MoveToEx (hTCSWindowDC, ixx,iyy, NULL);
02822
02823 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02824       MoveToEx (hTCSMetaFileDC, ixx,iyy, NULL);
02825 #elif (JOURNALTYP == 3)
02826       xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02827       if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02828       xJournalEntry->action=  XACTION_MOVABS;
02829       xJournalEntry->i1= iXClip;
02830       xJournalEntry->i2= iYClip;
02831       SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02832 #endif
02833
02834       ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip,&iYClip);
02835       ixx= HiRes(iXClip); iyy= HiRes(iYClip);       /* geclippter Endpunkt */
02836
02837       hPenDash= CreatePen (dwPenStyle[iMaskIndex], 0, dwColorTable[TKTRNX.iLinCol]);
02838       #if !defined(__WIN32__) && !defined(_WIN32)
02839        SelectPen (hTCSWindowDC, hPenDash); // 16bit: Makro aus windowsx.h
02840       #else
02841        SelectObject (hTCSWindowDC, hPenDash); // 32bit: GDI Standardaufruf
02842       #endif
02843       LineTo (hTCSWindowDC, ixx,iyy);    /* Ohne Endpunkt bei Dash o.k! */
02844       #if !defined(__WIN32__) && !defined(_WIN32)
02845        SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
02846       #else
02847        SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
02848       #endif
02849
```

```
02850 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02851      #if !defined(__WIN32__) && !defined(_WIN32)
02852       SelectPen (hTCSMetaFileDC, hPenDash); // 16bit: Makro aus windowsx.h
02853      #else
02854       SelectObject (hTCSMetaFileDC, hPenDash); // 32bit: GDI Standardaufruf
02855      #endif
02856      LineTo (hTCSMetaFileDC, ixx,iyy);
02857      #if !defined(__WIN32__) && !defined(_WIN32)
02858       SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
02859      #else
02860       SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
02861      #endif
02862 #elif (JOURNALTYP == 3)
02863      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02864      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02865      xJournalEntry->action=  XACTION_DSHSTYLE;
02866      xJournalEntry->i1= iMaskIndex;
02867      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02868
02869      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02870      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02871      xJournalEntry->action=  XACTION_DSHABS;
02872      xJournalEntry->i1= iXClip;
02873      xJournalEntry->i2= iYClip;
02874      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02875
02876      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02877      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
02878      xJournalEntry->action=  XACTION_MOVABS;
02879      xJournalEntry->i1= *ix;
02880      xJournalEntry->i2= *iy;
02881      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02882 #endif
02883
02884      #if !defined(__WIN32__) && !defined(_WIN32)
02885       DeletePen (hPenDash);
02886      #else
02887       DeleteObject (hPenDash);
02888      #endif
02889
02890     }
02891     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
02892 }
02893
02894
02895
02896 extern void pntabs (FTNINT *ix,FTNINT *iy)
02897 {
02898 int     ixx, iyy; /* Erzwingt Typangleichung Windows-GDI / Fortran */
02899
02900 #if (JOURNALTYP == 3)
02901  struct xJournalEntry_typ    * xJournalEntry;
02902 #endif
02903
02904     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
02905     if (PointInWindow (*ix, *iy)) {
02906      ixx= HiRes(*ix); iyy= HiRes(*iy);
02907      SetPixel (hTCSWindowDC,ixx,iyy, dwColorTable[TKTRNX.iLinCol]);
02908
02909 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02910      SetPixel (hTCSMetaFileDC,ixx,iyy,dwColorTable[TKTRNX.iLinCol]);
02911 #elif (JOURNALTYP == 3)
02912      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02913      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02914      xJournalEntry->action=  XACTION_PNTABS;
02915      xJournalEntry->i1= *ix;
02916      xJournalEntry->i2= *iy;
02917      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02918 #endif
02919
02920     }
02921 }
02922
02923
02924
02925 extern void bckcol (FTNINT *iCol)
02926 {
02927
02928 #if (JOURNALTYP == 3)
02929  struct xJournalEntry_typ    * xJournalEntry;
02930 #endif
02931
02932     TKTRNX.iBckCol= min(abs(*iCol),MAX_COLOR_INDEX);
02933
02934 #if (JOURNALTYP == 3)
02935     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02936     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
```

```
02937     xJournalEntry->action=  XACTION_BCKCOL;
02938     xJournalEntry->i1= TKTRNX.iBckCol;
02939     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02940 #endif
02941
02942 }
02943
02944
02945
02946 extern void lincol (FTNINT *iCol)
02947 {
02948
02949 HPEN    hPenOld;
02950
02951 #if (JOURNALTYP == 3)
02952  struct xJournalEntry_typ    * xJournalEntry;
02953 #endif
02954
02955     TKTRNX.iLinCol= min(abs(*iCol),MAX_COLOR_INDEX);
02956     hTCSPen= CreatePen (PS_SOLID, 0, dwColorTable[TKTRNX.iLinCol]);
02957     #if !defined(__WIN32__) && !defined(_WIN32)
02958      hPenOld= SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
02959     #else
02960      hPenOld= SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
02961     #endif
02962
02963 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02964     #if !defined(__WIN32__) && !defined(_WIN32)
02965      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
02966     #else
02967      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
02968     #endif
02969 #elif (JOURNALTYP == 3)
02970     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02971     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
02972     xJournalEntry->action=  XACTION_LINCOL;
02973     xJournalEntry->i1= TKTRNX.iLinCol;
02974     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
02975 #endif
02976
02977     #if !defined(__WIN32__) && !defined(_WIN32)
02978      DeletePen (hPenOld);
02979     #else
02980      DeleteObject (hPenOld);
02981     #endif
02982
02983 }
02984
02985
02986
02987
02988 extern void txtcol (FTNINT *iCol)
02989 {
02990
02991 #if (JOURNALTYP == 3)
02992  struct xJournalEntry_typ    * xJournalEntry;
02993 #endif
02994
02995     TKTRNX.iTxtCol= min(abs(*iCol),MAX_COLOR_INDEX);
02996     SetTextColor (hTCSWindowDC, dwColorTable[TKTRNX.iTxtCol]);
02997 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
02998     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
02999 #elif (JOURNALTYP == 3)
03000     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03001     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
03002     xJournalEntry->action=  XACTION_TXTCOL;
03003     xJournalEntry->i1= TKTRNX.iTxtCol;
03004     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03005 #endif
03006
03007 }
03008
03009
03010
03011 extern void DefaultColour (void)
03012 {
03013     TKTRNX.iLinCol= TCSDefaultLinCol;
03014     TKTRNX.iTxtCol= TCSDefaultTxtCol;
03015     TKTRNX.iBckCol= TCSDefaultBckCol;
03016
03017     lincol (&TKTRNX.iLinCol);
03018     txtcol (&TKTRNX.iTxtCol);
03019     bckcol (&TKTRNX.iBckCol);
03020 }
03021
03022
03023
```

```
03024 /*
03025 ---------------------- Userroutinen: Graphiktext -----------------------
03026 */
03027
03028
03029
03030 extern void outgtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
03031 {
03032 int iL;
03033 SIZE Size;
03034 POINT CPpos;
03035
03036 #if (JOURNALTYP == 3)
03037  int i;
03038  struct xJournalEntry_typ    * xJournalEntry;
03039 #endif
03040
03041 #ifdef extended_error_handling
03042  HDC         hdc;
03043  LPVOID      lpMsgBuf;
03044 #endif
03045
03046
03047     if (FTNSTRPARA(ftn_string)[0] == (FTNCHAR) 0 ) return; // Leerstring char(0)
03048
03049     iL= 1; // Stringbeginn bei 0 -> Dec Laenge
03050     while ( (FTNSTRPARA(ftn_string)[iL-1] != (FTNCHAR) 0) &&  // c-String bis \0
03051                     (iL < FTNSTRPARL(ftn_string)) ) iL++;  // oder Ftn-String
03052     if (FTNSTRPARA(ftn_string)[iL-1] == (FTNCHAR) 0 ) iL--;   // cString ohne \0
03053
03054
03055     #ifdef extended_error_handling
03056      if (GetTextExtentPoint (hTCSWindowDC, FTNSTRPARA(ftn_string),iL,&Size) == 0 ){
03057       hdc = CreateIC (_T ("DISPLAY"), NULL, NULL, NULL);
03058       #if !defined(__WIN32__) && !defined(_WIN32)
03059        SelectFont (hdc, hTCSFont);      // Aktuellen Zeichenstatus an
03060       #else
03061        SelectObject (hdc, hTCSFont);      // Aktuellen Zeichenstatus an
03062       #endif
03063       GetTextExtentPoint (hdc, FTNSTRPARA(ftn_string),iL,&Size);
03064       DeleteDC (hdc);
03065
03066       FormatMessage(
03067         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
03068         NULL,
03069         GetLastError(),
03070         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
03071         (LPTSTR) &lpMsgBuf,
03072         0,
03073         NULL
03074       );
03075       MessageBox( NULL, lpMsgBuf,
03076                         _T("Internal Error GRAPH2D - subroutine _OUTGTEXT"),
03077                                                 MB_OK|MB_ICONINFORMATION );
03078       LocalFree( lpMsgBuf ); // Free the buffer
03079      }
03080     #else
03081      #if !defined(__WIN32__) && !defined(_WIN32)
03082       GetTextExtentPoint (hTCSWindowDC, FTNSTRPARA(ftn_string),iL,&Size);
03083      #else
03084       GetTextExtentPoint32 (hTCSWindowDC, FTNSTRPARA(ftn_string),iL,&Size);
03085      #endif
03086     #endif
03087
03088     if (PointInWindow (TKTRNX.kBeamX+LoRes(Size.cx),
03089                                             TKTRNX.kBeamY+LoRes(Size.cy))) {
03090      MoveToEx (hTCSWindowDC,HiRes(TKTRNX.kBeamX),HiRes(TKTRNX.kBeamY),NULL);
03091      TextOut (hTCSWindowDC, 0,0,FTNSTRPARA(ftn_string), iL);
03092
03093 #if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
03094      MoveToEx (hTCSMetaFileDC,HiRes(TKTRNX.kBeamX),HiRes(TKTRNX.kBeamY),NULL);
03095      TextOut (hTCSMetaFileDC, 0,0, FTNSTRPARA(ftn_string), iL);
03096 #elif (JOURNALTYP == 3)
03097      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03098      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
03099      xJournalEntry->action=  XACTION_MOVABS;
03100      xJournalEntry->i1= TKTRNX.kBeamX;
03101      xJournalEntry->i2= TKTRNX.kBeamY;
03102      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03103
03104      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03105      xJournalEntry->action=  XACTION_GTEXT;
03106      xJournalEntry->i1= (FTNINT) iL;
03107      xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[0];
03108      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03109
03110      i= 1;
```

```
03111       while (i < iL) {
03112        xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03113        xJournalEntry->action=  XACTION_ASCII;
03114        xJournalEntry->i1= (FTNINT) FTNSTRPARA(ftn_string)[i++];
03115        if ( i<iL ) xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[i++];
03116        SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03117       }
03118 #endif
03119
03120       GetCurrentPositionEx (hTCSWindowDC, &CPpos); /* Update Beam */
03121       TKTRNX.kBeamX= LoRes(CPpos.x); TKTRNX.kBeamY= LoRes(CPpos.y);
03122
03123 #if (JOURNALTYP == 3) // Bei Metafiles ist auch nach Neuskalierung CP i.O.
03124       xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03125       xJournalEntry->action=  XACTION_MOVABS;
03126       xJournalEntry->i1= TKTRNX.kBeamX;
03127       xJournalEntry->i2= TKTRNX.kBeamY;
03128       SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03129 #endif
03130
03131     }
03132 }
03133
03134
03135
03136 extern void italic (void)
03137 {
03138 HFONT   hOldFont;
03139 #if (JOURNALTYP == 3)
03140  struct xJournalEntry_typ    * xJournalEntry;
03141 #endif
03142
03143     TKTRNX.kitalc = 1;
03144
03145     TCSFontdefinition.lfItalic= true;
03146     hTCSFont= CreateFontIndirect (&TCSFontdefinition);
03147     #if !defined(__WIN32__) && !defined(_WIN32)
03148      hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
03149     #else
03150      hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
03151     #endif
03152 #if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
03153     #if !defined(__WIN32__) && !defined(_WIN32)
03154      SelectFont (hTCSMetaFileDC, hTCSFont);
03155     #else
03156      SelectObject (hTCSMetaFileDC, hTCSFont);
03157     #endif
03158 #elif (JOURNALTYP == 3)
03159     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03160     xJournalEntry->action= XACTION_FONTATTR;
03161     xJournalEntry->i1= TKTRNX.kitalc;
03162     xJournalEntry->i2= TKTRNX.ksizef;
03163     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03164 #endif
03165     #if !defined(__WIN32__) && !defined(_WIN32)
03166      DeleteFont (hOldFont);
03167     #else
03168      DeleteObject (hOldFont);
03169     #endif
03170 }
03171
03172
03173
03174 extern void italir (void)
03175 {
03176 HFONT   hOldFont;
03177 #if (JOURNALTYP == 3)
03178  struct xJournalEntry_typ    * xJournalEntry;
03179 #endif
03180
03181     TKTRNX.kitalc = 0;
03182
03183     TCSFontdefinition.lfItalic= false;
03184     hTCSFont= CreateFontIndirect (&TCSFontdefinition);
03185     #if !defined(__WIN32__) && !defined(_WIN32)
03186      hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
03187     #else
03188      hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
03189     #endif
03190 #if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
03191     #if !defined(__WIN32__) && !defined(_WIN32)
03192      SelectFont (hTCSMetaFileDC, hTCSFont);
03193     #else
03194      SelectObject (hTCSMetaFileDC, hTCSFont);
03195     #endif
03196 #elif (JOURNALTYP == 3)
03197     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
```

```
03198     xJournalEntry->action= XACTION_FONTATTR;
03199     xJournalEntry->i1= TKTRNX.kitalc;
03200     xJournalEntry->i2= TKTRNX.ksizef;
03201     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03202 #endif
03203     #if !defined(__WIN32__) && !defined(_WIN32)
03204      DeleteFont (hOldFont);
03205     #else
03206      DeleteObject (hOldFont);
03207     #endif
03208 }
03209
03210
03211
03212 extern void dblsiz (void)
03213 {
03214 HFONT   hOldFont;
03215 #if (JOURNALTYP == 3)
03216  struct xJournalEntry_typ    * xJournalEntry;
03217 #endif
03218
03219     TKTRNX.ksizef = 1;
03220     TKTRNX.khomey = TEK_YMAX - 3.0f*TKTRNX.kversz;
03221
03222     TCSFontdefinition.lfHeight= 2* TCSCharHeight;
03223     TCSFontdefinition.lfWidth= 0;
03224     hTCSFont= CreateFontIndirect (&TCSFontdefinition);
03225     #if !defined(__WIN32__) && !defined(_WIN32)
03226      hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
03227     #else
03228      hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
03229     #endif
03230 #if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
03231     #if !defined(__WIN32__) && !defined(_WIN32)
03232      SelectFont (hTCSMetaFileDC, hTCSFont);
03233     #else
03234      SelectObject (hTCSMetaFileDC, hTCSFont);
03235     #endif
03236 #elif (JOURNALTYP == 3)
03237     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03238     xJournalEntry->action= XACTION_FONTATTR;
03239     xJournalEntry->i1= TKTRNX.kitalc;
03240     xJournalEntry->i2= TKTRNX.ksizef;
03241     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03242 #endif
03243     #if !defined(__WIN32__) && !defined(_WIN32)
03244      DeleteFont (hOldFont);
03245     #else
03246      DeleteObject (hOldFont);
03247     #endif
03248 }
03249
03250
03251
03252 extern void nrmsiz (void)
03253 {
03254 HFONT   hOldFont;
03255 #if (JOURNALTYP == 3)
03256  struct xJournalEntry_typ    * xJournalEntry;
03257 #endif
03258
03259     TKTRNX.ksizef = 0;
03260     TKTRNX.khomey = TEK_YMAX - 1.5f*TKTRNX.kversz;
03261
03262     TCSFontdefinition.lfHeight= TCSCharHeight;
03263     TCSFontdefinition.lfWidth= 0;
03264     hTCSFont= CreateFontIndirect (&TCSFontdefinition);
03265     #if !defined(__WIN32__) && !defined(_WIN32)
03266      hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
03267     #else
03268      hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
03269     #endif
03270 #if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
03271     #if !defined(__WIN32__) && !defined(_WIN32)
03272      SelectFont (hTCSMetaFileDC, hTCSFont);
03273     #else
03274      SelectObject (hTCSMetaFileDC, hTCSFont);
03275     #endif
03276 #elif (JOURNALTYP == 3)
03277     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
03278     xJournalEntry->action= XACTION_FONTATTR;
03279     xJournalEntry->i1= TKTRNX.kitalc;
03280     xJournalEntry->i2= TKTRNX.ksizef;
03281     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
03282 #endif
03283     #if !defined(__WIN32__) && !defined(_WIN32)
03284      DeleteFont (hOldFont);
```

```
03285     #else
03286      DeleteObject (hOldFont);
03287     #endif
03288 }
03289
03290
03291
03292 extern void csize (FTNINT *ix,FTNINT *iy)
03293 {
03294 TEXTMETRIC  lpTM;
03295
03296 #ifdef extended_error_handling
03297  HDC         hdc;
03298  LPVOID      lpMsgBuf;
03299 #endif
03300
03301     #ifdef extended_error_handling
03302      if (GetTextMetrics (hTCSWindowDC, &lpTM)== 0) {
03303       /* WATCOM ohne Default-Windowsystem(auch bei Consolenanwendungen):
03304          evtl. kein Message-Loop vorhanden.
03305          Workaround: Abfrageschleife in MessageBox                        */
03306
03307       hdc = CreateIC (_T ("DISPLAY"), NULL, NULL, NULL);
03308       #if !defined(__WIN32__) && !defined(_WIN32)
03309        SelectFont (hdc, hTCSFont);
03310       #else
03311        SelectObject (hdc, hTCSFont);
03312       #endif
03313       GetTextMetrics (hdc, &lpTM);
03314       DeleteDC (hdc);
03315
03316       FormatMessage(
03317         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
03318         NULL,
03319         GetLastError(),
03320         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
03321         (LPTSTR) &lpMsgBuf,
03322         0,
03323         NULL
03324       );
03325       MessageBox( NULL, lpMsgBuf, "Internal Error GRAPH2D - subroutine CSIZE",
03326                                                   MB_OK|MB_ICONINFORMATION );
03327       LocalFree( lpMsgBuf ); // Free the buffer
03328      }
03329     #else
03330      GetTextMetrics (hTCSWindowDC, &lpTM);
03331     #endif
03332     *ix= (int) ((float)LoRes((float)lpTM.tmAveCharWidth) + 0.25f);
03333     *iy= (int) ((float)LoRes((float)lpTM.tmHeight)  + 0.25f);
03334
03335 }
03336
03337
03338
03339
03340 /*
03341 ---------------------- Userroutinen: Graphic Input----------------------
03342 */
03343
03344
03345
03346 extern void tinput (FTNINT *ic)
03347 {
03348 MSG msg;        /* Message information */
03349 TCHAR iChar;
03350 HWND hAktWindowInThread;
03351
03352     if (!TCSinitialized) return;            /* Aufhängen vermeiden */
03353     TCSStatWindowAutomatic = false;          /* Meldungen lesbar */
03354     iChar= (TCHAR) 0;
03355     hAktWindowInThread= GetFocus(); // Fuer Texteingabe eigene Applikation
03356     while (iChar == (TCHAR) 0) { // Messageschleife jetzt hier -> Usereingabe
03357      SetFocus (hTCSWindow);         // Kein Zugang Elternfenster (Aufhängen!)
03358      #ifdef extended_error_handling
03359       if (GetMessage (&msg, NULL, WM_NULL, WM_USER) == -1) {
03360        MessageBox(NULL, "GetMessage failed in Mesageloop of Graphic Window",
03361                         "Internal Information GRAPH2D - Subroutine TINPUT",
03362                         MB_OK | MB_ICONINFORMATION);
03363       }
03364      #else
03365       GetMessage (&msg, NULL, WM_NULL, WM_USER); // Achtung wg. win7 nicht 0,0)
03366      #endif
03367      if ((msg.hwnd != hTCSWindow) && (msg.hwnd != hTCSstatWindow) ) {
03368       switch (msg.message) {
03369        case WM_NCLBUTTONDOWN:   /* Fensterbefehle der Elternfenster zulassen */
03370        case WM_NCLBUTTONUP:
03371        case WM_NCLBUTTONDBLCLK:
```

```
03372         case WM_SYSKEYDOWN:
03373         case WM_SYSKEYUP:
03374         case WM_SYSCOMMAND:
03375          DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
03376          break;
03377         case WM_PAINT:
03378          UpdateWindow( msg.hwnd);
03379          break;
03380         default:
03381          SetFocus (hTCSWindow);
03382          UpdateWindow (hTCSWindow);
03383        }
03384       } else if (msg.hwnd == hTCSstatWindow) { /* Meldungen Statusfenster */
03385        switch (msg.message) {
03386         case WM_NCLBUTTONDOWN:    /* Scrollen und Verschieben zulassen */
03387         case WM_NCLBUTTONUP:
03388         case WM_NCLBUTTONDBLCLK:
03389         case WM_VSCROLL:
03390          DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
03391          break;
03392         case WM_PAINT:
03393           TCSstatWndProc_OnPaint (hTCSstatWindow);
03394           break;
03395         case WM_LBUTTONDOWN:
03396          iChar= (FTNINT) 27;    /* Verlassen PRESSANY durch Statusfenster */
03397          break;
03398        }
03399       } else { /* eigene Meldungen des Graphikfensters */
03400        switch (msg.message) {
03401         case WM_PAINT:
03402          TCSWndProc_OnPaint (msg.hwnd);
03403          break;
03404         case WM_RBUTTONDOWN:      /* Auf Wunsch Statusfenster sichtbar */
03405          ShowWindow (hTCSstatWindow, SW_SHOWNA);
03406          UpdateWindow(hTCSstatWindow);
03407          SetFocus (hTCSWindow);
03408          UpdateWindow (hTCSWindow);
03409          break;
03410         case WM_LBUTTONDOWN:
03411          ShowWindow (hTCSstatWindow, SW_HIDE);
03412          break;
03413         case WM_LBUTTONUP:
03414         case WM_MBUTTONUP:
03415         case WM_RBUTTONUP:
03416         case WM_MBUTTONDOWN:
03417         case WM_LBUTTONDBLCLK:
03418         case WM_RBUTTONDBLCLK:
03419         case WM_MBUTTONDBLCLK:
03420          SetFocus (hTCSWindow);
03421          UpdateWindow (hTCSWindow);
03422          break;
03423         case WM_KEYDOWN:          /* Hardwareanpassung, dann WM_CHAR */
03424         case WM_KEYUP:
03425          TranslateMessage (&msg);
03426          break;
03427         case WM_CHAR:             /* nach WM_KEYDOWN jetzt ASCII */
03428          iChar= (TCHAR) msg.wParam;
03429          break;
03430         case WM_KILLFOCUS:
03431          TCSStatWindowAutomatic= true; /* Statusfenster unsichtbar */
03432          ShowWindow (hTCSstatWindow, SW_HIDE); /* jetzt DefWindowProc */
03433          UpdateWindow (hTCSstatWindow);
03434         case WM_NCLBUTTONDOWN:
03435         case WM_NCLBUTTONUP:
03436         case WM_NCLBUTTONDBLCLK:
03437         case WM_SYSKEYDOWN:       /* Uebersetzt in WM_SYSCOMMAND */
03438         case WM_SYSKEYUP:
03439          DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
03440          break;
03441         case WM_QUIT:
03442          #ifdef trace_calls
03443           MessageBox(NULL, "WM_QUIT Graphic Window",
03444                      "Internal Information GRAPH2D - Subroutine TINPUT",
03445                      MB_OK | MB_ICONINFORMATION);
03446          #endif
03447         case WM_SYSCOMMAND:       /* und nach WM_SYSKEYDOWN Befehlsauswertung */
03448          switch (msg.wParam) {
03449           case SC_CLOSE:
03450            iChar= (FTNINT) 27;    /* <ALT><F4> -> ESC */
03451            break;
03452           case TCS_WM_COPY:
03453            #ifdef trace_calls
03454             MessageBox(NULL, "WM_SYSCOMMAND (TCS_WM_COPY)",
03455                        "Internal Information GRAPH2D - Subroutine TINPUT",
03456                         MB_OK | MB_ICONINFORMATION);
03457            #endif
03458            TCSWndProc_OnCopyClipboard ();
```

```
03459           break;
03460          default:
03461           DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam);
03462           break;
03463         } /* Systembefehle */
03464       } /* Window-Messageauswertung */
03465      } /* Meldungen des Graphikfensters */
03466     } /* Ende Eingabeschleife */
03467     *ic= (FTNINT) iChar;
03468     TCSStatWindowAutomatic= true;
03469     ShowWindow (hTCSstatWindow, SW_HIDE); /* Statusfenster unsichtbar */
03470     if (hAktWindowInThread != NULL) SetFocus (hAktWindowInThread);
03471     return;
03472 }
03473
03474
03475
03476
03477 extern void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy)
03478 {
03479 MSG msg;        /* Message information */
03480 TCHAR iButton, iKey;
03481
03482 #if defined(__WIN32__) || defined(_WIN32)
03483  POINT MousePos;
03484 #endif
03485
03486     if (!TCSinitialized) return;             /* Aufhängen vermeiden */
03487     TCSStatWindowAutomatic = false;           /* Meldungen lesbar */
03488
03489     InvalidateRect (hTCSWindow, NULL, true); /* ,ClientArea, EraseFlag */
03490     UpdateWindow (hTCSWindow); /* Notwendig bei OnPaint mit Journaltyp=3 */
03491
03492     iButton= (TCHAR) 0; iKey= (TCHAR) 0;
03493
03494     /* Setzen der Maus auf die alte GinCursor Position */
03495
03496     #if defined(__WIN32__) || defined(_WIN32)
03497      MousePos.x= HiRes(TCSGinCurPos.x); MousePos.y= HiRes(TCSGinCurPos.y);
03498      LPtoDP (hTCSWindowDC, (LPPOINT)&MousePos, 1);
03499      MapWindowPoints(hTCSWindow, HWND_DESKTOP, (LPPOINT)&MousePos, 1);
03500      MousePos.x=  MousePos.x* MOUSE_XMAX / GetSystemMetrics (SM_CXSCREEN);
03501      MousePos.y=  MousePos.y* MOUSE_YMAX / GetSystemMetrics (SM_CYSCREEN);
03502      mouse_event(MOUSEEVENTF_MOVE | MOUSEEVENTF_ABSOLUTE,
03503                                       MousePos.x,MousePos.y, 0, 0);
03504     #endif
03505
03506     SetCursor(hGinCurs);       /* WM_SETCURSOR wird ab hier nicht erzeugt! */
03507     while (iButton == (TCHAR) 0) {  /* Messageschleife jetzt hier  */
03508      SetFocus (hTCSWindow);    /* Kein Zugang Elternfenster (Aufhängen!) */
03509      GetMessage (&msg, NULL, WM_NULL, WM_USER); // Achtung wg. win7 nicht 0,0)
03510      if (msg.hwnd == hTCSstatWindow) { /* Statusfenster stört -> unsichtbar */
03511       switch (msg.message) {
03512        case WM_MOUSEMOVE:                  /* falls Cursor über Client-Area */
03513         TCSStatWindowAutomatic= true;
03514         ShowWindow (hTCSstatWindow, SW_HIDE);
03515        case WM_NCMOUSEMOVE:            /* Cursor ueber Titelleiste -> Pfeil */
03516         SetCursor (hMouseCurs);
03517         break;
03518       }
03519      }              /* Statuszeile und Scrollbar können noch angewählt werden */
03520      if (msg.hwnd != hTCSWindow) {
03521       switch (msg.message) {
03522        case WM_NCLBUTTONDOWN:    /* Fensterbefehle der Elternfenster zulassen */
03523        case WM_NCLBUTTONUP:
03524        case WM_NCLBUTTONDBLCLK:
03525        case WM_SYSKEYDOWN:
03526        case WM_SYSKEYUP:
03527        case WM_SYSCOMMAND:
03528         DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
03529         break;
03530        case WM_PAINT:
03531         if (msg.hwnd == hTCSstatWindow) {
03532          TCSstatWndProc_OnPaint (hTCSstatWindow);
03533         } else {
03534          UpdateWindow( msg.hwnd);
03535         }
03536         break;
03537        default:
03538         SetFocus (hTCSWindow);
03539         UpdateWindow (hTCSWindow);
03540       }
03541      } else { /* eigene Meldungen des Graphikfensters */
03542       switch (msg.message) {
03543        case WM_PAINT:
03544         TCSWndProc_OnPaint (msg.hwnd);
03545         break;
```

```
03546         case WM_NCMOUSEMOVE:    /* Cursor ueber Titelleiste -> Pfeil */
03547          SetCursor (hMouseCurs);
03548          break;
03549         case WM_MOUSEMOVE:     /* GinCursor evtl. von Titelleiste zurück */
03550          SetCursor (hGinCurs);
03551          iKey= (TCHAR) 0;      /* Tastenbetätigung außerhalb Graphikfenster */
03552          break;
03553         case WM_NCLBUTTONDOWN: /* Titelleiste kann Statusfenster steuern */
03554          TCSStatWindowAutomatic= true;
03555          ShowWindow (hTCSstatWindow, SW_HIDE); /* jetzt DefWindowProc ! */
03556         case WM_NCLBUTTONUP:
03557         case WM_NCLBUTTONDBLCLK:
03558         case WM_SYSKEYDOWN:       /* Uebersetzt in WM_SYSCOMMAND */
03559         case WM_SYSKEYUP:
03560          DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
03561          break;
03562         case WM_NCRBUTTONDOWN:
03563          ShowWindow (hTCSstatWindow, SW_SHOWNA);
03564          UpdateWindow(hTCSstatWindow);
03565          break;
03566         case WM_LBUTTONDOWN: {
03567          #if !defined(__WIN32__) && !defined(_WIN32)
03568 LftDwn:
03569          #endif
03570          if (iKey== (TCHAR) 0) iButton= 1; else iButton=iKey;
03571         }
03572         case WM_RBUTTONDOWN: if (iButton== (TCHAR) 0) iButton= 2;
03573         case WM_MBUTTONDOWN: if (iButton== (TCHAR) 0) iButton= 4; // wie DOS
03574          #if !defined(__WIN32__) && !defined(_WIN32)
03575           TCSGinCurPos= MAKEPOINT (msg.lParam);
03576          #else
03577           TCSGinCurPos.x= GET_X_LPARAM (msg.lParam);
03578           TCSGinCurPos.y= GET_Y_LPARAM (msg.lParam);
03579          #endif
03580          DPtoLP (hTCSWindowDC, (LPPOINT)&TCSGinCurPos, 1);
03581          TCSGinCurPos.x= LoRes(TCSGinCurPos.x);
03582          TCSGinCurPos.y= LoRes(TCSGinCurPos.y);
03583          break;
03584         case WM_LBUTTONUP: /* Falls erneuter Aufruf nach Taste unten wird */
03585         case WM_RBUTTONUP: /* der Cursor sonst wieder auf Pfeil umgestellt */
03586         case WM_MBUTTONUP:
03587          SetCursor (hGinCurs);
03588          break;
03589         case WM_KEYDOWN:          /* Hardwareanpassung, dann WM_CHAR */
03590         case WM_KEYUP:
03591          TranslateMessage (&msg);
03592          break;
03593         case WM_CHAR:             /* nach WM_KEYDOWN jetzt ASCII */
03594          iKey= (TCHAR) msg.wParam;
03595          #if !defined(__WIN32__) && !defined(_WIN32)
03596           goto LftDwn;           /* Workaround Fehlen mouse_event */
03597          #else
03598           mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
03599           break;
03600          #endif
03601         case WM_SYSCOMMAND:       /* und nach WM_SYSKEYDOWN Befehlsauswertung */
03602          switch (msg.wParam) {
03603           case SC_CLOSE:
03604            iKey= (FTNINT) 27;     /* <ALT><F4> -> ESC */
03605            #if !defined(__WIN32__) && !defined(_WIN32)
03606             goto LftDwn;
03607            #else
03608             mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
03609             break;
03610            #endif
03611           case TCS_WM_COPY:
03612            TCSWndProc_OnCopyClipboard ();
03613            break;
03614           default:
03615            DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam);
03616            break;                 /* Sonst keine Befehle auswerten */
03617          } /* Systembefehle */
03618        } /* Window-Messageauswertung */
03619       } /* Messages fuer Graphikfenster */
03620      } /* Ende Eingabeschleife */
03621      *ic= (FTNINT) iButton;
03622      *ix=TCSGinCurPos.x;
03623      *iy=TCSGinCurPos.y;
03624
03625      TCSStatWindowAutomatic= true;
03626      ShowWindow (hTCSstatWindow, SW_HIDE); /* Statusfenster unsichtbar */
03627      return;
03628 }
03629
03630
03631
03632 /*
```

```
03633 ---------------------- Userroutinen: Statusmeldungen -------------------
03634 */
03635
03636
03637
03638 extern void bell (void)
03639 {
03640     MessageBeep (-1);
03641 }
03642
03643
03644
03645
03646 extern void outtext (FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
03647 {
03648 int i;
03649
03650     TCSstatRow++;
03651     if (TCSstatRow >= STAT_MAXROWS) {
03652      TCSstatRow= STAT_MAXROWS-1;
03653      for (i=0; i<TCSstatRow;i++)
03654       _tcscpy( TCSstatTextBuf[i],TCSstatTextBuf[i+1]);
03655     }
03656
03657     _tcsncpy( TCSstatTextBuf[TCSstatRow],FTNSTRPARA(ftn_string),
03658                          min (FTNSTRPARL(ftn_string), STAT_MAXCOLUMNS));
03659     TCSstatTextBuf[TCSstatRow][STAT_MAXCOLUMNS]= (FTNCHAR) 0;
03660     // TCSstatTextBuf ist mit STAT_MAXCOLUMNS+1 fuer char(0) dimensioniert!
03661
03662     TCSstatScrollY= TCSstatRow   /* Anzahl  Zeilen  im Display */;
03663     ScrollWindow (hTCSstatWindow, 0,
03664                 (TCSstatOrgY-TCSstatScrollY)*TextLineHeight, NULL, NULL);
03665
03666     TCSstatOrgY= TCSstatScrollY;
03667
03668     SetScrollPos (hTCSstatWindow, SB_VERT, TCSstatScrollY, true);
03669
03670     ShowWindow (hTCSstatWindow, SW_SHOW);
03671     UpdateWindow(hTCSstatWindow);
03672 }
03673
03674
03675
03676 extern void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
03677                                         FTNINT *iL  FTNSTRPAR_TAIL(ftn_string))
03678 {
03679     TCSGraphicError (*iErr, FTNSTRPARA(ftn_string));
03680
03681 }
03682
03683
03684
03685 /*
03686 ---------------------- Userroutinen: Hardcopy -------------------
03687 */
03688
03689
03690 extern void hdcopy (void)
03691 {
03692 FTNINT      iErr;
03693 // FTNSTRDESC  ftnstrg;
03694 TCHAR       FilNam[TCS_FILE_NAMELEN], OldFilNam[TCS_FILE_NAMELEN];
03695 OFSTRUCT    ReOpenBuf;
03696
03697 #if (JOURNALTYP == 1)
03698  HMETAFILE  hmf, hmf1;
03699  HDC        hTCSNewMetaFileDC;
03700  HRGN       hWindowRegion;
03701  HBRUSH     hBack;
03702 #elif (JOURNALTYP == 2)
03703  HENHMETAFILE   hmf, hmf1;
03704  HDC            hTCSNewMetaFileDC;
03705  ENHMETAHEADER  emh ;
03706  DWORD          ErrorCode;
03707  LPVOID         lpMsgBuf;
03708 #elif (JOURNALTYP == 3)
03709  struct xJournalEntry_typ    *xJournalEntry;
03710  FILE           *fHandle;
03711 #endif
03712
03713     FilNam[0] = (FTNCHAR) 0;
03714     OldFilNam[0] = (FTNCHAR) 0;
03715     do {     /* Suche erstes nicht existierendes File */
03716      _tcscpy(OldFilNam, FilNam);
03717      sprintf( FilNam, szTCSHardcopyFile, iHardcopyCount++ );
03718     } while ( (OpenFile (FilNam, &ReOpenBuf, OF_EXIST) != HFILE_ERROR) &&
03719               (_tcsicmp (FilNam,OldFilNam) > 0 )                          );
```

```
03720
03721     if (_tcsicmp (FilNam,OldFilNam) <= 0 ) { /* kein Filename vorhanden */
03722      iErr= WRN_HDCFILOPN;
03723      TCSGraphicError (iErr,"");
03724      return;                                 /* Error during Open -> ret */
03725     }
03726
03727     iErr= MSG_HDCACT;
03728     TCSGraphicError (iErr,FilNam);
03729
03730 #if (JOURNALTYP ==1)
03731     hTCSNewMetaFileDC  = CreateMetaFile (FilNam);
03732     if (hTCSNewMetaFileDC == NULL) {
03733      iErr= WRN_HDCFILOPN;
03734      TCSGraphicError (iErr,"");
03735      return;                                 /* Error during Open -> ret */
03736     }
03737
03738     hmf = CloseMetaFile (hTCSMetaFileDC);       /* Metafile für WM_PAINT */
03739
03740     SetWindowExtEx (hTCSNewMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
03741     SetWindowOrgEx (hTCSNewMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
03742
03743     ScaleViewportExtEx (hTCSNewMetaFileDC, 1,1,-1,1,NULL);
03744
03745     hWindowRegion= CreateRectRgn(TCSrect.left, TCSrect.top, TCSrect.right,TCSrect.bottom);
03746     hBack= CreateSolidBrush (dwColorTable[TCSBackgroundColour]); /* rechts,oben */
03747     FillRgn (hTCSNewMetaFileDC, hWindowRegion, hBack);  /* nicht eingeschlossen */
03748     #if !defined(__WIN32__) && !defined(_WIN32)
03749      DeleteBrush (hBack);
03750      DeleteRgn (hWindowRegion);                  /* Resourcen freigeben */
03751     #else
03752      DeleteObject (hBack);
03753      DeleteObject (hWindowRegion);
03754     #endif
03755
03756     PlayMetaFile (hTCSNewMetaFileDC, hmf);
03757     hmf1= CloseMetaFile (hTCSNewMetaFileDC);
03758     if (hmf1 == NULL) {
03759      iErr= WRN_HDCFILWRT;
03760      TCSGraphicError (iErr,"");
03761      return;                                 /* Error during Write -> ret */
03762     } else {
03763      DeleteMetaFile (hmf1); /* Freigabe Resourcen, nicht Löschen des Files! */
03764     }
03765
03766     hTCSNewMetaFileDC  = CreateMetaFile (NULL); /* 16bit Windows Metafile */
03767     PlayMetaFile (hTCSNewMetaFileDC, hmf);      /* für neues Journalfile */
03768     DeleteMetaFile (hmf);                       /* alter Status Bildschirm */
03769     hTCSMetaFileDC = hTCSNewMetaFileDC;         /* bereit Weiterzeichnen */
03770
03771 #elif (JOURNALTYP == 2)
03772     hmf = CloseEnhMetaFile (hTCSMetaFileDC);    /* Metafile für WM_PAINT */
03773     hmf1  = CopyEnhMetaFile (hmf, FilNam);
03774     if (hmf1 == NULL) {
03775      ErrorCode= GetLastError(); // immer win32 bei emf
03776 //   if (ErrorCode == ERROR_CLASS_ALREADY_EXISTS) {
03777 //    Hier bei Bedarf Fehlerbehandlung einführen
03778 //   } else {
03779       FormatMessage(
03780          FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
03781          NULL,
03782          ErrorCode,
03783          MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
03784          (LPTSTR) &lpMsgBuf,
03785          0,
03786          NULL
03787       );
03788       MessageBox (NULL, lpMsgBuf, szTCSWindowName, MB_ICONSTOP);
03789       LocalFree( lpMsgBuf ); // Free the buffer
03790 //   } // Ende der Fehlerbehandlung
03791      iErr= WRN_HDCFILOPN;
03792      TCSGraphicError (iErr,"");
03793      return;                                 /* Error during Open -> ret */
03794     }
03795     DeleteEnhMetaFile (hmf1); /* Handle freigeben, File  nicht geloescht! */
03796
03797     GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
03798     hTCSNewMetaFileDC  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
03799                          _T("TCS for Windows\0Subroutine HardCopy\0"));
03800     SetMapMode (hTCSNewMetaFileDC, MM_ANISOTROPIC);
03801     SetViewportExtEx (hTCSNewMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
03802     SetViewportOrgEx (hTCSNewMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
03803     SetWindowExtEx (hTCSNewMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
03804     SetWindowOrgEx (hTCSNewMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
03805
03806     PlayEnhMetaFile (hTCSNewMetaFileDC, hmf, &TCSrect); // neues Journal
```

```
03807
03808     DeleteEnhMetaFile (hmf);                      // alter Status Bildschirm
03809     hTCSMetaFileDC = hTCSNewMetaFileDC;           // bereit zum Weiterzeichnen
03810
03811     SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
03812     SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
03813     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
03814     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
03815
03816     #if !defined(__WIN32__) && !defined(_WIN32)
03817      SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
03818     #else
03819      SelectObject (hTCSMetaFileDC, hTCSFont);
03820     #endif
03821     SetBkMode (hTCSMetaFileDC, TRANSPARENT );   // Metafile weitergegeben !
03822     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); // CP
03823     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
03824     #if !defined(__WIN32__) && !defined(_WIN32)
03825      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
03826     #else
03827      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
03828     #endif
03829
03830 #elif (JOURNALTYP == 3)
03831     fHandle= fopen(FilNam, "w+");
03832     if ( fHandle == NULL) {
03833      iErr= WRN_HDCFILOPN;
03834      TCSGraphicError (iErr,"");
03835      return;                                  /* Error during Open -> ret */
03836     }
03837
03838     SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, hTCSJournal, previous, next, xJournalEntry)
03839
03840     while (xJournalEntry != NULL) {
03841      fprintf( fHandle, "%02i#%04i-%03i\n", xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2
      );
03842
03843 #ifdef TRACE_CALLS
03844      switch (xJournalEntry->action) {
03845        case XACTION_INITT: {
03846         printf  ("%s § \n","Initt ");
03847         break;
03848        }
03849        case XACTION_ERASE: {
03850         printf  ("%s § \n","Erase ");
03851         break;
03852        }
03853        case XACTION_MOVABS: {
03854         printf  ("%s x:%i - y: %i § \n","MovAbs ", xJournalEntry->i1, xJournalEntry->i2);
03855         break;
03856        }
03857        case XACTION_DRWABS: {
03858         printf  ("%s x:%i - y: %i § \n","DrwAbs ", xJournalEntry->i1, xJournalEntry->i2);
03859         break;
03860        }
03861        case XACTION_DSHSTYLE: {
03862         printf  ("%s x:%i § \n","DshStyle ", xJournalEntry->i1);
03863         break;
03864        }
03865        case XACTION_DSHABS: {
03866         printf  ("%s x:%i - y: %i § \n","DshAbs ", xJournalEntry->i1, xJournalEntry->i2);
03867         break;
03868        }
03869        case XACTION_PNTABS: {
03870         printf  ("%s x:%i - y: %i § \n","PntAbs ", xJournalEntry->i1, xJournalEntry->i2);
03871         break;
03872        }
03873        case XACTION_BCKCOL: {
03874         printf  ("%s x:%i § \n","BckCol ", xJournalEntry->i1);
03875         break;
03876        }
03877        case XACTION_TXTCOL: {
03878         printf  ("%s x:%i § \n","TxtCol ", xJournalEntry->i1);
03879         break;
03880        }
03881        case XACTION_LINCOL: {
03882         printf  ("%s x:%i § \n","LinCol ", xJournalEntry->i1);
03883         break;
03884        }
03885        case XACTION_FONTATTR: {
03886         printf  ("%s x:%i - %i § \n","Fontattr ", xJournalEntry->i1, xJournalEntry->i2);
03887         break;
03888        }
03889        case XACTION_GTEXT: {
03890         printf  ("%s iL:%i - C0: %i [ %c ] § \n","GText ", xJournalEntry->i1, xJournalEntry->i2,
03891                 xJournalEntry->i2);
03892         break;
```

```
03893         }
03894         case XACTION_ASCII: {
03895          printf  ("%s C1:%i - C2: %i [ %c %c ] $ \n","ASCII ", xJournalEntry->i1, xJournalEntry->i2,
03896                                 xJournalEntry->i1, xJournalEntry->i2);
03897          break;
03898         }
03899         default: {
03900          printf ("??? %i ??? \n", xJournalEntry->action) ;
03901          break;
03902         }
03903       }
03904 #endif // TRACE_CALLS
03905       xJournalEntry= xJournalEntry -> previous;
03906      }
03907      fclose (fHandle);
03908 #endif // Journaltyp=3
03909      ShowWindow (hTCSstatWindow, SW_HIDE);
03910      return;
03911 }
03912
03913
03914
03915 /*
03916 ---- subroutine LIB_MOVC3 fuer Watcom- und GNU-Compiler ---------------
03917 Hier nicht benoetigt, nur wg. Kompatibilitaet zur DOS-Version enthalten
03918 */
03919
03920
03921 extern void lib_movc3 (FTNINT *len,FTNSTRPAR *sou,FTNSTRPAR *dst
03922                                  FTNSTRPAR_TAIL(sou)  FTNSTRPAR_TAIL(dst) )
03923
03924 {
03925 int n;
03926      if (FTNSTRPARA(dst) <= FTNSTRPARA(sou) ) {
03927       for (n=0; n<*len; n++) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
03928      } else {
03929       for (n= (*len)-1; n>=0; n--) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
03930      };
03931 }
```

6.38 TCSdWINc.h File Reference

MS Windows Port: Low-Level Driver.

Macros

- #define false 0
- #define true !false
- #define TEK_XMAX 1023
- #define TEK_YMAX 780
- #define HiRes(iX) iX
- #define LoRes(iX) iX
- #define MOUSE_XMAX 65535 /* Mousekoordinatensystem (Mickeys) */
- #define MOUSE_YMAX 65535 /* s. MS-Dokumentation mouse_event */
- #define TCS_WM_COPY 0x0401 /* Raum für Applikationen: 0x0400-0x7fff */
- #define STAT_MAXROWS 25 /* Gemerkte Statuszeilen (scrollbar) */
- #define STAT_MAXCOLUMNS 80
- #define STAT_MINLINES 1 /* Default: Angezeigte Statuszeilen */
- #define STAT_ADDLINES 9 /* Zusätzlich durch Mausziehen anzeigbar */
- #define STAT_PAGESIZ 5 /* Scrollschritte bei großem Statusfenster */
- #define TCS_REL_CHR_HEIGHT 1.0f
- #define TCS_REL_CHR_SPACE 1.1f /* Zeilenabstand */
- #define TCS_WINDOW_NAMELEN 255
- #define TCS_FILE_NAMELEN 128
- #define TCS_MESSAGELEN 80
- #define TCS_MENUENTRY_LEN 15
- #define INIFILEXTTOKEN _T(".%") /* Token fuer den Filenamenparser */
- #define PROGDIRTOKEN _T("%:")
- #define TCS_WINDOWCLASS _T("Graph2DWindow")

- #define TCS_STAT_WINDOWCLASS _T("Graph2DstatWindow")
- #define TCS_DEFAULT_MAINWINDOWCLASS _T("WinMainFTN77")
- #define TCS_INIFILE_NAME _T("Graph2D")
- #define TCS_WINDOW_ICON _T("Graph2DIcon")
- #define TCS_WINDOW_ICONS _T("Graph2DIconS")
- #define XACTION_INITT 1
- #define XACTION_ERASE 2
- #define XACTION_MOVABS 3
- #define XACTION_DRWABS 4
- #define XACTION_DSHSTYLE 5
- #define XACTION_DSHABS 6
- #define XACTION_PNTABS 7
- #define XACTION_GTEXT 8
- #define XACTION_ASCII 9
- #define XACTION_BCKCOL 10
- #define XACTION_LINCOL 11
- #define XACTION_TXTCOL 12
- #define XACTION_FONTATTR 13
- #define XACTION_NOOP 14
- #define WRN_NOMSG 1
- #define ERR_UNKNGRAPHCARD 2
- #define ERR_NOFNTFIL 3
- #define ERR_NOFNT 4
- #define MSG_NOMOUSE 5
- #define WRN_HDCFILOPN 6
- #define WRN_HDCFILWRT 7
- #define WRN_HDCINTERN 8
- #define MSG_USR 9
- #define MSG_HDCACT 10
- #define WRN_USRPRESSANY 11
- #define ERR_EXIT 12
- #define WRN_COPYNOMEM 13
- #define WRN_COPYLOCK 14
- #define WRN_JOUCREATE 15
- #define WRN_JOUENTRY 16
- #define WRN_JOUADD 17
- #define WRN_JOUCLR 18
- #define WRN_JOUUNKWN 19
- #define ERR_XMLPARSER 20
- #define ERR_XMLOPEN 21
- #define ERR_UNKNAUDIO 22
- #define MSG_USR2 23
- #define WRN_INI2 24
- #define MSG_MAXERRNO 25
- #define TCS_INISECT0 "Graph2D"
- #define TCS_INISECT1 _T("Names")
- #define TCS_INIVAR_WINNAM _T("G2dGraphic")
- #define TCS_WINDOW_NAME _T("Graphics")
- #define TCS_INIVAR_STATNAM _T("G2dStatus")
- #define TCS_STATWINDOW_NAME _T("System Messages")
- #define TCS_INIVAR_HDCNAM _T("G2dHardcopy")
- #define TCS_HDCFILE_NAME _T("HDC%03i.UNKNOWN")
- #define TCS_INIVAR_MAINWINNAM _T("G2dMainWindow")
- #define TCS_MAINWINDOW_NAME _T("%:")
- #define TCS_INISECT2 _T("Layout")

- #define TCS_INIVAR_COPMEN _T("G2dSysMenuCopy")
- #define TCS_INIDEF_COPMEN _T("Copy")
- #define TCS_INIVAR_FONT _T("G2dGraphicFont")
- #define TCS_INIDEF_FONT _T("Arial Terminal")
- #define TCS_INIVAR_SYSFONT _T("G2dSystemFont")
- #define TCS_INIDEF_SYSFONT _T("Arial Terminal")
- #define TCS_INIVAR_ICONNAM _T("G2dIcon")
- #define TCS_ICONFILE_NAME _T("")
- #define TCS_INIVAR_WINPOSX _T("G2dGraphicPosX")
- #define TCS_INIDEF_WINPOSX 0
- #define TCS_INIVAR_WINPOSY _T("G2dGraphicPosY")
- #define TCS_INIDEF_WINPOSY 0
- #define TCS_INIVAR_WINSIZX _T("G2dGraphicSizeX")
- #define TCS_INIDEF_WINSIZX 100
- #define TCS_INIVAR_WINSIZY _T("G2dGraphicSizeY")
- #define TCS_INIDEF_WINSIZY 100
- #define TCS_INIVAR_STATPOSX _T("G2dStatusPosX")
- #define TCS_INIDEF_STATPOSX 0
- #define TCS_INIVAR_STATPOSY _T("G2dStatusPosY")
- #define TCS_INIDEF_STATPOSY 0
- #define TCS_INIVAR_STATSIZX _T("G2dStatusSizeX")
- #define TCS_INIDEF_STATSIZX 100
- #define TCS_INIVAR_STATSIZY _T("G2dStatusSizeY")
- #define TCS_INIDEF_STATSIZY 100
- #define TCS_INIVAR_LINCOL _T("G2dLinCol")
- #define TCS_INIDEF_LINCOL 1
- #define TCS_INIVAR_TXTCOL _T("G2dTxtCol")
- #define TCS_INIDEF_TXTCOL 1
- #define TCS_INIVAR_BCKCOL _T("G2dBckCol")
- #define TCS_INIDEF_BCKCOL 0
- #define TCS_INISECT3 _T("Messages")
- #define TCS_INIVAR_HDCOPN _T("G2dHdcOpen")
- #define TCS_INIDEF_HDCOPN _T("GRAPH2D HARDCOPY: Error during OPEN.")
- #define TCS_INIVAR_HDCOPNL _T("G2dHdcOpenL")
- #define TCS_INIDEF_HDCOPNL 5
- #define TCS_INIVAR_HDCWRT _T("G2dHdcWrite")
- #define TCS_INIDEF_HDCWRT _T("GRAPH2D HARDCOPY: Error during WRITE.")
- #define TCS_INIVAR_HDCWRTL _T("G2dHdcWriteL")
- #define TCS_INIDEF_HDCWRTL 5
- #define TCS_INIVAR_HDCINT _T("G2dHdcIntern")
- #define TCS_INIDEF_HDCINT _T("GRAPH2D HARDCOPY: Internal Error.")
- #define TCS_INIVAR_HDCINTL _T("G2dHdcInternL")
- #define TCS_INIDEF_HDCINTL 5
- #define TCS_INIVAR_USR _T("G2dUser")
- #define TCS_INIDEF_USR _T("%s")
- #define TCS_INIVAR_USRL _T("G2dUserL")
- #define TCS_INIDEF_USRL 5
- #define TCS_INIVAR_HDCACT _T("G2dHdcActive")
- #define TCS_INIDEF_HDCACT _T("Hardcopy in progress: File %s created.")
- #define TCS_INIVAR_HDCACTL _T("G2dHdcActiveL")
- #define TCS_INIDEF_HDCACTL 1
- #define TCS_INIVAR_USRWRN _T("G2dPressAny")
- #define TCS_INIDEF_USRWRN _T("Press any key to continue.")
- #define TCS_INIVAR_USRWRNL _T("G2dPressAnyL")
- #define TCS_INIDEF_USRWRNL 5

- #define TCS_INIVAR_EXIT _T("G2dExit")
- #define TCS_INIDEF_EXIT _T("Press any key to exit program.")
- #define TCS_INIVAR_EXITL _T("G2dExitL")
- #define TCS_INIDEF_EXITL 10
- #define TCS_INIVAR_COPMEM _T("G2dNoMemory")
- #define TCS_INIDEF_COPMEM _T("GRAPH2D Clipboard Manager: Out of Memory.")
- #define TCS_INIVAR_COPMEML _T("G2dNoMemoryL")
- #define TCS_INIDEF_COPMEML 1
- #define TCS_INIVAR_COPLCK _T("G2dClipLock")
- #define TCS_INIDEF_COPLCK _T("GRAPH2D Clipboard Manager: ClipBoard locked.")
- #define TCS_INIVAR_COPLCKL _T("G2dClipLockL")
- #define TCS_INIDEF_COPLCKL 1
- #define TCS_INIVAR_JOUCREATE _T("G2dJouCreate")
- #define TCS_INIDEF_JOUCREATE _T("GRAPH2D Error Creating Journal. Error-No: %s.")
- #define TCS_INIVAR_JOUCREATEL _T("G2dJouCreateL")
- #define TCS_INIDEF_JOUCREATEL 5
- #define TCS_INIVAR_JOUENTRY _T("G2dJouEntry")
- #define TCS_INIDEF_JOUENTRY _T("GRAPH2D Error Creating Journal Entry.")
- #define TCS_INIVAR_JOUENTRYL _T("G2dJouEntryL")
- #define TCS_INIDEF_JOUENTRYL 5
- #define TCS_INIVAR_JOUADD _T("G2dJouAdd")
- #define TCS_INIDEF_JOUADD _T("GRAPH2D Error Appending Journal Entry.")
- #define TCS_INIVAR_JOUADDL _T("G2dJouAddL")
- #define TCS_INIDEF_JOUADDL 5
- #define TCS_INIVAR_JOUCLR _T("G2dJouClr")
- #define TCS_INIDEF_JOUCLR _T("GRAPH2D Error Clearing Journal Entry.")
- #define TCS_INIVAR_JOUCLRL _T("G2dJouClrL")
- #define TCS_INIDEF_JOUCLRL 5
- #define TCS_INIVAR_JOUUNKWN _T("G2dJouEntryUnknwn")
- #define TCS_INIDEF_JOUUNKWN _T("GRAPH2D Unknown Journal Entry.")
- #define TCS_INIVAR_JOUUNKWNL _T("G2dJouEntryUnknwnL")
- #define TCS_INIDEF_JOUUNKWNL 1
- #define TCS_INIVAR_XMLPARSER _T("G2dXMLerror")
- #define TCS_INIDEF_XMLPARSER _T("GRAPH2D Error parsing XML-File: %s")
- #define TCS_INIVAR_XMLPARSERL _T("G2dXMLerrorL")
- #define TCS_INIDEF_XMLPARSERL 8
- #define TCS_INIVAR_XMLOPEN _T("G2dXMLopen")
- #define TCS_INIDEF_XMLOPEN _T("GRAPH2D Error opening %s")
- #define TCS_INIVAR_XMLOPENL _T("G2dXMLerrorL")
- #define TCS_INIDEF_XMLOPENL 8
- #define TCS_INIVAR_USR2 _T("G2dUser2")
- #define TCS_INIDEF_USR2 _T("%s")
- #define TCS_INIVAR_USR2L _T("G2dUser2L")
- #define TCS_INIDEF_USR2L 5
- #define TCS_INIVAR_INI2 _T("G2d2xInitt")
- #define TCS_INIDEF_INI2 _T("%s")
- #define TCS_INIVAR_INI2L _T("G2d2xInittL")
- #define TCS_INIDEF_INI2L 5
- #define LPTSTR LPSTR
- #define EXPORT16 __export /* __export bei virtuellem Adressraum unnötig */
- #define SM_CXMAXIMIZED SM_CXFULLSCREEN /* notduerftiger Ersatz für ... */
- #define SM_CYMAXIMIZED SM_CYFULLSCREEN /* ...Win32 Funktion */
- #define GetCommandLine() "WinApp" /* dito */

Typedefs

- typedef int bool
- typedef char TCHAR
- typedef char * PTCHAR

Functions

- void bell (void)
- void outtext (FTNSTRPAR *ftn_string FTNSTRPAR_TAIL(ftn_string))
- void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string, FTNINT *iL FTNSTRPAR_TAIL(ftn_string))
- void tinput (FTNINT *ic)
- void finitt ()

6.38.1 Detailed Description

MS Windows Port: Low-Level Driver.

Version

1.9

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Headerfile for TCSdWIN.c
Definition in file TCSdWINc.h.

6.38.2 Macro Definition Documentation

6.38.2.1 ERR_EXIT

```
#define ERR_EXIT 12
```

Definition at line 107 of file TCSdWINc.h.

6.38.2.2 ERR_NOFNT

```
#define ERR_NOFNT 4
```

Definition at line 99 of file TCSdWINc.h.

6.38.2.3 ERR_NOFNTFIL

```
#define ERR_NOFNTFIL 3
```

Definition at line 98 of file TCSdWINc.h.

6.38.2.4 ERR_UNKNAUDIO

```
#define ERR_UNKNAUDIO 22
```

Definition at line 117 of file TCSdWINc.h.

6.38.2.5 ERR_UNKNGRAPHCARD

```
#define ERR_UNKNGRAPHCARD 2
```
Definition at line 97 of file TCSdWINc.h.

6.38.2.6 ERR_XMLOPEN

```
#define ERR_XMLOPEN 21
```
Definition at line 116 of file TCSdWINc.h.

6.38.2.7 ERR_XMLPARSER

```
#define ERR_XMLPARSER 20
```
Definition at line 115 of file TCSdWINc.h.

6.38.2.8 EXPORT16

```
#define EXPORT16 __export /* __export bei virtuellem Adressraum unnötig */
```
Definition at line 266 of file TCSdWINc.h.

6.38.2.9 false

```
#define false 0
```
Definition at line 18 of file TCSdWINc.h.

6.38.2.10 GetCommandLine

```
#define GetCommandLine( ) "WinApp" /* dito */
```
Definition at line 269 of file TCSdWINc.h.

6.38.2.11 HiRes

```
#define HiRes(
              iX ) iX
```
Definition at line 33 of file TCSdWINc.h.

6.38.2.12 INIFILEXTTOKEN

```
#define INIFILEXTTOKEN _T(".%") /* Token fuer den Filenamenparser */
```
Definition at line 63 of file TCSdWINc.h.

6.38.2.13 LoRes

```
#define LoRes(
              iX ) iX
```
Definition at line 34 of file TCSdWINc.h.

6.38.2.14 LPTSTR

```
#define LPTSTR LPSTR
```
Definition at line 264 of file TCSdWINc.h.

6.38.2.15 MOUSE_XMAX

```
#define MOUSE_XMAX 65535 /* Mousekoordinatensystem (Mickeys) */
```
Definition at line 39 of file TCSdWINc.h.

6.38.2.16 MOUSE_YMAX

```
#define MOUSE_YMAX 65535 /* s.  MS-Dokumentation mouse_event */
```
Definition at line 40 of file TCSdWINc.h.

6.38.2.17 MSG_HDCACT

```
#define MSG_HDCACT 10
```
Definition at line 105 of file TCSdWINc.h.

6.38.2.18 MSG_MAXERRNO

```
#define MSG_MAXERRNO 25
```
Definition at line 120 of file TCSdWINc.h.

6.38.2.19 MSG_NOMOUSE

```
#define MSG_NOMOUSE 5
```
Definition at line 100 of file TCSdWINc.h.

6.38.2.20 MSG_USR

```
#define MSG_USR 9
```
Definition at line 104 of file TCSdWINc.h.

6.38.2.21 MSG_USR2

```
#define MSG_USR2 23
```
Definition at line 118 of file TCSdWINc.h.

6.38.2.22 PROGDIRTOKEN

```
#define PROGDIRTOKEN _T("%:")
```
Definition at line 64 of file TCSdWINc.h.

6.38.2.23 SM_CXMAXIMIZED

```
#define SM_CXMAXIMIZED SM_CXFULLSCREEN /* notduerftiger Ersatz für ...  */
```
Definition at line 267 of file TCSdWINc.h.

6.38.2.24 SM_CYMAXIMIZED

```
#define SM_CYMAXIMIZED SM_CYFULLSCREEN /* ...Win32 Funktion */
```
Definition at line 268 of file TCSdWINc.h.

6.38.2.25 STAT_ADDLINES

```
#define STAT_ADDLINES 9 /* Zusätzlich durch Mausziehen anzeigbar */
```
Definition at line 52 of file TCSdWINc.h.

6.38.2.26 STAT_MAXCOLUMNS

```
#define STAT_MAXCOLUMNS 80
```
Definition at line 50 of file TCSdWINc.h.

6.38.2.27 STAT_MAXROWS

```
#define STAT_MAXROWS 25 /* Gemerkte Statuszeilen (scrollbar) */
```
Definition at line 49 of file TCSdWINc.h.

6.38.2.28 STAT_MINLINES

```
#define STAT_MINLINES 1 /* Default:  Angezeigte Statuszeilen */
```
Definition at line 51 of file TCSdWINc.h.

6.38.2.29 STAT_PAGESIZ

```
#define STAT_PAGESIZ 5 /* Scrollschritte bei großem Statusfenster */
```
Definition at line 53 of file TCSdWINc.h.

6.38.2.30 TCS_DEFAULT_MAINWINDOWCLASS

```
#define TCS_DEFAULT_MAINWINDOWCLASS _T("WinMainFTN77")
```
Definition at line 68 of file TCSdWINc.h.

6.38.2.31 TCS_FILE_NAMELEN

```
#define TCS_FILE_NAMELEN 128
```
Definition at line 59 of file TCSdWINc.h.

6.38.2.32 TCS_HDCFILE_NAME

```
#define TCS_HDCFILE_NAME _T("HDC%03i.UNKNOWN")
```
Definition at line 146 of file TCSdWINc.h.

6.38.2.33 TCS_ICONFILE_NAME

```
#define TCS_ICONFILE_NAME _T("")
```
Definition at line 159 of file TCSdWINc.h.

6.38.2.34 TCS_INIDEF_BCKCOL

```
#define TCS_INIDEF_BCKCOL 0
```
Definition at line 181 of file TCSdWINc.h.

6.38.2.35 TCS_INIDEF_COPLCK

```
#define TCS_INIDEF_COPLCK _T("GRAPH2D Clipboard Manager:  ClipBoard locked.")
```
Definition at line 217 of file TCSdWINc.h.

6.38.2.36 TCS_INIDEF_COPLCKL

```
#define TCS_INIDEF_COPLCKL 1
```
Definition at line 219 of file TCSdWINc.h.

6.38.2.37 TCS_INIDEF_COPMEM

```
#define TCS_INIDEF_COPMEM _T("GRAPH2D Clipboard Manager:  Out of Memory.")
```
Definition at line 213 of file TCSdWINc.h.

6.38.2.38 TCS_INIDEF_COPMEML

```
#define TCS_INIDEF_COPMEML 1
```
Definition at line 215 of file TCSdWINc.h.

6.38.2.39 TCS_INIDEF_COPMEN

```
#define TCS_INIDEF_COPMEN _T("Copy")
```
Definition at line 153 of file TCSdWINc.h.

6.38.2.40 TCS_INIDEF_EXIT

```
#define TCS_INIDEF_EXIT _T("Press any key to exit program.")
```
Definition at line 209 of file TCSdWINc.h.

6.38.2.41 TCS_INIDEF_EXITL

```
#define TCS_INIDEF_EXITL 10
```
Definition at line 211 of file TCSdWINc.h.

6.38.2.42 TCS_INIDEF_FONT

```
#define TCS_INIDEF_FONT _T("Arial Terminal")
```
Definition at line 155 of file TCSdWINc.h.

6.38.2.43 TCS_INIDEF_HDCACT

```
#define TCS_INIDEF_HDCACT _T("Hardcopy in progress:  File %s created.")
```
Definition at line 201 of file TCSdWINc.h.

6.38.2.44 TCS_INIDEF_HDCACTL

```
#define TCS_INIDEF_HDCACTL 1
```
Definition at line 203 of file TCSdWINc.h.

6.38.2.45 TCS_INIDEF_HDCINT

```
#define TCS_INIDEF_HDCINT _T("GRAPH2D HARDCOPY: Internal Error.")
```
Definition at line 193 of file TCSdWINc.h.

6.38.2.46 TCS_INIDEF_HDCINTL

```
#define TCS_INIDEF_HDCINTL 5
```
Definition at line 195 of file TCSdWINc.h.

6.38.2.47 TCS_INIDEF_HDCOPN

```
#define TCS_INIDEF_HDCOPN _T("GRAPH2D HARDCOPY: Error during OPEN.")
```
Definition at line 185 of file TCSdWINc.h.

6.38.2.48 TCS_INIDEF_HDCOPNL

```
#define TCS_INIDEF_HDCOPNL 5
```
Definition at line 187 of file TCSdWINc.h.

6.38.2.49 TCS_INIDEF_HDCWRT

```
#define TCS_INIDEF_HDCWRT _T("GRAPH2D HARDCOPY: Error during WRITE.")
```
Definition at line 189 of file TCSdWINc.h.

6.38.2.50 TCS_INIDEF_HDCWRTL

```
#define TCS_INIDEF_HDCWRTL 5
```
Definition at line 191 of file TCSdWINc.h.

6.38.2.51 TCS_INIDEF_INI2

```
#define TCS_INIDEF_INI2 _T("%s")
```
Definition at line 253 of file TCSdWINc.h.

6.38.2.52 TCS_INIDEF_INI2L

```
#define TCS_INIDEF_INI2L 5
```
Definition at line 255 of file TCSdWINc.h.

6.38.2.53 TCS_INIDEF_JOUADD

```
#define TCS_INIDEF_JOUADD _T("GRAPH2D Error Appending Journal Entry.")
```
Definition at line 229 of file TCSdWINc.h.

6.38.2.54 TCS_INIDEF_JOUADDL

```
#define TCS_INIDEF_JOUADDL 5
```
Definition at line 231 of file TCSdWINc.h.

6.38.2.55 TCS_INIDEF_JOUCLR

```
#define TCS_INIDEF_JOUCLR _T("GRAPH2D Error Clearing Journal Entry.")
```
Definition at line 233 of file TCSdWINc.h.

6.38.2.56 TCS_INIDEF_JOUCLRL

```
#define TCS_INIDEF_JOUCLRL 5
```
Definition at line 235 of file TCSdWINc.h.

6.38.2.57 TCS_INIDEF_JOUCREATE

```
#define TCS_INIDEF_JOUCREATE _T("GRAPH2D Error Creating Journal.  Error-No:  %s.")
```
Definition at line 221 of file TCSdWINc.h.

6.38.2.58 TCS_INIDEF_JOUCREATEL

```
#define TCS_INIDEF_JOUCREATEL 5
```
Definition at line 223 of file TCSdWINc.h.

6.38.2.59 TCS_INIDEF_JOUENTRY

```
#define TCS_INIDEF_JOUENTRY _T("GRAPH2D Error Creating Journal Entry.")
```
Definition at line 225 of file TCSdWINc.h.

6.38.2.60 TCS_INIDEF_JOUENTRYL

```
#define TCS_INIDEF_JOUENTRYL 5
```
Definition at line 227 of file TCSdWINc.h.

6.38.2.61 TCS_INIDEF_JOUUNKWN

```
#define TCS_INIDEF_JOUUNKWN _T("GRAPH2D Unknown Journal Entry.")
```
Definition at line 237 of file TCSdWINc.h.

6.38.2.62 TCS_INIDEF_JOUUNKWNL

```
#define TCS_INIDEF_JOUUNKWNL 1
```
Definition at line 239 of file TCSdWINc.h.

6.38.2.63 TCS_INIDEF_LINCOL

```
#define TCS_INIDEF_LINCOL 1
```
Definition at line 177 of file TCSdWINc.h.

6.38.2.64 TCS_INIDEF_STATPOSX

```
#define TCS_INIDEF_STATPOSX 0
```
Definition at line 169 of file TCSdWINc.h.

6.38.2.65 TCS_INIDEF_STATPOSY

```
#define TCS_INIDEF_STATPOSY 0
```
Definition at line 171 of file TCSdWINc.h.

6.38.2.66 TCS_INIDEF_STATSIZX

```
#define TCS_INIDEF_STATSIZX 100
```
Definition at line 173 of file TCSdWINc.h.

6.38.2.67 TCS_INIDEF_STATSIZY

```
#define TCS_INIDEF_STATSIZY 100
```
Definition at line 175 of file TCSdWINc.h.

6.38.2.68 TCS_INIDEF_SYSFONT

```
#define TCS_INIDEF_SYSFONT _T("Arial Terminal")
```
Definition at line 157 of file TCSdWINc.h.

6.38.2.69 TCS_INIDEF_TXTCOL

```
#define TCS_INIDEF_TXTCOL 1
```
Definition at line 179 of file TCSdWINc.h.

6.38.2.70 TCS_INIDEF_USR

```
#define TCS_INIDEF_USR _T("%s")
```
Definition at line 197 of file TCSdWINc.h.

6.38.2.71 TCS_INIDEF_USR2

```
#define TCS_INIDEF_USR2 _T("%s")
```
Definition at line 249 of file TCSdWINc.h.

6.38.2.72 TCS_INIDEF_USR2L

```
#define TCS_INIDEF_USR2L 5
```
Definition at line 251 of file TCSdWINc.h.

6.38.2.73 TCS_INIDEF_USRL

```
#define TCS_INIDEF_USRL 5
```
Definition at line 199 of file TCSdWINc.h.

6.38.2.74 TCS_INIDEF_USRWRN

```
#define TCS_INIDEF_USRWRN _T("Press any key to continue.")
```
Definition at line 205 of file TCSdWINc.h.

6.38.2.75 TCS_INIDEF_USRWRNL

```
#define TCS_INIDEF_USRWRNL 5
```
Definition at line 207 of file TCSdWINc.h.

6.38.2.76 TCS_INIDEF_WINPOSX

```
#define TCS_INIDEF_WINPOSX 0
```
Definition at line 161 of file TCSdWINc.h.

6.38.2.77 TCS_INIDEF_WINPOSY

```
#define TCS_INIDEF_WINPOSY 0
```
Definition at line 163 of file TCSdWINc.h.

6.38.2.78 TCS_INIDEF_WINSIZX

```
#define TCS_INIDEF_WINSIZX 100
```
Definition at line 165 of file TCSdWINc.h.

6.38.2.79 TCS_INIDEF_WINSIZY

```
#define TCS_INIDEF_WINSIZY 100
```
Definition at line 167 of file TCSdWINc.h.

6.38.2.80 TCS_INIDEF_XMLOPEN

```
#define TCS_INIDEF_XMLOPEN _T("GRAPH2D Error opening %s")
```
Definition at line 245 of file TCSdWINc.h.

6.38.2.81 TCS_INIDEF_XMLOPENL

```
#define TCS_INIDEF_XMLOPENL 8
```
Definition at line 247 of file TCSdWINc.h.

6.38.2.82 TCS_INIDEF_XMLPARSER

```
#define TCS_INIDEF_XMLPARSER _T("GRAPH2D Error parsing XML-File:  %s")
```
Definition at line 241 of file TCSdWINc.h.

6.38.2.83 TCS_INIDEF_XMLPARSERL

```
#define TCS_INIDEF_XMLPARSERL 8
```
Definition at line 243 of file TCSdWINc.h.

6.38.2.84 TCS_INIFILE_NAME

```
#define TCS_INIFILE_NAME _T("Graph2D")
```
Definition at line 69 of file TCSdWINc.h.

6.38.2.85 TCS_INISECT0

```
#define TCS_INISECT0 "Graph2D"
```
Definition at line 131 of file TCSdWINc.h.

6.38.2.86 TCS_INISECT1

```
#define TCS_INISECT1 _T("Names")
```
Definition at line 133 of file TCSdWINc.h.

6.38.2.87 TCS_INISECT2

```
#define TCS_INISECT2 _T("Layout")
```
Definition at line 151 of file TCSdWINc.h.

6.38.2.88 TCS_INISECT3

```
#define TCS_INISECT3 _T("Messages")
```
Definition at line 183 of file TCSdWINc.h.

6.38.2.89 TCS_INIVAR_BCKCOL

```
#define TCS_INIVAR_BCKCOL _T("G2dBckCol")
```
Definition at line 180 of file TCSdWINc.h.

6.38.2.90 TCS_INIVAR_COPLCK

```
#define TCS_INIVAR_COPLCK _T("G2dClipLock")
```
Definition at line 216 of file TCSdWINc.h.

6.38.2.91 TCS_INIVAR_COPLCKL

```
#define TCS_INIVAR_COPLCKL _T("G2dClipLockL")
```
Definition at line 218 of file TCSdWINc.h.

6.38.2.92 TCS_INIVAR_COPMEM

```
#define TCS_INIVAR_COPMEM _T("G2dNoMemory")
```
Definition at line 212 of file TCSdWINc.h.

6.38.2.93 TCS_INIVAR_COPMEML

```
#define TCS_INIVAR_COPMEML _T("G2dNoMemoryL")
```
Definition at line 214 of file TCSdWINc.h.

6.38.2.94 TCS_INIVAR_COPMEN

```
#define TCS_INIVAR_COPMEN _T("G2dSysMenuCopy")
```
Definition at line 152 of file TCSdWINc.h.

6.38.2.95 TCS_INIVAR_EXIT

```
#define TCS_INIVAR_EXIT _T("G2dExit")
```
Definition at line 208 of file TCSdWINc.h.

6.38.2.96 TCS_INIVAR_EXITL

```
#define TCS_INIVAR_EXITL _T("G2dExitL")
```
Definition at line 210 of file TCSdWINc.h.

6.38.2.97 TCS_INIVAR_FONT

```
#define TCS_INIVAR_FONT _T("G2dGraphicFont")
```
Definition at line 154 of file TCSdWINc.h.

6.38.2.98 TCS_INIVAR_HDCACT

```
#define TCS_INIVAR_HDCACT _T("G2dHdcActive")
```
Definition at line 200 of file TCSdWINc.h.

6.38.2.99 TCS_INIVAR_HDCACTL

```
#define TCS_INIVAR_HDCACTL _T("G2dHdcActiveL")
```
Definition at line 202 of file TCSdWINc.h.

6.38.2.100 TCS_INIVAR_HDCINT

```
#define TCS_INIVAR_HDCINT _T("G2dHdcIntern")
```
Definition at line 192 of file TCSdWINc.h.

6.38.2.101 TCS_INIVAR_HDCINTL

```
#define TCS_INIVAR_HDCINTL _T("G2dHdcInternL")
```
Definition at line 194 of file TCSdWINc.h.

6.38.2.102 TCS_INIVAR_HDCNAM

```
#define TCS_INIVAR_HDCNAM _T("G2dHardcopy")
```
Definition at line 138 of file TCSdWINc.h.

6.38.2.103 TCS_INIVAR_HDCOPN

```
#define TCS_INIVAR_HDCOPN _T("G2dHdcOpen")
```
Definition at line 184 of file TCSdWINc.h.

6.38.2.104 TCS_INIVAR_HDCOPNL

```
#define TCS_INIVAR_HDCOPNL _T("G2dHdcOpenL")
```
Definition at line 186 of file TCSdWINc.h.

6.38.2.105 TCS_INIVAR_HDCWRT

```
#define TCS_INIVAR_HDCWRT _T("G2dHdcWrite")
```
Definition at line 188 of file TCSdWINc.h.

6.38.2.106 TCS_INIVAR_HDCWRTL

```
#define TCS_INIVAR_HDCWRTL _T("G2dHdcWriteL")
```
Definition at line 190 of file TCSdWINc.h.

6.38.2.107 TCS_INIVAR_ICONNAM

```
#define TCS_INIVAR_ICONNAM _T("G2dIcon")
```
Definition at line 158 of file TCSdWINc.h.

6.38.2.108 TCS_INIVAR_INI2

```
#define TCS_INIVAR_INI2 _T("G2d2xInitt")
```
Definition at line 252 of file TCSdWINc.h.

6.38.2.109 TCS_INIVAR_INI2L

```
#define TCS_INIVAR_INI2L _T("G2d2xInittL")
```
Definition at line 254 of file TCSdWINc.h.

6.38.2.110 TCS_INIVAR_JOUADD

```
#define TCS_INIVAR_JOUADD _T("G2dJouAdd")
```
Definition at line 228 of file TCSdWINc.h.

6.38.2.111 TCS_INIVAR_JOUADDL

```
#define TCS_INIVAR_JOUADDL _T("G2dJouAddL")
```
Definition at line 230 of file TCSdWINc.h.

6.38.2.112 TCS_INIVAR_JOUCLR

```
#define TCS_INIVAR_JOUCLR _T("G2dJouClr")
```
Definition at line 232 of file TCSdWINc.h.

6.38.2.113 TCS_INIVAR_JOUCLRL

```
#define TCS_INIVAR_JOUCLRL _T("G2dJouClrL")
```
Definition at line 234 of file TCSdWINc.h.

6.38.2.114 TCS_INIVAR_JOUCREATE

```
#define TCS_INIVAR_JOUCREATE _T("G2dJouCreate")
```
Definition at line 220 of file TCSdWINc.h.

6.38.2.115 TCS_INIVAR_JOUCREATEL

```
#define TCS_INIVAR_JOUCREATEL _T("G2dJouCreateL")
```
Definition at line 222 of file TCSdWINc.h.

6.38.2.116 TCS_INIVAR_JOUENTRY

```
#define TCS_INIVAR_JOUENTRY _T("G2dJouEntry")
```
Definition at line 224 of file TCSdWINc.h.

6.38.2.117 TCS_INIVAR_JOUENTRYL

```
#define TCS_INIVAR_JOUENTRYL _T("G2dJouEntryL")
```
Definition at line 226 of file TCSdWINc.h.

6.38.2.118 TCS_INIVAR_JOUUNKWN

```
#define TCS_INIVAR_JOUUNKWN _T("G2dJouEntryUnknwn")
```
Definition at line 236 of file TCSdWINc.h.

6.38.2.119 TCS_INIVAR_JOUUNKWNL

```
#define TCS_INIVAR_JOUUNKWNL _T("G2dJouEntryUnknwnL")
```
Definition at line 238 of file TCSdWINc.h.

6.38.2.120 TCS_INIVAR_LINCOL

```
#define TCS_INIVAR_LINCOL _T("G2dLinCol")
```
Definition at line 176 of file TCSdWINc.h.

6.38.2.121 TCS_INIVAR_MAINWINNAM

```
#define TCS_INIVAR_MAINWINNAM _T("G2dMainWindow")
```
Definition at line 148 of file TCSdWINc.h.

6.38.2.122 TCS_INIVAR_STATNAM

```
#define TCS_INIVAR_STATNAM _T("G2dStatus")
```
Definition at line 136 of file TCSdWINc.h.

6.38.2.123 TCS_INIVAR_STATPOSX

```
#define TCS_INIVAR_STATPOSX _T("G2dStatusPosX")
```
Definition at line 168 of file TCSdWINc.h.

6.38.2.124 TCS_INIVAR_STATPOSY

```
#define TCS_INIVAR_STATPOSY _T("G2dStatusPosY")
```
Definition at line 170 of file TCSdWINc.h.

6.38.2.125 TCS_INIVAR_STATSIZX

```
#define TCS_INIVAR_STATSIZX _T("G2dStatusSizeX")
```
Definition at line 172 of file TCSdWINc.h.

6.38.2.126 TCS_INIVAR_STATSIZY

```
#define TCS_INIVAR_STATSIZY _T("G2dStatusSizeY")
```
Definition at line 174 of file TCSdWINc.h.

6.38.2.127 TCS_INIVAR_SYSFONT

```
#define TCS_INIVAR_SYSFONT _T("G2dSystemFont")
```
Definition at line 156 of file TCSdWINc.h.

6.38.2.128 TCS_INIVAR_TXTCOL

```
#define TCS_INIVAR_TXTCOL _T("G2dTxtCol")
```
Definition at line 178 of file TCSdWINc.h.

6.38.2.129 TCS_INIVAR_USR

```
#define TCS_INIVAR_USR _T("G2dUser")
```
Definition at line 196 of file TCSdWINc.h.

6.38.2.130 TCS_INIVAR_USR2

```
#define TCS_INIVAR_USR2 _T("G2dUser2")
```
Definition at line 248 of file TCSdWINc.h.

6.38.2.131 TCS_INIVAR_USR2L

```
#define TCS_INIVAR_USR2L _T("G2dUser2L")
```
Definition at line 250 of file TCSdWINc.h.

6.38.2.132 TCS_INIVAR_USRL

```
#define TCS_INIVAR_USRL _T("G2dUserL")
```
Definition at line 198 of file TCSdWINc.h.

6.38.2.133 TCS_INIVAR_USRWRN

```
#define TCS_INIVAR_USRWRN _T("G2dPressAny")
```
Definition at line 204 of file TCSdWINc.h.

6.38.2.134 TCS_INIVAR_USRWRNL

```
#define TCS_INIVAR_USRWRNL _T("G2dPressAnyL")
```
Definition at line 206 of file TCSdWINc.h.

6.38.2.135 TCS_INIVAR_WINNAM

```
#define TCS_INIVAR_WINNAM _T("G2dGraphic")
```
Definition at line 134 of file TCSdWINc.h.

6.38.2.136 TCS_INIVAR_WINPOSX

```
#define TCS_INIVAR_WINPOSX _T("G2dGraphicPosX")
```
Definition at line 160 of file TCSdWINc.h.

6.38.2.137 TCS_INIVAR_WINPOSY

```
#define TCS_INIVAR_WINPOSY _T("G2dGraphicPosY")
```
Definition at line 162 of file TCSdWINc.h.

6.38.2.138 TCS_INIVAR_WINSIZX

```
#define TCS_INIVAR_WINSIZX _T("G2dGraphicSizeX")
```
Definition at line 164 of file TCSdWINc.h.

6.38.2.139 TCS_INIVAR_WINSIZY

```
#define TCS_INIVAR_WINSIZY _T("G2dGraphicSizeY")
```
Definition at line 166 of file TCSdWINc.h.

6.38.2.140 TCS_INIVAR_XMLOPEN

```
#define TCS_INIVAR_XMLOPEN _T("G2dXMLopen")
```
Definition at line 244 of file TCSdWINc.h.

6.38.2.141 TCS_INIVAR_XMLOPENL

```
#define TCS_INIVAR_XMLOPENL _T("G2dXMLerrorL")
```
Definition at line 246 of file TCSdWINc.h.

6.38.2.142 TCS_INIVAR_XMLPARSER

```
#define TCS_INIVAR_XMLPARSER _T("G2dXMLerror")
```
Definition at line 240 of file TCSdWINc.h.

6.38.2.143 TCS_INIVAR_XMLPARSERL

```
#define TCS_INIVAR_XMLPARSERL _T("G2dXMLerrorL")
```
Definition at line 242 of file TCSdWINc.h.

6.38.2.144 TCS_MAINWINDOW_NAME

```
#define TCS_MAINWINDOW_NAME _T("%:")
```
Definition at line 149 of file TCSdWINc.h.

6.38.2.145 TCS_MENUENTRY_LEN

```
#define TCS_MENUENTRY_LEN 15
```
Definition at line 61 of file TCSdWINc.h.

6.38.2.146 TCS_MESSAGELEN

```
#define TCS_MESSAGELEN 80
```
Definition at line 60 of file TCSdWINc.h.

6.38.2.147 TCS_REL_CHR_HEIGHT

```
#define TCS_REL_CHR_HEIGHT 1.0f
```
Definition at line 55 of file TCSdWINc.h.

6.38.2.148 TCS_REL_CHR_SPACE

```
#define TCS_REL_CHR_SPACE 1.1f /* Zeilenabstand */
```
Definition at line 56 of file TCSdWINc.h.

6.38.2.149 TCS_STAT_WINDOWCLASS

```
#define TCS_STAT_WINDOWCLASS _T("Graph2DstatWindow")
```
Definition at line 67 of file TCSdWINc.h.

6.38.2.150 TCS_STATWINDOW_NAME

```
#define TCS_STATWINDOW_NAME _T("System Messages")
```
Definition at line 137 of file TCSdWINc.h.

6.38.2.151 TCS_WINDOW_ICON

```
#define TCS_WINDOW_ICON _T("Graph2DIcon")
```
Definition at line 70 of file TCSdWINc.h.

6.38.2.152 TCS_WINDOW_ICONS

```
#define TCS_WINDOW_ICONS _T("Graph2DIconS")
```
Definition at line 71 of file TCSdWINc.h.

6.38.2.153 TCS_WINDOW_NAME

```
#define TCS_WINDOW_NAME _T("Graphics")
```
Definition at line 135 of file TCSdWINc.h.

6.38.2.154 TCS_WINDOW_NAMELEN

```
#define TCS_WINDOW_NAMELEN 255
```
Definition at line 58 of file TCSdWINc.h.

6.38.2.155 TCS_WINDOWCLASS

```
#define TCS_WINDOWCLASS _T("Graph2DWindow")
```
Definition at line 66 of file TCSdWINc.h.

6.38.2.156 TCS_WM_COPY

```
#define TCS_WM_COPY 0x0401 /* Raum für Applikationen:  0x0400-0x7fff */
```
Definition at line 42 of file TCSdWINc.h.

6.38.2.157 TEK_XMAX

```
#define TEK_XMAX 1023
```
Definition at line 24 of file TCSdWINc.h.

6.38.2.158 TEK_YMAX

```
#define TEK_YMAX 780
```
Definition at line 25 of file TCSdWINc.h.

6.38.2.159 true

```
#define true !false
```
Definition at line 19 of file TCSdWINc.h.

6.38.2.160 WRN_COPYLOCK

```
#define WRN_COPYLOCK 14
```
Definition at line 109 of file TCSdWINc.h.

6.38.2.161 WRN_COPYNOMEM

```
#define WRN_COPYNOMEM 13
```
Definition at line 108 of file TCSdWINc.h.

6.38.2.162 WRN_HDCFILOPN

```
#define WRN_HDCFILOPN 6
```
Definition at line 101 of file TCSdWINc.h.

6.38.2.163 WRN_HDCFILWRT

```
#define WRN_HDCFILWRT 7
```
Definition at line 102 of file TCSdWINc.h.

6.38.2.164 WRN_HDCINTERN

```
#define WRN_HDCINTERN 8
```
Definition at line 103 of file TCSdWINc.h.

6.38.2.165 WRN_INI2

```
#define WRN_INI2 24
```
Definition at line 119 of file TCSdWINc.h.

6.38.2.166 WRN_JOUADD

```
#define WRN_JOUADD 17
```
Definition at line 112 of file TCSdWINc.h.

6.38.2.167 WRN_JOUCLR

```
#define WRN_JOUCLR 18
```
Definition at line 113 of file TCSdWINc.h.

6.38.2.168 WRN_JOUCREATE

```
#define WRN_JOUCREATE 15
```
Definition at line 110 of file TCSdWINc.h.

6.38.2.169 WRN_JOUENTRY

```
#define WRN_JOUENTRY 16
```
Definition at line 111 of file TCSdWINc.h.

6.38.2.170 WRN_JOUUNKWN

```
#define WRN_JOUUNKWN 19
```
Definition at line 114 of file TCSdWINc.h.

6.38.2.171 WRN_NOMSG

```
#define WRN_NOMSG 1
```
Definition at line 96 of file TCSdWINc.h.

6.38.2.172 WRN_USRPRESSANY

```
#define WRN_USRPRESSANY 11
```
Definition at line 106 of file TCSdWINc.h.

6.38.2.173 XACTION_ASCII

```
#define XACTION_ASCII 9
```
Definition at line 85 of file TCSdWINc.h.

6.38.2.174 XACTION_BCKCOL

```
#define XACTION_BCKCOL 10
```
Definition at line 86 of file TCSdWINc.h.

6.38.2.175 XACTION_DRWABS

```
#define XACTION_DRWABS 4
```
Definition at line 80 of file TCSdWINc.h.

6.38.2.176 XACTION_DSHABS

```
#define XACTION_DSHABS 6
```
Definition at line 82 of file TCSdWINc.h.

6.38.2.177 XACTION_DSHSTYLE

```
#define XACTION_DSHSTYLE 5
```
Definition at line 81 of file TCSdWINc.h.

6.38.2.178 XACTION_ERASE

```
#define XACTION_ERASE 2
```
Definition at line 78 of file TCSdWINc.h.

6.38.2.179 XACTION_FONTATTR

```
#define XACTION_FONTATTR 13
```
Definition at line 89 of file TCSdWINc.h.

6.38.2.180 XACTION_GTEXT

```
#define XACTION_GTEXT 8
```
Definition at line 84 of file TCSdWINc.h.

6.38.2.181 XACTION_INITT

```
#define XACTION_INITT 1
```
Definition at line 77 of file TCSdWINc.h.

6.38.2.182 XACTION_LINCOL

```
#define XACTION_LINCOL 11
```
Definition at line 87 of file TCSdWINc.h.

6.38.2.183 XACTION_MOVABS

```
#define XACTION_MOVABS 3
```
Definition at line 79 of file TCSdWINc.h.

6.38.2.184 XACTION_NOOP

```
#define XACTION_NOOP 14
```
Definition at line 90 of file TCSdWINc.h.

6.38.2.185 XACTION_PNTABS

```
#define XACTION_PNTABS 7
```
Definition at line 83 of file TCSdWINc.h.

6.38.2.186 XACTION_TXTCOL

```
#define XACTION_TXTCOL 12
```
Definition at line 88 of file TCSdWINc.h.

6.38.3 Typedef Documentation

6.38.3.1 bool

```
typedef int bool
```
Definition at line 17 of file TCSdWINc.h.

6.38.3.2 PTCHAR

```
typedef char * PTCHAR
```
Definition at line 263 of file TCSdWINc.h.

6.38.3.3 TCHAR

```
typedef char TCHAR
```
Definition at line 263 of file TCSdWINc.h.

6.38.4 Function Documentation

6.38.4.1 bell()

```
void bell (
            void  )
```
Definition at line 3638 of file TCSdWINc.c.

6.38.4.2 finitt()

```
void finitt ( )
```
Definition at line 2520 of file TCSdWINc.c.

6.38.4.3 GraphicError()

```
void GraphicError (
            FTNINT * iErr,
            FTNSTRPAR * ftn_string,
            FTNINT *iL  FTNSTRPAR_TAILftn_string )
```
Definition at line 3676 of file TCSdWINc.c.

6.38.4.4 outtext()

```
void outtext (
            FTNSTRPAR *ftn_string  FTNSTRPAR_TAILftn_string )
```

Definition at line 3646 of file TCSdWINc.c.

6.38.4.5 tinput()

```
void tinput (
            FTNINT * ic )
```

Definition at line 3346 of file TCSdWINc.c.

6.39 TCSdWINc.h

```
00001 /** ****************************************************************************
00002 \file       TCSdWINc.h
00003 \brief      MS Windows Port: Low-Level Driver
00004 \version    1.9
00005 \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008           Headerfile zu TCSdWINc.c
00009 \~english
00010           Headerfile for TCSdWIN.c
00011 \~
00012
00013
00014 ***************************************************************************** */
00015
00016
00017  typedef int bool; // Typdefinition analog Cpp
00018  #define false 0
00019  #define true !false
00020
00021
00022 /* ---- Zeichenbereich im Tektronix-Koordinatensystem ---------------------- */
00023
00024 #define TEK_XMAX 1023
00025 #define TEK_YMAX 780
00026
00027 /* ---- Erhoehung der Zeichenauflösung fuer hochaufloesende Bildschirme --- */
00028
00029 #if defined PixFac
00030  #define HiRes(iX) (iX*PixFac)
00031  #define LoRes(iX) (iX/PixFac)
00032 #else
00033  #define HiRes(iX) iX
00034  #define LoRes(iX) iX
00035 #endif
00036
00037 /* ----- Systemparameter ------------------------------------------------- */
00038
00039 #define MOUSE_XMAX 65535        /* Mousekoordinatensystem (Mickeys) */
00040 #define MOUSE_YMAX 65535        /* s. MS-Dokumentation mouse_event */
00041
00042 #define TCS_WM_COPY 0x0401      /* Raum für Applikationen: 0x0400-0x7fff */
00043
00044
00045
00046
00047 /* ------------ Programmparameter ---------------------------------------- */
00048
00049 #define STAT_MAXROWS 25         /* Gemerkte Statuszeilen (scrollbar) */
00050 #define STAT_MAXCOLUMNS 80
00051 #define STAT_MINLINES 1         /* Default: Angezeigte Statuszeilen */
00052 #define STAT_ADDLINES 9         /* Zusätzlich durch Mausziehen anzeigbar */
00053 #define STAT_PAGESIZ 5          /* Scrollschritte bei großem Statusfenster */
00054
00055 #define TCS_REL_CHR_HEIGHT 1.0f
00056 #define TCS_REL_CHR_SPACE 1.1f  /* Zeilenabstand */
00057
00058 #define TCS_WINDOW_NAMELEN 255
00059 #define TCS_FILE_NAMELEN 128
00060 #define TCS_MESSAGELEN 80
00061 #define TCS_MENUENTRY_LEN 15
00062
00063 #define INIFILEXTTOKEN _T(".%")      /* Token fuer den Filenamenparser */
00064 #define PROGDIRTOKEN _T("%:")
00065
```

```
00066 #define TCS_WINDOWCLASS _T("Graph2DWindow")
00067 #define TCS_STAT_WINDOWCLASS _T("Graph2DstatWindow")
00068 #define TCS_DEFAULT_MAINWINDOWCLASS _T("WinMainFTN77")
00069 #define TCS_INIFILE_NAME _T("Graph2D")
00070 #define TCS_WINDOW_ICON _T("Graph2DIcon")
00071 #define TCS_WINDOW_ICONS _T("Graph2DIconS")
00072
00073
00074
00075 /* Actioncodes des Journalfiles */
00076
00077 #define XACTION_INITT       1
00078 #define XACTION_ERASE       2
00079 #define XACTION_MOVABS      3
00080 #define XACTION_DRWABS      4
00081 #define XACTION_DSHSTYLE    5
00082 #define XACTION_DSHABS      6
00083 #define XACTION_PNTABS      7
00084 #define XACTION_GTEXT       8
00085 #define XACTION_ASCII       9
00086 #define XACTION_BCKCOL      10
00087 #define XACTION_LINCOL      11
00088 #define XACTION_TXTCOL      12
00089 #define XACTION_FONTATTR    13
00090 #define XACTION_NOOP        14
00091
00092
00093
00094 /* Zuordnung Fehlernummern zu Meldungen */
00095
00096 #define WRN_NOMSG 1
00097 #define ERR_UNKNGRAPHCARD 2
00098 #define ERR_NOFNTFIL 3
00099 #define ERR_NOFNT 4
00100 #define MSG_NOMOUSE 5
00101 #define WRN_HDCFILOPN 6
00102 #define WRN_HDCFILWRT 7
00103 #define WRN_HDCINTERN 8
00104 #define MSG_USR 9
00105 #define MSG_HDCACT 10
00106 #define WRN_USRPRESSANY 11
00107 #define ERR_EXIT 12
00108 #define WRN_COPYNOMEM 13
00109 #define WRN_COPYLOCK 14
00110 #define WRN_JOUCREATE 15
00111 #define WRN_JOUENTRY 16
00112 #define WRN_JOUADD 17
00113 #define WRN_JOUCLR 18
00114 #define WRN_JOUUNKWN 19
00115 #define ERR_XMLPARSER 20
00116 #define ERR_XMLOPEN 21
00117 #define ERR_UNKNAUDIO 22
00118 #define MSG_USR2 23
00119 #define WRN_INI2 24
00120 #define MSG_MAXERRNO 25
00121
00122
00123
00124 /* Initialisierungskonstanten *.INI, werden sinngemaess auch bei der
00125     Registry und XML-Initialisierung verwendet.
00126     Bei Erweiterungen Variableninitialisierung szTCSErrorMsg und TCSErrorLev
00127     in TCSdWINc.c fuer Registry und XML-Initialisierung nicht vergessen und
00128     alle Parser (*.ini bei INITT1(), Registry bei StoreIni() und
00129     *.xml bei sax_callback() beruecksichtigen! */
00130
00131 #define TCS_INISECT0 "Graph2D" // Root-Section, derzeit nur bei XML verwendet
00132
00133 #define TCS_INISECT1 _T("Names")
00134  #define TCS_INIVAR_WINNAM _T("G2dGraphic")
00135     #define TCS_WINDOW_NAME _T("Graphics")
00136  #define TCS_INIVAR_STATNAM _T("G2dStatus")
00137     #define TCS_STATWINDOW_NAME _T("System Messages")
00138  #define TCS_INIVAR_HDCNAM _T("G2dHardcopy")
00139     #if (JOURNALTYP ==1)
00140        #define TCS_HDCFILE_NAME _T("HDC%03i.WMF")
00141     #elif (JOURNALTYP ==2)
00142        #define TCS_HDCFILE_NAME _T("HDC%03i.EMF")
00143     #elif (JOURNALTYP ==3)
00144        #define TCS_HDCFILE_NAME _T("HDC%03i.HDC")
00145     #else
00146        #define TCS_HDCFILE_NAME _T("HDC%03i.UNKNOWN")
00147     #endif
00148  #define TCS_INIVAR_MAINWINNAM _T("G2dMainWindow")
00149     #define TCS_MAINWINDOW_NAME _T("%:")
00150
00151 #define TCS_INISECT2 _T("Layout")
00152  #define TCS_INIVAR_COPMEN _T("G2dSysMenuCopy")
```

```
00153     #define TCS_INIDEF_COPMEN _T("Copy")
00154  #define TCS_INIVAR_FONT _T("G2dGraphicFont")
00155     #define TCS_INIDEF_FONT _T("Arial Terminal")
00156  #define TCS_INIVAR_SYSFONT _T("G2dSystemFont")
00157     #define TCS_INIDEF_SYSFONT _T("Arial Terminal")
00158  #define TCS_INIVAR_ICONNAM _T("G2dIcon")
00159     #define TCS_ICONFILE_NAME _T("")
00160  #define TCS_INIVAR_WINPOSX _T("G2dGraphicPosX")
00161     #define TCS_INIDEF_WINPOSX 0
00162  #define TCS_INIVAR_WINPOSY _T("G2dGraphicPosY")
00163     #define TCS_INIDEF_WINPOSY 0
00164  #define TCS_INIVAR_WINSIZX _T("G2dGraphicSizeX")
00165     #define TCS_INIDEF_WINSIZX 100
00166  #define TCS_INIVAR_WINSIZY _T("G2dGraphicSizeY")
00167     #define TCS_INIDEF_WINSIZY 100
00168  #define TCS_INIVAR_STATPOSX _T("G2dStatusPosX")
00169     #define TCS_INIDEF_STATPOSX 0
00170  #define TCS_INIVAR_STATPOSY _T("G2dStatusPosY")
00171     #define TCS_INIDEF_STATPOSY 0
00172  #define TCS_INIVAR_STATSIZX _T("G2dStatusSizeX")
00173     #define TCS_INIDEF_STATSIZX 100
00174  #define TCS_INIVAR_STATSIZY _T("G2dStatusSizeY")
00175     #define TCS_INIDEF_STATSIZY 100
00176  #define TCS_INIVAR_LINCOL _T("G2dLinCol")
00177     #define TCS_INIDEF_LINCOL 1
00178  #define TCS_INIVAR_TXTCOL _T("G2dTxtCol")
00179     #define TCS_INIDEF_TXTCOL 1
00180  #define TCS_INIVAR_BCKCOL _T("G2dBckCol")
00181     #define TCS_INIDEF_BCKCOL 0
00182
00183 #define TCS_INISECT3 _T("Messages")
00184  #define TCS_INIVAR_HDCOPN _T("G2dHdcOpen")
00185     #define TCS_INIDEF_HDCOPN _T("GRAPH2D HARDCOPY: Error during OPEN.")
00186     #define TCS_INIVAR_HDCOPNL _T("G2dHdcOpenL")
00187     #define TCS_INIDEF_HDCOPNL 5
00188  #define TCS_INIVAR_HDCWRT _T("G2dHdcWrite")
00189     #define TCS_INIDEF_HDCWRT _T("GRAPH2D HARDCOPY: Error during WRITE.")
00190     #define TCS_INIVAR_HDCWRTL _T("G2dHdcWriteL")
00191     #define TCS_INIDEF_HDCWRTL 5
00192  #define TCS_INIVAR_HDCINT _T("G2dHdcIntern")
00193     #define TCS_INIDEF_HDCINT _T("GRAPH2D HARDCOPY: Internal Error.")
00194     #define TCS_INIVAR_HDCINTL _T("G2dHdcInternL")
00195     #define TCS_INIDEF_HDCINTL 5
00196  #define TCS_INIVAR_USR _T("G2dUser")
00197     #define TCS_INIDEF_USR _T("%s")
00198     #define TCS_INIVAR_USRL _T("G2dUserL")
00199     #define TCS_INIDEF_USRL 5
00200  #define TCS_INIVAR_HDCACT _T("G2dHdcActive")
00201     #define TCS_INIDEF_HDCACT _T("Hardcopy in progress: File %s created.")
00202     #define TCS_INIVAR_HDCACTL _T("G2dHdcActiveL")
00203     #define TCS_INIDEF_HDCACTL 1
00204  #define TCS_INIVAR_USRWRN _T("G2dPressAny")
00205     #define TCS_INIDEF_USRWRN _T("Press any key to continue.")
00206     #define TCS_INIVAR_USRWRNL _T("G2dPressAnyL")
00207     #define TCS_INIDEF_USRWRNL 5
00208  #define TCS_INIVAR_EXIT _T("G2dExit")
00209     #define TCS_INIDEF_EXIT _T("Press any key to exit program.")
00210     #define TCS_INIVAR_EXITL _T("G2dExitL")
00211     #define TCS_INIDEF_EXITL 10
00212  #define TCS_INIVAR_COPMEM _T("G2dNoMemory")
00213     #define TCS_INIDEF_COPMEM _T("GRAPH2D Clipboard Manager: Out of Memory.")
00214     #define TCS_INIVAR_COPMEML _T("G2dNoMemoryL")
00215     #define TCS_INIDEF_COPMEML 1
00216  #define TCS_INIVAR_COPLCK _T("G2dClipLock")
00217     #define TCS_INIDEF_COPLCK _T("GRAPH2D Clipboard Manager: ClipBoard locked.")
00218     #define TCS_INIVAR_COPLCKL _T("G2dClipLockL")
00219     #define TCS_INIDEF_COPLCKL 1
00220  #define TCS_INIVAR_JOUCREATE _T("G2dJouCreate")
00221     #define TCS_INIDEF_JOUCREATE _T("GRAPH2D Error Creating Journal. Error-No: %s.")
00222     #define TCS_INIVAR_JOUCREATEL _T("G2dJouCreateL")
00223     #define TCS_INIDEF_JOUCREATEL 5
00224  #define TCS_INIVAR_JOUENTRY _T("G2dJouEntry")
00225     #define TCS_INIDEF_JOUENTRY _T("GRAPH2D Error Creating Journal Entry.")
00226     #define TCS_INIVAR_JOUENTRYL _T("G2dJouEntryL")
00227     #define TCS_INIDEF_JOUENTRYL 5
00228  #define TCS_INIVAR_JOUADD _T("G2dJouAdd")
00229     #define TCS_INIDEF_JOUADD _T("GRAPH2D Error Appending Journal Entry.")
00230     #define TCS_INIVAR_JOUADDL _T("G2dJouAddL")
00231     #define TCS_INIDEF_JOUADDL 5
00232  #define TCS_INIVAR_JOUCLR _T("G2dJouClr")
00233     #define TCS_INIDEF_JOUCLR _T("GRAPH2D Error Clearing Journal Entry.")
00234     #define TCS_INIVAR_JOUCLRL _T("G2dJouClrL")
00235     #define TCS_INIDEF_JOUCLRL 5
00236  #define TCS_INIVAR_JOUUNKWN _T("G2dJouEntryUnknwn")
00237     #define TCS_INIDEF_JOUUNKWN _T("GRAPH2D Unknown Journal Entry.")
00238     #define TCS_INIVAR_JOUUNKWNL _T("G2dJouEntryUnknwnL")
00239     #define TCS_INIDEF_JOUUNKWNL 1
```

```
00240  #define TCS_INIVAR_XMLPARSER _T("G2dXMLerror")
00241     #define TCS_INIDEF_XMLPARSER _T("GRAPH2D Error parsing XML-File: %s")
00242     #define TCS_INIVAR_XMLPARSERL _T("G2dXMLerrorL")
00243     #define TCS_INIDEF_XMLPARSERL 8
00244  #define TCS_INIVAR_XMLOPEN _T("G2dXMLopen")
00245     #define TCS_INIDEF_XMLOPEN _T("GRAPH2D Error opening %s")
00246     #define TCS_INIVAR_XMLOPENL _T("G2dXMLerrorL")
00247     #define TCS_INIDEF_XMLOPENL 8
00248  #define TCS_INIVAR_USR2 _T("G2dUser2")
00249     #define TCS_INIDEF_USR2 _T("%s")
00250     #define TCS_INIVAR_USR2L _T("G2dUser2L")
00251     #define TCS_INIDEF_USR2L 5
00252  #define TCS_INIVAR_INI2 _T("G2d2xInitt")
00253     #define TCS_INIDEF_INI2 _T("%s")
00254     #define TCS_INIVAR_INI2L _T("G2d2xInittL")
00255     #define TCS_INIDEF_INI2L 5
00256
00257
00258
00259 /* ------------------ Kompatibilität 16/32bit ----------------------------- */
00260
00261 #if !defined(__WIN32__) && !defined(_WIN32)
00262
00263  typedef char TCHAR, *PTCHAR;
00264  #define LPTSTR LPSTR
00265
00266  #define EXPORT16 __export /* __export bei virtuellem Adressraum unnötig */
00267  #define SM_CXMAXIMIZED SM_CXFULLSCREEN  /* notduerftiger Ersatz für ... */
00268  #define SM_CYMAXIMIZED SM_CYFULLSCREEN  /* ...Win32 Funktion */
00269  #define GetCommandLine() "WinApp"       /* dito */
00270
00271 #else
00272  #define EXPORT16
00273 #endif
00274
00275
00276
00277 /* ------------ Compilerspezifische Definitionen -------------------------- */
00278
00279 // _________________________________ Open-Watcom ___________________________________
00280 #if defined __WATCOMC__
00281  #ifdef _UNICODE
00282   #error "Watcom Ftn77 basiert nicht auf UNICODE !!!"
00283  #endif
00284
00285  #if !defined(__WIN32__) && !defined(_WIN32)
00286   #define TCSLEV3SYS 3 // TCSLEV(3) = 3 fuer Watcom/16 bit Windows
00287  #else
00288   #define TCSLEV3SYS 4 // TCSLEV(3) = 4 fuer Watcom/32 bit Windows
00289  #endif
00290
00291  /* Deklaration Parameteruebergabe Fortran <-> C */
00292  typedef long int LOGICAL;
00293  typedef long int FTNINT;
00294  typedef float FTNREAL;
00295  typedef double FTNDOUBLE;
00296  typedef struct {float real, imag;} FTNCOMPLEX;
00297  typedef char FTNCHAR;
00298  typedef unsigned FTNCHARLEN;
00299  typedef struct { FTNCHAR * addr; FTNCHARLEN len; } FTNSTRDESC;
00300  typedef FTNSTRDESC FTNSTRPAR;
00301  #define FTNSTRPAR_TAIL(ftns)
00302  #define FTNSTRPARA(ftns) ftns->addr
00303  #define FTNSTRPARL(ftns) ftns->len
00304  #define CALLFTNSTRA(ftns) & ftns
00305  #define CALLFTNSTRL(ftns)
00306  #define FWRDFTNSTRA(ftns) ftns
00307  #define FWRDFTNSTRL(ftns)
00308
00309  #pragma aux TKTRNX "^"; /* Fortran Naming Convention */
00310  #pragma aux tcslev3 "^";
00311  #pragma aux initt1 "^";
00312  #pragma aux finitt "^";
00313  #pragma aux GraphicError "^";
00314  #pragma aux winlbl "^";
00315  #pragma aux erase "^";
00316  #pragma aux swind1 "^";
00317  #pragma aux movabs "^";
00318  #pragma aux drwabs "^";
00319  #pragma aux dshabs "^";
00320  #pragma aux pntabs "^";
00321  #pragma aux bckcol "^";
00322  #pragma aux lincol "^";
00323  #pragma aux txtcol "^";
00324  #pragma aux DefaultColour "^"
00325  #pragma aux outgtext "^";
00326  #pragma aux italic "^";
```

```
00327  #pragma aux italir "^";
00328  #pragma aux dblsiz "^";
00329  #pragma aux nrmsiz "^";
00330  #pragma aux bell "^";
00331  #pragma aux outtext "^";
00332  #pragma aux tinput "^";
00333  #pragma aux dcursr "^";
00334  #pragma aux csize "^";
00335  #pragma aux hdcopy "^";
00336  #pragma aux lib_movc3 "^";
00337
00338 /* Deklarationen von durch C aufgerufenen FTN77-Unterprogrammen */
00339  #pragma aux igetarg "^"   // nur WATCOM: F77-Library
00340  FTNINT igetarg (FTNINT *iNo, FTNSTRDESC *Par);
00341
00342  #pragma aux initt2 "^"   // nur WATCOM: F77-Library
00343  void INITT2 (void);
00344
00345  #pragma aux SUBSTITUTE "^"        // aus STRINGS.FOR
00346  void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *n
00347                                   FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
00348                                   FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(n));
00349
00350
00351 // _____________________________ GNU-CC ____________________________________
00352 #elif defined __GNUC__
00353  #ifdef _UNICODE
00354   #error "GNU f77 basiert nicht auf UNICODE !!!"
00355  #endif
00356
00357  #if defined (WINVER)
00358   #if defined (_WIN64)
00359    #define TCSLEV3SYS 7 // TCSLEV(3) = 7 fuer GCC / 64bit Windows
00360   #else
00361    #define TCSLEV3SYS 5 // TCSLEV(3) = 5 fuer GCC / Windows
00362   #endif // defined
00363  #else
00364   #define TCSLEV3SYS 0 // TCSLEV(3) = 0 fuer unknown
00365  #endif
00366
00367 /* Deklaration Parameteruebergabe Fortran <-> C */
00368
00369  //  #include <g2c.h> // nur fuer g77, fuer gfortran s.u.
00370  typedef long int logical; // 3 (mit ftnlen) plattformabhaengige Definitionen
00371  typedef long int integer; // Ersatz fuer g2c.h: evtl. ueberpruefen
00372
00373  typedef logical LOGICAL;
00374  typedef integer FTNINT;
00375  typedef float FTNREAL;
00376  typedef double FTNDOUBLE;
00377  typedef struct {float real, imag;} FTNCOMPLEX;
00378
00379  typedef TCHAR FTNCHAR;
00380  #if __GNUC__ > 7  // GCC V7: size_t definiert, bei win64 8 Byte, nicht 4!
00381   typedef size_t ftnlen;  // Ersatz fuer g2c.h
00382   typedef size_t FTNCHARLEN;
00383  #else
00384   typedef long int ftnlen;  // Ersatz fuer g2c.h
00385   typedef ftnlen FTNCHARLEN; // size_t erst ab GCC > 7 definiert
00386  #endif
00387
00388  typedef struct { FTNCHAR * addr; FTNCHARLEN len; } FTNSTRDESC;
00389  typedef FTNCHAR FTNSTRPAR;
00390  #define FTNSTRPAR_TAIL(ftns) , FTNCHARLEN ftns##_len
00391  #define FTNSTRPARA(ftns) ftns
00392  #define FTNSTRPARL(ftns) ftns##_len
00393  #define CALLFTNSTRA(ftns) ftns.addr
00394  #define CALLFTNSTRL(ftns) , ftns.len
00395  #define FWRDFTNSTRA(ftns) ftns
00396  #define FWRDFTNSTRL(ftns) , ftns##_len
00397
00398  #define TKTRNX tktrnx_ /* Fortran Naming Convention */
00399  #define tcslev3 tcslev3_
00400  #define initt1 initt1_
00401  #define finitt finitt_
00402  #define GraphicError graphicerror_
00403  #define winlbl winlbl_
00404  #define erase erase_
00405  #define swind1 swind1_
00406  #define movabs movabs_
00407  #define drwabs drwabs_
00408  #define dshabs dshabs_
00409  #define pntabs pntabs_
00410  #define bckcol bckcol_
00411  #define lincol lincol_
00412  #define txtcol txtcol_
00413  #define DefaultColour defaultcolour_
```

```
00414  #define outgtext outgtext_
00415  #define italic italic_
00416  #define italir italir_
00417  #define dblsiz dblsiz_
00418  #define nrmsiz nrmsiz_
00419  #define bell bell_
00420  #define outtext outtext_
00421  #define tinput tinput_
00422  #define dcursr dcursr_
00423  #define csize csize_
00424  #define hdcopy hdcopy_
00425  #define lib_movc3 lib_movc3_
00426
00427 /* Deklarationen von durch C aufgerufenen FTN77-Unterprogrammen */
00428  #define getarg getarg_       // aus GNU F77-Library
00429  FTNINT GETARG (FTNINT *iNo, FTNCHAR *line, FTNCHARLEN line_len);
00430
00431  #define initt2 initt2_
00432  void INITT2 (void);
00433
00434  #define SUBSTITUTE substitute_ // universeller Aufruf Watcom/GNU moeglich
00435  void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *new
00436                                          FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
00437                                          FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(new));
00438
00439 #endif
00440 // _____________Ende systemabhaengige Deklarationen_____________________
00441
00442
00443 /* Forward Deklarationen: Codiert in C und auch in C verwendet */
00444
00445 void bell (void); //  -> Forward Deklaration
00446 void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) );
00447 void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
00448                    FTNINT *iL  FTNSTRPAR_TAIL(ftn_string));
00449 // void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy);
00450 void tinput (FTNINT *ic);
00451 void finitt (); // ueberpruefen !!!
00452
```

6.40 TCSinitt.for File Reference

MS Windows Port: initialization.

Functions/Subroutines

- subroutine initt (iDummy)

  MS Windows specific subroutines.

6.40.1 Detailed Description

MS Windows Port: initialization.

Version

1.4

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Definition in file TCSinitt.for.

6.40.2 Function/Subroutine Documentation

6.40.2.1 initt()

```
subroutine initt (
              iDummy )
```

MS Windows specific subroutines.

Note

Initialization of the DLL: The subroutine INITT must not be placed inside the DLL, but must be linked statically to the user program. Otherwise the instance of the DLL and not the instance of the main programm will be optained.

Attention with 64bit operating systems: The passing of pointers is done by Fortran77 integer variables. With Win64 the pointer length is 8 bytes, corresponding to 2 StorageUnits (integer∗4). In consequence the parameter nPtrStorageUnits must be set >= 2.

This routine can also be used for initializing Windows NT console programs. Init Hardware & Software

initt2() -> Reset Software
Definition at line 80 of file TCSinitt.for.

6.41 TCSinitt.for

```
00001 C> \file       TCSinitt.for
00002 C> \version    1.4
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   MS Windows Port: Initialisierung
00007 C> \~english
00008 C> \brief   MS Windows Port: initialization
00009 C> \~
00010 C
00011 C
00012 C> \~german
00013 C> MS Windows-spezifische TCS-Routinen
00014 C> \note
00015 C> Initialisierung der DLL: Das Unterprogramm INITT darf sich nicht
00016 C> in der DLL befinden, sondern muss statisch zu dem Anwenderprogramm
00017 C> gelinkt werden, da sonst die Instanz der DLL und nicht die des
00018 C> Anwenderprogramms ermittelt wird.
00019 C>
00020 C> \note
00021 C> Achtung bei 64bit Betriebssystemen: Die Übergabe von Pointern erfolgt
00022 C> durch Fortran77 Integer-Variablen. Bei Win64 beträgt die Pointerlänge
00023 C> 8 Bytes entsprechend 2 StorageUnits (integer*4). Entsprechend muss der
00024 C> Parameter nPtrStorageUnits angepasst werden.
00025 C>
00026 C> \note
00027 C> Die Routine kann auch zur Initialisierung von Windows NT
00028 C> Konsolenprogrammen verwendet werden.
00029 C>
00030 C
00031 C
00032 C> \~english
00033 C> MS Windows specific subroutines
00034 C> \note
00035 C> Initialization of the DLL: The subroutine INITT must not be
00036 C> placed inside the DLL, but must be linked statically to the user
00037 C> program. Otherwise the instance of the DLL and not the instance
00038 C> of the main programm will be optained.
00039 C>
00040 C> \note
00041 C> Attention with 64bit operating systems: The passing of pointers is done
00042 C> by Fortran77 integer variables. With Win64 the pointer length is
00043 C> 8 bytes, corresponding to 2 StorageUnits (integer*4). In consequence the
00044 C> parameter nPtrStorageUnits must be set >= 2.
00045 C>
00046 C> \note
00047 C> This routine can also be used for initializing Windows NT console programs.
00048 C>\~
00049 C>
00050 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00051 C
00052 C  Version 1.4, 30.4.2021, K. Friedewald
00053 C     Anpassung an Windows64: Pointerlänge 8 Byte > int*4 bei win32
00054 C
00055 C  Version 1.3, 17.8.2020, K. Friedewald
00056 C     Reaktivierung KHOMEY fuer HOME()
00057 C
```

```
00058 C  Version 1.2, 29.9.2004, K. Friedewald
00059 C     Zusammenfassung der DLL-Initialisierung mit der LIB-Version. INITT
00060 C     wird zusammen mit GetMainInstance.c in der LIB gehalten, die rest-
00061 C     lichen Programme können sich in einer DLL befinden.
00062 C
00063 C  Version 1.1, 22.6.2004, K. Friedewald
00064 C     Falls initt1 von dem Hauptprogramm ohne ein aktives Fenster aufgerufen
00065 C     wird treten schwer reproduzierbare Fehler auf, da die Rueckmeldungen
00066 C     auf die anfänglichen Windowsabfragen nicht eindeutig zugeordnet werden.
00067 C
00068 C     Abhilfe: Es wird jetzt bei Bedarf vor der Initialisierung ein eigenes
00069 C     Hauptprogrammfenster erstellt.
00070 C
00071 C  Version 1.0, 19.3.2003, K. Friedewald
00072 C
00073
00074
00075 C
00076 C>  Init Hardware & Software
00077 C
00078
00079
00080       subroutine initt (iDummy)
00081 C
00082       parameter(nptrstorageunits=2) ! max.Laenge Pointer in StorageUnits (2=64bit)
00083       integer iInstance(nPtrStorageUnits), iWindow(nPtrStorageUnits)
00084       call getmaininstandwin (iinstance, iwindow)
00085       call initt1 (iinstance, iwindow)
00086       call savemaininstandwin (iinstance, iwindow)
00087
00088 C> initt2() -> Reset Software
00089       entry initt2
00090       call lintrn
00091       call swindo (0,1023,0,780)
00092       call vwindo (0.,1023.,0.,780.)
00093       call rrotat (0.)
00094       call rscale (1.)
00095       call setmrg (0,1023)
00096       call nrmsiz
00097       call italir
00098       call home
00099       return
00100       end
```

6.42 TKTRNX.fd File Reference

MS Windows Port: TCS Common Block TKTRNX.

6.42.1 Detailed Description

MS Windows Port: TCS Common Block TKTRNX.

Version

1.4

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

header belonging to TKTRNX.h

Note

Because the following definition not beeing part of a module, the DOXYGEN parser is not able to handle the combination of COMMON and INTEGER declarations. Workaraound: \cond ... \endcond.

Definition in file TKTRNX.fd.

6.43 TKTRNX.fd

```
00001 C> \file       TKTRNX.fd
00002 C> \brief      MS Windows Port: TCS Common Block TKTRNX
00003 C> \version    1.4
00004 C> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C> Header passend zu TKTRNX.h
00009 C> \note
00010 C> Da die folgende Definition kein Bestandteil eines Moduls
00011 C> ist, versagt der DOXYGEN-Parser bei der Kombination von
00012 C> COMMON und INTEGER. Workaraound: \\cond ... \\endcond.
00013 C> \~english
00014 C> header belonging to TKTRNX.h
00015 C> \note
00016 C> Because the following definition not beeing part of a module, the
00017 C> DOXYGEN parser is not able to handle the combination of COMMON
00018 C> and INTEGER declarations.  Workaraound: \\cond ... \\endcond.
00019 C> \~
00020 C> \cond
00021 C Common Block TKTRNX, Version 1.3 für WINDOWS
00022 C
00023       COMMON /tktrnx/
00024 C           kbaudr,kerror,kgrafl,
00025      & khomey,
00026 C           kkmode,
00027      & khorsz,kversz,
00028      & kitalc,ksizef,
00029      & klmrgn,krmrgn,
00030 C           kScrX,kScrY,
00031 C           ktblsz,khorzt(10),kvertt(10),
00032      & kbeamx,kbeamy,
00033 C           kmovef,kpchar(4),kdasht,
00034      & kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
00035 C      trealx,trealy,timagx,timagy,
00036      & trcosf,trsinf,trscal
00037      & ,xfac,yfac,xlog,ylog,kstcol,
00038      & ilincol, ibckcol, itxtcol, imouse
00039
00040       SAVE /tktrnx/
00041       integer iTktrnxL
00042       parameter(itktrnxl=29) ! +11)
00043
00044 C Neue Variablen:
00045 C     kHorSz,kVerSz: Buchstabengröße im (1024/780) Koordinatensystem
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 Parameter iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
00053 C     Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
00054 C
00055 C> \endcond
```

6.44 TKTRNX.h File Reference

MS Windows Port: TCS Common Block TKTRNX.

Classes

- struct TKTRNXcommonBlock

Variables

- struct TKTRNXcommonBlock TKTRNX

6.44.1 Detailed Description

MS Windows Port: TCS Common Block TKTRNX.

Version

1.4

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

C header belonging to TKTRNX.fd

Note

Adaption to the compiler specific name convention is done in TCSdSDLc.h

Definition in file TKTRNX.h.

6.44.2 Variable Documentation

6.44.2.1 TKTRNX

```
struct TKTRNXcommonBlock TKTRNX
```

6.45 TKTRNX.h

```
00001 /** ****************************************************************************
00002 \file       TKTRNX.h
00003 \brief      MS Windows Port: TCS Common Block TKTRNX
00004 \version    1.4
00005 \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          C Header passend zu TKTRNX.fd
00009 \~english
00010          C header belonging to TKTRNX.fd
00011 \~
00012
00013 \~german
00014 \note
00015    Anpassung an die compilerabhaengige Namenskonvention erfolgt in TCSdSDLc.h
00016 \~english
00017 \note
00018    Adaption to the compiler specific name convention is done in TCSdSDLc.h
00019 \~
00020
00021 ******************************************************************************* */
00022
00023
00024 extern struct TKTRNXcommonBlock {
00025 FTNINT
00026 //           kbaudr,kerror,kgrafl,
00027      khomey,
00028 //           kkmode,
00029      khorsz,kversz,
00030      kitalc,ksizef,
00031      klmrgn,krmrgn,
00032 //           kScrX,kScrY,
00033 //           ktblsz,khorzt(10),kvertt(10),
00034      kBeamX,kBeamY,
00035 //           kmovef,kpchar(4),kdasht,
00036      kminsx,kminsy,kmaxsx,kmaxsy;
00037
00038 FTNREAL
00039      tminvx,tminvy,tmaxvx,tmaxvy,
00040 //      trealx,trealy,timagx,timagy,
00041      trcosf,trsinf,trscal
00042      ,xfac,yfac,xlog,ylog;
00043 FTNINT
00044     kStCol,
00045     iLinCol, iBckCol, iTxtCol, iMouse;
00046 } FAR TKTRNX;
00047
```

Index