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 iy1,
            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

```
C> \file       AG2.for
C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
C> \version    (2025,347, x)
C> \author     (C) 2025 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C>  Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
C> \note
C>     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
C>     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
C>
C> \~english
C> Layer 2: scientific 2-D graphic subroutines
C> \note
C>     The control character for exponent (originally -1) is now SOH=char(1)
C>     and for index (originally -2) STX=char(2).
C>
C> \~
C> \note \verbatim
C>   Package:
C>    - AG2.for:          chart plotting routines
C>    - AG2Holerith.for: deprecated routines
C>    - AG2USR.for:       default userroutines
C>    - G2dAG2.fd:        commonblock
C> \endverbatim
C
C
C  Tektronix Advanced Graphics 2 - Version 2.x
C
C
C     Neuer Code in Fortran 77. Die Verwendung der im Manual dokumentierten
C     Unterprogramme bleibt unveraendert, die direkte Manipulation von
C     Variablen des zugrundeliegenden Commonblockes ist jedoch nicht mehr
C     empfehlenswert. IBASEX (iPar) und IBASEY(iPar) mit ipar <>0,
C     IBASEC, COMGET und COMSET sollten in neuen Programmen nicht verwendet
C     werden.
C
C     Die Zwischenspeicherung der Statusvariablen ueber
```

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

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

      call tcslev (ilevel) ! level(3)= System aus TCS
      ilevel(1)=2025       ! Aenderungsjahr
      ilevel(2)=  70       ! Aenderungstag
      return
      end



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

      cline= ipar
      return
      end



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

      csymbl= ipar
      return
      end



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

      csteps= ipar
      return
      end



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

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



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

      ag2infin= cinfin
      return
      end



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

      cnpts= ipar
      return
      end



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

      cstepl= ipar
      return
      end



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

      csizes= par
      return
      end



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

      csizel= par
      return
      end



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

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



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

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



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

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



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

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



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

      cxyloc(1)= ipar
      return
      end



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

      cxyloc(2)= ipar
      return
      end



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

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



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

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



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

      cxylab(1)= ipar
      return
      end



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

      cxylab(2)= ipar
      return
      end



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

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



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

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



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

      cxytics(1)= abs(ipar)
      return
      end



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

      cxytics(2)= abs(ipar)
      return
      end



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

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



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

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



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

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



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

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



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

      cxymtcs(1)= abs(ipar)
      return
      end



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

      cxymtcs(2)= abs(ipar)
      return
      end



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

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



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

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



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

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



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

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



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

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



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

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



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

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

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



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

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



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

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



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

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



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

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



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

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



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

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



      subroutine setwin
      implicit none
      include 'G2dAG2.fd'

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



      subroutine dinitx
      implicit none
      include 'G2dAG2.fd'

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



      subroutine dinity
      implicit none
      include 'G2dAG2.fd'

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



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

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

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



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

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



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

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

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

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

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

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

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



C
C  Datenanalyse
C

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

      external SPREAD ! External wg. Namenskonflikt FTN90-Intrinsic

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

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



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

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



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

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



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

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



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


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

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



C
C  Ticmarkoptimierung
C

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

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

```

```

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

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

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

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

      aminor= amin
      amaxor= amax

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

310   continue ! repeat...
       if (labtyp .eq. 2) dataint= roundu(dataint,1.) ! logarithmische Achsen
       lsig= roundd(log10(dataint),1.) ! Anzahl signifikanter Nachkommastellen
       sigfac=10.**(lsig)
       if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
        if(labtyp .ne. 2) then ! nicht bei log. Achsen
         if ((dataint/sigfac) .le. 1.) then
          dataint= 1. * sigfac
          mintic= 10
         else if ((dataint/sigfac) .le. 2.) then
          dataint= 2. * sigfac
          mintic= 2
         else if ((dataint/sigfac) .le. 2.5) then
          dataint= 2.5 * sigfac
          mintic= 5
          lsig=lsig-1
         else if ((dataint/sigfac) .le. 5.) then
          dataint= 5. * sigfac
          mintic=  5
         else if ((dataint/sigfac) .le. 10.) then
          dataint= 10. * sigfac
          mintic= 10
          lsig=lsig+1
         else
          dataint= cinfin
          mintic= 0
         end if
        end if ! log. Achse
       else ! .not. neat
        lsig=lsig-2
       end if
       if (lsig .ge. 0) lsig=lsig+1
      if (cxyneat(ixy) .or. (labtyp .eq. 2) ) then ! ... until
       amin= roundd(amin+.01*sigfac,dataint) !  runde auf TicIntervall
       amax= roundu(amax-.01*sigfac,dataint) ! .01*sigfac= Genauigkeit Plot
       ntics= int(abs(amax-amin)/dataint+.0001)
       if(cxytics(ixy) .ne. 0) then ! until: ntics nicht vorbesetzt oder = vorbesetzt
        if(abs(cxytics(ixy)) .lt. ntics) then
         dataint= dataint * 1.1
         amin=aminor
         amax=amaxor
         goto 310 ! noch eine Iterationsschleife
        else if (abs(cxytics(ixy)) .gt. ntics) then
         ntics= abs(cxytics(ixy))
         amax= amin + real(ntics) * dataint
        end if ! abs(cxytics(ixy)) .eq. ntics: no action
       end if
      end if
      cxytics(ixy)= ntics

      if ((cxymtcs(ixy) .eq. 0) .and. (cxyden(ixy) .ge. 6)) then ! unbesetzt oder wenig TICS
       mtcs= mintic ! Bestimmung Minor TicMarcs
       if((mtcs .eq. 10) .or. (labtyp .eq. 2)) then
        if(cxyden(ixy) .lt. 9) mtcs=5
        if(cxyden(ixy) .lt. 7) mtcs=2
        if(labtyp .eq. 2) then ! log. Achsen
         idataint= nint(dataint)
         if (idataint .ne. 1) then ! mehrere Achsenintervalle
          i= 1
320       continue ! repeat...
           mtcs= idataint/i
          if ((mtcs*i .ne. idataint) .and. (i .lt. (idataint-1))) then ! ...until
           i= i+1
           goto 320
          else if (mtcs .gt. 10 ) then
           mtcs= 0   ! Failure
          end if
         else ! einzelne logarithmische Dekade
          if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 100* ntics) mtcs=-1 ! logarithm. Tics
          if ((cxysmax(ixy) - cxysmin(ixy)) .ge. 20* linhgt(1)) mtcs=-2 ! Label
         end if
        end if
       end if
       cxymtcs(ixy)= mtcs
      end if

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



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

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

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

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



C
C  Kalenderroutinen
C



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

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

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



       subroutine calcon (amin,amax,labtyp,ubgc)
       implicit none
       real amin, amax
       integer labtyp
       logical ubgc
       integer iubg1, iubg2, iday1, iadj, id, month1,month2 , imin,imax
       real dimin, dimax
       integer iweek1
       real fnoday
```

```
      integer iy1,iy2, iy3,iy4, idays
      save iweek1, fnoday
      save iy1,iy2, iy3, iy4, idays

      real ROUNDD, ROUNDU

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

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

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


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

      ijulyrout= igregyrin
      imon= igregmonin
100   if (imon .lt. 1) then ! while iMon .not. in [1..12]
       imon= imon + 12
       ijulyrout= ijulyrout-1
       goto 100
      else if (imon .gt. 12) then
       imon= imon -12
```

```
         ijulyrout= ijulyrout+1
         goto 100
        end if
        ijuldayout= igregdayin + idattab(imon)
        if (imon .gt.2) ijuldayout= ijuldayout + leap(ijulyrout)
        return

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

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

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



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



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

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



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

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



C
C  Zeichenroutinen
C
```

```
      subroutine frame
      implicit none
      include 'G2dAG2.fd'

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



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

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



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

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

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

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

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

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

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

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



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

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



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

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

```
C  Balkendiagramme

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

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

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

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



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

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

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

```fortran
      call drwabs (imaxx,imaxy)
      call drwabs (iminx,imaxy)
      call drwabs (iminx,iminy)

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

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

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

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

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

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



C  Zeichnen von Symbolen

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

      if (isym .ge. 0) then
       call symout (isym, csizes)
      else
       call users (x,y,isym)
      end if
      call movea (x,y)
      return
      end
      subroutine symout (isym,fac)
      implicit none
      integer isym
      real fac
      integer ix,iy, ihorz,ivert

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



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

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



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

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

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



C Netz und Ticmarks

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

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

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

```fortran
          call movabs (nint(xyext), cxybeg(1))
          call drwabs (nint(xyext), cxyend(1))
120      continue
        end if ! Labtyp=6: Monate
       end if ! Ende Zeichnen Ticmarks
      end if ! Ende Zeichnen der Achse
      return
      end



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

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

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

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



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

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

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

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



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

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

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



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

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



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

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



C Label

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

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



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

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

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

      return

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

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



      subroutine fformc (fnum,iwidth,idec, outstr)
      implicit none
      real fnum
      integer iwidth,idec
      character outstr *(*)
      integer nDgtM
      real fa
      include 'G2dAG2.fd'

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

      if ( (((fa .lt. 10./cinfin) .or. (fa .gt. .1**idec)) ! Zahl mit Dezimalen darstellbar
     1                        .and.(fa .lt. 10.**ndgtm))   ! Zahl mit Mantisse darstellbar
     2      .or. ((iwidth.lt.idec+7))              ) then  ! oder Laenge zu kurz fuer E-Format
       call fonlyc (fnum,iwidth,idec, outstr)
      else
       call eformc (fnum,iwidth,idec, outstr)
      end if
      return
      end



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

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

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

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

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



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

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

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

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

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



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

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



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

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

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

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

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

```
          outstr(1:)= '1' // char(0) ! 1 wird nicht als 10**0 ausgegeben
         end if
        else ! outstr zu kurz
         outstr= '?X?'
        end if

        return
        end



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

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



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

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

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

```
      end



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

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



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

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

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

      return
      end



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

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

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

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

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

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

      return
      end



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

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

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

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

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

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

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



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

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

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

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



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

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

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

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

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

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

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

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

```
        cxystep(nbase)= cxystep(nbase) + 1
       end if

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



C
C  Tabellensuche und Rundungen
C

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

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

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



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

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

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

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



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

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

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



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

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

```
        goto 100
 110   continue ! endwhile

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



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

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



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

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



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

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



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

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



       integer function iother (ipar)
       implicit none
       integer ipar

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

```
      end if
      return
      end
```

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

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

C
C Stringfunktionen fuer Holerithvariablen
C

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

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



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

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



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

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



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

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



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



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



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



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



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

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



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

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



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

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



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

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



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

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



C
C  Direkte Manipulation des Commonblocks
C

      integer function ibasec (iPar)
      implicit none
      integer ipar

      ibasec= -1-ipar
      return
      end


      integer function ibasex (ipar)
      implicit none
      integer ipar

      ibasex= 1 + 2*ipar
      return
      end


      integer function ibasey (ipar)
      implicit none
      integer ipar

      ibasey= 2 + 2*ipar
      return
      end


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

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

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


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

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

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


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

      call erase
      call home

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

```
       call newlin
       write (unit= buf,fmt=621, err=200) (cxyetyp(i),i=1,2)
 621   format (1x,'21: cxyetyp(1)=',i14,', (2)=',i14)
       call toutstc (buf)
       call newlin
       write (unit= buf,fmt=622, err=200) (cxybeg(i),i=1,2)
 622   format (1x,'22:  cxybeg(1)=',i14,', (2)=',i14)
       call toutstc (buf)
       call newlin
       write (unit= buf,fmt=623, err=200) (cxyend(i),i=1,2)
 623   format (1x,'23:  cxyend(1)=',i14,', (2)=',i14)
       call toutstc (buf)
       call newlin
       write (unit= buf,fmt=624, err=200) (cxymbeg(i),i=1,2)
 624   format (1x,'24: cxymbeg(1)=',i14,', (2)=',i14)
       call toutstc (buf)
       call newlin
       write (unit= buf,fmt=625, err=200) (cxymend(i),i=1,2)
 625   format (1x,'25: cxymend(1)=',i14,', (2)=',i14)
       call toutstc (buf)
       call newlin
       write (unit= buf,fmt=626, err=200) (cxyamin(i),i=1,2)
 626   format (1x,'26: cxyamin(1)=',e14.7,', (2)=',e14.7)
       call toutstc (buf)
       call newlin
       write (unit= buf,fmt=627, err=200) (cxyamax(i),i=1,2)
 627   format (1x,'27: cxyamax(1)=',e14.7,', (2)=',e14.7)
       call toutstc (buf)

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

 200   continue
       return
       end
```

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

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


      subroutine uline (x,y,i)
      return
      end

```

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

```
C> \file    AG2umnmx.for
C> \brief   Graph2D: Dummy User Routine
C
C  Tektronix Advanced Graphics 2 - Version 2.0
C
C     User Subroutinen
C

      subroutine umnmx (array,amin,amax)
      return
      end

```

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

```
C> \file    AG2useset.for
C> \brief   Graph2D: Dummy User Routine
C
C  Tektronix Advanced Graphics 2 - Version 2.0
C
C     User Subroutinen
C

      subroutine useset (fnum,iwidth,nbase,labeli)
      implicit none
      real fnum
      integer iwidth, nbase
      integer labeli(1)
      integer i

      do 100 i=1, iwidth
       labeli(i)= 32 ! Blank
100   continue
      return
      end

```

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

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

      subroutine softek (isym)
      return
      end
```

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

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

***************************************************************************** */

```

```
#if defined(__WATCOMC__) && defined(__WINDOWS__)
 #define NULL 0           // nur win16: Ueberlagern #define NULL ( (void *) 0)
#endif                    // aus aus stddef.h, string.h...

#define WIN32_LEAN_AND_MEAN
#include <windows.h>

#include <tchar.h>
#include "TCSdWINc.h"   // Unterstuetzung 16/32bit Kompatibilitaet

#if defined(__WATCOMC__) && defined(__SW_BW)
 #include <wdefwin.h>   // Compilerswitch -bw: Watcom Default Window System
#endif

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

/** ***************************************************************************

\~german
\brief Initialisierung der FTN77 Hauptprogramme

  Unterprogramm zur Initialisierung von Windows. Erzeugt und zeigt(!) ein
  Fenster für das Hauptprogramm, falls noch keine Windows-Initialisierung
  anderweitig (z.B. durch den Compiler) vorgenommen wurde. Die Klasse wird
  entsprechend der Konstante WINMAIN_DEFWINCLASS benannt.

  Das Icon kann über ein Resourcefile als WinMainIcon definiert werden.

\param[in] hMainProgInst Instanz des Hauptprogrammes
\param[in,out] hMainProgWindow Fenster des Hauptprogrammes
\param[in] szWinName Fenstername des evtl. erzeugten Fensters
\~english

  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.

\param[in] hMainProgInst Main instance
\param[in,out] hMainProgWindow Main window
\param[in] szWinName Window name in case a main window does not exist
\~


*************************************************************************** */


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

{

TCHAR           szClassName [] = WINMAIN_DEFWINCLASS; /* Class Name */
static WNDCLASS wincl;       /* SAVE Data structure for the windowclass */
#if defined(__WIN32__) || defined(_WIN32)
 DWORD           ErrorCode;
 LPVOID          lpMsgBuf;
#endif


    if (*hMainProgWindow == NULL ) { // Hauptprogramm ohne (bekanntes) Fenster

     /* Create MainWindow */

     wincl.hInstance = *hMainProgInst;
     wincl.lpszClassName = szClassName;
     wincl.lpfnWndProc = DefWindowProc;      /* keine eigene Windowsroutine */
     wincl.style =  CS_DBLCLKS;              /* Catch double-clicks */

     wincl.hIcon = LoadIcon (*hMainProgInst, WINMAIN_ICON);
     wincl.hCursor = NULL;
     wincl.lpszMenuName = NULL;    // No menu
     wincl.cbClsExtra = 0;         // No extra bytes after the window class
     wincl.cbWndExtra = 0;         // structure or the window instance
     wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

     /* Register the window class. Fail: most probable UNICODE on win98 */
     if (!RegisterClass (&wincl)) {
      #if defined(__WIN32__) || defined(_WIN32)
       ErrorCode= GetLastError(); // win32-Funktion
//     if (ErrorCode == ERROR_CLASS_ALREADY_EXISTS) {
//      Hier bei Bedarf Fehlerbehandlung einführen
//     } else {
        FormatMessage(
         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
         NULL,
         ErrorCode,
         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
         (LPTSTR) &lpMsgBuf,
         0,
         NULL
        );
        MessageBox (NULL, lpMsgBuf,_T("Error in CreateMainWindow"), MB_ICONSTOP);
        LocalFree( lpMsgBuf ); // Free the buffer
//        } // Ende der Fehlerbehandlung
      #else // rudimentaere Fehlerbehandlung 16bit Windows
       MessageBox (NULL, _T("Window Class not registered"),
                            _T("Error in CreateMainWindow"), MB_ICONSTOP);
      #endif
      return;
     }

     /* The class is registered, let's create the program */
     *hMainProgWindow = CreateWindow (
       szClassName,                    // Classname
       szWinName,                      // Title Text
       WS_POPUPWINDOW | WS_DISABLED,   // disabled -> Prozessverwaisung verhindern
       CW_USEDEFAULT,                  // Windows decides the position
       CW_USEDEFAULT,                  // of the Window
       0,                              // The programs width
       0,                              // and height in pixels
       HWND_DESKTOP,                   // Parent: desktop
       NULL,                           // No menu
       *hMainProgInst,                 // Program Instance handler
       NULL                            // No Window Creation data
     );
     ShowWindow (*hMainProgWindow, SW_SHOW);
    } else {     // Mainwindow bereits vorhanden
     #if defined(__WATCOMC__) && defined(__SW_BW)
      _dwSetAppTitle (szWinName);      // Fenstername Watcom Default Window
     #endif
    }
}

```

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

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

```

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

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

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

      integer G2dAG2L        ! Benoetigt von SAVCOM, RESCOM
      parameter(g2dag2l=65)  ! integer, real und logical gleich lang!
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)
```

```
        else if (action.eq.8) then ! XACTION_GTEXT
          iprntlen= i1
          if (iprntlen.gt.tcs_messagelen) iprntlen= tcs_messagelen
          txtstring(1:1)= char(i2)
          if (iprntlen.eq.1) then
            txtstring= txtstring(1:1) // char(0)
            call toutstc (txtstring)
          else
            iactlen= 1
          end if
        else if (action.eq.9) then ! XACTION_ASCII
          if (iactlen.lt.iprntlen) then
            iactlen= iactlen+1
            txtstring(iactlen:iactlen)= char(i1)
          end if
          if (iactlen.lt.iprntlen) then
            iactlen= iactlen+1
            txtstring(iactlen:iactlen)= char(i2)
          end if
          if (iactlen.ge.iprntlen) then
            txtstring(iactlen+1:iactlen+1) = char(0)
            call toutstc (txtstring)
          end if
        else if (action.eq.10) then ! XACTION_BCKCOL
          call bckcol(i1)
        else if (action.eq.11) then ! XACTION_LINCOL
          call lincol (i1)
        else if (action.eq.12) then ! XACTION_TXTCOL
          call txtcol (i1)
        else if (action.eq.13) then ! XACTION_FONTATTR
          if (i1.eq.0) call italir()
          if (i1.eq.1) call italic()
          if (i2.eq.0) call nrmsiz()
          if (i2.eq.1) call dblsiz()
        else if (action.eq.14) then ! XACTION_NOOP
          continue
        else if (action.eq.15) then ! XACTION_CLIP
          if (i1.eq.0) then ! clipping not active
            kminsx= 0
            kminsy= 0
            kmaxsx= 1023 ! TEK_XMAX
            kmaxsy= 780 ! TEK_YMAX
            call swind1(kminsx,kminsy,kmaxsx,kmaxsy) ! Set bool ClippingNotActive
          end if
        else if (action.eq.16) then ! XACTION_CLIP1
          kminsx= i1
          kminsy= i2
          call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
        else if (action.eq.17) then ! XACTION_CLIP2
          kmaxsx= i1
          kmaxsy= i2
          call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
        else ! unknown
          continue
        end if
      if (ios.eq.0) goto 10 ! until EOF

      close (iunit)
      gethdc= .false.
      return
      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

```
/** ****************************************************************************
\file       GetMainInstance.c
\brief      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
\~german
         Ermittlung Instanz und Fenster der FTN77 Hauptprogramme
\~english
         Get Instance and Window of the FTN77 Main Program
\~

***************************************************************************** */


#if defined(__WATCOMC__) && defined(__WINDOWS__)
 #define NULL 0           // nur win16: Ueberlagern #define NULL ( (void *) 0)
#endif                    // aus aus stddef.h, string.h...



#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <tchar.h>



/*
-------------------------- Externe Bezüge ------------------------------
*/

#ifdef __WATCOMC__     // Bis 11.0c: WATCOM Fortran Default Window System 10.0
 #if (__WATCOMC__ == 1100) // Source OpenWatcom 0.8, bld\clib\defwin\c bzw. \h
  extern HWND _MainWindow; // winglob.c, wmain.c, winmain.c, win.h
  #define EXTERN_WINDOW _MainWindow
  #undef EXTERN_INSTANCE
 #elif (__WATCOMC__ >= 1200)                 // Open Watcom 1.0 bis 1.9:
  #if (!defined(__WIN32__) && !defined(_WIN32))         // 16bit-Windows
   #ifndef __SW_BW
    #error 16bit Windows requieres Default Window System, use the /bw switch
   #else
    extern HWND _MainWindow;     // Open Watcom Default Window System 1.0
    #define EXTERN_WINDOW _MainWindow
    #undef EXTERN_INSTANCE
   #endif
  #else              // 32bit-Windows: Default Window System deaktiviert
   #if defined (__SW_BW)
    #pragma message ("OpenWatcom >=1.0: Default Window System disabled!")
    #undefine __SW_BW
   #endif
   HWND _TCSMainWindow= NULL;
   #define EXTERN_WINDOW _TCSMainWindow
   #undef EXTERN_INSTANCE
  #endif
  #if (__WATCOMC__ > 1300)
   #pragma message ("New Compiler. Check if _MainWindow is defined")
   #pragma message (" (in bld\clib\defwin\c\winglob.c to compile for win16)")
   #pragma message (" Status V2.0 (__WATCOMC__ = 1300): unmodified since 3 years")
  #endif
 #else
  #pragma message ("Untested Compiler.") // Alte kommerzielle Compilerversionen
  HWND _TCSMainWindow= NULL;     // Ohne Default Window System?
  #define EXTERN_WINDOW _TCSMainWindow
  #undef EXTERN_INSTANCE
 #endif
 #pragma aux GetMainInstAndWin "^";     // fuer DLL: Fenster muss im Haupt-
 #pragma aux SaveMainInstAndWin "^";    // programm gespeichert werden
#endif

#ifdef __GNUC__              // MinGW und GNU:
 #if __GNUC__<4 // bis GCC 4.0 Verwendung von g77, ab 4.0 gfortran
  extern HINSTANCE _MainInst; // Symbole werden durch das (selbstgeschriebene)
  extern HWND _MainWindow;    // WinMain.c erzeugt und belegt
 #else // gfortran: Init WinMain durch Constructor, nicht libfrtbegin
  static HINSTANCE _MainInst; // Falls von mehreren Bibliotheken(TekLib,ProcInp)
  static HWND _MainWindow;    // verwendet wird nur 1 Instanz gelinkt
 #endif
 #define EXTERN_INSTANCE _MainInst
 #define EXTERN_WINDOW _MainWindow
 #define GetMainInstAndWin getmaininstandwin_
 #define SaveMainInstAndWin savemaininstandwin_
#endif

#ifdef _MSC_VER         // Microsoft Visual Cpp 6.0, ungeprueft da ohne FTN
 extern HINSTANCE hInst;
 #define EXTERN_INSTANCE hInst
 #define EXTERN_WINDOW HWND_DESKTOP
#endif



/** ***************************************************************************

 \~german
 \brief  Ermittlung Instanz und Fenster der FTN77 Hauptprogramme

  Es muss in jedem Fall zu dem Hauptprogramm gelinkt werden und darf sich
  nicht in einer DLL befinden, da sonst die Instanz der DLL ermittelt wird!
  Das Unterprogramm ist von Fortran aufrufbar.

  \param[out] hMainProgInst Instanz des Hauptprogrammes
  \param[out] hMainProgWindow Fenster des Hauptprogrammes
         Ermittlung Instanz und Fenster der FTN77 Hauptprogramme
 \~english
 \brief  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.

  \param[out] hMainProgInst instance of main
  \param[out] hMainProgWindow window of main
 \~

*************************************************************************** **/


void GetMainInstAndWin (HINSTANCE * hMainProgInst, HWND * hMainProgWindow)

{
    #if defined EXTERN_WINDOW
     *hMainProgWindow= EXTERN_WINDOW;
    #else
     *hMainProgWindow= NULL;  // wird bei Bedarf spaeter erzeugt
    #endif

    #if defined EXTERN_INSTANCE
     *hMainProgInst= EXTERN_INSTANCE;
    #else
     *hMainProgInst= NULL;
    #endif

    if (*hMainProgInst == NULL) {
     #if defined EXTERN_WINDOW
      if (EXTERN_WINDOW != NULL ) { // Hauptprogramm besitzt (bekanntes) Fenster
       #if defined __WATCOMC__      // Watcom Default Window System 16/32 bit
        #if (!defined(__WIN32__) && !defined(_WIN32))
         *hMainProgInst= (HINSTANCE)GetWindowWord(EXTERN_WINDOW, GWW_HINSTANCE);
        #else                       // Watcom ohne 64bit Windows
         *hMainProgInst= (HINSTANCE)GetWindowLong(EXTERN_WINDOW, GWL_HINSTANCE);
        #endif
       #else                        // alle anderen Compiler ohne 16bit Windows
        #if (!defined(_WIN64))      // 32 bit
         *hMainProgInst= (HINSTANCE)GetWindowLong(EXTERN_WINDOW, GWL_HINSTANCE);
        #else                       // 64 bit
        *hMainProgInst= (HINSTANCE)GetWindowLongPtr(EXTERN_WINDOW, GWLP_HINSTANCE);
        #endif
       #endif
      } else { // kein offenes Fenster, z.B. Watcom-Consolenanwendung
       *hMainProgInst= GetModuleHandle (NULL);
      }
     #else      // kein Fenster ermittelbar
      *hMainProgInst= GetModuleHandle (NULL);
     #endif
    }
}

/** ****************************************************************************

```

```
\~german
\brief Aktualisierung globalen Speichervariablen Hauptinstanz und Hauptfenster.

Notwendig nach Aufruf von CreateMainWindow_IfNecessary, da dort evtl. ein neues
Fensterhandle erzeugt wird. Da sich das Unterprogramm im Modul des Hauptprogrammes
befindet, kann das Erzeugen des Fensters auch durch eine DLL erfolgen.

\param[in] hMainProgInst Instanzenhandle
\param[in] hMainProgWindow Fensterhandle
\~english
\brief  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.

 \param[in] hMainProgInst instance of main
 \param[in] hMainProgWindow window of main
\~

*************************************************************************** **/


void SaveMainInstAndWin (HINSTANCE * hMainProgInst, HWND * hMainProgWindow)

{
    #if defined EXTERN_INSTANCE
     EXTERN_INSTANCE= *hMainProgInst;
    #endif

    #if defined EXTERN_WINDOW
     EXTERN_WINDOW= *hMainProgWindow;
    #endif
}
```

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

```
C> \file       Strings.for
C> \brief      TCS: String functions
C> \version    1.26
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C> \~german
C> Hilfsfunktionen zur Fortran Stringverarbeitung
C> \~english
C> Fortran utility functions for string processing
C> \~
C>
C
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
C
C  Unterprogramme zur Behandlung von Fortran-Strings.
C  Die Stringenden werden entweder durch CHAR(0) markiert oder
C  ueber die Deklaration ermittelt.
C
C     9.11.88     K. Friedewald
C
C  Ergaenzungen:
C     iTrimLen
C
C     7.12.01     K. Friedewald
C
C  Version: 1.26
C
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      subroutine substitute (Source, Destination, Old1, New1)
C
C  Durchsucht SOURCE nach den Substrings OLD, ersetzt sie durch NEW
C  und uebergibt das Ergebniss in DESTINATION. Wenn New=CHAR(0), werden
C  die vorkommenden OLD nur geloescht.
C
C  Stringenden koennen durch CHAR(0) markiert werden.
C
      implicit none
      integer iNext, iNext2, TempLen
      integer iStringLen
      character *(*) Source, Destination, Old1, New1
      character*255 temp, old, new

      if (istringlen(old1).le.0) return
      if (istringlen(source) .le. 0) then
       destination= char(0)
       return
      end if

      old= old1 // char(0)           ! old evtl. = Destination
      new= new1 // char(0)           ! => retten!

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



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

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



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

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



      integer function itrimlen (string)
C
C  Bestimmt die Länge des Strings ohne angehängte Leerzeichen.
C  Bei Bedarf wird ein Char(0) angehaengt. Es darf in Ftn77 nie ein
C  Nullstring erzeugt werden, da sonst die RTL-Library abstuerzt. Deswegen
C  ist der kleinste erzeugte String ein Blank ' '.
C
      implicit none
      character *(*) string
      integer i, istringlen

      i=istringlen(string) +1

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

```

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

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

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

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



C
C Graphic Input
C

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

C
C  Virtuelle Graphik, relativ
```

```
C

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



      subroutine mover (X,Y)
      call rel2ab (x,y,xabs,yabs)
      call movea (xabs,yabs)
      return
      end



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



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



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

C
C  Virtuelles Zeichnen, absolut
C

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



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



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



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



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



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

C
C  Alphanumerische Ausgabe
C

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



      subroutine ancho (ichar)
      include 'Tktrnx.fd'

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

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



      subroutine newlin
      call cartn
      call linef
      return
      end



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



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



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



      subroutine newpag
      call erase
      call home
      return
      end



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



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

C
C  Initialisierungsroutinen
C

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



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



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



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



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



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



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



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



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



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



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



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



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



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

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

```
C> \file       TCSdrWIN.for
C> \brief      MS Windows Port: High-Level Driver
C> \version    (2022, 88,x)
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C> MS Windows-spezifische TCS-Routinen
C> \note \verbatim
C>    Erweiterungen gegenüber Tektronix:
C>     subroutine TOUTSTC (String): Ausgabe Fortran-String
C>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
```

```
C>     subroutine TXTCOL (iCol): Setzen Textfarbe
C>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
C>     subroutine DefaultColour: Wiederherstellung Defaultfarben
C> \endverbatim
C>
C>
C> \~english
C> MS Windows specific subroutines
C> \note \verbatim
C>    Supplement to Tektronix:
C>     subroutine TOUTSTC (String): Print Fortran-String
C>     subroutine LINCOL (iCol): Set line color (iCol=0..15)
C>     subroutine TXTCOL (iCol): Set text color
C>     subroutine BCKCOL (iCol): Set background color (shows after ERASE)
C>     subroutine DefaultColour: Reset default colors
C> \endverbatim
C> \~
C>
C
C
C  TCS Graphik Grundfunktionen für Windows
C
C     Version 1.95 bzw. (2022,88,x)
C     - Anpassung 64bit Windows 10 und kleinere Bugfixes
C
C     Version 1.94 bzw. (2021,123,x)
C     - Ergaenzung englische Dokumentation
C
C     Version 1.93 bzw. (2020,332,x)
C     - Fehlerbehandlung analog SDL-Version
C
C     Version 1.92 bzw. (2020,230,x)
C     - Harmonisierung Commonblock TKTRNX
C     - Verwendung von khorsz, kversz, khomey in Abhängigkeit vom Zeichensatz
C
C     Version 1.91 bzw. (2017,317,x)
C     - Bugfix
C
C     Version 1.9
C     - Anpassung Windows7
C
C     Version 1.8 bzw. (2008,134,x)
C     - Hardcopy fuer Journal=3 in Form von Postscriptfiles. TBD.
C     - Ergaenzung Journal=3: Implementation Schriftarten.
C     - DRWABS bei Journal=3: Der Endpunkt wird erst beim Neuzeichnen ge-
C       setzt, im Journal steht nur die Linie mit Endpunkt. Vorteil: UNIX
C       muss den Endpunkt so nicht zweimal setzen.
C     - Fehlermeldungen der Listenverwaltung fuer Journal=3 erfolgen durch
C       GraphError bzw. Unterprogramm TCSJouListError.
C     - Bugfix TCSdWINc.h: Eintrag von TCSLEV3 in C++ Klassendefinition.
C     - Bugfix OUTGTEXT: Prüfung auf freien Platz erfolgt mit gesamtem String.
C
C     Version 1.7 bzw. (2005,291,x)
C     - Einfuehrung des Windows-unabhaengigen Journals zur Vorbereitung
C       der X11-Version. Wahl des Journaltyps (Metafile oder Liste) durch
C       bedingte Kompilation, gesteuert von der Konstante JOURNALTYP
C       im File TCSdWINc.c
C     - Bugfix GraphicError: ErrSeverity=0 entspricht jetzt NO ACTION.
C     - Das System wird nicht mehr durch Fortran-Pragmas in TCSLEV, sondern
C       durch das neue Unterprogramm TCSLEV3 in TCSdWINc.c ermittelt.
C
C     Version 1.6 bzw. (2004,302,x)
C     - Auslagern der Subroutine INITT in ein eigenes File. So kann sicher-
C       gestellt werden, dass sich INITT stets im *.exe des Hauptprogrammes
C       und nicht in einer DLL befindet und eine Ermittlung der Programm-
C       instanz und nicht der DLL-Instanz erfolgt.
C     - Sources der LIB- und DLL-Version zusammengefasst
C
C     Version 1.5 bzw. (2004,167,x)
C     - Anpassung TCSLEV: 5= Alternative Win32-Version für GCC
C
C     Version 1.4 bzw. (2004, 22,x)
C     - Bugfix OUTGTEXT: Bei c-Strings auch char(0) als Stringende erkennen
C     - Bugfix INITT1: Wiederherstellung Charakterdefinitionsblock nach
C       Erzeugung des Statusfensterfonts -> Buchstabengroesse bei ITALIC,
C       ITALIR, DBLSIZ, NRMSIZ wird jetzt richtig gesetzt.
C     - Verschieben und Scrollen Statusfenster auch bei Eingabe möglich
C
C     Version 1.3 bzw. (2003, 78,x)
C     - Falls die eigene Applikation in einem anderen Fenster aktiv ist, setzt
C       TINPUT den Fokus wieder in dieses Fenster zurück
C     - Icon für das Graphikfenster
C     - Instanzermittlung ueber Programmnamen fuer die DLL-Version
C
C     Version 1.2 bzw. (2003, 36,x)
C     - Ergänzung lib$movc3 zur Kompatibilität DOS
C     - Verwirrendes Bildschirmverhalten bei sehr langsamen Rechnern nach Erase
```

```
C       -> Einfügen UpdateWindow
C
C     Version 1.1 bzw. (2002,292,x)
C     - Umbenennung TKTRNX.FOR in TKTRNX.FD zur Kompatibilität CP/M
C
C     Version 1.0
C     - Erweiterungen gegenüber Tektronix:
C           subroutine TOUTSTC (String): Ausgabe Fortran-String
C           subroutine STATST (String) : Ausgabe String in Statusfenster
C           subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
C           subroutine TXTCOL (iCol): Setzen Textfarbe
C           subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
C           subroutine DefaultColour: Wiederherstellung Defaultfarben
C
C
C     27.09.02         Dr.-Ing. K. Friedewald
C



C
C  Ausgabe der Softwareversion
C
      subroutine tcslev(LEVEL)
      integer LEVEL(3)
      level(1)=2022     ! Aenderungsjahr
      level(2)=  88     ! Aenderungstag
C Kennzeichnung des Systems, wird im systemabhaengigem Code gesetzt
C     3=Watcom && MS-Win16  4=Watcom && MS-Win32  5=GNU-Win32  7=GNU-Win64
      call tcslev3 (level(3))

      return
      end



C
C  Abspeichern Terminal Status Area (wie DOS)
C

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



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



C
C  Relative Zeichenbefehle (wie DOS)
C

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



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



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



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

C
C   Ersatz SEELOC der CP/M-Version, SEELOC1 unnötig (wie DOS)
C

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



C
C  Textausgabe, geändert zu DOS-Version
C



      subroutine toutpt (iChr)
      include 'TKTRNX.FD'
      call outgtext (char(ichr))
      return
      end



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



      subroutine toutstc (String)
      character *(*) String
      call outgtext (string)
      return
      end



      subroutine statst (String)
      character *(*) String
      call outtext (string)
      return
      end




C
C>  Entry Dummyroutinen
C         (WINLBL keine Dummyroutine, ALPHA zusätzlich)
C

      subroutine    anmode
C> AlfMod
      entry         alfmod
C> pClipt
      entry         pclipt
```

```
C> ioWait
      entry         iowait
C> alpha
      entry         alpha
      return
      end



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

```

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

```
/** ***************************************************************************
\file       TCSdWINc.c
\brief      MS Windows Port: Low-Level Driver
\version    1.97
\author     (C) 2023 Dr.-Ing. Klaus Friedewald
\copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
\~german
         Systemnahe Graphikroutinen für das Tektronix Graphiksystem
\~english
         system-specific subroutines of the teklib-library
\~
************************************************************************** */

/*
        Anmerkungen:
          1. Die Systemmeldungen erfolgen in einem eigenen, im Regelfall
             unsichtbaren, Fenster. Durch Drücken der rechten Maustaste
             im Graphikfenster kann es sichtbar gemacht werden, durch
             Setzen des Fokus auf das Graphikfenster verschwindet es wieder.
             Bei aktiviertem GIN-Cursor kann die Umschaltung über der Titel-
             zeile erfolgen.
          2. Die Art der Protokollierung zum Neuzeichnen eines Fensters wird
             durch die Konstante JOURNALTYP gesteuert:
             --- JOURNALTYP 1 ---
             Die Zeichenbefehle werden mithilfe eines Metafiles im Speicher
             aufgezeichnet. Das Abspielen eines Metafiles in ein anderes führt
             bei Windows bis 3.0 einschließlich zum Systemabsturz! Ab Windows
             3.1 aufwärts ist das Problem behoben. Mögliche Abhilfe bei Windows
             3.0: Verwendung von Festplatten-basierten Metafiles.
             (lt. MS-SDK Dokumentation).
             --- JOURNALTYP 2: ---
             Anstelle eines Windows-Metafiles (*.wmf) wird ein extended
             Metafile (*.emf) verwendet. Funktion wurde im Hinblick auf das
             64bit-Windows entwickelt, für 32bit Windows entsteht im Vergleich
             zum Journaltyp 1 lediglich ein Performancenachteil.
             Anmerkung: MS-WORD besitzt Filter sowohl für *.wmf als auch *.emf
                        Dateien. Jedoch ist der *.emf-Filter bis WORD 2000 SP1
                        fehlerhaft (Buchstaben des stehen evtl. auf dem Kopf)
                        In Windows XP wird nach jedem Neuskalieren das *.emf
                        Metafile immer größer. Hierdurch dauert das Neuzeich-
                        nen unakzeptabel lange. Dieses Problem tritt bei
                        Windows 2000 nicht auf
                        -> JOURNALFILE 1 bei 32-bit Windows Default.
             --- JOURNALTYP 3: ---
             Die Zeichenbefehle werden in einer Liste aufgezeichnet. Ein
             einzelner Befehl hat den Aufbau
             struct xaction_typ {
                        FTNINT action
                        FTNINT i1
                        FTNINT i2
                                } XACTION;
             Die TCS-Befehle im einzelnen:
                    erase ()
                     XACTION.action= XACTION_ERASE;
                    movabs (ix,iy)
                     XACTION.action= XACTION_MOVABS;
                     XACTION.i1= ix;
                     XACTION.i2= ix;
                    drwabs (ix.iy)
                     XACTION.action= XACTION_DRWABS;
                     XACTION.i1= ix;
                     XACTION.i2= ix;
                    dshabs (ix,iy,iDash)
                     XACTION.action= XACTION_DSHSTYLE;
                     XACTION.i1= iDash;
                     XACTION.action= XACTION_DSHABS;
                     XACTION.i1= ix;
                     XACTION.i2= ix;
```

```
                 pntabs (ix,iy)
                  XACTION.action= XACTION_PNTABS;
                  XACTION.i1= ix;
                  XACTION.i2= ix;
                 outgtext (string) - Graphiktext
                  XACTION.action= XACTION_GTEXT;
                  XACTION.i1= iChar;
                  XACTION.i2= iASCII_1;
                  XACTION.action= XACTION_ASCII;
                  XACTION.i1= iASCII_2;
                  XACTION.i2= iASCII_3;
                  ...
                  XACTION.action= XACTION_ASCII;
                  XACTION.i1= iASCII_iChar;
                 italic ()
                  XACTION.action= XACTION_FONTATTR;
                  XACTION.i1= 1; // Attribut 1
                  XACTION.i2= 1; // true
                 italir ()
                  XACTION.action= XACTION_FONTATTR;
                  XACTION.i1= 1; // Attribut 1
                  XACTION.i2= 0; // false
                 dblsiz ()
                  XACTION.action= XACTION_FONTATTR;
                  XACTION.i1= 2; // Attribut 2
                  XACTION.i2= 1; // true
                 nrmsiz ()
                  XACTION.action= XACTION_FONTATTR;
                  XACTION.i1= 2; // Attribut 2
                  XACTION.i2= 0; // false

                 bckcol (iCol) - keine Zeichenarbeit, nur Commonblock
                 lincol (iCol)
                 txtcol (iCol)
                 DefaultColour () - keine Zeichenarbeit, nur Commonblock

       3. Clipping: Windows erwartet die Angabe der Clipping-region in
          Devicekoordinaten, daher wird die Clipping-Region bei Vergrößern
          und Verzerren des Fensters nicht angepasst. Abhilfe: Implementa-
          tion einer eigen Clippingroutine, gesteuert über den Tektronix-
          Commonblock. Die (funktionierende) Definition der Clippingregion
          bei Ausgabe in die Zwischenablage wird so überflüssig.
       4. Linestyle in der Regel nur durchgezogen (wird auch durch LINCOL
          zurückgesetzt) -> Merken nicht nötig. Die aktuelle Farbe muß
          jedoch für DASH gemerkt werden!!!
       5. Übergabe der Windows-Instanz:
          A.  Subroutine INITT (iDummy) ruft GetMainInstAndWin auf und
              speichert Instanz und Windowhandle durch SaveMainInstAndWin.
          B.  Übergabe des Instanz-Handlers als Parameter von INITT1 (hInst)
              Der Aufruf von INITT1 kann auch mehrmals erfolgen, d.h. möglich
              ist ein Aufruf von INITT1 durch ein C-Hauptprogramm und ein
              erneuter INITT1-Aufruf durch FORTRAN-Unterprogramm. Hier gilt
              dann der erste Aufruf, also die durch C übergebene Instanz.
          C.  Zur Vereinfachung der Programmentwicklung mit MS-Visual C++
              wird bei INITT1(0) und Kompilierung durch den MS-Compiler
              die Standardvariable hInst des Visual Studio verwendet.
       6. Initialisierung erfolgt in dem File GRAPH2D.INI
          Default: im Windows-Directory (c:\WINNT)
       7. Abweichend zur DOS-Version entspricht der Farbindex 0 weiss
          (Hintergrund) und der Index 1 schwarz.
       8. Bei Kompilierung als Konsolenanwendung oder als Window-Anwendung
          ohne Default-Windowsystem Fehler möglich. Debuggen durch
          Definition von "extended_error_handling".
          Ursache: fehlendes Fenster für das Hauptprogramm, Fehler ist
          jetzt behoben.
       9. Bei Watcom-Compiler den C-Teil ohne Optimierung compilieren!!!
      10. Getestete Compiler: WATCOM 11.0c, OpenWatcom 1.0 - 2.0.
          Bei neuen Compilern erst mit #define trace_calls übersetzen.
          Prüfen, ob __MainWindow definiert!
      11. Anpassungen an GNU-Compiler. Anstelle des Watcom-Defaultwindow-
          systems wird die eigene Routine WinMain.c verwendet.
      12. Auf Wunsch kann das Statusfenster einen privaten Device-Kontext
          erhalten: Definition des Symbols STAT_WINDOW_PRIVATE
      13. Bei mehreren Fenstern des Hauptprogrammes kann durch <Alt><F6>
          zwischen den einzelnen Fenstern umgeschaltet werden.
      14. Fuer die 16bit-Version ist das Watcom Default Window System
          notwendig. Bei 32bit ist ab der OpenWatcom Version 1.0 das
          Defaultsystem deaktiviert.
      15. Skalierung des Tektronix-Bildschirmkoordinatensystems (1023/780)
          ist bei Bildschirmen höherer Auflösung nicht ausreichend. Falls
          Anzahl der Bildschirmpixel in x-Richtung größer als 1024*Pixfac
          ist, hinterläßt der Rahmen eines über das Graphikfenster gezogenes
          Fensters horizontale und vertikale dünne Linien, die nach Mini-
          mierung und Neuzeichnen des Graphikfensters verschwinden.
          Vorsicht: PixFac *1024 darf bis einschließlich Windows95 nicht
          den 2-Byte int  Zahlenbereich (-32768...+32767) überschreiten!!!
          Bei PixFac=100 kann derzeit kein Refresh des Bildschirms durchge-
```

```
              fuehrt werden, nach erstem Zeichnen der Linie ((0,0)->(1023,780))
              erfolgt kein Neuzeichnen. Nicht nur einzige (?!) Ursache ist die
              Verwendung der 16bit GDI Befehle um METAFILE.
              Falls PixFac nicht definiert wird, erfolgt keine zusaetzliche
              Koordinatentransformation -> Performancegewinn bei alten Systemen.
          16. Im Falle von JOURNALTYP=3 darf der Fehler JOUUNKWN nur als
              Warnung definiert werden (G2dJouEntryUnknwnL= 1), da sonst inner-
              halb von TINPUT ein rekursiver Aufruf von TCSWndProc_OnPaint
              ueber GraphicError erfolgt!
              Dieser Punkt ist ab Version 1.93 mit der Verlagerung der Routine
              GraphicError in den c-Teil behoben.
          17. Die Defaultwerte des *.ini-Files müssen fuer die Initialisierung
              durch die Registry und/oder XML-Files auch bei der Variablen-
              definition angegeben werden, da GetPrivateProfileString nicht
              mehr in jedem Fall aufgerufen wird und somit Variablen evtl.
              nicht mehr vorbelegt sein koennen.
          18. Die Steuerung der Initialisierungmethode erfolgt ueber die File-
              extension des Initialisierungfiles.
              *.INI: Windows Initialisierungsfile
              *.REG: 32bit-Windows Registry
              *.XML: XML-Dateien
              Der Default (steuerbar durch das Extensiontoken .%) wird durch
               #define INIFILEXT _TEXT(".REG")         // win32: Registry
              bestimmt.
              Durch die Definition der Konstanten REGSUPPORT bzw. XMLSUPPORT
              wird der entsprechende Programmteil eingebunden.
          19. Aufgrund eines Bugs in der 32-bit Version von win7 darf eine
              Tastaturabfrage nicht ohne Filter efolgen, also nicht
               GetMessage (&msg, NULL, 0, 0);
              sondern
               GetMessage (&msg, NULL, WM_NULL, WM_USER);
              oder
               GetMessage (&msg, hWIND, 0, 0);
              Die früheren Versionen bis XP und auch die 64bit Version von Win7
              sind hiervon nicht betroffen.
          20. XML-Dateien verwenden i.d.R. UTF-8 Codierungen, deswegen erfolgt
              bei _UNICODE keine Einbindung des XML-Parsers.
          21. Journalfile Typ 3: Die verwendete Listenbibliothek verträgt sich
              nicht mit den Makros LoRes und HiRes. Deswegen darf dann PixFac
              nicht definiert werden.

*/


// #define UNICODE   // fuer Windows-Headerfiles -> jedoch Watcom FTN77 nicht
// #define _UNICODE  // fuer C-Runtime Headerfiles  UNICODEfähig !?!


/*
-------------------- Konfiguration des Zielystems ----------------------
*/

// #define PixFac  30                // s. Kommentar 15, 21
// #define STAT_WINDOW_PRIVATE       // s. Kommentar 12
// #define REGSUPPORT                // s. Kommentar 18
// #define XMLSUPPORT                // s. Kommentar 18
// #define INIFILEXT _TEXT(".XML")   // s. Kommentar 18
// #define JOURNALTYP 3              // s. Kommentar 2, 21

#if !defined(JOURNALTYP) // Defaultwerte, falls nicht oben definiert
 #if !defined(__WIN32__) && !defined(_WIN32)
  /* Defaultvorgabe 16bit: langsame CPU, Aufloesung <= 1024x780 Pxl */
  #define JOURNALTYP 1       // s. Kommentar 2, nur *.wmf implementiert
  #undef PixFac              // s. Kommentar 15, LoRes
  #undef STAT_WINDOW_PRIVATE // s. Kommentar 12
 #else
  // Default 32bit: kein extended Metafile, Auflösung <= 30*1024 x 30*780 Pxl
  #define JOURNALTYP 1       // *.emf hoeherer Overhead -> unnoetig
  #define PixFac  30         // Koordinatentransformation hochauflösende CRT's
  #undef STAT_WINDOW_PRIVATE // s. Kommentar 12
 #endif
#endif

#if !defined(INIFILEXT)
 #if !defined(__WIN32__) && !defined(_WIN32)
  #define INIFILEXT _TEXT(".INI") // s. Kommentar 18, win16: *.ini Dateien
  #undef REGSUPPORT               // Keine vollwertige Registry, nur win.ini
  #undef XMLSUPPORT               // Programmgroesse verringern
 #else
  #define INIFILEXT _TEXT(".REG") // win32: Registry
  #define REGSUPPORT
  #if (defined(__WIN64__) || defined(_WIN64))
   #define XMLSUPPORT
  #else
   #undef XMLSUPPORT
  #endif
 #endif
```

```
#endif

#if (JOURNALTYP == 3)
 #undef PixFac                   // s. Kommentar 21
#endif

#if defined(UNICODE) || defined(_UNICODE)
 #undef XMLSUPPORT               // s. Kommentar 20
#endif

/*
-------------------- Headerfiles ---------------------------------------
*/

#define WIN32_LEAN_AND_MEAN
#include <windows.h>   // Muss unbedingt vor den Standard C-Headern stehen, da
#include <windowsx.h>  // hier NULL fuer 16bit Windows als 0 definiert wird

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <tchar.h>     // Public Domain ueber MINGW-Package, nicht nur MS

#if defined(__WATCOMC__) && defined(__SW_BW)
 #include <wdefwin.h>   // Compilerswitch -bw: Watcom Default Window System
#endif

#ifdef XMLSUPPORT
 #include  "mxml.h"
#endif

#if (JOURNALTYP == 3)
 #include "sglib.h"
#endif

#include "TCSdWINc.h"
#include "TKTRNX.h"

/*
-------------------- Debug Compiler Switches ---------------------------
*/

// #define extended_error_handling
#if !defined(__WIN32__) && !defined(_WIN32)
  #undef extended_error_handling
#endif

// #define trace_calls
/* Debug-Messageboxen / Compilermessages, nach include definieren! */

#ifdef trace_calls

 #ifdef __WATCOMC__
  #if (__WATCOMC__ == 1100)
   #pragma message ( "Symbol __WATCOMC__ defined to 1100 (Version 11.0c)")
  #elif (__WATCOMC__ >= 1200)
   #pragma message ( "Symbol __WATCOMC__ defined (OpenWatcom Version >= 1.0)")
  #else
   /* Andere Versionen noch nicht getestet! */
   #pragma message ( "Untested Version: Symbol __WATCOMC__ defined to :")
   #pragma message (__WATCOMC__) // Erzwingen Fehler zur Erweiterung
  #endif
  #if !defined(__WIN32__) && !defined(_WIN32)
   #pragma message ( "16 bit Windows" )
  #else
   #pragma message ( "32 bit Windows" )
  #endif
 #endif

 #ifdef _MSC_VER
  #pragma message ( "Symbol _MSC_VER defined" )
  #if !defined(__WIN32__) && !defined(_WIN32)
   #pragma message ( "16 bit Windows" )
  #else
   #pragma message ( "32 bit Windows" )
  #endif
 #endif

 #ifdef __GNUC__
  #warning "GNU-Compiler"
  #if !defined(__WIN32__) && !defined(_WIN32)
   #warning "16 bit Windows"
  #elif !defined(__WIN64__) && !defined(_WIN64)
   #warning "32 bit Windows"
  #else
   #warning "64 bit Windows"
  #endif
 #endif

#endif

/*
-------------- Compilerunabhaengige externe Bezüge ---------------------
*/


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


/*
------------------------- Globale Variablen ----------------------------
*/

static  RECT    TCSrect = {0,0, HiRes(TEK_XMAX),HiRes(TEK_YMAX)}; // Plotbereich

static  bool    TCSinitialized = false,
                ClippingNotActive = true,
                TCSStatWindowAutomatic = true;

static  HINSTANCE hTCSInst = NULL;

static  HWND    hTCSWindow = NULL,
                hTCSstatWindow = NULL,
                hOwnerWindow = NULL;

static  HDC     hTCSWindowDC;     // privater DC, gilt ganze Fensterlebensdauer

#if (JOURNALTYP == 1)
 static  HDC    hTCSMetaFileDC;   // Metafile als Recorder für WM_PAINT
#elif (JOURNALTYP == 2)
 static  HDC    hTCSMetaFileDC;   // extended Metafile als Recorder WM_PAINT
#elif (JOURNALTYP == 3)
 struct xJournalEntry_typ {struct xJournalEntry_typ * previous;
                           struct xJournalEntry_typ * next;
                           FTNINT action; FTNINT i1; FTNINT i2;};
 static struct xJournalEntry_typ* hTCSJournal = NULL;
                                  // Journal zum Neuzeichnen des Fensters
#endif

#ifdef STAT_WINDOW_PRIVATE
 static HDC     hTCSstatWindowDC;
#endif


static  LOGFONT TCSFontdefinition;

static  HFONT   hTCSFont,
                hTCSSysFont;

static  HPEN    hTCSPen;

static  HCURSOR hGinCurs,
                hMouseCurs;

static  TCHAR   szTCSWindowName[TCS_WINDOW_NAMELEN] = "", // Default TCS_WINDOW_NAME erst in ??
 gesetzt
                szTCSstatWindowName[TCS_WINDOW_NAMELEN] = "", // TCS_STATWINDOW_NAME,
                szTCSMainWindowName[TCS_WINDOW_NAMELEN] = TCS_MAINWINDOW_NAME,
                szTCSIniFile[TCS_FILE_NAMELEN] = TCS_INIFILE_NAME INIFILEXT,
                szTCSIconFile[TCS_FILE_NAMELEN] = TCS_ICONFILE_NAME,
                szTCSMenuCopyText[TCS_MENUENTRY_LEN] = TCS_INIDEF_COPMEN,
                szTCSHardcopyFile[TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME,
                szTCSGraphicFont[TCS_FILE_NAMELEN] = TCS_INIDEF_FONT,
                szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT,
                szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0;


typedef TCHAR   StatLine[STAT_MAXCOLUMNS+1];
static  StatLine TCSstatTextBuf[STAT_MAXROWS];

static  int     TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX, // rel. Bildschirmpos.
                TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY, // bei Init in %
                TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX,
                TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY,
                TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX, // dito
                TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY, // Statusfenster
                TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX,
                TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY,
                TCSstatScrollY, // Position des sichtbaren Scrollbereichs
                TCSstatOrgY,    // Ursprung des log. Koordinatensystems
                TCSstatCursorPosY,
                TCSstatRow,
                TextLineHeight,
                  TCSCharHeight,
                  TCSBackgroundColour,
                  TCSDefaultLinCol = TCS_INIDEF_LINCOL,
                  TCSDefaultTxtCol = TCS_INIDEF_TXTCOL,
                  TCSDefaultBckCol = TCS_INIDEF_BCKCOL,
                  iHardcopyCount =1;  // Zähler zur Erzeugung Filenamen

static   POINT   TCSGinCurPos = { TEK_XMAX / 2, TEK_YMAX / 2};


/* Zuordnung Fehlernummern zu Meldungen,  */

typedef TCHAR   ErrMsg[STAT_MAXCOLUMNS];
static  ErrMsg  szTCSErrorMsg[(int) MSG_MAXERRNO+1] =
                {_T("Element 0 unused"),_T("DOS"),_T("DOS"),_T("DOS"),
                 _T("DOS"),_T("DOS"),    // Errno 0..5
                TCS_INIDEF_HDCOPN,       // Errno 6
                TCS_INIDEF_HDCWRT,       // Errno 7
                TCS_INIDEF_HDCINT,       // Errno 8
                TCS_INIDEF_USR,          // Errno 9
                TCS_INIDEF_HDCACT,       // Errno 10
                TCS_INIDEF_USRWRN,       // Errno 11
                TCS_INIDEF_EXIT,         // Errno 12
                TCS_INIDEF_COPMEM,       // Errno 13
                TCS_INIDEF_COPLCK,       // Errno 14
                TCS_INIDEF_JOUCREATE,    // Errno 15
                TCS_INIDEF_JOUENTRY,     // Errno 16
                TCS_INIDEF_JOUADD,       // Errno 17
                TCS_INIDEF_JOUCLR,       // Errno 18
                TCS_INIDEF_JOUUNKWN,     // Errno 19
                TCS_INIDEF_XMLPARSER,    // Errno 20
                TCS_INIDEF_XMLOPEN,      // Errno 21
                _T("SDL"),
                TCS_INIDEF_USR2,         // Errno 23
                TCS_INIDEF_INI2,         // Errno 24
                _T("Maxerr only for internal Use") };

static  int     TCSErrorLev[(int) MSG_MAXERRNO+1] =
                {10,10,10,10,10,10,
                TCS_INIDEF_HDCOPNL,      // Errno 6
                TCS_INIDEF_HDCWRTL,      // Errno 7
                TCS_INIDEF_HDCINTL,      // Errno 8
                TCS_INIDEF_USRL,         // Errno 9
                TCS_INIDEF_HDCACTL,      // Errno 10
                TCS_INIDEF_USRWRNL,      // Errno 11
                TCS_INIDEF_EXITL,        // Errno 12
                TCS_INIDEF_COPMEML,      // Errno 13
                TCS_INIDEF_COPLCKL,      // Errno 14
                TCS_INIDEF_JOUCREATEL,   // Errno 15
                TCS_INIDEF_JOUENTRYL,    // Errno 16
                TCS_INIDEF_JOUADDL,      // Errno 17
                TCS_INIDEF_JOUCLRL,      // Errno 18
                TCS_INIDEF_JOUUNKWNL,    // Errno 19
                TCS_INIDEF_XMLPARSERL,   // Errno 20
                TCS_INIDEF_XMLOPENL,     // Errno 21
                10,
                TCS_INIDEF_USR2L,        // Errno 23
                TCS_INIDEF_INI2L,        // Errno 24
                10};



/* Zuordnung der Linienarten zu Liniennummern */

static  DWORD dwPenStyle[] = {
                            PS_SOLID,    /* iMask= 0 */
                            PS_DOT,      /* iMask= 1 */
                            PS_DASHDOT,  /* iMask= 2 */
                            PS_DASH      /* iMask= 3 */
                           };
#define MAX_PENSTYLE_INDEX 3


/* Zuordnung der Farbennummern zur VGA-Palette */

static  DWORD dwColorTable[] = {
                            RGB (240,240,240), /* iCol= 00: weiss (DOS: 01) */
                            RGB (  0,  0,  0), /* iCol= 01: schwarz(DOS:00) */
                            RGB (240, 80, 80), /* iCol= 02: rot             */
                            RGB ( 80,240, 80), /* iCol= 03: gruen           */
                            RGB ( 80,240,240), /* iCol= 04: blau            */
                            RGB ( 80, 80,240), /* iCol= 05: lila            */
                            RGB (240,240, 80), /* iCol= 06: gelb            */
                            RGB (160,160,160), /* iCol= 07: grau            */
                            RGB (240, 80,240), /* iCol= 08: violett         */
                            RGB (160,  0,  0), /* iCol= 09: mattrot         */
                            RGB (  0,160,  0), /* iCol= 10: mattgruen       */
```

```
                             RGB (  0,  0,160), /* iCol= 11: mattblau        */
                             RGB (  0,160,160), /* iCol= 12: mattlila        */
                             RGB (160, 80,  0), /* iCol= 13: orange          */
                             RGB ( 80, 80, 80), /* iCol= 14: mattgrau        */
                             RGB (160,  0,160)  /* iCol= 15: mattviolett     */
                            };
#define MAX_COLOR_INDEX 15



/*
----------------------- Globale Unterprogramme ------------------------
*/



void TCSGraphicError (int iErr, const char* msg)
{
char cBuf[TCS_MESSAGELEN];
FTNINT i; // Dummyparameter
FTNSTRDESC  ftnstrg;

    snprintf( cBuf, TCS_MESSAGELEN, szTCSErrorMsg[iErr], msg );
    if ((iErr == WRN_JOUUNKWN) || // Rekursion von TCSWndProc_OnPaint vermeiden
        (iErr == ERR_XMLOPEN)           ) { // System noch nicht initialisiert
     MessageBox (NULL, _T(cBuf), szTCSWindowName, MB_ICONINFORMATION);
    } else { // ab jetzt mit bell, outtext...
     InvalidateRect (hTCSWindow, NULL, true); /* ,ClientArea, EraseFlag */
     UpdateWindow (hTCSWindow); /* Notwendig bei OnPaint mit Journaltyp=3 */
     bell (); // -> MessgageBeep / winuser.h, ohne Initialisierung verwendbar
     ftnstrg.addr= cBuf; ftnstrg.len= strlen (cBuf);
     outtext (CALLFTNSTRA(ftnstrg) CALLFTNSTRL(ftnstrg));
     if (TCSErrorLev[iErr] >1) {
      if (TCSErrorLev[iErr] < 10) {
       if (TCSErrorLev[iErr] == 5) {
        tinput (&i); // Press Any Key
       }
       if (TCSErrorLev[iErr]==8) {
        MessageBox (NULL, _T(cBuf), szTCSWindowName, MB_ICONINFORMATION);
       }
      } else {
       if (TCSErrorLev[iErr] == 10) {
        tinput (&i); // Press Any Key
       }
       if (TCSErrorLev[iErr]==12) {
         MessageBox (NULL, _T(cBuf), szTCSWindowName, MB_ICONSTOP);
       }
       if (iErr != ERR_EXIT) { // Error-Level von finitt durch XML veraenderbar
         TCSErrorLev[ERR_EXIT] = 10; // Hier: Fehler mit Programmabbruch
         finitt ();                  // Erzwungenes Beenden durch finitt
       }
      }
     }
    }
}



// ------------- Unterprogramme fuer die Event Handler  ----------------




// ------------- Unterprogramme für die Userroutinen -------------------


#if defined(REGSUPPORT)
 void StoreIni (TCHAR * szSection, TCHAR * szField, TCHAR * szValue)
 {

    if (_tcsicmp (szSection,TCS_INISECT1) == 0 ) { // Section1: Names ---------
     if (_tcsicmp (szField,TCS_INIVAR_WINNAM) == 0 ) {
      if (_tcslen(szTCSWindowName)==0) _tcsncpy(szTCSWindowName,
                                            szValue,TCS_WINDOW_NAMELEN-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_STATNAM) == 0 ) {
      if (_tcslen(szTCSstatWindowName)==0) _tcsncpy(szTCSstatWindowName,
                                            szValue,TCS_WINDOW_NAMELEN-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_MAINWINNAM) == 0 ) {
      _tcsncpy(szTCSMainWindowName, szValue,TCS_WINDOW_NAMELEN-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_HDCNAM) == 0 ) {
      _tcsncpy(szTCSHardcopyFile, szValue,TCS_FILE_NAMELEN-1);
     }

    } else if (_tcsicmp (szSection,TCS_INISECT2) == 0 ) { // Section2: Layout -
     if (_tcsicmp (szField,TCS_INIVAR_COPMEN) == 0 ) {
      _tcsncpy(szTCSMenuCopyText, szValue,TCS_MENUENTRY_LEN-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_FONT) == 0 ) {
      _tcsncpy(szTCSGraphicFont, szValue,TCS_FILE_NAMELEN-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_SYSFONT) == 0 ) {
      _tcsncpy(szTCSSysFont, szValue,TCS_FILE_NAMELEN-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_ICONNAM) == 0 ) {
      _tcsncpy(szTCSIconFile, szValue,TCS_FILE_NAMELEN-1);

     } else if (_tcsicmp (szField,TCS_INIVAR_WINPOSX) == 0 ) {
      TCSwindowIniXrelpos= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_WINPOSY) == 0 ) {
      TCSwindowIniYrelpos= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_WINSIZX) == 0 ) {
      TCSwindowIniXrelsiz= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_WINSIZY) == 0 ) {
      TCSwindowIniYrelsiz= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_STATPOSX) == 0 ) {
      TCSstatWindowIniXrelpos= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_STATPOSY) == 0 ) {
      TCSstatWindowIniYrelpos= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_STATSIZX) == 0 ) {
      TCSstatWindowIniXrelsiz= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_STATSIZY) == 0 ) {
      TCSstatWindowIniYrelsiz= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_LINCOL) == 0 ) {
      TCSDefaultLinCol= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_TXTCOL) == 0 ) {
      TCSDefaultTxtCol= * (int*) szValue;
     } else if (_tcsicmp (szField,TCS_INIVAR_BCKCOL) == 0 ) {
      TCSDefaultBckCol= * (int*) szValue;
     }

    } else if (_tcsicmp (szSection,TCS_INISECT3) == 0 ) { // Section3: Messages
     if (_tcsicmp (szField,TCS_INIVAR_HDCOPN) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_HDCFILOPN], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_HDCOPNL) == 0 ) {
      TCSErrorLev[WRN_HDCFILOPN]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_HDCWRT) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_HDCFILWRT], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_HDCWRTL) == 0 ) {
      TCSErrorLev[WRN_HDCFILWRT]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_HDCINT) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_HDCINTERN], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_HDCINTL) == 0 ) {
      TCSErrorLev[WRN_HDCINTERN]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_USR) == 0 ) {
      _tcsncpy(szTCSErrorMsg[MSG_USR], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_USRL) == 0 ) {
      TCSErrorLev[MSG_USR]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_HDCACT) == 0 ) {
      _tcsncpy(szTCSErrorMsg[MSG_HDCACT], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_HDCACTL) == 0 ) {
      TCSErrorLev[MSG_HDCACT]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_USRWRN) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_USRPRESSANY], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_USRWRNL) == 0 ) {
      TCSErrorLev[WRN_USRPRESSANY]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_EXIT) == 0 ) {
      _tcsncpy(szTCSErrorMsg[ERR_EXIT], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_EXITL) == 0 ) {
      TCSErrorLev[ERR_EXIT]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_COPMEM) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_COPYNOMEM], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_COPMEML) == 0 ) {
      TCSErrorLev[WRN_COPYNOMEM]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_COPLCK) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_COPYLOCK], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_COPLCKL) == 0 ) {
      TCSErrorLev[WRN_COPYLOCK]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_JOUCREATE) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_JOUCREATE], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_JOUCREATEL) == 0 ) {
      TCSErrorLev[WRN_JOUCREATE]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_JOUENTRY) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_JOUENTRY], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_JOUENTRYL) == 0 ) {
      TCSErrorLev[WRN_JOUENTRY]= * (int*) szValue;
```

```

     } else if (_tcsicmp (szField,TCS_INIVAR_JOUADD) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_JOUADD], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_JOUADDL) == 0 ) {
      TCSErrorLev[WRN_JOUADD]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_JOUCLR) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_JOUCLR], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_JOUCLRL) == 0 ) {
      TCSErrorLev[WRN_JOUCLR]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_JOUUNKWN) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_JOUUNKWN], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_JOUUNKWNL) == 0 ) {
      TCSErrorLev[WRN_JOUUNKWN]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_XMLPARSER) == 0 ) {
      _tcsncpy(szTCSErrorMsg[ERR_XMLPARSER], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_XMLPARSERL) == 0 ) {
      TCSErrorLev[ERR_XMLPARSER]= * (int*) szValue;

     } else if (_tcsicmp (szField,ERR_XMLOPEN) == 0 ) {
      _tcsncpy(szTCSErrorMsg[ERR_XMLOPEN], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_XMLOPENL) == 0 ) {
      TCSErrorLev[ERR_XMLOPEN]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_USR2) == 0 ) {
      _tcsncpy(szTCSErrorMsg[MSG_USR2], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_USR2L) == 0 ) {
      TCSErrorLev[MSG_USR2]= * (int*) szValue;

     } else if (_tcsicmp (szField,TCS_INIVAR_INI2) == 0 ) {
      _tcsncpy(szTCSErrorMsg[WRN_INI2], szValue,STAT_MAXCOLUMNS-1);
     } else if (_tcsicmp (szField,TCS_INIVAR_INI2L) == 0 ) {
      TCSErrorLev[WRN_INI2]= * (int*) szValue;

     }

    } // End case section

 }
#endif


bool PointInWindow (FTNINT ix1, FTNINT iy1)
{
    if (ClippingNotActive ) return true;
    return ( (TKTRNX.kminsx <= ix1) && (TKTRNX.kmaxsx >= ix1) &&
                     (TKTRNX.kminsy <= iy1) && (TKTRNX.kmaxsy >= iy1));
}



bool ClipLineStart (FTNINT ix1, FTNINT iy1, FTNINT ix2, FTNINT iy2,
                                                 FTNINT *isx, FTNINT *isy)
/* ClipLineStart=true: isx,isy Startpunkt; =false: Linie nicht zeichnen */
{
    if (ClippingNotActive) {
     *isx= ix1; *isy= iy1;
     return true;
    }

    if (ix1 < TKTRNX.kminsx) { /* Start links vom Fenster */
     if (ix2 < TKTRNX.kminsx) return false;
     *isy= iy1+((TKTRNX.kminsx-ix1) * (iy2-iy1)) / (ix2-ix1);
     if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
      *isx= TKTRNX.kminsx;
      return true;
     }
     if (iy1 == iy2) return false;
     if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
      *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
      *isy= TKTRNX.kminsy;
     } else {
      *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
      *isy= TKTRNX.kmaxsy;
     }
     if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
     return true;

    } else if (ix1 > TKTRNX.kmaxsx) { /* Start rechts vom Fenster */
     if (ix2 > TKTRNX.kmaxsx) return false;
     *isy= iy1+((TKTRNX.kmaxsx-ix1) * (iy2-iy1)) / (ix2-ix1);
     if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
      *isx= TKTRNX.kmaxsx;
      return true;
     }
     if (iy1 == iy2) return false;
     if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
      *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
      *isy= TKTRNX.kmaxsy;
     } else {
      *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
      *isy= TKTRNX.kminsy;
     }
     if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
     return true;

    } else if (iy1 < TKTRNX.kminsy) { /* Start unter dem Fenster */
     if (iy2 < TKTRNX.kminsy) return false;
     *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
     if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
     *isy= TKTRNX.kminsy;
     return true;

    } else if (iy1 > TKTRNX.kmaxsy) { /* Start ueber dem Fenster */
     if (iy2 > TKTRNX.kmaxsy) return false;
     *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
     if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
     *isy= TKTRNX.kmaxsy;
     return true;

    }
    *isx= ix1;                         /* Startpunkt liegt im Fenster */
    *isy= iy1;
    return true;
}



/*
------------------- Event Handler zum Parsen von XML-Dateien ----------
*/

#if defined(XMLSUPPORT)

void sax_callback (mxml_node_t *node, mxml_sax_event_t event, void *usr)
{
char * StorePtr;

    switch (event) {
     case MXML_SAX_ELEMENT_OPEN: {
      switch (*(int*)usr ) {
       case -1: { // Statemachine: noch keine aktive Sektion
        if (strcmp(mxmlGetElement(node),szTCSsect0) == 0) {
         *(int*)usr= 0;  // Parsing active
         mxmlElementSetAttr (node,"typ","none");
        }
        break;
       }
       case 0: {
        if ((strcmp(mxmlGetElement(node),TCS_INISECT1) == 0)   ) {
         *(int*)usr= 1; // State: TCS_INISECT1
        } else if ((strcmp(mxmlGetElement(node),TCS_INISECT2) == 0)   ) {
         *(int*)usr= 2; // State: TCS_INISECT2
        } else if ((strcmp(mxmlGetElement(node),TCS_INISECT3) == 0)   ) {
         *(int*)usr= 3; // State: TCS_INISECT3
        }
        mxmlElementSetAttr (node,"typ","none");
        break;
       }

       case 1: { // Section = Names
        if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINNAM) == 0)   ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSWindowName);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATNAM) == 0)   ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSstatWindowName);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_MAINWINNAM) == 0)   ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSMainWindowName);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCNAM) == 0)   ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSHardcopyFile);
        }
        break;
       }

       case 2: { // Section = Layout
        if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPMEN) == 0)   ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSMenuCopyText);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_FONT) == 0)   ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSGraphicFont);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_SYSFONT) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSSysFont);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_ICONNAM) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSIconFile);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSX) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelpos);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSY) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelpos);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZX) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelsiz);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZY) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelsiz);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSX) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelpos);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSY) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelpos);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZX) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelsiz);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZY) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelsiz);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_LINCOL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSDefaultLinCol);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_TXTCOL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSDefaultTxtCol);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_BCKCOL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSDefaultBckCol);
        }
        break;
       }

       case 3: { // Section = Messages
        if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPN) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILOPN]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPNL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILOPN]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRT) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILWRT]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRTL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILWRT]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINT) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCINTERN]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINTL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCINTERN]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACT) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_HDCACT]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACTL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_HDCACT]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRN) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_USRPRESSANY]);
```

```
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRNL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_USRPRESSANY]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXIT) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_EXIT]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXITL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_EXIT]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPMEM) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_COPYNOMEM]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPMEML) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_COPYNOMEM]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPLCK) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_COPYLOCK]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_COPLCKL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_COPYLOCK]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATE) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCREATE]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATEL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCREATE]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRY) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUENTRY]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRYL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUENTRY]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADD) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUADD]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADDL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUADD]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLR) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCLR]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLRL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCLR]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWN) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUUNKWN]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWNL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUUNKWN]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSER) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLPARSER]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSERL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLPARSER]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPEN) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLOPEN]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPENL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLOPEN]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR2]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2L) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR2]);

        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_INI2]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2L) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_INI2]);

        }
        break;
       }

      }
      break;
     }

     case MXML_SAX_DATA: {
      switch (mxmlGetType(node)) {
       case MXML_INTEGER: {
        sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
        (*(int*)StorePtr)= mxmlGetInteger(node);
        break;
       }
       case MXML_REAL: {
        sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
        (*(float*)StorePtr)= mxmlGetReal(node);
        break;
       }
       case MXML_TEXT: {
        sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
        strcpy (StorePtr, mxmlGetText(node, NULL));
        break;
       }
       case MXML_OPAQUE: {
        sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
        strcpy (StorePtr, mxmlGetOpaque(node));
        break;
       }
      }
      break;
     }

     case MXML_SAX_ELEMENT_CLOSE: {
      if ((*(int*)usr==0) && (strcmp(mxmlGetElement(node),szTCSsect0)==0)) {
       *(int*)usr= -1; // State: idle
      } else if (
             ((*(int*)usr==1) && (strcmp(mxmlGetElement(node),TCS_INISECT1)==0))
          || ((*(int*)usr==2) && (strcmp(mxmlGetElement(node),TCS_INISECT2)==0))
          || ((*(int*)usr==3) && (strcmp(mxmlGetElement(node),TCS_INISECT3)==0))
          ) {
       *(int*)usr= 0; // State: Parsing active
      }
      break;
     }
    }
}


/* ------------------------------------------------------------------------- */


mxml_type_t      sax_type_callback(mxml_node_t  *node)
{
const char *type;

    if ((type = mxmlElementGetAttr(node, "typ")) == NULL) type = "none";
    if (!strcmp(type, "integer"))
     return (MXML_INTEGER);
    else if (!strcmp(type, "opaque") || !strcmp(type, "pre"))
     return (MXML_OPAQUE);
    else if (!strcmp(type, "real"))
     return (MXML_REAL);
    else if (!strcmp(type, "text"))
     return (MXML_TEXT);
    else
     return (MXML_IGNORE);
}

/* ------------------------------------------------------------------------- */


mxml_error_cb_t sax_error_callback (char *mssg)
{
    TCSGraphicError (ERR_XMLPARSER, mssg);
    return;
}

/* ------------------------------------------------------------------------- */

#endif   // Ende XML-Unterstützung




```

```
/*
------------------- Event Handler Graphikfenster -----------------------
*/




void TCSWndProc_OnPaint (HWND hWindow)
{
PAINTSTRUCT ps;
#if (JOURNALTYP == 1)
 HMETAFILE hmf;
 HDC hTCSMetaFileDC1;
#elif (JOURNALTYP == 2)
 HENHMETAFILE hmf;
 ENHMETAHEADER emh ;
 HDC hTCSMetaFileDC1;
 RECT   crtrect;
#elif (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
 HPEN   hPenDash, hPenOld;
 HFONT  hOldFont;
 int    iMaskIndex;
 int    iGraphTextLen, iGraphTextLenAkt;
 TCHAR  GraphTextBuf[STAT_MAXCOLUMNS+1];
#endif


    BeginPaint (hWindow, &ps);

#if (JOURNALTYP == 1)
    hmf = CloseMetaFile (hTCSMetaFileDC);
    PlayMetaFile (hTCSWindowDC, hmf);           /* Wiederherstellung Anzeige */

    hTCSMetaFileDC1  = CreateMetaFile (NULL);   /* 16bit Windows Metafile */
    PlayMetaFile (hTCSMetaFileDC1, hmf);        /* für neues Journalfile */
    DeleteMetaFile (hmf);                       /* alter Status Bildschirm */
    hTCSMetaFileDC = hTCSMetaFileDC1;           /* bereit zum Weiterzeichnen */

#elif (JOURNALTYP == 2)
    hmf = CloseEnhMetaFile (hTCSMetaFileDC);
    GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
    GetClientRect(hTCSWindow, &crtrect); // Zeichenbereich CRT in Pixeln

    SetViewportExtEx (hTCSWindowDC, crtrect.right-crtrect.left,
                        crtrect.bottom-crtrect.top, NULL); // Zeichne EMF 1:1
    SetViewportOrgEx (hTCSWindowDC, crtrect.left, crtrect.bottom, NULL);
    SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);

    PlayEnhMetaFile (hTCSWindowDC, hmf, &TCSrect); // Wiederherstellung Anzeige

    SetViewportExtEx (hTCSWindowDC, crtrect.right-crtrect.left,
                         crtrect.top-crtrect.bottom, NULL); // Skaliere auf TEK
    SetViewportOrgEx (hTCSWindowDC, crtrect.left, crtrect.top, NULL);
    SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);


    hTCSMetaFileDC1  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
                        _T("TCS for Windows\0Journalfile created by OnPaint\0"));

    SetMapMode (hTCSMetaFileDC1, MM_ANISOTROPIC);
    SetViewportExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);
    SetWindowExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);

    PlayEnhMetaFile (hTCSMetaFileDC1, hmf, &TCSrect); // neues Journal

    DeleteEnhMetaFile (hmf);                    // Bildschirminhalt restauriert
    hTCSMetaFileDC = hTCSMetaFileDC1;           // bereit zum Weiterzeichnen
    SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
    SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #endif
    SetBkMode (hTCSMetaFileDC, TRANSPARENT );   // Metafile weitergegeben !
    SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); // CP
    SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif

#elif (JOURNALTYP == 3)
//      if (hTCSJournal != NULL) {
    SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, hTCSJournal, previous, next, xJournalEntry)
    while (xJournalEntry != NULL) {
     switch (xJournalEntry->action) {
      case XACTION_INITT: {
       TKTRNX.iLinCol= TCSDefaultLinCol;
       TKTRNX.iTxtCol= TCSDefaultTxtCol;
       TKTRNX.iBckCol= TCSDefaultBckCol;
       initt2(); // HOME, Font, Scale...
      } // weiter mit Erase
      case XACTION_ERASE: {
       SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
       SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);
       SetBkMode (hTCSWindowDC, TRANSPARENT );
       SetTextAlign (hTCSWindowDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
       SetTextColor (hTCSWindowDC, dwColorTable[TKTRNX.iTxtCol]);
       #if !defined(__WIN32__) && !defined(_WIN32)
        SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
       #else
        SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
       #endif
       break;
      }
      case XACTION_MOVABS: {
       MoveToEx (hTCSWindowDC, HiRes(xJournalEntry->i1),
                                            HiRes(xJournalEntry->i2), NULL);
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_DRWABS: {
       LineTo (hTCSWindowDC, HiRes(xJournalEntry->i1),
                   HiRes(xJournalEntry->i2) ); // Endpunkt nicht mitgezeichnet!
       SetPixel (hTCSWindowDC,HiRes(xJournalEntry->i1),
                       HiRes(xJournalEntry->i2), dwColorTable[TKTRNX.iLinCol]);
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_DSHSTYLE: {
       iMaskIndex= xJournalEntry->i1;
       break;
      }
      case XACTION_DSHABS: {
       hPenDash= CreatePen (dwPenStyle[iMaskIndex], 0,
                                              dwColorTable[TKTRNX.iLinCol]);
       #if !defined(__WIN32__) && !defined(_WIN32)
        SelectPen (hTCSWindowDC, hPenDash); // 16bit: Makro aus windowsx.h
       #else
        SelectObject (hTCSWindowDC, hPenDash); // 32bit: GDI Standardaufruf
       #endif
       LineTo (hTCSWindowDC, HiRes(xJournalEntry->i1),
                                              HiRes(xJournalEntry->i2) );
       #if !defined(__WIN32__) && !defined(_WIN32)
        SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
        DeletePen (hPenDash);
       #else
        SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
        DeleteObject (hPenDash);
       #endif
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_PNTABS: {
       SetPixel (hTCSWindowDC,HiRes(xJournalEntry->i1),
                   HiRes(xJournalEntry->i2), dwColorTable[TKTRNX.iLinCol]);
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_BCKCOL: {
       TKTRNX.iBckCol= xJournalEntry->i1;
       break;
      }
      case XACTION_LINCOL: {
       hTCSPen= CreatePen (PS_SOLID, 0, dwColorTable[xJournalEntry->i1]);
       #if !defined(__WIN32__) && !defined(_WIN32)
        hPenOld= SelectPen (hTCSWindowDC, hTCSPen);// 16bit: Makro aus windowsx.h
        DeletePen (hPenOld);
       #else
        hPenOld= SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
        DeleteObject (hPenOld);
       #endif
       TKTRNX.iLinCol= xJournalEntry->i1;
       break;
      }
      case XACTION_TXTCOL: {
       SetTextColor (hTCSWindowDC, dwColorTable[xJournalEntry->i1]);
       TKTRNX.iTxtCol= xJournalEntry->i1;
       break;
      }
      case XACTION_FONTATTR: {
       TKTRNX.kitalc= xJournalEntry->i1;
       TCSFontdefinition.lfItalic= (TKTRNX.kitalc > 0);
       hTCSFont= CreateFontIndirect (&TCSFontdefinition);
       #if !defined(__WIN32__) && !defined(_WIN32)
        hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
        DeleteFont (hOldFont);
       #else
        hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
        DeleteObject (hOldFont);
       #endif

       if (TKTRNX.ksizef != xJournalEntry->i2) {
        TKTRNX.ksizef= xJournalEntry->i2;
        TCSFontdefinition.lfHeight= (1+TKTRNX.ksizef)*TCSCharHeight;
        TCSFontdefinition.lfWidth= 0;
        hTCSFont= CreateFontIndirect (&TCSFontdefinition);
        #if !defined(__WIN32__) && !defined(_WIN32)
         hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
         DeleteFont (hOldFont);
        #else
         hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
         DeleteObject (hOldFont);
        #endif
        TKTRNX.khomey = TEK_YMAX - 1.5f*(1+TKTRNX.ksizef)*TCS_REL_CHR_HEIGHT;
       }
       break;
      }
      case XACTION_GTEXT: {
       iGraphTextLenAkt= 0;
       iGraphTextLen= (int) xJournalEntry->i1;
       if (iGraphTextLen > STAT_MAXCOLUMNS) iGraphTextLen= STAT_MAXCOLUMNS;
       if (iGraphTextLen == 0) break;
       GraphTextBuf[iGraphTextLenAkt++]= (TCHAR) xJournalEntry->i2;
       if (iGraphTextLen == 1) {
        GraphTextBuf[iGraphTextLenAkt]= (FTNCHAR) 0;
        TextOut (hTCSWindowDC, 0,0,GraphTextBuf, iGraphTextLen);
       }
       break;
      }
      case XACTION_ASCII: {
       if (iGraphTextLenAkt < iGraphTextLen) {
        GraphTextBuf[iGraphTextLenAkt++]= (TCHAR) xJournalEntry->i1;
        if (iGraphTextLenAkt < iGraphTextLen)
         GraphTextBuf[iGraphTextLenAkt++]= (TCHAR) xJournalEntry->i2;
        if (iGraphTextLenAkt >= iGraphTextLen)
         TextOut (hTCSWindowDC, 0,0,GraphTextBuf, iGraphTextLen);
       }
       break;
      }
      case XACTION_NOOP: {
       break;
      }
      default: {
       TCSGraphicError (WRN_JOUUNKWN,"");
       break;
      }
     }
     xJournalEntry= xJournalEntry -> previous;
    }
//      }
#endif

    EndPaint( hWindow, &ps );
}



void TCSWndProc_OnSize (HWND hWindow, UINT message, WPARAM width, LPARAM height)
{
    switch (message) {
     case SIZE_MINIMIZED:   /* Minimierung -> keine Aktion notwendig */
      break;
     case SIZE_RESTORED:    /*(Erst- oder Neu)Skalierung des Fensters */
     case SIZE_MAXIMIZED:       /* sichtbar: 0<=ix<=1023 / 0<=iy<=780 */
      SetMapMode (hTCSWindowDC, MM_ANISOTROPIC);
      SetViewportExtEx (hTCSWindowDC, width, -height, NULL);
      SetViewportOrgEx (hTCSWindowDC, 0, 0, NULL);
      /* Bei erneuter Änderung des Viewport geht die Auflösung verloren! */
    }
}



void TCSWndProc_OnRbuttondown (HWND hWindow, BOOL DoubleClick, int MouseX,
                                        int MouseY, UINT ShftCtrlKeyMask)
{
    ShowWindow (hTCSstatWindow, SW_SHOW);
    UpdateWindow(hTCSstatWindow);
}



bool TCSWndProc_OnErasebkgnd (HWND hWindow, HDC hDC)
{
RECT ClientArea;
HBRUSH hBack;

    GetClientRect (hWindow, &ClientArea);
    DPtoLP (hDC, (LPPOINT)&ClientArea.left,2);

    hBack= CreateSolidBrush (dwColorTable[TCSBackgroundColour]);
    FillRect(hTCSWindowDC, &ClientArea, hBack);
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteBrush (hBack);
    #else
     DeleteObject (hBack);
    #endif

    return false;
}



bool TCSWndProc_OnCopyClipboard ()
{
#if (JOURNALTYP == 1)
 FTNINT iErr;
 HMETAFILE hmf;
 HDC hTCSNewMetaFileDC;
 HGLOBAL hGlobalMem;
 LPMETAFILEPICT lpMfp;
 HRGN    hWindowRegion;
#elif (JOURNALTYP == 2)
 FTNINT iErr;
 HENHMETAFILE hmf, hmf1;
 ENHMETAHEADER emh ;
 HDC hTCSMetaFileDC1;
#endif


#if (JOURNALTYP == 1)
    hmf = CloseMetaFile (hTCSMetaFileDC);       /* Metafile für WM_PAINT */

    hGlobalMem= GlobalAlloc(GMEM_MOVEABLE | GMEM_SHARE, sizeof(METAFILEPICT));
    if (hGlobalMem == NULL) {
     iErr= WRN_COPYNOMEM;
     TCSGraphicError (iErr,"");
     return false;                      /* Error: OutOfMemory -> ret */
    }
    lpMfp= (LPMETAFILEPICT) GlobalLock (hGlobalMem);

    lpMfp->mm= MM_ANISOTROPIC;
    lpMfp->xExt= 0;             /* Keine Defaultgröße vorgeben */
    lpMfp->yExt= 0;             /* sonst in MM_HIMETRIC Device-Einheiten! */

    hTCSNewMetaFileDC  = CreateMetaFile (NULL);

    ScaleViewportExtEx (hTCSNewMetaFileDC, 1,1,-1,1,NULL);   // für Clipboard

    hWindowRegion= CreateRectRgn(TCSrect.left, TCSrect.top, TCSrect.right,TCSrect.bottom); // rechts,oben
    SelectClipRgn (hTCSNewMetaFileDC, hWindowRegion); // nicht eingeschlossen
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteRgn (hWindowRegion); // Resource freigeben
    #else
     DeleteObject (hWindowRegion);
    #endif

    PlayMetaFile (hTCSNewMetaFileDC, hmf);

    lpMfp->hMF= CloseMetaFile (hTCSNewMetaFileDC);

    GlobalUnlock(hGlobalMem);

    hTCSNewMetaFileDC  = CreateMetaFile (NULL); /* 16bit Windows Metafile */
    PlayMetaFile (hTCSNewMetaFileDC, hmf);      /* für neues Journalfile */
    DeleteMetaFile (hmf);                       /* alter Status Bildschirm */
    hTCSMetaFileDC = hTCSNewMetaFileDC;         /* bereit Weiterzeichnen */

    if (!OpenClipboard (hTCSWindow)) {          /* Error: Clipboard locked */
     GlobalFree (hGlobalMem);
     iErr= WRN_COPYLOCK;
     TCSGraphicError (iErr,"");
     return false;
    }
    EmptyClipboard ();
    SetClipboardData (CF_METAFILEPICT, hGlobalMem);
    CloseClipboard ();  /* Jetzt GlobalFree() NICHT mehr aufrufen */

#elif (JOURNALTYP == 2)
    hmf = CloseEnhMetaFile (hTCSMetaFileDC);    /* Metafile für WM_PAINT */
    hmf1 = CopyEnhMetaFile (hmf, NULL) ;
    if (!OpenClipboard (hTCSWindow)) {          /* Error: Clipboard locked */
     iErr= WRN_COPYLOCK;
     TCSGraphicError (iErr,"");
     return false;
    }
    EmptyClipboard () ;
    SetClipboardData (CF_ENHMETAFILE, hmf1) ;
    CloseClipboard () ;

    GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
    hTCSMetaFileDC1  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
                 _T("TCS for Windows\0Journalfile created by CopyClipboard\0"));
    SetMapMode (hTCSMetaFileDC1, MM_ANISOTROPIC);
    SetViewportExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);
    SetWindowExtEx (hTCSMetaFileDC1, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC1, TCSrect.left, TCSrect.bottom, NULL);

    SetBkMode (hTCSMetaFileDC, TRANSPARENT );
    SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);

    PlayEnhMetaFile (hTCSMetaFileDC1, hmf, &TCSrect); // neues Journal

    DeleteEnhMetaFile (hmf);                      // alter Status Bildschirm
    hTCSMetaFileDC = hTCSMetaFileDC1;           // bereit zum Weiterzeichnen

    SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
    SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #endif
    SetBkMode (hTCSMetaFileDC, TRANSPARENT );   // Metafile weitergegeben !
    SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); // CP
    SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif

#endif

    return true;
}



LRESULT CALLBACK EXPORT16 TCSWndProc(HWND hWindow, UINT Message,
                            WPARAM wParam, LPARAM lParam)
{
    switch( Message ) {
     HANDLE_MSG(hWindow, WM_PAINT, TCSWndProc_OnPaint);
     HANDLE_MSG(hWindow, WM_RBUTTONDOWN, TCSWndProc_OnRbuttondown);
     HANDLE_MSG(hWindow, WM_SIZE, TCSWndProc_OnSize);
     HANDLE_MSG(hWindow, WM_ERASEBKGND, TCSWndProc_OnErasebkgnd);
     case WM_SYSCOMMAND:
      if (wParam == TCS_WM_COPY) {
       #ifdef trace_calls
        MessageBox(NULL, "WM_SYSCOMMAND (TCS_WM_COPY)",
                       "Internal Information GRAPH2D - TCSwindowProc",
                        MB_OK | MB_ICONINFORMATION);
       #endif
       TCSWndProc_OnCopyClipboard ();
       break;
      } else {
       return DefWindowProc( hWindow, Message, wParam, lParam );
      }
     case WM_CLOSE: // Schliessen des Graphikfensters nicht zulassen! Meldung
      break;        // kann trotz Menuesperre über <ALT><F4> erzeugt werden
     case WM_ACTIVATEAPP: // Neuzeichnen wg. Fensterminimierung fremde Appl.
      UpdateWindow (hWindow);
      return 0;
     default:
      return DefWindowProc( hWindow, Message, wParam, lParam );
    }
    return 0;
}



/*
------------------- Event Handler Statusfenster ------------------------
*/



void TCSstatWndProc_OnPaint (HWND hWindow)
{
int i;
PAINTSTRUCT ps;

    BeginPaint (hWindow, &ps);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (ps.hdc, hTCSSysFont);      // Aktuellen Zeichenstatus an
    #else
     SelectObject (ps.hdc, hTCSSysFont);      // Aktuellen Zeichenstatus an
    #endif
    SetMapMode (ps.hdc, MM_TEXT);
    SetWindowOrgEx (ps.hdc, 0,TCSstatOrgY*TextLineHeight, NULL);
    for (i=0; i <= TCSstatRow; i++ )
     TextOut (ps.hdc, 0, i*TextLineHeight, TCSstatTextBuf[i],
                                          _tcslen (TCSstatTextBuf[i]));
    EndPaint( hWindow, &ps );
}



void TCSstatWndProc_OnKillfocus (HWND hWindow, HWND hNewWindow)
{
    if (TCSStatWindowAutomatic) ShowWindow (hWindow, SW_HIDE);
}



void TCSstatWndProc_OnGetminmaxinfo (HWND hWindow, MINMAXINFO FAR* lpMinMaxInfo)
/* Beschränkung User-erzeugbare Fenstergröße */
{
    lpMinMaxInfo -> ptMaxSize.x = GetSystemMetrics (SM_CXMAXIMIZED);
    lpMinMaxInfo -> ptMaxSize.y = (int) (TCS_REL_CHR_SPACE*TextLineHeight) +
                                STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
    lpMinMaxInfo -> ptMaxPosition.x = 0;
    #if !defined(__WIN32__) && !defined(_WIN32)
     lpMinMaxInfo -> ptMaxPosition.y = GetSystemMetrics (SM_CYFULLSCREEN) -
                                STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
    #else
     lpMinMaxInfo -> ptMaxPosition.y = GetSystemMetrics (SM_CYMAXIMIZED) -
                                (lpMinMaxInfo -> ptMaxSize.y);
    #endif
    lpMinMaxInfo -> ptMinTrackSize.x = GetSystemMetrics (SM_CXMINTRACK);
    lpMinMaxInfo -> ptMinTrackSize.y = GetSystemMetrics (SM_CYMINTRACK);
    lpMinMaxInfo -> ptMaxTrackSize.x = GetSystemMetrics (SM_CXMAXIMIZED);
    lpMinMaxInfo -> ptMaxTrackSize.y = STAT_ADDLINES*TextLineHeight+
                                (lpMinMaxInfo -> ptMaxSize.y);
}



void TCSstatWndProc_OnVScroll (HWND hWindow, HWND hNewWindow, WPARAM wParam,
                                                            LPARAM lParam)
{
    switch (wParam) {
     case SB_LINEUP:
      TCSstatScrollY --;
      if (TCSstatScrollY < 0) TCSstatScrollY=0;
      break;
     case SB_LINEDOWN:
      TCSstatScrollY ++;
      if (TCSstatScrollY >= STAT_MAXROWS) TCSstatScrollY=STAT_MAXROWS-1;
      break;
     case SB_PAGEUP:
      TCSstatScrollY -= STAT_PAGESIZ;
      if (TCSstatScrollY < 0) TCSstatScrollY=0;
      break;
     case SB_PAGEDOWN:
      TCSstatScrollY += STAT_PAGESIZ;
      if (TCSstatScrollY >= STAT_MAXROWS) TCSstatScrollY=STAT_MAXROWS-1;
      break;
     case SB_THUMBPOSITION:
      TCSstatScrollY= (int) lParam;
      if (TCSstatScrollY < 0) TCSstatScrollY=0;
      if (TCSstatScrollY >= STAT_MAXROWS) TCSstatScrollY=STAT_MAXROWS-1;
      InvalidateRect (hWindow, NULL, true); /* ,ClientArea, EraseFlag */
      UpdateWindow (hWindow);               /* zwingend notwendig für Win16 */
      break;
    }
    ScrollWindow (hWindow, 0, (TCSstatOrgY-TCSstatScrollY)*TextLineHeight,
                                                                NULL, NULL);
    SetScrollPos (hWindow, SB_VERT, TCSstatScrollY, true);
    TCSstatOrgY= TCSstatScrollY;
}



LRESULT CALLBACK EXPORT16 TCSstatWndProc(HWND hWindow, UINT Message,
                            WPARAM wParam, LPARAM lParam)
{
    switch( Message ) {
     HANDLE_MSG(hWindow, WM_PAINT, TCSstatWndProc_OnPaint);
     HANDLE_MSG(hWindow, WM_KILLFOCUS, TCSstatWndProc_OnKillfocus);
     HANDLE_MSG(hWindow, WM_GETMINMAXINFO, TCSstatWndProc_OnGetminmaxinfo);
     HANDLE_MSG(hWindow, WM_VSCROLL, TCSstatWndProc_OnVScroll);
     default:
      return DefWindowProc( hWindow, Message, wParam, lParam );
    }
    return 0;
}



/*
---------------------- Userroutinen: Initialisierung -------------------
*/



extern void tcslev3 (FTNINT *SysLev)

{
    *SysLev= TCSLEV3SYS;
}



#ifdef XMLSUPPORT

void XMLreadProgPar (const char * filname)
{
int ParserState;
FILE *fp;
mxml_node_t *tree;

    fp = fopen(filname, "r");
    if (fp == NULL) {
     TCSGraphicError (ERR_XMLOPEN, filname);
    } else {
      ParserState= -1; // State= idle
      mxmlSetErrorCallback ((mxml_error_cb_t)sax_error_callback);
      tree = mxmlSAXLoadFile(NULL, fp, sax_type_callback, sax_callback, &ParserState);
      fclose(fp);
    }
}

#endif    // Ende XML-Unterstützung



/*
Defaultwerte sind bereits durch Compiler initialisiert worden. Hier werden nur
die Parameter wiederhergestellt, die fuer einen erneuten Aufruf von initt nach
finitt sinnvoll sind.
*/

void PresetProgPar ()
{
    TCSDefaultLinCol= TCS_INIDEF_LINCOL;
    TCSDefaultTxtCol= TCS_INIDEF_TXTCOL;
    TCSDefaultBckCol= TCS_INIDEF_BCKCOL;
```

```

    TCSwindowIniXrelpos= TCS_INIDEF_WINPOSX;
    TCSwindowIniYrelpos= TCS_INIDEF_WINPOSY;
    TCSwindowIniXrelsiz= TCS_INIDEF_WINSIZX;
    TCSwindowIniYrelsiz= TCS_INIDEF_WINSIZY;

    TCSstatWindowIniXrelpos= TCS_INIDEF_STATPOSX;
    TCSstatWindowIniYrelpos= TCS_INIDEF_STATPOSY;
    TCSstatWindowIniXrelsiz= TCS_INIDEF_STATSIZX;
    TCSstatWindowIniYrelsiz= TCS_INIDEF_STATSIZY;

    // Fensternamen werden nur durch winlbl vorher veraendert

    // Hardcopyname und Zaehlerstand bleibt!

    // Fehlermeldungen werden bei der Variablendefinition durch den Compiler initialisiert
}



/*
Anpassung der Dateinamen an die Laufzeitumgebung
*/

void CustomizeProgPar ()
{
// Absicherung der Definition der Programmparameter
#if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN)
 #define TMPSTRLEN TCS_FILE_NAMELEN
#else
 #define TMPSTRLEN TCS_WINDOW_NAMELEN
#endif

int          iL;
char         szTmpString[TMPSTRLEN];
FTNSTRDESC   ftn_WorkString, o, n;

szTmpString[0]= '\0';
n.addr= szTmpString; // Token bei Fonts werden geloescht
n.len= TMPSTRLEN;

#ifdef XMLSUPPORT // Angabe von Dateinamen fuer Fonts bei Windows nicht moeglich
    o.addr= PROGDIRTOKEN; // Token %: loeschen
    o.len= strlen (o.addr);
    ftn_WorkString.len= TCS_FILE_NAMELEN; // Font Graphikfenster
    ftn_WorkString.addr= szTCSGraphicFont;
    o.addr= PROGDIRTOKEN; // Substring %: loeschen
    o.len= strlen (o.addr);
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );

    ftn_WorkString.addr= szTCSSysFont; // Font Statusfenster
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );


    o.addr= INIFILEXTTOKEN; // Token .% loeschen
    o.len= strlen (o.addr); // Font Statusfenster
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );

    ftn_WorkString.addr= szTCSGraphicFont; // Font Graphikfenster
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
#endif // Ende XML-Unterstützung, in *.INI und Registry keine Verwendung Token

    if (strlen(szTCSWindowName) == 0) { // '/0' durch WINLBL -> Default
        strncpy(szTCSWindowName, TCS_WINDOW_NAME, TCS_WINDOW_NAMELEN);
    }
    if (strlen(szTCSstatWindowName) == 0) {
        strncpy(szTCSstatWindowName, TCS_STATWINDOW_NAME, TCS_WINDOW_NAMELEN);
    }

    o.addr= PROGDIRTOKEN; // Substring %: vollstaendiger Programmname
    o.len= strlen (o.addr);
    #if !defined(__WIN32__) && !defined(_WIN32)   /* nicht bei DLL möglich */
     #if defined __WATCOMC__
      iL= 0;              /* Argument 0= Voller Programmname mit Directory */
      iL= igetarg ((FTNINT *) &iL, &n);
     #else
      #error "Kompilation für 16bit Windows nur mit Watcom-Compiler möglich"
     #endif
    #else   /* alternativ nur Win32: hInst=NULL: prozesserzeugende Instanz */
     iL= GetModuleFileName(NULL, n.addr, n.len);
    #endif
    if (iL <= 0) {
     n.addr[0]= (FTNCHAR) 0; /* kein Programmnamen bekannt */
    }
    ftn_WorkString.len= TCS_WINDOW_NAMELEN; // Ersatz %: im Graphikfenster
    ftn_WorkString.addr= szTCSWindowName;
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
   ftn_WorkString.addr= szTCSstatWindowName; // Ersatz %: im Statusfenster
   SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );

// Absicherung TMPSTRLEN nicht mehr benoetigt
#undef TMPSTRLEN
}




extern void winlbl (FTNSTRPAR * PloWinNam, FTNSTRPAR * StatWinNam,
                                            FTNSTRPAR *IniFilNam
                                            FTNSTRPAR_TAIL(PloWinNam)
                                            FTNSTRPAR_TAIL(StatWinNam)
                                            FTNSTRPAR_TAIL(IniFilNam)      )

{

#if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN)
 #define TMPSTRLREN TCS_FILE_NAMELEN
#else
 #define TMPSTRLREN TCS_WINDOW_NAMELEN
#endif

FTNCHARLEN  i, iL;
FTNCHAR     szTmpString[TMPSTRLREN], szTmpString1[TMPSTRLREN];
FTNCHAR *   iAt;
FTNSTRDESC  o, n, ftn_WorkString;


    iL= min(FTNSTRPARL(PloWinNam), TMPSTRLREN-1);     // Name des Grahikfensters
    _tcsncpy(szTmpString, FTNSTRPARA(PloWinNam),iL);
    szTmpString[iL]= (FTNCHAR) 0; // Fortranstring evtl. ohne \0
    iL= min (_tcslen (szTmpString), TCS_WINDOW_NAMELEN-1);
    if (iL > 0) {
     _tcsncpy( szTCSWindowName, szTmpString, iL);
     szTCSWindowName[iL]= (FTNCHAR) 0;
    }

    iL= min(FTNSTRPARL(StatWinNam), TMPSTRLREN-1);    // Name des Statusfensters
    _tcsncpy(szTmpString, FTNSTRPARA(StatWinNam), iL);
    szTmpString[iL]= (FTNCHAR) 0; // Fortranstring evtl. ohne \0
    iL= min (_tcslen (szTmpString), TCS_WINDOW_NAMELEN-1);
    if (iL > 0) {
     _tcsncpy( szTCSstatWindowName, szTmpString, iL);
     szTCSstatWindowName[iL]= (FTNCHAR) 0;
    }

    iL= min(FTNSTRPARL(IniFilNam), TMPSTRLREN-1); // Name Initialisierungsdatei
    _tcsncpy(szTmpString, FTNSTRPARA(IniFilNam), iL);
    szTmpString[iL]= (FTNCHAR) 0; // Fortranstring evtl. ohne \0

    iL= min (_tcslen (szTmpString), TCS_FILE_NAMELEN-1);

    if (iL > 0) {
     _tcsncpy( szTCSIniFile, szTmpString, iL);
     szTCSIniFile[iL]= (FTNCHAR) 0;

     iAt= _tcsstr (szTCSIniFile, _T("@")); // Section Level0?
     if (iAt != 0) {
      _tcsncpy(szTCSsect0, &iAt[1], iL); // Abspeichern
      iAt[0]= (FTNCHAR) 0; // Abschneiden von @Section0 in szTCSIniFile
     }

     ftn_WorkString.len= TCS_FILE_NAMELEN;
     ftn_WorkString.addr= szTCSIniFile;

     n.len= _tcslen (INIFILEXT);
     n.addr= INIFILEXT;
     o.len= _tcslen (INIFILEXTTOKEN);
     o.addr= INIFILEXTTOKEN;
     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                 CALLFTNSTRL(ftn_WorkString)
                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );

     n.len= TCS_FILE_NAMELEN;
     n.addr= (FTNCHAR *) &szTmpString1;
     o.len= _tcslen (PROGDIRTOKEN);
     o.addr= PROGDIRTOKEN;

     _tcsncpy (szTmpString1, szTCSIniFile, TCS_FILE_NAMELEN);
     _tcsrev (szTmpString1); // Abfrage Ende des Strings, Extension rueckwaerts!

     if (_tcsnicmp (szTmpString1, _T("GER."),4) == 0) { // Filename endet .REG?
      n.addr[0]= (FTNCHAR) 0; /* keine Directory sinnvoll -> Token loeschen */
     } else {
      #if !defined(__WIN32__) && !defined(_WIN32)   /* nicht bei DLL möglich */
       #if defined __WATCOMC__
        iL= 0;             /* Argument 0= Voller Programmname mit Directory */
        iL= igetarg ((FTNINT *) &iL, &n);
       #else
        #error "Kompilation für 16bit Windows nur mit Watcom-Compiler möglich"
       #endif
      #else   /* alternativ nur Win32: hInst=NULL: prozesserzeugende Instanz */
      iL= GetModuleFileName(NULL, n.addr, n.len);
      #endif
      if (iL>0) {
       for (i=iL-1;(n.addr[i]!= (FTNCHAR) '\\' ) || (i==0); i--);
       i++;
       if (i < n.len) n.addr[i]= (FTNCHAR) 0; /* jetzt: Programmname entfernt */
      } else {
       n.addr[0]= (FTNCHAR) 0; /* keine Directory bekannt */
      }
     }
     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                 CALLFTNSTRL(ftn_WorkString)
                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );

    }

#undef TMPSTRLREN
}



extern void initt1 (HINSTANCE *hParentInstance, HWND *hParentWindow)
{
int          nCmdShow, iX,iY, iSizeX, iSizeY;
DWORD        FirstShow;
WNDCLASS     TCSWndClass;
HMENU        SysMenu;
TCHAR        szTmpString[TCS_FILE_NAMELEN];
TEXTMETRIC   lpTM;

#if defined(__WIN32__) || defined(_WIN32) || defined (REGSUPPORT)
 DWORD       retValue;
 LPVOID      lpMsgBuf;
#endif

#if defined(REGSUPPORT)
 HKEY hSysrootKey, hRootKey,hSectionKey;
 TCHAR szRootKey[TCS_FILE_NAMELEN]= _T("Software\\"); // +IniFilename ohne Ext.
 TCHAR szSectionKey[TCS_FILE_NAMELEN];
 TCHAR szTmpString2[TCS_FILE_NAMELEN];
 DWORD dwSectionKeyLen;
 DWORD TmpStringLen, TmpStringLen2;
 DWORD i, j;
 DWORD retValue2;
#endif

#if (JOURNALTYP == 2)
 RECT   screenrect;
 int iWidthMM, iHeightMM, iWidthPixel, iHeightPixel;
#elif (JOURNALTYP == 3)
 struct xJournalEntry_typ * xJournalEntry;
#endif


    if (TCSinitialized) return;   /* Bereits initialisiert */
    TCSinitialized= true;

    PresetProgPar (); // Nach 2.Aufruf: nur Farben keine Namen wiederherstellen

    if ( _tcslen (szTCSIniFile) <= 4) { // Extension muss angegeben werden!
     _tcsncpy (szTCSIniFile, _T("TooShortInitfilename"), TCS_FILE_NAMELEN);
    }

    _tcsncpy (szTmpString, szTCSIniFile, TCS_FILE_NAMELEN);
    _tcsrev (szTmpString); // Abfrage Ende des Strings, Extension rueckwaerts!

    /*
        Falls Extension des Ini-Files .XML: XML-Parser
    */
#if defined(XMLSUPPORT)
    if (_tcsnicmp (szTmpString, _T("LMX."),4) == 0) { // Filename endet .XML?
     XMLreadProgPar (szTCSIniFile);
    } else  // endif Initialisierung ueber *.xml
#endif


    /*
        Falls Extension des Ini-Files .REG: Auswertung der Registry
    */
#if defined(REGSUPPORT)
    if (_tcsnicmp (szTmpString, _T("GER."),4) == 0) { // Filename endet .REG?
     _tcsncat (szRootKey, szTCSIniFile, _tcslen (szTCSIniFile)-4);
     for (hSysrootKey= HKEY_LOCAL_MACHINE; hSysrootKey!= NULL; ) {
      if (!RegOpenKeyEx( hSysrootKey, szRootKey, 0, KEY_READ, &hRootKey)) {
       szSectionKey[0]= (FTNCHAR) 0; // 1. Durchlauf ohne Section
       for (i = 0, retValue= false; !retValue; i++) {
        if (!RegOpenKeyEx( hRootKey, szSectionKey, 0, KEY_READ, &hSectionKey)) {
         for (j = 0, retValue2 = false; !retValue2; j++) {
          TmpStringLen= TCS_FILE_NAMELEN;         // Codewort
          TmpStringLen2= TCS_FILE_NAMELEN;        // Wert des Codewortes
          retValue2= RegEnumValue(hSectionKey, j, szTmpString, &TmpStringLen,
                            NULL, NULL, (LPBYTE) szTmpString2, &TmpStringLen2);
          if (!retValue2) StoreIni (szSectionKey,szTmpString, szTmpString2);
         }
         RegCloseKey(hSectionKey);
        }
        dwSectionKeyLen= TCS_FILE_NAMELEN;
        retValue= RegEnumKeyEx(hRootKey, i, szSectionKey, &dwSectionKeyLen,
                                                    NULL,  NULL, NULL, NULL);
       }
       RegCloseKey(hRootKey);
      }
      if (hSysrootKey == HKEY_LOCAL_MACHINE) {
       hSysrootKey= HKEY_CURRENT_USER;
      } else if (hSysrootKey == HKEY_CURRENT_USER) {
       hSysrootKey= NULL;
      }
     } // 2x: HKEY_LOCAL_MACHINE, HKEY_CURRENT_USER (ueberschreibt LOCAL_MACH.)
    } else  // endif Registryinitialisierung
#endif

    /*
        Falls Extension des Ini-Files .INI: Auswertung der Initialisierungsdatei
    */

    if (_tcsnicmp (szTmpString, _T("INI."),4) == 0) { // Filename endet .INI?
     if (_tcslen(szTCSWindowName)==0)
      GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_WINNAM,
      TCS_WINDOW_NAME, szTCSWindowName, TCS_WINDOW_NAMELEN, szTCSIniFile);
     if (_tcslen(szTCSstatWindowName)==0)
      GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_STATNAM,
      TCS_STATWINDOW_NAME,szTCSstatWindowName,TCS_WINDOW_NAMELEN,szTCSIniFile);

     GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_MAINWINNAM,
      TCS_MAINWINDOW_NAME,szTCSMainWindowName,TCS_WINDOW_NAMELEN,szTCSIniFile);

     GetPrivateProfileString(TCS_INISECT1,TCS_INIVAR_HDCNAM, TCS_HDCFILE_NAME,
                              szTCSHardcopyFile,TCS_FILE_NAMELEN,szTCSIniFile);


     GetPrivateProfileString (TCS_INISECT2,TCS_INIVAR_COPMEN,TCS_INIDEF_COPMEN,
                              szTCSMenuCopyText, STAT_MAXCOLUMNS, szTCSIniFile);
     GetPrivateProfileString (TCS_INISECT2,TCS_INIVAR_FONT,TCS_INIDEF_FONT,
                              szTCSGraphicFont, TCS_FILE_NAMELEN, szTCSIniFile);
     GetPrivateProfileString (TCS_INISECT2,TCS_INIVAR_SYSFONT,TCS_INIDEF_SYSFONT,
                              szTCSSysFont, TCS_FILE_NAMELEN, szTCSIniFile);
     GetPrivateProfileString(TCS_INISECT2,TCS_INIVAR_ICONNAM, TCS_ICONFILE_NAME,
                              szTCSIconFile,TCS_FILE_NAMELEN,szTCSIniFile);

     TCSwindowIniXrelpos= GetPrivateProfileInt (TCS_INISECT2,
                       TCS_INIVAR_WINPOSX, TCS_INIDEF_WINPOSX, szTCSIniFile);
     TCSwindowIniYrelpos= GetPrivateProfileInt (TCS_INISECT2,
                       TCS_INIVAR_WINPOSY, TCS_INIDEF_WINPOSY, szTCSIniFile);
     TCSwindowIniXrelsiz= GetPrivateProfileInt (TCS_INISECT2,
                       TCS_INIVAR_WINSIZX, TCS_INIDEF_WINSIZX, szTCSIniFile);
     TCSwindowIniYrelsiz= GetPrivateProfileInt (TCS_INISECT2,
                       TCS_INIVAR_WINSIZY, TCS_INIDEF_WINSIZY, szTCSIniFile);

     TCSstatWindowIniXrelpos= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_STATPOSX, TCS_INIDEF_STATPOSX, szTCSIniFile);
     TCSstatWindowIniYrelpos= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_STATPOSY, TCS_INIDEF_STATPOSY, szTCSIniFile);
     TCSstatWindowIniXrelsiz= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_STATSIZX, TCS_INIDEF_STATSIZX, szTCSIniFile);
     TCSstatWindowIniYrelsiz= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_STATSIZY, TCS_INIDEF_STATSIZY, szTCSIniFile);

     TCSDefaultLinCol= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_LINCOL,TCS_INIDEF_LINCOL, szTCSIniFile);
     TCSDefaultTxtCol= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_TXTCOL,TCS_INIDEF_TXTCOL, szTCSIniFile);
     TCSDefaultBckCol= GetPrivateProfileInt (TCS_INISECT2,
                        TCS_INIVAR_BCKCOL,TCS_INIDEF_BCKCOL, szTCSIniFile);


     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_HDCOPN,TCS_INIDEF_HDCOPN,
                  szTCSErrorMsg[WRN_HDCFILOPN], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_HDCFILOPN]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_HDCOPNL,TCS_INIDEF_HDCOPNL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_HDCWRT,TCS_INIDEF_HDCWRT,
                  szTCSErrorMsg[WRN_HDCFILWRT], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_HDCFILWRT]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_HDCWRTL,TCS_INIDEF_HDCWRTL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_HDCINT,TCS_INIDEF_HDCINT,
                  szTCSErrorMsg[WRN_HDCINTERN], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_HDCFILWRT]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_HDCINTL,TCS_INIDEF_HDCINTL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3, TCS_INIVAR_USR,TCS_INIDEF_USR,
                     szTCSErrorMsg[MSG_USR], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[MSG_USR]= GetPrivateProfileInt (TCS_INISECT3, TCS_INIVAR_USRL,
                     TCS_INIDEF_USRL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3, TCS_INIVAR_HDCACT,TCS_INIDEF_HDCACT,
                     szTCSErrorMsg[MSG_HDCACT], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[MSG_HDCACT]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_HDCACTL,TCS_INIDEF_HDCACTL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_USRWRN,TCS_INIDEF_USRWRN,
                  szTCSErrorMsg[WRN_USRPRESSANY],STAT_MAXCOLUMNS,szTCSIniFile);
     TCSErrorLev[WRN_USRPRESSANY]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_USRWRNL,TCS_INIDEF_USRWRNL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3, TCS_INIVAR_EXIT,TCS_INIDEF_EXIT,
                     szTCSErrorMsg[ERR_EXIT], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[ERR_EXIT]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_EXITL,TCS_INIDEF_EXITL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_COPMEM,TCS_INIDEF_COPMEM,
                  szTCSErrorMsg[WRN_COPYNOMEM], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_COPYNOMEM]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_COPMEML,TCS_INIDEF_COPMEML, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_COPLCK,TCS_INIDEF_COPLCK,
                   szTCSErrorMsg[WRN_COPYLOCK], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_COPYLOCK]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_COPLCKL,TCS_INIDEF_COPLCKL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUCREATE,TCS_INIDEF_JOUCREATE,
                   szTCSErrorMsg[WRN_JOUCREATE], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUCREATE]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_JOUCREATEL,TCS_INIDEF_JOUCREATEL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUENTRY,TCS_INIDEF_JOUENTRY,
                   szTCSErrorMsg[WRN_JOUENTRY], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUENTRY]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_JOUENTRYL,TCS_INIDEF_JOUENTRYL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUADD,TCS_INIDEF_JOUADD,
                   szTCSErrorMsg[WRN_JOUADD], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUADD]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_JOUADDL,TCS_INIDEF_JOUADDL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUCLR,TCS_INIDEF_JOUCLR,
                   szTCSErrorMsg[WRN_JOUCLR], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUCLR]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_JOUCLRL,TCS_INIDEF_JOUCLRL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_JOUUNKWN,TCS_INIDEF_JOUUNKWN,
                   szTCSErrorMsg[WRN_JOUUNKWN], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_JOUUNKWNL,TCS_INIDEF_JOUUNKWNL, szTCSIniFile);
```

```


     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_XMLPARSER,TCS_INIDEF_XMLPARSER,
                   szTCSErrorMsg[ERR_XMLPARSER], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_XMLPARSERL,TCS_INIDEF_XMLPARSERL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_XMLOPEN,TCS_INIDEF_XMLOPEN,
                   szTCSErrorMsg[ERR_XMLOPEN], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_XMLOPENL,TCS_INIDEF_XMLOPENL, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_USR2,TCS_INIDEF_USR2,
                   szTCSErrorMsg[MSG_USR2], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_USR2L,TCS_INIDEF_USR2L, szTCSIniFile);

     GetPrivateProfileString (TCS_INISECT3,TCS_INIVAR_INI2,TCS_INIDEF_INI2,
                   szTCSErrorMsg[WRN_INI2], STAT_MAXCOLUMNS, szTCSIniFile);
     TCSErrorLev[WRN_JOUUNKWN]= GetPrivateProfileInt (TCS_INISECT3,
                     TCS_INIVAR_INI2L,TCS_INIDEF_INI2L, szTCSIniFile);

    } // endif Initialisierung ueber *.ini


    CustomizeProgPar (); // Ersatz %: durch Programmverzeichnis

    /*
    Übernahme der durch den Nutzer angepassten Initialisierungsdaten
    */

    TKTRNX.iLinCol= TCSDefaultLinCol;
    TKTRNX.iTxtCol= TCSDefaultTxtCol;
    TKTRNX.iBckCol= TCSDefaultBckCol;

    /*
        Ermittlung der Instanz des Processes
    */

    hTCSInst= *hParentInstance; // In Hauptprogramm durch INITT ermittelt
    hOwnerWindow= *hParentWindow;

    if (_tcscmp(szTCSMainWindowName,_T("%:")) == 0) {
     _tcsncpy( szTCSMainWindowName,GetCommandLine(), STAT_MAXCOLUMNS);
    }

    CreateMainWindow_IfNecessary (&hTCSInst,&hOwnerWindow,szTCSMainWindowName);

    *hParentWindow= hOwnerWindow;    // Publizieren evtl. neues Handle DLL->Main

    /*
        Ermittlung allgemeiner systemspezifischer Parameter
    */

    TextLineHeight= GetSystemMetrics (SM_CYMENU); /* Höhe Menüeintrag */
    TCSCharHeight= (int)(TCS_REL_CHR_HEIGHT* (float)(HiRes(TextLineHeight)));

    TCSBackgroundColour= TKTRNX.iBckCol;

    TKTRNX.kStCol = STAT_MAXCOLUMNS;
    TKTRNX.iMouse = 3; /* werden z.Zt. bei DCURSR () ausgewertet */

    /*
        Erzeugung des Graphikfensters
    */

    TCSWndClass.style           = CS_OWNDC | CS_HREDRAW | CS_VREDRAW;
    TCSWndClass.lpfnWndProc     = TCSWndProc;
    TCSWndClass.cbClsExtra      = 0;
    TCSWndClass.cbWndExtra      = 0;
    TCSWndClass.hInstance       = hTCSInst;

    #if (defined(__WIN32__) || defined(_WIN32))
     if (_tcslen (szTCSIconFile) != 0) {
      TCSWndClass.hIcon          = LoadImage (NULL, szTCSIconFile,
                                          IMAGE_ICON,0,0,LR_LOADFROMFILE);
     } else {
      TCSWndClass.hIcon          = LoadIcon (hTCSInst, TCS_WINDOW_ICON);
                                /* Falls Icon nicht definiert->LoadIcon=NULL */
     }
    #else
     TCSWndClass.hIcon          = LoadIcon (hTCSInst, TCS_WINDOW_ICON);
    #endif

    TCSWndClass.hCursor         = LoadCursor(NULL, IDC_ARROW);
    TCSWndClass.hbrBackground   = NULL; /* Erase-Handler, Brush unnötig */
    TCSWndClass.lpszMenuName    = NULL;
    TCSWndClass.lpszClassName  = TCS_WINDOWCLASS;

     /* Register the window class. Fail: most probable UNICODE on win98 */
    if (!RegisterClass (&TCSWndClass)) {
     #if defined(__WIN32__) || defined(_WIN32)
      retValue= GetLastError(); // win32-Funktion
//    if (retValue == ERROR_CLASS_ALREADY_EXISTS) {
//     Hier bei Bedarf Fehlerbehandlung einführen
//    } else {
       FormatMessage(
         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
         NULL,
         retValue,
         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
         (LPTSTR) &lpMsgBuf,
         0,
         NULL
       );
       MessageBox (NULL, lpMsgBuf, szTCSWindowName, MB_ICONSTOP);
       LocalFree( lpMsgBuf ); // Free the buffer
//    } // Ende der Fehlerbehandlung
     #else // rudimentaere Fehlerbehandlung 16bit Windows
      MessageBox (NULL, _T("Window Class not registered"),
                                  szTCSWindowName, MB_ICONSTOP);
     #endif
     return;
    }

    if ((TCSwindowIniXrelsiz < 100) || (TCSwindowIniYrelsiz < 100) ) {
     nCmdShow= SW_SHOWNORMAL; /* Achtung, int = 2Byte bei WIN16!!! */
     iX= (int) ( ( (long int) TCSwindowIniXrelpos *
                 (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
     iY= (int) ( ( (long int) TCSwindowIniYrelpos *
                 (long int) GetSystemMetrics (SM_CYMAXIMIZED)) / 100);
     iSizeX= (int) ( ( (long int) TCSwindowIniXrelsiz *
                 (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
     iSizeY= (int) ( ( (long int) TCSwindowIniYrelsiz *
                 (long int) GetSystemMetrics (SM_CYMAXIMIZED)) / 100);
    } else {
     nCmdShow= SW_SHOWMAXIMIZED;
     iX= 0;
     iY= 0;
     iSizeX= GetSystemMetrics (SM_CXMAXIMIZED);
     iSizeY= GetSystemMetrics (SM_CYMAXIMIZED);
    }

    hTCSWindow = CreateWindow(TCS_WINDOWCLASS, szTCSWindowName,
                           WS_OVERLAPPEDWINDOW,
                           iX, iY,
                           iSizeX, iSizeY,
                           hOwnerWindow,
                           (HMENU) NULL,
                           (HINSTANCE) hTCSInst, (LPSTR) NULL);

    if (hTCSWindow == NULL) return;

    hTCSWindowDC = GetDC (hTCSWindow);

    SetWindowExtEx (hTCSWindowDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSWindowDC, TCSrect.left, TCSrect.bottom, NULL);

#if (JOURNALTYP == 1)
    hTCSMetaFileDC = CreateMetaFile (NULL); /* Memory-based 16bit Metafile */
    SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
    MoveToEx (hTCSMetaFileDC, 0, 0, NULL);  /* Cursorposition Neuzeichnen */

#elif (JOURNALTYP == 2)
    iWidthMM = GetDeviceCaps(hTCSWindowDC, HORZSIZE); // Bildschirmgroesse(mm)
    iHeightMM = GetDeviceCaps(hTCSWindowDC, VERTSIZE);
    iWidthPixel = GetDeviceCaps(hTCSWindowDC, HORZRES); // Bildschirm (Pixel)
    iHeightPixel = GetDeviceCaps(hTCSWindowDC, VERTRES);

    screenrect.left= (TCSrect.left *iWidthMM *100)/iWidthPixel; // in .01 mm
    screenrect.top= (TCSrect.top *iHeightMM *100)/iHeightPixel;
    screenrect.right= (TCSrect.right *iWidthMM *100)/iWidthPixel; // right > left!
    screenrect.bottom= (TCSrect.bottom *iHeightMM *100)/iHeightPixel; // bottom > top!

    hTCSMetaFileDC = CreateEnhMetaFile (hTCSWindowDC, NULL, &screenrect,
            _T("TCS for Windows\0Journalfile created by INITT\0" ));

    SetMapMode (hTCSMetaFileDC, MM_ANISOTROPIC);
    SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);

    SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

    MoveToEx (hTCSMetaFileDC, 0, 0, NULL);  /* Cursorposition Neuzeichnen */
#endif

    ShowWindow (hTCSWindow, nCmdShow);      /* Skalierung Viewport  */
    UpdateWindow(hTCSWindow);               /* in TCSWndProc_OnSize */

    SysMenu = GetSystemMenu (hTCSWindow, FALSE); /* Systemmenu: kein Close */
    DeleteMenu (SysMenu, 6, MF_BYPOSITION);
    AppendMenu (SysMenu,MF_STRING,TCS_WM_COPY,szTCSMenuCopyText);  /* Copy */

    TCSFontdefinition.lfHeight= TCSCharHeight; /* Höhe, Breite */
    TCSFontdefinition.lfWidth= 0;
    TCSFontdefinition.lfEscapement= 0; /* lfEscapement=lfOrientation */
    TCSFontdefinition.lfOrientation= 0;
    TCSFontdefinition.lfWeight= FW_NORMAL; /* Strichstärke */
    TCSFontdefinition.lfItalic= false;
    TCSFontdefinition.lfUnderline= false;
    TCSFontdefinition.lfStrikeOut= false;
    TCSFontdefinition.lfCharSet= ANSI_CHARSET;
    TCSFontdefinition.lfOutPrecision= OUT_TT_ONLY_PRECIS;
    TCSFontdefinition.lfClipPrecision= CLIP_DEFAULT_PRECIS;
    TCSFontdefinition.lfQuality= DRAFT_QUALITY;
    TCSFontdefinition.lfPitchAndFamily= FF_MODERN | FIXED_PITCH;
    _tcscpy (TCSFontdefinition.lfFaceName, szTCSGraphicFont);
                        /* Bevorzugter Font, keine Proportionalschrift!!! */

    hTCSFont= CreateFontIndirect (&TCSFontdefinition);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSWindowDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #else
     SelectObject (hTCSWindowDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #endif
    SetTextColor (hTCSWindowDC, dwColorTable[TKTRNX.iTxtCol]);

    GetTextMetrics (hTCSWindowDC, &lpTM);
    TKTRNX.kitalc= 0;
    TKTRNX.ksizef= 0;
    TKTRNX.khorsz= (FTNINT) ((float)LoRes((float)lpTM.tmAveCharWidth *TEK_XMAX/iSizeX) + 0.25f);
    TKTRNX.kversz= (FTNINT) ((float)LoRes((float)lpTM.tmHeight *TEK_YMAX/iSizeY)  + 0.25f);

    SetBkMode (hTCSWindowDC, TRANSPARENT );   /* Attribut statisch, durch */
    SetTextAlign (hTCSWindowDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); /* Ort: */

    hTCSPen= CreatePen (PS_SOLID, 0, dwColorTable[TKTRNX.iLinCol]);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif

    hGinCurs=LoadCursor(NULL, IDC_CROSS);
    hMouseCurs=LoadCursor(NULL, IDC_ARROW);

#if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
    #endif
    SetBkMode (hTCSMetaFileDC, TRANSPARENT );
    SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
    SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif

#elif (JOURNALTYP == 3)
    hTCSJournal= NULL;
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");

    xJournalEntry->action=  XACTION_NOOP; // Erkennung Listenanfang: Wurzelelement ohne Funktion
    xJournalEntry->i1= 0;
    xJournalEntry->i2= 0;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
    xJournalEntry->action=  XACTION_INITT;
    xJournalEntry->i1= 0;
    xJournalEntry->i2= 0;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

```

```
    /*
        Erzeugung des Statusfensters
    */

    TCSWndClass.style           = CS_HREDRAW | CS_VREDRAW;  //  CS_OWNDC |
    TCSWndClass.lpfnWndProc     = TCSstatWndProc;
    TCSWndClass.hInstance       = hTCSInst;
    TCSWndClass.hIcon           = NULL;
    TCSWndClass.hCursor         = LoadCursor(NULL, IDC_ARROW);
    #if !defined(__WIN32__) && !defined(_WIN32)
     TCSWndClass.hbrBackground  = (HBRUSH) GetStockBrush(WHITE_BRUSH);
    #else
     TCSWndClass.hbrBackground  = GetStockObject(WHITE_BRUSH);
    #endif
    TCSWndClass.lpszMenuName    = NULL;
    TCSWndClass.lpszClassName   = TCS_STAT_WINDOWCLASS;

    if (!RegisterClass (&TCSWndClass)) {
     #if defined(__WIN32__) || defined(_WIN32)
      retValue= GetLastError(); // win32-Funktion
//    if (retValue == ERROR_CLASS_ALREADY_EXISTS) {
//     Hier bei Bedarf Fehlerbehandlung einführen
//    } else {
       FormatMessage(
         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
         NULL,
         retValue,
         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
         (LPTSTR) &lpMsgBuf,
         0,
         NULL
       );
       MessageBox (NULL, lpMsgBuf, szTCSWindowName, MB_ICONSTOP);
       LocalFree( lpMsgBuf ); // Free the buffer
//    } // Ende der Fehlerbehandlung
     #else // rudimentaere Fehlerbehandlung 16bit Windows
      MessageBox (NULL, _T("Window Class not registered"),
                                    szTCSWindowName, MB_ICONSTOP);
     #endif
     return;
    }

    if ((TCSstatWindowIniXrelsiz < 100) || (TCSstatWindowIniYrelsiz < 100) ) {
     FirstShow= WS_OVERLAPPED | WS_SIZEBOX | WS_VSCROLL; // WIN16: int*2 !
     iX= (int) ( ( (long int) TCSstatWindowIniXrelpos *
                   (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
     iY= (int) ( ( (long int) TCSstatWindowIniYrelpos *
                   (long int) GetSystemMetrics (SM_CYMAXIMIZED)) / 100);
     iSizeX= (int) ( ( (long int) TCSstatWindowIniXrelsiz *
                       (long int) GetSystemMetrics (SM_CXMAXIMIZED)) / 100);
     iSizeY= (int) ( ( (long int)  TCSstatWindowIniYrelsiz *
                       (long int) GetSystemMetrics (SM_CYMAXIMIZED) ) / 100);
    } else {
     FirstShow= WS_OVERLAPPED | WS_SIZEBOX | WS_VSCROLL | WS_MAXIMIZE;
     iX= 0;
     iY = GetSystemMetrics (SM_CYMAXIMIZED) -
                    #if defined(__WIN32__) || defined(_WIN32)
                                 (int) (TCS_REL_CHR_SPACE*TextLineHeight) -
                    #endif
                             STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
     iSizeX= GetSystemMetrics (SM_CXMAXIMIZED);
     iSizeY= (int) (TCS_REL_CHR_SPACE*TextLineHeight) +
                             STAT_MINLINES*GetSystemMetrics (SM_CYMINTRACK);
    }

    hTCSstatWindow = CreateWindow(TCS_STAT_WINDOWCLASS, szTCSstatWindowName,
                        FirstShow,
                        iX, iY,
                        iSizeX, iSizeY,
                        (HWND) hTCSWindow, (HMENU) NULL,
                        (HINSTANCE) hTCSInst, (LPSTR) NULL);

    if (hTCSstatWindow == NULL) return;

    #ifdef STAT_WINDOW_PRIVATE
     hTCSstatWindowDC = GetDC (hTCSstatWindow);
    #endif

    TCSFontdefinition.lfHeight= TextLineHeight; /* Buchstabenhöhe */
    _tcscpy (TCSFontdefinition.lfFaceName, szTCSSysFont);
                        /* Bevorzugter Font, keine Proportionalschrift!!! */
    hTCSSysFont= CreateFontIndirect (&TCSFontdefinition);

    TCSFontdefinition.lfHeight= TCSCharHeight; /* Wiederherstellung Graphikzeichensatz */
    _tcscpy (TCSFontdefinition.lfFaceName, szTCSGraphicFont);


    TCSStatWindowAutomatic = true;
    TCSstatCursorPosY= 0;
    TCSstatScrollY= 0;
    TCSstatRow= -1;
    TCSstatOrgY= TCSstatScrollY;
    SetScrollRange (hTCSstatWindow, SB_VERT, 0,STAT_MAXROWS-1, true);
    SetScrollPos (hTCSstatWindow, SB_VERT, TCSstatScrollY, true);

    ShowWindow (hTCSstatWindow, SW_HIDE);

    ClippingNotActive= true;

    return;
}



extern void finitt ()
{
// FTNINT iErr;
#if (JOURNALTYP == 1)
 HMETAFILE hmf;
#elif (JOURNALTYP == 2)
 HENHMETAFILE hmf;
#elif (JOURNALTYP == 3)
 struct xJournalEntry_typ * xJournalEntry;
#endif


    if (!TCSinitialized) return; /* Graphiksystem nicht initialisiert */

    TCSGraphicError (ERR_EXIT,"");  /* TCSinitialized verhindert Rekursion*/

    TCSinitialized= false;           /* Ab jetzt nicht mehr funktionsfähig */

    ReleaseDC (hTCSWindow, hTCSWindowDC);
    DestroyWindow (hTCSWindow);
    UnregisterClass (TCS_WINDOWCLASS, hTCSInst);

#if (JOURNALTYP == 1)
    hmf = CloseMetaFile (hTCSMetaFileDC);
    DeleteMetaFile (hmf);
#elif (JOURNALTYP == 2)
    hmf = CloseEnhMetaFile (hTCSMetaFileDC);
    DeleteEnhMetaFile (hmf);
#elif (JOURNALTYP == 3)
    SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, hTCSJournal,
           xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
    hTCSJournal= NULL;
#endif

    #ifdef STAT_WINDOW_PRIVATE
     ReleaseDC (hTCSstatWindow, hTCSstatWindowDC);
    #endif
    DestroyWindow (hTCSstatWindow);
    UnregisterClass (TCS_STAT_WINDOWCLASS, hTCSInst);

    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteFont (hTCSFont);
     DeleteFont (hTCSSysFont);
     DeletePen (hTCSPen);
    #else
     DeleteObject (hTCSFont);
     DeleteObject (hTCSSysFont);
     DeleteObject (hTCSPen);
    #endif

    #if defined(__WATCOMC__) && defined(__SW_BW)
     _dwShutDown();          // Shutdown Watcom Default Window System
    #endif

    if (TCSErrorLev[ERR_EXIT] >= 10) exit (EXIT_SUCCESS); // Programmende
    return; // Bei Fehlerlevel <10 zurück zum Hauptprogramm
}



/*
---------------------- Userroutinen: Zeichnen -------------------
*/



extern void swind1 (FTNINT *ix1,FTNINT *iy1,FTNINT *ix2,FTNINT *iy2)
{
    ClippingNotActive = (*ix1==0) && (*iy1==0) &&
                                          (*ix2==TEK_XMAX) && (*iy2==TEK_YMAX);
    /* Berechnung BOOL zur Wahrung der Programmstruktur der DOS-Version */
}



extern void erase (void)
{
#if (JOURNALTYP == 1)
 HMETAFILE hmf;
 HRGN       hWindowRegion;
 HBRUSH     hBack;
#elif (JOURNALTYP == 2)
 HENHMETAFILE   hmf;
 ENHMETAHEADER emh ;
#elif (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

#if (JOURNALTYP == 1)
     hmf = CloseMetaFile (hTCSMetaFileDC);   /* Cursor, Farben unverändert! */
     DeleteMetaFile (hmf);                   /* alter Status Bildschirm */
     hTCSMetaFileDC  = CreateMetaFile (NULL);/* für neues Journalfile */
     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

     hBack= CreateSolidBrush (dwColorTable[TKTRNX.iBckCol]);
     hWindowRegion= CreateRectRgn (TCSrect.left, TCSrect.top, TCSrect.right,TCSrect.bottom); // rechts,oben
     FillRgn (hTCSMetaFileDC, hWindowRegion, hBack);         // nicht eingeschlossen
     #if !defined(__WIN32__) && !defined(_WIN32)
      DeleteBrush (hBack);
      DeleteRgn (hWindowRegion);                   /* Resourcen freigeben */
      SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
     #else
      DeleteObject (hBack);
      DeleteObject (hWindowRegion);
      SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
     #endif

     SetBkMode (hTCSMetaFileDC, TRANSPARENT );
     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
     #else
      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
     #endif

     MoveToEx (hTCSMetaFileDC, HiRes(TKTRNX.kBeamX), HiRes(TKTRNX.kBeamY), NULL);

#elif (JOURNALTYP == 2)
     hmf = CloseEnhMetaFile (hTCSMetaFileDC);
     GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
     DeleteEnhMetaFile (hmf);                        // alter Status Bildschirm

     hTCSMetaFileDC  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
                       _T("TCS for Windows\0Journalfile created by Erase\0\0"));

     SetMapMode (hTCSMetaFileDC, MM_ANISOTROPIC);
     SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
     SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
     SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
     SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectFont (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
     #else
      SelectObject (hTCSMetaFileDC, hTCSFont);      // Aktuellen Zeichenstatus an
     #endif
     SetBkMode (hTCSMetaFileDC, TRANSPARENT );
     SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP);
     SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
     #else
      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
     #endif

     MoveToEx (hTCSMetaFileDC, HiRes(TKTRNX.kBeamX),HiRes(TKTRNX.kBeamY), NULL);

#elif (JOURNALTYP == 3)
     SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, hTCSJournal,
           xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
     hTCSJournal= NULL;

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_NOOP;
     xJournalEntry->i1= 0;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_LINCOL;
     xJournalEntry->i1= TKTRNX.iLinCol;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_TXTCOL;
     xJournalEntry->i1= TKTRNX.iTxtCol;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_BCKCOL;
     xJournalEntry->i1= TKTRNX.iBckCol;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_ERASE;
     xJournalEntry->i1= 0;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

    TCSBackgroundColour=TKTRNX.iBckCol; /* Jetzt in ERASE-Handler wirksam */

    InvalidateRect (hTCSWindow, NULL, true); /* ,ClientArea, EraseFlag */
    UpdateWindow (hTCSWindow); /* 16bit Rechner: gegen Irritation Anwender */

}



extern void movabs (FTNINT *ix,FTNINT *iy)
{
int ixx, iyy; /* Erzwingt Typangleichung Windows-GDI / Fortran */

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
    if (PointInWindow (*ix, *iy)) {
     ixx= HiRes(*ix); iyy= HiRes(*iy);
     MoveToEx (hTCSWindowDC, ixx, iyy, NULL);

#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     MoveToEx (hTCSMetaFileDC, ixx, iyy, NULL);
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= *ix;
     xJournalEntry->i2= *iy;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif
    }
}



extern void drwabs (FTNINT *ix,FTNINT *iy)
{
FTNINT iXClip, iYClip;
int ixx, iyy;

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
     ixx= HiRes(iXClip); iyy= HiRes(iYClip);
     MoveToEx (hTCSWindowDC, ixx,iyy, NULL);
#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     MoveToEx (hTCSMetaFileDC, ixx,iyy, NULL);
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= iXClip;
     xJournalEntry->i2= iYClip;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

     ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip,&iYClip);
     ixx= HiRes(iXClip); iyy= HiRes(iYClip);    /* geclippter Endpunkt */
     LineTo (hTCSWindowDC, ixx,iyy);      /* Endpunkt nicht mitgezeichnet! */
     SetPixel (hTCSWindowDC,ixx,iyy,dwColorTable[TKTRNX.iLinCol]);

#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     LineTo (hTCSMetaFileDC, ixx,iyy);
     SetPixel (hTCSMetaFileDC,ixx,iyy, dwColorTable[TKTRNX.iLinCol]);
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_DRWABS;
     xJournalEntry->i1= iXClip;
     xJournalEntry->i2= iYClip;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= *ix;
     xJournalEntry->i2= *iy;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

    }
    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;

}



extern void dshabs (FTNINT *ix,FTNINT *iy, FTNINT *iMask)
{
HPEN     hPenDash;
FTNINT iXClip, iYClip;
int     iMaskIndex, ixx, iyy;

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    if (*iMask < 0) {     /* Verhindern eines Access-Errors bei Integermaskenübergabe */
     iMaskIndex= 0;
    } else if (*iMask > MAX_PENSTYLE_INDEX) {
     iMaskIndex= 1;      /* Style: dotted */
    } else {
     iMaskIndex= *iMask;
    }

    if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
     ixx= HiRes(iXClip); iyy= HiRes(iYClip);
     MoveToEx (hTCSWindowDC, ixx,iyy, NULL);

#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     MoveToEx (hTCSMetaFileDC, ixx,iyy, NULL);
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= iXClip;
     xJournalEntry->i2= iYClip;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

     ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip,&iYClip);
     ixx= HiRes(iXClip); iyy= HiRes(iYClip);      /* geclippter Endpunkt */

     hPenDash= CreatePen (dwPenStyle[iMaskIndex], 0, dwColorTable[TKTRNX.iLinCol]);
     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectPen (hTCSWindowDC, hPenDash); // 16bit: Makro aus windowsx.h
     #else
      SelectObject (hTCSWindowDC, hPenDash); // 32bit: GDI Standardaufruf
     #endif
     LineTo (hTCSWindowDC, ixx,iyy);   /* Ohne Endpunkt bei Dash o.k! */
     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
     #else
      SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
     #endif

```

```
#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectPen (hTCSMetaFileDC, hPenDash); // 16bit: Makro aus windowsx.h
     #else
      SelectObject (hTCSMetaFileDC, hPenDash); // 32bit: GDI Standardaufruf
     #endif
     LineTo (hTCSMetaFileDC, ixx,iyy);
     #if !defined(__WIN32__) && !defined(_WIN32)
      SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
     #else
      SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
     #endif
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_DSHSTYLE;
     xJournalEntry->i1= iMaskIndex;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_DSHABS;
     xJournalEntry->i1= iXClip;
     xJournalEntry->i2= iYClip;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= *ix;
     xJournalEntry->i2= *iy;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

     #if !defined(__WIN32__) && !defined(_WIN32)
      DeletePen (hPenDash);
     #else
      DeleteObject (hPenDash);
     #endif

    }
    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
}



extern void pntabs (FTNINT *ix,FTNINT *iy)
{
int     ixx, iyy; /* Erzwingt Typangleichung Windows-GDI / Fortran */

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
    if (PointInWindow (*ix, *iy)) {
     ixx= HiRes(*ix); iyy= HiRes(*iy);
     SetPixel (hTCSWindowDC,ixx,iyy, dwColorTable[TKTRNX.iLinCol]);

#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     SetPixel (hTCSMetaFileDC,ixx,iyy,dwColorTable[TKTRNX.iLinCol]);
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_PNTABS;
     xJournalEntry->i1= *ix;
     xJournalEntry->i2= *iy;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

    }
}



extern void bckcol (FTNINT *iCol)
{

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.iBckCol= min(abs(*iCol),MAX_COLOR_INDEX);

#if (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
    xJournalEntry->action=  XACTION_BCKCOL;
    xJournalEntry->i1= TKTRNX.iBckCol;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

}



extern void lincol (FTNINT *iCol)
{

HPEN    hPenOld;

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.iLinCol= min(abs(*iCol),MAX_COLOR_INDEX);
    hTCSPen= CreatePen (PS_SOLID, 0, dwColorTable[TKTRNX.iLinCol]);
    #if !defined(__WIN32__) && !defined(_WIN32)
     hPenOld= SelectPen (hTCSWindowDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     hPenOld= SelectObject (hTCSWindowDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif

#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif
#elif (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
    xJournalEntry->action=  XACTION_LINCOL;
    xJournalEntry->i1= TKTRNX.iLinCol;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

    #if !defined(__WIN32__) && !defined(_WIN32)
     DeletePen (hPenOld);
    #else
     DeleteObject (hPenOld);
    #endif

}




extern void txtcol (FTNINT *iCol)
{

#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.iTxtCol= min(abs(*iCol),MAX_COLOR_INDEX);
    SetTextColor (hTCSWindowDC, dwColorTable[TKTRNX.iTxtCol]);
#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
    SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
#elif (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
    xJournalEntry->action=  XACTION_TXTCOL;
    xJournalEntry->i1= TKTRNX.iTxtCol;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

}



extern void DefaultColour (void)
{
    TKTRNX.iLinCol= TCSDefaultLinCol;
    TKTRNX.iTxtCol= TCSDefaultTxtCol;
    TKTRNX.iBckCol= TCSDefaultBckCol;

    lincol (&TKTRNX.iLinCol);
    txtcol (&TKTRNX.iTxtCol);
    bckcol (&TKTRNX.iBckCol);
}



```

```
/*
---------------------- Userroutinen: Graphiktext -----------------------
*/



extern void outgtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
{
int iL;
SIZE Size;
POINT CPpos;

#if (JOURNALTYP == 3)
 int i;
 struct xJournalEntry_typ    * xJournalEntry;
#endif

#ifdef extended_error_handling
 HDC         hdc;
 LPVOID      lpMsgBuf;
#endif


    if (FTNSTRPARA(ftn_string)[0] == (FTNCHAR) 0 ) return; // Leerstring char(0)

    iL= 1; // Stringbeginn bei 0 -> Dec Laenge
    while ( (FTNSTRPARA(ftn_string)[iL-1] != (FTNCHAR) 0) &&  // c-String bis \0
                    (iL < FTNSTRPARL(ftn_string)) ) iL++;  // oder Ftn-String
    if (FTNSTRPARA(ftn_string)[iL-1] == (FTNCHAR) 0 ) iL--;   // cString ohne \0


    #ifdef extended_error_handling
     if (GetTextExtentPoint (hTCSWindowDC, FTNSTRPARA(ftn_string),iL,&Size) == 0 ){
      hdc = CreateIC (_T ("DISPLAY"), NULL, NULL, NULL);
      #if !defined(__WIN32__) && !defined(_WIN32)
       SelectFont (hdc, hTCSFont);      // Aktuellen Zeichenstatus an
      #else
       SelectObject (hdc, hTCSFont);      // Aktuellen Zeichenstatus an
      #endif
      GetTextExtentPoint (hdc, FTNSTRPARA(ftn_string),iL,&Size);
      DeleteDC (hdc);

      FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        GetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
        (LPTSTR) &lpMsgBuf,
        0,
        NULL
      );
      MessageBox( NULL, lpMsgBuf,
                        _T("Internal Error GRAPH2D - subroutine _OUTGTEXT"),
                                                    MB_OK|MB_ICONINFORMATION );
      LocalFree( lpMsgBuf ); // Free the buffer
     }
    #else
     #if !defined(__WIN32__) && !defined(_WIN32)
      GetTextExtentPoint (hTCSWindowDC, FTNSTRPARA(ftn_string),iL,&Size);
     #else
      GetTextExtentPoint32 (hTCSWindowDC, FTNSTRPARA(ftn_string),iL,&Size);
     #endif
    #endif

    if (PointInWindow (TKTRNX.kBeamX+LoRes(Size.cx),
                                            TKTRNX.kBeamY+LoRes(Size.cy))) {
     MoveToEx (hTCSWindowDC,HiRes(TKTRNX.kBeamX),HiRes(TKTRNX.kBeamY),NULL);
     TextOut (hTCSWindowDC, 0,0,FTNSTRPARA(ftn_string), iL);

#if ((JOURNALTYP == 1) || (JOURNALTYP == 2))
     MoveToEx (hTCSMetaFileDC,HiRes(TKTRNX.kBeamX),HiRes(TKTRNX.kBeamY),NULL);
     TextOut (hTCSMetaFileDC, 0,0, FTNSTRPARA(ftn_string), iL);
#elif (JOURNALTYP == 3)
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= TKTRNX.kBeamX;
     xJournalEntry->i2= TKTRNX.kBeamY;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     xJournalEntry->action=  XACTION_GTEXT;
     xJournalEntry->i1= (FTNINT) iL;
     xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[0];
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)

     i= 1;
     while (i < iL) {
      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
      xJournalEntry->action=  XACTION_ASCII;
      xJournalEntry->i1= (FTNINT) FTNSTRPARA(ftn_string)[i++];
      if ( i<iL ) xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[i++];
      SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
     }
#endif

     GetCurrentPositionEx (hTCSWindowDC, &CPpos); /* Update Beam */
     TKTRNX.kBeamX= LoRes(CPpos.x); TKTRNX.kBeamY= LoRes(CPpos.y);

#if (JOURNALTYP == 3) // Bei Metafiles ist auch nach Neuskalierung CP i.O.
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= TKTRNX.kBeamX;
     xJournalEntry->i2= TKTRNX.kBeamY;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif

    }
}



extern void italic (void)
{
HFONT   hOldFont;
#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.kitalc = 1;

    TCSFontdefinition.lfItalic= true;
    hTCSFont= CreateFontIndirect (&TCSFontdefinition);
    #if !defined(__WIN32__) && !defined(_WIN32)
     hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
    #else
     hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
    #endif
#if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);
    #endif
#elif (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteFont (hOldFont);
    #else
     DeleteObject (hOldFont);
    #endif
}



extern void italir (void)
{
HFONT   hOldFont;
#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.kitalc = 0;

    TCSFontdefinition.lfItalic= false;
    hTCSFont= CreateFontIndirect (&TCSFontdefinition);
    #if !defined(__WIN32__) && !defined(_WIN32)
     hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
    #else
     hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
    #endif
#if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);
    #endif
#elif (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteFont (hOldFont);
    #else
     DeleteObject (hOldFont);
    #endif
}



extern void dblsiz (void)
{
HFONT   hOldFont;
#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.ksizef = 1;
    TKTRNX.khomey = TEK_YMAX - 3.0f*TKTRNX.kversz;

    TCSFontdefinition.lfHeight= 2* TCSCharHeight;
    TCSFontdefinition.lfWidth= 0;
    hTCSFont= CreateFontIndirect (&TCSFontdefinition);
    #if !defined(__WIN32__) && !defined(_WIN32)
     hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
    #else
     hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
    #endif
#if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);
    #endif
#elif (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteFont (hOldFont);
    #else
     DeleteObject (hOldFont);
    #endif
}



extern void nrmsiz (void)
{
HFONT   hOldFont;
#if (JOURNALTYP == 3)
 struct xJournalEntry_typ    * xJournalEntry;
#endif

    TKTRNX.ksizef = 0;
    TKTRNX.khomey = TEK_YMAX - 1.5f*TKTRNX.kversz;

    TCSFontdefinition.lfHeight= TCSCharHeight;
    TCSFontdefinition.lfWidth= 0;
    hTCSFont= CreateFontIndirect (&TCSFontdefinition);
    #if !defined(__WIN32__) && !defined(_WIN32)
     hOldFont= SelectFont (hTCSWindowDC, hTCSFont);
    #else
     hOldFont= SelectObject (hTCSWindowDC, hTCSFont);
    #endif
#if ( (JOURNALTYP == 1) || (JOURNALTYP == 2) )
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);
    #endif
#elif (JOURNALTYP == 3)
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, hTCSJournal, xJournalEntry, previous, next)
#endif
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteFont (hOldFont);
    #else
     DeleteObject (hOldFont);
    #endif
}



extern void csize (FTNINT *ix,FTNINT *iy)
{
TEXTMETRIC  lpTM;

#ifdef extended_error_handling
 HDC         hdc;
 LPVOID      lpMsgBuf;
#endif

    #ifdef extended_error_handling
     if (GetTextMetrics (hTCSWindowDC, &lpTM)== 0) {
      /* WATCOM ohne Default-Windowsystem(auch bei Consolenanwendungen):
         evtl. kein Message-Loop vorhanden.
         Workaround: Abfrageschleife in MessageBox                        */

      hdc = CreateIC (_T ("DISPLAY"), NULL, NULL, NULL);
      #if !defined(__WIN32__) && !defined(_WIN32)
       SelectFont (hdc, hTCSFont);
      #else
       SelectObject (hdc, hTCSFont);
      #endif
      GetTextMetrics (hdc, &lpTM);
      DeleteDC (hdc);

      FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        GetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
        (LPTSTR) &lpMsgBuf,
        0,
        NULL
      );
      MessageBox( NULL, lpMsgBuf, "Internal Error GRAPH2D - subroutine CSIZE",
                                              MB_OK|MB_ICONINFORMATION );
      LocalFree( lpMsgBuf ); // Free the buffer
     }
    #else
     GetTextMetrics (hTCSWindowDC, &lpTM);
    #endif
    *ix= (int) ((float)LoRes((float)lpTM.tmAveCharWidth) + 0.25f);
    *iy= (int) ((float)LoRes((float)lpTM.tmHeight)  + 0.25f);

}




/*
---------------------- Userroutinen: Graphic Input----------------------
*/



extern void tinput (FTNINT *ic)
{
MSG msg;         /* Message information */
TCHAR iChar;
HWND hAktWindowInThread;

    if (!TCSinitialized) return;            /* Aufhängen vermeiden */
    TCSStatWindowAutomatic = false;          /* Meldungen lesbar */
    iChar= (TCHAR) 0;
    hAktWindowInThread= GetFocus(); // Fuer Texteingabe eigene Applikation
    while (iChar == (TCHAR) 0) { // Messageschleife jetzt hier -> Usereingabe
     SetFocus (hTCSWindow);          // Kein Zugang Elternfenster (Aufhängen!)
     #ifdef extended_error_handling
      if (GetMessage (&msg, NULL, WM_NULL, WM_USER) == -1) {
       MessageBox(NULL, "GetMessage failed in Mesageloop of Graphic Window",
                        "Internal Information GRAPH2D - Subroutine TINPUT",
                        MB_OK | MB_ICONINFORMATION);
      }
     #else
      GetMessage (&msg, NULL, WM_NULL, WM_USER); // Achtung wg. win7 nicht 0,0)
     #endif
     if ((msg.hwnd != hTCSWindow) && (msg.hwnd != hTCSstatWindow) ) {
      switch (msg.message) {
       case WM_NCLBUTTONDOWN:   /* Fensterbefehle der Elternfenster zulassen */
       case WM_NCLBUTTONUP:
       case WM_NCLBUTTONDBLCLK:
```

```
  case WM_SYSKEYDOWN:
  case WM_SYSKEYUP:
  case WM_SYSCOMMAND:
   DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
   break;
  case WM_PAINT:
   UpdateWindow( msg.hwnd);
   break;
  default:
   SetFocus (hTCSWindow);
   UpdateWindow (hTCSWindow);
 }
} else if (msg.hwnd == hTCSstatWindow) { /* Meldungen Statusfenster */
 switch (msg.message) {
  case WM_NCLBUTTONDOWN:    /* Scrollen und Verschieben zulassen */
  case WM_NCLBUTTONUP:
  case WM_NCLBUTTONDBLCLK:
  case WM_VSCROLL:
   DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
   break;
  case WM_PAINT:
    TCSstatWndProc_OnPaint (hTCSstatWindow);
    break;
  case WM_LBUTTONDOWN:
   iChar= (FTNINT) 27;    /* Verlassen PRESSANY durch Statusfenster */
   break;
 }
} else { /* eigene Meldungen des Graphikfensters */
 switch (msg.message) {
  case WM_PAINT:
   TCSWndProc_OnPaint (msg.hwnd);
   break;
  case WM_RBUTTONDOWN:      /* Auf Wunsch Statusfenster sichtbar */
   ShowWindow (hTCSstatWindow, SW_SHOWNA);
   UpdateWindow(hTCSstatWindow);
   SetFocus (hTCSWindow);
   UpdateWindow (hTCSWindow);
   break;
  case WM_LBUTTONDOWN:
   ShowWindow (hTCSstatWindow, SW_HIDE);
   break;
  case WM_LBUTTONUP:
  case WM_MBUTTONUP:
  case WM_RBUTTONUP:
  case WM_MBUTTONDOWN:
  case WM_LBUTTONDBLCLK:
  case WM_RBUTTONDBLCLK:
  case WM_MBUTTONDBLCLK:
   SetFocus (hTCSWindow);
   UpdateWindow (hTCSWindow);
   break;
  case WM_KEYDOWN:          /* Hardwareanpassung, dann WM_CHAR */
  case WM_KEYUP:
   TranslateMessage (&msg);
   break;
  case WM_CHAR:             /* nach WM_KEYDOWN jetzt ASCII */
   iChar= (TCHAR) msg.wParam;
   break;
  case WM_KILLFOCUS:
   TCSStatWindowAutomatic= true; /* Statusfenster unsichtbar */
   ShowWindow (hTCSstatWindow, SW_HIDE); /* jetzt DefWindowProc */
   UpdateWindow (hTCSstatWindow);
  case WM_NCLBUTTONDOWN:
  case WM_NCLBUTTONUP:
  case WM_NCLBUTTONDBLCLK:
  case WM_SYSKEYDOWN:       /* Uebersetzt in WM_SYSCOMMAND */
  case WM_SYSKEYUP:
   DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
   break;
  case WM_QUIT:
   #ifdef trace_calls
    MessageBox(NULL, "WM_QUIT Graphic Window",
                    "Internal Information GRAPH2D - Subroutine TINPUT",
                    MB_OK | MB_ICONINFORMATION);
   #endif
  case WM_SYSCOMMAND:       /* und nach WM_SYSKEYDOWN Befehlsauswertung */
   switch (msg.wParam) {
    case SC_CLOSE:
     iChar= (FTNINT) 27;    /* <ALT><F4> -> ESC */
     break;
    case TCS_WM_COPY:
     #ifdef trace_calls
      MessageBox(NULL, "WM_SYSCOMMAND (TCS_WM_COPY)",
                  "Internal Information GRAPH2D - Subroutine TINPUT",
                   MB_OK | MB_ICONINFORMATION);
     #endif
     TCSWndProc_OnCopyClipboard ();
```

```
        break;
       default:
        DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam);
        break;
      } /* Systembefehle */
    } /* Window-Messageauswertung */
   } /* Meldungen des Graphikfensters */
  } /* Ende Eingabeschleife */
  *ic= (FTNINT) iChar;
  TCSStatWindowAutomatic= true;
  ShowWindow (hTCSstatWindow, SW_HIDE); /* Statusfenster unsichtbar */
  if (hAktWindowInThread != NULL) SetFocus (hAktWindowInThread);
  return;
}




extern void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy)
{
MSG msg;         /* Message information */
TCHAR iButton, iKey;

#if defined(__WIN32__) || defined(_WIN32)
 POINT MousePos;
#endif

  if (!TCSinitialized) return;              /* Aufhängen vermeiden */
  TCSStatWindowAutomatic = false;            /* Meldungen lesbar */

  InvalidateRect (hTCSWindow, NULL, true); /* ,ClientArea, EraseFlag */
  UpdateWindow (hTCSWindow); /* Notwendig bei OnPaint mit Journaltyp=3 */

  iButton= (TCHAR) 0; iKey= (TCHAR) 0;

  /* Setzen der Maus auf die alte GinCursor Position */

  #if defined(__WIN32__) || defined(_WIN32)
   MousePos.x= HiRes(TCSGinCurPos.x); MousePos.y= HiRes(TCSGinCurPos.y);
   LPtoDP (hTCSWindowDC, (LPPOINT)&MousePos, 1);
   MapWindowPoints(hTCSWindow, HWND_DESKTOP, (LPPOINT)&MousePos, 1);
   MousePos.x=  MousePos.x* MOUSE_XMAX / GetSystemMetrics (SM_CXSCREEN);
   MousePos.y=  MousePos.y* MOUSE_YMAX / GetSystemMetrics (SM_CYSCREEN);
   mouse_event(MOUSEEVENTF_MOVE | MOUSEEVENTF_ABSOLUTE,
                                    MousePos.x,MousePos.y, 0, 0);
  #endif

  SetCursor(hGinCurs);       /* WM_SETCURSOR wird ab hier nicht erzeugt! */
  while (iButton == (TCHAR) 0) {  /* Messageschleife jetzt hier  */
   SetFocus (hTCSWindow);     /* Kein Zugang Elternfenster (Aufhängen!) */
   GetMessage (&msg, NULL, WM_NULL, WM_USER); // Achtung wg. win7 nicht 0,0)
   if (msg.hwnd == hTCSstatWindow) { /* Statusfenster stört -> unsichtbar */
    switch (msg.message) {
     case WM_MOUSEMOVE:                  /* falls Cursor über Client-Area */
      TCSStatWindowAutomatic= true;
      ShowWindow (hTCSstatWindow, SW_HIDE);
     case WM_NCMOUSEMOVE:             /* Cursor ueber Titelleiste -> Pfeil */
      SetCursor (hMouseCurs);
      break;
    }
   }             /* Statuszeile und Scrollbar können noch angewählt werden */
   if (msg.hwnd != hTCSWindow) {
    switch (msg.message) {
     case WM_NCLBUTTONDOWN:    /* Fensterbefehle der Elternfenster zulassen */
     case WM_NCLBUTTONUP:
     case WM_NCLBUTTONDBLCLK:
     case WM_SYSKEYDOWN:
     case WM_SYSKEYUP:
     case WM_SYSCOMMAND:
      DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
      break;
     case WM_PAINT:
      if (msg.hwnd == hTCSstatWindow) {
       TCSstatWndProc_OnPaint (hTCSstatWindow);
      } else {
       UpdateWindow( msg.hwnd);
      }
      break;
     default:
      SetFocus (hTCSWindow);
      UpdateWindow (hTCSWindow);
    }
   } else { /* eigene Meldungen des Graphikfensters */
    switch (msg.message) {
     case WM_PAINT:
      TCSWndProc_OnPaint (msg.hwnd);
      break;
```

```
       case WM_NCMOUSEMOVE:    /* Cursor ueber Titelleiste -> Pfeil */
        SetCursor (hMouseCurs);
        break;
       case WM_MOUSEMOVE:     /* GinCursor evtl. von Titelleiste zurück */
        SetCursor (hGinCurs);
        iKey= (TCHAR) 0;      /* Tastenbetätigung außerhalb Graphikfenster */
        break;
       case WM_NCLBUTTONDOWN: /* Titelleiste kann Statusfenster steuern */
        TCSStatWindowAutomatic= true;
        ShowWindow (hTCSstatWindow, SW_HIDE); /* jetzt DefWindowProc ! */
       case WM_NCLBUTTONUP:
       case WM_NCLBUTTONDBLCLK:
       case WM_SYSKEYDOWN:       /* Uebersetzt in WM_SYSCOMMAND */
       case WM_SYSKEYUP:
        DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam );
        break;
       case WM_NCRBUTTONDOWN:
        ShowWindow (hTCSstatWindow, SW_SHOWNA);
        UpdateWindow(hTCSstatWindow);
        break;
       case WM_LBUTTONDOWN: {
        #if !defined(__WIN32__) && !defined(_WIN32)
LftDwn:
        #endif
        if (iKey== (TCHAR) 0) iButton= 1; else iButton=iKey;
       }
       case WM_RBUTTONDOWN: if (iButton== (TCHAR) 0) iButton= 2;
       case WM_MBUTTONDOWN: if (iButton== (TCHAR) 0) iButton= 4; // wie DOS
        #if !defined(__WIN32__) && !defined(_WIN32)
         TCSGinCurPos= MAKEPOINT (msg.lParam);
        #else
         TCSGinCurPos.x= GET_X_LPARAM (msg.lParam);
         TCSGinCurPos.y= GET_Y_LPARAM (msg.lParam);
        #endif
        DPtoLP (hTCSWindowDC, (LPPOINT)&TCSGinCurPos, 1);
        TCSGinCurPos.x= LoRes(TCSGinCurPos.x);
        TCSGinCurPos.y= LoRes(TCSGinCurPos.y);
        break;
       case WM_LBUTTONUP: /* Falls erneuter Aufruf nach Taste unten wird */
       case WM_RBUTTONUP: /* der Cursor sonst wieder auf Pfeil umgestellt */
       case WM_MBUTTONUP:
        SetCursor (hGinCurs);
        break;
       case WM_KEYDOWN:          /* Hardwareanpassung, dann WM_CHAR */
       case WM_KEYUP:
        TranslateMessage (&msg);
        break;
       case WM_CHAR:             /* nach WM_KEYDOWN jetzt ASCII */
        iKey= (TCHAR) msg.wParam;
        #if !defined(__WIN32__) && !defined(_WIN32)
         goto LftDwn;           /* Workaround Fehlen mouse_event */
        #else
         mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
         break;
        #endif
       case WM_SYSCOMMAND:       /* und nach WM_SYSKEYDOWN Befehlsauswertung */
        switch (msg.wParam) {
         case SC_CLOSE:
          iKey= (FTNINT) 27;     /* <ALT><F4> -> ESC */
          #if !defined(__WIN32__) && !defined(_WIN32)
           goto LftDwn;
          #else
           mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
           break;
          #endif
         case TCS_WM_COPY:
          TCSWndProc_OnCopyClipboard ();
          break;
         default:
          DefWindowProc( msg.hwnd, msg.message, msg.wParam, msg.lParam);
          break;                 /* Sonst keine Befehle auswerten */
        } /* Systembefehle */
      } /* Window-Messageauswertung */
     } /* Messages fuer Graphikfenster */
    } /* Ende Eingabeschleife */
    *ic= (FTNINT) iButton;
    *ix=TCSGinCurPos.x;
    *iy=TCSGinCurPos.y;

    TCSStatWindowAutomatic= true;
    ShowWindow (hTCSstatWindow, SW_HIDE); /* Statusfenster unsichtbar */
    return;
}



/*
```

```
---------------------- Userroutinen: Statusmeldungen -------------------
*/



extern void bell (void)
{
    MessageBeep (-1);
}




extern void outtext (FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
{
int i;

    TCSstatRow++;
    if (TCSstatRow >= STAT_MAXROWS) {
     TCSstatRow= STAT_MAXROWS-1;
     for (i=0; i<TCSstatRow;i++)
      _tcscpy( TCSstatTextBuf[i],TCSstatTextBuf[i+1]);
    }

    _tcsncpy( TCSstatTextBuf[TCSstatRow],FTNSTRPARA(ftn_string),
                    min (FTNSTRPARL(ftn_string), STAT_MAXCOLUMNS));
    TCSstatTextBuf[TCSstatRow][STAT_MAXCOLUMNS]= (FTNCHAR) 0;
    // TCSstatTextBuf ist mit STAT_MAXCOLUMNS+1 fuer char(0) dimensioniert!

    TCSstatScrollY= TCSstatRow   /* Anzahl  Zeilen  im Display */;
    ScrollWindow (hTCSstatWindow, 0,
                (TCSstatOrgY-TCSstatScrollY)*TextLineHeight, NULL, NULL);

    TCSstatOrgY= TCSstatScrollY;

    SetScrollPos (hTCSstatWindow, SB_VERT, TCSstatScrollY, true);

    ShowWindow (hTCSstatWindow, SW_SHOW);
    UpdateWindow(hTCSstatWindow);
}



extern void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
                                    FTNINT *iL  FTNSTRPAR_TAIL(ftn_string))
{
    TCSGraphicError (*iErr, FTNSTRPARA(ftn_string));

}



/*
---------------------- Userroutinen: Hardcopy -------------------
*/


extern void hdcopy (void)
{
FTNINT      iErr;
// FTNSTRDESC  ftnstrg;
TCHAR       FilNam[TCS_FILE_NAMELEN], OldFilNam[TCS_FILE_NAMELEN];
OFSTRUCT    ReOpenBuf;

#if (JOURNALTYP == 1)
 HMETAFILE  hmf, hmf1;
 HDC        hTCSNewMetaFileDC;
 HRGN       hWindowRegion;
 HBRUSH     hBack;
#elif (JOURNALTYP == 2)
 HENHMETAFILE   hmf, hmf1;
 HDC            hTCSNewMetaFileDC;
 ENHMETAHEADER  emh ;
 DWORD          ErrorCode;
 LPVOID         lpMsgBuf;
#elif (JOURNALTYP == 3)
 struct xJournalEntry_typ    *xJournalEntry;
 FILE           *fHandle;
#endif

    FilNam[0] = (FTNCHAR) 0;
    OldFilNam[0] = (FTNCHAR) 0;
    do {     /* Suche erstes nicht existierendes File */
     _tcscpy(OldFilNam, FilNam);
     sprintf( FilNam, szTCSHardcopyFile, iHardcopyCount++ );
    } while ( (OpenFile (FilNam, &ReOpenBuf, OF_EXIST) != HFILE_ERROR) &&
               (_tcsicmp (FilNam,OldFilNam) > 0 )                      );

    if (_tcsicmp (FilNam,OldFilNam) <= 0 ) { /* kein Filename vorhanden */
     iErr= WRN_HDCFILOPN;
     TCSGraphicError (iErr,"");
     return;                                /* Error during Open -> ret */
    }

    iErr= MSG_HDCACT;
    TCSGraphicError (iErr,FilNam);

#if (JOURNALTYP ==1)
    hTCSNewMetaFileDC  = CreateMetaFile (FilNam);
    if (hTCSNewMetaFileDC == NULL) {
     iErr= WRN_HDCFILOPN;
     TCSGraphicError (iErr,"");
     return;                                /* Error during Open -> ret */
    }

    hmf = CloseMetaFile (hTCSMetaFileDC);       /* Metafile für WM_PAINT */

    SetWindowExtEx (hTCSNewMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSNewMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

    ScaleViewportExtEx (hTCSNewMetaFileDC, 1,1,-1,1,NULL);

    hWindowRegion= CreateRectRgn(TCSrect.left, TCSrect.top, TCSrect.right,TCSrect.bottom);
    hBack= CreateSolidBrush (dwColorTable[TCSBackgroundColour]); /* rechts,oben */
    FillRgn (hTCSNewMetaFileDC, hWindowRegion, hBack);  /* nicht eingeschlossen */
    #if !defined(__WIN32__) && !defined(_WIN32)
     DeleteBrush (hBack);
     DeleteRgn (hWindowRegion);                   /* Resourcen freigeben */
    #else
     DeleteObject (hBack);
     DeleteObject (hWindowRegion);
    #endif

    PlayMetaFile (hTCSNewMetaFileDC, hmf);
    hmf1= CloseMetaFile (hTCSNewMetaFileDC);
    if (hmf1 == NULL) {
     iErr= WRN_HDCFILWRT;
     TCSGraphicError (iErr,"");
     return;                                /* Error during Write -> ret */
    } else {
     DeleteMetaFile (hmf1); /* Freigabe Resourcen, nicht Löschen des Files! */
    }

    hTCSNewMetaFileDC  = CreateMetaFile (NULL); /* 16bit Windows Metafile */
    PlayMetaFile (hTCSNewMetaFileDC, hmf);      /* für neues Journalfile */
    DeleteMetaFile (hmf);                       /* alter Status Bildschirm */
    hTCSMetaFileDC = hTCSNewMetaFileDC;         /* bereit Weiterzeichnen */

#elif (JOURNALTYP == 2)
    hmf = CloseEnhMetaFile (hTCSMetaFileDC);    /* Metafile für WM_PAINT */
    hmf1  = CopyEnhMetaFile (hmf, FilNam);
    if (hmf1 == NULL) {
     ErrorCode= GetLastError(); // immer win32 bei emf
//   if (ErrorCode == ERROR_CLASS_ALREADY_EXISTS) {
//    Hier bei Bedarf Fehlerbehandlung einführen
//   } else {
      FormatMessage(
         FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
         NULL,
         ErrorCode,
         MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
         (LPTSTR) &lpMsgBuf,
         0,
         NULL
      );
      MessageBox (NULL, lpMsgBuf, szTCSWindowName, MB_ICONSTOP);
      LocalFree( lpMsgBuf ); // Free the buffer
//   } // Ende der Fehlerbehandlung
     iErr= WRN_HDCFILOPN;
     TCSGraphicError (iErr,"");
     return;                                /* Error during Open -> ret */
    }
    DeleteEnhMetaFile (hmf1); /* Handle freigeben, File  nicht geloescht! */

    GetEnhMetaFileHeader (hmf, sizeof (emh), &emh) ;
    hTCSNewMetaFileDC  = CreateEnhMetaFile (hTCSWindowDC, NULL, &emh.rclFrame,
                         _T("TCS for Windows\0Subroutine HardCopy\0"));
    SetMapMode (hTCSNewMetaFileDC, MM_ANISOTROPIC);
    SetViewportExtEx (hTCSNewMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSNewMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);
    SetWindowExtEx (hTCSNewMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSNewMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

    PlayEnhMetaFile (hTCSNewMetaFileDC, hmf, &TCSrect); // neues Journal

    DeleteEnhMetaFile (hmf);                         // alter Status Bildschirm
    hTCSMetaFileDC = hTCSNewMetaFileDC;              // bereit zum Weiterzeichnen

    SetViewportExtEx (hTCSMetaFileDC, TCSrect.right, -TCSrect.bottom, NULL);
    SetViewportOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.top, NULL);
    SetWindowExtEx (hTCSMetaFileDC, TCSrect.right, TCSrect.bottom, NULL);
    SetWindowOrgEx (hTCSMetaFileDC, TCSrect.left, TCSrect.bottom, NULL);

    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectFont (hTCSMetaFileDC, hTCSFont);        // Aktuellen Zeichenstatus an
    #else
     SelectObject (hTCSMetaFileDC, hTCSFont);
    #endif
    SetBkMode (hTCSMetaFileDC, TRANSPARENT );   // Metafile weitergegeben !
    SetTextAlign (hTCSMetaFileDC, TA_LEFT | TA_BOTTOM | TA_UPDATECP); // CP
    SetTextColor (hTCSMetaFileDC, dwColorTable[TKTRNX.iTxtCol]);
    #if !defined(__WIN32__) && !defined(_WIN32)
     SelectPen (hTCSMetaFileDC, hTCSPen); // 16bit: Makro aus windowsx.h
    #else
     SelectObject (hTCSMetaFileDC, hTCSPen); // 32bit: GDI Standardaufruf
    #endif

#elif (JOURNALTYP == 3)
    fHandle= fopen(FilNam, "w+");
    if ( fHandle == NULL) {
     iErr= WRN_HDCFILOPN;
     TCSGraphicError (iErr,"");
     return;                               /* Error during Open -> ret */
    }

    SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, hTCSJournal, previous, next, xJournalEntry)

    while (xJournalEntry != NULL) {
     fprintf( fHandle, "%02i#%04i-%03i\n", xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2
 );

#ifdef TRACE_CALLS
     switch (xJournalEntry->action) {
       case XACTION_INITT: {
        printf  ("%s $ \n","Initt ");
        break;
       }
       case XACTION_ERASE: {
        printf  ("%s $ \n","Erase ");
        break;
       }
       case XACTION_MOVABS: {
        printf  ("%s x:%i - y: %i $ \n","MovAbs ", xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_DRWABS: {
        printf  ("%s x:%i - y: %i $ \n","DrwAbs ", xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_DSHSTYLE: {
        printf  ("%s x:%i $ \n","DshStyle ", xJournalEntry->i1);
        break;
       }
       case XACTION_DSHABS: {
        printf  ("%s x:%i - y: %i $ \n","DshAbs ", xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_PNTABS: {
        printf  ("%s x:%i - y: %i $ \n","PntAbs ", xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_BCKCOL: {
        printf  ("%s x:%i $ \n","BckCol ", xJournalEntry->i1);
        break;
       }
       case XACTION_TXTCOL: {
        printf  ("%s x:%i $ \n","TxtCol ", xJournalEntry->i1);
        break;
       }
       case XACTION_LINCOL: {
        printf  ("%s x:%i $ \n","LinCol ", xJournalEntry->i1);
        break;
       }
       case XACTION_FONTATTR: {
        printf  ("%s x:%i - %i $ \n","Fontattr ", xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_GTEXT: {
        printf  ("%s iL:%i - C0: %i [ %c ] $ \n","GText ", xJournalEntry->i1, xJournalEntry->i2,
                xJournalEntry->i2);
        break;
       }
       case XACTION_ASCII: {
        printf  ("%s C1:%i - C2: %i [ %c %c ] § \n","ASCII ", xJournalEntry->i1, xJournalEntry->i2,
                            xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       default: {
        printf ("??? %i ??? \n", xJournalEntry->action) ;
        break;
       }
     }
#endif // TRACE_CALLS
     xJournalEntry= xJournalEntry -> previous;
    }
    fclose (fHandle);
#endif // Journaltyp=3
    ShowWindow (hTCSstatWindow, SW_HIDE);
    return;
}



/*
---- subroutine LIB_MOVC3 fuer Watcom- und GNU-Compiler ----------------
Hier nicht benoetigt, nur wg. Kompatibilitaet zur DOS-Version enthalten
*/


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

{
int n;
    if (FTNSTRPARA(dst) <= FTNSTRPARA(sou) ) {
     for (n=0; n<*len; n++) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
    } else {
     for (n= (*len)-1; n>=0; n--) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
    };
}
```

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

```
/** ******************************************************************************
\file       TCSdWINc.h
\brief      MS Windows Port: Low-Level Driver
\version    1.9
\author     (C) 2023 Dr.-Ing. Klaus Friedewald
\copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
\~german
         Headerfile zu TCSdWINc.c
\~english
         Headerfile for TCSdWIN.c
\~


****************************************************************************** */


 typedef int bool; // Typdefinition analog Cpp
 #define false 0
 #define true !false


/* ---- Zeichenbereich im Tektronix-Koordinatensystem --------------------- */

#define TEK_XMAX 1023
#define TEK_YMAX 780

/* ---- Erhoehung der Zeichenauflösung fuer hochaufloesende Bildschirme --- */

#if defined PixFac
 #define HiRes(iX) (iX*PixFac)
 #define LoRes(iX) (iX/PixFac)
#else
 #define HiRes(iX) iX
 #define LoRes(iX) iX
#endif

/* ----- Systemparameter -------------------------------------------------- */

#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 */




/* ------------ Programmparameter ----------------------------------------- */

#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("%:")

```

```c
#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")



/* Actioncodes des Journalfiles */

#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



/* Zuordnung Fehlernummern zu Meldungen */

#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



/* Initialisierungskonstanten *.INI, werden sinngemaess auch bei der
    Registry und XML-Initialisierung verwendet.
    Bei Erweiterungen Variableninitialisierung szTCSErrorMsg und TCSErrorLev
    in TCSdWINc.c fuer Registry und XML-Initialisierung nicht vergessen und
    alle Parser (*.ini bei INITT1(), Registry bei StoreIni() und
    *.xml bei sax_callback() beruecksichtigen! */

#define TCS_INISECT0 "Graph2D" // Root-Section, derzeit nur bei XML verwendet

#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")
    #if (JOURNALTYP ==1)
       #define TCS_HDCFILE_NAME _T("HDC%03i.WMF")
    #elif (JOURNALTYP ==2)
       #define TCS_HDCFILE_NAME _T("HDC%03i.EMF")
    #elif (JOURNALTYP ==3)
       #define TCS_HDCFILE_NAME _T("HDC%03i.HDC")
    #else
       #define TCS_HDCFILE_NAME _T("HDC%03i.UNKNOWN")
    #endif
 #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



/* ------------------ Kompatibilität 16/32bit ----------------------------- */

#if !defined(__WIN32__) && !defined(_WIN32)

 typedef char TCHAR, *PTCHAR;
 #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 */

#else
 #define EXPORT16
#endif



/* ------------ Compilerspezifische Definitionen -------------------------- */

// _____________________________ Open-Watcom ________________________________
#if defined __WATCOMC__
 #ifdef _UNICODE
  #error "Watcom Ftn77 basiert nicht auf UNICODE !!!"
 #endif

 #if !defined(__WIN32__) && !defined(_WIN32)
  #define TCSLEV3SYS 3 // TCSLEV(3) = 3 fuer Watcom/16 bit Windows
 #else
  #define TCSLEV3SYS 4 // TCSLEV(3) = 4 fuer Watcom/32 bit Windows
 #endif

 /* Deklaration Parameteruebergabe Fortran <-> C */
 typedef long int LOGICAL;
 typedef long int FTNINT;
 typedef float FTNREAL;
 typedef double FTNDOUBLE;
 typedef struct {float real, imag;} FTNCOMPLEX;
 typedef char FTNCHAR;
 typedef unsigned FTNCHARLEN;
 typedef struct { FTNCHAR * addr; FTNCHARLEN len; } FTNSTRDESC;
 typedef FTNSTRDESC FTNSTRPAR;
 #define FTNSTRPAR_TAIL(ftns)
 #define FTNSTRPARA(ftns) ftns->addr
 #define FTNSTRPARL(ftns) ftns->len
 #define CALLFTNSTRA(ftns) & ftns
 #define CALLFTNSTRL(ftns)
 #define FWRDFTNSTRA(ftns) ftns
 #define FWRDFTNSTRL(ftns)

 #pragma aux TKTRNX "^"; /* Fortran Naming Convention */
 #pragma aux tcslev3 "^";
 #pragma aux initt1 "^";
 #pragma aux finitt "^";
 #pragma aux GraphicError "^";
 #pragma aux winlbl "^";
 #pragma aux erase "^";
 #pragma aux swind1 "^";
 #pragma aux movabs "^";
 #pragma aux drwabs "^";
 #pragma aux dshabs "^";
 #pragma aux pntabs "^";
 #pragma aux bckcol "^";
 #pragma aux lincol "^";
 #pragma aux txtcol "^";
 #pragma aux DefaultColour "^"
 #pragma aux outgtext "^";
 #pragma aux italic "^";
 #pragma aux italir "^";
 #pragma aux dblsiz "^";
 #pragma aux nrmsiz "^";
 #pragma aux bell "^";
 #pragma aux outtext "^";
 #pragma aux tinput "^";
 #pragma aux dcursr "^";
 #pragma aux csize "^";
 #pragma aux hdcopy "^";
 #pragma aux lib_movc3 "^";

/* Deklarationen von durch C aufgerufenen FTN77-Unterprogrammen */
 #pragma aux igetarg "^"   // nur WATCOM: F77-Library
 FTNINT igetarg (FTNINT *iNo, FTNSTRDESC *Par);

 #pragma aux initt2 "^"   // nur WATCOM: F77-Library
 void INITT2 (void);

 #pragma aux SUBSTITUTE "^"        // aus STRINGS.FOR
 void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *n
                                  FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
                                  FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(n));


// ____________________________ GNU-CC ____________________________________
#elif defined __GNUC__
 #ifdef _UNICODE
  #error "GNU f77 basiert nicht auf UNICODE !!!"
 #endif

 #if defined (WINVER)
  #if defined (_WIN64)
   #define TCSLEV3SYS 7 // TCSLEV(3) = 7 fuer GCC / 64bit Windows
  #else
   #define TCSLEV3SYS 5 // TCSLEV(3) = 5 fuer GCC / Windows
  #endif // defined
 #else
  #define TCSLEV3SYS 0 // TCSLEV(3) = 0 fuer unknown
 #endif

/* Deklaration Parameteruebergabe Fortran <-> C */

 //  #include <g2c.h> // nur fuer g77, fuer gfortran s.u.
 typedef long int logical; // 3 (mit ftnlen) plattformabhaengige Definitionen
 typedef long int integer; // Ersatz fuer g2c.h: evtl. ueberpruefen

 typedef logical LOGICAL;
 typedef integer FTNINT;
 typedef float FTNREAL;
 typedef double FTNDOUBLE;
 typedef struct {float real, imag;} FTNCOMPLEX;

 typedef TCHAR FTNCHAR;
 #if __GNUC__ > 7  // GCC V7: size_t definiert, bei win64 8 Byte, nicht 4!
  typedef size_t ftnlen;  // Ersatz fuer g2c.h
  typedef size_t FTNCHARLEN;
 #else
  typedef long int ftnlen;  // Ersatz fuer g2c.h
  typedef ftnlen FTNCHARLEN; // size_t erst ab GCC > 7 definiert
 #endif

 typedef struct { FTNCHAR * addr; FTNCHARLEN len; } FTNSTRDESC;
 typedef FTNCHAR FTNSTRPAR;
 #define FTNSTRPAR_TAIL(ftns) , FTNCHARLEN ftns##_len
 #define FTNSTRPARA(ftns) ftns
 #define FTNSTRPARL(ftns) ftns##_len
 #define CALLFTNSTRA(ftns) ftns.addr
 #define CALLFTNSTRL(ftns) , ftns.len
 #define FWRDFTNSTRA(ftns) ftns
 #define FWRDFTNSTRL(ftns) , ftns##_len

 #define TKTRNX tktrnx_ /* Fortran Naming Convention */
 #define tcslev3 tcslev3_
 #define initt1 initt1_
 #define finitt finitt_
 #define GraphicError graphicerror_
 #define winlbl winlbl_
 #define erase erase_
 #define swindl swindl_
 #define movabs movabs_
 #define drwabs drwabs_
 #define dshabs dshabs_
 #define pntabs pntabs_
 #define bckcol bckcol_
 #define lincol lincol_
 #define txtcol txtcol_
 #define DefaultColour defaultcolour_
 #define outgtext outgtext_
 #define italic italic_
 #define italir italir_
 #define dblsiz dblsiz_
 #define nrmsiz nrmsiz_
 #define bell bell_
 #define outtext outtext_
 #define tinput tinput_
 #define dcursr dcursr_
 #define csize csize_
 #define hdcopy hdcopy_
 #define lib_movc3 lib_movc3_

/* Deklarationen von durch C aufgerufenen FTN77-Unterprogrammen */
 #define getarg getarg_       // aus GNU F77-Library
 FTNINT GETARG (FTNINT *iNo, FTNCHAR *line, FTNCHARLEN line_len);

 #define initt2 initt2_
 void INITT2 (void);

 #define SUBSTITUTE substitute_ // universeller Aufruf Watcom/GNU moeglich
 void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *new
                                          FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
                                          FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(new));

#endif
// ____________Ende systemabhaengige Deklarationen____________________


/* Forward Deklarationen: Codiert in C und auch in C verwendet */

void bell (void); //  -> Forward Deklaration
void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) );
void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
                   FTNINT *iL  FTNSTRPAR_TAIL(ftn_string));
// void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy);
void tinput (FTNINT *ic);
void finitt (); // ueberpruefen !!!

```

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

```
C> \file       TCSinitt.for
C> \version    1.4
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C> \~german
C> \brief   MS Windows Port: Initialisierung
C> \~english
C> \brief   MS Windows Port: initialization
C> \~
C
C
C> \~german
C> MS Windows-spezifische TCS-Routinen
C> \note
C> Initialisierung der DLL: Das Unterprogramm INITT darf sich nicht
C> in der DLL befinden, sondern muss statisch zu dem Anwenderprogramm
C> gelinkt werden, da sonst die Instanz der DLL und nicht die des
C> Anwenderprogramms ermittelt wird.
C>
C> \note
C> Achtung bei 64bit Betriebssystemen: Die Übergabe von Pointern erfolgt
C> durch Fortran77 Integer-Variablen. Bei Win64 beträgt die Pointerlänge
C> 8 Bytes entsprechend 2 StorageUnits (integer*4). Entsprechend muss der
C> Parameter nPtrStorageUnits angepasst werden.
C>
C> \note
C> Die Routine kann auch zur Initialisierung von Windows NT
C> Konsolenprogrammen verwendet werden.
C>
C
C
C> \~english
C> MS Windows specific subroutines
C> \note
C> Initialization of the DLL: The subroutine INITT must not be
C> placed inside the DLL, but must be linked statically to the user
C> program. Otherwise the instance of the DLL and not the instance
C> of the main programm will be optained.
C>
C> \note
C> Attention with 64bit operating systems: The passing of pointers is done
C> by Fortran77 integer variables. With Win64 the pointer length is
C> 8 bytes, corresponding to 2 StorageUnits (integer*4). In consequence the
C> parameter nPtrStorageUnits must be set >= 2.
C>
C> \note
C> This routine can also be used for initializing Windows NT console programs.
C>\~
C>
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Version 1.4, 30.4.2021, K. Friedewald
C     Anpassung an Windows64: Pointerlänge 8 Byte > int*4 bei win32
C
C  Version 1.3, 17.8.2020, K. Friedewald
C     Reaktivierung KHOMEY fuer HOME()
C
```

```
C  Version 1.2, 29.9.2004, K. Friedewald
C     Zusammenfassung der DLL-Initialisierung mit der LIB-Version. INITT
C     wird zusammen mit GetMainInstance.c in der LIB gehalten, die rest-
C     lichen Programme können sich in einer DLL befinden.
C
C  Version 1.1, 22.6.2004, K. Friedewald
C     Falls initt1 von dem Hauptprogramm ohne ein aktives Fenster aufgerufen
C     wird treten schwer reproduzierbare Fehler auf, da die Rueckmeldungen
C     auf die anfänglichen Windowsabfragen nicht eindeutig zugeordnet werden.
C
C     Abhilfe: Es wird jetzt bei Bedarf vor der Initialisierung ein eigenes
C     Hauptprogrammfenster erstellt.
C
C  Version 1.0, 19.3.2003, K. Friedewald
C


C
C>  Init Hardware & Software
C


      subroutine initt (iDummy)
C
      parameter(nptrstorageunits=2) ! max.Laenge Pointer in StorageUnits (2=64bit)
      integer iInstance(nPtrStorageUnits), iWindow(nPtrStorageUnits)
      call getmaininstandwin (iinstance, iwindow)
      call initt1 (iinstance, iwindow)
      call savemaininstandwin (iinstance, iwindow)

C> initt2() -> Reset Software
      entry initt2
      call lintrn
      call swindo (0,1023,0,780)
      call vwindo (0.,1023.,0.,780.)
      call rrotat (0.)
      call rscale (1.)
      call setmrg (0,1023)
      call nrmsiz
      call italir
      call home
      return
      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

```
C> \file       TKTRNX.fd
C> \brief      MS Windows Port: TCS Common Block TKTRNX
C> \version    1.4
C> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C> Header passend zu TKTRNX.h
C> \note
C> Da die folgende Definition kein Bestandteil eines Moduls
C> ist, versagt der DOXYGEN-Parser bei der Kombination von
C> COMMON und INTEGER. Workaraound: \\cond ... \\endcond.
C> \~english
C> header belonging to TKTRNX.h
C> \note
C> Because the following definition not beeing part of a module, the
C> DOXYGEN parser is not able to handle the combination of COMMON
C> and INTEGER declarations.  Workaraound: \\cond ... \\endcond.
C> \~
C> \cond
C Common Block TKTRNX, Version 1.3 für WINDOWS
C
      COMMON /tktrnx/
C           kbaudr,kerror,kgrafl,
     & khomey,
C           kkmode,
     & khorsz,kversz,
     & kitalc,ksizef,
     & klmrgn,krmrgn,
C           kScrX,kScrY,
C           ktblsz,khorzt(10),kvertt(10),
     & kbeamx,kbeamy,
C           kmovef,kpchar(4),kdasht,
     & kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
C      trealx,trealy,timagx,timagy,
     & trcosf,trsinf,trscal
     & ,xfac,yfac,xlog,ylog,kstcol,
     & ilincol, ibckcol, itxtcol, imouse

      SAVE /tktrnx/
      integer iTktrnxL
      parameter(itktrnxl=29) ! +11)

C Neue Variablen:
C     kHorSz,kVerSz: Buchstabengröße im (1024/780) Koordinatensystem
C     kBeamX, kBeamY: Aktuelle Strahlposition im (1024/780) Koordinatensystem
C     kStCol: Maximale Zeichenzahl in der Statuszeile
C     iLinCol, iBckCol, iTxtCol: Farbindices
C     iMouse: Anzahl der Maustasten. iMouse=0: keine Maus vorhanden
C
C Achtung:
C       Anpassung Parameter iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
C     Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
C
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

```
/** ***************************************************************************
\file       TKTRNX.h
\brief      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
\~german
         C Header passend zu TKTRNX.fd
\~english
         C header belonging to TKTRNX.fd
\~

\~german
\note
   Anpassung an die compilerabhaengige Namenskonvention erfolgt in TCSdSDLc.h
\~english
\note
   Adaption to the compiler specific name convention is done in TCSdSDLc.h
\~

*************************************************************************** */


extern struct TKTRNXcommonBlock {
FTNINT
//           kbaudr,kerror,kgrafl,
     khomey,
//           kkmode,
     khorsz,kversz,
     kitalc,ksizef,
     klmrgn,krmrgn,
//           kScrX,kScrY,
//           ktblsz,khorzt(10),kvertt(10),
     kBeamX,kBeamY,
//           kmovef,kpchar(4),kdasht,
     kminsx,kminsy,kmaxsx,kmaxsy;

FTNREAL
     tminvx,tminvy,tmaxvx,tmaxvy,
//      trealx,trealy,timagx,timagy,
     trcosf,trsinf,trscal
     ,xfac,yfac,xlog,ylog;
FTNINT
    kStCol,
    iLinCol, iBckCol, iTxtCol, iMouse;
} FAR TKTRNX;

```

Index