Graph2D Library --- SDL2 ---

Generated by Doxygen 1.8.19

Chapter 1

Plot10 & Advanced Graphing II

Graph2D is completely written in FTN77 and ANSI C90. Detailed compilation instructions are available for Windows (MinGW) and Debian (Raspberry Pi).

1.0.0.1 How to build the library:

Copy the sources into the /build subdirectory by running "$getfiles.bat sdlxx". Then use the workspace files for CodeBlocks (Windows IDE) or the bash script for Linux.

1.0.0.2 Using the library:

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

- Initialization: by calling the WINLBL subroutine and/or using ∗.xml files
- Icons (Windows only): by linking against a resource

1.0.0.3 Hardcopies

create proprietary ASCII journal files with the default ∗.hdc extension.

Chapter 2

Compiler Settings for Windows

2.0.1 Setting up the Windows IDE

2.0.1.1 MingGW for Windows 32bit and 64bit

2.0.1.1.1 Basic configuration (TDM and CodeBlocks) Install both TDM Toolchains, for 32-bit and for 64-bit (e.g. in C:\UsrProg\TDM-GCC-64 and C:\UsrProg\TDM-GCC-32). Then edit the following entries in CodeBlocks under 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.

2.0.1.2 Building the open source libraries SDL2, SDL2_ttf, miniXML and sglib

Building and storing of the binaries in /OpenContent/binaries/gcc is only necessary once, and only if a new compiler is used.

SDL2: Unzip SDL2-devel-2.x.y-mingw.tar.gz (currently version 2.0.20) and copy

- SDL2-2.0.20\i686-w64-mingw32∗.∗ -> TekLib\OpenContent\binaries\gccSDL2-2.0.20\i686-w64-mingw32\bin\↩
  SDL2.dll ->TekLib\OpenContent\binaries\gcc\lib
- SDL2-2.0.20\i686-w64-mingw32\lib\SDL2\libSDL2.a, libSDL2.dll.a -> TekLib\OpenContent\binaries\gcc\lib

SDL2_ttf: Unzip SDL2_ttf-devel-x.y.z-mingw.tar.gz (currently version 2.0.18) and copy

- SDL2_ttf-2.0.18\i686-w64-mingw32\include\SDL2\SDL_ttf.h -> TekLib\OpenContent\binaries\gccSDL2_ttf-2.0.18\i686-w64-mingw32\bin\SDL2_ttf.dll, zlib1.dll, libfreetype-6.dll ->TekLib\OpenContent\binaries\gcc\lib

- SDL2_ttf-2.0.18\i686-w64-mingw32\lib\SDL2\libSDL2_ttf.a, libSDL2_ttf.dll.a -> TekLib\OpenContent\binaries\gcc\lib

MiniXML: Compilation uses a MSYS Terminal, seperate for 32-bit 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 insert the following flags:
  LIBS = -lpthread -lssp
- $ make
- $ make test
- $ exit
- Copy (within 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

sglib: This 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

2.0.1.3 Settings for custom applications

2.0.1.3.1 Fortran 32bit Compilerswitches:

- maximum -O1 optimization for compililing the library is possible. If -O2 and -O3 (higher speed) or -Os (size) are used, the labels of the sample program AG2DEMO4 are not drawn at the axis!
- "Strip all symbols from binary [-s]" is possible.

2.0.1.3.2 Fortran 64bit Compilerswitches:

- maximum -O2 optimization for compililing the library is possible. If -O3 (higher speed) or -Os (size) are used, the labels of the sample program AG2DEMO4 are not drawn on the axis!
- "Strip all symbols from binary [-s]" is possible.

2.0.1.3.3 Link

- static: allows to run the programs on machines without MinGW installed.

Chapter 3

Compiler settings for Linux

3.0.1 Raspberry Pi with Debian 11 (Bullseye)

3.0.1.1 Preparing the OS

Basic installation: Raspberry Pi OS with desktop, Debian Version 11 (Bullseye), 32-bit

Install Fortran:

- # sudo apt-get update
- # sudo apt-get upgrade
- # sudo apt-get install gfortran

Install SDL2 (apt-get install libsdl2 unnecessary, already part of the standard distribution):

- # sudo apt-get install libsdl2-dev
- # sudo apt-get install libsdl2-ttf-dev

Install MiniXML:

- # sudo apt-get install libmxml-dev

3.0.1.2 Compiling

Copy the Teklib\Build directory to the target machine. Make the batch file executable:

- # chmod 755 build.sh

Build the library and example programs:

- # ./build.sh

Chapter 4

Data Type Index

4.1 Data Types List

Here are the data types with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Data Type Documentation

6.1 FTNCOMPLEX Struct Reference

```
#include <TCSdSDLc.h>
```

Public Attributes

- float real
- float imag

6.1.1 Detailed Description

Definition at line 46 of file TCSdSDLc.h.

6.1.2 Member Data Documentation

6.1.2.1 imag

```
float FTNCOMPLEX::imag
```

Definition at line 46 of file TCSdSDLc.h.

6.1.2.2 real

```
float FTNCOMPLEX::real
```

Definition at line 46 of file TCSdSDLc.h.

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

- TCSdSDLc.h

6.2 FTNSTRDESC Struct Reference

```
#include <TCSdSDLc.h>
```

Public Attributes

- FTNCHAR ∗ addr
- FTNCHARLEN len

6.2.1 Detailed Description

Definition at line 53 of file TCSdSDLc.h.

6.2.2 Member Data Documentation

6.2.2.1 addr

```
FTNCHAR* FTNSTRDESC::addr
```

Definition at line 53 of file TCSdSDLc.h.

6.2.2.2 len

```
FTNCHARLEN FTNSTRDESC::len
```

Definition at line 53 of file TCSdSDLc.h.

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

- TCSdSDLc.h

6.3 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

6.3.1 Detailed Description

Definition at line 19 of file TKTRNX.h.

6.3.2 Member Data Documentation

6.3.2.1 iBckCol

```
FTNINT TKTRNXcommonBlock::iBckCol
```

Definition at line 34 of file TKTRNX.h.

6.3.2.2 iLinCol

```
FTNINT TKTRNXcommonBlock::iLinCol
```

Definition at line 34 of file TKTRNX.h.

6.3.2.3 iTxtCol

```
FTNINT TKTRNXcommonBlock::iTxtCol
```

Definition at line 34 of file TKTRNX.h.

6.3.2.4 kBeamX

```
FTNINT TKTRNXcommonBlock::kBeamX
```

Definition at line 25 of file TKTRNX.h.

6.3.2.5 kBeamY

```
FTNINT TKTRNXcommonBlock::kBeamY
```

Definition at line 25 of file TKTRNX.h.

6.3.2.6 khomey

```
FTNINT TKTRNXcommonBlock::khomey
```

Definition at line 21 of file TKTRNX.h.

6.3.2.7 khorsz

```
FTNINT TKTRNXcommonBlock::khorsz
```

Definition at line 22 of file TKTRNX.h.

6.3.2.8 kitalc

```
FTNINT TKTRNXcommonBlock::kitalc
```

Definition at line 23 of file TKTRNX.h.

6.3.2.9 klmrgn

```
FTNINT TKTRNXcommonBlock::klmrgn
```

Definition at line 24 of file TKTRNX.h.

6.3.2.10 kmaxsx

```
FTNINT TKTRNXcommonBlock::kmaxsx
```

Definition at line 26 of file TKTRNX.h.

6.3.2.11 kmaxsy

```
FTNINT TKTRNXcommonBlock::kmaxsy
```

Definition at line 26 of file TKTRNX.h.

6.3.2.12 kminsx

```
FTNINT TKTRNXcommonBlock::kminsx
```

Definition at line 26 of file TKTRNX.h.

6.3.2.13 kminsy

```
FTNINT TKTRNXcommonBlock::kminsy
```

Definition at line 26 of file TKTRNX.h.

6.3.2.14 krmrgn

```
FTNINT TKTRNXcommonBlock::krmrgn
```

Definition at line 24 of file TKTRNX.h.

6.3.2.15 ksizef

```
FTNINT TKTRNXcommonBlock::ksizef
```

Definition at line 23 of file TKTRNX.h.

6.3.2.16 kStCol

```
FTNINT TKTRNXcommonBlock::kStCol
```

Definition at line 33 of file TKTRNX.h.

6.3.2.17 kversz

```
FTNINT TKTRNXcommonBlock::kversz
```

Definition at line 22 of file TKTRNX.h.

6.3.2.18 tmaxvx

```
FTNREAL TKTRNXcommonBlock::tmaxvx
```

Definition at line 29 of file TKTRNX.h.

6.3.2.19 tmaxvy

```
FTNREAL TKTRNXcommonBlock::tmaxvy
```

Definition at line 29 of file TKTRNX.h.

6.3.2.20 tminvx

```
FTNREAL TKTRNXcommonBlock::tminvx
```

Definition at line 29 of file TKTRNX.h.

6.3.2.21 tminvy

```
FTNREAL TKTRNXcommonBlock::tminvy
```

Definition at line 29 of file TKTRNX.h.

6.3.2.22 trcosf

```
FTNREAL TKTRNXcommonBlock::trcosf
```

Definition at line 30 of file TKTRNX.h.

6.3.2.23 trscal

```
FTNREAL TKTRNXcommonBlock::trscal
```

Definition at line 30 of file TKTRNX.h.

6.3.2.24 trsinf

```
FTNREAL TKTRNXcommonBlock::trsinf
```

Definition at line 30 of file TKTRNX.h.

6.3.2.25 xfac

```
FTNREAL TKTRNXcommonBlock::xfac
```

Definition at line 31 of file TKTRNX.h.

6.3.2.26 xlog

```
FTNREAL TKTRNXcommonBlock::xlog
```

Definition at line 31 of file TKTRNX.h.

6.3.2.27 yfac

```
FTNREAL TKTRNXcommonBlock::yfac
```

Definition at line 31 of file TKTRNX.h.

6.3.2.28 ylog

```
FTNREAL TKTRNXcommonBlock::ylog
```

Definition at line 31 of file TKTRNX.h.

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

- TKTRNX.h

6.4 xJournalEntry_typ Struct Reference

Public Attributes

- struct xJournalEntry_typ ∗ previous
- struct xJournalEntry_typ ∗ next
- FTNINT action
- FTNINT i1
- FTNINT i2

6.4.1 Detailed Description

Definition at line 237 of file TCSdSDLc.c.

6.4.2 Member Data Documentation

6.4.2.1 action

```
FTNINT xJournalEntry_typ::action
```

Definition at line 239 of file TCSdSDLc.c.

6.4.2.2 i1

```
FTNINT xJournalEntry_typ::i1
```

Definition at line 239 of file TCSdSDLc.c.

6.4.2.3 i2

```
FTNINT xJournalEntry_typ::i2
```

Definition at line 239 of file TCSdSDLc.c.

6.4.2.4 next

```
struct xJournalEntry_typ* xJournalEntry_typ::next
```

Definition at line 238 of file TCSdSDLc.c.

6.4.2.5 previous

```
struct xJournalEntry_typ* xJournalEntry_typ::previous
```

Definition at line 237 of file TCSdSDLc.c.

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

- TCSdSDLc.c

Chapter 7

File Documentation

7.1 AG2.for File Reference

Graph2D: Tektronix Advanced Graphing II Emulation.

Functions/Subroutines

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

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

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

7.1.1 Detailed Description

Graph2D: Tektronix Advanced Graphing II Emulation.

Version

(2023,135, x)

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Layer 2: scientific 2-D graphic subroutines

Note

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

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

Definition in file AG2.for.

7.1.2 Function/Subroutine Documentation

7.1.2.1 ag2lev()

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

Definition at line 94 of file AG2.for.

7.1.2.2 alfsetc()

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

Definition at line 2563 of file AG2.for.

7.1.2.3 bar()

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

Definition at line 1688 of file AG2.for.

7.1.2.4 binitt()

```
subroutine binitt
```

Definition at line 714 of file AG2.for.

7.1.2.5 bsyms()

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

Definition at line 1840 of file AG2.for.

7.1.2.6 calcon()

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

Definition at line 1326 of file AG2.for.

7.1.2.7 calpnt()

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

Definition at line 1271 of file AG2.for.

7.1.2.8 check()

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

Definition at line 798 of file AG2.for.

7.1.2.9 cmnmx()

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

Definition at line 920 of file AG2.for.

7.1.2.10 coptim()

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

Definition at line 1115 of file AG2.for.

7.1.2.11 cplot()

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

Definition at line 1538 of file AG2.for.

7.1.2.12 datget()

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

Definition at line 1660 of file AG2.for.

7.1.2.13 dinitx()

```
subroutine dinitx
```

Definition at line 644 of file AG2.for.

7.1.2.14 dinity()

```
subroutine dinity
```

Definition at line 658 of file AG2.for.

7.1.2.15 dlimx()

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

Definition at line 464 of file AG2.for.

7.1.2.16 dlimy()

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

Definition at line 476 of file AG2.for.

7.1.2.17 dsplay()

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

Definition at line 1524 of file AG2.for.

7.1.2.18 eformc()

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

Definition at line 2434 of file AG2.for.

7.1.2.19 esplit()

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

Definition at line 2467 of file AG2.for.

7.1.2.20 expoutc()

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

Definition at line 2487 of file AG2.for.

7.1.2.21 fformc()

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

Definition at line 2375 of file AG2.for.

7.1.2.22 filbox()

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

Definition at line 1755 of file AG2.for.

7.1.2.23 findge()

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

Definition at line 2922 of file AG2.for.

7.1.2.24 findle()

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

Definition at line 2941 of file AG2.for.

7.1.2.25 fonlyc()

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

Definition at line 2403 of file AG2.for.

7.1.2.26 frame()

```
subroutine frame
```

Definition at line 1510 of file AG2.for.

7.1.2.27 gline()

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

Definition at line 2173 of file AG2.for.

7.1.2.28 grid()

```
subroutine grid
```

Definition at line 1956 of file AG2.for.

7.1.2.29 hbarst()

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

Definition at line 672 of file AG2.for.

7.1.2.30 iformc()

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

Definition at line 2343 of file AG2.for.

7.1.2.31 infin()

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

Definition at line 142 of file AG2.for.

7.1.2.32 iother()

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

Definition at line 3066 of file AG2.for.

7.1.2.33 iubgc()

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

Definition at line 1473 of file AG2.for.

7.1.2.34 justerc()

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

Definition at line 2666 of file AG2.for.

7.1.2.35 keyset()

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

Definition at line 1634 of file AG2.for.

7.1.2.36 label()

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

Definition at line 2200 of file AG2.for.

7.1.2.37 leap()

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

Definition at line 1459 of file AG2.for.

7.1.2.38 line()

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

Definition at line 109 of file AG2.for.

7.1.2.39 locge()

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

Definition at line 2963 of file AG2.for.

7.1.2.40 locle()

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

Definition at line 2981 of file AG2.for.

7.1.2.41 logtix()

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

Definition at line 2042 of file AG2.for.

7.1.2.42 loptim()

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

Definition at line 988 of file AG2.for.

7.1.2.43 lwidth()

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

Definition at line 2732 of file AG2.for.

7.1.2.44 mnmx()

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

Definition at line 881 of file AG2.for.

7.1.2.45 monpos()

```
subroutine monpos (
            integer nbase,
            integer iy1,
            real dpos,
            integer spos )
```

Definition at line 2159 of file AG2.for.

7.1.2.46 notatec()

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

Definition at line 2618 of file AG2.for.

7.1.2.47 npts()

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

Definition at line 155 of file AG2.for.

7.1.2.48 numsetc()

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

Definition at line 2316 of file AG2.for.

7.1.2.49 optim()

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

Definition at line 971 of file AG2.for.

7.1.2.50 oubgc()

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

Definition at line 1487 of file AG2.for.

7.1.2.51 place()

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

Definition at line 512 of file AG2.for.

7.1.2.52 remlab()

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

Definition at line 2807 of file AG2.for.

7.1.2.53 rescom()

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

Definition at line 3050 of file AG2.for.

7.1.2.54 rgchek()

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

Definition at line 854 of file AG2.for.

7.1.2.55 roundd()

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

Definition at line 2999 of file AG2.for.

7.1.2.56 roundu()

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

Definition at line 3015 of file AG2.for.

7.1.2.57 savcom()

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

Definition at line 3034 of file AG2.for.

7.1.2.58 setwin()

```
subroutine setwin
```

Definition at line 622 of file AG2.for.

7.1.2.59 sizel()

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

Definition at line 188 of file AG2.for.

7.1.2.60 sizes()

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

Definition at line 177 of file AG2.for.

7.1.2.61 slimx()

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

Definition at line 488 of file AG2.for.

7.1.2.62 slimy()

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

Definition at line 500 of file AG2.for.

7.1.2.63 spread()

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

Definition at line 2870 of file AG2.for.

7.1.2.64 stepl()

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

Definition at line 166 of file AG2.for.

7.1.2.65 steps()

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

Definition at line 131 of file AG2.for.

7.1.2.66 symbl()

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

Definition at line 120 of file AG2.for.

7.1.2.67 symout()

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

Definition at line 1857 of file AG2.for.

7.1.2.68 teksym()

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

Definition at line 1882 of file AG2.for.

7.1.2.69 teksym1()

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

Definition at line 1930 of file AG2.for.

7.1.2.70 tset()

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

Definition at line 2089 of file AG2.for.

7.1.2.71 tset2()

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

Definition at line 2127 of file AG2.for.

7.1.2.72 typck()

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

Definition at line 823 of file AG2.for.

7.1.2.73 vbarst()

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

Definition at line 692 of file AG2.for.

7.1.2.74 vlablc()

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

Definition at line 2643 of file AG2.for.

7.1.2.75 width()

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

Definition at line 2691 of file AG2.for.

7.1.2.76 xden()

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

Definition at line 312 of file AG2.for.

7.1.2.77 xetyp()

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

Definition at line 596 of file AG2.for.

7.1.2.78 xfrm()

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

Definition at line 390 of file AG2.for.

7.1.2.79 xlab()

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

Definition at line 290 of file AG2.for.

7.1.2.80 xlen()

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

Definition at line 364 of file AG2.for.

7.1.2.81 xloc()

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

Definition at line 246 of file AG2.for.

7.1.2.82 xloctp()

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

Definition at line 268 of file AG2.for.

7.1.2.83 xmfrm()

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

Definition at line 438 of file AG2.for.

7.1.2.84 xmtcs()

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

Definition at line 416 of file AG2.for.

7.1.2.85 xneat()

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

Definition at line 202 of file AG2.for.

7.1.2.86 xtics()

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

Definition at line 342 of file AG2.for.

7.1.2.87 xtype()

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

Definition at line 544 of file AG2.for.

7.1.2.88 xwdth()

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

Definition at line 570 of file AG2.for.

7.1.2.89 xzero()

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

Definition at line 224 of file AG2.for.

7.1.2.90 yden()

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

Definition at line 327 of file AG2.for.

7.1.2.91 yetyp()

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

Definition at line 609 of file AG2.for.

7.1.2.92 yfrm()

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

Definition at line 403 of file AG2.for.

7.1.2.93 ylab()

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

Definition at line 301 of file AG2.for.

7.1.2.94 ylen()

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

Definition at line 377 of file AG2.for.

7.1.2.95 yloc()

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

Definition at line 257 of file AG2.for.

7.1.2.96 ylocrt()

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

Definition at line 279 of file AG2.for.

7.1.2.97 ymdyd()

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

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

Definition at line 1404 of file AG2.for.

7.1.2.98 ymfrm()

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

Definition at line 451 of file AG2.for.

7.1.2.99 ymtcs()

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

Definition at line 427 of file AG2.for.

7.1.2.100 yneat()

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

Definition at line 213 of file AG2.for.

7.1.2.101 ytics()

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

Definition at line 353 of file AG2.for.

7.1.2.102 ytype()

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

Definition at line 557 of file AG2.for.

7.1.2.103 ywdth()

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

Definition at line 583 of file AG2.for.

7.1.2.104 yzero()

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

Definition at line 235 of file AG2.for.

7.2 AG2.for

```
C> \file       AG2.for
C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
C> \version    (2023,135, x)
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C>  Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
C> \note
C>     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
C>     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
C>
C> \~english
C> Layer 2: scientific 2-D graphic subroutines
C> \note
C>     The control character for exponent (originally -1) is now SOH=char(1)
C>     and for index (originally -2) STX=char(2).
C>
C> \~
C> \note \verbatim
C>   Package:
C>    - AG2.for:          chart plotting routines
C>    - AG2Holerith.for: deprecated routines
C>    - AG2USR.for:       default userroutines
C>    - G2dAG2.fd:        commonblock
C> \endverbatim
C
C
C  Tektronix Advanced Graphics 2 - Version 2.x
C
C
C     Neuer Code in Fortran 77. Die Verwendung der im Manual dokumentierten
C     Unterprogramme bleibt unveraendert, die direkte Manipulation von
C     Variablen des zugrundeliegenden Commonblockes ist jedoch nicht mehr
C     empfehlenswert. IBASEX (iPar) und IBASEY(iPar) mit ipar <>0,
C     IBASEC, COMGET und COMSET sollten in neuen Programmen nicht verwendet
C     werden.
C
C     Die Zwischenspeicherung der Statusvariablen ueber
C           SAVCOM und RESCOM
C     und die Achsensteuerung ueber
C           IBASEX(0), IBASEY(0) und IOTHER
C     werden weiterhin unterstuetzt.
C
C     Die Implementation der Unterprogramme COMGET und COMSET setzt die gleiche
C     Laenge von REAL und INTEGER-Variablen voraus.
C
C     Da Holerithvariablen von modernen Compilern uneinheitlich unterstuetzt
C     werden (4Habcd entweder als gepackte Integervariable oder als Character-
C     variable interpretiert), wurden die folgenden Routinen angepasst:
C      - subroutine PLACE (Lit): Lit wird nur noch als Ordnungszahl (1..13)
C        und nicht mehr alternativ als Literal ('STD', 'UPH') interpretiert.
C
C     subroutine LEAP (iyear): Die Schaltjahrkorrektur erfolgt nicht mehr
C     als SUBROUTINE ueber einen Common-Block, sondern direkt als
C     integer function LEAP (iyear) ! = 1: Schaltjahr, sonst 0
C
C     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
C     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
C
C     Intern erfolgt die Stringverarbeitung ueber Charactervariablen als
C     nullterminierte C-Strings.
C
C     Der User-API wurden die folgenden Unterprogramme als Charactervarianten
C     der Original-Holerithroutinen hinzugefuegt:
C      - subroutine NUMSETC (fnum,nbase, outstr,fillstr)
C      - subroutine FONLYC (fnum,iwidth,idec, outstr,fillstr)
C      - subroutine EFORMC (fnum,iwidth,idec, outstr,fillstr)
C      - subroutine EXPOUTC (nbase,iexp, outstr,fillstr)
C      - subroutine ALFSETC (fnum,iwidth,labtyp,outstr)
C      - subroutine NOTATEC (IX,IY,LENCHR,IARRAY)
```

```
C       - subroutine JUSTERC
C
C       - subroutine USESETC (fnum, iwidth, nbase, labstr)
C
C       subroutine MONPOS (nbase,iy1,dpos, spos) ! spos ist INTEGER
C       subroutine GLINE (nbase,datapt,spos) ! spos ist INTEGER
C
C      Der Code ab Version 2.0 wird nicht mehr fuer CP/M entwickelt. Letzte
C      unter CP/M compilierbare Version: (2006, 013, 1)
C
C      Zugehoerige Module:
C       - AG2.FOR:      Basisfunktionen
C       - AG2Holerith: Veraltete Unterprogramme zur Wahrung der Kompatibilitaet
C                      (Unterstuetzung Holerithvariablen und vektorisierter Zu-
C                      griff auf den Commonblock)
C       - AG2USR.FOR:  Userroutinen
C       - G2dAG2.fd:   Commonblockdefinition
C

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

      call tcslev (ilevel) ! level(3)= System aus TCS
      ilevel(1)=2023       ! Aenderungsjahr
      ilevel(2)= 135       ! 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



      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)
```

```fortran

      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
```

```fortran
       leap= 1
      else
       leap= 0
      end if
      return
      end



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

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



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

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



C
C  Zeichenroutinen
C

      subroutine frame
      implicit none
      include 'G2dAG2.fd'

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



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

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



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

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

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

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

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

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

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

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



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

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



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

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



C  Balkendiagramme

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

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

       call seetrn(xfac,yfac, key)
       if (key .eq. 2) then ! logarithmische Werte
        ibegx= cxysmin(1)
```

```
          ibegy= cxysmin(2)
         else
          call wincot (0.,0.,ibegx,ibegy)
         end if
        end if

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



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

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

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

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

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

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

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

         if (ishade .ge. 8) then ! Bit3: diagonal fallend
          idely= iminx-imaxx
          iymax= imaxy+imaxx-iminx
          i= iminy+lspace
120       continue ! repeat ...
```

```
         call movea (ximin,real(i))
         call drawa (ximax,real(i+idely))
         i= i+lspace
        if (i .lt. iymax) goto 120 ! ... until
        ishift= ishade -8
       else
        ishift= ishade
       end if

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



C  Zeichnen von Symbolen

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

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



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

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



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

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



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

      b= real(istart)*.01745
      mx= nint(siz*cos(b))
      my= nint(siz*sin(b))
      call movrel (mx,my)
      do 100 i= istart+incr, iend, incr
       b= real(i)*.01745
       mix= nint(siz*cos(b))
       miy= nint(siz*sin(b))
       call drwrel (mix-mx,miy-my)
       mx= mix
       my= miy
100   continue
      return
      end



C Netz und Ticmarks

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

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

```
             mlim=mlim-1
             goto 110
            else if (mlim. lt. 0) then
             call logtix (2,xyext,tintvl,cxymbeg(2),cxymend(2))
            end if
           end if
           xyext= xyext+tintvl
           call movabs (cxybeg(2), nint(xyext))
           call drwabs (cxyend(2), nint(xyext))
100       continue
         end if ! Labtyp=6: Monate
        end if ! Ende Zeichnen Ticmarks
       end if ! Ende Zeichnen der Achse

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



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

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

        else if (nbase .eq. 2) then ! y-Achse
         if  (mstart .gt. mend) then
          idx= ihorz
         else
          idx= -ihorz
```

```fortran
        end if
        idy=  -ivert / 3
        call movabs (mend,logtic)
        call drwabs (mstart,logtic)
       end if

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



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

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

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



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

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

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



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

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



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

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



C Label

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

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

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

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

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

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

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

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

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

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

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

       if (nbase .eq. 1) then ! x-Achse
        iy= level1
        if(stag .and. even) iy= level2
        even= .not. even
        call notatec (ispos+ioff,iy, labstr)
       else ! y-Achse
        call notatec (level1+ioff,ispos-igap,labstr)
       end if
       dpos= dpos+dintv
       ispos= ispos+isintv
100   continue ! end do

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



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

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

```
       call fformc (fnum,iwidth, cxydec(nbase), outstr)
      else
       call iformc (fnum,iwidth, outstr)
      end if
      return
      end



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

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

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

      return

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

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



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

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

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



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

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

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

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

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



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

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

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

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

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



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

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



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

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

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

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

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

      return
      end



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

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



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

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

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



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

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



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

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

      if (iposflag .lt. 0) then ! linksbuendig
       ioff= 0
      else ! rechtsbuendig und zentriert
```

```
         ioff= -linwdt((ilen-nctrl)*8-2)/8       ! rechtsbuendig
         if (iposflag.eq.0) ioff= ioff / 2        ! zentriert
        end if

        return
        end



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

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

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

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

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

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

        return
        end



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

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

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

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

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

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

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



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

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

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

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



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

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

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

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

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

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

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

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

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



  C
  C  Tabellensuche und Rundungen
  C

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

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

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



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

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

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

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



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

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

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



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

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

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



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

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



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

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


```

```

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

      integer i
      integer arr(1)
      equivalence(arr(1),cline)
      do 10 i=1,g2dag21
       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,g2dag21
       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
```

7.3 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

7.3.1 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.

7.3.2 Function/Subroutine Documentation

7.3.2.1 alfset()

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

Definition at line 45 of file AG2Holerith.for.

7.3.2.2 comdmp()

```
subroutine comdmp
```

Definition at line 328 of file AG2Holerith.for.

7.3.2.3 comget()

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

Definition at line 271 of file AG2Holerith.for.

7.3.2.4 comset()

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

Definition at line 299 of file AG2Holerith.for.

7.3.2.5 eform()

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

Definition at line 173 of file AG2Holerith.for.

7.3.2.6 expout()

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

Definition at line 90 of file AG2Holerith.for.

7.3.2.7 fform()

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

Definition at line 189 of file AG2Holerith.for.

7.3.2.8 fonly()

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

Definition at line 205 of file AG2Holerith.for.

7.3.2.9 hlabel()

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

Definition at line 121 of file AG2Holerith.for.

7.3.2.10 hstrin()

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

Definition at line 112 of file AG2Holerith.for.

7.3.2.11 ibasec()

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

Definition at line 241 of file AG2Holerith.for.

7.3.2.12 ibasex()

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

Definition at line 251 of file AG2Holerith.for.

7.3.2.13 ibasey()

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

Definition at line 261 of file AG2Holerith.for.

7.3.2.14 iform()

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

Definition at line 221 of file AG2Holerith.for.

7.3.2.15 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.

7.3.2.16 notate()

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

Definition at line 30 of file AG2Holerith.for.

7.3.2.17 numset()

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

Definition at line 67 of file AG2Holerith.for.

7.3.2.18 vlabel()

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

Definition at line 139 of file AG2Holerith.for.

7.3.2.19 vstrin()

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

Definition at line 130 of file AG2Holerith.for.

7.4 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
```

7.5 AG2uline.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine uline (x, y, i)

7.5.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2uline.for.

7.5.2 Function/Subroutine Documentation

7.5.2.1 uline()

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

Definition at line 10 of file AG2uline.for.

7.6 AG2uline.for

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

7.7 AG2umnmx.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine umnmx (array, amin, amax)

7.7.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2umnmx.for.

7.7.2 Function/Subroutine Documentation

7.7.2.1 umnmx()

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

Definition at line 9 of file AG2umnmx.for.

7.8 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

```

7.9 AG2upoint.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- real function upoint (arr, ii, oldone)

7.9.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2upoint.for.

7.9.2 Function/Subroutine Documentation

7.9.2.1 upoint()

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

Definition at line 9 of file AG2upoint.for.

7.10 AG2upoint.for

```
C> \file    AG2upoint.for
C> \brief   Graph2D: Dummy User Routine
C
C  Tektronix Advanced Graphics 2 - Version 2.0
C
C     User Subroutinen
C

      real function upoint (arr,ii,oldone)
      upoint=0.
      return
      end
```

7.11 AG2users.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine users (x, y, i)

7.11.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2users.for.

7.11.2 Function/Subroutine Documentation

7.11.2.1 users()

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

Definition at line 9 of file AG2users.for.

7.12 AG2users.for

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

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

7.13 AG2useset.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

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

7.13.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2useset.for.

7.13.2 Function/Subroutine Documentation

7.13.2.1 useset()

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

Definition at line 9 of file AG2useset.for.

7.14 AG2useset.for

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

7.15 AG2usesetC.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

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

7.15.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2usesetC.for.

7.15.2 Function/Subroutine Documentation

7.15.2.1 usesetc()

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

Definition at line 9 of file AG2usesetC.for.

7.16 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
```

7.17 AG2UsrSoftek.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine softek (isym)

7.17.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2UsrSoftek.for.

7.17.2 Function/Subroutine Documentation

7.17.2.1 softek()

```
subroutine softek (
             isym )
```

Definition at line 9 of file AG2UsrSoftek.for.

7.18 AG2UsrSoftek.for

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

7.19 G2dAG2.fd File Reference

Graph2D: AG2 Common Block G2dAG2.

7.19.1 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.

7.20 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
```

7.21 GetHDC.for File Reference

Restore Hardcopies.

Functions/Subroutines

- logical function gethdc (Filnam)

7.21.1 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.

7.21.2 Function/Subroutine Documentation

7.21.2.1 gethdc()

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

Parameters

FilNam	Hardcopyfie

Returns

(optional) .true. -> Error

Definition at line 15 of file GetHDC.for.

7.22 GetHDC.for

```
00001 C> \file       GetHDC.for
00002 C> \brief      Restore Hardcopies
00003 C> \version    1.2
00004 C> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Einlesen und Zeichnen von Hardcopydateien\n
00008 C> Verwendete temporaeres Ein/Ausgabeunit: 41. Falls bereits belegt, wird ein freier Kanal gesucht
00009 C> \~english
00010 C> Read and plot hardcopies\n
00011 C> Temporary input unit: 41. If already used, an other channel will be searched.
00012 C> \~
00013 C
00014
00015       logical function gethdc (Filnam)
00016 C> \param FilNam: Hardcopyfie
00017 C> \result (optional) .true. -> Error
00018       include 'Tktrnx.fd'
00019       integer tcs_messagelen, iunit
00020       parameter(tcs_messagelen=132)
00021       character *(*) filnam
00022       logical iunitused
00023       character *(TCS_MESSAGELEN+1) txtstring
00024
00025       integer ios, idash, iprntlen, iactlen
00026       integer action, i1, i2
00027
00028       iunit= 40
00029       gethdc= .true.
00030
00031  5    continue ! repeat
00032         iunit= iunit+1
00033         inquire (unit=iunit, opened= iunitused)
00034       if (iunitused) goto 5
00035
00036       open (iunit,file=filnam,status='old',iostat=ios,form='formatted')
00037       if (ios.ne.0) then
00038         call graphicerror (6, ' ')
00039         return
00040       end if
00041
00042  10   continue ! repeat
00043         read (iunit, fmt='(i2,1x,i4,1x,i3)', iostat=ios)action, i1, i2
00044         if (ios.gt.0) then ! Error, not EOF
00045          call graphicerror (8, ' ')
00046          return
00047         end if
00048         if (action.eq.1) then ! XACTION_INITT
00049           call defaultcolour()
00050           call erase ()
00051         else if (action.eq.2) then ! XACTION_ERASE
00052           call erase ()
00053         else if (action.eq.3) then ! XACTION_MOVABS
00054           call movabs (i1,i2)
00055         else if (action.eq.4) then ! XACTION_DRWABS
00056           call drwabs (i1,i2)
00057         else if (action.eq.5) then ! XACTION_DSHSTYLE
00058           idash= i1
00059         else if (action.eq.6) then ! XACTION_DSHABS
00060           call dshabs (i1,i2,idash)
00061         else if (action.eq.7) then ! XACTION_PNTABS
00062           call pntabs (i1,i2)
00063         else if (action.eq.8) then ! XACTION_GTEXT
00064           iprntlen= i1
00065           if (iprntlen.gt.tcs_messagelen) iprntlen= tcs_messagelen
00066           txtstring(1:1)= char(i2)
00067           if (iprntlen.eq.1) then
00068             txtstring= txtstring(1:1) // char(0)
00069             call toutstc (txtstring)
00070           else
00071             iactlen= 1
00072           end if
00073         else if (action.eq.9) then ! XACTION_ASCII
00074           if (iactlen.lt.iprntlen) then
00075             iactlen= iactlen+1
00076             txtstring(iactlen:iactlen)= char(i1)
00077           end if
00078           if (iactlen.lt.iprntlen) then
00079             iactlen= iactlen+1
```

```
          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
```

7.23 Mainpage.dox File Reference

7.24 PlotHDC.f03 File Reference

Utility: Plot Journalfiles.

Functions/Subroutines

- program plothdc

7.24.1 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 ISO Fortran 2003 intrinsic procedures.

Note

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

Definition in file PlotHDC.f03.

7.24.2 Function/Subroutine Documentation

7.24.2.1 plothdc()

```
program plothdc
```

Definition at line 26 of file PlotHDC.f03.

7.25 PlotHDC.f03

```
00001 !> \file       PlotHDC.f03
00002 !> \brief      Utility: Plot Journalfiles
00003 !> \version    1.0-GCC
00004 !> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 !> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 !>
00007 !> \~german
00008 !> Hilfsprogramm zur Anzeige von Journal-Hardcopies von SDL2 und wX-Programmen.
00009 !> Diese koennen dann ueber Cut/Paste in andere Windowsprogramme uebernommen werden.
00010 !> Die Abfrage der Programmparameter erfolgt durch ISO-Fortran 2003 Intrinsics.
00011 !> \note \verbatim
00012 !>    Aufruf durch:
00013 !>     $> plothdc FileName
00014 !> \endverbatim
00015 !>
00016 !> \~english
00017 !> Utility to draw journal-hardcopies from SDL2 and wX programs.
00018 !> With cut/paste they could be used by other MS-win programs.
00019 !> Program parameters are optained by calling ISO Fortran 2003 intrinsic procedures.
00020 !> \note \verbatim
00021 !>    Invoke by:
00022 !>     $> plothdc FileName
00023 !> \endverbatim
00024 !> \~
00025 !>
00026       program plothdc
00027       implicit none
00028       integer itrimlen
00029       integer ipar
00030       character * 128 filnam
00031
00032       call initt (0)
00033       ipar = command_argument_count() ! FTN03 Standard
00034       call get_command_argument (1,filnam)
00035       if (ipar.gt.0) then
00036         call gethdc (filnam(1:itrimlen(filnam))//char(0))
00037       else
00038         call graphicerror (9, 'Please invoke by: PlotHDC FileName')
00039       end if
00040       call finitt
00041       end
```

7.26 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)

7.26.1 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.

7.26.2 Function/Subroutine Documentation

7.26.2.1 istringlen()

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

Definition at line 94 of file Strings.for.

7.26.2.2 itrimlen()

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

Definition at line 133 of file Strings.for.

7.26.2.3 printstring()

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

Definition at line 114 of file Strings.for.

7.26.2.4 substitute()

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

Definition at line 30 of file Strings.for.

7.27 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
Cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      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
```

```
C
      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

```

7.28 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)

7.28.1 Detailed Description

TCS: Tektronix Plot 10 Emulation.

Version

4.0

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

System independent subroutines

Definition in file TCS.for.

7.28.2 Function/Subroutine Documentation

7.28.2.1 ancho()

```
subroutine ancho (
             ichar )
```

Definition at line 315 of file TCS.for.

7.28.2.2 anstr()

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

Definition at line 305 of file TCS.for.

7.28.2.3 baksp()

```
subroutine baksp
```

Definition at line 360 of file TCS.for.

7.28.2.4 cartn()

```
subroutine cartn
```

Definition at line 341 of file TCS.for.

7.28.2.5 dasha()

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

Definition at line 266 of file TCS.for.

7.28.2.6 dashr()

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

Definition at line 212 of file TCS.for.

7.28.2.7 drawa()

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

Definition at line 233 of file TCS.for.

7.28.2.8 drawr()

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

Definition at line 188 of file TCS.for.

7.28.2.9 dwindo()

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

Definition at line 438 of file TCS.for.

7.28.2.10 genflg()

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

Definition at line 534 of file TCS.for.

7.28.2.11 home()

```
subroutine home
```

Definition at line 494 of file TCS.for.

7.28.2.12 linef()

```
subroutine linef
```

Definition at line 350 of file TCS.for.

7.28.2.13 linhgt()

```
function linhgt (
              Numlin )
```

Definition at line 376 of file TCS.for.

7.28.2.14 lintrn()

```
subroutine lintrn
```

Definition at line 394 of file TCS.for.

7.28.2.15 linwdt()

```
function linwdt (
              NumChr )
```

Definition at line 384 of file TCS.for.

7.28.2.16 logtrn()

```
subroutine logtrn (
              IMODE )
```

Definition at line 404 of file TCS.for.

7.28.2.17 movea()

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

Definition at line 244 of file TCS.for.

7.28.2.18 mover()

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

Definition at line 196 of file TCS.for.

7.28.2.19 newlin()

```
subroutine newlin
```

Definition at line 333 of file TCS.for.

7.28.2.20 newpag()

```
subroutine newpag
```

Definition at line 368 of file TCS.for.

7.28.2.21 pointa()

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

Definition at line 255 of file TCS.for.

7.28.2.22 pointr()

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

Definition at line 204 of file TCS.for.

7.28.2.23 rel2ab()

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

Definition at line 220 of file TCS.for.

7.28.2.24 rescal()

```
subroutine rescal
```

Definition at line 457 of file TCS.for.

7.28.2.25 revcot()

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

Definition at line 290 of file TCS.for.

7.28.2.26 rrotat()

```
subroutine rrotat (
              Grad )
```

Definition at line 477 of file TCS.for.

7.28.2.27 rscale()

```
subroutine rscale (
              Faktor )
```

Definition at line 486 of file TCS.for.

7.28.2.28 seetrm()

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

Definition at line 512 of file TCS.for.

7.28.2.29 seetrn()

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

Definition at line 523 of file TCS.for.

7.28.2.30 setmrg()

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

Definition at line 503 of file TCS.for.

7.28.2.31 swindo()

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

Definition at line 426 of file TCS.for.

7.28.2.32 twindo()

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

Definition at line 419 of file TCS.for.

7.28.2.33 vcursr()

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

Definition at line 178 of file TCS.for.

7.28.2.34 vwindo()

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

Definition at line 445 of file TCS.for.

7.28.2.35 wincot()

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

Definition at line 277 of file TCS.for.

7.29 TCS.for

```
00001 C> \file       TCS.for
00002 C> \brief      TCS: Tektronix Plot 10 Emulation
00003 C> \version    4.0
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Systemübergreifende TCS-Routinen
00008 C> \~english
00009 C> System independent subroutines
00010 C> \~
00011 C
00012 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00013 C
00014 C      27.11.20 Version 4.0:
00015 C               Einheitliche Version CPM/DOS/Windows/SDL2
00016 C
00017 C      17.08.20 Version 3.2
00018 C               Harmonisierung der Verwendung des Commonblocks TKTRNX
00019 C               Variable KHOMEY wird jetzt (analog alter DOS-Version) verwendet.
00020 C               Da KHOMEY nicht in der CP/M Version vorhanden ist, muss ab dieser
00021 C               Version fuer eine Complilation unter CP/M die entsprechende Zeile
00022 C               in der SUBROUTINE HOME geändert werden.
00023 C
00024 C      13.11.17 Version 3.1
00025 C               Anpassung an OpenWatcom 2.0
00026 C               Bugfix: Unterscheidung Aufrufe ueber windowsx.h (win16) und GDI (win32)
00027 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      Änderungen gegenüber 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       Zugehörige 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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   Anpassungen an Microsoft-Windows:
C
C       Änderungen gegenüber DOS-Version:
C               INITT befinden sich jetzt in TCSdrWIN.FOR bzw. TCSinitt.FOR
C
C       Zugehörige 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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C   Anpassungen an SDL2:
C
C       Änderungen gegenüber Windows-Version:
C               Fehlerausgabe in den Windows-Debug-Channel (bzw. *ix Fehlerkanal)
C               Statusfenster analog DOS nur einzeilig ohne Scrollmöglichkeit
C
C       Zugehörige 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


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

```

7.30 TCSdrSDL.for File Reference

SDL Port: High-Level Driver.

Functions/Subroutines

- subroutine tcslev (LEVEL)
- subroutine initt (iDummy)

  Initialisierung Hard- und Software.
- subroutine initt2
- 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 tinput (iChr)
- subroutine anmode

  Entry Dummyroutinen.
- logical function winselect (iDummy)

7.30.1 Detailed Description

SDL Port: High-Level Driver.

Version

(2022,305,6)

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

SDL2 specific subroutines

Note

```
Supplement to Tektronix:
 subroutine TOUTSTC (String): Ausgabe Fortran-String
 subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
 subroutine TXTCOL (iCol): Setzen Textfarbe
 subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
 subroutine DefaultColour: Wiederherstellung Defaultfarben
```

Definition in file TCSdrSDL.for.

7.30.2 Function/Subroutine Documentation

7.30.2.1 anmode()

```
subroutine anmode
```

Entry Dummyroutinen.

AlfMod

pClipt

alpha

Definition at line 219 of file TCSdrSDL.for.

7.30.2.2 drwrel()

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

Definition at line 132 of file TCSdrSDL.for.

7.30.2.3 dshrel()

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

Definition at line 142 of file TCSdrSDL.for.

7.30.2.4 initt()

```
subroutine initt (
             iDummy )
```

Initialisierung Hard- und Software.

Definition at line 50 of file TCSdrSDL.for.

7.30.2.5 initt2()

```
subroutine initt2
```

Definition at line 62 of file TCSdrSDL.for.

7.30.2.6 movrel()

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

Definition at line 112 of file TCSdrSDL.for.

7.30.2.7 pntrel()

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

Definition at line 122 of file TCSdrSDL.for.

7.30.2.8 restat()

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

Definition at line 94 of file TCSdrSDL.for.

7.30.2.9 seeloc()

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

Definition at line 156 of file TCSdrSDL.for.

7.30.2.10 statst()

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

Definition at line 196 of file TCSdrSDL.for.

7.30.2.11 svstat()

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

Definition at line 81 of file TCSdrSDL.for.

7.30.2.12 tcslev()

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

Definition at line 37 of file TCSdrSDL.for.

7.30.2.13 tinput()

```
subroutine tinput (
             iChr )
```

Definition at line 208 of file TCSdrSDL.for.

7.30.2.14 toutpt()

```
subroutine toutpt (
             iChr )
```

Definition at line 169 of file TCSdrSDL.for.

7.30.2.15 toutst()

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

Definition at line 177 of file TCSdrSDL.for.

7.30.2.16 toutstc()

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

Definition at line 188 of file TCSdrSDL.for.

7.30.2.17 winselect()

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

Definition at line 231 of file TCSdrSDL.for.

7.31 TCSdrSDL.for

```
C> \file       TCSdrSDL.for
C> \brief      SDL Port: High-Level Driver
C> \version    (2022,305,6)
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C>
C> \~german
C> SDL2-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> SDL2 specific subroutines
C> \note \verbatim
C>    Supplement to 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
C  Ausgabe der Softwareversion
C
      subroutine tcslev(LEVEL)
      integer LEVEL(3)
      level(1)=2022      ! Aenderungsjahr
      level(2)= 305      ! Aenderungstag
      level(3)=   6      ! System= SDL
      return
      end



C
C>  Initialisierung Hard- und Software
C
      subroutine initt (iDummy)
      include 'Tktrnx.fd'
      call initt1 ! Init Hardware
      call initt2 ! Reset Common TKTRNX ohne Einfluss auf das Journal
      call nrmsiz
      call italir
      call home
      return
      end



      subroutine initt2
C INITT2 auch durch RepaintBuffer aufgerufen -> Schreiben Journal unmoeglich!
      include 'Tktrnx.fd'
      call lintrn
      call swindo (0,1023,0,780)
      call vwindo (0.,1023.,0.,780.)
      call rrotat (0.)
      call rscale (1.)
      call setmrg (0,1023)
      return
      end




C
C  Abspeichern Terminal Status Area (wie MS Windows und 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 MS Windows und 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 (wie MS Windows, DOS)
C

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



C
C  Textausgabe
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  Eingabe
C

      subroutine tinput (iChr)
      call dcursr (ichr, ichr,ichr)
C     Aufruf von DCURSR mit ix=iy: Maustasten ausser Funktion
      return
      end



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



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

```

7.32 TCSdSDLc.c File Reference

SDL Port: Low-Level Driver.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include "SDL.h"
#include "SDL_ttf.h"
#include "SDL_audio.h"
#include "mxml.h"
#include "sglib.h"
#include "TCSdSDLc.h"
#include "TKTRNX.h"
```

Classes

- struct xJournalEntry_typ

Macros

- #define INIFILEXT ".xml"
- #define FNTFILEXT ".ttf"
- #define AUDIOSUPPORT
- #define HIGHQUALCHAR
- #define LOGLEVEL SDL_LOG_PRIORITY_ERROR
- #define MAX_COLOR_INDEX 15
- #define TMPSTRLEN TCS_FILE_NAMELEN

Typedefs

- typedef char ErrMsg[TCS_MESSAGELEN]

Functions

- int HiResX (FTNINT iX)
- int HiResY (FTNINT iY)
- int LoResX (FTNINT iX)
- int LoResY (FTNINT iY)
- bool PointInWindow (FTNINT ix1, FTNINT iy1)
- bool ClipLineStart (FTNINT ix1, FTNINT iy1, FTNINT ix2, FTNINT iy2, FTNINT ∗isx, FTNINT ∗isy)
- void DrawHiResDashLine (FTNINT ix, FTNINT iy, FTNINT ix2, FTNINT iy2, FTNINT ∗iMask)
- void PlotText (const char ∗outtxt)
- void RepaintBuffer ()
- void TCSGraphicError (int iErr, const char ∗msg)
- int TCSEventFilter (void ∗UserData, SDL_Event ∗event)
- void audio_callback (void ∗sample_nr, Uint8 ∗raw_buffer, int bytes)
- void sax_callback (mxml_node_t ∗node, mxml_sax_event_t event, void ∗usr)
- mxml_type_t sax_type_callback (mxml_node_t ∗node)
- void sax_error_callback (char ∗mssg)
- void XMLreadProgPar (const char ∗filname)
- void PresetProgPar ()
- void CustomizeProgPar ()
- void winlbl (FTNSTRPAR ∗PloWinNam, FTNSTRPAR ∗StatWinNam, FTNSTRPAR ∗IniFilNam FTNSTRPAR_TAIL(Ini←FilNam))
- void initt1 ()
- void finitt ()
- void iowait (void)
- 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 outtext (FTNSTRPAR ∗ftn_string FTNSTRPAR_TAIL(ftn_string))
- void bell (void)
- void GraphicError (FTNINT ∗iErr, FTNSTRPAR ∗ftn_string, FTNINT ∗iL FTNSTRPAR_TAIL(ftn_string))
- void dcursr (FTNINT ∗ic, FTNINT ∗ix, FTNINT ∗iy)
- void hdcopy (void)
- void lib_movc3 (FTNINT ∗len, FTNSTRPAR ∗sou, FTNSTRPAR ∗dst FTNSTRPAR_TAIL(sou) FTNSTRPAR_TAIL(dst))

Variables

- static int TCSEventFilterData
- static float PixFacX
- static float PixFacY
- static bool TCSinitialized = false
- static bool ClippingNotActive = true
- static char szTCSWindowName [TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME
- static char szTCSstatWindowName [TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME
- static char szTCSIniFile [TCS_FILE_NAMELEN] = ""
- static char szTCSHardcopyFile [TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME
- static char szTCSGraphicFont [TCS_FILE_NAMELEN] = TCS_INIDEF_FONT
- static char szTCSSysFont [TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT
- static char szTCSsect0 [TCS_FILE_NAMELEN] = TCS_INISECT0
- 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 TextLineHeight
- static int TCSDefaultLinCol = TCS_INIDEF_LINCOL
- static int TCSDefaultTxtCol = TCS_INIDEF_TXTCOL
- static int TCSDefaultBckCol = TCS_INIDEF_BCKCOL
- static int iHardcopyCount = 1
- static ErrMsg szTCSErrorMsg [(int) MSG_MAXERRNO+1]
- static int TCSErrorLev [(int) MSG_MAXERRNO+1]
- static SDL_Color sdlColorTable [ ]
- static SDL_Window ∗ TCSwindow = NULL
- static SDL_Renderer ∗ TCSrenderer = NULL
- static TTF_Font ∗ TCSfont = NULL
- static TTF_Font ∗ TCSstatusfont = NULL
- static SDL_Window ∗ TCSstatwindow = NULL
- static SDL_Renderer ∗ TCSstatrenderer = NULL
- static struct xJournalEntry_typ ∗ xTCSJournal = NULL
- static SDL_AudioSpec SDL_AudioDev_optained
- static SDL_AudioSpec SDL_AudioDev_wanted
- static int AudioSample_nr = 0

7.32.1 Detailed Description

SDL Port: Low-Level Driver.

Version

1.5

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

system-specific subroutines of the Tektronix emulation

Note

```
1. If the first letter of the window name is '~', the window will be
   drawn without title and frame.
2. System- and status messages are shown in an one-line window. If
   the height of the window is <= 0, only system errors are signaled
   through the error channel.
3. When called inside a ssh terminal, the Raspberry Pi videodriver
   crashes during the second call of SDL_renderer . If the height of
   the status window is 0, no problem arises.
4. If the parameter HIGHQUALCHAR is defined, textoutput is "Blended".
   Undefining HIGHQUALCHAR on slow systems changes output to "Solid".
```

Definition in file TCSdSDLc.c.

7.32.2 Macro Definition Documentation

7.32.2.1 AUDIOSUPPORT

```
#define AUDIOSUPPORT
```

Definition at line 67 of file TCSdSDLc.c.

7.32.2.2 FNTFILEXT

```
#define FNTFILEXT ".ttf"
```

Definition at line 66 of file TCSdSDLc.c.

7.32.2.3 HIGHQUALCHAR

```
#define HIGHQUALCHAR
```

Definition at line 68 of file TCSdSDLc.c.

7.32.2.4 INIFILEXT

```
#define INIFILEXT ".xml"
```

Definition at line 65 of file TCSdSDLc.c.

7.32.2.5 LOGLEVEL

```
#define LOGLEVEL SDL_LOG_PRIORITY_ERROR
```

Definition at line 75 of file TCSdSDLc.c.

7.32.2.6 MAX_COLOR_INDEX

```
#define MAX_COLOR_INDEX 15
```

Definition at line 226 of file TCSdSDLc.c.

7.32.2.7 TMPSTRLEN

```
#define TMPSTRLEN TCS_FILE_NAMELEN
```

7.32.3 Typedef Documentation

7.32.3.1 ErrMsg

```
typedef char ErrMsg[TCS_MESSAGELEN]
```

Definition at line 147 of file TCSdSDLc.c.

7.32.4 Function Documentation

7.32.4.1 audio_callback()

```
void audio_callback (
             void * sample_nr,
             Uint8 * raw_buffer,
             int bytes )
```

Definition at line 722 of file TCSdSDLc.c.

7.32.4.2 bckcol()

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

Definition at line 1709 of file TCSdSDLc.c.

7.32.4.3 bell()

```
void bell (
             void  )
```

Definition at line 1988 of file TCSdSDLc.c.

7.32.4.4 ClipLineStart()

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

Definition at line 293 of file TCSdSDLc.c.

7.32.4.5 csize()

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

Definition at line 1930 of file TCSdSDLc.c.

7.32.4.6 CustomizeProgPar()

```
void CustomizeProgPar ( )
```

Definition at line 1111 of file TCSdSDLc.c.

7.32.4.7 dblsiz()

```
void dblsiz (
             void )
```

Definition at line 1865 of file TCSdSDLc.c.

7.32.4.8 dcursr()

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

Definition at line 2015 of file TCSdSDLc.c.

7.32.4.9 DefaultColour()

```
void DefaultColour (
             void )
```

Definition at line 1761 of file TCSdSDLc.c.

7.32.4.10 DrawHiResDashLine()

```
void DrawHiResDashLine (
             FTNINT ix,
             FTNINT iy,
             FTNINT ix2,
             FTNINT iy2,
             FTNINT * iMask )
```

Definition at line 360 of file TCSdSDLc.c.

7.32.4.11 drwabs()

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

Definition at line 1597 of file TCSdSDLc.c.

7.32.4.12 dshabs()

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

Definition at line 1636 of file TCSdSDLc.c.

7.32.4.13 erase()

```
void erase (
             void )
```

Definition at line 1527 of file TCSdSDLc.c.

7.32.4.14 finitt()

```
void finitt ( )
```

Definition at line 1465 of file TCSdSDLc.c.

7.32.4.15 GraphicError()

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

Definition at line 2000 of file TCSdSDLc.c.

7.32.4.16 hdcopy()

```
void hdcopy (
             void  )
```

Definition at line 2059 of file TCSdSDLc.c.

7.32.4.17 HiResX()

```
int HiResX (
             FTNINT iX )
```

Definition at line 258 of file TCSdSDLc.c.

7.32.4.18 HiResY()

```
int HiResY (
             FTNINT iY )
```

Definition at line 264 of file TCSdSDLc.c.

7.32.4.19 initt1()

```
void initt1 ( )
```

Definition at line 1258 of file TCSdSDLc.c.

7.32.4.20 iowait()

```
void iowait (
             void  )
```

Definition at line 1504 of file TCSdSDLc.c.

7.32.4.21 italic()

```
void italic (
             void  )
```

Definition at line 1831 of file TCSdSDLc.c.

7.32.4.22 italir()

```
void italir (
             void  )
```

Definition at line 1848 of file TCSdSDLc.c.

7.32.4.23 lib_movc3()

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

Definition at line 2185 of file TCSdSDLc.c.

7.32.4.24 lincol()

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

Definition at line 1726 of file TCSdSDLc.c.

7.32.4.25 LoResX()

```
int LoResX (
             FTNINT iX )
```

Definition at line 270 of file TCSdSDLc.c.

7.32.4.26 LoResY()

```
int LoResY (
             FTNINT iY )
```

Definition at line 276 of file TCSdSDLc.c.

7.32.4.27 movabs()

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

Definition at line 1580 of file TCSdSDLc.c.

7.32.4.28 nrmsiz()

```
void nrmsiz (
             void  )
```

Definition at line 1896 of file TCSdSDLc.c.

7.32.4.29 outgtext()

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

Definition at line 1780 of file TCSdSDLc.c.

7.32.4.30 outtext()

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

Definition at line 1938 of file TCSdSDLc.c.

7.32.4.31 PlotText()

```
void PlotText (
             const char * outtxt )
```
Definition at line 417 of file TCSdSDLc.c.

7.32.4.32 pntabs()

```
void pntabs (
             FTNINT * ix,
             FTNINT * iy )
```
Definition at line 1683 of file TCSdSDLc.c.

7.32.4.33 PointInWindow()

```
bool PointInWindow (
             FTNINT ix1,
             FTNINT iy1 )
```
Definition at line 285 of file TCSdSDLc.c.

7.32.4.34 PresetProgPar()

```
void PresetProgPar ( )
```
Definition at line 1083 of file TCSdSDLc.c.

7.32.4.35 RepaintBuffer()

```
void RepaintBuffer ( )
```
Definition at line 444 of file TCSdSDLc.c.

7.32.4.36 sax_callback()

```
void sax_callback (
             mxml_node_t * node,
             mxml_sax_event_t event,
             void * usr )
```
Definition at line 752 of file TCSdSDLc.c.

7.32.4.37 sax_error_callback()

```
void sax_error_callback (
             char * mssg )
```
Definition at line 1046 of file TCSdSDLc.c.

7.32.4.38 sax_type_callback()

```
mxml_type_t sax_type_callback (
             mxml_node_t * node )
```
Definition at line 1026 of file TCSdSDLc.c.

7.32.4.39 swind1()

```
void swind1 (
             FTNINT * ix1,
             FTNINT * iy1,
             FTNINT * ix2,
             FTNINT * iy2 )
```
Definition at line 1518 of file TCSdSDLc.c.

7.32.4.40 TCSEventFilter()

```
int TCSEventFilter (
             void * UserData,
             SDL_Event * event )
```
Definition at line 686 of file TCSdSDLc.c.

7.32.4.41 TCSGraphicError()

```
void TCSGraphicError (
             int iErr,
             const char * msg )
```
Definition at line 634 of file TCSdSDLc.c.

7.32.4.42 txtcol()

```
void txtcol (
             FTNINT * iCol )
```
Definition at line 1744 of file TCSdSDLc.c.

7.32.4.43 winlbl()

```
void winlbl (
             FTNSTRPAR * PloWinNam,
             FTNSTRPAR * StatWinNam,
             FTNSTRPAR *IniFilNam  FTNSTRPAR_TAILIniFilNam )
```
Definition at line 1162 of file TCSdSDLc.c.

7.32.4.44 XMLreadProgPar()

```
void XMLreadProgPar (
             const char * filname )
```
Definition at line 1059 of file TCSdSDLc.c.

7.32.5 Variable Documentation

7.32.5.1 AudioSample_nr

```
int AudioSample_nr = 0  [static]
```
Definition at line 246 of file TCSdSDLc.c.

7.32.5.2 ClippingNotActive

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

Definition at line 117 of file TCSdSDLc.c.

7.32.5.3 iHardcopyCount

```
int iHardcopyCount = 1  [static]
```

Definition at line 139 of file TCSdSDLc.c.

7.32.5.4 PixFacX

```
float PixFacX  [static]
```

Definition at line 114 of file TCSdSDLc.c.

7.32.5.5 PixFacY

```
float PixFacY  [static]
```

Definition at line 114 of file TCSdSDLc.c.

7.32.5.6 SDL_AudioDev_optained

```
SDL_AudioSpec SDL_AudioDev_optained  [static]
```

Definition at line 243 of file TCSdSDLc.c.

7.32.5.7 SDL_AudioDev_wanted

```
SDL_AudioSpec SDL_AudioDev_wanted  [static]
```

Definition at line 244 of file TCSdSDLc.c.

7.32.5.8 sdlColorTable

```
SDL_Color sdlColorTable[]  [static]
```

Initial value:

```
= {
                {240,240,240,SDL_ALPHA_OPAQUE },
                {  0,  0,  0,SDL_ALPHA_OPAQUE },
                {240, 80, 80,SDL_ALPHA_OPAQUE },
                { 80,240, 80,SDL_ALPHA_OPAQUE },
                { 80,240,240,SDL_ALPHA_OPAQUE },
                { 80, 80,240,SDL_ALPHA_OPAQUE },
                {240,240, 80,SDL_ALPHA_OPAQUE },
                {160,160,160,SDL_ALPHA_OPAQUE },
                {240, 80,240,SDL_ALPHA_OPAQUE },
                {160,  0,  0,SDL_ALPHA_OPAQUE },
                {  0,160,  0,SDL_ALPHA_OPAQUE },
                {  0,  0,160,SDL_ALPHA_OPAQUE },
                {  0,160,160,SDL_ALPHA_OPAQUE },
                {160, 80,  0,SDL_ALPHA_OPAQUE },
                { 80, 80, 80,SDL_ALPHA_OPAQUE },
                {160,  0,160,SDL_ALPHA_OPAQUE }
             }
```

Definition at line 208 of file TCSdSDLc.c.

7.32.5.9 szTCSErrorMsg

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

Initial value:

```
=
                {"Element 0 unused","DOS",
                TCS_INIDEF_UNKNGRAPHCARD,
                TCS_INIDEF_NOFNTFIL,
                TCS_INIDEF_NOFNT,
                "DOS",
                TCS_INIDEF_HDCOPN,
                TCS_INIDEF_HDCWRT,
                TCS_INIDEF_HDCINT,
                TCS_INIDEF_USR,
                TCS_INIDEF_HDCACT,
                TCS_INIDEF_USRWRN,
                TCS_INIDEF_EXIT,
                "Windows",
                "Windows",
                TCS_INIDEF_JOUCREATE,
                TCS_INIDEF_JOUENTRY,
                TCS_INIDEF_JOUADD,
                TCS_INIDEF_JOUCLR,
                TCS_INIDEF_JOUUNKWN,
                TCS_INIDEF_XMLPARSER,
                TCS_INIDEF_XMLOPEN,
                TCS_INIDEF_UNKNAUDIO,
                TCS_INIDEF_USR2,
                TCS_INIDEF_INI2,
                "Maxerr only for internal Use" }
```

Definition at line 148 of file TCSdSDLc.c.

7.32.5.10 szTCSGraphicFont

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

Definition at line 123 of file TCSdSDLc.c.

7.32.5.11 szTCSHardcopyFile

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

Definition at line 122 of file TCSdSDLc.c.

7.32.5.12 szTCSIniFile

`char szTCSIniFile[TCS_FILE_NAMELEN] = ""  [static]`

Definition at line 121 of file TCSdSDLc.c.

7.32.5.13 szTCSsect0

`char szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0  [static]`

Definition at line 125 of file TCSdSDLc.c.

7.32.5.14 szTCSstatWindowName

`char szTCSstatWindowName[TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME  [static]`

Definition at line 120 of file TCSdSDLc.c.

7.32.5.15 szTCSSysFont

`char szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT  [static]`

Definition at line 124 of file TCSdSDLc.c.

7.32.5.16 szTCSWindowName

`char szTCSWindowName[TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME  [static]`

Definition at line 119 of file TCSdSDLc.c.

7.32.5.17 TCSDefaultBckCol

```
int TCSDefaultBckCol = TCS_INIDEF_BCKCOL  [static]
```

Definition at line 138 of file TCSdSDLc.c.

7.32.5.18 TCSDefaultLinCol

```
int TCSDefaultLinCol = TCS_INIDEF_LINCOL  [static]
```

Definition at line 136 of file TCSdSDLc.c.

7.32.5.19 TCSDefaultTxtCol

```
int TCSDefaultTxtCol = TCS_INIDEF_TXTCOL  [static]
```

Definition at line 137 of file TCSdSDLc.c.

7.32.5.20 TCSErrorLev

```
int TCSErrorLev[(int) MSG_MAXERRNO+1]  [static]
```

Initial value:

```
=
                {10,10,
                TCS_INIDEF_UNKNGRAPHCARDL,
                TCS_INIDEF_NOFNTFILL,
                TCS_INIDEF_NOFNTL,
                10,
                TCS_INIDEF_HDCOPNL,
                TCS_INIDEF_HDCWRTL,
                TCS_INIDEF_HDCINTL,
                TCS_INIDEF_USRL,
                TCS_INIDEF_HDCACTL,
                TCS_INIDEF_USRWRNL,
                TCS_INIDEF_EXITL,
                10,
                10,
                TCS_INIDEF_JOUCREATEL,
                TCS_INIDEF_JOUENTRYL,
                TCS_INIDEF_JOUADDL,
                TCS_INIDEF_JOUCLRL,
                TCS_INIDEF_JOUUNKWNL,
                TCS_INIDEF_XMLPARSERL,
                TCS_INIDEF_XMLOPENL,
                TCS_INIDEF_UNKNAUDIOL,
                TCS_INIDEF_USR2L,
                TCS_INIDEF_INI2L,
                10}
```

Definition at line 175 of file TCSdSDLc.c.

7.32.5.21 TCSEventFilterData

```
int TCSEventFilterData  [static]
```

Definition at line 112 of file TCSdSDLc.c.

7.32.5.22 TCSfont

```
TTF_Font* TCSfont = NULL  [static]
```

Definition at line 231 of file TCSdSDLc.c.

7.32.5.23 TCSinitialized

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

Definition at line 116 of file TCSdSDLc.c.

7.32.5.24 TCSrenderer

```
SDL_Renderer* TCSrenderer = NULL [static]
```

Definition at line 230 of file TCSdSDLc.c.

7.32.5.25 TCSstatrenderer

```
SDL_Renderer* TCSstatrenderer = NULL [static]
```

Definition at line 235 of file TCSdSDLc.c.

7.32.5.26 TCSstatusfont

```
TTF_Font* TCSstatusfont = NULL [static]
```

Definition at line 232 of file TCSdSDLc.c.

7.32.5.27 TCSstatwindow

```
SDL_Window* TCSstatwindow = NULL [static]
```

Definition at line 234 of file TCSdSDLc.c.

7.32.5.28 TCSstatWindowIniXrelpos

```
int TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX [static]
```

Definition at line 131 of file TCSdSDLc.c.

7.32.5.29 TCSstatWindowIniXrelsiz

```
int TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX [static]
```

Definition at line 133 of file TCSdSDLc.c.

7.32.5.30 TCSstatWindowIniYrelpos

```
int TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY [static]
```

Definition at line 132 of file TCSdSDLc.c.

7.32.5.31 TCSstatWindowIniYrelsiz

```
int TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY [static]
```

Definition at line 134 of file TCSdSDLc.c.

7.32.5.32 TCSwindow

```
SDL_Window* TCSwindow = NULL [static]
```

Definition at line 229 of file TCSdSDLc.c.

7.32.5.33 TCSwindowIniXrelpos

```
int TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX  [static]
```

Definition at line 127 of file TCSdSDLc.c.

7.32.5.34 TCSwindowIniXrelsiz

```
int TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX  [static]
```

Definition at line 129 of file TCSdSDLc.c.

7.32.5.35 TCSwindowIniYrelpos

```
int TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY  [static]
```

Definition at line 128 of file TCSdSDLc.c.

7.32.5.36 TCSwindowIniYrelsiz

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

Definition at line 130 of file TCSdSDLc.c.

7.32.5.37 TextLineHeight

```
int TextLineHeight  [static]
```

Definition at line 135 of file TCSdSDLc.c.

7.32.5.38 xTCSJournal

```
struct xJournalEntry_typ* xTCSJournal = NULL  [static]
```

Definition at line 240 of file TCSdSDLc.c.

7.33 TCSdSDLc.c

```
/** ***************************************************************************
\file       TCSdSDLc.c
\brief      SDL Port: Low-Level Driver
\version    1.5
\author     (C) 2023 Dr.-Ing. Klaus Friedewald
\copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
\~german
         Systemnahe Graphikroutinen für die Tektronix Emulation
\note \verbatim
          1. Falls der erste Buchstabe des Fensternamens ein '~' ist, wird
             das betreffende Fenster ohne Titel und Rahmen gezeichnet.
          2. Die System- und Statusmeldungen erfolgen in einem eigenen
             einzeiligem Fenster. Falls die Statusfensterhöhe <= 0 ist,
             erfolgen nur noch Systemfehlermeldungen über den Error-Channel.
          3. Der Videotreiber des Raspberry Pi4 kann über SSH keine zwei
             unabhängige Renderer für die beiden Fenster verwalten. Jedoch
             liefert der zweite Aufruf von SDL_CreateRenderer für das
             Statusfenster keinen Errorcode, sondern führt zu einem Programm-
             absturz. Entweder MUSS hier die Statusfensterhöhe <= 0 gesetzt
             oder X11 gestartet sein.
          4. Durch den Parameter HIGHQUALCHAR erfolgt die Textausgabe "Blended".
             Zur Performancesteigerung kann bei leistungsschwachen Systemen
             durch Auskommentieren auf "Solid" gewechselt werden.
\endverbatim
\~english
         system-specific subroutines of the Tektronix emulation
\note \verbatim
          1. If the first letter of the window name is '~', the window will be
             drawn without title and frame.
          2. System- and status messages are shown in an one-line window. If
             the height of the window is <= 0, only system errors are signaled
             through the error channel.
          3. When called inside a ssh terminal, the Raspberry Pi videodriver
```

```
              crashes during the second call of SDL_renderer . If the height of
              the status window is 0, no problem arises.
           4. If the parameter HIGHQUALCHAR is defined, textoutput is "Blended".
              Undefining HIGHQUALCHAR on slow systems changes output to "Solid".
\endverbatim
\~
****************************************************************************** */

/*
        Anmerkungen:
          1. In der Routine WINLBL werden die SDL-Funktion SDL_GetBasePath ()
             sowie SDL_free verwendet. In der Dokumentation ist jedoch nicht
             explizit beschrieben, dass diese Funktion immer (wie SDL_logxxx)
             bereits vor dem Aufruf von SDL_Init() funktioniert. Die in der
             Source herauskommentierten Zeilen
             SDL_Init (0); und SDL_Quit(); koennen dann bei Problemen wieder
             verwendet werden.
          2. Skalierung vom Tektronix- auf das Bildschirmkoordinatensystem muss
             von Hand erfolgen, da SDL_RenderSetLogicalSize nicht durchgängig
             implementiert ist (Bug bis SDL2 Version 2.0.5 verifiziert).
             Insbesondere verwendet DrawLine die Skalierung nicht bei geneigten
             Geraden.
          3. Journalfile wird verwendet um Hardcopies erzeugen zu können

*/


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

#define INIFILEXT ".xml"
#define FNTFILEXT ".ttf"
#define AUDIOSUPPORT
#define HIGHQUALCHAR


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

#define LOGLEVEL   SDL_LOG_PRIORITY_ERROR
// #define LOGLEVEL   SDL_LOG_PRIORITY_DEBUG
// #define LOGLEVEL   SDL_LOG_PRIORITY_VERBOSE // Ausgaben < Error in Fehlerkanal
// #define TRACE_CALLS // zusaetzliche Debugausgaben


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

#include <stdlib.h>
#include <string.h>
#include <stdio.h> // Fuer HDCOPY: sprintf

#ifdef AUDIOSUPPORT
 #include <math.h>
#endif

#include "SDL.h"
#include "SDL_ttf.h"

#ifdef AUDIOSUPPORT
 #include "SDL_audio.h"
#endif

#include "mxml.h"

#include "sglib.h"

#include "TCSdSDLc.h"
#include "TKTRNX.h"


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

static int      TCSEventFilterData; // Userdata, z.Zt. nicht verwendet

static  float   PixFacX, PixFacY; // Anpassung Bildschirmauflösung

static  bool    TCSinitialized = false,
                ClippingNotActive = true;

static char     szTCSWindowName[TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME,
                szTCSstatWindowName[TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME,
                szTCSIniFile[TCS_FILE_NAMELEN] = "",
                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;

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,
                TextLineHeight,
                TCSDefaultLinCol = TCS_INIDEF_LINCOL,
                TCSDefaultTxtCol = TCS_INIDEF_TXTCOL,
                TCSDefaultBckCol = TCS_INIDEF_BCKCOL,
                iHardcopyCount = 1;  // Zähler zur Erzeugung Filenamen



/*
  Zuordnung Fehlernummern zu Meldungen
*/

typedef char   ErrMsg[TCS_MESSAGELEN];
static  ErrMsg  szTCSErrorMsg[(int) MSG_MAXERRNO+1] =
                {"Element 0 unused","DOS",
                TCS_INIDEF_UNKNGRAPHCARD, // Errno 2
                TCS_INIDEF_NOFNTFIL,      // Errno 3
                TCS_INIDEF_NOFNT,         // Errno 4
                "DOS",
                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
                "Windows",
                "Windows",
                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
                TCS_INIDEF_UNKNAUDIO,     // Errno 22
                TCS_INIDEF_USR2,          // Errno 23
                TCS_INIDEF_INI2,          // Errno 24
                "Maxerr only for internal Use" };

static  int      TCSErrorLev[(int) MSG_MAXERRNO+1] =
                {10,10,
                TCS_INIDEF_UNKNGRAPHCARDL,// Errno 2
                TCS_INIDEF_NOFNTFILL,     // Errno 3
                TCS_INIDEF_NOFNTL,        // Errno 4
                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
                10,
                10,
                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
                TCS_INIDEF_UNKNAUDIOL,    // Errno 22
                TCS_INIDEF_USR2L,         // Errno 23
                TCS_INIDEF_INI2L,         // Errno 24
                10};



/*
  Zuordnung der Farbennummern zur VGA-Palette
*/

```

```
static  SDL_Color sdlColorTable[] = {
                {240,240,240,SDL_ALPHA_OPAQUE }, /* iCol= 00: weiss (DOS: 01) */
                {  0,  0,  0,SDL_ALPHA_OPAQUE }, /* iCol= 01: schwarz(DOS:00) */
                {240, 80, 80,SDL_ALPHA_OPAQUE }, /* iCol= 02: rot             */
                { 80,240, 80,SDL_ALPHA_OPAQUE }, /* iCol= 03: gruen           */
                { 80,240,240,SDL_ALPHA_OPAQUE }, /* iCol= 04: blau            */
                { 80, 80,240,SDL_ALPHA_OPAQUE }, /* iCol= 05: lila            */
                {240,240, 80,SDL_ALPHA_OPAQUE }, /* iCol= 06: gelb            */
                {160,160,160,SDL_ALPHA_OPAQUE }, /* iCol= 07: grau            */
                {240, 80,240,SDL_ALPHA_OPAQUE }, /* iCol= 08: violett         */
                {160,  0,  0,SDL_ALPHA_OPAQUE }, /* iCol= 09: mattrot         */
                {  0,160,  0,SDL_ALPHA_OPAQUE }, /* iCol= 10: mattgruen       */
                {  0,  0,160,SDL_ALPHA_OPAQUE }, /* iCol= 11: mattblau        */
                {  0,160,160,SDL_ALPHA_OPAQUE }, /* iCol= 12: mattlila        */
                {160, 80,  0,SDL_ALPHA_OPAQUE }, /* iCol= 13: orange          */
                { 80, 80, 80,SDL_ALPHA_OPAQUE }, /* iCol= 14: mattgrau        */
                {160,  0,160,SDL_ALPHA_OPAQUE }  /* iCol= 15: mattviolett     */
             };
#define MAX_COLOR_INDEX 15


static  SDL_Window *TCSwindow = NULL;
static  SDL_Renderer *TCSrenderer = NULL;
static  TTF_Font* TCSfont = NULL;
static  TTF_Font* TCSstatusfont = NULL;

static  SDL_Window *TCSstatwindow = NULL;
static  SDL_Renderer *TCSstatrenderer = NULL;

struct xJournalEntry_typ {struct xJournalEntry_typ * previous;
                          struct xJournalEntry_typ * next;
                          FTNINT action; FTNINT i1; FTNINT i2;};
 static struct xJournalEntry_typ* xTCSJournal = NULL;

#ifdef AUDIOSUPPORT
 static SDL_AudioSpec      SDL_AudioDev_optained;
 static SDL_AudioSpec      SDL_AudioDev_wanted;

 static int                AudioSample_nr = 0;
#endif





// ---------------- interne Unterprogramme -------------------------------------


/* --- Anpassung der Zeichenaufloesung an die Bildschirme --- */

int HiResX(FTNINT iX)
{
    return (PixFacX*iX) +0.25f;
}


int HiResY(FTNINT iY)
{
    return (PixFacY*iY)+0.25f;
}


int LoResX(FTNINT iX)
{
    return (int)( ( (float)iX/PixFacX) +0.25f );
}


int LoResY(FTNINT iY)
{
    return (int)( ((float)iY/PixFacY)+0.25f );
}



/* --- Clippingroutinen --- */

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;
}

/* Zeichnen einer gestrichelten Linie in den Backbuffer */

void DrawHiResDashLine (FTNINT ix,FTNINT iy, FTNINT ix2,FTNINT iy2,FTNINT *iMask)
{
FTNINT ixx,iyy, ixx2,iyy2;
float xx,yy, dx,dy, dLin,dBlank;

    if (*iMask <= 0) {
     dLin= 10., dBlank=0.; // solid
    } else if (*iMask == 1) {
     dLin= 1.; dBlank=1.; // dotted
    } else if (*iMask == 2) {
     dLin= 3.; dBlank=1.; //  substitute dashed-dotted
    } else if (*iMask == 3) {
     dLin= 3.; dBlank=3.; //  dashed
    } else {
     dLin= 3., dBlank=3.; // unrecognized -> dashed
    }

    if (abs(ix2-ix) >= abs(iy2-iy)) {
     dx= ix2 >= ix ?  3. : -3.;
     dy= ((float)(iy2-iy))/((float)(ix2-ix))*dx;

     xx= (float)ix; yy= (float)iy;
     while (dx != 0.) {
      ixx= (FTNINT) xx; iyy= (FTNINT) yy;
      ixx2=(FTNINT) (xx+dLin*dx); iyy2=(FTNINT) (yy+dLin*dy);
      xx+= (dLin+dBlank)*dx; yy+= (dLin+dBlank)*dy;
      if (   ((dx>=0.) && ((FTNINT)xx>=ix2) )
          || ((dx<=0.) && ((FTNINT)xx<=ix2) )   ) {
       ixx2= ix2; iyy2= iy2;
       dx= 0.;
      }
      SDL_RenderDrawLine(TCSrenderer, HiResX(ixx),HiResY(TEK_YMAX-iyy),
                                      HiResX(ixx2),HiResY(TEK_YMAX-iyy2));
     }

    } else {
     dy= iy2 >= iy ?  3. : -3.;
     dx= ((float)(ix2-ix))/((float)(iy2-iy))*dy;

     xx= (float)ix; yy= (float)iy;
     while (dy != 0.) {
      ixx= (FTNINT) xx; iyy= (FTNINT) yy;
      ixx2=(FTNINT) (xx+dLin*dx); iyy2=(FTNINT) (yy+dLin*dy);
      xx+= (dLin+dBlank)*dx; yy+= (dLin+dBlank)*dy;
      if (   ((dy>=0.) && ((FTNINT)yy>=iy2) )
          || ((dy<=0.) && ((FTNINT)yy<=iy2) )   ) {
       ixx2= ix2; iyy2= iy2;
       dy= 0.;
      }
      SDL_RenderDrawLine(TCSrenderer, HiResX(ixx), HiResY(TEK_YMAX-iyy),
                                      HiResX(ixx2), HiResY(TEK_YMAX-iyy2));
     }
    }
}



void PlotText (const char *outtxt)
{
SDL_Rect dstrect;
SDL_Surface* surface;
SDL_Texture* texture;

#ifdef HIGHQUALCHAR
    surface = TTF_RenderUTF8_Blended(TCSfont, outtxt, sdlColorTable[TKTRNX.iTxtCol]);
#else
    surface = TTF_RenderUTF8_Solid(TCSfont, outtxt, sdlColorTable[TKTRNX.iTxtCol]);
#endif
    texture = SDL_CreateTextureFromSurface(TCSrenderer, surface);

    SDL_QueryTexture(texture, NULL, NULL, &dstrect.w, &dstrect.h);
    dstrect.x= HiResX(TKTRNX.kBeamX);
    dstrect.y= HiResY(TEK_YMAX-TKTRNX.kBeamY)-dstrect.h;

    SDL_RenderCopy(TCSrenderer, texture, NULL, &dstrect);

    SDL_DestroyTexture(texture);
    SDL_FreeSurface(surface);

    TKTRNX.kBeamX= TKTRNX.kBeamX + LoResX(dstrect.w);
}



void RepaintBuffer () // Hier nicht GraphicError verwenden(Rekursionsschleifen)!
{
FTNINT DashStyle;
int wx, wz, iStringLen, iStringActual;
char szString [TCS_MESSAGELEN+1];
struct xJournalEntry_typ *xJournalEntry;

#ifdef TRACE_CALLS
    SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> called");
#endif

    DashStyle= 0; // Vorbesetzung nur notwendig bei fehlerhaftem Journal
    iStringActual= 0; // Zahler Einlesen String ueber XACTION_ASCII
    SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
                                      , sdlColorTable[TKTRNX.iBckCol].g
                                      , sdlColorTable[TKTRNX.iBckCol].b
                                      , sdlColorTable[TKTRNX.iBckCol].a);
    SDL_RenderClear (TCSrenderer); // Backbuffer nach RenderPresent undefiniert

 #ifdef TRACE_CALLS
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p", xTCSJournal);
 #endif
    SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, xTCSJournal, previous, next, xJournalEntry)
    while (xJournalEntry != NULL) {
 #ifdef TRACE_CALLS
```

```
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p", xTCSJournal);
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> Current Entry: Ptr= %p / previous: Ptr= %p / next: Ptr= %p",
                    xJournalEntry, xJournalEntry->previous, xJournalEntry->next);
    SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_??? = %i (i1= %i, i2= %i)",
                  xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2 );
#endif
    switch (xJournalEntry->action) {
      case XACTION_INITT: {
       TKTRNX.iLinCol= TCSDefaultLinCol;
       TKTRNX.iTxtCol= TCSDefaultTxtCol;
       TKTRNX.iBckCol= TCSDefaultBckCol;

       INITT2(); // Reset TKTRNX (Margin, Scale...)

       TKTRNX.ksizef = 0; // Reset FONT
       TKTRNX.kitalc = 0;
       if (!TCSfont)TTF_CloseFont(TCSfont);
       TCSfont = TTF_OpenFont(szTCSGraphicFont,
                              HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
       if (!TCSfont) {
        SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_INITT Error Opening Fontfile");
       } else {
        TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
        if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
         SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_INITT Fontsize?");
        } else {
         TKTRNX.khorsz= LoResX(wx);
         TKTRNX.kversz= LoResY(wz);
         TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
        }
       }
       TKTRNX.kBeamX= TKTRNX.klmrgn; // HOME
       TKTRNX.kBeamY= TKTRNX.khomey;

      } // weiter mit Erase
      case XACTION_ERASE: {
       SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
                                         , sdlColorTable[TKTRNX.iBckCol].g
                                         , sdlColorTable[TKTRNX.iBckCol].b
                                         , sdlColorTable[TKTRNX.iBckCol].a);
       SDL_RenderClear (TCSrenderer);
       break; // Erase ohne Auswirkungen auf die Cursorposition!
      }
      case XACTION_MOVABS: {
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_DRWABS: {
       SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
                                         , sdlColorTable[TKTRNX.iLinCol].g
                                         , sdlColorTable[TKTRNX.iLinCol].b
                                         , sdlColorTable[TKTRNX.iLinCol].a );
       SDL_RenderDrawLine(TCSrenderer, HiResX(TKTRNX.kBeamX),
                                       HiResY(TEK_YMAX-TKTRNX.kBeamY),
                                       HiResX(xJournalEntry->i1),
                                       HiResY(TEK_YMAX-xJournalEntry->i2) );
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_DSHSTYLE: {
       DashStyle= xJournalEntry->i1;
       break;
      }
      case XACTION_DSHABS: {
       SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
                                         , sdlColorTable[TKTRNX.iLinCol].g
                                         , sdlColorTable[TKTRNX.iLinCol].b
                                         , sdlColorTable[TKTRNX.iLinCol].a );
       DrawHiResDashLine (TKTRNX.kBeamX,TKTRNX.kBeamY, xJournalEntry->i1,xJournalEntry->i2,&DashStyle);
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
      }
      case XACTION_PNTABS: {
       SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
                                         , sdlColorTable[TKTRNX.iLinCol].g
                                         , sdlColorTable[TKTRNX.iLinCol].b
                                         , sdlColorTable[TKTRNX.iLinCol].a );
       SDL_RenderDrawPoint(TCSrenderer, HiResX(xJournalEntry->i1),
                                        HiResY(TEK_YMAX-xJournalEntry->i2) );
       TKTRNX.kBeamX= xJournalEntry->i1;
       TKTRNX.kBeamY= xJournalEntry->i2;
       break;
       }
       case XACTION_BCKCOL: {
        TKTRNX.iBckCol= xJournalEntry->i1;
        break;
       }
       case XACTION_LINCOL: {
        TKTRNX.iLinCol= xJournalEntry->i1;
        break;
       }
      case XACTION_TXTCOL: {
        TKTRNX.iTxtCol= xJournalEntry->i1;
        break;
       }
        case XACTION_FONTATTR: {
        TKTRNX.kitalc= xJournalEntry->i1;
        if (TKTRNX.kitalc > 0) {
         TTF_SetFontStyle(TCSfont, TTF_STYLE_ITALIC);
        } else {
         TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
        }

        if (TKTRNX.ksizef != xJournalEntry->i2) {
         TKTRNX.ksizef= xJournalEntry->i2;
         if (!TCSfont) TTF_CloseFont(TCSfont);
         TCSfont = TTF_OpenFont(szTCSGraphicFont,
                         HiResY((1+TKTRNX.ksizef)*TCS_REL_CHR_HEIGHT*TEK_YMAX));
         if (!TCSfont) {
          SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_FONTATTR");
         } else {
          if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
           SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_FONTATTR Size");
          } else {
           TKTRNX.khorsz= LoResX(wx);
           TKTRNX.kversz= LoResY(wz);
           TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
          }
         }
        }
        break;
       }
       case XACTION_GTEXT: {
        iStringActual= 0;
        iStringLen= xJournalEntry->i1;
        if (iStringLen > TCS_MESSAGELEN) iStringLen= TCS_MESSAGELEN;
        if (iStringLen == 0) break;
        szString[iStringActual++]= xJournalEntry->i2;
        if (iStringLen == 1) {
         szString[iStringActual]= '\0';
         PlotText (szString);
        }
        break;
       }
       case XACTION_ASCII: {
        if (iStringActual < iStringLen) {
         szString[iStringActual++]= xJournalEntry->i1;
         if (iStringActual < iStringLen) szString[iStringActual++]= xJournalEntry->i2;
         if (iStringActual >= iStringLen ) {
          szString[iStringActual]= '\0';
          PlotText (szString);
         }
        }
        break;
       }
       case XACTION_NOOP: {
        break;
       }
       default: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_XXX");
        break;
       }
     }
     xJournalEntry= xJournalEntry -> previous;
    }
 #ifdef TRACE_CALLS
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p / Last Entry: Ptr=
 %p", xTCSJournal, xJournalEntry);
 #endif
}



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

    snprintf( cBuf, TCS_MESSAGELEN, szTCSErrorMsg[iErr], msg );
    if (!TCSinitialized) { // Vor Systeminitalisierung nur Basismeldungen
     SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
     SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR,
                       szTCSstatWindowName, cBuf, TCSwindow);
    } else { // ab jetzt mit bell, outtext...
     SDL_RenderPresent (TCSrenderer);
     RepaintBuffer ();
     if (TCSErrorLev[iErr] > 0) {
      bell ();
      outtext (cBuf, strlen (cBuf) );
      if (TCSErrorLev[iErr] == 2) {
       SDL_LogInfo (SDL_LOG_CATEGORY_VIDEO, cBuf);
      }
      if (TCSErrorLev[iErr] == 3) {
       SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
      } else if (TCSErrorLev[iErr] < 10) {
       SDL_LogWarn (SDL_LOG_CATEGORY_VIDEO, cBuf);
       if (TCSErrorLev[iErr] == 5) {
        dcursr (&i,&i,&i); // Press Any Key
       } else if (TCSErrorLev[iErr]==8) {
        SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_INFORMATION,
                         szTCSstatWindowName, cBuf, TCSwindow);
       }
      } else {
       if (TCSErrorLev[iErr] == 10) {
        dcursr (&i,&i,&i); // Press Any Key
       }
       if (TCSErrorLev[iErr] == 12) {
        SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR,
                         szTCSstatWindowName, cBuf, TCSwindow);
       }
       if (iErr != ERR_EXIT) { // Error-Level von finitt durch XML veraenderbar
        SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
        finitt ();                  // Erzwungenes Beenden durch finitt
       }
      }
     }
    }
}





/* Eventhandler zum Fensterhandling */

int TCSEventFilter(void* UserData, SDL_Event* event)
{
SDL_Point winsiz;

    if (event->type == SDL_WINDOWEVENT) {
     switch (event->window.event) {
      case SDL_WINDOWEVENT_RESIZED:
      case SDL_WINDOWEVENT_MAXIMIZED:
      case SDL_WINDOWEVENT_RESTORED:
       if (event->window.windowID == SDL_GetWindowID(TCSwindow)) {
        if (SDL_GetRendererOutputSize(TCSrenderer, &winsiz.x, &winsiz.y) != 0) {
         TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
        } else {
         PixFacX= (float)(winsiz.x) / (float) TEK_XMAX;
         PixFacY= (float)(winsiz.y) / (float) TEK_YMAX;
         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "WINSIZ> PixFac: x= %f, y= %f", PixFacX, PixFacY);
        }
       }
      case SDL_WINDOWEVENT_EXPOSED:
       if (event->window.windowID == SDL_GetWindowID(TCSwindow)) {
        SDL_RenderPresent (TCSrenderer);
        RepaintBuffer ();
       } else { if (event->window.windowID == SDL_GetWindowID(TCSstatwindow)) {
        SDL_RenderPresent (TCSstatrenderer);
       } }
       break;
      default:
       break;
     }
    }
    return 1;
}



#ifdef AUDIOSUPPORT
 void audio_callback(void *sample_nr, Uint8 *raw_buffer, int bytes)
 {
 int i, length;
 float time, value;
 Sint16* buffer;
 SDL_AudioCVT cvt;

    buffer= (Sint16*) raw_buffer;
    length = 8*bytes /SDL_AUDIO_BITSIZE(SDL_AudioDev_optained.format) / SDL_AudioDev_optained.channels; // Bytes = Variablenlänge (Bit/8) pro Kanal
    for(i=0; i < length; i++, *((int*)sample_nr)=*((int*)sample_nr)+1 ) {
     time = ((float)( *((int*)sample_nr)) / SAMPLE_RATE);
     value= BELL_AMPLITUDE * sin(2.0f * M_PI * BELL_FREQUENCY * time);
     buffer[i] = (Sint16)(value);
    }
    SDL_BuildAudioCVT(&cvt, AUDIO_S16SYS, 1, SAMPLE_RATE, SDL_AudioDev_optained.format, SDL_AudioDev_optained.freq);
    cvt.len = length*2;  // Sint16 = 2 Bytes
    cvt.buf = raw_buffer;
    SDL_ConvertAudio(&cvt); // Konvertiere in das Deviceformat
#ifdef TRACE_CALLS
    SDL_LogVerbose (SDL_LOG_CATEGORY_AUDIO, "audio_callback» Number of Samples= %d Bytes allocated= %d", length,bytes);
    SDL_LogVerbose (SDL_LOG_CATEGORY_AUDIO, "audio_callback» Bytes 16bit Audio= %d Bytes needed= %d", cvt.len,cvt.len_cvt);
#endif
 }
#endif



/* Eventhandler zum Parsen von XML-Dateien */


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_HDCNAM) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSHardcopyFile);
        }
        break;
       }

       case 2: { // Section = Layout
        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_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_UNKNGRAPHCARD) == 0)  ) {
         mxmlElementSetAttr (node,"typ","opaque");
         mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_UNKNGRAPHCARD]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNGRAPHCARDL) == 0)  ) {
         mxmlElementSetAttr (node,"typ","integer");
         mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_UNKNGRAPHCARD]);

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

        } else 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_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_UNKNAUDIO) == 0)  ) {
          mxmlElementSetAttr (node,"typ","opaque");
          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_UNKNAUDIO]);
        } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNAUDIOL) == 0)  ) {
          mxmlElementSetAttr (node,"typ","integer");
          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_UNKNAUDIO]);

        } 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);
}

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


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



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


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

    if (filname[0] != '\0') {
      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);
      }
    }
}


/*
Setzen der Defaultwerte vor dem Einlesen der Initialisierungsdaten
*/

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 ()
{
char        szTmpString[TCS_FILE_NAMELEN], szTmpString1[TCS_FILE_NAMELEN];
FTNSTRDESC  ftn_WorkString, o, n;

    ftn_WorkString.len= TCS_FILE_NAMELEN; // Ersatz %: durch Programmverzeichnis
    ftn_WorkString.addr= szTCSGraphicFont;
    n.addr= SDL_GetBasePath(); // Neuer Substring = Directory
    n.len= strlen(n.addr);
    o.addr= PROGDIRTOKEN; // Alter Substring
    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) );
    strncpy(szTCSGraphicFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);

    ftn_WorkString.addr= szTCSSysFont;
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
    strncpy(szTCSSysFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);

    SDL_free (n.addr); // SDL_BasePath nicht mehr benoetigt

    n.addr= FNTFILEXT; // "Ersatz .% durch .TTF oder kein Punkt durch .TTF
    n.len= strlen(n.addr);
    o.addr= INIFILEXTTOKEN; // Alter Substring
    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) );
    strncpy(szTCSSysFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
    if (strchr(szTCSSysFont,'.') == 0) {
        strncat (szTCSSysFont, n.addr, TCS_FILE_NAMELEN-n.len);
    }

    ftn_WorkString.addr= szTCSGraphicFont;
    SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
                CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
                CALLFTNSTRL(ftn_WorkString)
                CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
    strncpy(szTCSGraphicFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
    if (strchr(szTCSGraphicFont,'.') == 0) {
        strncat (szTCSGraphicFont, n.addr, TCS_FILE_NAMELEN-n.len);
    }
}


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

{
// 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         i;
FTNINT      iL;
char        szTmpString[TMPSTRLEN], szTmpString1[TCS_FILE_NAMELEN];
char *      iAt;
FTNSTRDESC  ftn_WorkString, o, n;

    iL= FTNSTRPARL(PloWinNam);                       // Name des Grahikfensters
    if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
    strncpy(szTmpString, FTNSTRPARA(PloWinNam),iL);
    szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
    iL= strlen (szTmpString);
    if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
    if (iL > 0) {
     strncpy( szTCSWindowName, szTmpString, iL);
     szTCSWindowName[iL]= '\0';
    }

    iL= FTNSTRPARL(StatWinNam);                      // Name des Statusfensters
    if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
    strncpy(szTmpString, FTNSTRPARA(StatWinNam), iL);
    szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
    iL= strlen (szTmpString);
    if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
    if (iL > 0) {
     strncpy( szTCSstatWindowName, szTmpString, iL);
     szTCSstatWindowName[iL]= '\0';
    }

    iL= FTNSTRPARL(IniFilNam);                // Name der Initialisierungsdatei
    if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
    strncpy(szTmpString, FTNSTRPARA(IniFilNam), iL);
    szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0

    iL= strlen(szTmpString);
    if (iL > (TCS_FILE_NAMELEN-1)) iL= TCS_FILE_NAMELEN-1;
    if (iL > 0) {
     strncpy( szTCSIniFile, szTmpString, iL);
     szTCSIniFile[iL]= '\0';

     iAt= strstr (szTCSIniFile, "@"); // Section Level0?
     if (iAt != 0) {
      strncpy (szTCSsect0, &iAt[1], iL);
      iAt[0]= '\0'; // Abschneiden von @Section0 in szTCSIniFile
     }

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

     n.addr= SDL_GetBasePath(); // Neuer Substring = Directory
     n.len= strlen(n.addr);
     o.addr= PROGDIRTOKEN; // Alter Substring
     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) );
     SDL_free (n.addr);

     n.addr= INIFILEXT; // Neuer Substring = Default Extension
     n.len= strlen (INIFILEXT);
     o.addr= INIFILEXTTOKEN; // Alter Substring
     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) );
     strncpy(szTCSIniFile, ftn_WorkString.addr, TCS_FILE_NAMELEN);
    }

```

```
#ifdef TRACE_CALLS
    SDL_LogSetAllPriority(LOGLEVEL); // Ausgabe in Fehlerkanal vor INIT moeglich
    SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM,
             "WINLBL> Setting Windowname >%s< Statusname >%s< Inifile >%s<\n\r",
                           szTCSWindowName, szTCSstatWindowName, szTCSIniFile);
#endif

// Absicherung TMPSTRLEN nicht mehr benoetigt
#undef TMPSTRLEN
}



extern void initt1 ()
{
int iD;
Uint32 flags;
SDL_Point winsiz;
SDL_Rect rect;

struct xJournalEntry_typ * xJournalEntry;


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

    SDL_LogSetAllPriority(LOGLEVEL); // Ausgabe in Fehlerkanal bereits moeglich

    PresetProgPar(); // Compilerinitialisierung nach finitt() wiederherstellen

    /*
        Falls Extension des Ini-Files .XML: XML-Parser -> hier immer XML
    */
#if defined(XMLSUPPORT)
    XMLreadProgPar (szTCSIniFile);
#endif

    CustomizeProgPar (); // Ersatz %: durch Programmverzeichnis

     /*
     Übernahme der durch den Nutzer angepassten Initialisierungsdaten
     */

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

    /*
        Initialisierung des SDL2-Systems
    */

    if (SDL_Init(SDL_INIT_VIDEO) != 0) {
     TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
    }
    if (TTF_Init() != 0) {
     TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
    }
#ifdef AUDIOSUPPORT
    if (SDL_InitSubSystem(SDL_INIT_AUDIO) != 0) {
     TCSGraphicError (ERR_UNKNAUDIO, SDL_GetError());
    }
#endif

    /*
        Ermittlung allgemeiner systemspezifischer Parameter
    */

    iD= SDL_GetNumVideoDisplays();
    if (iD <= 0) {
     TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
    } else {
     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> SDL_GetNumVideoDisplays = %i", iD);
    }

    iD= iD-1;
    if (SDL_GetDisplayUsableBounds(iD, &rect) != 0) {
     TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
    } else {
     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> UsableDisplayBounds: x= %i, y= %i, w= %i, h= %i",
 rect.x,rect.y,rect.w,rect.h);
    }

    SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "linear");
    SDL_SetEventFilter(TCSEventFilter,&TCSEventFilterData);

    /*
        Erzeugung des Graphikfensters
    */

    flags= SDL_WINDOW_RESIZABLE;
    if (szTCSWindowName[0] == '~') {
     flags= flags | SDL_WINDOW_BORDERLESS;
    }
    TCSwindow = SDL_CreateWindow(szTCSWindowName,
                             TCSwindowIniXrelpos *rect.w / 100,
                             TCSwindowIniYrelpos *rect.h / 100,
                             TCSwindowIniXrelsiz *rect.w / 100,
                             TCSwindowIniYrelsiz *rect.h / 100,
                             flags );
    TCSrenderer = SDL_CreateRenderer(TCSwindow, -1, 0);



    if (SDL_GetRendererOutputSize(TCSrenderer, &winsiz.x, &winsiz.y) != 0) {
     TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
    } else {
     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> RendererBounds: x= %i, y= %i", winsiz.x,winsiz.y);
     PixFacX= (float)(winsiz.x) / (float) TEK_XMAX;
     PixFacY= (float)(winsiz.y) / (float) TEK_YMAX;
     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> PixFac: x= %f, y= %f", PixFacX, PixFacY);
    }

    SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
                                      , sdlColorTable[TKTRNX.iBckCol].g
                                      , sdlColorTable[TKTRNX.iBckCol].b
                                      , sdlColorTable[TKTRNX.iBckCol].a );
    SDL_RenderClear (TCSrenderer);
    SDL_RenderPresent (TCSrenderer);

    TCSfont = TTF_OpenFont(szTCSGraphicFont,
                  HiResY(TCS_REL_CHR_HEIGHT*TEK_YMAX));
    if (!TCSfont) {
     TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
    }         // TKTRNX wird durch INITT gesetzt

    /*
        Erzeugung des Statusfensters
    */

    if (TCSstatWindowIniYrelsiz > 0 ) {
     flags= SDL_WINDOW_RESIZABLE;
     if (szTCSstatWindowName[0] == '~') {
      flags= flags | SDL_WINDOW_BORDERLESS;
     }
     TCSstatwindow = SDL_CreateWindow(szTCSstatWindowName,
                             TCSstatWindowIniXrelpos *rect.w / 100,
                             TCSstatWindowIniYrelpos *rect.h / 100,
                             TCSstatWindowIniXrelsiz *rect.w / 100,
                             TCSstatWindowIniYrelsiz *rect.h / 100,
                             flags);

     TCSstatrenderer = SDL_CreateRenderer(TCSstatwindow, -1, 0);

     SDL_SetRenderDrawColor(TCSstatrenderer, sdlColorTable[TCSDefaultBckCol].r
                                      , sdlColorTable[TCSDefaultBckCol].g
                                      , sdlColorTable[TCSDefaultBckCol].b
                                      , sdlColorTable[TCSDefaultBckCol].a );
     SDL_RenderClear (TCSstatrenderer);
     SDL_RenderPresent (TCSstatrenderer);

     TextLineHeight= HiResY(TCS_REL_CHR_HEIGHT*TEK_YMAX);
     TCSstatusfont = TTF_OpenFont(szTCSSysFont, TextLineHeight);
     if (!TCSstatusfont) {
      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
     }
     TKTRNX.kStCol= 1; // Nur einzeilige Ausgabe
    }

    /*
        Initialisierung des Audiosystems
    */

#ifdef AUDIOSUPPORT

    SDL_AudioDev_wanted.freq = SAMPLE_RATE;
    SDL_AudioDev_wanted.format = AUDIO_S16SYS; // 16 bit integer
    SDL_AudioDev_wanted.channels = 1; // Mono
    SDL_AudioDev_wanted.samples = 2048; // buffer-size
    SDL_AudioDev_wanted.callback = audio_callback;
    SDL_AudioDev_wanted.userdata = &AudioSample_nr; // Zaehler zur Sinusberechnung

    if(SDL_OpenAudio(&SDL_AudioDev_wanted, &SDL_AudioDev_optained) < 0) {
     TCSGraphicError (ERR_UNKNAUDIO, SDL_GetError());
    } else {
     if(SDL_AudioDev_wanted.format != SDL_AudioDev_optained.format) {
      SDL_LogInfo(SDL_LOG_CATEGORY_AUDIO, "INITT1> Failed to get the desired AudioSpec");
     }
    }
    SDL_LogDebug (SDL_LOG_CATEGORY_AUDIO, "INITT1> want.frequ= %i  want.channels= %i  want.samples= %i  want.size= %i",
                  SDL_AudioDev_wanted.freq, SDL_AudioDev_wanted.channels, SDL_AudioDev_wanted.samples, SDL_AudioDev_wanted.size);
    SDL_LogDebug (SDL_LOG_CATEGORY_AUDIO, "INITT1> optained.frequ= %i  optained.channels= %i  optained.samples= %i  optained.size= %i",
                  SDL_AudioDev_optained.freq, SDL_AudioDev_optained.channels, SDL_AudioDev_optained.samples, SDL_AudioDev_optained.size);
#endif

    /*
        Anlegen des Journals
    */

    xTCSJournal= NULL;
    SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> xTCSJournal initialisiert: Ptr= %p", xTCSJournal);

    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");
    SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> Nach 1. malloc: xJournalEntry: Ptr= %p", xJournalEntry);

    xJournalEntry->action=  XACTION_NOOP; // Erkennung Listenanfang: Wurzelelement ohne Funktion
    xJournalEntry->i1= 0;
    xJournalEntry->i2= 0;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
    SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> LIST_ADD=Create Journal: xTCSJournal: Ptr= %p / xJournalEntry: Ptr= %p", xTCSJournal, xJournalEntry);
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "INITT1> previous: Ptr= %p / next: Ptr= %p", xJournalEntry -> previous, xJournalEntry -> 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, xTCSJournal, xJournalEntry, previous, next)
    SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> Nach 2. LIST_ADD: xTCSJournal: Ptr= %p / xJournalEntry: Ptr= %p", xTCSJournal, xJournalEntry);
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "INITT1> previous: Ptr= %p / next: Ptr= %p", xJournalEntry -> previous, xJournalEntry -> next);

    /*
        Initialisierung erfolgreich abgeschlossen
    */

    TCSinitialized= true;

    return;
}



extern void finitt ()
{
struct xJournalEntry_typ    * xJournalEntry;

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

    TCSGraphicError (ERR_EXIT,"");
    SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "finitt> Quit SDL");

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

    SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, xTCSJournal,
           xJournalEntry,previous,next, { free (xJournalEntry);}); // free all
    xTCSJournal= NULL;

    TTF_CloseFont(TCSfont);
    TTF_CloseFont(TCSstatusfont);

    SDL_DestroyRenderer(TCSrenderer);
    SDL_DestroyWindow(TCSwindow);

    if (TCSstatWindowIniYrelsiz > 0 ) {
     SDL_DestroyRenderer(TCSstatrenderer);
     SDL_DestroyWindow(TCSstatwindow);
    }

#ifdef AUDIOSUPPORT
    SDL_CloseAudio();
#endif

    TTF_Quit();
    SDL_Quit();

    if (TCSErrorLev[ERR_EXIT] >= 10) exit (EXIT_SUCCESS);
    return;
}



extern void iowait (void)
{
    SDL_RenderPresent (TCSrenderer);
    RepaintBuffer ();
}



/*
-------------------- 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)
{
struct xJournalEntry_typ    * xJournalEntry;

    SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
                                      , sdlColorTable[TKTRNX.iBckCol].g
                                      , sdlColorTable[TKTRNX.iBckCol].b
                                      , sdlColorTable[TKTRNX.iBckCol].a );
    SDL_RenderClear (TCSrenderer);
    SDL_RenderPresent (TCSrenderer);

     SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, xTCSJournal,
           xJournalEntry,previous,next, {free (xJournalEntry);}); // free all

     xTCSJournal= NULL; // create new journal
     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCLR,"");
     xJournalEntry->action=  XACTION_NOOP; // Wurzelelement ohne Vorgaenger
     xJournalEntry->i1= 0;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, 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, xTCSJournal, 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, xTCSJournal, 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, xTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ)); // New
 Plot
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
     xJournalEntry->action=  XACTION_ERASE;
     xJournalEntry->i1= 0;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
}



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

```
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
    if (PointInWindow (*ix, *iy)) {
     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, xTCSJournal, xJournalEntry, previous, next)
    }
}



extern void drwabs (FTNINT *ix,FTNINT *iy)
{
FTNINT iXClip, iYClip, iXClip2, iYClip2;
struct xJournalEntry_typ    * xJournalEntry;

    if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
     ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip2,&iYClip2); // geclippter Endpunkt
     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
                                       , sdlColorTable[TKTRNX.iLinCol].g
                                       , sdlColorTable[TKTRNX.iLinCol].b
                                       , sdlColorTable[TKTRNX.iLinCol].a );
     SDL_RenderDrawLine(TCSrenderer, HiResX(iXClip),HiResY(TEK_YMAX-iYClip),
                                   HiResX(iXClip2),HiResY(TEK_YMAX-iYClip2));

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

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_DRWABS;
     xJournalEntry->i1= iXClip2;
     xJournalEntry->i2= iYClip2;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
    }
    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
    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, xTCSJournal, xJournalEntry, previous, next)
}



extern void dshabs (FTNINT *ix,FTNINT *iy, FTNINT *iMask)
{
FTNINT iXClip,iYClip, iXClip2, iYClip2;
FTNINT ixx,iyy, ixx2,iyy2;
float xx,yy, dx,dy, dLin,dBlank;
struct xJournalEntry_typ    * xJournalEntry;

    if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
     ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip2,&iYClip2); // Clip Endpunkt
     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
                                       , sdlColorTable[TKTRNX.iLinCol].g
                                       , sdlColorTable[TKTRNX.iLinCol].b
                                       , sdlColorTable[TKTRNX.iLinCol].a );
     DrawHiResDashLine (iXClip,iYClip, iXClip2,iYClip2,iMask);

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

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_DSHSTYLE;
     xJournalEntry->i1= *iMask;
     xJournalEntry->i2= 0;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_DSHABS;
     xJournalEntry->i1= iXClip2;
     xJournalEntry->i2= iYClip2;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
    }
    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
    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, xTCSJournal, xJournalEntry, previous, next)
}



extern void pntabs (FTNINT *ix,FTNINT *iy)
{
struct xJournalEntry_typ    * xJournalEntry;
FTNINT ActPntMov;

    TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
    if (PointInWindow (*ix, *iy)) {
     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
                                       , sdlColorTable[TKTRNX.iLinCol].g
                                       , sdlColorTable[TKTRNX.iLinCol].b
                                       , sdlColorTable[TKTRNX.iLinCol].a );
     SDL_RenderDrawPoint(TCSrenderer, HiResX(*ix),HiResX(TEK_YMAX-*iy));
     ActPntMov= XACTION_PNTABS;
    } else {
     ActPntMov= XACTION_MOVABS;
    }
    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
    xJournalEntry->action= ActPntMov;
    xJournalEntry->i1= *ix;
    xJournalEntry->i2= *iy;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
}



extern void bckcol (FTNINT *iCol)
{
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.iBckCol= *iCol;
    if (*iCol > MAX_COLOR_INDEX) TKTRNX.iBckCol= MAX_COLOR_INDEX;

    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, xTCSJournal, xJournalEntry, previous, next)
}



extern void lincol (FTNINT *iCol)
{
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.iLinCol= *iCol;
    if (*iCol > MAX_COLOR_INDEX) TKTRNX.iLinCol= MAX_COLOR_INDEX;

    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, xTCSJournal, xJournalEntry, previous, next)
}




extern void txtcol (FTNINT *iCol)
{
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.iTxtCol= *iCol;
    if (*iCol > MAX_COLOR_INDEX) TKTRNX.iTxtCol= MAX_COLOR_INDEX;

    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, xTCSJournal, xJournalEntry, previous, next)
}



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 i, iL;
char outbuf [TCS_MESSAGELEN+1];
struct xJournalEntry_typ    * xJournalEntry;

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

    iL= 0; // Bei Bedarf String mit char(0) abschliessen -> Kopie in outbuf
    while ( (FTNSTRPARA(ftn_string)[iL] != '\0')  &&      // c-String bis \0
                     (iL < FTNSTRPARL(ftn_string)) &&      // String= Fortran Konstante
                     (iL < TCS_MESSAGELEN-1)       ) {     // Buffer Overflow
     outbuf[iL]= FTNSTRPARA(ftn_string)[iL];
     iL++;
    }
    outbuf[iL]= '\0'; //

    PlotText (outbuf);

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_GTEXT;
     xJournalEntry->i1= (FTNINT) iL;
     xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[0];
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)

     i= 1;
     while (i < iL) {
      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
      xJournalEntry->action=  XACTION_ASCII;
      xJournalEntry->i1= (FTNINT) FTNSTRPARA(ftn_string)[i++];
      if ( i<iL ) {
       xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[i++];
      } else {
       xJournalEntry->i2= (FTNINT) 0;
      }
      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
     }

     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
     xJournalEntry->action=  XACTION_MOVABS;
     xJournalEntry->i1= TKTRNX.kBeamX;
     xJournalEntry->i2= TKTRNX.kBeamY;
     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)

}



extern void italic (void)
{
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.kitalc = 1;
    TTF_SetFontStyle(TCSfont, TTF_STYLE_ITALIC);

    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
}



extern void italir (void)
{
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.kitalc = 0;
    TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);

    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
}



extern void dblsiz (void)
{
int wx,wz;
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.ksizef = 1;

    if (!TCSfont)TTF_CloseFont(TCSfont);
    TCSfont = TTF_OpenFont(szTCSGraphicFont, 2*HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
    if (!TCSfont) {
     TCSGraphicError (ERR_NOFNT,TTF_GetError() );
    } else {
     if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
      TCSGraphicError (ERR_NOFNT,TTF_GetError() );
     } else {
      TKTRNX.khorsz= LoResX(wx);
      TKTRNX.kversz= LoResY(wz);
      TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
     }
    }

    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
}



extern void nrmsiz (void)
{
int wx, wz;
struct xJournalEntry_typ    * xJournalEntry;

    TKTRNX.ksizef = 0;

    if (!TCSfont)TTF_CloseFont(TCSfont);
    TCSfont = TTF_OpenFont(szTCSGraphicFont, HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
    if (!TCSfont) {
     TCSGraphicError (ERR_NOFNT,TTF_GetError() );
    } else {
     if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
      TCSGraphicError (ERR_NOFNT,TTF_GetError() );
     } else {
      TKTRNX.khorsz= LoResX(wx);
      TKTRNX.kversz= LoResY(wz);
      TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
     }
    }

    xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
    if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
    xJournalEntry->action= XACTION_FONTATTR;
    xJournalEntry->i1= TKTRNX.kitalc;
    xJournalEntry->i2= TKTRNX.ksizef;
    SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
}






```

```
extern void csize (FTNINT *ix,FTNINT *iy)
{
     *ix=  TKTRNX.khorsz;
     *iy=  TKTRNX.kversz;
}



extern void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
{
int iL;
char outbuf [TCS_MESSAGELEN+1];
SDL_Rect dstrect;
SDL_Surface* surface;
SDL_Texture* texture;

    if ( (FTNSTRPARA(ftn_string)[0] == '\0' ) // Leerstring char(0)
       || (TCSstatWindowIniYrelsiz <= 0 ) ) { // kein Statusfenster
     return;
    }
    SDL_RenderPresent (TCSrenderer);
    RepaintBuffer ();

    iL= 0; // Bei Bedarf String mit char(0) abschliessen -> Kopie in outbuf
    while ( (FTNSTRPARA(ftn_string)[iL] != '\0')  &&      // c-String bis \0
                    (iL < FTNSTRPARL(ftn_string)) &&      // String= Fortran Konstante
                    (iL < TCS_MESSAGELEN-1)       ) {     // Buffer Overflow
     outbuf[iL]= FTNSTRPARA(ftn_string)[iL];
     iL++;
    }
    outbuf[iL]= '\0'; //

    SDL_SetRenderDrawColor(TCSstatrenderer, sdlColorTable[TCSDefaultBckCol].r
                                          , sdlColorTable[TCSDefaultBckCol].g
                                          , sdlColorTable[TCSDefaultBckCol].b
                                          , sdlColorTable[TCSDefaultBckCol].a );
    SDL_RenderClear (TCSstatrenderer);

#ifdef HIGHQUALCHAR
    surface = TTF_RenderUTF8_Blended (TCSstatusfont, outbuf, sdlColorTable[TCSDefaultLinCol]);
#else
    surface = TTF_RenderUTF8_Solid (TCSstatusfont, outbuf, sdlColorTable[TCSDefaultLinCol]);
#endif

    texture = SDL_CreateTextureFromSurface(TCSstatrenderer, surface);

    dstrect.x= 0;
    dstrect.y= 0;
    SDL_QueryTexture(texture, NULL, NULL, &dstrect.w, &dstrect.h);
    SDL_RenderCopy(TCSstatrenderer, texture, NULL, &dstrect);

    SDL_RenderPresent(TCSstatrenderer);
    SDL_DestroyTexture(texture);
    SDL_FreeSurface(surface);
}



extern void bell (void)
{
#ifdef AUDIOSUPPORT
    AudioSample_nr= 0;
    SDL_PauseAudio(0); // start playing sound
    SDL_Delay(BELL_DURATION); // wait while sound is playing
    SDL_PauseAudio(1); // stop playing sound
#endif
    return;
}


extern void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
                                         FTNINT *iL  FTNSTRPAR_TAIL(ftn_string))
{
    TCSGraphicError (*iErr, FTNSTRPARA(ftn_string));

}



/*
------------------- Userroutinen: Graphic Input---------------------------------
*/



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

```
SDL_Event  event;

    if (!TCSinitialized) return;            /* Aufhängen vermeiden */

    SDL_RenderPresent (TCSrenderer);
    RepaintBuffer ();
    SDL_RaiseWindow(TCSwindow); // Set input focus

    *ic= 0;
    while (*ic == 0) {
     SDL_WaitEvent (&event);
     switch (event.type) {
      case SDL_KEYDOWN:
       if (event.key.keysym.sym < 256) {
        *ic= (FTNINT) event.key.keysym.sym;
       }
       break;
      case SDL_MOUSEBUTTONDOWN:
       if (ix == iy) break; // Aufruf TINPUT, nicht  DCURSR
       switch (event.button.button) { // Tastaturcode analog DOS
        case SDL_BUTTON_LEFT: *ic= 1; break;
        case SDL_BUTTON_RIGHT: *ic= 2; break;
        case SDL_BUTTON_MIDDLE: *ic= 4; break;
       }
       *ix= (FTNINT) (LoResX(event.button.x));
       *iy= (FTNINT) (TEK_YMAX-LoResY(event.button.y));
       break;
      default:
       TCSEventFilter(NULL, &event); // Weiterleitung Standardhandler, ic = Dummy
       break;
     }
    }
}



/*
------------------- Userroutinen: Hardcopy -------------------------------------
*/



extern void hdcopy (void)
{

FTNINT      iErr;
FTNSTRDESC  ftnstrg;
char        szTmpString[TCS_FILE_NAMELEN];
SDL_RWops*  hFile;
struct xJournalEntry_typ *xJournalEntry;

    snprintf( szTmpString,TCS_FILE_NAMELEN, szTCSHardcopyFile, iHardcopyCount++ );
    hFile = SDL_RWFromFile( szTmpString, "r" );
    while ((iHardcopyCount < MAX_HDCCOUNT) && (hFile != NULL) ) {
     SDL_RWclose (hFile);
     snprintf( szTmpString,TCS_FILE_NAMELEN, szTCSHardcopyFile, iHardcopyCount++ );
     hFile = SDL_RWFromFile( szTmpString, "r" );
    }
    SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> iHardcopyCount Next= %i", iHardcopyCount);
    SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Filnam= %s", szTmpString);
    if (hFile != NULL) { // iHardcopyCount zu klein
     SDL_RWclose (hFile);
     SDL_LogError (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Open HDC_File: kein freier Filename");
     return;
    }

    hFile = SDL_RWFromFile( szTmpString, "wb" );
    if (hFile == NULL) {
     SDL_LogError (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Error openening %s",szTmpString);
     return;
    }

    TCSGraphicError (MSG_HDCACT, szTmpString);

    SGLIB_DL_LIST_GET_LAST (struct xJournalEntry_typ, xTCSJournal, previous, next, xJournalEntry)
#ifdef TRACE_CALLS
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xTCSJournal: Ptr= %p", xTCSJournal);
    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> 1. Entry: Ptr= %p / previous: Ptr= %p / next: Ptr= %p", xJournalEntry, xJournalEntry -> previous, xJournalEntry -> next);
#endif
    while (xJournalEntry != NULL) {
     snprintf( szTmpString,TCS_FILE_NAMELEN, "%02i#%04i-%03i\n", xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2 );
     SDL_RWwrite(hFile, szTmpString, 1, strlen(szTmpString));
#ifdef TRACE_CALLS
     switch (xJournalEntry->action) {
       case XACTION_INITT: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_INITT");
        break;
       }
       case XACTION_ERASE: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_ERASE");
        break;
       }
       case XACTION_MOVABS: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_MOVABS: x= %i, y= %i",
 xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_DRWABS: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DRWABS: x= %i, y= %i",
 xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_DSHSTYLE: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DSHSTYLE: x= %i", xJournalEntry->i1);
        break;
       }
       case XACTION_DSHABS: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DSHABS: x= %i, y= %i",
 xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_PNTABS: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_PNTABS: x= %i, y= %i",
 xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_BCKCOL: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_BCKCOL: x= %i", xJournalEntry->i1);
        break;
       }
       case XACTION_TXTCOL: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_TXTCOL: x= %i", xJournalEntry->i1);
        break;
       }
       case XACTION_LINCOL: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_LINCOL: x= %i", xJournalEntry->i1);
        break;
       }
       case XACTION_FONTATTR: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_FONTATTR: x= %i, y= %i",
 xJournalEntry->i1, xJournalEntry->i2);
        break;
       }
       case XACTION_GTEXT: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_GTEXT: Len= %i, Char[%i]= %c",
                      xJournalEntry->i1, xJournalEntry->i2, xJournalEntry->i2);
        break;
       }
       case XACTION_ASCII: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_ASCII: Char1[%i]= %c, Char2[%i]= %c",
                      xJournalEntry->i1, xJournalEntry->i1, xJournalEntry->i2, xJournalEntry->i2);
        break;
       }
       case XACTION_NOOP: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_NOOP");
        break;
       }
       default: {
        SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_XXX");
        break;
       }
     }
     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xJournalEntry: Ptr= %i / previous: Ptr= %i /
 next: Ptr= %i", xJournalEntry, xJournalEntry -> previous, xJournalEntry -> next);
#endif // TRACE_CALLS
     xJournalEntry= xJournalEntry -> previous;
    }

   SDL_RWclose (hFile);
#ifdef TRACE_CALLS
   SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xTCSJournal New Current Entry: Ptr= %p",
 xJournalEntry);
   SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> Previous: Ptr= %p  Next: Ptr= %p",
 xJournalEntry->previous, xJournalEntry->next);
#endif // TRACE_CALLS

}



/*
------------------- 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];
    };
}
```

7.34 TCSdSDLc.h File Reference

SDL Port: Low-Level Driver.

Classes

- struct FTNCOMPLEX
- struct FTNSTRDESC

Macros

- #define TEK_XMAX 1023
- #define TEK_YMAX 780
- #define false 0
- #define true !false
- #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 iowait iowait_
- #define GraphicError graphicerror_
- #define winlbl winlbl_
- #define erase erase_
- #define swind1 swind1_
- #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_
- #define GETARG getarg_
- #define INITT2 initt2_
- #define SUBSTITUTE substitute_
- #define STAT_MAXROWS 1 /∗ vorhandene Statuszeilen ∗/
- #define TCS_REL_CHR_HEIGHT 0.023f
- #define TCS_WINDOW_NAMELEN 50
- #define TCS_FILE_NAMELEN 128
- #define TCS_MESSAGELEN 132
- #define MAX_HDCCOUNT 1000 /∗ s.u.: Format TCS_HDCFILE_NAME ∗/
- #define INIFILEXTTOKEN ".%" /∗ Token fuer den Filenamenparser ∗/
- #define PROGDIRTOKEN "%:"
- #define TCS_INIFILE_NAME "Graph2D"
- #define SAMPLE_RATE 41000
- #define BELL_AMPLITUDE 32000.0
- #define BELL_FREQUENCY 441.0f
- #define BELL_DURATION 200
- #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 "Names"
- #define TCS_INIVAR_WINNAM "G2dGraphic"
- #define TCS_WINDOW_NAME "Graphics"
- #define TCS_INIVAR_STATNAM "G2dStatus"
- #define TCS_STATWINDOW_NAME "System Messages"
- #define TCS_INIVAR_HDCNAM "G2dHardcopy"
- #define TCS_HDCFILE_NAME "HDC%03i.UNKNOWN"
- #define TCS_INISECT2 "Layout"
- #define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
- #define TCS_INIDEF_COPMEN "Copy"
- #define TCS_INIVAR_FONT "G2dGraphicFont"
- #define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
- #define TCS_INIVAR_SYSFONT "G2dSystemFont"
- #define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
- #define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
- #define TCS_INIDEF_WINPOSX 1
- #define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
- #define TCS_INIDEF_WINPOSY 3
- #define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
- #define TCS_INIDEF_WINSIZX 98
- #define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
- #define TCS_INIDEF_WINSIZY 85
- #define TCS_INIVAR_STATPOSX "G2dStatusPosX"
- #define TCS_INIDEF_STATPOSX 1
- #define TCS_INIVAR_STATPOSY "G2dStatusPosY"
- #define TCS_INIDEF_STATPOSY 91
- #define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
- #define TCS_INIDEF_STATSIZX 98
- #define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
- #define TCS_INIDEF_STATSIZY 3
- #define TCS_INIVAR_LINCOL "G2dLinCol"
- #define TCS_INIDEF_LINCOL 1
- #define TCS_INIVAR_TXTCOL "G2dTxtCol"
- #define TCS_INIDEF_TXTCOL 1
- #define TCS_INIVAR_BCKCOL "G2dBckCol"
- #define TCS_INIDEF_BCKCOL 0
- #define TCS_INISECT3 "Messages"
- #define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
- #define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System: Error %s."
- #define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
- #define TCS_INIDEF_UNKNGRAPHCARDL 10
- #define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
- #define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
- #define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
- #define TCS_INIDEF_NOFNTFILL 10
- #define TCS_INIVAR_NOFNT "G2dFntfilOpen"
- #define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."

- #define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
- #define TCS_INIDEF_NOFNTL 10
- #define TCS_INIVAR_HDCOPN "G2dHdcOpen"
- #define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
- #define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
- #define TCS_INIDEF_HDCOPNL 5
- #define TCS_INIVAR_HDCWRT "G2dHdcWrite"
- #define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
- #define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
- #define TCS_INIDEF_HDCWRTL 5
- #define TCS_INIVAR_HDCINT "G2dHdcIntern"
- #define TCS_INIDEF_HDCINT "GRAPH2D HARDCOPY: Internal Error."
- #define TCS_INIVAR_HDCINTL "G2dHdcInternL"
- #define TCS_INIDEF_HDCINTL 5
- #define TCS_INIVAR_USR "G2dUser"
- #define TCS_INIDEF_USR "%s"
- #define TCS_INIVAR_USRL "G2dUserL"
- #define TCS_INIDEF_USRL 5
- #define TCS_INIVAR_HDCACT "G2dHdcActive"
- #define TCS_INIDEF_HDCACT "Hardcopy in progress: File %s created."
- #define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
- #define TCS_INIDEF_HDCACTL 1
- #define TCS_INIVAR_USRWRN "G2dPressAny"
- #define TCS_INIDEF_USRWRN "Press any key to continue."
- #define TCS_INIVAR_USRWRNL "G2dPressAnyL"
- #define TCS_INIDEF_USRWRNL 5
- #define TCS_INIVAR_EXIT "G2dExit"
- #define TCS_INIDEF_EXIT "Press any key to exit program."
- #define TCS_INIVAR_EXITL "G2dExitL"
- #define TCS_INIDEF_EXITL 10
- #define TCS_INIVAR_COPMEM "G2dNoMemory"
- #define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager: Out of Memory."
- #define TCS_INIVAR_COPMEML "G2dNoMemoryL"
- #define TCS_INIDEF_COPMEML 1
- #define TCS_INIVAR_COPLCK "G2dClipLock"
- #define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager: ClipBoard locked."
- #define TCS_INIVAR_COPLCKL "G2dClipLockL"
- #define TCS_INIDEF_COPLCKL 1
- #define TCS_INIVAR_JOUCREATE "G2dJouCreate"
- #define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal. Error-No: %s."
- #define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
- #define TCS_INIDEF_JOUCREATEL 5
- #define TCS_INIVAR_JOUENTRY "G2dJouEntry"
- #define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
- #define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
- #define TCS_INIDEF_JOUENTRYL 5
- #define TCS_INIVAR_JOUADD "G2dJouAdd"
- #define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
- #define TCS_INIVAR_JOUADDL "G2dJouAddL"
- #define TCS_INIDEF_JOUADDL 5
- #define TCS_INIVAR_JOUCLR "G2dJouClr"
- #define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
- #define TCS_INIVAR_JOUCLRL "G2dJouClrL"
- #define TCS_INIDEF_JOUCLRL 5
- #define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"

- #define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
- #define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
- #define TCS_INIDEF_JOUUNKWNL 5
- #define TCS_INIVAR_XMLPARSER "G2dXMLerror"
- #define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File: %s"
- #define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
- #define TCS_INIDEF_XMLPARSERL 8
- #define TCS_INIVAR_XMLOPEN "G2dXMLopen"
- #define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
- #define TCS_INIVAR_XMLOPENL "G2dXMLerrorL"
- #define TCS_INIDEF_XMLOPENL 8
- #define TCS_INIVAR_UNKNAUDIO "G2dAudio"
- #define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System: Error %s."
- #define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
- #define TCS_INIDEF_UNKNAUDIOL 5
- #define TCS_INIVAR_USR2 "G2dUser2"
- #define TCS_INIDEF_USR2 "%s"
- #define TCS_INIVAR_USR2L "G2dUser2L"
- #define TCS_INIDEF_USR2L 5
- #define TCS_INIVAR_INI2 "G2d2xInitt"
- #define TCS_INIDEF_INI2 "%s"
- #define TCS_INIVAR_INI2L "G2d2xInittL"
- #define TCS_INIDEF_INI2L 5

Typedefs

- typedef int bool
- typedef long int logical
- typedef long int integer
- typedef logical LOGICAL
- typedef integer FTNINT
- typedef float FTNREAL
- typedef double FTNDOUBLE
- typedef char FTNCHAR
- typedef size_t ftnlen
- typedef size_t FTNCHARLEN
- typedef FTNCHAR FTNSTRPAR

Functions

- FTNINT GETARG (FTNINT ∗iNo, FTNCHAR ∗line, FTNCHARLEN line_len)
- void SUBSTITUTE (FTNSTRPAR ∗Src, FTNSTRPAR ∗Dst, FTNSTRPAR ∗old, FTNSTRPAR ∗new FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst) FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(new))
- void GraphicError (FTNINT ∗iErr, FTNSTRPAR ∗ftn_string, FTNINT ∗iL FTNSTRPAR_TAIL(ftn_string))
- void outtext (FTNSTRPAR ∗ftn_string FTNSTRPAR_TAIL(ftn_string))
- void dcursr (FTNINT ∗ic, FTNINT ∗ix, FTNINT ∗iy)

7.34.1 Detailed Description

SDL Port: Low-Level Driver.

Version

1.2

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Headerfile for TCSdSDL.c
Definition in file TCSdSDLc.h.

7.34.2 Macro Definition Documentation

7.34.2.1 bckcol

```
#define bckcol bckcol_
```
Definition at line 76 of file TCSdSDLc.h.

7.34.2.2 bell

```
void bell bell_
```
Definition at line 85 of file TCSdSDLc.h.

7.34.2.3 BELL_AMPLITUDE

```
#define BELL_AMPLITUDE 32000.0
```
Definition at line 136 of file TCSdSDLc.h.

7.34.2.4 BELL_DURATION

```
#define BELL_DURATION 200
```
Definition at line 138 of file TCSdSDLc.h.

7.34.2.5 BELL_FREQUENCY

```
#define BELL_FREQUENCY 441.0f
```
Definition at line 137 of file TCSdSDLc.h.

7.34.2.6 CALLFTNSTRA

```
#define CALLFTNSTRA(
              ftns ) ftns.addr
```
Definition at line 58 of file TCSdSDLc.h.

7.34.2.7 CALLFTNSTRL

```
#define CALLFTNSTRL(
              ftns ) , ftns.len
```
Definition at line 59 of file TCSdSDLc.h.

7.34.2.8 csize

```
#define csize csize_
```

Definition at line 89 of file TCSdSDLc.h.

7.34.2.9 dblsiz

```
#define dblsiz(
              void ) dblsiz_
```

Definition at line 83 of file TCSdSDLc.h.

7.34.2.10 dcursr

```
#define dcursr dcursr_
```

Definition at line 88 of file TCSdSDLc.h.

7.34.2.11 DefaultColour

```
#define DefaultColour(
              void ) defaultcolour_
```

Definition at line 79 of file TCSdSDLc.h.

7.34.2.12 drwabs

```
#define drwabs drwabs_
```

Definition at line 73 of file TCSdSDLc.h.

7.34.2.13 dshabs

```
#define dshabs dshabs_
```

Definition at line 74 of file TCSdSDLc.h.

7.34.2.14 erase

```
#define erase(
              void ) erase_
```

Definition at line 70 of file TCSdSDLc.h.

7.34.2.15 ERR_EXIT

```
#define ERR_EXIT 12
```

Definition at line 173 of file TCSdSDLc.h.

7.34.2.16 ERR_NOFNT

```
#define ERR_NOFNT 4
```

Definition at line 165 of file TCSdSDLc.h.

7.34.2.17 ERR_NOFNTFIL

```
#define ERR_NOFNTFIL 3
```

Definition at line 164 of file TCSdSDLc.h.

7.34.2.18 ERR_UNKNAUDIO

```
#define ERR_UNKNAUDIO 22
```

Definition at line 183 of file TCSdSDLc.h.

7.34.2.19 ERR_UNKNGRAPHCARD

```
#define ERR_UNKNGRAPHCARD 2
```

Definition at line 163 of file TCSdSDLc.h.

7.34.2.20 ERR_XMLOPEN

```
#define ERR_XMLOPEN 21
```

Definition at line 182 of file TCSdSDLc.h.

7.34.2.21 ERR_XMLPARSER

```
#define ERR_XMLPARSER 20
```

Definition at line 181 of file TCSdSDLc.h.

7.34.2.22 false

```
#define false 0
```

Definition at line 33 of file TCSdSDLc.h.

7.34.2.23 finitt

```
void finitt finitt_
```

Definition at line 66 of file TCSdSDLc.h.

7.34.2.24 FTNSTRPAR_TAIL

```
#define FTNSTRPAR_TAIL(
              ftns ) , FTNCHARLEN ftns##_len
```

Definition at line 55 of file TCSdSDLc.h.

7.34.2.25 FTNSTRPARA

```
#define FTNSTRPARA(
              ftns ) ftns
```

Definition at line 56 of file TCSdSDLc.h.

7.34.2.26 FTNSTRPARL

```
#define FTNSTRPARL(
              ftns ) ftns##_len
```
Definition at line 57 of file TCSdSDLc.h.

7.34.2.27 FWRDFTNSTRA

```
#define FWRDFTNSTRA(
              ftns ) ftns
```
Definition at line 60 of file TCSdSDLc.h.

7.34.2.28 FWRDFTNSTRL

```
#define FWRDFTNSTRL(
              ftns ) , ftns##_len
```
Definition at line 61 of file TCSdSDLc.h.

7.34.2.29 GETARG

```
#define GETARG getarg_
```
Definition at line 95 of file TCSdSDLc.h.

7.34.2.30 GraphicError

```
#define GraphicError graphicerror_
```
Definition at line 68 of file TCSdSDLc.h.

7.34.2.31 hdcopy

```
#define hdcopy(
              void ) hdcopy_
```
Definition at line 90 of file TCSdSDLc.h.

7.34.2.32 INIFILEXTTOKEN

```
#define INIFILEXTTOKEN ".%" /* Token fuer den Filenamenparser */
```
Definition at line 130 of file TCSdSDLc.h.

7.34.2.33 initt1

```
#define initt1 initt1_
```
Definition at line 65 of file TCSdSDLc.h.

7.34.2.34 INITT2

```
void INITT2 initt2_
```
Definition at line 98 of file TCSdSDLc.h.

7.34.2.35 iowait

```
#define iowait(
                void ) iowait_
```

Definition at line 67 of file TCSdSDLc.h.

7.34.2.36 italic

```
#define italic(
                void ) italic_
```

Definition at line 81 of file TCSdSDLc.h.

7.34.2.37 italir

```
#define italir(
                void ) italir_
```

Definition at line 82 of file TCSdSDLc.h.

7.34.2.38 lib_movc3

```
#define lib_movc3 lib_movc3_
```

Definition at line 91 of file TCSdSDLc.h.

7.34.2.39 lincol

```
#define lincol lincol_
```

Definition at line 77 of file TCSdSDLc.h.

7.34.2.40 MAX_HDCCOUNT

```
#define MAX_HDCCOUNT 1000 /* s.u.:  Format TCS_HDCFILE_NAME */
```

Definition at line 128 of file TCSdSDLc.h.

7.34.2.41 movabs

```
#define movabs movabs_
```

Definition at line 72 of file TCSdSDLc.h.

7.34.2.42 MSG_HDCACT

```
#define MSG_HDCACT 10
```

Definition at line 171 of file TCSdSDLc.h.

7.34.2.43 MSG_MAXERRNO

```
#define MSG_MAXERRNO 25
```

Definition at line 186 of file TCSdSDLc.h.

7.34.2.44 MSG_NOMOUSE

```
#define MSG_NOMOUSE 5
```
Definition at line 166 of file TCSdSDLc.h.

7.34.2.45 MSG_USR

```
#define MSG_USR 9
```
Definition at line 170 of file TCSdSDLc.h.

7.34.2.46 MSG_USR2

```
#define MSG_USR2 23
```
Definition at line 184 of file TCSdSDLc.h.

7.34.2.47 nrmsiz

```
#define nrmsiz(
              void ) nrmsiz_
```
Definition at line 84 of file TCSdSDLc.h.

7.34.2.48 outgtext

```
#define outgtext outgtext_
```
Definition at line 80 of file TCSdSDLc.h.

7.34.2.49 outtext

```
#define outtext outtext_
```
Definition at line 86 of file TCSdSDLc.h.

7.34.2.50 pntabs

```
#define pntabs pntabs_
```
Definition at line 75 of file TCSdSDLc.h.

7.34.2.51 PROGDIRTOKEN

```
#define PROGDIRTOKEN "%:"
```
Definition at line 131 of file TCSdSDLc.h.

7.34.2.52 SAMPLE_RATE

```
#define SAMPLE_RATE 41000
```
Definition at line 135 of file TCSdSDLc.h.

7.34.2.53 STAT_MAXROWS

```
#define STAT_MAXROWS 1 /* vorhandene Statuszeilen */
```
Definition at line 120 of file TCSdSDLc.h.

7.34.2.54 SUBSTITUTE

```
#define SUBSTITUTE substitute_
```

Definition at line 101 of file TCSdSDLc.h.

7.34.2.55 swind1

```
#define swind1 swind1_
```

Definition at line 71 of file TCSdSDLc.h.

7.34.2.56 TCS_FILE_NAMELEN

```
#define TCS_FILE_NAMELEN 128
```

Definition at line 125 of file TCSdSDLc.h.

7.34.2.57 TCS_HDCFILE_NAME

```
#define TCS_HDCFILE_NAME "HDC%03i.UNKNOWN"
```

Definition at line 211 of file TCSdSDLc.h.

7.34.2.58 TCS_INIDEF_BCKCOL

```
#define TCS_INIDEF_BCKCOL 0
```

Definition at line 243 of file TCSdSDLc.h.

7.34.2.59 TCS_INIDEF_COPLCK

```
#define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager:  ClipBoard locked."
```

Definition at line 291 of file TCSdSDLc.h.

7.34.2.60 TCS_INIDEF_COPLCKL

```
#define TCS_INIDEF_COPLCKL 1
```

Definition at line 293 of file TCSdSDLc.h.

7.34.2.61 TCS_INIDEF_COPMEM

```
#define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager:  Out of Memory."
```

Definition at line 287 of file TCSdSDLc.h.

7.34.2.62 TCS_INIDEF_COPMEML

```
#define TCS_INIDEF_COPMEML 1
```

Definition at line 289 of file TCSdSDLc.h.

7.34.2.63 TCS_INIDEF_COPMEN

```
#define TCS_INIDEF_COPMEN "Copy"
```

Definition at line 216 of file TCSdSDLc.h.

7.34.2.64 TCS_INIDEF_EXIT

```
#define TCS_INIDEF_EXIT "Press any key to exit program."
```
Definition at line 283 of file TCSdSDLc.h.

7.34.2.65 TCS_INIDEF_EXITL

```
#define TCS_INIDEF_EXITL 10
```
Definition at line 285 of file TCSdSDLc.h.

7.34.2.66 TCS_INIDEF_FONT

```
#define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
```
Definition at line 218 of file TCSdSDLc.h.

7.34.2.67 TCS_INIDEF_HDCACT

```
#define TCS_INIDEF_HDCACT "Hardcopy in progress:  File %s created."
```
Definition at line 275 of file TCSdSDLc.h.

7.34.2.68 TCS_INIDEF_HDCACTL

```
#define TCS_INIDEF_HDCACTL 1
```
Definition at line 277 of file TCSdSDLc.h.

7.34.2.69 TCS_INIDEF_HDCINT

```
#define TCS_INIDEF_HDCINT "GRAPH2D HARDCOPY: Internal Error."
```
Definition at line 267 of file TCSdSDLc.h.

7.34.2.70 TCS_INIDEF_HDCINTL

```
#define TCS_INIDEF_HDCINTL 5
```
Definition at line 269 of file TCSdSDLc.h.

7.34.2.71 TCS_INIDEF_HDCOPN

```
#define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
```
Definition at line 259 of file TCSdSDLc.h.

7.34.2.72 TCS_INIDEF_HDCOPNL

```
#define TCS_INIDEF_HDCOPNL 5
```
Definition at line 261 of file TCSdSDLc.h.

7.34.2.73 TCS_INIDEF_HDCWRT

```
#define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
```
Definition at line 263 of file TCSdSDLc.h.

7.34.2.74 TCS_INIDEF_HDCWRTL

```
#define TCS_INIDEF_HDCWRTL 5
```
Definition at line 265 of file TCSdSDLc.h.

7.34.2.75 TCS_INIDEF_INI2

```
#define TCS_INIDEF_INI2 "%s"
```
Definition at line 331 of file TCSdSDLc.h.

7.34.2.76 TCS_INIDEF_INI2L

```
#define TCS_INIDEF_INI2L 5
```
Definition at line 333 of file TCSdSDLc.h.

7.34.2.77 TCS_INIDEF_JOUADD

```
#define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
```
Definition at line 303 of file TCSdSDLc.h.

7.34.2.78 TCS_INIDEF_JOUADDL

```
#define TCS_INIDEF_JOUADDL 5
```
Definition at line 305 of file TCSdSDLc.h.

7.34.2.79 TCS_INIDEF_JOUCLR

```
#define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
```
Definition at line 307 of file TCSdSDLc.h.

7.34.2.80 TCS_INIDEF_JOUCLRL

```
#define TCS_INIDEF_JOUCLRL 5
```
Definition at line 309 of file TCSdSDLc.h.

7.34.2.81 TCS_INIDEF_JOUCREATE

```
#define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal.  Error-No:  %s."
```
Definition at line 295 of file TCSdSDLc.h.

7.34.2.82 TCS_INIDEF_JOUCREATEL

```
#define TCS_INIDEF_JOUCREATEL 5
```
Definition at line 297 of file TCSdSDLc.h.

7.34.2.83 TCS_INIDEF_JOUENTRY

```
#define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
```
Definition at line 299 of file TCSdSDLc.h.

7.34.2.84 TCS_INIDEF_JOUENTRYL

```
#define TCS_INIDEF_JOUENTRYL 5
```
Definition at line 301 of file TCSdSDLc.h.

7.34.2.85 TCS_INIDEF_JOUUNKWN

```
#define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
```
Definition at line 311 of file TCSdSDLc.h.

7.34.2.86 TCS_INIDEF_JOUUNKWNL

```
#define TCS_INIDEF_JOUUNKWNL 5
```
Definition at line 313 of file TCSdSDLc.h.

7.34.2.87 TCS_INIDEF_LINCOL

```
#define TCS_INIDEF_LINCOL 1
```
Definition at line 239 of file TCSdSDLc.h.

7.34.2.88 TCS_INIDEF_NOFNT

```
#define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
```
Definition at line 255 of file TCSdSDLc.h.

7.34.2.89 TCS_INIDEF_NOFNTFIL

```
#define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
```
Definition at line 251 of file TCSdSDLc.h.

7.34.2.90 TCS_INIDEF_NOFNTFILL

```
#define TCS_INIDEF_NOFNTFILL 10
```
Definition at line 253 of file TCSdSDLc.h.

7.34.2.91 TCS_INIDEF_NOFNTL

```
#define TCS_INIDEF_NOFNTL 10
```
Definition at line 257 of file TCSdSDLc.h.

7.34.2.92 TCS_INIDEF_STATPOSX

```
#define TCS_INIDEF_STATPOSX 1
```
Definition at line 230 of file TCSdSDLc.h.

7.34.2.93 TCS_INIDEF_STATPOSY

```
#define TCS_INIDEF_STATPOSY 91
```
Definition at line 232 of file TCSdSDLc.h.

7.34.2.94 TCS_INIDEF_STATSIZX

```
#define TCS_INIDEF_STATSIZX 98
```
Definition at line 234 of file TCSdSDLc.h.

7.34.2.95 TCS_INIDEF_STATSIZY

```
#define TCS_INIDEF_STATSIZY 3
```
Definition at line 236 of file TCSdSDLc.h.

7.34.2.96 TCS_INIDEF_SYSFONT

```
#define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
```
Definition at line 220 of file TCSdSDLc.h.

7.34.2.97 TCS_INIDEF_TXTCOL

```
#define TCS_INIDEF_TXTCOL 1
```
Definition at line 241 of file TCSdSDLc.h.

7.34.2.98 TCS_INIDEF_UNKNAUDIO

```
#define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System:  Error %s."
```
Definition at line 323 of file TCSdSDLc.h.

7.34.2.99 TCS_INIDEF_UNKNAUDIOL

```
#define TCS_INIDEF_UNKNAUDIOL 5
```
Definition at line 325 of file TCSdSDLc.h.

7.34.2.100 TCS_INIDEF_UNKNGRAPHCARD

```
#define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System:  Error %s."
```
Definition at line 247 of file TCSdSDLc.h.

7.34.2.101 TCS_INIDEF_UNKNGRAPHCARDL

```
#define TCS_INIDEF_UNKNGRAPHCARDL 10
```
Definition at line 249 of file TCSdSDLc.h.

7.34.2.102 TCS_INIDEF_USR

```
#define TCS_INIDEF_USR "%s"
```
Definition at line 271 of file TCSdSDLc.h.

7.34.2.103 TCS_INIDEF_USR2

```
#define TCS_INIDEF_USR2 "%s"
```
Definition at line 327 of file TCSdSDLc.h.

7.34.2.104 TCS_INIDEF_USR2L

```
#define TCS_INIDEF_USR2L 5
```

Definition at line 329 of file TCSdSDLc.h.

7.34.2.105 TCS_INIDEF_USRL

```
#define TCS_INIDEF_USRL 5
```

Definition at line 273 of file TCSdSDLc.h.

7.34.2.106 TCS_INIDEF_USRWRN

```
#define TCS_INIDEF_USRWRN "Press any key to continue."
```

Definition at line 279 of file TCSdSDLc.h.

7.34.2.107 TCS_INIDEF_USRWRNL

```
#define TCS_INIDEF_USRWRNL 5
```

Definition at line 281 of file TCSdSDLc.h.

7.34.2.108 TCS_INIDEF_WINPOSX

```
#define TCS_INIDEF_WINPOSX 1
```

Definition at line 222 of file TCSdSDLc.h.

7.34.2.109 TCS_INIDEF_WINPOSY

```
#define TCS_INIDEF_WINPOSY 3
```

Definition at line 224 of file TCSdSDLc.h.

7.34.2.110 TCS_INIDEF_WINSIZX

```
#define TCS_INIDEF_WINSIZX 98
```

Definition at line 226 of file TCSdSDLc.h.

7.34.2.111 TCS_INIDEF_WINSIZY

```
#define TCS_INIDEF_WINSIZY 85
```

Definition at line 228 of file TCSdSDLc.h.

7.34.2.112 TCS_INIDEF_XMLOPEN

```
#define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
```

Definition at line 319 of file TCSdSDLc.h.

7.34.2.113 TCS_INIDEF_XMLOPENL

```
#define TCS_INIDEF_XMLOPENL 8
```

Definition at line 321 of file TCSdSDLc.h.

7.34.2.114 TCS_INIDEF_XMLPARSER

```
#define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File:  %s"
```
Definition at line 315 of file TCSdSDLc.h.

7.34.2.115 TCS_INIDEF_XMLPARSERL

```
#define TCS_INIDEF_XMLPARSERL 8
```
Definition at line 317 of file TCSdSDLc.h.

7.34.2.116 TCS_INIFILE_NAME

```
#define TCS_INIFILE_NAME "Graph2D"
```
Definition at line 133 of file TCSdSDLc.h.

7.34.2.117 TCS_INISECT0

```
#define TCS_INISECT0 "Graph2D"
```
Definition at line 196 of file TCSdSDLc.h.

7.34.2.118 TCS_INISECT1

```
#define TCS_INISECT1 "Names"
```
Definition at line 198 of file TCSdSDLc.h.

7.34.2.119 TCS_INISECT2

```
#define TCS_INISECT2 "Layout"
```
Definition at line 214 of file TCSdSDLc.h.

7.34.2.120 TCS_INISECT3

```
#define TCS_INISECT3 "Messages"
```
Definition at line 245 of file TCSdSDLc.h.

7.34.2.121 TCS_INIVAR_BCKCOL

```
#define TCS_INIVAR_BCKCOL "G2dBckCol"
```
Definition at line 242 of file TCSdSDLc.h.

7.34.2.122 TCS_INIVAR_COPLCK

```
#define TCS_INIVAR_COPLCK "G2dClipLock"
```
Definition at line 290 of file TCSdSDLc.h.

7.34.2.123 TCS_INIVAR_COPLCKL

```
#define TCS_INIVAR_COPLCKL "G2dClipLockL"
```
Definition at line 292 of file TCSdSDLc.h.

7.34.2.124 TCS_INIVAR_COPMEM

```
#define TCS_INIVAR_COPMEM "G2dNoMemory"
```

Definition at line 286 of file TCSdSDLc.h.

7.34.2.125 TCS_INIVAR_COPMEML

```
#define TCS_INIVAR_COPMEML "G2dNoMemoryL"
```

Definition at line 288 of file TCSdSDLc.h.

7.34.2.126 TCS_INIVAR_COPMEN

```
#define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
```

Definition at line 215 of file TCSdSDLc.h.

7.34.2.127 TCS_INIVAR_EXIT

```
#define TCS_INIVAR_EXIT "G2dExit"
```

Definition at line 282 of file TCSdSDLc.h.

7.34.2.128 TCS_INIVAR_EXITL

```
#define TCS_INIVAR_EXITL "G2dExitL"
```

Definition at line 284 of file TCSdSDLc.h.

7.34.2.129 TCS_INIVAR_FONT

```
#define TCS_INIVAR_FONT "G2dGraphicFont"
```

Definition at line 217 of file TCSdSDLc.h.

7.34.2.130 TCS_INIVAR_HDCACT

```
#define TCS_INIVAR_HDCACT "G2dHdcActive"
```

Definition at line 274 of file TCSdSDLc.h.

7.34.2.131 TCS_INIVAR_HDCACTL

```
#define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
```

Definition at line 276 of file TCSdSDLc.h.

7.34.2.132 TCS_INIVAR_HDCINT

```
#define TCS_INIVAR_HDCINT "G2dHdcIntern"
```

Definition at line 266 of file TCSdSDLc.h.

7.34.2.133 TCS_INIVAR_HDCINTL

```
#define TCS_INIVAR_HDCINTL "G2dHdcInternL"
```

Definition at line 268 of file TCSdSDLc.h.

7.34.2.134 TCS_INIVAR_HDCNAM

```
#define TCS_INIVAR_HDCNAM "G2dHardcopy"
```

Definition at line 203 of file TCSdSDLc.h.

7.34.2.135 TCS_INIVAR_HDCOPN

```
#define TCS_INIVAR_HDCOPN "G2dHdcOpen"
```

Definition at line 258 of file TCSdSDLc.h.

7.34.2.136 TCS_INIVAR_HDCOPNL

```
#define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
```

Definition at line 260 of file TCSdSDLc.h.

7.34.2.137 TCS_INIVAR_HDCWRT

```
#define TCS_INIVAR_HDCWRT "G2dHdcWrite"
```

Definition at line 262 of file TCSdSDLc.h.

7.34.2.138 TCS_INIVAR_HDCWRTL

```
#define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
```

Definition at line 264 of file TCSdSDLc.h.

7.34.2.139 TCS_INIVAR_INI2

```
#define TCS_INIVAR_INI2 "G2d2xInitt"
```

Definition at line 330 of file TCSdSDLc.h.

7.34.2.140 TCS_INIVAR_INI2L

```
#define TCS_INIVAR_INI2L "G2d2xInittL"
```

Definition at line 332 of file TCSdSDLc.h.

7.34.2.141 TCS_INIVAR_JOUADD

```
#define TCS_INIVAR_JOUADD "G2dJouAdd"
```

Definition at line 302 of file TCSdSDLc.h.

7.34.2.142 TCS_INIVAR_JOUADDL

```
#define TCS_INIVAR_JOUADDL "G2dJouAddL"
```

Definition at line 304 of file TCSdSDLc.h.

7.34.2.143 TCS_INIVAR_JOUCLR

```
#define TCS_INIVAR_JOUCLR "G2dJouClr"
```

Definition at line 306 of file TCSdSDLc.h.

7.34.2.144 TCS_INIVAR_JOUCLRL

```
#define TCS_INIVAR_JOUCLRL "G2dJouClrL"
```
Definition at line 308 of file TCSdSDLc.h.

7.34.2.145 TCS_INIVAR_JOUCREATE

```
#define TCS_INIVAR_JOUCREATE "G2dJouCreate"
```
Definition at line 294 of file TCSdSDLc.h.

7.34.2.146 TCS_INIVAR_JOUCREATEL

```
#define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
```
Definition at line 296 of file TCSdSDLc.h.

7.34.2.147 TCS_INIVAR_JOUENTRY

```
#define TCS_INIVAR_JOUENTRY "G2dJouEntry"
```
Definition at line 298 of file TCSdSDLc.h.

7.34.2.148 TCS_INIVAR_JOUENTRYL

```
#define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
```
Definition at line 300 of file TCSdSDLc.h.

7.34.2.149 TCS_INIVAR_JOUUNKWN

```
#define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
```
Definition at line 310 of file TCSdSDLc.h.

7.34.2.150 TCS_INIVAR_JOUUNKWNL

```
#define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
```
Definition at line 312 of file TCSdSDLc.h.

7.34.2.151 TCS_INIVAR_LINCOL

```
#define TCS_INIVAR_LINCOL "G2dLinCol"
```
Definition at line 238 of file TCSdSDLc.h.

7.34.2.152 TCS_INIVAR_NOFNT

```
#define TCS_INIVAR_NOFNT "G2dFntfilOpen"
```
Definition at line 254 of file TCSdSDLc.h.

7.34.2.153 TCS_INIVAR_NOFNTFIL

```
#define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
```
Definition at line 250 of file TCSdSDLc.h.

7.34.2.154 TCS_INIVAR_NOFNTFILL

```
#define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
```
Definition at line 252 of file TCSdSDLc.h.

7.34.2.155 TCS_INIVAR_NOFNTL

```
#define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
```
Definition at line 256 of file TCSdSDLc.h.

7.34.2.156 TCS_INIVAR_STATNAM

```
#define TCS_INIVAR_STATNAM "G2dStatus"
```
Definition at line 201 of file TCSdSDLc.h.

7.34.2.157 TCS_INIVAR_STATPOSX

```
#define TCS_INIVAR_STATPOSX "G2dStatusPosX"
```
Definition at line 229 of file TCSdSDLc.h.

7.34.2.158 TCS_INIVAR_STATPOSY

```
#define TCS_INIVAR_STATPOSY "G2dStatusPosY"
```
Definition at line 231 of file TCSdSDLc.h.

7.34.2.159 TCS_INIVAR_STATSIZX

```
#define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
```
Definition at line 233 of file TCSdSDLc.h.

7.34.2.160 TCS_INIVAR_STATSIZY

```
#define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
```
Definition at line 235 of file TCSdSDLc.h.

7.34.2.161 TCS_INIVAR_SYSFONT

```
#define TCS_INIVAR_SYSFONT "G2dSystemFont"
```
Definition at line 219 of file TCSdSDLc.h.

7.34.2.162 TCS_INIVAR_TXTCOL

```
#define TCS_INIVAR_TXTCOL "G2dTxtCol"
```
Definition at line 240 of file TCSdSDLc.h.

7.34.2.163 TCS_INIVAR_UNKNAUDIO

```
#define TCS_INIVAR_UNKNAUDIO "G2dAudio"
```
Definition at line 322 of file TCSdSDLc.h.

7.34.2.164 TCS_INIVAR_UNKNAUDIOL

```
#define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
```
Definition at line 324 of file TCSdSDLc.h.

7.34.2.165 TCS_INIVAR_UNKNGRAPHCARD

```
#define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
```
Definition at line 246 of file TCSdSDLc.h.

7.34.2.166 TCS_INIVAR_UNKNGRAPHCARDL

```
#define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
```
Definition at line 248 of file TCSdSDLc.h.

7.34.2.167 TCS_INIVAR_USR

```
#define TCS_INIVAR_USR "G2dUser"
```
Definition at line 270 of file TCSdSDLc.h.

7.34.2.168 TCS_INIVAR_USR2

```
#define TCS_INIVAR_USR2 "G2dUser2"
```
Definition at line 326 of file TCSdSDLc.h.

7.34.2.169 TCS_INIVAR_USR2L

```
#define TCS_INIVAR_USR2L "G2dUser2L"
```
Definition at line 328 of file TCSdSDLc.h.

7.34.2.170 TCS_INIVAR_USRL

```
#define TCS_INIVAR_USRL "G2dUserL"
```
Definition at line 272 of file TCSdSDLc.h.

7.34.2.171 TCS_INIVAR_USRWRN

```
#define TCS_INIVAR_USRWRN "G2dPressAny"
```
Definition at line 278 of file TCSdSDLc.h.

7.34.2.172 TCS_INIVAR_USRWRNL

```
#define TCS_INIVAR_USRWRNL "G2dPressAnyL"
```
Definition at line 280 of file TCSdSDLc.h.

7.34.2.173 TCS_INIVAR_WINNAM

```
#define TCS_INIVAR_WINNAM "G2dGraphic"
```
Definition at line 199 of file TCSdSDLc.h.

7.34.2.174 TCS_INIVAR_WINPOSX

```
#define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
```

Definition at line 221 of file TCSdSDLc.h.

7.34.2.175 TCS_INIVAR_WINPOSY

```
#define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
```

Definition at line 223 of file TCSdSDLc.h.

7.34.2.176 TCS_INIVAR_WINSIZX

```
#define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
```

Definition at line 225 of file TCSdSDLc.h.

7.34.2.177 TCS_INIVAR_WINSIZY

```
#define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
```

Definition at line 227 of file TCSdSDLc.h.

7.34.2.178 TCS_INIVAR_XMLOPEN

```
#define TCS_INIVAR_XMLOPEN "G2dXMLopen"
```

Definition at line 318 of file TCSdSDLc.h.

7.34.2.179 TCS_INIVAR_XMLOPENL

```
#define TCS_INIVAR_XMLOPENL "G2dXMLerrorL"
```

Definition at line 320 of file TCSdSDLc.h.

7.34.2.180 TCS_INIVAR_XMLPARSER

```
#define TCS_INIVAR_XMLPARSER "G2dXMLerror"
```

Definition at line 314 of file TCSdSDLc.h.

7.34.2.181 TCS_INIVAR_XMLPARSERL

```
#define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
```

Definition at line 316 of file TCSdSDLc.h.

7.34.2.182 TCS_MESSAGELEN

```
#define TCS_MESSAGELEN 132
```

Definition at line 126 of file TCSdSDLc.h.

7.34.2.183 TCS_REL_CHR_HEIGHT

```
#define TCS_REL_CHR_HEIGHT 0.023f
```

Definition at line 122 of file TCSdSDLc.h.

7.34.2.184 TCS_STATWINDOW_NAME

```
#define TCS_STATWINDOW_NAME "System Messages"
```
Definition at line 202 of file TCSdSDLc.h.

7.34.2.185 TCS_WINDOW_NAME

```
#define TCS_WINDOW_NAME "Graphics"
```
Definition at line 200 of file TCSdSDLc.h.

7.34.2.186 TCS_WINDOW_NAMELEN

```
#define TCS_WINDOW_NAMELEN 50
```
Definition at line 124 of file TCSdSDLc.h.

7.34.2.187 tcslev3

```
#define tcslev3 tcslev3_
```
Definition at line 64 of file TCSdSDLc.h.

7.34.2.188 TEK_XMAX

```
#define TEK_XMAX 1023
```
Definition at line 19 of file TCSdSDLc.h.

7.34.2.189 TEK_YMAX

```
#define TEK_YMAX 780
```
Definition at line 20 of file TCSdSDLc.h.

7.34.2.190 tinput

```
#define tinput tinput_
```
Definition at line 87 of file TCSdSDLc.h.

7.34.2.191 TKTRNX

```
#define TKTRNX tktrnx_ /* Fortran Naming Convention */
```
Definition at line 63 of file TCSdSDLc.h.

7.34.2.192 true

```
#define true !false
```
Definition at line 34 of file TCSdSDLc.h.

7.34.2.193 txtcol

```
#define txtcol txtcol_
```
Definition at line 78 of file TCSdSDLc.h.

7.34.2.194 winlbl

```
#define winlbl winlbl_
```
Definition at line 69 of file TCSdSDLc.h.

7.34.2.195 WRN_COPYLOCK

```
#define WRN_COPYLOCK 14
```
Definition at line 175 of file TCSdSDLc.h.

7.34.2.196 WRN_COPYNOMEM

```
#define WRN_COPYNOMEM 13
```
Definition at line 174 of file TCSdSDLc.h.

7.34.2.197 WRN_HDCFILOPN

```
#define WRN_HDCFILOPN 6
```
Definition at line 167 of file TCSdSDLc.h.

7.34.2.198 WRN_HDCFILWRT

```
#define WRN_HDCFILWRT 7
```
Definition at line 168 of file TCSdSDLc.h.

7.34.2.199 WRN_HDCINTERN

```
#define WRN_HDCINTERN 8
```
Definition at line 169 of file TCSdSDLc.h.

7.34.2.200 WRN_INI2

```
#define WRN_INI2 24
```
Definition at line 185 of file TCSdSDLc.h.

7.34.2.201 WRN_JOUADD

```
#define WRN_JOUADD 17
```
Definition at line 178 of file TCSdSDLc.h.

7.34.2.202 WRN_JOUCLR

```
#define WRN_JOUCLR 18
```
Definition at line 179 of file TCSdSDLc.h.

7.34.2.203 WRN_JOUCREATE

```
#define WRN_JOUCREATE 15
```
Definition at line 176 of file TCSdSDLc.h.

7.34.2.204 WRN_JOUENTRY

```
#define WRN_JOUENTRY 16
```
Definition at line 177 of file TCSdSDLc.h.

7.34.2.205 WRN_JOUUNKWN

```
#define WRN_JOUUNKWN 19
```
Definition at line 180 of file TCSdSDLc.h.

7.34.2.206 WRN_NOMSG

```
#define WRN_NOMSG 1
```
Definition at line 162 of file TCSdSDLc.h.

7.34.2.207 WRN_USRPRESSANY

```
#define WRN_USRPRESSANY 11
```
Definition at line 172 of file TCSdSDLc.h.

7.34.2.208 XACTION_ASCII

```
#define XACTION_ASCII 9
```
Definition at line 151 of file TCSdSDLc.h.

7.34.2.209 XACTION_BCKCOL

```
#define XACTION_BCKCOL 10
```
Definition at line 152 of file TCSdSDLc.h.

7.34.2.210 XACTION_DRWABS

```
#define XACTION_DRWABS 4
```
Definition at line 146 of file TCSdSDLc.h.

7.34.2.211 XACTION_DSHABS

```
#define XACTION_DSHABS 6
```
Definition at line 148 of file TCSdSDLc.h.

7.34.2.212 XACTION_DSHSTYLE

```
#define XACTION_DSHSTYLE 5
```
Definition at line 147 of file TCSdSDLc.h.

7.34.2.213 XACTION_ERASE

```
#define XACTION_ERASE 2
```
Definition at line 144 of file TCSdSDLc.h.

7.34.2.214 XACTION_FONTATTR

```
#define XACTION_FONTATTR 13
```
Definition at line 155 of file TCSdSDLc.h.

7.34.2.215 XACTION_GTEXT

```
#define XACTION_GTEXT 8
```
Definition at line 150 of file TCSdSDLc.h.

7.34.2.216 XACTION_INITT

```
#define XACTION_INITT 1
```
Definition at line 143 of file TCSdSDLc.h.

7.34.2.217 XACTION_LINCOL

```
#define XACTION_LINCOL 11
```
Definition at line 153 of file TCSdSDLc.h.

7.34.2.218 XACTION_MOVABS

```
#define XACTION_MOVABS 3
```
Definition at line 145 of file TCSdSDLc.h.

7.34.2.219 XACTION_NOOP

```
#define XACTION_NOOP 14
```
Definition at line 156 of file TCSdSDLc.h.

7.34.2.220 XACTION_PNTABS

```
#define XACTION_PNTABS 7
```
Definition at line 149 of file TCSdSDLc.h.

7.34.2.221 XACTION_TXTCOL

```
#define XACTION_TXTCOL 12
```
Definition at line 154 of file TCSdSDLc.h.

7.34.3 Typedef Documentation

7.34.3.1 bool

```
typedef int bool
```
Definition at line 32 of file TCSdSDLc.h.

7.34.3.2 FTNCHAR

```
typedef char FTNCHAR
```
Definition at line 48 of file TCSdSDLc.h.

7.34.3.3 FTNCHARLEN

```
typedef size_t FTNCHARLEN
```
Definition at line 51 of file TCSdSDLc.h.

7.34.3.4 FTNDOUBLE

```
typedef double FTNDOUBLE
```
Definition at line 45 of file TCSdSDLc.h.

7.34.3.5 FTNINT

```
typedef integer FTNINT
```
Definition at line 43 of file TCSdSDLc.h.

7.34.3.6 ftnlen

```
typedef size_t ftnlen
```
Definition at line 50 of file TCSdSDLc.h.

7.34.3.7 FTNREAL

```
typedef float FTNREAL
```
Definition at line 44 of file TCSdSDLc.h.

7.34.3.8 FTNSTRPAR

```
typedef FTNCHAR FTNSTRPAR
```
Definition at line 54 of file TCSdSDLc.h.

7.34.3.9 integer

```
typedef long int integer
```
Definition at line 40 of file TCSdSDLc.h.

7.34.3.10 logical

```
typedef long int logical
```
Definition at line 39 of file TCSdSDLc.h.

7.34.3.11 LOGICAL

```
typedef logical LOGICAL
```
Definition at line 42 of file TCSdSDLc.h.

7.34.4 Function Documentation

7.34.4.1 dcursr()

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

Definition at line 2015 of file TCSdSDLc.c.

7.34.4.2 GETARG()

```
FTNINT GETARG (
            FTNINT * iNo,
            FTNCHAR * line,
            FTNCHARLEN line_len )
```

7.34.4.3 GraphicError()

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

Definition at line 2000 of file TCSdSDLc.c.

7.34.4.4 outtext()

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

Definition at line 1938 of file TCSdSDLc.c.

7.34.4.5 SUBSTITUTE()

```
void SUBSTITUTE (
            FTNSTRPAR * Src,
            FTNSTRPAR * Dst,
            FTNSTRPAR * old,
            FTNSTRPAR *new  FTNSTRPAR_TAILSrc) FTNSTRPAR_TAIL(Dst) FTNSTRPAR_TAIL(old) FTNST←
RPAR_TAIL(new )
```

7.35 TCSdSDLc.h

```
/** ***************************************************************************
\file    TCSdSDLc.h
\brief   SDL Port: Low-Level Driver
\version 1.2
\author  (C) 2023 Dr.-Ing. Klaus Friedewald
\copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
\~german
         Headerfile zu TCSdSDLc.c
\~english
         Headerfile for TCSdSDL.c
\~

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



```

```c
/* ---- Zeichenbereich im Tektronix-Koordinatensystem --------------------- */

#define TEK_XMAX 1023
#define TEK_YMAX 780


/* ------------ Compilerspezifische Definitionen -------------------------- */

#ifdef _UNICODE
 #error "GNU f77 basiert nicht auf UNICODE !!!"
#endif


/* Deklaration analog C++ */

 typedef int bool;
 #define false 0
 #define true !false


/* Deklaration Parameteruebergabe Fortran <-> C */

typedef long int logical; // 3 plattformabhaengige Definitionen
typedef long int integer; // evtl. ueberpruefen

typedef logical LOGICAL;
typedef integer FTNINT;
typedef float FTNREAL;
typedef double FTNDOUBLE;
typedef struct {float real, imag;} FTNCOMPLEX;

typedef char FTNCHAR;

typedef size_t ftnlen;  // Ersatz fuer g2c.h
typedef size_t FTNCHARLEN;

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 iowait iowait_
#define GraphicError graphicerror_
#define winlbl winlbl_
#define erase erase_
#define swind1 swind1_
#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_
void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *new
                                           FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
                                           FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(new));

/* Forward Deklarationen: Codiert in C und auch in C verwendet */

void bell (void); //  -> Forward Deklaration
void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
                   FTNINT *iL  FTNSTRPAR_TAIL(ftn_string));
void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) );
void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy);
void finitt ();



/* ------------ Programmparameter ----------------------------------------- */


#define STAT_MAXROWS 1             /* vorhandene Statuszeilen */

#define TCS_REL_CHR_HEIGHT 0.023f

#define TCS_WINDOW_NAMELEN 50
#define TCS_FILE_NAMELEN 128
#define TCS_MESSAGELEN 132

#define MAX_HDCCOUNT 1000          /* s.u.: Format TCS_HDCFILE_NAME */

#define INIFILEXTTOKEN ".%"        /* Token fuer den Filenamenparser */
#define PROGDIRTOKEN "%:"

#define TCS_INIFILE_NAME "Graph2D"

#define SAMPLE_RATE 41000 // fuer SDL-Audioausgabe
#define BELL_AMPLITUDE 32000.0
#define BELL_FREQUENCY 441.0f
#define BELL_DURATION 200


/* 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, Registry und *.xml) beruecksichtigen! */

#define TCS_INISECT0 "Graph2D" // Root-Section, derzeit nur bei XML verwendet

#define TCS_INISECT1 "Names"
 #define TCS_INIVAR_WINNAM "G2dGraphic"
    #define TCS_WINDOW_NAME "Graphics"
 #define TCS_INIVAR_STATNAM "G2dStatus"
    #define TCS_STATWINDOW_NAME "System Messages"
 #define TCS_INIVAR_HDCNAM "G2dHardcopy"
    #if (JOURNALTYP ==1)
       #define TCS_HDCFILE_NAME "HDC%03i.WMF"
    #elif (JOURNALTYP ==2)
       #define TCS_HDCFILE_NAME "HDC%03i.EMF"
    #elif (JOURNALTYP ==3)
       #define TCS_HDCFILE_NAME "HDC%03i.HDC"
    #else
       #define TCS_HDCFILE_NAME "HDC%03i.UNKNOWN"
    #endif

#define TCS_INISECT2 "Layout"
 #define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
    #define TCS_INIDEF_COPMEN "Copy"
 #define TCS_INIVAR_FONT "G2dGraphicFont"
    #define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
 #define TCS_INIVAR_SYSFONT "G2dSystemFont"
    #define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
 #define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
    #define TCS_INIDEF_WINPOSX 1
 #define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
    #define TCS_INIDEF_WINPOSY 3
 #define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
    #define TCS_INIDEF_WINSIZX 98
 #define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
    #define TCS_INIDEF_WINSIZY 85
 #define TCS_INIVAR_STATPOSX "G2dStatusPosX"
    #define TCS_INIDEF_STATPOSX 1
 #define TCS_INIVAR_STATPOSY "G2dStatusPosY"
    #define TCS_INIDEF_STATPOSY 91
 #define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
    #define TCS_INIDEF_STATSIZX 98
 #define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
    #define TCS_INIDEF_STATSIZY 3   // mit X11 o.k.
//    #define TCS_INIDEF_STATSIZY 0 // sonst nur 1 Fenster
 #define TCS_INIVAR_LINCOL "G2dLinCol"
    #define TCS_INIDEF_LINCOL 1
 #define TCS_INIVAR_TXTCOL "G2dTxtCol"
    #define TCS_INIDEF_TXTCOL 1
 #define TCS_INIVAR_BCKCOL "G2dBckCol"
    #define TCS_INIDEF_BCKCOL 0

#define TCS_INISECT3 "Messages"
 #define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
    #define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System: Error %s."
    #define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
    #define TCS_INIDEF_UNKNGRAPHCARDL 10
 #define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
    #define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
    #define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
    #define TCS_INIDEF_NOFNTFILL 10
 #define TCS_INIVAR_NOFNT "G2dFntfilOpen"
    #define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
    #define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
    #define TCS_INIDEF_NOFNTL 10
 #define TCS_INIVAR_HDCOPN "G2dHdcOpen"
    #define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
    #define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
    #define TCS_INIDEF_HDCOPNL 5
 #define TCS_INIVAR_HDCWRT "G2dHdcWrite"
    #define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
    #define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
    #define TCS_INIDEF_HDCWRTL 5
 #define TCS_INIVAR_HDCINT "G2dHdcIntern"
    #define TCS_INIDEF_HDCINT "GRAPH2D HARDCOPY: Internal Error."
    #define TCS_INIVAR_HDCINTL "G2dHdcInternL"
    #define TCS_INIDEF_HDCINTL 5
 #define TCS_INIVAR_USR "G2dUser"
    #define TCS_INIDEF_USR "%s"
    #define TCS_INIVAR_USRL "G2dUserL"
    #define TCS_INIDEF_USRL 5
 #define TCS_INIVAR_HDCACT "G2dHdcActive"
    #define TCS_INIDEF_HDCACT "Hardcopy in progress: File %s created."
    #define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
    #define TCS_INIDEF_HDCACTL 1
```

```
#define TCS_INIVAR_USRWRN "G2dPressAny"
   #define TCS_INIDEF_USRWRN "Press any key to continue."
   #define TCS_INIVAR_USRWRNL "G2dPressAnyL"
   #define TCS_INIDEF_USRWRNL 5
#define TCS_INIVAR_EXIT "G2dExit"
   #define TCS_INIDEF_EXIT "Press any key to exit program."
   #define TCS_INIVAR_EXITL "G2dExitL"
   #define TCS_INIDEF_EXITL 10
#define TCS_INIVAR_COPMEM "G2dNoMemory"
   #define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager: Out of Memory."
   #define TCS_INIVAR_COPMEML "G2dNoMemoryL"
   #define TCS_INIDEF_COPMEML 1
#define TCS_INIVAR_COPLCK "G2dClipLock"
   #define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager: ClipBoard locked."
   #define TCS_INIVAR_COPLCKL "G2dClipLockL"
   #define TCS_INIDEF_COPLCKL 1
#define TCS_INIVAR_JOUCREATE "G2dJouCreate"
   #define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal. Error-No: %s."
   #define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
   #define TCS_INIDEF_JOUCREATEL 5
#define TCS_INIVAR_JOUENTRY "G2dJouEntry"
   #define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
   #define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
   #define TCS_INIDEF_JOUENTRYL 5
#define TCS_INIVAR_JOUADD "G2dJouAdd"
   #define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
   #define TCS_INIVAR_JOUADDL "G2dJouAddL"
   #define TCS_INIDEF_JOUADDL 5
#define TCS_INIVAR_JOUCLR "G2dJouClr"
   #define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
   #define TCS_INIVAR_JOUCLRL "G2dJouClrL"
   #define TCS_INIDEF_JOUCLRL 5
#define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
   #define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
   #define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
   #define TCS_INIDEF_JOUUNKWNL 5
#define TCS_INIVAR_XMLPARSER "G2dXMLerror"
   #define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File: %s"
   #define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
   #define TCS_INIDEF_XMLPARSERL 8
#define TCS_INIVAR_XMLOPEN "G2dXMLopen"
   #define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
   #define TCS_INIVAR_XMLOPENL "G2dXMLerrorL"
   #define TCS_INIDEF_XMLOPENL 8
#define TCS_INIVAR_UNKNAUDIO "G2dAudio"
   #define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System: Error %s."
   #define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
   #define TCS_INIDEF_UNKNAUDIOL 5
#define TCS_INIVAR_USR2 "G2dUser2"
   #define TCS_INIDEF_USR2 "%s"
   #define TCS_INIVAR_USR2L "G2dUser2L"
   #define TCS_INIDEF_USR2L 5
#define TCS_INIVAR_INI2 "G2d2xInitt"
   #define TCS_INIDEF_INI2 "%s"
   #define TCS_INIVAR_INI2L "G2d2xInittL"
   #define TCS_INIDEF_INI2L 5
```

7.36 Tktrnx.fd File Reference

SDL Port: TCS Common Block TKTRNX.

7.36.1 Detailed Description

SDL Port: TCS Common Block TKTRNX.

Version

1.2

Author

Dr.-Ing. Klaus Friedewald

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.

7.37 Tktrnx.fd

```
C> \file Tktrnx.fd
C> \brief   SDL Port: TCS Common Block TKTRNX
C> \version 1.2
C> \author  Dr.-Ing. Klaus Friedewald
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

      COMMON /tktrnx/
     & khomey,
     & khorsz,kversz,
     & kitalc,ksizef,
     & klmrgn,krmrgn,
     & kbeamx,kbeamy,
     & kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
     & trcosf,trsinf,trscal
     & ,xfac,yfac,xlog,ylog,kstcol,
     & ilincol, ibckcol, itxtcol

      SAVE /tktrnx/
      integer iTktrnxL
      parameter(itktrnxl=28) ! +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
C Achtung:
C       Anpassung Parameters iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
C       Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
C
C> \endcond

```

7.38 TKTRNX.h File Reference

SDL Port: TCS Common Block TKTRNX.

Classes

- struct TKTRNXcommonBlock

Variables

- struct TKTRNXcommonBlock TKTRNX

7.38.1 Detailed Description

SDL Port: TCS Common Block TKTRNX.

Version

1.2

Author

Dr.-Ing. Klaus Friedewald

C header belonging to TKTRNX.fd

Note

SDL-Version auf Basis der Windows-Version 1.2 Anpassung an die compilerabhaengige Namenskonvention erfolgt in TCSdSDLc.h

Definition in file TKTRNX.h.

7.38.2 Variable Documentation

7.38.2.1 TKTRNX

```
struct TKTRNXcommonBlock TKTRNX
```

7.39 TKTRNX.h

```
/** ****************************************************************************
\file    TKTRNX.h
\brief   SDL Port: TCS Common Block TKTRNX
\version 1.2
\author  Dr.-Ing. Klaus Friedewald
\~german
         C Header passend zu TKTRNX.fd
\~english
         C header belonging to TKTRNX.fd
\~

\note
   SDL-Version auf Basis der Windows-Version 1.2
   Anpassung an die compilerabhaengige Namenskonvention erfolgt in TCSdSDLc.h

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


extern struct TKTRNXcommonBlock {
FTNINT
     khomey,
     khorsz,kversz,
     kitalc,ksizef,
     klmrgn,krmrgn,
     kBeamX,kBeamY,
     kminsx,kminsy,kmaxsx,kmaxsy;

FTNREAL
     tminvx,tminvy,tmaxvx,tmaxvy,
     trcosf,trsinf,trscal
     ,xfac,yfac,xlog,ylog;
FTNINT
    kStCol,
    iLinCol, iBckCol, iTxtCol;
} TKTRNX;
```

Index