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

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

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

7.1.1 Detailed Description

Graph2D: Tektronix Advanced Graphing II Emulation.

Version

(2024,347, x)

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Layer 2: scientific 2-D graphic subroutines

Note

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

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

Definition in file AG2.for.

7.1.2 Function/Subroutine Documentation

7.1.2.1 ag2infin()

```
real function ag2infin
```

Definition at line 155 of file AG2.for.

7.1.2.2 ag2lev()

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

Definition at line 94 of file AG2.for.

7.1.2.3 alfsetc()

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

Definition at line 2573 of file AG2.for.

7.1.2.4 bar()

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

Definition at line 1698 of file AG2.for.

7.1.2.5 binitt()

```
subroutine binitt
```

Definition at line 724 of file AG2.for.

7.1.2.6 bsyms()

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

Definition at line 1850 of file AG2.for.

7.1.2.7 calcon()

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

Definition at line 1336 of file AG2.for.

7.1.2.8 calpnt()

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

Definition at line 1281 of file AG2.for.

7.1.2.9 check()

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

Definition at line 808 of file AG2.for.

7.1.2.10 cmnmx()

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

Definition at line 930 of file AG2.for.

7.1.2.11 coptim()

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

Definition at line 1125 of file AG2.for.

7.1.2.12 cplot()

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

Definition at line 1548 of file AG2.for.

7.1.2.13 datget()

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

Definition at line 1670 of file AG2.for.

7.1.2.14 dinitx()

```
subroutine dinitx
```

Definition at line 654 of file AG2.for.

7.1.2.15 dinity()

```
subroutine dinity
```

Definition at line 668 of file AG2.for.

7.1.2.16 dlimx()

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

Definition at line 474 of file AG2.for.

7.1.2.17 dlimy()

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

Definition at line 486 of file AG2.for.

7.1.2.18 dsplay()

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

Definition at line 1534 of file AG2.for.

7.1.2.19 eformc()

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

Definition at line 2444 of file AG2.for.

7.1.2.20 esplit()

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

Definition at line 2477 of file AG2.for.

7.1.2.21 expoutc()

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

Definition at line 2497 of file AG2.for.

7.1.2.22 fformc()

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

Definition at line 2385 of file AG2.for.

7.1.2.23 filbox()

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

Definition at line 1765 of file AG2.for.

7.1.2.24 findge()

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

Definition at line 2932 of file AG2.for.

7.1.2.25 findle()

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

Definition at line 2951 of file AG2.for.

7.1.2.26 fonlyc()

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

Definition at line 2413 of file AG2.for.

7.1.2.27 frame()

```
subroutine frame
```

Definition at line 1520 of file AG2.for.

7.1.2.28 gline()

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

Definition at line 2183 of file AG2.for.

7.1.2.29 grid()

```
subroutine grid
```

Definition at line 1966 of file AG2.for.

7.1.2.30 hbarst()

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

Definition at line 682 of file AG2.for.

7.1.2.31 iformc()

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

Definition at line 2353 of file AG2.for.

7.1.2.32 infin()

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

Definition at line 142 of file AG2.for.

7.1.2.33 iother()

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

Definition at line 3076 of file AG2.for.

7.1.2.34 iubgc()

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

Definition at line 1483 of file AG2.for.

7.1.2.35 justerc()

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

Definition at line 2676 of file AG2.for.

7.1.2.36 keyset()

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

Definition at line 1644 of file AG2.for.

7.1.2.37 label()

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

Definition at line 2210 of file AG2.for.

7.1.2.38 leap()

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

Definition at line 1469 of file AG2.for.

7.1.2.39 line()

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

Definition at line 109 of file AG2.for.

7.1.2.40 locge()

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

Definition at line 2973 of file AG2.for.

7.1.2.41 locle()

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

Definition at line 2991 of file AG2.for.

7.1.2.42 logtix()

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

Definition at line 2052 of file AG2.for.

7.1.2.43 loptim()

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

Definition at line 998 of file AG2.for.

7.1.2.44 lwidth()

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

Definition at line 2742 of file AG2.for.

7.1.2.45 mnmx()

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

Definition at line 891 of file AG2.for.

7.1.2.46 monpos()

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

Definition at line 2169 of file AG2.for.

7.1.2.47 notatec()

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

Definition at line 2628 of file AG2.for.

7.1.2.48 npts()

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

Definition at line 165 of file AG2.for.

7.1.2.49 numsetc()

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

Definition at line 2326 of file AG2.for.

7.1.2.50 optim()

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

Definition at line 981 of file AG2.for.

7.1.2.51 oubgc()

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

Definition at line 1497 of file AG2.for.

7.1.2.52 place()

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

Definition at line 522 of file AG2.for.

7.1.2.53 remlab()

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

Definition at line 2817 of file AG2.for.

7.1.2.54 rescom()

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

Definition at line 3060 of file AG2.for.

7.1.2.55 rgchek()

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

Definition at line 864 of file AG2.for.

7.1.2.56 roundd()

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

Definition at line 3009 of file AG2.for.

7.1.2.57 roundu()

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

Definition at line 3025 of file AG2.for.

7.1.2.58 savcom()

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

Definition at line 3044 of file AG2.for.

7.1.2.59 setwin()

```
subroutine setwin
```

Definition at line 632 of file AG2.for.

7.1.2.60 sizel()

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

Definition at line 198 of file AG2.for.

7.1.2.61 sizes()

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

Definition at line 187 of file AG2.for.

7.1.2.62 slimx()

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

Definition at line 498 of file AG2.for.

7.1.2.63 slimy()

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

Definition at line 510 of file AG2.for.

7.1.2.64 spread()

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

Definition at line 2880 of file AG2.for.

7.1.2.65 stepl()

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

Definition at line 176 of file AG2.for.

7.1.2.66 steps()

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

Definition at line 131 of file AG2.for.

7.1.2.67 symbl()

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

Definition at line 120 of file AG2.for.

7.1.2.68 symout()

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

Definition at line 1867 of file AG2.for.

7.1.2.69 teksym()

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

Definition at line 1892 of file AG2.for.

7.1.2.70 teksym1()

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

Definition at line 1940 of file AG2.for.

7.1.2.71 tset()

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

Definition at line 2099 of file AG2.for.

7.1.2.72 tset2()

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

Definition at line 2137 of file AG2.for.

7.1.2.73 typck()

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

Definition at line 833 of file AG2.for.

7.1.2.74 vbarst()

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

Definition at line 702 of file AG2.for.

7.1.2.75 vlablc()

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

Definition at line 2653 of file AG2.for.

7.1.2.76 width()

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

Definition at line 2701 of file AG2.for.

7.1.2.77 xden()

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

Definition at line 322 of file AG2.for.

7.1.2.78 xetyp()

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

Definition at line 606 of file AG2.for.

7.1.2.79 xfrm()

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

Definition at line 400 of file AG2.for.

7.1.2.80 xlab()

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

Definition at line 300 of file AG2.for.

7.1.2.81 xlen()

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

Definition at line 374 of file AG2.for.

7.1.2.82 xloc()

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

Definition at line 256 of file AG2.for.

7.1.2.83 xloctp()

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

Definition at line 278 of file AG2.for.

7.1.2.84 xmfrm()

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

Definition at line 448 of file AG2.for.

7.1.2.85 xmtcs()

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

Definition at line 426 of file AG2.for.

7.1.2.86 xneat()

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

Definition at line 212 of file AG2.for.

7.1.2.87 xtics()

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

Definition at line 352 of file AG2.for.

7.1.2.88 xtype()

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

Definition at line 554 of file AG2.for.

7.1.2.89 xwdth()

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

Definition at line 580 of file AG2.for.

7.1.2.90 xzero()

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

Definition at line 234 of file AG2.for.

7.1.2.91 yden()

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

Definition at line 337 of file AG2.for.

7.1.2.92 yetyp()

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

Definition at line 619 of file AG2.for.

7.1.2.93 yfrm()

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

Definition at line 413 of file AG2.for.

7.1.2.94 ylab()

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

Definition at line 311 of file AG2.for.

7.1.2.95 ylen()

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

Definition at line 387 of file AG2.for.

7.1.2.96 yloc()

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

Definition at line 267 of file AG2.for.

7.1.2.97 ylocrt()

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

Definition at line 289 of file AG2.for.

7.1.2.98 ymdyd()

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

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

Definition at line 1414 of file AG2.for.

7.1.2.99 ymfrm()

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

Definition at line 461 of file AG2.for.

7.1.2.100 ymtcs()

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

Definition at line 437 of file AG2.for.

7.1.2.101 yneat()

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

Definition at line 223 of file AG2.for.

7.1.2.102 ytics()

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

Definition at line 363 of file AG2.for.

7.1.2.103 ytype()

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

Definition at line 567 of file AG2.for.

7.1.2.104 ywdth()

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

Definition at line 593 of file AG2.for.

7.1.2.105 yzero()

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

Definition at line 245 of file AG2.for.

7.2 AG2.for

```
00001 C> \file       AG2.for
00002 C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
00003 C> \version    (2024,347, x)
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C>  Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
00009 C> \note
00010 C>     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00011 C>     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00012 C>
00013 C> \~english
00014 C> Layer 2: scientific 2-D graphic subroutines
00015 C> \note
00016 C>     The control character for exponent (originally -1) is now SOH=char(1)
00017 C>     and for index (originally -2) STX=char(2).
00018 C>
00019 C> \~
00020 C> \note \verbatim
00021 C>   Package:
00022 C>    - AG2.for:         chart plotting routines
00023 C>    - AG2Holerith.for: deprecated routines
00024 C>    - AG2USR.for:      default userroutines
00025 C>    - G2dAG2.fd:       commonblock
00026 C> \endverbatim
00027 C
00028 C
00029 C  Tektronix Advanced Graphics 2 - Version 2.x
00030 C
00031 C
00032 C     Neuer Code in Fortran 77. Die Verwendung der im Manual dokumentierten
00033 C     Unterprogramme bleibt unveraendert, die direkte Manipulation von
00034 C     Variablen des zugrundeliegenden Commonblockes ist jedoch nicht mehr
00035 C     empfehlenswert. IBASEX (iPar) und IBASEY(iPar) mit ipar <>0,
00036 C     IBASEC, COMGET und COMSET sollten in neuen Programmen nicht verwendet
00037 C     werden.
00038 C
00039 C     Die Zwischenspeicherung der Statusvariablen ueber
```

```
00040 C           SAVCOM und RESCOM
00041 C     und die Achsensteuerung ueber
00042 C           IBASEX(0), IBASEY(0) und IOTHER
00043 C     werden weiterhin unterstuetzt.
00044 C
00045 C     Die Implementation der Unterprogramme COMGET und COMSET setzt die gleiche
00046 C     Laenge von REAL und INTEGER-Variablen voraus.
00047 C
00048 C     Da Holerithvariablen von modernen Compilern uneinheitlich unterstuetzt
00049 C     werden (4Habcd entweder als gepackte Integervariable oder als Character-
00050 C     variable interpretiert), wurden die folgenden Routinen angepasst:
00051 C      - subroutine PLACE (Lit): Lit wird nur noch als Ordnungszahl (1..13)
00052 C        und nicht mehr alternativ als Literal ('STD', 'UPH') interpretiert.
00053 C
00054 C     subroutine LEAP (iyear): Die Schaltjahrkorrektur erfolgt nicht mehr
00055 C     als SUBROUTINE ueber einen Common-Block, sondern direkt als
00056 C     integer function LEAP (iyear) ! = 1: Schaltjahr, sonst 0
00057 C
00058 C     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00059 C     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00060 C
00061 C     Intern erfolgt die Stringverarbeitung ueber Charactervariablen als
00062 C     nullterminierte C-Strings.
00063 C
00064 C     Der User-API wurden die folgenden Unterprogramme als Charactervarianten
00065 C     der Original-Holerithroutinen hinzugefuegt:
00066 C      - subroutine NUMSETC (fnum,nbase, outstr,fillstr)
00067 C      - subroutine FONLYC (fnum,iwidth,idec, outstr,fillstr)
00068 C      - subroutine EFORMC (fnum,iwidth,idec, outstr,fillstr)
00069 C      - subroutine EXPOUTC (nbase,iexp, outstr,fillstr)
00070 C      - subroutine ALFSETC (fnum,iwidth,labtyp,outstr)
00071 C      - subroutine NOTATEC (IX,IY,LENCHR,IARRAY)
00072 C      - subroutine JUSTERC
00073 C
00074 C      - subroutine USESETC (fnum, iwidth, nbase, labstr)
00075 C
00076 C      subroutine MONPOS (nbase,iy1,dpos, spos) ! spos ist INTEGER
00077 C      subroutine GLINE (nbase,datapt,spos) ! spos ist INTEGER
00078 C
00079 C     Der Code ab Version 2.0 wird nicht mehr fuer CP/M entwickelt. Letzte
00080 C     unter CP/M compilierbare Version: (2006, 013, 1)
00081 C
00082 C     Zugehoerige Module:
00083 C      - AG2.FOR:     Basisfunktionen
00084 C      - AG2Holerith: Veraltete Unterprogramme zur Wahrung der Kompatibilitaet
00085 C                     (Unterstuetzung Holerithvariablen und vektorisierter Zu-
00086 C                     griff auf den Commonblock)
00087 C      - AG2USR.FOR:  Userroutinen
00088 C      - G2dAG2.fd:   Commonblockdefinition
00089 C
00090
00091 C
00092 C  Ausgabe der Softwareversion
00093 C
00094       subroutine ag2lev (ilevel)
00095       implicit none
00096       integer ilevel(3)
00097
00098       call tcslev (ilevel) ! level(3)= System aus TCS
00099       ilevel(1)=2024       ! Aenderungsjahr
00100       ilevel(2)= 347       ! Aenderungstag
00101       return
00102       end
00103
00104
00105
00106 C
00107 C  Setzen allgemeiner Commonvariablen
00108 C
00109       subroutine line (ipar)
00110       implicit none
00111       integer ipar
00112       include 'G2dAG2.fd'
00113
00114       cline= ipar
00115       return
00116       end
00117
00118
00119
00120       subroutine symbl (ipar)
00121       implicit none
00122       integer ipar
00123       include 'G2dAG2.fd'
00124
00125       csymbl= ipar
00126       return
```

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

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

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

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

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

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

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

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

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

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

```
00997
00998       subroutine loptim (ixy)
00999       implicit none
01000       integer ixy ,i, labtyp, ntics, lsig, mtcs
01001       real dataint, amin,amax, aminor,amaxor, sigfac
01002       integer idataint
01003       integer mintic
01004       integer LINWDT, LINHGT
01005       real ROUNDD, ROUNDU
01006       include 'G2dAG2.fd'
01007
01008       labtyp=abs( cxylab(ixy)) ! <0: Userlabel
01009       if (labtyp .le. 1) labtyp= cxytype(ixy) ! Default: Achsentyp = Datentyp
01010
01011       amin= cxydmin(ixy)
01012       amax= cxydmax(ixy)
01013       ntics= abs(cxytics(ixy)) ! Anzahl >=1, 0= Flag fuer autoscale
01014       mintic= 0
01015
01016       if (labtyp .eq. 2) then ! logarithmische Achsen
01017        amin= log10(max(amin,1./cinfin)) + 1.e-7  ! !> 0 => log10 definiert
01018        amax= log10(amax)
01019       end if
01020
01021       aminor= amin
01022       amaxor= amax
01023
01024       if (ntics .eq. 0) then ! = F( X-Achsenlaenge,Buchstabengroesse)
01025        if (ixy.eq.1) then
01026         i= linwdt(8) ! 100 + LINWDT(3)
01027        else
01028         i= linhgt(3) ! 50 + LINHGT(3)
01029        end if
01030        ntics= (cxysmax(ixy) - cxysmin(ixy)) / i
01031        if (ntics .lt. 1) ntics= 1
01032       end if
01033       dataint= abs(amax-amin) / real(ntics)
01034
01035 310   continue ! repeat...
01036        if (labtyp .eq. 2) dataint= roundu(dataint,1.) ! logarithmische Achsen
01037        lsig= rounddlog10(dataint),1.) ! Anzahl signifikanter Nachkommastellen
01038        sigfac=10.**(lsig)
01039        if (cxyneat(ixy)) then ! Achsenteilung aus Tabelle
01040         if(labtyp .ne. 2) then ! nicht bei log. Achsen
01041          if ((dataint/sigfac) .le. 1.) then
01042           dataint= 1. * sigfac
01043           mintic= 10
01044          else if ((dataint/sigfac) .le. 2.) then
01045           dataint= 2. * sigfac
01046           mintic= 2
01047          else if ((dataint/sigfac) .le. 2.5) then
01048           dataint= 2.5 * sigfac
01049           mintic= 5
01050           lsig=lsig-1
01051          else if ((dataint/sigfac) .le. 5.) then
01052           dataint= 5. * sigfac
01053           mintic=  5
01054          else if ((dataint/sigfac) .le. 10.) then
01055           dataint= 10. * sigfac
01056           mintic= 10
01057           lsig=lsig+1
01058          else
01059           dataint= cinfin
01060           mintic= 0
01061          end if
01062         end if ! log. Achse
01063        else ! .not. neat
01064         lsig=lsig-2
01065        end if
01066        if (lsig .ge. 0) lsig=lsig+1
01067       if (cxyneat(ixy) .or. (labtyp .eq. 2) ) then ! ... until
01068        amin= rounddamin+.01*sigfac,dataint) !  runde auf TicIntervall
01069        amax= roundu(amax-.01*sigfac,dataint) ! .01*sigfac= Genauigkeit Plot
01070        ntics= int(abs(amax-amin)/dataint+.0001)
01071        if(cxytics(ixy) .ne. 0) then ! until: ntics nicht vorbesetzt oder = vorbesetzt
01072         if(abs(cxytics(ixy)) .lt. ntics) then
01073          dataint= dataint * 1.1
01074          amin=aminor
01075          amax=amaxor
01076          goto 310 ! noch eine Iterationsschleife
01077         else if (abs(cxytics(ixy)) .gt. ntics) then
01078          ntics= abs(cxytics(ixy))
01079          amax= amin + real(ntics) * dataint
01080         end if ! abs(cxytics(ixy)) .eq. ntics: no action
01081        end if
01082       end if
01083       cxytics(ixy)= ntics
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
03085       return
03086       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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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
00080             txtstring(iactlen:iactlen)= char(i2)
00081           end if
00082           if (iactlen.ge.iprntlen) then
00083             txtstring(iactlen+1:iactlen+1) = char(0)
00084             call toutstc (txtstring)
00085           end if
00086         else if (action.eq.10) then ! XACTION_BCKCOL
00087           call bckcol(i1)
00088         else if (action.eq.11) then ! XACTION_LINCOL
00089           call lincol (i1)
00090         else if (action.eq.12) then ! XACTION_TXTCOL
00091           call txtcol (i1)
00092         else if (action.eq.13) then ! XACTION_FONTATTR
00093           if (i1.eq.0) call italir()
00094           if (i1.eq.1) call italic()
00095           if (i2.eq.0) call nrmsiz()
00096           if (i2.eq.1) call dblsiz()
00097         else if (action.eq.14) then ! XACTION_NOOP
00098           continue
00099         else if (action.eq.15) then ! XACTION_CLIP
00100           if (i1.eq.0) then ! clipping not active
00101             kminsx= 0
00102             kminsy= 0
00103             kmaxsx= 1023 ! TEK_XMAX
00104             kmaxsy= 780 ! TEK_YMAX
00105             call swind1(kminsx,kminsy,kmaxsx,kmaxsy) ! Set bool ClippingNotActive
00106           end if
00107         else if (action.eq.16) then ! XACTION_CLIP1
00108           kminsx= i1
00109           kminsy= i2
00110           call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
```

```
00111         else if (action.eq.17) then ! XACTION_CLIP2
00112           kmaxsx= i1
00113           kmaxsy= i2
00114           call swindl(kminsx,kminsy,kmaxsx,kmaxsy)
00115         else ! unknown
00116           continue
00117         end if
00118       if (ios.eq.0) goto 10 ! until EOF
00119
00120       close (iunit)
00121       gethdc= .false.
00122       return
00123       end
```

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

```
00001 C> \file       Strings.for
00002 C> \brief      TCS: String functions
00003 C> \version    1.26
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Hilfsfunktionen zur Fortran Stringverarbeitung
00008 C> \~english
00009 C> Fortran utility functions for string processing
00010 C> \~
00011 C>
00012 C
00013 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
00014 C
00015 C  Unterprogramme zur Behandlung von Fortran-Strings.
00016 C  Die Stringenden werden entweder durch CHAR(0) markiert oder
00017 C  ueber die Deklaration ermittelt.
00018 C
00019 C     9.11.88     K. Friedewald
00020 C
00021 C  Ergaenzungen:
00022 C     iTrimLen
00023 C
00024 C     7.12.01     K. Friedewald
00025 C
00026 C  Version: 1.26
00027 C
00028 Ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
00029
00030       subroutine substitute (Source, Destination, Old1, New1)
00031 C
00032 C  Durchsucht SOURCE nach den Substrings OLD, ersetzt sie durch NEW
00033 C  und uebergibt das Ergebniss in DESTINATION. Wenn New=CHAR(0), werden
00034 C  die vorkommenden OLD nur geloescht.
00035 C
00036 C  Stringenden koennen durch CHAR(0) markiert werden.
00037 C
00038       implicit none
00039       integer iNext, iNext2, TempLen
00040       integer iStringLen
00041       character *(*) Source, Destination, Old1, New1
00042       character*255 temp, old, new
00043
00044       if (istringlen(old1).le.0) return
00045       if (istringlen(source) .le. 0) then
00046        destination= char(0)
00047        return
00048       end if
00049
00050       old= old1 // char(0)          ! old evtl. = Destination
00051       new= new1 // char(0)          ! => retten!
00052
00053       temp= source(1:istringlen(source)) // char(0) ! evtl. Ueberlappung!
00054       destination= temp
00055       inext= index( destination(:istringlen(destination)),
00056      1                                      old(:istringlen(old)) )
00057       do while (inext.gt.0)
00058        if (inext.eq.1) then
00059         temp= destination
00060         if (new.eq.char(0)) then
00061          destination= temp(istringlen(old)+1:)
00062         else
00063          destination= new(:istringlen(new)) // temp(istringlen(old)+1:)
00064         end if
00065        else
00066         temp= destination(1:inext-1)
```

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

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

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

4.1

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 339 of file TCS.for.

7.28.2.2 anstr()

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

Definition at line 329 of file TCS.for.

7.28.2.3 baksp()

```
subroutine baksp
```

Definition at line 384 of file TCS.for.

7.28.2.4 cartn()

```
subroutine cartn
```

Definition at line 365 of file TCS.for.

7.28.2.5 dasha()

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

Definition at line 290 of file TCS.for.

7.28.2.6 dashr()

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

Definition at line 236 of file TCS.for.

7.28.2.7 drawa()

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

Definition at line 257 of file TCS.for.

7.28.2.8 drawr()

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

Definition at line 212 of file TCS.for.

7.28.2.9 dwindo()

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

Definition at line 462 of file TCS.for.

7.28.2.10 genflg()

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

Definition at line 558 of file TCS.for.

7.28.2.11 home()

```
subroutine home
```

Definition at line 518 of file TCS.for.

7.28.2.12 linef()

```
subroutine linef
```

Definition at line 374 of file TCS.for.

7.28.2.13 linhgt()

```
function linhgt (
              Numlin )
```

Definition at line 400 of file TCS.for.

7.28.2.14 lintrn()

```
subroutine lintrn
```

Definition at line 418 of file TCS.for.

7.28.2.15 linwdt()

```
function linwdt (
              NumChr )
```

Definition at line 408 of file TCS.for.

7.28.2.16 logtrn()

```
subroutine logtrn (
              IMODE )
```

Definition at line 428 of file TCS.for.

7.28.2.17 movea()

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

Definition at line 268 of file TCS.for.

7.28.2.18 mover()

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

Definition at line 220 of file TCS.for.

7.28.2.19 newlin()

```
subroutine newlin
```

Definition at line 357 of file TCS.for.

7.28.2.20 newpag()

```
subroutine newpag
```

Definition at line 392 of file TCS.for.

7.28.2.21 pointa()

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

Definition at line 279 of file TCS.for.

7.28.2.22 pointr()

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

Definition at line 228 of file TCS.for.

7.28.2.23 rel2ab()

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

Definition at line 244 of file TCS.for.

7.28.2.24 rescal()

```
subroutine rescal
```

Definition at line 481 of file TCS.for.

7.28.2.25 revcot()

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

Definition at line 314 of file TCS.for.

7.28.2.26 rrotat()

```
subroutine rrotat (
              Grad )
```

Definition at line 501 of file TCS.for.

7.28.2.27 rscale()

```
subroutine rscale (
              Faktor )
```

Definition at line 510 of file TCS.for.

7.28.2.28 seetrm()

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

Definition at line 536 of file TCS.for.

7.28.2.29 seetrn()

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

Definition at line 547 of file TCS.for.

7.28.2.30 setmrg()

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

Definition at line 527 of file TCS.for.

7.28.2.31 swindo()

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

Definition at line 450 of file TCS.for.

7.28.2.32 twindo()

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

Definition at line 443 of file TCS.for.

7.28.2.33 vcursr()

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

Definition at line 202 of file TCS.for.

7.28.2.34 vwindo()

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

Definition at line 469 of file TCS.for.

7.28.2.35 wincot()

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

Definition at line 301 of file TCS.for.

7.29 TCS.for

```
00001 C> \file       TCS.for
00002 C> \brief      TCS: Tektronix Plot 10 Emulation
00003 C> \version    4.1
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Systemübergreifende TCS-Routinen
00008 C> \~english
00009 C> System independent subroutines
00010 C> \~
00011 C
00012 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00013 C
00014 C      26.07.23 Version 5.0:
00015 C               Einheitliche Version CPM/DOS/Windows/SDL2/wX
00016 C
00017 C      27.11.20 Version 4.0:
00018 C               Einheitliche Version CPM/DOS/Windows/SDL2
00019 C
00020 C      17.08.20 Version 3.2
00021 C               Harmonisierung der Verwendung des Commonblocks TKTRNX
00022 C               Variable KHOMEY wird jetzt (analog alter DOS-Version) verwendet.
00023 C               Da KHOMEY nicht in der CP/M Version vorhanden ist, muss ab dieser
00024 C               Version fuer eine Complilation unter CP/M die entsprechende Zeile
00025 C               in der SUBROUTINE HOME geändert werden.
00026 C
00027 C      13.11.17 Version 3.1
00028 C               Anpassung an OpenWatcom 2.0
00029 C               Bugfix: Unterscheidung Aufrufe ueber windowsx.h (win16) und GDI (win32)
00030 C                - SelectPen -> SelectObject
00031 C                - DeletePen -> DeleteObject
00032 C                - DeleteBrush -> DeleteObject
00033 C                - GetStockBrush -> GetStockObject
00034 C                - DeleteRgn -> DeleteObject
00035 C                - SelectFont -> SelectObject
00036 C                - DeleteFont -> DeleteObject
00037 C
00038 C      27.03.13 Version 3.0
00039 C               Anpassung an Windows 7 und OpenWatcom 1.9
00040 C               Anpassung an gfortran anstelle von g77 der GCC
00041 C
00042 C      22.12.05 Version 2.19
00043 C               Elimination berechnetes GOTO in LOGTRN
00044 C
00045 C      18.10.05 Version 2.18
00046 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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassung an DOS:
C
C      Aenderungen gegenueber CP/M-Version:
C               SEELOC, DCURSR, SVSTAT, RESTAT, CSIZE in TCSdrDOS.FOR
C      Bugfix:  DASHA, DASHR - Korrektur Parameterliste
C               SEETRM - ibaud statt ibaudr
C
C      Zugehoerige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TCSdrDOS.FOR  Bildschirmtreiber
C               TCSdDOSa.ASM  Betriebssystemspezifische Low-Level Routinen
C               HDCOPY.FOR    Hardcopyroutine
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C               OUTTEXT.FOR   nur für WATCOM-Compiler
C
C      25.10.01 Version 2.00:  Dr.-Ing. K. Friedewald
C
C      07.02.02 Version 2.10:
C               Implementierung multilinguale Fehlermeldungen
C
C      11.10.02 Version 2.12:
```

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



C
C Graphic Input
C

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

C
C  Virtuelle Graphik, relativ
C

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



      subroutine mover (X,Y)
```

```
00221       call rel2ab (x,y,xabs,yabs)
00222       call movea (xabs,yabs)
00223       return
00224       end
00225
00226
00227
00228       subroutine pointr (X,Y)
00229       call rel2ab (x,y,xabs,yabs)
00230       call pointa (xabs,yabs)
00231       return
00232       end
00233
00234
00235
00236       subroutine dashr (X,Y, iL)
00237       call rel2ab (x,y,xabs,yabs)
00238       call dasha (xabs,yabs, il)
00239       return
00240       end
00241
00242
00243
00244       subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
00245       include 'Tktrnx.fd'
00246       call seeloc (ix,iy)
00247       call revcot (ix,iy,xabs,yabs)
00248       xabs= (( xrel*trcosf - yrel*trsinf)*trscal)+xabs
00249       yabs= (( xrel*trsinf + yrel*trcosf)*trscal)+yabs
00250       return
00251       end
00252
00253 C
00254 C  Virtuelles Zeichnen, absolut
00255 C
00256
00257       subroutine drawa (X,Y)
00258       include 'Tktrnx.fd'
00259       call wincot (x,y,ix,iy)
00260       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00261       call drwabs (ix,iy)
00262       call swind1 (0,0,1023,780)
00263       return
00264       end
00265
00266
00267
00268       subroutine movea (X,Y)
00269       include 'Tktrnx.fd'
00270       call wincot (x,y,ix,iy)
00271       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00272       call movabs (ix,iy)
00273       call swind1 (0,0,1023,780)
00274       return
00275       end
00276
00277
00278
00279       subroutine pointa (X,Y)
00280       include 'Tktrnx.fd'
00281       call wincot (x,y,ix,iy)
00282       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00283       call pntabs (ix,iy)
00284       call swind1 (0,0,1023,780)
00285       return
00286       end
00287
00288
00289
00290       subroutine dasha (X,Y, iL)
00291       include 'Tktrnx.fd'
00292       call wincot (x,y,ix,iy)
00293       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00294       call dshabs (ix,iy, il)
00295       call swind1 (0,0,1023,780)
00296       return
00297       end
00298
00299
00300
00301       subroutine wincot (X,Y,IX,IY)
00302       include 'Tktrnx.fd'
00303       dx= x-tminvx
00304       dy= y-tminvy
00305       if ((xlog.lt.255.).and.(x.gt.0.)) dx= alog(x)-xlog
00306       if ((ylog.lt.255.).and.(y.gt.0.)) dy= alog(y)-ylog
00307       ix= ifix(dx*xfac+.5)+kminsx
```

```
00308       iy= ifix(dy*yfac+.5)+kminsy
00309       return
00310       end
00311
00312
00313
00314       subroutine revcot (IX,IY,X,Y)
00315       include 'Tktrnx.fd'
00316       dx= float(ix-kminsx) / xfac
00317       dy= float(iy-kminsy) / yfac
00318       x= dx + tminvx
00319       y= dy + tminvy
00320       if (xlog.lt.255.) x= 2.718282**(dx+xlog)
00321       if (ylog.lt.255.) y= 2.718282**(dy+ylog)
00322       return
00323       end
00324
00325 C
00326 C  Alphanumerische Ausgabe
00327 C
00328
00329       subroutine anstr (NChar, IStrin)
00330       dimension istrin(1)
00331       do 10 i=1,nchar
00332        call ancho (istrin(i))
00333 10    continue
00334       return
00335       end
00336
00337
00338
00339       subroutine ancho (ichar)
00340       include 'Tktrnx.fd'
00341
00342       if (ichar.gt.31) goto 10
00343       if (ichar.eq.7) call bell
00344       if (ichar.eq.10) call linef
00345       if (ichar.eq.13) call cartn
00346       return
00347
00348  10   call seeloc (ix,k)
00349       call csize (ixlen,k)
00350       if (ix.gt.krmrgn-ixlen) call newlin
00351       call toutpt (ichar)
00352       return
00353       end
00354
00355
00356
00357       subroutine newlin
00358       call cartn
00359       call linef
00360       return
00361       end
00362
00363
00364
00365       subroutine cartn
00366       include 'Tktrnx.fd'
00367       call seeloc (ix,iy)
00368       call movabs (klmrgn,iy)
00369       return
00370       end
00371
00372
00373
00374       subroutine linef
00375       call seeloc (j,iy)
00376       call csize (j,iylen)
00377       if (iy.lt.iylen) call home
00378       call movrel (0,-iylen)
00379       return
00380       end
00381
00382
00383
00384       subroutine baksp
00385       call csize (ix,iy)
00386       call movrel (-ix,0)
00387       return
00388       end
00389
00390
00391
00392       subroutine newpag
00393       call erase
00394       call home
```

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

```
00482       include 'Tktrnx.fd'
00483       xfac= 0.
00484       yfac= 0.
00485       if ((tmaxvx.eq.tminvx) .or. (tmaxvy.eq.tminvy)) return
00486       dx= tmaxvx-tminvx
00487       dy= tmaxvy-tminvy
00488       if ((xlog.eq.255.).or.(amin1(tminvx,tmaxvx).le.0.)) goto 10
00489        xlog= alog(tminvx)
00490        dx= alog(tmaxvx)-xlog
00491 10    if ((ylog.eq.255.).or.(amin1(tminvy,tmaxvy).le.0.)) goto 20
00492        ylog= alog(tminvy)
00493        dy= alog(tmaxvy)-ylog
00494 20    xfac= float(kmaxsx-kminsx) / dx
00495       yfac= float(kmaxsy-kminsy) / dy
00496       return
00497       end
00498
00499
00500
00501       subroutine rrotat (Grad)
00502       include 'Tktrnx.fd'
00503       trsinf= sin(grad/57.29578)
00504       trcosf= cos(grad/57.29578)
00505       return
00506       end
00507
00508
00509
00510       subroutine rscale (Faktor)
00511       include 'Tktrnx.fd'
00512       trscal= faktor
00513       return
00514       end
00515
00516
00517
00518       subroutine home
00519       include 'Tktrnx.fd'
00520 C      call movabs(klmrgn,750) Fuer CP/M (kein khomey verfuegbar, -> !=750)
00521       call movabs(klmrgn,khomey)
00522       return
00523       end
00524
00525
00526
00527       subroutine setmrg (Mlinks, Mrecht)
00528       include 'Tktrnx.fd'
00529       klmrgn= mlinks
00530       krmrgn= mrecht
00531       return
00532       end
00533
00534
00535
00536       subroutine seetrm (IBaud,Iterm,ICSize,MaxScr)
00537       include 'Tktrnx.fd'
00538       ibaud= 0
00539       iterm= 1
00540       icsize= 1
00541       maxscr= 1023
00542       return
00543       end
00544
00545
00546
00547       subroutine seetrn (xf,yf,key)
00548       include 'Tktrnx.fd'
00549       xf= xfac
00550       yf= yfac
00551       key= 1
00552       if ((xlog.lt.255.).or.(ylog.lt.255.)) key=2
00553       return
00554       end
00555
00556
00557
00558       logical function genflg (ITEM)
00559       genflg= item.eq.0
00560       return
00561       end
```

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

```
00001 C> \file       TCSdrSDL.for
00002 C> \brief      SDL Port: High-Level Driver
00003 C> \version    (2022,305,6)
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C> SDL2-spezifische TCS-Routinen
00009 C> \note \verbatim
00010 C>    Erweiterungen gegenüber Tektronix:
00011 C>     subroutine TOUTSTC (String): Ausgabe Fortran-String
00012 C>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
00013 C>     subroutine TXTCOL (iCol): Setzen Textfarbe
00014 C>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00015 C>     subroutine DefaultColour: Wiederherstellung Defaultfarben
00016 C> \endverbatim
00017 C>
00018 C>
00019 C> \~english
00020 C> SDL2 specific subroutines
00021 C> \note \verbatim
00022 C>    Supplement to Tektronix:
00023 C>     subroutine TOUTSTC (String): Ausgabe Fortran-String
00024 C>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
00025 C>     subroutine TXTCOL (iCol): Setzen Textfarbe
00026 C>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00027 C>     subroutine DefaultColour: Wiederherstellung Defaultfarben
00028 C> \endverbatim
00029 C> \~
00030 C>
00031
00032
00033
00034 C
00035 C  Ausgabe der Softwareversion
00036 C
00037       subroutine tcslev(LEVEL)
```

```
00038       integer LEVEL(3)
00039       level(1)=2022     ! Aenderungsjahr
00040       level(2)= 305     ! Aenderungstag
00041       level(3)=   6     ! System= SDL
00042       return
00043       end
00044
00045
00046
00047 C
00048 C>  Initialisierung Hard- und Software
00049 C
00050       subroutine initt (iDummy)
00051       include 'Tktrnx.fd'
00052       call initt1 ! Init Hardware
00053       call initt2 ! Reset Common TKTRNX ohne Einfluss auf das Journal
00054       call nrmsiz
00055       call italir
00056       call home
00057       return
00058       end
00059
00060
00061
00062       subroutine initt2
00063 C INITT2 auch durch RepaintBuffer aufgerufen -> Schreiben Journal unmoeglich!
00064       include 'Tktrnx.fd'
00065       call lintrn
00066       call swindo (0,1023,0,780)
00067       call vwindo (0.,1023.,0.,780.)
00068       call rrotat (0.)
00069       call rscale (1.)
00070       call setmrg (0,1023)
00071       return
00072       end
00073
00074
00075
00076
00077 C
00078 C  Abspeichern Terminal Status Area (wie MS Windows und DOS)
00079 C
00080
00081       subroutine svstat (Array)
00082       integer array(1)
00083       include 'Tktrnx.fd'
00084       integer arr(1)
00085       equivalence(arr(1),khomey)
00086       do 10 i=1,itktrnxl
00087        array(i)= arr(i)
00088 10    continue
00089       return
00090       end
00091
00092
00093
00094       subroutine restat (Array)
00095       integer array(1)
00096       include 'Tktrnx.fd'
00097       integer arr(1)
00098       equivalence(arr(1),khomey)
00099       do 10 i=1,itktrnxl
00100        arr(i)= array(i)
00101 10    continue
00102       call movabs (kbeamx, kbeamy)
00103       return
00104       end
00105
00106
00107
00108 C
00109 C  Relative Zeichenbefehle (wie MS Windows und DOS)
00110 C
00111
00112       subroutine movrel (iX, iY)
00113       include 'Tktrnx.fd'
00114       ixx= kbeamx + ix
00115       iyy= kbeamy + iy
00116       call movabs (ixx, iyy)
00117       return
00118       end
00119
00120
00121
00122       subroutine pntrel (iX, iY)
00123       include 'Tktrnx.fd'
00124       ixx= kbeamx + ix
```

```
00125       iyy= kbeamy + iy
00126       call pntabs (ixx, iyy)
00127       return
00128       end
00129
00130
00131
00132       subroutine drwrel (iX, iY)
00133       include 'Tktrnx.fd'
00134       ixx= kbeamx + ix
00135       iyy= kbeamy + iy
00136       call drwabs (ixx, iyy)
00137       return
00138       end
00139
00140
00141
00142       subroutine dshrel (iX, iY, iMask)
00143       include 'Tktrnx.fd'
00144       ixx= kbeamx + ix
00145       iyy= kbeamy + iy
00146       call dshabs (ixx, iyy, imask)
00147       return
00148       end
00149
00150
00151
00152 C
00153 C   Ersatz SEELOC der CP/M-Version (wie MS Windows, DOS)
00154 C
00155
00156       subroutine seeloc (IX,IY)
00157       include 'Tktrnx.fd'
00158       ix= kbeamx
00159       iy= kbeamy
00160       return
00161       end
00162
00163
00164
00165 C
00166 C  Textausgabe
00167 C
00168
00169       subroutine toutpt (iChr)
00170       include 'Tktrnx.fd'
00171       call outgtext (char(ichr))
00172       return
00173       end
00174
00175
00176
00177       subroutine toutst (nChr, iChrArr)
00178       integer iChrArr (1)
00179       if (nchr.eq.0) return
00180       do 10 i=1,nchr
00181        call toutpt (ichrarr(i))
00182 10    continue
00183       return
00184       end
00185
00186
00187
00188       subroutine toutstc (String)
00189       character *(*) String
00190       call outgtext (string)
00191       return
00192       end
00193
00194
00195
00196       subroutine statst (String)
00197       character *(*) String
00198       call outtext (string)
00199       return
00200       end
00201
00202
00203
00204 C
00205 C  Eingabe
00206 C
00207
00208       subroutine tinput (iChr)
00209       call dcursr (ichr, ichr,ichr)
00210 C     Aufruf von DCURSR mit ix=iy: Maustasten ausser Funktion
00211       return
```

```
00212       end
00213
00214
00215
00216 C
00217 C>  Entry Dummyroutinen
00218 C
00219       subroutine  anmode
00220 C> AlfMod
00221       entry       alfmod
00222 C> pClipt
00223       entry       pclipt
00224 C> alpha
00225       entry       alpha
00226       return
00227       end
00228
00229
00230
00231       logical function winselect (iDummy)
00232       winselect= .false.
00233       return
00234       end
00235
```

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(IniFilNam))
- 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

```
00001 /** ****************************************************************************
00002 \file       TCSdSDLc.c
00003 \brief      SDL Port: Low-Level Driver
00004 \version    1.5
00005 \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          Systemnahe Graphikroutinen für die Tektronix Emulation
00009 \note \verbatim
00010           1. Falls der erste Buchstabe des Fensternamens ein '~' ist, wird
00011              das betreffende Fenster ohne Titel und Rahmen gezeichnet.
00012           2. Die System- und Statusmeldungen erfolgen in einem eigenen
00013              einzeiligem Fenster. Falls die Statusfensterhöhe <= 0 ist,
00014              erfolgen nur noch Systemfehlermeldungen über den Error-Channel.
00015           3. Der Videotreiber des Raspberry Pi4 kann über SSH keine zwei
00016              unabhängige Renderer für die beiden Fenster verwalten. Jedoch
00017              liefert der zweite Aufruf von SDL_CreateRenderer für das
00018              Statusfenster keinen Errorcode, sondern führt zu einem Programm-
00019              absturz. Entweder MUSS hier die Statusfensterhöhe <= 0 gesetzt
00020              oder X11 gestartet sein.
00021           4. Durch den Parameter HIGHQUALCHAR erfolgt die Textausgabe "Blended".
00022              Zur Performancesteigerung kann bei leistungsschwachen Systemen
00023              durch Auskommentieren auf "Solid" gewechselt werden.
00024 \endverbatim
00025 \~english
00026          system-specific subroutines of the Tektronix emulation
00027 \note \verbatim
00028           1. If the first letter of the window name is '~', the window will be
00029              drawn without title and frame.
00030           2. System- and status messages are shown in an one-line window. If
00031              the height of the window is <= 0, only system errors are signaled
00032              through the error channel.
00033           3. When called inside a ssh terminal, the Raspberry Pi videodriver
```

```
00034              crashes during the second call of SDL_renderer . If the height of
00035              the status window is 0, no problem arises.
00036           4. If the parameter HIGHQUALCHAR is defined, textoutput is "Blended".
00037              Undefining HIGHQUALCHAR on slow systems changes output to "Solid".
00038 \endverbatim
00039 \~
00040 ********************************************************************************* */
00041
00042 /*
00043         Anmerkungen:
00044           1. In der Routine WINLBL werden die SDL-Funktion SDL_GetBasePath ()
00045              sowie SDL_free verwendet. In der Dokumentation ist jedoch nicht
00046              explizit beschrieben, dass diese Funktion immer (wie SDL_logxxx)
00047              bereits vor dem Aufruf von SDL_Init() funktioniert. Die in der
00048              Source herauskommentierten Zeilen
00049              SDL_Init (0); und SDL_Quit(); koennen dann bei Problemen wieder
00050              verwendet werden.
00051           2. Skalierung vom Tektronix- auf das Bildschirmkoordinatensystem muss
00052              von Hand erfolgen, da SDL_RenderSetLogicalSize nicht durchgängig
00053              implementiert ist (Bug bis SDL2 Version 2.0.5 verifiziert).
00054              Insbesondere verwendet DrawLine die Skalierung nicht bei geneigten
00055              Geraden.
00056           3. Journalfile wird verwendet um Hardcopies erzeugen zu können
00057
00058 */
00059
00060
00061 /*
00062 ------------------- Konfiguration des Zielystems -------------------------------
00063 */
00064
00065 #define INIFILEXT ".xml"
00066 #define FNTFILEXT ".ttf"
00067 #define AUDIOSUPPORT
00068 #define HIGHQUALCHAR
00069
00070
00071 /*
00072 ------------------- Debug Switches ---------------------------------------------
00073 */
00074
00075 #define LOGLEVEL   SDL_LOG_PRIORITY_ERROR
00076 // #define LOGLEVEL   SDL_LOG_PRIORITY_DEBUG
00077 // #define LOGLEVEL   SDL_LOG_PRIORITY_VERBOSE // Ausgaben < Error in Fehlerkanal
00078 // #define TRACE_CALLS // zusaetzliche Debugausgaben
00079
00080
00081 /*
00082 ------------------- Headerfiles ------------------------------------------------
00083 */
00084
00085 #include <stdlib.h>
00086 #include <string.h>
00087 #include <stdio.h> // Fuer HDCOPY: sprintf
00088
00089 #ifdef AUDIOSUPPORT
00090  #include <math.h>
00091 #endif
00092
00093 #include "SDL.h"
00094 #include "SDL_ttf.h"
00095
00096 #ifdef AUDIOSUPPORT
00097  #include "SDL_audio.h"
00098 #endif
00099
00100 #include "mxml.h"
00101
00102 #include "sglib.h"
00103
00104 #include "TCSdSDLc.h"
00105 #include "TKTRNX.h"
00106
00107
00108 /*
00109 ------------------- Globale Variablen ------------------------------------------
00110 */
00111
00112 static int      TCSEventFilterData; // Userdata, z.Zt. nicht verwendet
00113
00114 static  float   PixFacX, PixFacY; // Anpassung Bildschirmauflösung
00115
00116 static  bool    TCSinitialized = false,
00117                 ClippingNotActive = true;
00118
00119 static char     szTCSWindowName[TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME,
00120                 szTCSstatWindowName[TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME,
```

```
00121                 szTCSIniFile[TCS_FILE_NAMELEN] = "",
00122                 szTCSHardcopyFile[TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME,
00123                 szTCSGraphicFont[TCS_FILE_NAMELEN] = TCS_INIDEF_FONT,
00124                 szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT,
00125                 szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0;
00126
00127 static  int     TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX, // rel. Bildschirmpos.
00128                 TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY, // bei Init in %
00129                 TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX,
00130                 TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY,
00131                 TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX, // dito
00132                 TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY, // Statusfenster
00133                 TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX,
00134                 TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY,
00135                 TextLineHeight,
00136                 TCSDefaultLinCol = TCS_INIDEF_LINCOL,
00137                 TCSDefaultTxtCol = TCS_INIDEF_TXTCOL,
00138                 TCSDefaultBckCol = TCS_INIDEF_BCKCOL,
00139                 iHardcopyCount = 1;  // Zähler zur Erzeugung Filenamen
00140
00141
00142
00143 /*
00144   Zuordnung Fehlernummern zu Meldungen
00145 */
00146
00147 typedef char   ErrMsg[TCS_MESSAGELEN];
00148 static  ErrMsg  szTCSErrorMsg[(int) MSG_MAXERRNO+1] =
00149                 {"Element 0 unused","DOS",
00150                 TCS_INIDEF_UNKNGRAPHCARD, // Errno 2
00151                 TCS_INIDEF_NOFNTFIL,      // Errno 3
00152                 TCS_INIDEF_NOFNT,         // Errno 4
00153                 "DOS",
00154                 TCS_INIDEF_HDCOPN,        // Errno 6
00155                 TCS_INIDEF_HDCWRT,        // Errno 7
00156                 TCS_INIDEF_HDCINT,        // Errno 8
00157                 TCS_INIDEF_USR,           // Errno 9
00158                 TCS_INIDEF_HDCACT,        // Errno 10
00159                 TCS_INIDEF_USRWRN,        // Errno 11
00160                 TCS_INIDEF_EXIT,          // Errno 12
00161                 "Windows",
00162                 "Windows",
00163                 TCS_INIDEF_JOUCREATE,     // Errno 15
00164                 TCS_INIDEF_JOUENTRY,      // Errno 16
00165                 TCS_INIDEF_JOUADD,        // Errno 17
00166                 TCS_INIDEF_JOUCLR,        // Errno 18
00167                 TCS_INIDEF_JOUUNKWN,      // Errno 19
00168                 TCS_INIDEF_XMLPARSER,     // Errno 20
00169                 TCS_INIDEF_XMLOPEN,       // Errno 21
00170                 TCS_INIDEF_UNKNAUDIO,     // Errno 22
00171                 TCS_INIDEF_USR2,          // Errno 23
00172                 TCS_INIDEF_INI2,          // Errno 24
00173                 "Maxerr only for internal Use" };
00174
00175 static  int     TCSErrorLev[(int) MSG_MAXERRNO+1] =
00176                 {10,10,
00177                 TCS_INIDEF_UNKNGRAPHCARDL,// Errno 2
00178                 TCS_INIDEF_NOFNTFILL,     // Errno 3
00179                 TCS_INIDEF_NOFNTL,        // Errno 4
00180                 10,
00181                 TCS_INIDEF_HDCOPNL,       // Errno 6
00182                 TCS_INIDEF_HDCWRTL,       // Errno 7
00183                 TCS_INIDEF_HDCINTL,       // Errno 8
00184                 TCS_INIDEF_USRL,          // Errno 9
00185                 TCS_INIDEF_HDCACTL,       // Errno 10
00186                 TCS_INIDEF_USRWRNL,       // Errno 11
00187                 TCS_INIDEF_EXITL,         // Errno 12
00188                 10,
00189                 10,
00190                 TCS_INIDEF_JOUCREATEL,    // Errno 15
00191                 TCS_INIDEF_JOUENTRYL,     // Errno 16
00192                 TCS_INIDEF_JOUADDL,       // Errno 17
00193                 TCS_INIDEF_JOUCLRL,       // Errno 18
00194                 TCS_INIDEF_JOUUNKWNL,     // Errno 19
00195                 TCS_INIDEF_XMLPARSERL,    // Errno 20
00196                 TCS_INIDEF_XMLOPENL,      // Errno 21
00197                 TCS_INIDEF_UNKNAUDIOL,    // Errno 22
00198                 TCS_INIDEF_USR2L,         // Errno 23
00199                 TCS_INIDEF_INI2L,         // Errno 24
00200                 10};
00201
00202
00203
00204 /*
00205   Zuordnung der Farbennummern zur VGA-Palette
00206 */
00207
```

```
00208 static  SDL_Color sdlColorTable[] = {
00209                 {240,240,240,SDL_ALPHA_OPAQUE }, /* iCol= 00: weiss (DOS: 01) */
00210                 {  0,  0,  0,SDL_ALPHA_OPAQUE }, /* iCol= 01: schwarz(DOS:00) */
00211                 {240, 80, 80,SDL_ALPHA_OPAQUE }, /* iCol= 02: rot             */
00212                 { 80,240, 80,SDL_ALPHA_OPAQUE }, /* iCol= 03: gruen           */
00213                 { 80,240,240,SDL_ALPHA_OPAQUE }, /* iCol= 04: blau            */
00214                 { 80, 80,240,SDL_ALPHA_OPAQUE }, /* iCol= 05: lila            */
00215                 {240,240, 80,SDL_ALPHA_OPAQUE }, /* iCol= 06: gelb            */
00216                 {160,160,160,SDL_ALPHA_OPAQUE }, /* iCol= 07: grau            */
00217                 {240, 80,240,SDL_ALPHA_OPAQUE }, /* iCol= 08: violett         */
00218                 {160,  0,  0,SDL_ALPHA_OPAQUE }, /* iCol= 09: mattrot         */
00219                 {  0,160,  0,SDL_ALPHA_OPAQUE }, /* iCol= 10: mattgruen       */
00220                 {  0,  0,160,SDL_ALPHA_OPAQUE }, /* iCol= 11: mattblau        */
00221                 {  0,160,160,SDL_ALPHA_OPAQUE }, /* iCol= 12: mattlila        */
00222                 {160, 80,  0,SDL_ALPHA_OPAQUE }, /* iCol= 13: orange          */
00223                 { 80, 80, 80,SDL_ALPHA_OPAQUE }, /* iCol= 14: mattgrau        */
00224                 {160,  0,160,SDL_ALPHA_OPAQUE }  /* iCol= 15: mattviolett     */
00225               };
00226 #define MAX_COLOR_INDEX 15
00227
00228
00229 static  SDL_Window *TCSwindow = NULL;
00230 static  SDL_Renderer *TCSrenderer = NULL;
00231 static  TTF_Font* TCSfont = NULL;
00232 static  TTF_Font* TCSstatusfont = NULL;
00233
00234 static  SDL_Window *TCSstatwindow = NULL;
00235 static  SDL_Renderer *TCSstatrenderer = NULL;
00236
00237 struct xJournalEntry_typ {struct xJournalEntry_typ * previous;
00238                           struct xJournalEntry_typ * next;
00239                           FTNINT action; FTNINT i1; FTNINT i2;};
00240  static struct xJournalEntry_typ* xTCSJournal = NULL;
00241
00242 #ifdef AUDIOSUPPORT
00243  static SDL_AudioSpec      SDL_AudioDev_optained;
00244  static SDL_AudioSpec      SDL_AudioDev_wanted;
00245
00246  static int                AudioSample_nr = 0;
00247 #endif
00248
00249
00250
00251
00252
00253 // ---------------- interne Unterprogramme -----------------------------------
00254
00255
00256 /* --- Anpassung der Zeichenaufloesung an die Bildschirme --- */
00257
00258 int HiResX(FTNINT iX)
00259 {
00260     return (PixFacX*iX) +0.25f;
00261 }
00262
00263
00264 int HiResY(FTNINT iY)
00265 {
00266     return (PixFacY*iY)+0.25f;
00267 }
00268
00269
00270 int LoResX(FTNINT iX)
00271 {
00272     return (int)( ( (float)iX/PixFacX) +0.25f );
00273 }
00274
00275
00276 int LoResY(FTNINT iY)
00277 {
00278     return (int)( ((float)iY/PixFacY)+0.25f );
00279 }
00280
00281
00282
00283 /* --- Clippingroutinen --- */
00284
00285 bool PointInWindow (FTNINT ix1, FTNINT iy1)
00286 {
00287     if (ClippingNotActive ) return true;
00288     return ( (TKTRNX.kminsx <= ix1) && (TKTRNX.kmaxsx >= ix1) &&
00289                     (TKTRNX.kminsy <= iy1) && (TKTRNX.kmaxsy >= iy1));
00290 }
00291
00292
00293 bool ClipLineStart (FTNINT ix1, FTNINT iy1, FTNINT ix2, FTNINT iy2,
00294                                                 FTNINT *isx, FTNINT *isy)
```

```
00295 /* ClipLineStart=true: isx,isy Startpunkt; =false: Linie nicht zeichnen */
00296 {
00297     if (ClippingNotActive) {
00298      *isx= ix1; *isy= iy1;
00299      return true;
00300     }
00301
00302     if (ix1 < TKTRNX.kminsx) { /* Start links vom Fenster */
00303      if (ix2 < TKTRNX.kminsx) return false;
00304      *isy= iy1+((TKTRNX.kminsx-ix1) * (iy2-iy1)) / (ix2-ix1);
00305      if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
00306       *isx= TKTRNX.kminsx;
00307       return true;
00308      }
00309      if (iy1 == iy2) return false;
00310      if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
00311       *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00312       *isy= TKTRNX.kminsy;
00313      } else {
00314       *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00315       *isy= TKTRNX.kmaxsy;
00316      }
00317      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00318      return true;
00319
00320     } else if (ix1 > TKTRNX.kmaxsx) { /* Start rechts vom Fenster */
00321      if (ix2 > TKTRNX.kmaxsx) return false;
00322      *isy= iy1+((TKTRNX.kmaxsx-ix1) * (iy2-iy1)) / (ix2-ix1);
00323      if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
00324       *isx= TKTRNX.kmaxsx;
00325       return true;
00326      }
00327      if (iy1 == iy2) return false;
00328      if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
00329       *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00330       *isy= TKTRNX.kmaxsy;
00331      } else {
00332       *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00333       *isy= TKTRNX.kminsy;
00334      }
00335      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00336      return true;
00337
00338     } else if (iy1 < TKTRNX.kminsy) { /* Start unter dem Fenster */
00339      if (iy2 < TKTRNX.kminsy) return false;
00340      *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00341      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00342      *isy= TKTRNX.kminsy;
00343      return true;
00344
00345     } else if (iy1 > TKTRNX.kmaxsy) { /* Start ueber dem Fenster */
00346      if (iy2 > TKTRNX.kmaxsy) return false;
00347      *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00348      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00349      *isy= TKTRNX.kmaxsy;
00350      return true;
00351
00352     }
00353     *isx= ix1;                          /* Startpunkt liegt im Fenster */
00354     *isy= iy1;
00355     return true;
00356 }
00357
00358 /* Zeichnen einer gestrichelten Linie in den Backbuffer */
00359
00360 void DrawHiResDashLine (FTNINT ix,FTNINT iy, FTNINT ix2,FTNINT iy2,FTNINT *iMask)
00361 {
00362 FTNINT ixx,iyy, ixx2,iyy2;
00363 float xx,yy, dx,dy, dLin,dBlank;
00364
00365     if (*iMask <= 0) {
00366      dLin= 10., dBlank=0.; // solid
00367     } else if (*iMask == 1) {
00368      dLin= 1.; dBlank=1.; // dotted
00369     } else if (*iMask == 2) {
00370      dLin= 3.; dBlank=1.; //  substitute dashed-dotted
00371     } else if (*iMask == 3) {
00372      dLin= 3.; dBlank=3.; //  dashed
00373     } else {
00374      dLin= 3., dBlank=3.; // unrecognized -> dashed
00375     }
00376
00377     if (abs(ix2-ix) >= abs(iy2-iy)) {
00378      dx= ix2 >= ix ?  3. : -3.;
00379      dy= ((float)(iy2-iy))/((float)(ix2-ix))*dx;
00380
00381      xx= (float)ix; yy= (float)iy;
```

```
00382       while (dx != 0.) {
00383        ixx= (FTNINT) xx; iyy= (FTNINT) yy;
00384        ixx2=(FTNINT) (xx+dLin*dx); iyy2=(FTNINT) (yy+dLin*dy);
00385        xx+= (dLin+dBlank)*dx; yy+= (dLin+dBlank)*dy;
00386        if (   ((dx>=0.) && ((FTNINT)xx>=ix2) )
00387            || ((dx<=0.) && ((FTNINT)xx<=ix2) )   ) {
00388         ixx2= ix2; iyy2= iy2;
00389         dx= 0.;
00390        }
00391        SDL_RenderDrawLine(TCSrenderer, HiResX(ixx),HiResY(TEK_YMAX-iyy),
00392                                        HiResX(ixx2),HiResY(TEK_YMAX-iyy2));
00393       }
00394
00395      } else {
00396       dy= iy2 >= iy ?  3. : -3.;
00397       dx= ((float)(ix2-ix))/((float)(iy2-iy))*dy;
00398
00399       xx= (float)ix; yy= (float)iy;
00400       while (dy != 0.) {
00401        ixx= (FTNINT) xx; iyy= (FTNINT) yy;
00402        ixx2=(FTNINT) (xx+dLin*dx); iyy2=(FTNINT) (yy+dLin*dy);
00403        xx+= (dLin+dBlank)*dx; yy+= (dLin+dBlank)*dy;
00404        if (   ((dy>=0.) && ((FTNINT)yy>=iy2) )
00405            || ((dy<=0.) && ((FTNINT)yy<=iy2) )   ) {
00406         ixx2= ix2; iyy2= iy2;
00407         dy= 0.;
00408        }
00409        SDL_RenderDrawLine(TCSrenderer, HiResX(ixx), HiResY(TEK_YMAX-iyy),
00410                                        HiResX(ixx2), HiResY(TEK_YMAX-iyy2));
00411       }
00412      }
00413 }
00414
00415
00416
00417 void PlotText (const char *outtxt)
00418 {
00419 SDL_Rect dstrect;
00420 SDL_Surface* surface;
00421 SDL_Texture* texture;
00422
00423 #ifdef HIGHQUALCHAR
00424     surface = TTF_RenderUTF8_Blended(TCSfont, outtxt, sdlColorTable[TKTRNX.iTxtCol]);
00425 #else
00426     surface = TTF_RenderUTF8_Solid(TCSfont, outtxt, sdlColorTable[TKTRNX.iTxtCol]);
00427 #endif
00428     texture = SDL_CreateTextureFromSurface(TCSrenderer, surface);
00429
00430     SDL_QueryTexture(texture, NULL, NULL, &dstrect.w, &dstrect.h);
00431     dstrect.x= HiResX(TKTRNX.kBeamX);
00432     dstrect.y= HiResY(TEK_YMAX-TKTRNX.kBeamY)-dstrect.h;
00433
00434     SDL_RenderCopy(TCSrenderer, texture, NULL, &dstrect);
00435
00436     SDL_DestroyTexture(texture);
00437     SDL_FreeSurface(surface);
00438
00439     TKTRNX.kBeamX= TKTRNX.kBeamX + LoResX(dstrect.w);
00440 }
00441
00442
00443
00444 void RepaintBuffer () // Hier nicht GraphicError verwenden(Rekursionsschleifen)!
00445 {
00446 FTNINT DashStyle;
00447 int wx, wz, iStringLen, iStringActual;
00448 char szString [TCS_MESSAGELEN+1];
00449 struct xJournalEntry_typ *xJournalEntry;
00450
00451 #ifdef TRACE_CALLS
00452     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> called");
00453 #endif
00454
00455     DashStyle= 0; // Vorbesetzung nur notwendig bei fehlerhaftem Journal
00456     iStringActual= 0; // Zahler Einlesen String ueber XACTION_ASCII
00457     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
00458                                       , sdlColorTable[TKTRNX.iBckCol].g
00459                                       , sdlColorTable[TKTRNX.iBckCol].b
00460                                       , sdlColorTable[TKTRNX.iBckCol].a);
00461     SDL_RenderClear (TCSrenderer); // Backbuffer nach RenderPresent undefiniert
00462
00463  #ifdef TRACE_CALLS
00464     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p", xTCSJournal);
00465  #endif
00466     SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, xTCSJournal, previous, next, xJournalEntry)
00467     while (xJournalEntry != NULL) {
00468  #ifdef TRACE_CALLS
```

```
00469     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p", xTCSJournal);
00470     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> Current Entry: Ptr= %p / previous: Ptr=
      %p / next: Ptr= %p",
00471                     xJournalEntry, xJournalEntry->previous, xJournalEntry->next);
00472     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_??? = %i (i1= %i, i2= %i)",
00473                   xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2 );
00474 #endif
00475     switch (xJournalEntry->action) {
00476       case XACTION_INITT: {
00477        TKTRNX.iLinCol= TCSDefaultLinCol;
00478        TKTRNX.iTxtCol= TCSDefaultTxtCol;
00479        TKTRNX.iBckCol= TCSDefaultBckCol;
00480
00481        INITT2(); // Reset TKTRNX (Margin, Scale...)
00482
00483        TKTRNX.ksizef = 0; // Reset FONT
00484        TKTRNX.kitalc = 0;
00485        if (!TCSfont)TTF_CloseFont(TCSfont);
00486        TCSfont = TTF_OpenFont(szTCSGraphicFont,
00487                               HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
00488        if (!TCSfont) {
00489         SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_INITT Error Opening Fontfile");
00490        } else {
00491         TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
00492         if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
00493          SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_INITT Fontsize?");
00494         } else {
00495          TKTRNX.khorsz= LoResX(wx);
00496          TKTRNX.kversz= LoResY(wz);
00497          TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
00498         }
00499        }
00500        TKTRNX.kBeamX= TKTRNX.klmrgn; // HOME
00501        TKTRNX.kBeamY= TKTRNX.khomey;
00502
00503       } // weiter mit Erase
00504       case XACTION_ERASE: {
00505        SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
00506                                          , sdlColorTable[TKTRNX.iBckCol].g
00507                                          , sdlColorTable[TKTRNX.iBckCol].b
00508                                          , sdlColorTable[TKTRNX.iBckCol].a);
00509        SDL_RenderClear (TCSrenderer);
00510        break; // Erase ohne Auswirkungen auf die Cursorposition!
00511       }
00512       case XACTION_MOVABS: {
00513        TKTRNX.kBeamX= xJournalEntry->i1;
00514        TKTRNX.kBeamY= xJournalEntry->i2;
00515        break;
00516       }
00517       case XACTION_DRWABS: {
00518        SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
00519                                          , sdlColorTable[TKTRNX.iLinCol].g
00520                                          , sdlColorTable[TKTRNX.iLinCol].b
00521                                          , sdlColorTable[TKTRNX.iLinCol].a );
00522        SDL_RenderDrawLine(TCSrenderer, HiResX(TKTRNX.kBeamX),
00523                                        HiResY(TEK_YMAX-TKTRNX.kBeamY),
00524                                        HiResX(xJournalEntry->i1),
00525                                        HiResY(TEK_YMAX-xJournalEntry->i2) );
00526        TKTRNX.kBeamX= xJournalEntry->i1;
00527        TKTRNX.kBeamY= xJournalEntry->i2;
00528        break;
00529       }
00530       case XACTION_DSHSTYLE: {
00531        DashStyle= xJournalEntry->i1;
00532        break;
00533       }
00534       case XACTION_DSHABS: {
00535        SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
00536                                          , sdlColorTable[TKTRNX.iLinCol].g
00537                                          , sdlColorTable[TKTRNX.iLinCol].b
00538                                          , sdlColorTable[TKTRNX.iLinCol].a );
00539        DrawHiResDashLine (TKTRNX.kBeamX,TKTRNX.kBeamY,
      xJournalEntry->i1,xJournalEntry->i2,&DashStyle);
00540        TKTRNX.kBeamX= xJournalEntry->i1;
00541        TKTRNX.kBeamY= xJournalEntry->i2;
00542        break;
00543       }
00544       case XACTION_PNTABS: {
00545        SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
00546                                          , sdlColorTable[TKTRNX.iLinCol].g
00547                                          , sdlColorTable[TKTRNX.iLinCol].b
00548                                          , sdlColorTable[TKTRNX.iLinCol].a );
00549        SDL_RenderDrawPoint(TCSrenderer, HiResX(xJournalEntry->i1),
00550                                         HiResY(TEK_YMAX-xJournalEntry->i2) );
00551        TKTRNX.kBeamX= xJournalEntry->i1;
00552        TKTRNX.kBeamY= xJournalEntry->i2;
00553        break;
```

```
00554         }
00555         case XACTION_BCKCOL: {
00556          TKTRNX.iBckCol= xJournalEntry->i1;
00557          break;
00558         }
00559         case XACTION_LINCOL: {
00560          TKTRNX.iLinCol= xJournalEntry->i1;
00561          break;
00562         }
00563        case XACTION_TXTCOL: {
00564          TKTRNX.iTxtCol= xJournalEntry->i1;
00565          break;
00566         }
00567          case XACTION_FONTATTR: {
00568          TKTRNX.kitalc= xJournalEntry->i1;
00569          if (TKTRNX.kitalc > 0) {
00570           TTF_SetFontStyle(TCSfont, TTF_STYLE_ITALIC);
00571          } else {
00572           TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
00573          }
00574
00575          if (TKTRNX.ksizef != xJournalEntry->i2) {
00576           TKTRNX.ksizef= xJournalEntry->i2;
00577           if (!TCSfont) TTF_CloseFont(TCSfont);
00578           TCSfont = TTF_OpenFont(szTCSGraphicFont,
00579                                  HiResY((1+TKTRNX.ksizef)*TCS_REL_CHR_HEIGHT*TEK_YMAX));
00580           if (!TCSfont) {
00581            SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_FONTATTR");
00582           } else {
00583            if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
00584             SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_FONTATTR Size");
00585            } else {
00586             TKTRNX.khorsz= LoResX(wx);
00587             TKTRNX.kversz= LoResY(wz);
00588             TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
00589            }
00590           }
00591          }
00592          break;
00593         }
00594         case XACTION_GTEXT: {
00595          iStringActual= 0;
00596          iStringLen= xJournalEntry->i1;
00597          if (iStringLen > TCS_MESSAGELEN) iStringLen= TCS_MESSAGELEN;
00598          if (iStringLen == 0) break;
00599          szString[iStringActual++]= xJournalEntry->i2;
00600          if (iStringLen == 1) {
00601           szString[iStringActual]= '\0';
00602           PlotText (szString);
00603          }
00604          break;
00605         }
00606         case XACTION_ASCII: {
00607          if (iStringActual < iStringLen) {
00608           szString[iStringActual++]= xJournalEntry->i1;
00609           if (iStringActual < iStringLen) szString[iStringActual++]= xJournalEntry->i2;
00610           if (iStringActual >= iStringLen ) {
00611            szString[iStringActual]= '\0';
00612            PlotText (szString);
00613           }
00614          }
00615          break;
00616         }
00617         case XACTION_NOOP: {
00618          break;
00619         }
00620         default: {
00621          SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_XXX");
00622          break;
00623         }
00624       }
00625       xJournalEntry= xJournalEntry -> previous;
00626      }
00627  #ifdef TRACE_CALLS
00628     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p / Last Entry: Ptr=
       %p", xTCSJournal, xJournalEntry);
00629  #endif
00630 }
00631
00632
00633
00634 void TCSGraphicError (int iErr, const char* msg)
00635 {
00636 char cBuf[TCS_MESSAGELEN];
00637 FTNINT i; // Dummyparameter
00638
00639     snprintf( cBuf, TCS_MESSAGELEN, szTCSErrorMsg[iErr], msg );
```

```
00640     if (!TCSinitialized) { // Vor Systeminitalisierung nur Basismeldungen
00641      SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
00642      SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR,
00643                               szTCSstatWindowName, cBuf, TCSwindow);
00644     } else { // ab jetzt mit bell, outtext...
00645      SDL_RenderPresent (TCSrenderer);
00646      RepaintBuffer ();
00647      if (TCSErrorLev[iErr] > 0) {
00648       bell ();
00649       outtext (cBuf, strlen (cBuf) );
00650       if (TCSErrorLev[iErr] == 2) {
00651        SDL_LogInfo (SDL_LOG_CATEGORY_VIDEO, cBuf);
00652       }
00653       if (TCSErrorLev[iErr] == 3) {
00654        SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
00655       } else if (TCSErrorLev[iErr] < 10) {
00656        SDL_LogWarn (SDL_LOG_CATEGORY_VIDEO, cBuf);
00657        if (TCSErrorLev[iErr] == 5) {
00658         dcursr (&i,&i,&i); // Press Any Key
00659        } else if (TCSErrorLev[iErr]==8) {
00660         SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_INFORMATION,
00661                                  szTCSstatWindowName, cBuf, TCSwindow);
00662        }
00663       } else {
00664        if (TCSErrorLev[iErr] == 10) {
00665         dcursr (&i,&i,&i); // Press Any Key
00666        }
00667        if (TCSErrorLev[iErr] == 12) {
00668         SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR,
00669                                  szTCSstatWindowName, cBuf, TCSwindow);
00670        }
00671        if (iErr != ERR_EXIT) { // Error-Level von finitt durch XML veraenderbar
00672         SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
00673         finitt ();                  // Erzwungenes Beenden durch finitt
00674        }
00675       }
00676      }
00677     }
00678 }
00679
00680
00681
00682
00683
00684 /* Eventhandler zum Fensterhandling */
00685
00686 int TCSEventFilter(void* UserData, SDL_Event* event)
00687 {
00688 SDL_Point winsiz;
00689
00690     if (event->type == SDL_WINDOWEVENT) {
00691      switch (event->window.event) {
00692       case SDL_WINDOWEVENT_RESIZED:
00693       case SDL_WINDOWEVENT_MAXIMIZED:
00694       case SDL_WINDOWEVENT_RESTORED:
00695        if (event->window.windowID == SDL_GetWindowID(TCSwindow)) {
00696         if (SDL_GetRendererOutputSize(TCSrenderer, &winsiz.x, &winsiz.y) != 0) {
00697          TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
00698         } else {
00699          PixFacX= (float)(winsiz.x) / (float) TEK_XMAX;
00700          PixFacY= (float)(winsiz.y) / (float) TEK_YMAX;
00701          SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "WINSIZ> PixFac: x= %f, y= %f", PixFacX, PixFacY);
00702         }
00703        }
00704       case SDL_WINDOWEVENT_EXPOSED:
00705        if (event->window.windowID == SDL_GetWindowID(TCSwindow)) {
00706         SDL_RenderPresent (TCSrenderer);
00707         RepaintBuffer ();
00708        } else { if (event->window.windowID == SDL_GetWindowID(TCSstatwindow)) {
00709         SDL_RenderPresent (TCSstatrenderer);
00710        } }
00711        break;
00712       default:
00713        break;
00714      }
00715     }
00716     return 1;
00717 }
00718
00719
00720
00721 #ifdef AUDIOSUPPORT
00722  void audio_callback(void *sample_nr, Uint8 *raw_buffer, int bytes)
00723  {
00724  int i, length;
00725  float time, value;
00726  Sint16* buffer;
```

```
00727  SDL_AudioCVT cvt;
00728
00729     buffer= (Sint16*) raw_buffer;
00730     length = 8*bytes /SDL_AUDIO_BITSIZE(SDL_AudioDev_optained.format) /
       SDL_AudioDev_optained.channels; // Bytes = Variablenlänge (Bit/8) pro Kanal
00731     for(i=0; i < length; i++, *((int*)sample_nr)=*((int*)sample_nr)+1 ) {
00732      time = ((float)( *((int*)sample_nr)) / SAMPLE_RATE);
00733      value= BELL_AMPLITUDE * sin(2.0f * M_PI * BELL_FREQUENCY * time);
00734      buffer[i] = (Sint16)(value);
00735     }
00736     SDL_BuildAudioCVT(&cvt, AUDIO_S16SYS, 1, SAMPLE_RATE, SDL_AudioDev_optained.format,
       SDL_AudioDev_optained.channels, SDL_AudioDev_optained.freq);
00737     cvt.len = length*2;  // Sint16 = 2 Bytes
00738     cvt.buf = raw_buffer;
00739     SDL_ConvertAudio(&cvt); // Konvertiere in das Deviceformat
00740 #ifdef TRACE_CALLS
00741     SDL_LogVerbose (SDL_LOG_CATEGORY_AUDIO, "audio_callback» Number of Samples= %d Bytes allocated= %d
       ", length,bytes);
00742     SDL_LogVerbose (SDL_LOG_CATEGORY_AUDIO, "audio_callback» Bytes 16bit Audio= %d Bytes needed= %d",
       cvt.len,cvt.len_cvt);
00743 #endif
00744  }
00745 #endif
00746
00747
00748
00749 /* Eventhandler zum Parsen von XML-Dateien */
00750
00751
00752 void sax_callback (mxml_node_t *node, mxml_sax_event_t event, void *usr)
00753 {
00754 char * StorePtr;
00755
00756     switch (event) {
00757      case MXML_SAX_ELEMENT_OPEN: {
00758       switch (*(int*)usr ) {
00759        case -1: { // Statemachine: noch keine aktive Sektion
00760         if (strcmp(mxmlGetElement(node),szTCSsect0) == 0) {
00761          *(int*)usr= 0;  // Parsing active
00762          mxmlElementSetAttr (node,"typ","none");
00763         }
00764         break;
00765        }
00766        case 0: {
00767         if ((strcmp(mxmlGetElement(node),TCS_INISECT1) == 0)  ) {
00768          *(int*)usr= 1; // State: TCS_INISECT1
00769         } else if ((strcmp(mxmlGetElement(node),TCS_INISECT2) == 0)  ) {
00770          *(int*)usr= 2; // State: TCS_INISECT2
00771         } else if ((strcmp(mxmlGetElement(node),TCS_INISECT3) == 0)  ) {
00772          *(int*)usr= 3; // State: TCS_INISECT3
00773         }
00774         mxmlElementSetAttr (node,"typ","none");
00775         break;
00776        }
00777
00778        case 1: { // Section = Names
00779         if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINNAM) == 0)  ) {
00780          mxmlElementSetAttr (node,"typ","opaque");
00781          mxmlElementSetAttrf(node,"store","%p",&szTCSWindowName);
00782         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATNAM) == 0)  ) {
00783          mxmlElementSetAttr (node,"typ","opaque");
00784          mxmlElementSetAttrf(node,"store","%p",&szTCSstatWindowName);
00785         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCNAM) == 0)  ) {
00786          mxmlElementSetAttr (node,"typ","opaque");
00787          mxmlElementSetAttrf(node,"store","%p",&szTCSHardcopyFile);
00788         }
00789         break;
00790        }
00791
00792        case 2: { // Section = Layout
00793         if ((strcmp(mxmlGetElement(node),TCS_INIVAR_FONT) == 0)  ) {
00794          mxmlElementSetAttr (node,"typ","opaque");
00795          mxmlElementSetAttrf(node,"store","%p",&szTCSGraphicFont);
00796         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_SYSFONT) == 0)  ) {
00797          mxmlElementSetAttr (node,"typ","opaque");
00798          mxmlElementSetAttrf(node,"store","%p",&szTCSSysFont);
00799
00800         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSX) == 0)  ) {
00801          mxmlElementSetAttr (node,"typ","integer");
00802          mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelpos);
00803         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSY) == 0)  ) {
00804          mxmlElementSetAttr (node,"typ","integer");
00805          mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelpos);
00806         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZX) == 0)  ) {
00807          mxmlElementSetAttr (node,"typ","integer");
00808          mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelsiz);
00809         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZY) == 0)  ) {
```

```
00810          mxmlElementSetAttr (node,"typ","integer");
00811          mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelsiz);
00812
00813         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSX) == 0)  ) {
00814          mxmlElementSetAttr (node,"typ","integer");
00815          mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelpos);
00816         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSY) == 0)  ) {
00817          mxmlElementSetAttr (node,"typ","integer");
00818          mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelpos);
00819         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZX) == 0)  ) {
00820          mxmlElementSetAttr (node,"typ","integer");
00821          mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelsiz);
00822         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZY) == 0)  ) {
00823          mxmlElementSetAttr (node,"typ","integer");
00824          mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelsiz);
00825
00826         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_LINCOL) == 0)  ) {
00827          mxmlElementSetAttr (node,"typ","integer");
00828          mxmlElementSetAttrf(node,"store","%p",&TCSDefaultLinCol);
00829         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_TXTCOL) == 0)  ) {
00830          mxmlElementSetAttr (node,"typ","integer");
00831          mxmlElementSetAttrf(node,"store","%p",&TCSDefaultTxtCol);
00832         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_BCKCOL) == 0)  ) {
00833          mxmlElementSetAttr (node,"typ","integer");
00834          mxmlElementSetAttrf(node,"store","%p",&TCSDefaultBckCol);
00835         }
00836         break;
00837        }
00838
00839        case 3: { // Section = Messages
00840         if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNGRAPHCARD) == 0)  ) {
00841          mxmlElementSetAttr (node,"typ","opaque");
00842          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_UNKNGRAPHCARD]);
00843         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNGRAPHCARDL) == 0)  ) {
00844          mxmlElementSetAttr (node,"typ","integer");
00845          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_UNKNGRAPHCARD]);
00846
00847         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_NOFNTFIL) == 0)  ) {
00848          mxmlElementSetAttr (node,"typ","opaque");
00849          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_NOFNTFIL]);
00850         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_NOFNTFILL) == 0)  ) {
00851          mxmlElementSetAttr (node,"typ","integer");
00852          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_NOFNTFIL]);
00853
00854         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPN) == 0)  ) {
00855          mxmlElementSetAttr (node,"typ","opaque");
00856          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILOPN]);
00857         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPNL) == 0)  ) {
00858          mxmlElementSetAttr (node,"typ","integer");
00859          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILOPN]);
00860
00861         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRT) == 0)  ) {
00862          mxmlElementSetAttr (node,"typ","opaque");
00863          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILWRT]);
00864         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRTL) == 0)  ) {
00865          mxmlElementSetAttr (node,"typ","integer");
00866          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILWRT]);
00867
00868         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINT) == 0)  ) {
00869          mxmlElementSetAttr (node,"typ","opaque");
00870          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCINTERN]);
00871         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINTL) == 0)  ) {
00872          mxmlElementSetAttr (node,"typ","integer");
00873          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCINTERN]);
00874
00875         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR) == 0)  ) {
00876          mxmlElementSetAttr (node,"typ","opaque");
00877          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR]);
00878         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRL) == 0)  ) {
00879          mxmlElementSetAttr (node,"typ","integer");
00880          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR]);
00881
00882         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACT) == 0)  ) {
00883          mxmlElementSetAttr (node,"typ","opaque");
00884          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_HDCACT]);
00885         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACTL) == 0)  ) {
00886          mxmlElementSetAttr (node,"typ","integer");
00887          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_HDCACT]);
00888
00889         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRN) == 0)  ) {
00890          mxmlElementSetAttr (node,"typ","opaque");
00891          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_USRPRESSANY]);
00892         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRNL) == 0)  ) {
00893          mxmlElementSetAttr (node,"typ","integer");
00894          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_USRPRESSANY]);
00895
00896         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXIT) == 0)  ) {
```

```
00897          mxmlElementSetAttr (node,"typ","opaque");
00898          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_EXIT]);
00899         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXITL) == 0)  ) {
00900          mxmlElementSetAttr (node,"typ","integer");
00901          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_EXIT]);
00902
00903         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATE) == 0)  ) {
00904          mxmlElementSetAttr (node,"typ","opaque");
00905          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCREATE]);
00906         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATEL) == 0)  ) {
00907          mxmlElementSetAttr (node,"typ","integer");
00908          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCREATE]);
00909
00910         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRY) == 0)  ) {
00911          mxmlElementSetAttr (node,"typ","opaque");
00912          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUENTRY]);
00913         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRYL) == 0)  ) {
00914          mxmlElementSetAttr (node,"typ","integer");
00915          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUENTRY]);
00916
00917         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADD) == 0)  ) {
00918          mxmlElementSetAttr (node,"typ","opaque");
00919          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUADD]);
00920         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADDL) == 0)  ) {
00921          mxmlElementSetAttr (node,"typ","integer");
00922          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUADD]);
00923
00924         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLR) == 0)  ) {
00925          mxmlElementSetAttr (node,"typ","opaque");
00926          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCLR]);
00927         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLRL) == 0)  ) {
00928          mxmlElementSetAttr (node,"typ","integer");
00929          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCLR]);
00930
00931         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWN) == 0)  ) {
00932          mxmlElementSetAttr (node,"typ","opaque");
00933          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUUNKWN]);
00934         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWNL) == 0)  ) {
00935          mxmlElementSetAttr (node,"typ","integer");
00936          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUUNKWN]);
00937
00938         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSER) == 0)  ) {
00939          mxmlElementSetAttr (node,"typ","opaque");
00940          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLPARSER]);
00941         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSERL) == 0)  ) {
00942          mxmlElementSetAttr (node,"typ","integer");
00943          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLPARSER]);
00944
00945         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPEN) == 0)  ) {
00946          mxmlElementSetAttr (node,"typ","opaque");
00947          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLOPEN]);
00948         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPENL) == 0)  ) {
00949          mxmlElementSetAttr (node,"typ","integer");
00950          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLOPEN]);
00951
00952         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNAUDIO) == 0)  ) {
00953          mxmlElementSetAttr (node,"typ","opaque");
00954          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_UNKNAUDIO]);
00955         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNAUDIOL) == 0)  ) {
00956          mxmlElementSetAttr (node,"typ","integer");
00957          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_UNKNAUDIO]);
00958
00959         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2) == 0)  ) {
00960          mxmlElementSetAttr (node,"typ","opaque");
00961          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR2]);
00962         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2L) == 0)  ) {
00963          mxmlElementSetAttr (node,"typ","integer");
00964          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR2]);
00965
00966         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2) == 0)  ) {
00967          mxmlElementSetAttr (node,"typ","opaque");
00968          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_INI2]);
00969         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2L) == 0)  ) {
00970          mxmlElementSetAttr (node,"typ","integer");
00971          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_INI2]);
00972
00973         }
00974          break;
00975        }
00976
00977       }
00978       break;
00979      }
00980
00981      case MXML_SAX_DATA: {
00982       switch (mxmlGetType(node)) {
00983        case MXML_INTEGER: {
```

```
00984          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
00985          (*(int*)StorePtr)= mxmlGetInteger(node);
00986          break;
00987         }
00988         case MXML_REAL: {
00989          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
00990          (*(float*)StorePtr)= mxmlGetReal(node);
00991          break;
00992         }
00993         case MXML_TEXT: {
00994          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
00995          strcpy (StorePtr, mxmlGetText(node, NULL));
00996          break;
00997         }
00998         case MXML_OPAQUE: {
00999          sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01000          strcpy (StorePtr, mxmlGetOpaque(node));
01001          break;
01002         }
01003        }
01004        break;
01005       }
01006
01007       case MXML_SAX_ELEMENT_CLOSE: {
01008        if ((*(int*)usr==0) && (strcmp(mxmlGetElement(node),szTCSsect0)==0)) {
01009         *(int*)usr= -1; // State: idle
01010        } else if (
01011               ((*(int*)usr==1) && (strcmp(mxmlGetElement(node),TCS_INISECT1)==0))
01012            || ((*(int*)usr==2) && (strcmp(mxmlGetElement(node),TCS_INISECT2)==0))
01013            || ((*(int*)usr==3) && (strcmp(mxmlGetElement(node),TCS_INISECT3)==0))
01014            ) {
01015         *(int*)usr= 0; // State: Parsing active
01016        }
01017        break;
01018       }
01019      }
01020 }
01021
01022
01023 /* ------------------------------------------------------------------------ */
01024
01025
01026 mxml_type_t  sax_type_callback(mxml_node_t  *node)
01027 {
01028 const char *type;
01029
01030     if ((type = mxmlElementGetAttr(node, "typ")) == NULL) type = "none";
01031     if (!strcmp(type, "integer"))
01032      return (MXML_INTEGER);
01033     else if (!strcmp(type, "opaque") || !strcmp(type, "pre"))
01034      return (MXML_OPAQUE);
01035     else if (!strcmp(type, "real"))
01036      return (MXML_REAL);
01037     else if (!strcmp(type, "text"))
01038      return (MXML_TEXT);
01039     else
01040      return (MXML_IGNORE);
01041 }
01042
01043 /* ------------------------------------------------------------------------ */
01044
01045
01046 void sax_error_callback (char *mssg)
01047 {
01048     TCSGraphicError (ERR_XMLPARSER, mssg);
01049     return;
01050 }
01051
01052
01053
01054 /*
01055 ------------------- Userroutinen: Initialisierung ------------------------------
01056 */
01057
01058
01059 void XMLreadProgPar (const char * filname)
01060 {
01061 int ParserState;
01062 FILE *fp;
01063 mxml_node_t *tree;
01064
01065     if (filname[0] != '\0') {
01066       fp = fopen(filname, "r");
01067       if (fp == NULL) {
01068        TCSGraphicError (ERR_XMLOPEN, filname);
01069       } else {
01070         ParserState= -1; // State= idle
```

```
01071         mxmlSetErrorCallback ((mxml_error_cb_t)sax_error_callback);
01072         tree = mxmlSAXLoadFile(NULL, fp, sax_type_callback, sax_callback, &ParserState);
01073         fclose(fp);
01074       }
01075     }
01076 }
01077
01078
01079 /*
01080 Setzen der Defaultwerte vor dem Einlesen der Initialisierungsdaten
01081 */
01082
01083 void PresetProgPar ()
01084 {
01085     TCSDefaultLinCol= TCS_INIDEF_LINCOL;
01086     TCSDefaultTxtCol= TCS_INIDEF_TXTCOL;
01087     TCSDefaultBckCol= TCS_INIDEF_BCKCOL;
01088
01089     TCSwindowIniXrelpos= TCS_INIDEF_WINPOSX;
01090     TCSwindowIniYrelpos= TCS_INIDEF_WINPOSY;
01091     TCSwindowIniXrelsiz= TCS_INIDEF_WINSIZX;
01092     TCSwindowIniYrelsiz= TCS_INIDEF_WINSIZY;
01093
01094     TCSstatWindowIniXrelpos= TCS_INIDEF_STATPOSX;
01095     TCSstatWindowIniYrelpos= TCS_INIDEF_STATPOSY;
01096     TCSstatWindowIniXrelsiz= TCS_INIDEF_STATSIZX;
01097     TCSstatWindowIniYrelsiz= TCS_INIDEF_STATSIZY;
01098
01099     // Fensternamen werden nur durch winlbl vorher veraendert
01100
01101     // Hardcopyname und Zaehlerstand bleibt!
01102
01103     // Fehlermeldungen werden bei der Variablendefinition durch den Compiler initialisiert
01104 }
01105
01106
01107 /*
01108 Anpassung der Dateinamen an die Laufzeitumgebung
01109 */
01110
01111 void CustomizeProgPar ()
01112 {
01113 char        szTmpString[TCS_FILE_NAMELEN], szTmpString1[TCS_FILE_NAMELEN];
01114 FTNSTRDESC  ftn_WorkString, o, n;
01115
01116     ftn_WorkString.len= TCS_FILE_NAMELEN; // Ersatz %: durch Programmverzeichnis
01117     ftn_WorkString.addr= szTCSGraphicFont;
01118     n.addr= SDL_GetBasePath(); // Neuer Substring = Directory
01119     n.len= strlen(n.addr);
01120     o.addr= PROGDIRTOKEN; // Alter Substring
01121     o.len= strlen (o.addr);
01122     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01123                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01124                 CALLFTNSTRL(ftn_WorkString)
01125                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01126     strncpy(szTCSGraphicFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01127
01128     ftn_WorkString.addr= szTCSSysFont;
01129     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01130                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01131                 CALLFTNSTRL(ftn_WorkString)
01132                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01133     strncpy(szTCSSysFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01134
01135     SDL_free (n.addr); // SDL_BasePath nicht mehr benoetigt
01136
01137     n.addr= FNTFILEXT; // "Ersatz .% durch .TTF oder kein Punkt durch .TTF
01138     n.len= strlen(n.addr);
01139     o.addr= INIFILEXTTOKEN; // Alter Substring
01140     o.len= strlen (o.addr);
01141     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01142                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01143                 CALLFTNSTRL(ftn_WorkString)
01144                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01145     strncpy(szTCSSysFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01146     if (strchr(szTCSSysFont,'.') == 0) {
01147         strncat (szTCSSysFont, n.addr, TCS_FILE_NAMELEN-n.len);
01148     }
01149
01150     ftn_WorkString.addr= szTCSGraphicFont;
01151     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01152                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01153                 CALLFTNSTRL(ftn_WorkString)
01154                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01155     strncpy(szTCSGraphicFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01156     if (strchr(szTCSGraphicFont,'.') == 0) {
01157         strncat (szTCSGraphicFont, n.addr, TCS_FILE_NAMELEN-n.len);
```

```
01158     }
01159 }
01160
01161
01162 extern void winlbl (FTNSTRPAR * PloWinNam, FTNSTRPAR * StatWinNam,
01163                                            FTNSTRPAR *IniFilNam
01164                                            FTNSTRPAR_TAIL(PloWinNam)
01165                                            FTNSTRPAR_TAIL(StatWinNam)
01166                                            FTNSTRPAR_TAIL(IniFilNam)      )
01167
01168 {
01169 // Absicherung der Definition der Programmparameter
01170 #if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN)
01171  #define TMPSTRLEN TCS_FILE_NAMELEN
01172 #else
01173  #define TMPSTRLEN TCS_WINDOW_NAMELEN
01174 #endif
01175
01176 int         i;
01177 FTNINT      iL;
01178 char        szTmpString[TMPSTRLEN], szTmpString1[TCS_FILE_NAMELEN];
01179 char *      iAt;
01180 FTNSTRDESC  ftn_WorkString, o, n;
01181
01182     iL= FTNSTRPARL(PloWinNam);                      // Name des Grahikfensters
01183     if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
01184     strncpy(szTmpString, FTNSTRPARA(PloWinNam),iL);
01185     szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
01186     iL= strlen (szTmpString);
01187     if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
01188     if (iL > 0) {
01189      strncpy( szTCSWindowName, szTmpString, iL);
01190      szTCSWindowName[iL]= '\0';
01191     }
01192
01193     iL= FTNSTRPARL(StatWinNam);                     // Name des Statusfensters
01194     if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
01195     strncpy(szTmpString, FTNSTRPARA(StatWinNam), iL);
01196     szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
01197     iL= strlen (szTmpString);
01198     if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
01199     if (iL > 0) {
01200      strncpy( szTCSstatWindowName, szTmpString, iL);
01201      szTCSstatWindowName[iL]= '\0';
01202     }
01203
01204     iL= FTNSTRPARL(IniFilNam);              // Name der Initialisierungsdatei
01205     if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
01206     strncpy(szTmpString, FTNSTRPARA(IniFilNam), iL);
01207     szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
01208
01209     iL= strlen(szTmpString);
01210     if (iL > (TCS_FILE_NAMELEN-1)) iL= TCS_FILE_NAMELEN-1;
01211     if (iL > 0) {
01212      strncpy( szTCSIniFile, szTmpString, iL);
01213      szTCSIniFile[iL]= '\0';
01214
01215      iAt= strstr (szTCSIniFile, "@"); // Section Level0?
01216      if (iAt != 0) {
01217       strncpy (szTCSsect0, &iAt[1], iL);
01218       iAt[0]= '\0'; // Abschneiden von @Section0 in szTCSIniFile
01219      }
01220
01221      ftn_WorkString.len= TCS_FILE_NAMELEN;
01222      ftn_WorkString.addr= szTCSIniFile;
01223
01224      n.addr= SDL_GetBasePath(); // Neuer Substring = Directory
01225      n.len= strlen(n.addr);
01226      o.addr= PROGDIRTOKEN; // Alter Substring
01227      o.len= strlen (o.addr);
01228      SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01229                  CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01230                  CALLFTNSTRL(ftn_WorkString)
01231                  CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01232      SDL_free (n.addr);
01233
01234      n.addr= INIFILEXT; // Neuer Substring = Default Extension
01235      n.len= strlen (INIFILEXT);
01236      o.addr= INIFILEXTTOKEN; // Alter Substring
01237      o.len= strlen (o.addr);
01238      SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01239                  CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01240                  CALLFTNSTRL(ftn_WorkString)
01241                  CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01242      strncpy(szTCSIniFile, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01243     }
01244
```

```
01245 #ifdef TRACE_CALLS
01246     SDL_LogSetAllPriority(LOGLEVEL); // Ausgabe in Fehlerkanal vor INIT moeglich
01247     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM,
01248             "WINLBL> Setting Windowname >%s< Statusname >%s< Inifile >%s<\n\r",
01249                           szTCSWindowName, szTCSstatWindowName, szTCSIniFile);
01250 #endif
01251
01252 // Absicherung TMPSTRLEN nicht mehr benoetigt
01253 #undef TMPSTRLEN
01254 }
01255
01256
01257
01258 extern void initt1 ()
01259 {
01260 int iD;
01261 Uint32 flags;
01262 SDL_Point winsiz;
01263 SDL_Rect rect;
01264
01265 struct xJournalEntry_typ * xJournalEntry;
01266
01267
01268     if (TCSinitialized) return;   /* Bereits initialisiert */
01269
01270     SDL_LogSetAllPriority(LOGLEVEL); // Ausgabe in Fehlerkanal bereits moeglich
01271
01272     PresetProgPar(); // Compilerinitialisierung nach finitt() wiederherstellen
01273
01274     /*
01275         Falls Extension des Ini-Files .XML: XML-Parser -> hier immer XML
01276     */
01277 #if defined(XMLSUPPORT)
01278     XMLreadProgPar (szTCSIniFile);
01279 #endif
01280
01281     CustomizeProgPar (); // Ersatz %: durch Programmverzeichnis
01282
01283      /*
01284      Übernahme der durch den Nutzer angepassten Initialisierungsdaten
01285      */
01286
01287     TKTRNX.iLinCol= TCSDefaultLinCol;
01288     TKTRNX.iTxtCol= TCSDefaultTxtCol;
01289     TKTRNX.iBckCol= TCSDefaultBckCol;
01290
01291     /*
01292         Initialisierung des SDL2-Systems
01293     */
01294
01295     if (SDL_Init(SDL_INIT_VIDEO) != 0) {
01296      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01297     }
01298     if (TTF_Init() != 0) {
01299      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01300     }
01301 #ifdef AUDIOSUPPORT
01302     if (SDL_InitSubSystem(SDL_INIT_AUDIO) != 0) {
01303      TCSGraphicError (ERR_UNKNAUDIO, SDL_GetError());
01304     }
01305 #endif
01306
01307     /*
01308         Ermittlung allgemeiner systemspezifischer Parameter
01309     */
01310
01311     iD= SDL_GetNumVideoDisplays();
01312     if (iD <= 0) {
01313      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01314     } else {
01315      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> SDL_GetNumVideoDisplays = %i", iD);
01316     }
01317
01318     iD= iD-1;
01319     if (SDL_GetDisplayUsableBounds(iD, &rect) != 0) {
01320      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01321     } else {
01322      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> UsableDisplayBounds: x= %i, y= %i, w= %i, h= %i",
       rect.x,rect.y,rect.w,rect.h);
01323     }
01324
01325     SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "linear");
01326     SDL_SetEventFilter(TCSEventFilter,&TCSEventFilterData);
01327
01328     /*
01329         Erzeugung des Graphikfensters
01330     */
```

```
01331
01332     flags= SDL_WINDOW_RESIZABLE;
01333     if (szTCSWindowName[0] == '~') {
01334      flags= flags | SDL_WINDOW_BORDERLESS;
01335     }
01336     TCSwindow = SDL_CreateWindow(szTCSWindowName,
01337                                  TCSwindowIniXrelpos *rect.w / 100,
01338                                  TCSwindowIniYrelpos *rect.h / 100,
01339                                  TCSwindowIniXrelsiz *rect.w / 100,
01340                                  TCSwindowIniYrelsiz *rect.h / 100,
01341                                  flags );
01342     TCSrenderer = SDL_CreateRenderer(TCSwindow, -1, 0);
01343
01344
01345
01346     if (SDL_GetRendererOutputSize(TCSrenderer, &winsiz.x, &winsiz.y) != 0) {
01347      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01348     } else {
01349      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> RendererBounds: x= %i, y= %i", winsiz.x,winsiz.y);
01350      PixFacX= (float)(winsiz.x) / (float) TEK_XMAX;
01351      PixFacY= (float)(winsiz.y) / (float) TEK_YMAX;
01352      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> PixFac: x= %f, y= %f", PixFacX, PixFacY);
01353     }
01354
01355     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
01356                                       , sdlColorTable[TKTRNX.iBckCol].g
01357                                       , sdlColorTable[TKTRNX.iBckCol].b
01358                                       , sdlColorTable[TKTRNX.iBckCol].a );
01359     SDL_RenderClear (TCSrenderer);
01360     SDL_RenderPresent (TCSrenderer);
01361
01362     TCSfont = TTF_OpenFont(szTCSGraphicFont,
01363                  HiResY(TCS_REL_CHR_HEIGHT*TEK_YMAX));
01364     if (!TCSfont) {
01365      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01366     }         // TKTRNX wird durch INITT gesetzt
01367
01368     /*
01369         Erzeugung des Statusfensters
01370     */
01371
01372     if (TCSstatWindowIniYrelsiz > 0 ) {
01373      flags= SDL_WINDOW_RESIZABLE;
01374      if (szTCSstatWindowName[0] == '~') {
01375       flags= flags | SDL_WINDOW_BORDERLESS;
01376      }
01377      TCSstatwindow = SDL_CreateWindow(szTCSstatWindowName,
01378                                  TCSstatWindowIniXrelpos *rect.w / 100,
01379                                  TCSstatWindowIniYrelpos *rect.h / 100,
01380                                  TCSstatWindowIniXrelsiz *rect.w / 100,
01381                                  TCSstatWindowIniYrelsiz *rect.h / 100,
01382                                  flags);
01383
01384      TCSstatrenderer = SDL_CreateRenderer(TCSstatwindow, -1, 0);
01385
01386      SDL_SetRenderDrawColor(TCSstatrenderer, sdlColorTable[TCSDefaultBckCol].r
01387                                       , sdlColorTable[TCSDefaultBckCol].g
01388                                       , sdlColorTable[TCSDefaultBckCol].b
01389                                       , sdlColorTable[TCSDefaultBckCol].a );
01390      SDL_RenderClear (TCSstatrenderer);
01391      SDL_RenderPresent (TCSstatrenderer);
01392
01393      TextLineHeight= HiResY(TCS_REL_CHR_HEIGHT*TEK_YMAX);
01394      TCSstatusfont = TTF_OpenFont(szTCSSysFont, TextLineHeight);
01395      if (!TCSstatusfont) {
01396       TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01397      }
01398      TKTRNX.kStCol= 1; // Nur einzeilige Ausgabe
01399     }
01400
01401     /*
01402         Initialisierung des Audiosystems
01403     */
01404
01405 #ifdef AUDIOSUPPORT
01406
01407     SDL_AudioDev_wanted.freq = SAMPLE_RATE;
01408     SDL_AudioDev_wanted.format = AUDIO_S16SYS; // 16 bit integer
01409     SDL_AudioDev_wanted.channels = 1; // Mono
01410     SDL_AudioDev_wanted.samples = 2048; // buffer-size
01411     SDL_AudioDev_wanted.callback = audio_callback;
01412     SDL_AudioDev_wanted.userdata = &AudioSample_nr; // Zaehler zur Sinusberechnung
01413
01414     if(SDL_OpenAudio(&SDL_AudioDev_wanted, &SDL_AudioDev_optained) < 0) {
01415      TCSGraphicError (ERR_UNKNAUDIO, SDL_GetError());
01416     } else {
01417      if(SDL_AudioDev_wanted.format != SDL_AudioDev_optained.format) {
```

```
01418       SDL_LogInfo(SDL_LOG_CATEGORY_AUDIO, "INITT1> Failed to get the desired AudioSpec");
01419      }
01420     }
01421     SDL_LogDebug (SDL_LOG_CATEGORY_AUDIO, "INITT1> want.frequ= %i  want.channels= %i  want.samples= %i
      want.size= %i",
01422                   SDL_AudioDev_wanted.freq, SDL_AudioDev_wanted.channels, SDL_AudioDev_wanted.samples,
      SDL_AudioDev_wanted.size);
01423     SDL_LogDebug (SDL_LOG_CATEGORY_AUDIO, "INITT1> optained.frequ= %i  optained.channels= %i
      optained.samples= %i  optained.size= %i",
01424                   SDL_AudioDev_optained.freq, SDL_AudioDev_optained.channels,
      SDL_AudioDev_optained.samples, SDL_AudioDev_optained.size);
01425 #endif
01426
01427     /*
01428         Anlegen des Journals
01429     */
01430
01431     xTCSJournal= NULL;
01432     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> xTCSJournal initialisiert: Ptr= %p", xTCSJournal);
01433
01434     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01435     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");
01436     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> Nach 1. malloc: xJournalEntry: Ptr= %p",
      xJournalEntry);
01437
01438     xJournalEntry->action=  XACTION_NOOP; // Erkennung Listenanfang: Wurzelelement ohne Funktion
01439     xJournalEntry->i1= 0;
01440     xJournalEntry->i2= 0;
01441     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01442     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> LIST_ADD=Create Journal: xTCSJournal: Ptr= %p /
      xJournalEntry: Ptr= %p", xTCSJournal, xJournalEntry);
01443     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "INITT1> previous: Ptr= %p / next: Ptr= %p", xJournalEntry
      -> previous, xJournalEntry -> next);
01444
01445     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01446     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
01447     xJournalEntry->action=  XACTION_INITT;
01448     xJournalEntry->i1= 0;
01449     xJournalEntry->i2= 0;
01450     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01451     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> Nach 2. LIST_ADD: xTCSJournal: Ptr= %p /
      xJournalEntry: Ptr= %p", xTCSJournal, xJournalEntry);
01452     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "INITT1> previous: Ptr= %p / next: Ptr= %p", xJournalEntry
      -> previous, xJournalEntry -> next);
01453
01454     /*
01455         Initialisierung erfolgreich abgeschlossen
01456     */
01457
01458     TCSinitialized= true;
01459
01460     return;
01461 }
01462
01463
01464
01465 extern void finitt ()
01466 {
01467 struct xJournalEntry_typ    * xJournalEntry;
01468
01469     if (!TCSinitialized) return; /* Graphiksystem nicht initialisiert */
01470
01471     TCSGraphicError (ERR_EXIT,"");
01472     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "finitt> Quit SDL");
01473
01474     TCSinitialized= false;       /* Ab jetzt nicht mehr funktionsfähig */
01475
01476     SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, xTCSJournal,
01477            xJournalEntry,previous,next, { free (xJournalEntry);}); // free all
01478     xTCSJournal= NULL;
01479
01480     TTF_CloseFont(TCSfont);
01481     TTF_CloseFont(TCSstatusfont);
01482
01483     SDL_DestroyRenderer(TCSrenderer);
01484     SDL_DestroyWindow(TCSwindow);
01485
01486     if (TCSstatWindowIniYrelsiz > 0 ) {
01487      SDL_DestroyRenderer(TCSstatrenderer);
01488      SDL_DestroyWindow(TCSstatwindow);
01489     }
01490
01491 #ifdef AUDIOSUPPORT
01492     SDL_CloseAudio();
01493 #endif
01494
01495     TTF_Quit();
```

```
01496     SDL_Quit();
01497
01498     if (TCSErrorLev[ERR_EXIT] >= 10) exit (EXIT_SUCCESS);
01499     return;
01500 }
01501
01502
01503
01504 extern void iowait (void)
01505 {
01506     SDL_RenderPresent (TCSrenderer);
01507     RepaintBuffer ();
01508 }
01509
01510
01511
01512 /*
01513 ------------------- Userroutinen: Zeichnen -------------------------------------
01514 */
01515
01516
01517
01518 extern void swind1 (FTNINT *ix1,FTNINT *iy1,FTNINT *ix2,FTNINT *iy2)
01519 {
01520     ClippingNotActive = (*ix1==0) && (*iy1==0) &&
01521                                         (*ix2==TEK_XMAX) && (*iy2==TEK_YMAX);
01522     /* Berechnung BOOL zur Wahrung der Programmstruktur der DOS-Version */
01523 }
01524
01525
01526
01527 extern void erase (void)
01528 {
01529 struct xJournalEntry_typ    * xJournalEntry;
01530
01531     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
01532                                       , sdlColorTable[TKTRNX.iBckCol].g
01533                                       , sdlColorTable[TKTRNX.iBckCol].b
01534                                       , sdlColorTable[TKTRNX.iBckCol].a );
01535     SDL_RenderClear (TCSrenderer);
01536     SDL_RenderPresent (TCSrenderer);
01537
01538      SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, xTCSJournal,
01539            xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
01540
01541      xTCSJournal= NULL; // create new journal
01542      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01543      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCLR,"");
01544      xJournalEntry->action=  XACTION_NOOP; // Wurzelelement ohne Vorgaenger
01545      xJournalEntry->i1= 0;
01546      xJournalEntry->i2= 0;
01547      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01548
01549      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01550      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01551      xJournalEntry->action=  XACTION_LINCOL;
01552      xJournalEntry->i1= TKTRNX.iLinCol;
01553      xJournalEntry->i2= 0;
01554      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01555
01556      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01557      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01558      xJournalEntry->action=  XACTION_TXTCOL;
01559      xJournalEntry->i1= TKTRNX.iTxtCol;
01560      xJournalEntry->i2= 0;
01561      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01562
01563      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01564      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01565      xJournalEntry->action=  XACTION_BCKCOL;
01566      xJournalEntry->i1= TKTRNX.iBckCol;
01567      xJournalEntry->i2= 0;
01568      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01569
01570      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ)); // New
      Plot
01571      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
01572      xJournalEntry->action=  XACTION_ERASE;
01573      xJournalEntry->i1= 0;
01574      xJournalEntry->i2= 0;
01575      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01576 }
01577
01578
01579
01580 extern void movabs (FTNINT *ix,FTNINT *iy)
01581 {
```

```
01582 struct xJournalEntry_typ    * xJournalEntry;
01583
01584     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01585     if (PointInWindow (*ix, *iy)) {
01586      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01587      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01588      xJournalEntry->action=  XACTION_MOVABS;
01589      xJournalEntry->i1= *ix;
01590      xJournalEntry->i2= *iy;
01591      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01592     }
01593 }
01594
01595
01596
01597 extern void drwabs (FTNINT *ix,FTNINT *iy)
01598 {
01599 FTNINT iXClip, iYClip, iXClip2, iYClip2;
01600 struct xJournalEntry_typ    * xJournalEntry;
01601
01602     if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
01603      ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip2,&iYClip2); // geclippter Endpunkt
01604      SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
01605                                        , sdlColorTable[TKTRNX.iLinCol].g
01606                                        , sdlColorTable[TKTRNX.iLinCol].b
01607                                        , sdlColorTable[TKTRNX.iLinCol].a );
01608      SDL_RenderDrawLine(TCSrenderer, HiResX(iXClip),HiResY(TEK_YMAX-iYClip),
01609                                    HiResX(iXClip2),HiResY(TEK_YMAX-iYClip2));
01610
01611      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01612      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01613      xJournalEntry->action=  XACTION_MOVABS;
01614      xJournalEntry->i1= iXClip;
01615      xJournalEntry->i2= iYClip;
01616      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01617
01618      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01619      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01620      xJournalEntry->action=  XACTION_DRWABS;
01621      xJournalEntry->i1= iXClip2;
01622      xJournalEntry->i2= iYClip2;
01623      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01624     }
01625     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01626     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01627     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01628     xJournalEntry->action=  XACTION_MOVABS;
01629     xJournalEntry->i1= *ix;
01630     xJournalEntry->i2= *iy;
01631     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01632 }
01633
01634
01635
01636 extern void dshabs (FTNINT *ix,FTNINT *iy, FTNINT *iMask)
01637 {
01638 FTNINT iXClip,iYClip, iXClip2, iYClip2;
01639 FTNINT ixx,iyy, ixx2,iyy2;
01640 float xx,yy, dx,dy, dLin,dBlank;
01641 struct xJournalEntry_typ    * xJournalEntry;
01642
01643     if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
01644      ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip2,&iYClip2); // Clip Endpunkt
01645      SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
01646                                        , sdlColorTable[TKTRNX.iLinCol].g
01647                                        , sdlColorTable[TKTRNX.iLinCol].b
01648                                        , sdlColorTable[TKTRNX.iLinCol].a );
01649      DrawHiResDashLine (iXClip,iYClip, iXClip2,iYClip2,iMask);
01650
01651      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01652      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01653      xJournalEntry->action=  XACTION_MOVABS;
01654      xJournalEntry->i1= iXClip;
01655      xJournalEntry->i2= iYClip;
01656      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01657
01658      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01659      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01660      xJournalEntry->action=  XACTION_DSHSTYLE;
01661      xJournalEntry->i1= *iMask;
01662      xJournalEntry->i2= 0;
01663      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01664
01665      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01666      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01667      xJournalEntry->action=  XACTION_DSHABS;
01668      xJournalEntry->i1= iXClip2;
```

```
01669      xJournalEntry->i2= iYClip2;
01670      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01671     }
01672     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01673     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01674     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01675     xJournalEntry->action=  XACTION_MOVABS;
01676     xJournalEntry->i1= *ix;
01677     xJournalEntry->i2= *iy;
01678     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01679 }
01680
01681
01682
01683 extern void pntabs (FTNINT *ix,FTNINT *iy)
01684 {
01685 struct xJournalEntry_typ    * xJournalEntry;
01686 FTNINT ActPntMov;
01687
01688     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01689     if (PointInWindow (*ix, *iy)) {
01690      SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
01691                                        , sdlColorTable[TKTRNX.iLinCol].g
01692                                        , sdlColorTable[TKTRNX.iLinCol].b
01693                                        , sdlColorTable[TKTRNX.iLinCol].a );
01694      SDL_RenderDrawPoint(TCSrenderer, HiResX(*ix),HiResX(TEK_YMAX-*iy));
01695      ActPntMov= XACTION_PNTABS;
01696     } else {
01697      ActPntMov= XACTION_MOVABS;
01698     }
01699     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01700     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01701     xJournalEntry->action= ActPntMov;
01702     xJournalEntry->i1= *ix;
01703     xJournalEntry->i2= *iy;
01704     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01705 }
01706
01707
01708
01709 extern void bckcol (FTNINT *iCol)
01710 {
01711 struct xJournalEntry_typ    * xJournalEntry;
01712
01713     TKTRNX.iBckCol= *iCol;
01714     if (*iCol > MAX_COLOR_INDEX) TKTRNX.iBckCol= MAX_COLOR_INDEX;
01715
01716     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01717     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01718     xJournalEntry->action=  XACTION_BCKCOL;
01719     xJournalEntry->i1= TKTRNX.iBckCol;
01720     xJournalEntry->i2= 0;
01721     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01722 }
01723
01724
01725
01726 extern void lincol (FTNINT *iCol)
01727 {
01728 struct xJournalEntry_typ    * xJournalEntry;
01729
01730     TKTRNX.iLinCol= *iCol;
01731     if (*iCol > MAX_COLOR_INDEX) TKTRNX.iLinCol= MAX_COLOR_INDEX;
01732
01733     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01734     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01735     xJournalEntry->action=  XACTION_LINCOL;
01736     xJournalEntry->i1= TKTRNX.iLinCol;
01737     xJournalEntry->i2= 0;
01738     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01739 }
01740
01741
01742
01743
01744 extern void txtcol (FTNINT *iCol)
01745 {
01746 struct xJournalEntry_typ    * xJournalEntry;
01747
01748     TKTRNX.iTxtCol= *iCol;
01749     if (*iCol > MAX_COLOR_INDEX) TKTRNX.iTxtCol= MAX_COLOR_INDEX;
01750
01751     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01752     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01753     xJournalEntry->action=  XACTION_TXTCOL;
01754     xJournalEntry->i1= TKTRNX.iTxtCol;
01755     xJournalEntry->i2= 0;
```

```
01756     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01757 }
01758
01759
01760
01761 extern void DefaultColour (void)
01762 {
01763     TKTRNX.iLinCol= TCSDefaultLinCol;
01764     TKTRNX.iTxtCol= TCSDefaultTxtCol;
01765     TKTRNX.iBckCol= TCSDefaultBckCol;
01766
01767     lincol (&TKTRNX.iLinCol);
01768     txtcol (&TKTRNX.iTxtCol);
01769     bckcol (&TKTRNX.iBckCol);
01770 }
01771
01772
01773
01774 /*
01775 ------------------- Userroutinen: Graphiktext ----------------------------------
01776 */
01777
01778
01779
01780 extern void outgtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
01781 {
01782 int i, iL;
01783 char outbuf [TCS_MESSAGELEN+1];
01784 struct xJournalEntry_typ    * xJournalEntry;
01785
01786     if (FTNSTRPARA(ftn_string)[0] == '\0' ) return; // Leerstring char(0)
01787
01788     iL= 0; // Bei Bedarf String mit char(0) abschliessen -> Kopie in outbuf
01789     while ( (FTNSTRPARA(ftn_string)[iL] != '\0')  &&     // c-String bis \0
01790                  (iL < FTNSTRPARL(ftn_string)) &&     // String= Fortran Konstante
01791                  (iL < TCS_MESSAGELEN-1)       ) {    // Buffer Overflow
01792      outbuf[iL]= FTNSTRPARA(ftn_string)[iL];
01793      iL++;
01794     }
01795     outbuf[iL]= '\0'; //
01796
01797     PlotText (outbuf);
01798
01799      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01800      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01801      xJournalEntry->action=  XACTION_GTEXT;
01802      xJournalEntry->i1= (FTNINT) iL;
01803      xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[0];
01804      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01805
01806      i= 1;
01807      while (i < iL) {
01808       xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01809       if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01810       xJournalEntry->action=  XACTION_ASCII;
01811       xJournalEntry->i1= (FTNINT) FTNSTRPARA(ftn_string)[i++];
01812       if ( i<iL ) {
01813        xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[i++];
01814       } else {
01815        xJournalEntry->i2= (FTNINT) 0;
01816       }
01817       SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01818      }
01819
01820      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01821      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01822      xJournalEntry->action=  XACTION_MOVABS;
01823      xJournalEntry->i1= TKTRNX.kBeamX;
01824      xJournalEntry->i2= TKTRNX.kBeamY;
01825      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01826
01827 }
01828
01829
01830
01831 extern void italic (void)
01832 {
01833 struct xJournalEntry_typ    * xJournalEntry;
01834
01835     TKTRNX.kitalc = 1;
01836     TTF_SetFontStyle(TCSfont, TTF_STYLE_ITALIC);
01837
01838     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01839     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01840     xJournalEntry->action= XACTION_FONTATTR;
01841     xJournalEntry->i1= TKTRNX.kitalc;
01842     xJournalEntry->i2= TKTRNX.ksizef;
```

```
01843     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01844 }
01845
01846
01847
01848 extern void italir (void)
01849 {
01850 struct xJournalEntry_typ    * xJournalEntry;
01851
01852     TKTRNX.kitalc = 0;
01853     TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
01854
01855     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01856     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01857     xJournalEntry->action= XACTION_FONTATTR;
01858     xJournalEntry->i1= TKTRNX.kitalc;
01859     xJournalEntry->i2= TKTRNX.ksizef;
01860     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01861 }
01862
01863
01864
01865 extern void dblsiz (void)
01866 {
01867 int wx,wz;
01868 struct xJournalEntry_typ    * xJournalEntry;
01869
01870     TKTRNX.ksizef = 1;
01871
01872     if (!TCSfont)TTF_CloseFont(TCSfont);
01873     TCSfont = TTF_OpenFont(szTCSGraphicFont, 2*HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
01874     if (!TCSfont) {
01875      TCSGraphicError (ERR_NOFNT,TTF_GetError() );
01876     } else {
01877      if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
01878       TCSGraphicError (ERR_NOFNT,TTF_GetError() );
01879      } else {
01880       TKTRNX.khorsz= LoResX(wx);
01881       TKTRNX.kversz= LoResY(wz);
01882       TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
01883      }
01884     }
01885
01886     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01887     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01888     xJournalEntry->action= XACTION_FONTATTR;
01889     xJournalEntry->i1= TKTRNX.kitalc;
01890     xJournalEntry->i2= TKTRNX.ksizef;
01891     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01892 }
01893
01894
01895
01896 extern void nrmsiz (void)
01897 {
01898 int wx, wz;
01899 struct xJournalEntry_typ    * xJournalEntry;
01900
01901     TKTRNX.ksizef = 0;
01902
01903     if (!TCSfont)TTF_CloseFont(TCSfont);
01904     TCSfont = TTF_OpenFont(szTCSGraphicFont, HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
01905     if (!TCSfont) {
01906      TCSGraphicError (ERR_NOFNT,TTF_GetError() );
01907     } else {
01908      if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
01909       TCSGraphicError (ERR_NOFNT,TTF_GetError() );
01910      } else {
01911       TKTRNX.khorsz= LoResX(wx);
01912       TKTRNX.kversz= LoResY(wz);
01913       TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
01914      }
01915     }
01916
01917     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01918     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01919     xJournalEntry->action= XACTION_FONTATTR;
01920     xJournalEntry->i1= TKTRNX.kitalc;
01921     xJournalEntry->i2= TKTRNX.ksizef;
01922     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01923 }
01924
01925
01926
01927
01928
01929
```

```
01930 extern void csize (FTNINT *ix,FTNINT *iy)
01931 {
01932      *ix=  TKTRNX.khorsz;
01933      *iy=  TKTRNX.kversz;
01934 }
01935
01936
01937
01938 extern void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
01939 {
01940 int iL;
01941 char outbuf [TCS_MESSAGELEN+1];
01942 SDL_Rect dstrect;
01943 SDL_Surface* surface;
01944 SDL_Texture* texture;
01945
01946     if ( (FTNSTRPARA(ftn_string)[0] == '\0' ) // Leerstring char(0)
01947        || (TCSstatWindowIniYrelsiz <= 0 ) ) { // kein Statusfenster
01948      return;
01949     }
01950     SDL_RenderPresent (TCSrenderer);
01951     RepaintBuffer ();
01952
01953     iL= 0; // Bei Bedarf String mit char(0) abschliessen -> Kopie in outbuf
01954     while ( (FTNSTRPARA(ftn_string)[iL] != '\0')  &&     // c-String bis \0
01955                     (iL < FTNSTRPARL(ftn_string)) &&     // String= Fortran Konstante
01956                     (iL < TCS_MESSAGELEN-1)       ) {    // Buffer Overflow
01957      outbuf[iL]= FTNSTRPARA(ftn_string)[iL];
01958      iL++;
01959     }
01960     outbuf[iL]= '\0'; //
01961
01962     SDL_SetRenderDrawColor(TCSstatrenderer, sdlColorTable[TCSDefaultBckCol].r
01963                                       , sdlColorTable[TCSDefaultBckCol].g
01964                                       , sdlColorTable[TCSDefaultBckCol].b
01965                                       , sdlColorTable[TCSDefaultBckCol].a );
01966     SDL_RenderClear (TCSstatrenderer);
01967
01968 #ifdef HIGHQUALCHAR
01969     surface = TTF_RenderUTF8_Blended (TCSstatusfont, outbuf, sdlColorTable[TCSDefaultLinCol]);
01970 #else
01971     surface = TTF_RenderUTF8_Solid (TCSstatusfont, outbuf, sdlColorTable[TCSDefaultLinCol]);
01972 #endif
01973
01974     texture = SDL_CreateTextureFromSurface(TCSstatrenderer, surface);
01975
01976     dstrect.x= 0;
01977     dstrect.y= 0;
01978     SDL_QueryTexture(texture, NULL, NULL, &dstrect.w, &dstrect.h);
01979     SDL_RenderCopy(TCSstatrenderer, texture, NULL, &dstrect);
01980
01981     SDL_RenderPresent(TCSstatrenderer);
01982     SDL_DestroyTexture(texture);
01983     SDL_FreeSurface(surface);
01984 }
01985
01986
01987
01988 extern void bell (void)
01989 {
01990 #ifdef AUDIOSUPPORT
01991     AudioSample_nr= 0;
01992     SDL_PauseAudio(0); // start playing sound
01993     SDL_Delay(BELL_DURATION); // wait while sound is playing
01994     SDL_PauseAudio(1); // stop playing sound
01995 #endif
01996     return;
01997 }
01998
01999
02000 extern void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
02001                                       FTNINT *iL  FTNSTRPAR_TAIL(ftn_string))
02002 {
02003     TCSGraphicError (*iErr, FTNSTRPARA(ftn_string));
02004
02005 }
02006
02007
02008
02009 /*
02010 ------------------- Userroutinen: Graphic Input---------------------------------
02011 */
02012
02013
02014
02015 extern void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy)
02016 {
```

```
02017 SDL_Event  event;
02018
02019     if (!TCSinitialized) return;            /* Aufhängen vermeiden */
02020
02021     SDL_RenderPresent (TCSrenderer);
02022     RepaintBuffer ();
02023     SDL_RaiseWindow(TCSwindow); // Set input focus
02024
02025     *ic= 0;
02026     while (*ic == 0) {
02027      SDL_WaitEvent (&event);
02028      switch (event.type) {
02029       case SDL_KEYDOWN:
02030        if (event.key.keysym.sym < 256) {
02031         *ic= (FTNINT) event.key.keysym.sym;
02032        }
02033        break;
02034       case SDL_MOUSEBUTTONDOWN:
02035        if (ix == iy) break; // Aufruf TINPUT, nicht  DCURSR
02036        switch (event.button.button) { // Tastaturcode analog DOS
02037         case SDL_BUTTON_LEFT: *ic= 1; break;
02038         case SDL_BUTTON_RIGHT: *ic= 2; break;
02039         case SDL_BUTTON_MIDDLE: *ic= 4; break;
02040        }
02041        *ix= (FTNINT) (LoResX(event.button.x));
02042        *iy= (FTNINT) (TEK_YMAX-LoResY(event.button.y));
02043        break;
02044       default:
02045        TCSEventFilter(NULL, &event); // Weiterleitung Standardhandler, ic = Dummy
02046        break;
02047      }
02048     }
02049 }
02050
02051
02052
02053 /*
02054 ------------------- Userroutinen: Hardcopy -----------------------------------
02055 */
02056
02057
02058
02059 extern void hdcopy (void)
02060 {
02061
02062 FTNINT      iErr;
02063 FTNSTRDESC  ftnstrg;
02064 char        szTmpString[TCS_FILE_NAMELEN];
02065 SDL_RWops*  hFile;
02066 struct xJournalEntry_typ *xJournalEntry;
02067
02068     snprintf( szTmpString,TCS_FILE_NAMELEN, szTCSHardcopyFile, iHardcopyCount++ );
02069     hFile = SDL_RWFromFile( szTmpString, "r" );
02070     while ((iHardcopyCount < MAX_HDCCOUNT) && (hFile != NULL) ) {
02071      SDL_RWclose (hFile);
02072      snprintf( szTmpString,TCS_FILE_NAMELEN, szTCSHardcopyFile, iHardcopyCount++ );
02073      hFile = SDL_RWFromFile( szTmpString, "r" );
02074     }
02075     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> iHardcopyCount Next= %i", iHardcopyCount);
02076     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Filnam= %s", szTmpString);
02077     if (hFile != NULL) { // iHardcopyCount zu klein
02078      SDL_RWclose (hFile);
02079      SDL_LogError (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Open HDC_File: kein freier Filename");
02080      return;
02081     }
02082
02083     hFile = SDL_RWFromFile( szTmpString, "wb" );
02084     if (hFile == NULL) {
02085      SDL_LogError (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Error openening %s",szTmpString);
02086      return;
02087     }
02088
02089     TCSGraphicError (MSG_HDCACT, szTmpString);
02090
02091     SGLIB_DL_LIST_GET_LAST (struct xJournalEntry_typ, xTCSJournal, previous, next, xJournalEntry)
02092 #ifdef TRACE_CALLS
02093     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xTCSJournal: Ptr= %p", xTCSJournal);
02094     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> 1. Entry: Ptr= %p / previous: Ptr= %p / next:
       Ptr= %p", xJournalEntry, xJournalEntry -> previous, xJournalEntry -> next);
02095 #endif
02096     while (xJournalEntry != NULL) {
02097      snprintf( szTmpString,TCS_FILE_NAMELEN, "%02i#%04i-%03i\n", xJournalEntry->action,
       xJournalEntry->i1, xJournalEntry->i2 );
02098      SDL_RWwrite(hFile, szTmpString, 1, strlen(szTmpString));
02099 #ifdef TRACE_CALLS
02100      switch (xJournalEntry->action) {
02101        case XACTION_INITT: {
```

```
02102         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_INITT");
02103         break;
02104        }
02105        case XACTION_ERASE: {
02106         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_ERASE");
02107         break;
02108        }
02109        case XACTION_MOVABS: {
02110         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_MOVABS: x= %i, y= %i",
      xJournalEntry->i1, xJournalEntry->i2);
02111         break;
02112        }
02113        case XACTION_DRWABS: {
02114         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DRWABS: x= %i, y= %i",
      xJournalEntry->i1, xJournalEntry->i2);
02115         break;
02116        }
02117        case XACTION_DSHSTYLE: {
02118         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DSHSTYLE: x= %i", xJournalEntry->i1);
02119         break;
02120        }
02121        case XACTION_DSHABS: {
02122         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DSHABS: x= %i, y= %i",
      xJournalEntry->i1, xJournalEntry->i2);
02123         break;
02124        }
02125        case XACTION_PNTABS: {
02126         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_PNTABS: x= %i, y= %i",
      xJournalEntry->i1, xJournalEntry->i2);
02127         break;
02128        }
02129        case XACTION_BCKCOL: {
02130         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_BCKCOL: x= %i", xJournalEntry->i1);
02131         break;
02132        }
02133        case XACTION_TXTCOL: {
02134         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_TXTCOL: x= %i", xJournalEntry->i1);
02135         break;
02136        }
02137        case XACTION_LINCOL: {
02138         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_LINCOL: x= %i", xJournalEntry->i1);
02139         break;
02140        }
02141        case XACTION_FONTATTR: {
02142         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_FONTATTR: x= %i, y= %i",
      xJournalEntry->i1, xJournalEntry->i2);
02143         break;
02144        }
02145        case XACTION_GTEXT: {
02146         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_GTEXT: Len= %i, Char[%i]= %c",
02147                       xJournalEntry->i1, xJournalEntry->i2, xJournalEntry->i2);
02148         break;
02149        }
02150        case XACTION_ASCII: {
02151         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_ASCII: Char1[%i]= %c, Char2[%i]= %c",
02152                       xJournalEntry->i1, xJournalEntry->i1, xJournalEntry->i2, xJournalEntry->i2);
02153         break;
02154        }
02155        case XACTION_NOOP: {
02156         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_NOOP");
02157         break;
02158        }
02159        default: {
02160         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_XXX");
02161         break;
02162        }
02163      }
02164      SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xJournalEntry: Ptr= %i / previous: Ptr= %i /
      next: Ptr= %i", xJournalEntry, xJournalEntry -> previous, xJournalEntry -> next);
02165 #endif // TRACE_CALLS
02166      xJournalEntry= xJournalEntry -> previous;
02167     }
02168
02169    SDL_RWclose (hFile);
02170 #ifdef TRACE_CALLS
02171    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xTCSJournal New Current Entry: Ptr= %p",
      xJournalEntry);
02172    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> Previous: Ptr= %p  Next: Ptr= %p",
      xJournalEntry->previous, xJournalEntry->next);
02173 #endif // TRACE_CALLS
02174
02175 }
02176
02177
02178
02179 /*
02180 ------------------- subroutine LIB_MOVC3 fuer Watcom- und GNU-Compiler ---------
```

```
02181 Hier nicht benoetigt, nur wg. Kompatibilitaet zur DOS-Version enthalten
02182 */
02183
02184
02185 extern void lib_movc3 (FTNINT *len,FTNSTRPAR *sou,FTNSTRPAR *dst
02186                                 FTNSTRPAR_TAIL(sou)  FTNSTRPAR_TAIL(dst) )
02187
02188 {
02189 int n;
02190     if (FTNSTRPARA(dst) <= FTNSTRPARA(sou) ) {
02191      for (n=0; n<*len; n++) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
02192     } else {
02193      for (n= (*len)-1; n>=0; n--) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
02194     };
02195 }
```

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

```
00001 /** ****************************************************************************
00002 \file    TCSdSDLc.h
00003 \brief   SDL Port: Low-Level Driver
00004 \version 1.2
00005 \author  (C) 2023 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          Headerfile zu TCSdSDLc.c
00009 \~english
00010          Headerfile for TCSdSDL.c
00011 \~
00012
00013 ***************************************************************************** */
00014
00015
00016
```

```
00017 /* ---- Zeichenbereich im Tektronix-Koordinatensystem --------------------- */
00018
00019 #define TEK_XMAX 1023
00020 #define TEK_YMAX 780
00021
00022
00023 /* ------------ Compilerspezifische Definitionen -------------------------- */
00024
00025 #ifdef _UNICODE
00026  #error "GNU f77 basiert nicht auf UNICODE !!!"
00027 #endif
00028
00029
00030 /* Deklaration analog C++ */
00031
00032  typedef int bool;
00033  #define false 0
00034  #define true !false
00035
00036
00037 /* Deklaration Parameteruebergabe Fortran <-> C */
00038
00039 typedef long int logical; // 3 plattformabhaengige Definitionen
00040 typedef long int integer; // evtl. ueberpruefen
00041
00042 typedef logical LOGICAL;
00043 typedef integer FTNINT;
00044 typedef float FTNREAL;
00045 typedef double FTNDOUBLE;
00046 typedef struct {float real, imag;} FTNCOMPLEX;
00047
00048 typedef char FTNCHAR;
00049
00050 typedef size_t ftnlen;  // Ersatz fuer g2c.h
00051 typedef size_t FTNCHARLEN;
00052
00053 typedef struct { FTNCHAR * addr; FTNCHARLEN len; } FTNSTRDESC;
00054 typedef FTNCHAR FTNSTRPAR;
00055 #define FTNSTRPAR_TAIL(ftns) , FTNCHARLEN ftns##_len
00056 #define FTNSTRPARA(ftns) ftns
00057 #define FTNSTRPARL(ftns) ftns##_len
00058 #define CALLFTNSTRA(ftns) ftns.addr
00059 #define CALLFTNSTRL(ftns) , ftns.len
00060 #define FWRDFTNSTRA(ftns) ftns
00061 #define FWRDFTNSTRL(ftns) , ftns##_len
00062
00063 #define TKTRNX tktrnx_ /* Fortran Naming Convention */
00064 #define tcslev3 tcslev3_
00065 #define initt1 initt1_
00066 #define finitt finitt_
00067 #define iowait iowait_
00068 #define GraphicError graphicerror_
00069 #define winlbl winlbl_
00070 #define erase erase_
00071 #define swind1 swind1_
00072 #define movabs movabs_
00073 #define drwabs drwabs_
00074 #define dshabs dshabs_
00075 #define pntabs pntabs_
00076 #define bckcol bckcol_
00077 #define lincol lincol_
00078 #define txtcol txtcol_
00079 #define DefaultColour defaultcolour_
00080 #define outgtext outgtext_
00081 #define italic italic_
00082 #define italir italir_
00083 #define dblsiz dblsiz_
00084 #define nrmsiz nrmsiz_
00085 #define bell bell_
00086 #define outtext outtext_
00087 #define tinput tinput_
00088 #define dcursr dcursr_
00089 #define csize csize_
00090 #define hdcopy hdcopy_
00091 #define lib_movc3 lib_movc3_
00092
00093 /* Deklarationen von durch C aufgerufenen FTN77-Unterprogrammen */
00094
00095 #define GETARG getarg_       // aus GNU F77-Library
00096 FTNINT GETARG (FTNINT *iNo, FTNCHAR *line, FTNCHARLEN line_len);
00097
00098 #define INITT2 initt2_
00099 void INITT2 (void);
00100
00101 #define SUBSTITUTE substitute_
00102 void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *new
00103                                          FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
```

```
00104                                       FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(new));
00105
00106 /* Forward Deklarationen: Codiert in C und auch in C verwendet */
00107
00108 void bell (void); //  -> Forward Deklaration
00109 void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
00110                    FTNINT *iL  FTNSTRPAR_TAIL(ftn_string));
00111 void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) );
00112 void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy);
00113 void finitt ();
00114
00115
00116
00117 /* ------------ Programmparameter ---------------------------------------- */
00118
00119
00120 #define STAT_MAXROWS 1             /* vorhandene Statuszeilen */
00121
00122 #define TCS_REL_CHR_HEIGHT 0.023f
00123
00124 #define TCS_WINDOW_NAMELEN 50
00125 #define TCS_FILE_NAMELEN 128
00126 #define TCS_MESSAGELEN 132
00127
00128 #define MAX_HDCCOUNT 1000          /* s.u.: Format TCS_HDCFILE_NAME */
00129
00130 #define INIFILEXTTOKEN ".%"        /* Token fuer den Filenamenparser */
00131 #define PROGDIRTOKEN "%:"
00132
00133 #define TCS_INIFILE_NAME "Graph2D"
00134
00135 #define SAMPLE_RATE 41000 // fuer SDL-Audioausgabe
00136 #define BELL_AMPLITUDE 32000.0
00137 #define BELL_FREQUENCY 441.0f
00138 #define BELL_DURATION 200
00139
00140
00141 /* Actioncodes des Journalfiles */
00142
00143 #define XACTION_INITT       1
00144 #define XACTION_ERASE       2
00145 #define XACTION_MOVABS      3
00146 #define XACTION_DRWABS      4
00147 #define XACTION_DSHSTYLE    5
00148 #define XACTION_DSHABS      6
00149 #define XACTION_PNTABS      7
00150 #define XACTION_GTEXT       8
00151 #define XACTION_ASCII       9
00152 #define XACTION_BCKCOL      10
00153 #define XACTION_LINCOL      11
00154 #define XACTION_TXTCOL      12
00155 #define XACTION_FONTATTR    13
00156 #define XACTION_NOOP        14
00157
00158
00159
00160 /* Zuordnung Fehlernummern zu Meldungen */
00161
00162 #define WRN_NOMSG 1
00163 #define ERR_UNKNGRAPHCARD 2
00164 #define ERR_NOFNTFIL 3
00165 #define ERR_NOFNT 4
00166 #define MSG_NOMOUSE 5
00167 #define WRN_HDCFILOPN 6
00168 #define WRN_HDCFILWRT 7
00169 #define WRN_HDCINTERN 8
00170 #define MSG_USR 9
00171 #define MSG_HDCACT 10
00172 #define WRN_USRPRESSANY 11
00173 #define ERR_EXIT 12
00174 #define WRN_COPYNOMEM 13
00175 #define WRN_COPYLOCK 14
00176 #define WRN_JOUCREATE 15
00177 #define WRN_JOUENTRY 16
00178 #define WRN_JOUADD 17
00179 #define WRN_JOUCLR 18
00180 #define WRN_JOUUNKWN 19
00181 #define ERR_XMLPARSER 20
00182 #define ERR_XMLOPEN 21
00183 #define ERR_UNKNAUDIO 22
00184 #define MSG_USR2 23
00185 #define WRN_INI2 24
00186 #define MSG_MAXERRNO 25
00187
00188
00189
00190 /* Initialisierungskonstanten *.INI, werden sinngemaess auch bei der
```

```
00191     Registry und XML-Initialisierung verwendet.
00192     Bei Erweiterungen Variableninitialisierung szTCSErrorMsg und TCSErrorLev
00193     in TCSdWINc.c fuer Registry und XML-Initialisierung nicht vergessen und
00194     alle Parser (*.ini, Registry und *.xml) beruecksichtigen! */
00195
00196 #define TCS_INISECT0 "Graph2D" // Root-Section, derzeit nur bei XML verwendet
00197
00198 #define TCS_INISECT1 "Names"
00199  #define TCS_INIVAR_WINNAM "G2dGraphic"
00200     #define TCS_WINDOW_NAME "Graphics"
00201  #define TCS_INIVAR_STATNAM "G2dStatus"
00202     #define TCS_STATWINDOW_NAME "System Messages"
00203  #define TCS_INIVAR_HDCNAM "G2dHardcopy"
00204     #if (JOURNALTYP ==1)
00205        #define TCS_HDCFILE_NAME "HDC%03i.WMF"
00206     #elif (JOURNALTYP ==2)
00207        #define TCS_HDCFILE_NAME "HDC%03i.EMF"
00208     #elif (JOURNALTYP ==3)
00209        #define TCS_HDCFILE_NAME "HDC%03i.HDC"
00210     #else
00211        #define TCS_HDCFILE_NAME "HDC%03i.UNKNOWN"
00212     #endif
00213
00214 #define TCS_INISECT2 "Layout"
00215  #define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
00216     #define TCS_INIDEF_COPMEN "Copy"
00217  #define TCS_INIVAR_FONT "G2dGraphicFont"
00218     #define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
00219  #define TCS_INIVAR_SYSFONT "G2dSystemFont"
00220     #define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
00221  #define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
00222     #define TCS_INIDEF_WINPOSX 1
00223  #define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
00224     #define TCS_INIDEF_WINPOSY 3
00225  #define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
00226     #define TCS_INIDEF_WINSIZX 98
00227  #define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
00228     #define TCS_INIDEF_WINSIZY 85
00229  #define TCS_INIVAR_STATPOSX "G2dStatusPosX"
00230     #define TCS_INIDEF_STATPOSX 1
00231  #define TCS_INIVAR_STATPOSY "G2dStatusPosY"
00232     #define TCS_INIDEF_STATPOSY 91
00233  #define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
00234     #define TCS_INIDEF_STATSIZX 98
00235  #define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
00236     #define TCS_INIDEF_STATSIZY 3   // mit X11 o.k.
00237 //    #define TCS_INIDEF_STATSIZY 0 // sonst nur 1 Fenster
00238  #define TCS_INIVAR_LINCOL "G2dLinCol"
00239     #define TCS_INIDEF_LINCOL 1
00240  #define TCS_INIVAR_TXTCOL "G2dTxtCol"
00241     #define TCS_INIDEF_TXTCOL 1
00242  #define TCS_INIVAR_BCKCOL "G2dBckCol"
00243     #define TCS_INIDEF_BCKCOL 0
00244
00245 #define TCS_INISECT3 "Messages"
00246  #define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
00247     #define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System: Error %s."
00248     #define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
00249     #define TCS_INIDEF_UNKNGRAPHCARDL 10
00250  #define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
00251     #define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
00252     #define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
00253     #define TCS_INIDEF_NOFNTFILL 10
00254  #define TCS_INIVAR_NOFNT "G2dFntfilOpen"
00255     #define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
00256     #define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
00257     #define TCS_INIDEF_NOFNTL 10
00258  #define TCS_INIVAR_HDCOPN "G2dHdcOpen"
00259     #define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
00260     #define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
00261     #define TCS_INIDEF_HDCOPNL 5
00262  #define TCS_INIVAR_HDCWRT "G2dHdcWrite"
00263     #define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
00264     #define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
00265     #define TCS_INIDEF_HDCWRTL 5
00266  #define TCS_INIVAR_HDCINT "G2dHdcIntern"
00267     #define TCS_INIDEF_HDCINT "GRAPH2D HARDCOPY: Internal Error."
00268     #define TCS_INIVAR_HDCINTL "G2dHdcInternL"
00269     #define TCS_INIDEF_HDCINTL 5
00270  #define TCS_INIVAR_USR "G2dUser"
00271     #define TCS_INIDEF_USR "%s"
00272     #define TCS_INIVAR_USRL "G2dUserL"
00273     #define TCS_INIDEF_USRL 5
00274  #define TCS_INIVAR_HDCACT "G2dHdcActive"
00275     #define TCS_INIDEF_HDCACT "Hardcopy in progress: File %s created."
00276     #define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
00277     #define TCS_INIDEF_HDCACTL 1
```

```
00278  #define TCS_INIVAR_USRWRN "G2dPressAny"
00279     #define TCS_INIDEF_USRWRN "Press any key to continue."
00280     #define TCS_INIVAR_USRWRNL "G2dPressAnyL"
00281     #define TCS_INIDEF_USRWRNL 5
00282  #define TCS_INIVAR_EXIT "G2dExit"
00283     #define TCS_INIDEF_EXIT "Press any key to exit program."
00284     #define TCS_INIVAR_EXITL "G2dExitL"
00285     #define TCS_INIDEF_EXITL 10
00286  #define TCS_INIVAR_COPMEM "G2dNoMemory"
00287     #define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager: Out of Memory."
00288     #define TCS_INIVAR_COPMEML "G2dNoMemoryL"
00289     #define TCS_INIDEF_COPMEML 1
00290  #define TCS_INIVAR_COPLCK "G2dClipLock"
00291     #define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager: ClipBoard locked."
00292     #define TCS_INIVAR_COPLCKL "G2dClipLockL"
00293     #define TCS_INIDEF_COPLCKL 1
00294  #define TCS_INIVAR_JOUCREATE "G2dJouCreate"
00295     #define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal. Error-No: %s."
00296     #define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
00297     #define TCS_INIDEF_JOUCREATEL 5
00298  #define TCS_INIVAR_JOUENTRY "G2dJouEntry"
00299     #define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
00300     #define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
00301     #define TCS_INIDEF_JOUENTRYL 5
00302  #define TCS_INIVAR_JOUADD "G2dJouAdd"
00303     #define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
00304     #define TCS_INIVAR_JOUADDL "G2dJouAddL"
00305     #define TCS_INIDEF_JOUADDL 5
00306  #define TCS_INIVAR_JOUCLR "G2dJouClr"
00307     #define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
00308     #define TCS_INIVAR_JOUCLRL "G2dJouClrL"
00309     #define TCS_INIDEF_JOUCLRL 5
00310  #define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
00311     #define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
00312     #define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
00313     #define TCS_INIDEF_JOUUNKWNL 5
00314  #define TCS_INIVAR_XMLPARSER "G2dXMLerror"
00315     #define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File: %s"
00316     #define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
00317     #define TCS_INIDEF_XMLPARSERL 8
00318  #define TCS_INIVAR_XMLOPEN "G2dXMLopen"
00319     #define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
00320     #define TCS_INIVAR_XMLOPENL "G2dXMLerrorL"
00321     #define TCS_INIDEF_XMLOPENL 8
00322  #define TCS_INIVAR_UNKNAUDIO "G2dAudio"
00323     #define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System: Error %s."
00324     #define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
00325     #define TCS_INIDEF_UNKNAUDIOL 5
00326  #define TCS_INIVAR_USR2 "G2dUser2"
00327     #define TCS_INIDEF_USR2 "%s"
00328     #define TCS_INIVAR_USR2L "G2dUser2L"
00329     #define TCS_INIDEF_USR2L 5
00330  #define TCS_INIVAR_INI2 "G2d2xInitt"
00331     #define TCS_INIDEF_INI2 "%s"
00332     #define TCS_INIVAR_INI2L "G2d2xInittL"
00333     #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

```
00001 C> \file Tktrnx.fd
00002 C> \brief   SDL Port: TCS Common Block TKTRNX
00003 C> \version 1.2
00004 C> \author  Dr.-Ing. Klaus Friedewald
00005 C> \~german
00006 C> Header passend zu TKTRNX.h
00007 C> \note
00008 C> Da die folgende Definition kein Bestandteil eines Moduls
00009 C> ist, versagt der DOXYGEN-Parser bei der Kombination von
00010 C> COMMON und INTEGER. Workaraound: \\cond ... \\endcond.
00011 C> \~english
00012 C> header belonging to TKTRNX.h
00013 C> \note
00014 C> Because the following definition not beeing part of a module, the
00015 C> DOXYGEN parser is not able to handle the combination of COMMON
00016 C> and INTEGER declarations.  Workaraound: \\cond ... \\endcond.
00017 C> \~
00018 C> \cond
00019
00020       COMMON /tktrnx/
00021      & khomey,
00022      & khorsz,kversz,
00023      & kitalc,ksizef,
00024      & klmrgn,krmrgn,
00025      & kbeamx,kbeamy,
00026      & kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,
00027      & trcosf,trsinf,trscal
00028      & ,xfac,yfac,xlog,ylog,kstcol,
00029      & ilincol, ibckcol, itxtcol
00030
00031       SAVE /tktrnx/
00032       integer iTktrnxL
00033       parameter(itktrnxl=28) ! +11)
00034 C Neue Variablen:
00035 C     kHorSz,kVerSz: Buchstabengröße im (1024/780) Koordinatensystem
00036 C     kBeamX, kBeamY: Aktuelle Strahlposition im (1024/780) Koordinatensystem
00037 C     kStCol: Maximale Zeichenzahl in der Statuszeile
00038 C     iLinCol, iBckCol, iTxtCol: Farbindices
00039 C
00040 C Achtung:
00041 C       Anpassung Parameters iTktrnxL der Routinen SVSTAT, RESTAT aus TCS.FOR!
00042 C       Vorsicht, bei Integer*2 Variablen zählen Real-Variablen doppelt (*4!)
00043 C
00044 C> \endcond
00045
```

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

```
00001 /** ****************************************************************************
00002 \file     TKTRNX.h
00003 \brief    SDL Port: TCS Common Block TKTRNX
00004 \version 1.2
00005 \author  Dr.-Ing. Klaus Friedewald
00006 \~german
00007          C Header passend zu TKTRNX.fd
00008 \~english
00009          C header belonging to TKTRNX.fd
00010 \~
00011
00012 \note
00013    SDL-Version auf Basis der Windows-Version 1.2
00014    Anpassung an die compilerabhaengige Namenskonvention erfolgt in TCSdSDLc.h
00015
00016 ******************************************************************************* */
00017
00018
00019 extern struct TKTRNXcommonBlock {
00020 FTNINT
00021      khomey,
00022      khorsz,kversz,
00023      kitalc,ksizef,
00024      klmrgn,krmrgn,
00025      kBeamX,kBeamY,
00026      kminsx,kminsy,kmaxsx,kmaxsy;
00027
00028 FTNREAL
00029      tminvx,tminvy,tmaxvx,tmaxvy,
00030      trcosf,trsinf,trscal
00031      ,xfac,yfac,xlog,ylog;
00032 FTNINT
00033     kStCol,
00034     iLinCol, iBckCol, iTxtCol;
00035 } TKTRNX;
```

Index