Graph2D Library --- SDL2 ---

Generated by Doxygen 1.8.19

Chapter 1

Plot10 & Advanced Graphing II

Graph2D is completly written in FTN77 and ANSI C90. Detailed compiling 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 invoking "$getfiles.bat sdlxx". Then use the workspace files for CodeBlocks (Windows IDE) or the bashscript for Linux.

1.0.0.2 Using the library:

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

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

1.0.0.3 Hardcopies

generate proprietary ASCII-journalfiles with the default extension ∗.hdc.

Chapter 2

Compilersettings for Windows

2.0.1 Setup of 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- 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 at Settings -> Compiler:

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

In order to build 32bit programs the global GCC settings have to be changed accordingly. The 32bit settings define new compilers and can now be distinguished from the 64bit versions when used inside the 32bit workspaces.

2.0.1.2 Building the OpenSource libraries SDL2, SDL2_ttf, miniXML und 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, seperately for 32- and 64-bit.

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

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

2.0.1.3 Settings for own 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 at the axis!
- "Strip all symbols from binary [-s]" is possible.

2.0.1.3.3 Link

- static: enables executing of the programs on machines without MinGW installed.

Chapter 3

Compilersettings 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

Installation Fortran:

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

Installation 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

Installation MiniXML:

- # sudo apt-get install libmxml-dev

3.0.1.2 Compilation

Copy the directory Teklib\Build to the target machine. Set the batchfile executable:

- # chmod 755 build.sh

Build the library and the 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 39 of file TCSdSDLc.h.

6.1.2 Member Data Documentation

6.1.2.1 imag

`float FTNCOMPLEX::imag`

Definition at line 39 of file TCSdSDLc.h.

6.1.2.2 real

`float FTNCOMPLEX::real`

Definition at line 39 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 46 of file TCSdSDLc.h.

6.2.2 Member Data Documentation

6.2.2.1 addr

`FTNCHAR* FTNSTRDESC::addr`

Definition at line 46 of file TCSdSDLc.h.

6.2.2.2 len

`FTNCHARLEN FTNSTRDESC::len`

Definition at line 46 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 244 of file TCSdSDLc.c.

6.4.2 Member Data Documentation

6.4.2.1 action

```
FTNINT xJournalEntry_typ::action
```

Definition at line 246 of file TCSdSDLc.c.

6.4.2.2 i1

```
FTNINT xJournalEntry_typ::i1
```

Definition at line 246 of file TCSdSDLc.c.

6.4.2.3 i2

```
FTNINT xJournalEntry_typ::i2
```

Definition at line 246 of file TCSdSDLc.c.

6.4.2.4 next

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

Definition at line 245 of file TCSdSDLc.c.

6.4.2.5 previous

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

Definition at line 244 of file TCSdSDLc.c.

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

- TCSdSDLc.c

Chapter 7

File Documentation

7.1 AG2.for File Reference

Graph2D: Tektronix Advanced Graphing II Emulation.

Functions/Subroutines

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

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

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

7.1.1 Detailed Description

Graph2D: Tektronix Advanced Graphing II Emulation.

Version

(2022,284, x)

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Layer 2: scientific 2-D graphic subroutines

Note

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

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

Definition in file AG2.for.

7.1.2 Function/Subroutine Documentation

7.1.2.1 ag2lev()

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

Definition at line 94 of file AG2.for.

7.1.2.2 alfsetc()

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

Definition at line 2564 of file AG2.for.

7.1.2.3 bar()

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

Definition at line 1689 of file AG2.for.

7.1.2.4 binitt()

```
subroutine binitt
```

Definition at line 714 of file AG2.for.

7.1.2.5 bsyms()

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

Definition at line 1841 of file AG2.for.

7.1.2.6 calcon()

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

Definition at line 1326 of file AG2.for.

7.1.2.7 calpnt()

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

Definition at line 1271 of file AG2.for.

7.1.2.8 check()

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

Definition at line 798 of file AG2.for.

7.1.2.9 cmnmx()

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

Definition at line 920 of file AG2.for.

7.1.2.10 coptim()

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

Definition at line 1115 of file AG2.for.

7.1.2.11 cplot()

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

Definition at line 1539 of file AG2.for.

7.1.2.12 datget()

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

Definition at line 1661 of file AG2.for.

7.1.2.13 dinitx()

```
subroutine dinitx
```

Definition at line 644 of file AG2.for.

7.1.2.14 dinity()

```
subroutine dinity
```

Definition at line 658 of file AG2.for.

7.1.2.15 dlimx()

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

Definition at line 464 of file AG2.for.

7.1.2.16 dlimy()

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

Definition at line 476 of file AG2.for.

7.1.2.17 dsplay()

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

Definition at line 1525 of file AG2.for.

7.1.2.18 eformc()

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

Definition at line 2435 of file AG2.for.

7.1.2.19 esplit()

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

Definition at line 2468 of file AG2.for.

7.1.2.20 expoutc()

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

Definition at line 2488 of file AG2.for.

7.1.2.21 fformc()

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

Definition at line 2376 of file AG2.for.

7.1.2.22 filbox()

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

Definition at line 1756 of file AG2.for.

7.1.2.23 findge()

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

Definition at line 2923 of file AG2.for.

7.1.2.24 findle()

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

Definition at line 2942 of file AG2.for.

7.1.2.25 fonlyc()

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

Definition at line 2404 of file AG2.for.

7.1.2.26 frame()

```
subroutine frame
```

Definition at line 1511 of file AG2.for.

7.1.2.27 gline()

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

Definition at line 2174 of file AG2.for.

7.1.2.28 grid()

```
subroutine grid
```

Definition at line 1957 of file AG2.for.

7.1.2.29 hbarst()

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

Definition at line 672 of file AG2.for.

7.1.2.30 iformc()

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

Definition at line 2344 of file AG2.for.

7.1.2.31 infin()

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

Definition at line 142 of file AG2.for.

7.1.2.32 iother()

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

Definition at line 3067 of file AG2.for.

7.1.2.33 iubgc()

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

Definition at line 1474 of file AG2.for.

7.1.2.34 justerc()

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

Definition at line 2667 of file AG2.for.

7.1.2.35 keyset()

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

Definition at line 1635 of file AG2.for.

7.1.2.36 label()

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

Definition at line 2201 of file AG2.for.

7.1.2.37 leap()

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

Definition at line 1460 of file AG2.for.

7.1.2.38 line()

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

Definition at line 109 of file AG2.for.

7.1.2.39 locge()

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

Definition at line 2964 of file AG2.for.

7.1.2.40 locle()

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

Definition at line 2982 of file AG2.for.

7.1.2.41 logtix()

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

Definition at line 2043 of file AG2.for.

7.1.2.42 loptim()

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

Definition at line 988 of file AG2.for.

7.1.2.43 lwidth()

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

Definition at line 2733 of file AG2.for.

7.1.2.44 mnmx()

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

Definition at line 881 of file AG2.for.

7.1.2.45 monpos()

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

Definition at line 2160 of file AG2.for.

7.1.2.46 notatec()

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

Definition at line 2619 of file AG2.for.

7.1.2.47 npts()

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

Definition at line 155 of file AG2.for.

7.1.2.48 numsetc()

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

Definition at line 2317 of file AG2.for.

7.1.2.49 optim()

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

Definition at line 971 of file AG2.for.

7.1.2.50 oubgc()

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

Definition at line 1488 of file AG2.for.

7.1.2.51 place()

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

Definition at line 512 of file AG2.for.

7.1.2.52 remlab()

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

Definition at line 2808 of file AG2.for.

7.1.2.53 rescom()

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

Definition at line 3051 of file AG2.for.

7.1.2.54 rgchek()

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

Definition at line 854 of file AG2.for.

7.1.2.55 roundd()

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

Definition at line 3000 of file AG2.for.

7.1.2.56 roundu()

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

Definition at line 3016 of file AG2.for.

7.1.2.57 savcom()

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

Definition at line 3035 of file AG2.for.

7.1.2.58 setwin()

```
subroutine setwin
```

Definition at line 622 of file AG2.for.

7.1.2.59 sizel()

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

Definition at line 188 of file AG2.for.

7.1.2.60 sizes()

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

Definition at line 177 of file AG2.for.

7.1.2.61 slimx()

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

Definition at line 488 of file AG2.for.

7.1.2.62 slimy()

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

Definition at line 500 of file AG2.for.

7.1.2.63 spread()

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

Definition at line 2871 of file AG2.for.

7.1.2.64 stepl()

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

Definition at line 166 of file AG2.for.

7.1.2.65 steps()

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

Definition at line 131 of file AG2.for.

7.1.2.66 symbl()

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

Definition at line 120 of file AG2.for.

7.1.2.67 symout()

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

Definition at line 1858 of file AG2.for.

7.1.2.68 teksym()

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

Definition at line 1883 of file AG2.for.

7.1.2.69 teksym1()

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

Definition at line 1931 of file AG2.for.

7.1.2.70 tset()

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

Definition at line 2090 of file AG2.for.

7.1.2.71 tset2()

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

Definition at line 2128 of file AG2.for.

7.1.2.72 typck()

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

Definition at line 823 of file AG2.for.

7.1.2.73 vbarst()

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

Definition at line 692 of file AG2.for.

7.1.2.74 vlablc()

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

Definition at line 2644 of file AG2.for.

7.1.2.75 width()

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

Definition at line 2692 of file AG2.for.

7.1.2.76 xden()

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

Definition at line 312 of file AG2.for.

7.1.2.77 xetyp()

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

Definition at line 596 of file AG2.for.

7.1.2.78 xfrm()

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

Definition at line 390 of file AG2.for.

7.1.2.79 xlab()

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

Definition at line 290 of file AG2.for.

7.1.2.80 xlen()

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

Definition at line 364 of file AG2.for.

7.1.2.81 xloc()

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

Definition at line 246 of file AG2.for.

7.1.2.82 xloctp()

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

Definition at line 268 of file AG2.for.

7.1.2.83 xmfrm()

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

Definition at line 438 of file AG2.for.

7.1.2.84 xmtcs()

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

Definition at line 416 of file AG2.for.

7.1.2.85 xneat()

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

Definition at line 202 of file AG2.for.

7.1.2.86 xtics()

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

Definition at line 342 of file AG2.for.

7.1.2.87 xtype()

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

Definition at line 544 of file AG2.for.

7.1.2.88 xwdth()

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

Definition at line 570 of file AG2.for.

7.1.2.89 xzero()

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

Definition at line 224 of file AG2.for.

7.1.2.90 yden()

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

Definition at line 327 of file AG2.for.

7.1.2.91 yetyp()

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

Definition at line 609 of file AG2.for.

7.1.2.92 yfrm()

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

Definition at line 403 of file AG2.for.

7.1.2.93 ylab()

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

Definition at line 301 of file AG2.for.

7.1.2.94 ylen()

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

Definition at line 377 of file AG2.for.

7.1.2.95 yloc()

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

Definition at line 257 of file AG2.for.

7.1.2.96 ylocrt()

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

Definition at line 279 of file AG2.for.

7.1.2.97 ymdyd()

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

Definition at line 1405 of file AG2.for.

7.1.2.98 ymfrm()

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

Definition at line 451 of file AG2.for.

7.1.2.99 ymtcs()

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

Definition at line 427 of file AG2.for.

7.1.2.100 yneat()

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

Definition at line 213 of file AG2.for.

7.1.2.101 ytics()

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

Definition at line 353 of file AG2.for.

7.1.2.102 ytype()

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

Definition at line 557 of file AG2.for.

7.1.2.103 ywdth()

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

Definition at line 583 of file AG2.for.

7.1.2.104 yzero()

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

Definition at line 235 of file AG2.for.

7.2 AG2.for

```
00001 C> \file       AG2.for
00002 C> \brief      Graph2D: Tektronix Advanced Graphing II Emulation
00003 C> \version    (2022,284, x)
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C>
00007 C> \~german
00008 C>  Schicht 2: Unterprogramme zur Erzeugung wissenschaftlicher 2-D Graphiken
00009 C> \note
00010 C>     Die Sonderzeichen Hochindex (alt: -1) und Index (alt: -2) sind jetzt
00011 C>     SOH=char(1) (Hochindex) bzw. STX=char(2) (Index).
00012 C>
00013 C> \~english
00014 C> Layer 2: scientific 2-D graphic subroutines
00015 C> \note
00016 C>     The control character for exponent (originally -1) is now SOH=char(1)
00017 C>     and for index (originally -2) STX=char(2).
00018 C>
00019 C> \~
00020 C> \note \verbatim
00021 C>   Package:
00022 C>    - AG2.for:         chart plotting routines
00023 C>    - AG2Holerith.for: deprecated routines
00024 C>    - AG2USR.for:      default 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)=2022       ! Aenderungsjahr
00100       ilevel(2)= 284       ! 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       subroutine npts (ipar)
00156       implicit none
00157       integer ipar
00158       include 'G2dAG2.fd'
```

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

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

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

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

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

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

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

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

```
00855       implicit none
00856       integer ixy
00857       real arr(5)
00858       real amin, amax
00859       include 'G2dAG2.fd'
00860
00861       if (cxydmax(ixy) .eq. cxydmin(ixy)) then ! Bereich schon bestimmt?
00862        if (cxyzero(ixy)) then ! Nullpunktunterdrueckung?
00863         amin= cinfin
00864        else
00865         amin= 0.
00866        end if
00867        amax= -amin
00868        call mnmx (arr, amin, amax)
00869        if (amax .eq. amin) then
00870         amin= amin - 0.5
00871         amax= amax + 0.5
00872        end if
00873        cxydmin(ixy)= amin
00874        cxydmax(ixy)= amax
00875       end if
00876       return
00877       end
00878
00879
00880
00881       subroutine mnmx (arr,amin,amax)
00882       implicit none
00883       real arr(5), amin,amax, aminmax
00884       integer i, itype, nstart,nlim
00885       include 'G2dAG2.fd'
00886
00887       if (cnpts .eq. 0) then                          ! Tek Standard-Format
00888        nlim= nint(arr(1)) + 1
00889        nstart= 2
00890       else
00891        nlim= cnpts
00892        nstart= 1
00893       end if
00894       if ((arr(1) .lt. 0.) .and. (cnpts .eq. 0))  then ! Kurzformate
00895        itype= abs(arr(1))
00896        if (itype .eq. 1) then
00897         aminmax= arr(3) + (arr(2)-1.) * arr(4)
00898         amin= amin1(arr(3),aminmax,amin)
00899         amax= amax1(arr(3),aminmax,amax)
00900        else if (itype .eq. 2) then
00901         call cmnmx (arr,amin,amax)
00902        else
00903         call umnmx (arr,amin,amax)
00904        end if
00905       else                                            ! Langformate
00906        if (nstart .le. nlim) then
00907         do 100 i= nstart, nlim
00908          if (arr(i) .lt. cinfin) then
00909           if (arr(i).lt. amin) amin= arr(i)
00910           if (arr(i).gt. amax) amax= arr(i)
00911          end if
00912 100     continue
00913        end if
00914       end if
00915       return
00916       end
00917
00918
00919
00920       subroutine cmnmx (arr,amin,amax)
00921       implicit none
00922       real arr(5), amin, amax
00923       integer nTage, iStUBGC, nIntv, iadj, imin,imax
00924       integer minTg,minJr, maxTg,maxJr
00925
00926
00927       nintv= nint(arr(3))
00928       if ((nintv .eq. 52).or.(nintv .eq. 13).or.(nintv .eq. 4)) then
00929        if (nintv .eq. 52) then          ! Wochen
00930         ntage=7
00931        else if (nintv .eq. 13) then     ! 28 Tagemonat
00932         ntage= 28
00933        else if (nintv .eq. 4) then      ! Quartal
00934         ntage=91
00935        end if
00936        call iubgc (nint(arr(4)),1, istubgc)    ! Start: Jahr=arr(4), Tag=1
00937        iadj= mod(istubgc,7)
00938        if (iadj .gt. 3) iadj=iadj-7
00939        imin= istubgc-iadj + nint(arr(5))*ntage ! Min= f(Startjahr,StartIntervall)
00940        imax= imin + nint(arr(2))*ntage
00941
```

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

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

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

```
01203          else if (labtyp.eq.7) then ! Labeltyp: Quartale
01204           if (dataint .le. 1.) then
01205            dataint= 1.
01206           else if (dataint .le. 2.) then
01207            dataint= 2.
01208           else if (dataint .le. 4.) then
01209            dataint= 4.
01210           else if (dataint .le. 8.) then
01211            dataint= 8.
01212           else if (dataint .le. 12.) then
01213            dataint= 12.
01214           else if (dataint .le. 16.) then
01215            dataint= 16.
01216           else if (dataint .le. 24.) then
01217            dataint= 24.
01218           end if ! dataint -> unveraendert
01219          else if (labtyp.eq.8) then ! Labeltyp: Jahre
01220           if (dataint .le. 1.) then
01221            dataint= 1.
01222           else if (dataint .le. 2.) then
01223            dataint= 2.
01224           else if (dataint .le. 5.) then
01225            dataint= 5.
01226           else if (dataint .le. 10.) then
01227            dataint= 10.
01228           else if (dataint .le. 20.) then
01229            dataint= 20.
01230           else if (dataint .le. 50.) then
01231            dataint= 50.
01232           else if (dataint .le. 100.) then
01233            dataint= 100.
01234           end if ! dataint -> unveraendert
01235          end if ! labtyp 3..8
01236         end if ! manuelle Vorbesetzung
01237         amin= roundd(amin,dataint) !  runde auf TicIntervall
01238         amax= roundu(amax,dataint)
01239         ntics= ifix(abs(amax-amin)/dataint+.0001)
01240         if (ntics .eq. 0) ntics = 2
01241        if(cxytics(ixy) .ne. 0) then ! until: ntics nicht oder = vorbesetzt
01242         if(abs(cxytics(ixy)) .lt. ntics) then ! Verringere Ticanzahl
01243          dataint= dataint * 1.1
01244          amin=aminor
01245          amax=amaxor
01246          goto 310 ! noch eine Iterationsschleife
01247         else if (abs(cxytics(ixy)) .gt. ntics) then ! Vergroessere Ticanzahl
01248          ntics= abs(cxytics(ixy))
01249          amax= amin + real(ntics) * dataint
01250         end if ! abs(cxytics(ixy)) .eq. ntics: no action
01251        end if ! Ende der Schleife
01252       end if ! neat
01253       cxytics(ixy)= ntics
01254       cxylsig(ixy)= 0
01255       cxyamin(ixy)= amin
01256       cxyamax(ixy)= amax
01257       call calcon (amin,amax,labtyp,.false.) ! Labelzeiteinheit -> UBGC
01258       cxydmin(ixy)= amin
01259       cxydmax(ixy)= amax
01260       return
01261       end
01262
01263
01264
01265 C
01266 C  Kalenderroutinen
01267 C
01268
01269
01270
01271       real function calpnt (arr,i)
01272       implicit none
01273       integer i
01274       real arr(5)
01275       integer iy,idays, itmp
01276       integer icltyp, istyr, istper, iubg1, iweek1, nodays
01277       save icltyp, istyr, istper, iubg1, iweek1, nodays
01278
01279       if (i .eq. 1) then ! 1. Datenpunkt: Formatanalyse, Parameterberechnung
01280        istyr= nint(arr(4))
01281        istper= nint(arr(5))
01282        itmp= nint(arr(3)) ! Laenge Intervall in Tagen
01283        if (itmp .eq. 12) then ! Zeitintervall Monat
01284         icltyp= 2
01285        else if (itmp .eq. 365) then ! Zeitintervall Tage
01286         icltyp=3
01287         call iubgc (istyr,istper,iubg1)
01288        else if (itmp .eq. 52) then ! Zeitintervall Wochen
01289         icltyp= 4
```

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

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

```
01464       1     ((mod(iyear,100).ne.0) .or. (mod(iyear,400).eq.0))  ) then
01465         leap= 1
01466        else
01467         leap= 0
01468        end if
01469        return
01470        end
01471
01472
01473
01474        subroutine iubgc(iyear,iday, iubgcO)
01475        implicit none
01476        integer iyear,iday,iubgcO
01477        integer iYr1
01478
01479        iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
01480        iubgco= 365* (iyear-1901) ! Verhinderung Overflow: Offset im Faktor
01481        iubgco= iubgco + int(iyr1/4) - int(iyr1/100) + int(iyr1/400)
01482        iubgco= iubgco + iday -460 ! Bezugsdatum 1.1.1901= 365*1901 + 460 Schalttage
01483        return
01484        end
01485
01486
01487
01488        subroutine oubgc(iyear,iday,iubgcI)
01489        implicit none
01490        integer iyear,iday,iubgcI
01491        integer iYr1
01492
01493        iyear= int( (real(iubgci) + 694325.99) / 365.2425 )
01494 100    continue ! Schleife der evtl. Nachiteration
01495         iyr1= iyear-1 ! Schaltjahreskorrektur erst nach Jahresabschluss
01496         iday= iubgci + 460 - 365*(iyear-1901)
01497         iday= iday + int(iyr1/100) - int(iyr1/4) - int(iyr1/400)
01498        if (iday .lt. 1) then ! Nachiteration?
01499         iyear= iyear-1
01500         goto 100
01501        end if
01502        return
01503        end
01504
01505
01506
01507 C
01508 C  Zeichenroutinen
01509 C
01510
01511        subroutine frame
01512        implicit none
01513        include 'G2dAG2.fd'
01514
01515        call movabs (cxysmax(1),cxysmin(2))
01516        call drwabs (cxysmax(1),cxysmax(2))
01517        call drwabs (cxysmin(1),cxysmax(2))
01518        call drwabs (cxysmin(1),cxysmin(2))
01519        call drwabs (cxysmax(1),cxysmin(2))
01520        return
01521        end
01522
01523
01524
01525        subroutine dsplay (x,y)
01526        implicit none
01527        real x(5),y(5)
01528
01529        call setwin
01530        call cplot (x,y)
01531        call grid
01532        call label (1)
01533        call label (2)
01534        return
01535        end
01536
01537
01538
01539        subroutine cplot (x,y)
01540        implicit none
01541        real x(5),y(5)
01542        logical symbol
01543        integer i,i1, keyx, keyy, lines, linsav, icount, imax
01544        real xpoint(1), ypoint(1)
01545        real DATGET
01546        include 'G2dAG2.fd'
01547
01548        call keyset (x,keyx)
01549        call keyset (y,keyy)
01550        if (keyx .eq. 1) then ! standard long
```

```
01551        imax= x(1)
01552       else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
01553        imax= x(2)
01554       else ! nonstandard
01555        imax= cnpts
01556       end if
01557       if (keyy .eq. 1) then ! standard long
01558        if (imax .lt. y(1)) imax= y(1)
01559       else if ((keyx .ge. 2) .and. (keyx .le. 4)) then ! short
01560        if (imax .lt. y(2)) imax= y(2)
01561       else ! nonstandard
01562        if (imax .lt. cnpts) imax= cnpts
01563       end if
01564
01565       symbol= (csymbl .ne. 0) .and.(cline .ne.-2) .and.(cline .ne.-3)
01566
01567       i= 1 ! Suche Startpunkt
01568 100   continue ! repeat
01569        if (i .gt. imax) return ! kein Punkt zu zeichnen
01570        xpoint(1)= datget(x,i,keyx)
01571        ypoint(1)= datget(y,i,keyy)
01572       if ((xpoint(1) .ge. cinfin) .or. (ypoint(1) .ge. cinfin)) then ! while
01573        i= i+cstepl
01574        goto 100
01575       end if
01576
01577       call movea (xpoint(1),ypoint(1))
01578       if (cline .eq. -4) call pointa (xpoint(1),ypoint(1))
01579       if (cline .lt. -10) call uline (xpoint(1),ypoint(1),1)
01580       if (cline .eq.-2 .or. cline .eq.-3) then
01581        call bar (xpoint(1),ypoint(1),cline)
01582       end if
01583       if (symbol) call bsyms (xpoint(1),ypoint(1),csymbl)
01584
01585       if (cline .eq. -1) then
01586        lines= 2
01587       else if ((cline .eq. -2) .or. (cline .eq. -3)) then
01588        lines= 3
01589       else if (cline .eq. -4) then
01590        lines=4
01591       else if (cline .lt. -10) then
01592        lines=5
01593       else
01594        lines=1 ! bei cline = 0: dash ergibt durchgezogene Linie
01595       end if
01596
01597       i1= i+cstepl
01598       if (i1 .ge. imax) return
01599       icount= csteps
01600       linsav= lines
01601
01602       do 900 i=i1,imax,cstepl
01603        xpoint(1)= datget(x,i,keyx)
01604        ypoint(1)= datget(y,i,keyy)
01605        if ((xpoint(1) .ge. cinfin) .or. (ypoint(1) .ge. cinfin)) then
01606         if (i.gt.imax-cstepl) return ! Der letzte Punkt ist ungueltig -> done
01607         if ((cline .ne. -2) .and. (cline .ne. 3)) lines= 2
01608        else
01609         if (lines .eq. 1 ) then
01610          call dasha (xpoint(1),ypoint(1), cline) ! dashed or solid
01611         else if (lines .eq. 2 ) then
01612          call movea (xpoint(1),ypoint(1))
01613          lines=linsav ! restore after missing data
01614         else if (lines .eq. 3 ) then
01615          call bar (xpoint(1),ypoint(1),0)
01616         else if (lines .eq. 4 ) then
01617          call pointa (xpoint(1),ypoint(1))
01618         else
01619          call uline (xpoint(1),ypoint(1),i)
01620         end if
01621         if (symbol) then
01622          icount=icount-1
01623          if(icount .le. 0) then
01624           icount= csteps
01625           call bsyms (xpoint(1),ypoint(1),csymbl)
01626          end if
01627         end if
01628        end if
01629 900   continue
01630       return
01631       end
01632
01633
01634
01635       subroutine keyset (array,key)
01636       implicit none
01637       integer key
```

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

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

```
01812 120     continue ! repeat ...
01813          call movea (ximin,real(i))
01814          call drawa (ximax,real(i+idely))
01815          i= i+lspace
01816         if (i .lt. iymax) goto 120 ! ... until
01817         ishift= ishade -8
01818        else
01819         ishift= ishade
01820        end if
01821
01822        if (ishift .ge. 4) then ! Bit2: diagonal steigend
01823         idely= imaxx-iminx
01824         iymax= real(imaxy)
01825         i= iminy - idely + lspace
01826 130     continue ! repeat...
01827          call movea (ximin,real(i))
01828          call drawa (ximax,real(i+idely))
01829          i= i+lspace
01830         if (i .lt. iymax) goto 130 ! ...until
01831        end if
01832        call restat (savcom)
01833       end if ! Diagonalen
01834       return
01835       end
01836
01837
01838
01839 C  Zeichnen von Symbolen
01840
01841       subroutine bsyms (x,y,isym)
01842       implicit none
01843       real x,y
01844       integer isym
01845       include 'G2dAG2.fd'
01846
01847       if (isym .ge. 0) then
01848        call symout (isym, csizes)
01849       else
01850        call users (x,y,isym)
01851       end if
01852       call movea (x,y)
01853       return
01854       end
01855
01856
01857
01858       subroutine symout (isym,fac)
01859       implicit none
01860       integer isym
01861       real fac
01862       integer ix,iy, ihorz,ivert
01863
01864       call seeloc (ix,iy)
01865       if (isym .gt. 127) then
01866        call softek (isym)
01867       else if (isym .ge. 33) then
01868        call csize (ihorz,ivert)
01869        ihorz= int( real(ihorz)*.3572)
01870        ivert= int( real(ivert)*.3182)
01871        call movrel (-ihorz,-ivert)
01872        call alfmod
01873        call toutpt (isym)
01874       else if (isym .le. 11) then
01875        call teksym (isym,fac)
01876       end if
01877       call movabs (ix,iy)
01878       return
01879       end
01880
01881
01882
01883       subroutine teksym (isym,amult)
01884       implicit none
01885       integer isym
01886       real amult
01887       integer ihalf, ifull
01888
01889       ihalf= nint(8.* amult)
01890       ifull=ihalf * 2
01891       if (isym .eq. 1) then ! Kreis
01892        call teksym1 (0, 360, 30, 8.*amult)
01893       else if (isym .eq. 2) then ! X
01894        call movrel (ihalf,ihalf)
01895        call drwrel (-ifull,-ifull)
01896        call movrel (0,ifull)
01897        call drwrel (ifull,-ifull)
01898       else if (isym .eq. 3) then ! Dreieck
```

```
01899         call teksym1 (90, 450, 120, 8.*amult)
01900       else if (isym .eq. 4) then ! Quadrat
01901         call teksym1 (45, 405, 90, 8.*amult)
01902       else if (isym .eq. 5) then ! Stern
01903         call teksym1 (90, 810, 144, 8.*amult)
01904       else if (isym .eq. 6) then ! Raute
01905         call teksym1 (90, 450, 90, 8.*amult)
01906       else if (isym .eq. 7) then ! vertikaler Balken
01907         call teksym1 (90, 270, 180, 8.*amult)
01908       else if (isym .eq. 8) then ! Kreuz
01909         call movrel (0,ihalf)
01910         call drwrel (0,-ifull)
01911         call movrel (-ihalf,ihalf)
01912         call drwrel (ifull,0)
01913       else if (isym .eq. 9) then ! Pfeil nach oben
01914         call drwrel (-2,-6)
01915         call drwrel (4,0)
01916         call drwrel (-2,6)
01917         call drwrel (0,-ifull)
01918       else if (isym .eq. 10) then ! Pfeil nach unten
01919         call drwrel (-2,6)
01920         call drwrel (4,0)
01921         call drwrel (-2,-6)
01922         call drwrel (0,ifull)
01923       else if (isym .eq. 11) then ! Durchstreichung
01924         call teksym1 (270, 630, 120, 8.*amult)
01925       end if
01926       return
01927       end
01928
01929
01930
01931       subroutine teksym1 (istart, iend, incr, siz)
01932       implicit none
01933       integer istart, iend, incr
01934       real siz
01935       integer i, mx,my,mix,miy
01936       real b
01937
01938       b= real(istart)*.01745
01939       mx= nint(siz*cos(b))
01940       my= nint(siz*sin(b))
01941       call movrel (mx,my)
01942       do 100 i= istart+incr, iend, incr
01943         b= real(i)*.01745
01944         mix= nint(siz*cos(b))
01945         miy= nint(siz*sin(b))
01946         call drwrel (mix-mx,miy-my)
01947         mx= mix
01948         my= miy
01949 100   continue
01950       return
01951       end
01952
01953
01954
01955 C Netz und Ticmarks
01956
01957       subroutine grid
01958       implicit none
01959       integer i, mlim
01960       real xyext,xyextm, tintvl,tmntvl
01961       include 'G2dAG2.fd'
01962
01963       if (cxyfrm(2) .ne. 0) then ! Zeichnen der y-Achse
01964         i= min0(cxysmin(1),cxysmax(1)) + cxyloc(2)
01965         call movabs (i, cxysmax(2))
01966         call drwabs (i, cxysmin(2))
01967         if (cxybeg(2) .ne. cxyend(2)) then ! Zeichnen y-Ticmarks
01968           i= cxylab(2) ! Labeltyp
01969           if (i .eq. 1) i= cxytype(2) ! =1: Typ entsprechend Daten
01970           if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
01971             if(cxytics(2) .ne. 0) then
01972               tintvl= real(cxysmax(2)-cxysmin(2)) / real( cxytics(2))
01973             end if
01974             if (cxymtcs(2) .gt. 0) tmntvl= tintvl / real(cxymtcs(2))
01975             call movabs(cxybeg(2),cxysmin(2))
01976             call drwabs(cxyend(2),cxysmin(2))
01977             xyext= real(cxysmin(2))
01978             do 100, i=1,cxytics(2)
01979               if (cxymbeg(2) .ne. cxymend(2)) then ! Zeichnen Minor Ticmarks
01980                 mlim= cxymtcs(2)-1
01981                 xyextm= xyext
01982 110             continue ! repeat...
01983                 if (mlim.gt.0) then ! ...until mlim <= 0
01984                   xyextm= xyextm+tmntvl
01985                   call movabs (cxymbeg(2), nint(xyextm))
```

```
01986             call drwabs (cxymend(2), nint(xyextm))
01987             mlim=mlim-1
01988             goto 110
01989            else if (mlim. lt. 0) then
01990             call logtix (2,xyext,tintvl,cxymbeg(2),cxymend(2))
01991            end if
01992           end if
01993           xyext= xyext+tintvl
01994           call movabs (cxybeg(2), nint(xyext))
01995           call drwabs (cxyend(2), nint(xyext))
01996 100      continue
01997         end if ! Labtyp=6: Monate
01998        end if ! Ende Zeichnen Ticmarks
01999       end if ! Ende Zeichnen der Achse
02000
02001       if (cxyfrm(1) .ne. 0) then ! Zeichnen der x-Achse
02002        i= min0(cxysmin(2),cxysmax(2)) + cxyloc(1)
02003        call movabs (cxysmin(1), i)
02004        call drwabs (cxysmax(1), i)
02005        if (cxybeg(1) .ne. cxyend(1)) then ! Zeichnen y-Ticmarks
02006         i= cxylab(1) ! Labeltyp
02007         if (i .eq. 1) i= cxytype(1) ! =1: Typ entsprechend Daten
02008         if (i .ne. 6) then ! =6 (Monate): Tics durch GLINE zeichnen lassen
02009          if(cxytics(1) .ne. 0) then
02010           tintvl= real(cxysmax(1)-cxysmin(1)) / real( cxytics(1))
02011          end if
02012          if (cxymtcs(1) .gt. 0) tmntvl= tintvl / real(cxymtcs(1))
02013          call movabs(cxysmin(1), cxybeg(1))
02014          call drwabs(cxysmin(1), cxyend(1))
02015          xyext= real(cxysmin(1))
02016          do 120, i=1,cxytics(1)
02017           if (cxymbeg(1) .ne. cxymend(1)) then ! Zeichnen Minor Ticmarks
02018            mlim= cxymtcs(1)-1
02019            xyextm= xyext
02020 130          continue ! repeat...
02021            if (mlim.gt.0) then ! ...until mlim <= 0
02022             xyextm= xyextm+tmntvl
02023             call movabs (nint(xyextm), cxymbeg(1))
02024             call drwabs (nint(xyextm), cxymend(1))
02025             mlim=mlim-1
02026             goto 130
02027            else if (mlim. lt. 0) then
02028             call logtix (1,xyext,tintvl,cxymbeg(1),cxymend(1))
02029            end if
02030           end if
02031           xyext= xyext+tintvl
02032           call movabs (nint(xyext), cxybeg(1))
02033           call drwabs (nint(xyext), cxyend(1))
02034 120      continue
02035         end if ! Labtyp=6: Monate
02036        end if ! Ende Zeichnen Ticmarks
02037       end if ! Ende Zeichnen der Achse
02038       return
02039       end
02040
02041
02042
02043       subroutine logtix (nbase,start,tintvl,mstart,mend)
02044       implicit none
02045       integer nbase,mstart,mend
02046       real start, tintvl
02047       integer i, logtic, ihorz, ivert, idx,idy
02048       character*1 loglab
02049       include 'G2dAG2.fd'
02050
02051       call csize (ihorz,ivert)
02052       do 100 i=2,9
02053        write (unit=loglab, fmt='(i1)') i ! Unicodefaehig durch Compilerfeature
02054        logtic= nint(log10(real(i))*tintvl + start)
02055        if (nbase .eq. 1) then ! x-Achse
02056         idx=  -ihorz/3
02057         if  (mstart .gt. mend) then
02058          idy= ivert
02059         else
02060          idy= -ivert
02061         end if
02062         call movabs (logtic,mend)
02063         call drwabs (logtic,mstart)
02064         if (cxymtcs(nbase) .eq. -2) then ! numerisches Ticmarklabel
02065          call movrel (idx,idy)
02066          call toutstc (loglab)
02067         end if
02068
02069        else if (nbase .eq. 2) then ! y-Achse
02070         if  (mstart .gt. mend) then
02071          idx= ihorz
02072         else
```

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

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

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

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

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

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

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

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

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

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

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

```
03030
03031
03032 C
03033 C  Generelle Manipulationen der Commonvariablen
03034 C
03035       subroutine savcom (Array)
03036       implicit none
03037       integer array(1)
03038       include 'G2dAG2.fd'
03039
03040       integer i
03041       integer arr(1)
03042       equivalence(arr(1),cline)
03043       do 10 i=1,g2dag2l
03044        array(i)= arr(i)
03045 10    continue
03046       return
03047       end
03048
03049
03050
03051       subroutine rescom (Array)
03052       implicit none
03053       integer array(1)
03054       include 'G2dAG2.fd'
03055
03056       integer i
03057       integer arr(1)
03058       equivalence(arr(1),cline)
03059       do 10 i=1,g2dag2l
03060        arr(i)= array(i)
03061 10    continue
03062       return
03063       end
03064
03065
03066
03067       integer function iother (ipar)
03068       implicit none
03069       integer ipar
03070
03071       if (mod(ipar,2) .eq. 1) then ! ungerader Parameter=x-Achse
03072        iother= ipar+1
03073       else
03074        iother= ipar-1
03075       end if
03076       return
03077       end
```

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)=',l14,', (2)=',l14,',   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)=',l14,', (2)=',l14,',  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

Utility: Restore Hardcopies.

Functions/Subroutines

- logical function gethdc (Filnam)

7.21.1 Detailed Description

Utility: Restore Hardcopies.

Version

1.0

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      Utility: Restore Hardcopies
00003 C> \version    1.0
00004 C> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Einlesen und Zeichnen von Hardcopydateien\n
00008 C> Verwendete temporaeres Ein/Ausgabeunit: 41. Falls bereits belegt, wird ein freier Kanal gesucht
00009 C> \~english
00010 C> Read and plot hardcopies\n
00011 C> Temporary input unit: 41. If already used, an other channel will be searched.
00012 C> \~
00013 C
00014
00015       logical function gethdc (Filnam)
00016 C> \param FilNam: Hardcopyfie
00017 C> \result (optional) .true. -> Error
00018       implicit none
00019       integer tcs_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 ! unknown
00100           continue
00101         end if
00102       if (ios.eq.0) goto 10 ! until EOF
00103
00104       close (iunit)
00105       gethdc= .false.
00106       return
00107
00108  99       continue ! Error Exit
00109           call graphicerror (8, ' ')
00110       return
00111       end
```

7.23 Mainpage.dox File Reference

7.24 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)

7.24.1 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.

7.24.2 Function/Subroutine Documentation

7.24.2.1 istringlen()

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

Definition at line 94 of file Strings.for.

7.24.2.2 itrimlen()

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

Definition at line 133 of file Strings.for.

7.24.2.3 printstring()

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

Definition at line 114 of file Strings.for.

7.24.2.4 substitute()

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

Definition at line 30 of file Strings.for.

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

```
00086         inext=0
00087        end if
00088       end do
00089       return
00090       end
00091
00092
00093
00094       function istringlen (String)
00095 C
00096 C Ermittelt die Stringlänge bei durch char(0) abgeschlossenen STRINGs.
00097 C Falls kein char(0) vorhanden ist, wird die Gesamtlänge übergeben.
00098 C
00099       implicit none
00100       character *(*) string
00101       integer istringlen, i
00102
00103       i= index(string,char(0))-1
00104       if (i.ge.0) then
00105        istringlen=i
00106       else
00107        istringlen= len(string)
00108       end if
00109       return
00110       end
00111
00112
00113
00114       character*(*) function printstring (String)
00115 C
00116 C  Kopiert STRING in einen variabel langen PRINTSTRING. Hierdurch wird
00117 C  der Ausdruck von Nullstrings (Fortran-Fehler!) vermieden.
00118 C
00119       implicit none
00120       character string *(*)
00121       integer istringlen
00122
00123       if (istringlen(string).gt.0) then
00124        printstring= string(1:istringlen(string))
00125       else
00126        printstring= ' '
00127       end if
00128       return
00129       end
00130
00131
00132
00133       integer function itrimlen (string)
00134 C
00135 C  Bestimmt die Länge des Strings ohne angehängte Leerzeichen.
00136 C  Bei Bedarf wird ein Char(0) angehaengt. Es darf in Ftn77 nie ein
00137 C  Nullstring erzeugt werden, da sonst die RTL-Library abstuerzt. Deswegen
00138 C  ist der kleinste erzeugte String ein Blank ' '.
00139 C
00140       implicit none
00141       character *(*) string
00142       integer i, istringlen
00143
00144       i=istringlen(string) +1
00145
00146  10   continue
00147        i= i-1
00148        if (i.ge.1) then
00149         if (string(i:i).eq.' ') goto 10
00150       end if
00151       itrimlen=i
00152       if ((i.lt.len(string)).and.(len(string).gt.1)) then
00153        string(i+1:i+1)= char(0) ! .gt.1: Achtung, nie Nullstring erzeugen!
00154       end if
00155       return
00156       end
00157
```

7.26 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)

7.26.1 Detailed Description

TCS: Tektronix Plot 10 Emulation.

Version

4.0

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

System independent subroutines

Definition in file TCS.for.

7.26.2 Function/Subroutine Documentation

7.26.2.1 ancho()

```
subroutine ancho (
              ichar )
```

Definition at line 315 of file TCS.for.

7.26.2.2 anstr()

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

Definition at line 305 of file TCS.for.

7.26.2.3 baksp()

```
subroutine baksp
```

Definition at line 360 of file TCS.for.

7.26.2.4 cartn()

```
subroutine cartn
```

Definition at line 341 of file TCS.for.

7.26.2.5 dasha()

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

Definition at line 266 of file TCS.for.

7.26.2.6 dashr()

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

Definition at line 212 of file TCS.for.

7.26.2.7 drawa()

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

Definition at line 233 of file TCS.for.

7.26.2.8 drawr()

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

Definition at line 188 of file TCS.for.

7.26.2.9 dwindo()

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

Definition at line 438 of file TCS.for.

7.26.2.10 genflg()

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

Definition at line 534 of file TCS.for.

7.26.2.11 home()

```
subroutine home
```

Definition at line 494 of file TCS.for.

7.26.2.12 linef()

```
subroutine linef
```

Definition at line 350 of file TCS.for.

7.26.2.13 linhgt()

```
function linhgt (
              Numlin )
```

Definition at line 376 of file TCS.for.

7.26.2.14 lintrn()

```
subroutine lintrn
```

Definition at line 394 of file TCS.for.

7.26.2.15 linwdt()

```
function linwdt (
              NumChr )
```

Definition at line 384 of file TCS.for.

7.26.2.16 logtrn()

```
subroutine logtrn (
              IMODE )
```

Definition at line 404 of file TCS.for.

7.26.2.17 movea()

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

Definition at line 244 of file TCS.for.

7.26.2.18 mover()

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

Definition at line 196 of file TCS.for.

7.26.2.19 newlin()

```
subroutine newlin
```

Definition at line 333 of file TCS.for.

7.26.2.20 newpag()

```
subroutine newpag
```

Definition at line 368 of file TCS.for.

7.26.2.21 pointa()

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

Definition at line 255 of file TCS.for.

7.26.2.22 pointr()

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

Definition at line 204 of file TCS.for.

7.26.2.23 rel2ab()

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

Definition at line 220 of file TCS.for.

7.26.2.24 rescal()

```
subroutine rescal
```

Definition at line 457 of file TCS.for.

7.26.2.25 revcot()

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

Definition at line 290 of file TCS.for.

7.26.2.26 rrotat()

```
subroutine rrotat (
              Grad )
```

Definition at line 477 of file TCS.for.

7.26.2.27 rscale()

```
subroutine rscale (
              Faktor )
```

Definition at line 486 of file TCS.for.

7.26.2.28 seetrm()

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

Definition at line 512 of file TCS.for.

7.26.2.29 seetrn()

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

Definition at line 523 of file TCS.for.

7.26.2.30 setmrg()

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

Definition at line 503 of file TCS.for.

7.26.2.31 swindo()

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

Definition at line 426 of file TCS.for.

7.26.2.32 twindo()

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

Definition at line 419 of file TCS.for.

7.26.2.33 vcursr()

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

Definition at line 178 of file TCS.for.

7.26.2.34 vwindo()

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

Definition at line 445 of file TCS.for.

7.26.2.35 wincot()

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

Definition at line 277 of file TCS.for.

7.27 TCS.for

```
C> \file       TCS.for
C> \brief      TCS: Tektronix Plot 10 Emulation
C> \version    4.0
C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
C> \~german
C> Systemübergreifende TCS-Routinen
C> \~english
C> System independent subroutines
C> \~
C
C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      27.11.20 Version 4.0:
C               Einheitliche Version CPM/DOS/Windows/SDL2
C
C      17.08.20 Version 3.2
C               Harmonisierung der Verwendung des Commonblocks TKTRNX
C               Variable KHOMEY wird jetzt (analog alter DOS-Version) verwendet.
C               Da KHOMEY nicht in der CP/M Version vorhanden ist, muss ab dieser
C               Version fuer eine Complilation unter CP/M die entsprechende Zeile
C               in der SUBROUTINE HOME geändert werden.
C
C      13.11.17 Version 3.1
C               Anpassung an OpenWatcom 2.0
C               Bugfix: Unterscheidung Aufrufe ueber windowsx.h (win16) und GDI (win32)
C                - SelectPen -> SelectObject
C                - DeletePen -> DeleteObject
C                - DeleteBrush -> DeleteObject
C                - GetStockBrush -> GetStockObject
C                - DeleteRgn -> DeleteObject
C                - SelectFont -> SelectObject
C                - DeleteFont -> DeleteObject
C
C      27.03.13 Version 3.0
C               Anpassung an Windows 7 und OpenWatcom 1.9
C               Anpassung an gfortran anstelle von g77 der GCC
C
C      22.12.05 Version 2.19
C               Elimination berechnetes GOTO in LOGTRN
C
C      18.10.05 Version 2.18
C               Anpassung der Windowsversionen zur gemeinsamen Verwendung SDL2:
C                 TCSdrWIN.for
C                 TCSdWINc.h
C                 - Überfuehrung der Deklaration aus TCSdWIN.c nach *.h:
C                   GraphicError und CreateMainWindow_IfNecessary
C                 - Definition der Fehlernummern als Konstante statt enum
C               Abhaengigkeit Watcom-Defaultwindowsystem eliminiert
C               - TCSdWINc.c: Kein Abbruch bei OpenWatcom > 1.3 und
C                 definiertem Symbol trace_calls
C
C      26.10.04 Version 2.17
C               Bugfix Windows-System: Größe und Defaultposition des Status-
C                fensters wird bei der Erzeugung berechnet -> 1. RESTORE nach
C                Verkleinern des Graphikfensters entspricht dem vorherigen
C                Bild. 2. Angleichung des Verhaltens von 16- und 32bit Windows
C               Bei Definition des Symbols STAT_WINDOW_PRIVATE erhält das
C                Statusfenster einen privaten Devicekontext.
C               Zusammenfuehrung Initialisierung der Windows-Library und
C                Windows-DLL -> zusaetzliche Sourcefiles
C                TCSinitt.for, CreateMainWindow.c, GetMainInstance.c
C
C      23.06.04 Version 2.16:
C               Anpassungen an GNU-Compiler fuer Win32. Zusätzliches Sourcefile
C                fuer die GNU-Version: WinMain.c
C               CSIZE in Windows-Version: Korrektur Rundungsfehler
C
C      08.06.04 Version 2.15:
C               Umbenennung lib$movc3 in lib_movc3 (entsprechend ANSI-Fortran)
C               Modul STRINGS.FOR: Version 1.24
C
C      27.06.03 Version 2.14:
C               Verarbeitung Steuerzeichen in ANCHO
C
C      21.10.02 Version 2.13:
C               Einheitliche Version CPM/DOS/Windows
C
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
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassung an DOS:
C
C      Änderungen gegenüber CP/M-Version:
C               SEELOC, DCURSR, SVSTAT, RESTAT, CSIZE in TCSdrDOS.FOR
C      Bugfix:  DASHA, DASHR - Korrektur Parameterliste
C               SEETRM - ibaud statt ibaudr
C
C      Zugehörige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TCSdrDOS.FOR  Bildschirmtreiber
C               TCSdDOSa.ASM  Betriebssystemspezifische Low-Level Routinen
C               HDCOPY.FOR    Hardcopyroutine
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C               OUTTEXT.FOR   nur für WATCOM-Compiler
C
C      25.10.01 Version 2.00:  Dr.-Ing. K. Friedewald
C
C      07.02.02 Version 2.10:
C               Implementierung multilinguale Fehlermeldungen
C
C      11.10.02 Version 2.12:
C               Vereinheitlichung DOS/Windowsversion
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an Microsoft-Windows:
C
C      Änderungen gegenüber DOS-Version:
C               INITT befinden sich jetzt in TCSdrWIN.FOR bzw. TCSinitt.FOR
C
C      Zugehörige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TKTRNX.h      Common-Block TKTRNX für Zugriff durch C
C               TCSdrWIN.FOR  Bildschirmtreiber
C               TCSdWINc.c    Windowspezifische API-Routinen
C               TCSdWINc.h    Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C
C      27.10.01 Version 2.11: Dr.-Ing. K. Friedewald
C
C      11.10.02 Version 2.12:
C               Vereinheitlichung DOS/Windowsversion
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassungen an SDL2:
C
C      Änderungen gegenüber Windows-Version:
C               Fehlerausgabe in den Windows-Debug-Channel (bzw. *ix Fehlerkanal)
C               Statusfenster analog DOS nur einzeilig ohne Scrollmöglichkeit
C
C      Zugehörige Module:
C               TKTRNX.FOR    identisch mit Windows-Version
C               TKTRNX.h      identisch mit Windows-Version
C               TCSdrSDL.FOR  SDL2-spezifische API-Routinen
C               TCSdSDLc.c    SDL2-spezifische API-Routinen
C               TCSdSDLc.h    Compiler- und systemspezifische Deklarationen
C               STRINGS.FOR   identisch mit Windows-Version
C
C      27.11.20 Version 4.00: Dr.-Ing. K. Friedewald
C

```

```
00173
00174 C
00175 C Graphic Input
00176 C
00177
00178       subroutine vcursr (IC,X,Y)
00179       call dcursr (ic,ix,iy)
00180       call revcot (ix,iy,x,y)
00181       return
00182       end
00183
00184 C
00185 C  Virtuelle Graphik, relativ
00186 C
00187
00188       subroutine drawr (X,Y)
00189       call rel2ab (x,y,xabs,yabs)
00190       call drawa (xabs,yabs)
00191       return
00192       end
00193
00194
00195
00196       subroutine mover (X,Y)
00197       call rel2ab (x,y,xabs,yabs)
00198       call movea (xabs,yabs)
00199       return
00200       end
00201
00202
00203
00204       subroutine pointr (X,Y)
00205       call rel2ab (x,y,xabs,yabs)
00206       call pointa (xabs,yabs)
00207       return
00208       end
00209
00210
00211
00212       subroutine dashr (X,Y, iL)
00213       call rel2ab (x,y,xabs,yabs)
00214       call dasha (xabs,yabs, il)
00215       return
00216       end
00217
00218
00219
00220       subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
00221       include 'Tktrnx.fd'
00222       call seeloc (ix,iy)
00223       call revcot (ix,iy,xabs,yabs)
00224       xabs= (( xrel*trcosf - yrel*trsinf)*trscal)+xabs
00225       yabs= (( xrel*trsinf + yrel*trcosf)*trscal)+yabs
00226       return
00227       end
00228
00229 C
00230 C  Virtuelles Zeichnen, absolut
00231 C
00232
00233       subroutine drawa (X,Y)
00234       include 'Tktrnx.fd'
00235       call wincot (x,y,ix,iy)
00236       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00237       call drwabs (ix,iy)
00238       call swind1 (0,0,1023,780)
00239       return
00240       end
00241
00242
00243
00244       subroutine movea (X,Y)
00245       include 'Tktrnx.fd'
00246       call wincot (x,y,ix,iy)
00247       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00248       call movabs (ix,iy)
00249       call swind1 (0,0,1023,780)
00250       return
00251       end
00252
00253
00254
00255       subroutine pointa (X,Y)
00256       include 'Tktrnx.fd'
00257       call wincot (x,y,ix,iy)
00258       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00259       call pntabs (ix,iy)
```

```
00260       call swind1 (0,0,1023,780)
00261       return
00262       end
00263
00264
00265
00266       subroutine dasha (X,Y, iL)
00267       include 'Tktrnx.fd'
00268       call wincot (x,y,ix,iy)
00269       call swind1 (kminsx,kminsy,kmaxsx,kmaxsy)
00270       call dshabs (ix,iy, il)
00271       call swind1 (0,0,1023,780)
00272       return
00273       end
00274
00275
00276
00277       subroutine wincot (X,Y,IX,IY)
00278       include 'Tktrnx.fd'
00279       dx= x-tminvx
00280       dy= y-tminvy
00281       if ((xlog.lt.255.).and.(x.gt.0.)) dx= alog(x)-xlog
00282       if ((ylog.lt.255.).and.(y.gt.0.)) dy= alog(y)-ylog
00283       ix= ifix(dx*xfac+.5)+kminsx
00284       iy= ifix(dy*yfac+.5)+kminsy
00285       return
00286       end
00287
00288
00289
00290       subroutine revcot (IX,IY,X,Y)
00291       include 'Tktrnx.fd'
00292       dx= float(ix-kminsx) / xfac
00293       dy= float(iy-kminsy) / yfac
00294       x= dx + tminvx
00295       y= dy + tminvy
00296       if (xlog.lt.255.) x= 2.718282**(dx+xlog)
00297       if (ylog.lt.255.) y= 2.718282**(dy+ylog)
00298       return
00299       end
00300
00301 C
00302 C  Alphanumerische Ausgabe
00303 C
00304
00305       subroutine anstr (NChar, IStrin)
00306       dimension istrin(1)
00307       do 10 i=1,nchar
00308        call ancho (istrin(i))
00309 10    continue
00310       return
00311       end
00312
00313
00314
00315       subroutine ancho (ichar)
00316       include 'Tktrnx.fd'
00317
00318       if (ichar.gt.31) goto 10
00319       if (ichar.eq.7) call bell
00320       if (ichar.eq.10) call linef
00321       if (ichar.eq.13) call cartn
00322       return
00323
00324  10   call seeloc (ix,k)
00325       call csize (ixlen,k)
00326       if (ix.gt.krmrgn-ixlen) call newlin
00327       call toutpt (ichar)
00328       return
00329       end
00330
00331
00332
00333       subroutine newlin
00334       call cartn
00335       call linef
00336       return
00337       end
00338
00339
00340
00341       subroutine cartn
00342       include 'Tktrnx.fd'
00343       call seeloc (ix,iy)
00344       call movabs (klmrgn,iy)
00345       return
00346       end
```

```
00347
00348
00349
00350       subroutine linef
00351       call seeloc (j,iy)
00352       call csize (j,iylen)
00353       if (iy.lt.iylen) call home
00354       call movrel (0,-iylen)
00355       return
00356       end
00357
00358
00359
00360       subroutine baksp
00361       call csize (ix,iy)
00362       call movrel (-ix,0)
00363       return
00364       end
00365
00366
00367
00368       subroutine newpag
00369       call erase
00370       call home
00371       return
00372       end
00373
00374
00375
00376       function linhgt (Numlin)
00377       call csize (ix,iy)
00378       linhgt= numlin*iy
00379       return
00380       end
00381
00382
00383
00384       function linwdt (NumChr)
00385       call csize (ix,iy)
00386       linwdt= numchr*ix
00387       return
00388       end
00389
00390 C
00391 C  Initialisierungsroutinen
00392 C
00393
00394       subroutine lintrn
00395       include 'Tktrnx.fd'
00396       xlog= 255.
00397       ylog= 255.
00398       call rescal
00399       return
00400       end
00401
00402
00403
00404       subroutine logtrn (IMODE)
00405       include 'Tktrnx.fd'
00406       call lintrn
00407       if ((imode .eq. 1) .or. (imode .eq. 3)) then
00408        xlog= 0.
00409       end if
00410       if ((imode .eq. 2) .or. (imode .eq. 3)) then
00411        ylog= 0.
00412       end if
00413       call rescal
00414       return
00415       end
00416
00417
00418
00419       subroutine twindo (IX1,IX2,IY1,IY2)
00420       call swindo (ix1,ix2-ix1,iy1,iy2-iy1)
00421       return
00422       end
00423
00424
00425
00426       subroutine swindo (IX,LX,IY,LY)
00427       include 'Tktrnx.fd'
00428       kminsx= ix
00429       kmaxsx= ix+lx
00430       kminsy= iy
00431       kmaxsy= iy+ly
00432       call rescal
00433       return
```

```
00434       end
00435
00436
00437
00438       subroutine dwindo (X1,X2,Y1,Y2)
00439       call vwindo (x1,x2-x1,y1,y2-y1)
00440       return
00441       end
00442
00443
00444
00445       subroutine vwindo (X,XL,Y,YL)
00446       include 'Tktrnx.fd'
00447       tminvx= x
00448       tmaxvx= x+xl
00449       tminvy= y
00450       tmaxvy= y+yl
00451       call rescal
00452       return
00453       end
00454
00455
00456
00457       subroutine rescal
00458       include 'Tktrnx.fd'
00459       xfac= 0.
00460       yfac= 0.
00461       if ((tmaxvx.eq.tminvx) .or. (tmaxvy.eq.tminvy)) return
00462       dx= tmaxvx-tminvx
00463       dy= tmaxvy-tminvy
00464       if ((xlog.eq.255.).or.(amin1(tminvx,tmaxvx).le.0.)) goto 10
00465        xlog= alog(tminvx)
00466        dx= alog(tmaxvx)-xlog
00467 10    if ((ylog.eq.255.).or.(amin1(tminvy,tmaxvy).le.0.)) goto 20
00468        ylog= alog(tminvy)
00469        dy= alog(tmaxvy)-ylog
00470 20    xfac= float(kmaxsx-kminsx) / dx
00471       yfac= float(kmaxsy-kminsy) / dy
00472       return
00473       end
00474
00475
00476
00477       subroutine rrotat (Grad)
00478       include 'Tktrnx.fd'
00479       trsinf= sin(grad/57.29578)
00480       trcosf= cos(grad/57.29578)
00481       return
00482       end
00483
00484
00485
00486       subroutine rscale (Faktor)
00487       include 'Tktrnx.fd'
00488       trscal= faktor
00489       return
00490       end
00491
00492
00493
00494       subroutine home
00495       include 'Tktrnx.fd'
00496 C      call movabs(klmrgn,750) Fuer CP/M (kein khomey verfuegbar, -> !=750)
00497       call movabs(klmrgn,khomey)
00498       return
00499       end
00500
00501
00502
00503       subroutine setmrg (Mlinks, Mrecht)
00504       include 'Tktrnx.fd'
00505       klmrgn= mlinks
00506       krmrgn= mrecht
00507       return
00508       end
00509
00510
00511
00512       subroutine seetrm (IBaud,Iterm,ICSize,MaxScr)
00513       include 'Tktrnx.fd'
00514       ibaud= 0
00515       iterm= 1
00516       icsize= 1
00517       maxscr= 1023
00518       return
00519       end
00520
```

```
00521
00522
00523       subroutine seetrn (xf,yf,key)
00524       include 'Tktrnx.fd'
00525       xf= xfac
00526       yf= yfac
00527       key= 1
00528       if ((xlog.lt.255.).or.(ylog.lt.255.)) key=2
00529       return
00530       end
00531
00532
00533
00534       logical function genflg (ITEM)
00535       genflg= item.eq.0
00536       return
00537       end
00538
```

7.28 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.

7.28.1 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.

7.28.2 Function/Subroutine Documentation

7.28.2.1 anmode()

```
subroutine anmode
```

Entry Dummyroutinen.

AlfMod

pClipt

alpha

Definition at line 219 of file TCSdrSDL.for.

7.28.2.2 drwrel()

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

Definition at line 132 of file TCSdrSDL.for.

7.28.2.3 dshrel()

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

Definition at line 142 of file TCSdrSDL.for.

7.28.2.4 initt()

```
subroutine initt (
              iDummy )
```

Initialisierung Hard- und Software.

Definition at line 50 of file TCSdrSDL.for.

7.28.2.5 initt2()

```
subroutine initt2
```

Definition at line 62 of file TCSdrSDL.for.

7.28.2.6 movrel()

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

Definition at line 112 of file TCSdrSDL.for.

7.28.2.7 pntrel()

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

Definition at line 122 of file TCSdrSDL.for.

7.28.2.8 restat()

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

Definition at line 94 of file TCSdrSDL.for.

7.28.2.9 seeloc()

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

Definition at line 156 of file TCSdrSDL.for.

7.28.2.10 statst()

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

Definition at line 196 of file TCSdrSDL.for.

7.28.2.11 svstat()

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

Definition at line 81 of file TCSdrSDL.for.

7.28.2.12 tcslev()

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

Definition at line 37 of file TCSdrSDL.for.

7.28.2.13 tinput()

```
subroutine tinput (
              iChr )
```

Definition at line 208 of file TCSdrSDL.for.

7.28.2.14 toutpt()

```
subroutine toutpt (
              iChr )
```

Definition at line 169 of file TCSdrSDL.for.

7.28.2.15 toutst()

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

Definition at line 177 of file TCSdrSDL.for.

7.28.2.16 toutstc()

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

Definition at line 188 of file TCSdrSDL.for.

7.29 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
```

7.30 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 JOURNALTYP 3
- #define XMLSUPPORT
- #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 TCSdrWIN__ winlbl (FTNSTRPAR *PloWinNam, FTNSTRPAR *StatWinNam, FTNSTRPAR *IniFilNam FTNSTRPAR_TAIL(IniFilNam))
- void initt1 ()
- void finitt ()
- void iowait (void)
- void TCSdrWIN__ swind1 (FTNINT *ix1, FTNINT *iy1, FTNINT *ix2, FTNINT *iy2)
- void TCSdrWIN__ erase (void)
- void TCSdrWIN__ movabs (FTNINT *ix, FTNINT *iy)
- void TCSdrWIN__ drwabs (FTNINT *ix, FTNINT *iy)
- void TCSdrWIN__ dshabs (FTNINT *ix, FTNINT *iy, FTNINT *iMask)
- void TCSdrWIN__ pntabs (FTNINT *ix, FTNINT *iy)
- void TCSdrWIN__ bckcol (FTNINT *iCol)
- void TCSdrWIN__ lincol (FTNINT *iCol)
- void TCSdrWIN__ txtcol (FTNINT *iCol)
- void TCSdrWIN__ DefaultColour (void)
- void TCSdrWIN__ outgtext (FTNSTRPAR *ftn_string FTNSTRPAR_TAIL(ftn_string))
- void TCSdrWIN__ italic (void)
- void TCSdrWIN__ italir (void)
- void TCSdrWIN__ dblsiz (void)
- void TCSdrWIN__ nrmsiz (void)
- void TCSdrWIN__ csize (FTNINT *ix, FTNINT *iy)
- void TCSdrWIN__ outtext (FTNSTRPAR *ftn_string FTNSTRPAR_TAIL(ftn_string))
- void TCSdrWIN__ bell (void)
- void TCSdrWIN__ GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string, FTNINT *iL FTNSTRPAR_TAIL(ftn↩ _string))
- void TCSdrWIN__ dcursr (FTNINT *ic, FTNINT *ix, FTNINT *iy)
- void TCSdrWIN__ 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

7.30.1 Detailed Description

SDL Port: Low-Level Driver.

Version

1.3

Author

(C) 2022 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.

7.30.2 Macro Definition Documentation

7.30.2.1 AUDIOSUPPORT

```
#define AUDIOSUPPORT
```

Definition at line 69 of file TCSdSDLc.c.

7.30.2.2 FNTFILEXT

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

Definition at line 66 of file TCSdSDLc.c.

7.30.2.3 HIGHQUALCHAR

```
#define HIGHQUALCHAR
```

Definition at line 70 of file TCSdSDLc.c.

7.30.2.4 INIFILEXT

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

Definition at line 65 of file TCSdSDLc.c.

7.30.2.5 JOURNALTYP

```
#define JOURNALTYP 3
```

Definition at line 67 of file TCSdSDLc.c.

7.30.2.6 LOGLEVEL

```
#define LOGLEVEL SDL_LOG_PRIORITY_ERROR
```

Definition at line 77 of file TCSdSDLc.c.

7.30.2.7 MAX_COLOR_INDEX

```
#define MAX_COLOR_INDEX 15
```
Definition at line 232 of file TCSdSDLc.c.

7.30.2.8 TMPSTRLEN

```
#define TMPSTRLEN TCS_FILE_NAMELEN
```

7.30.2.9 XMLSUPPORT

```
#define XMLSUPPORT
```
Definition at line 68 of file TCSdSDLc.c.

7.30.3 Typedef Documentation

7.30.3.1 ErrMsg

```
typedef char ErrMsg[TCS_MESSAGELEN]
```
Definition at line 153 of file TCSdSDLc.c.

7.30.4 Function Documentation

7.30.4.1 audio_callback()

```
void audio_callback (
            void * sample_nr,
            Uint8 * raw_buffer,
            int bytes )
```
Definition at line 736 of file TCSdSDLc.c.

7.30.4.2 bckcol()

```
void TCSdrWIN__ bckcol (
            FTNINT * iCol )
```
Definition at line 1772 of file TCSdSDLc.c.

7.30.4.3 bell()

```
void TCSdrWIN__ bell (
            void  )
```
Definition at line 2089 of file TCSdSDLc.c.

7.30.4.4 ClipLineStart()

```
bool ClipLineStart (
            FTNINT ix1,
            FTNINT iy1,
            FTNINT ix2,
            FTNINT iy2,
```

```
            FTNINT * isx,
            FTNINT * isy )
```

Definition at line 301 of file TCSdSDLc.c.

7.30.4.5 csize()

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

Definition at line 2031 of file TCSdSDLc.c.

7.30.4.6 CustomizeProgPar()

```
void CustomizeProgPar ( )
```

Definition at line 1134 of file TCSdSDLc.c.

7.30.4.7 dblsiz()

```
void TCSdrWIN__ dblsiz (
            void  )
```

Definition at line 1958 of file TCSdSDLc.c.

7.30.4.8 dcursr()

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

Definition at line 2116 of file TCSdSDLc.c.

7.30.4.9 DefaultColour()

```
void TCSdrWIN__ DefaultColour (
            void  )
```

Definition at line 1839 of file TCSdSDLc.c.

7.30.4.10 DrawHiResDashLine()

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

Definition at line 368 of file TCSdSDLc.c.

7.30.4.11 drwabs()

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

Definition at line 1639 of file TCSdSDLc.c.

7.30.4.12 dshabs()

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

Definition at line 1684 of file TCSdSDLc.c.

7.30.4.13 erase()

```
void TCSdrWIN__ erase (
            void  )
```

Definition at line 1558 of file TCSdSDLc.c.

7.30.4.14 finitt()

```
void finitt ( )
```

Definition at line 1491 of file TCSdSDLc.c.

7.30.4.15 GraphicError()

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

Definition at line 2101 of file TCSdSDLc.c.

7.30.4.16 hdcopy()

```
void TCSdrWIN__ hdcopy (
            void  )
```

Definition at line 2160 of file TCSdSDLc.c.

7.30.4.17 HiResX()

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

Definition at line 266 of file TCSdSDLc.c.

7.30.4.18 HiResY()

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

Definition at line 272 of file TCSdSDLc.c.

7.30.4.19 initt1()

```
void initt1 ( )
```

Definition at line 1281 of file TCSdSDLc.c.

7.30.4.20 iowait()

```
void iowait (
            void  )
```

Definition at line 1535 of file TCSdSDLc.c.

7.30.4.21 italic()

```
void TCSdrWIN__ italic (
            void  )
```

Definition at line 1916 of file TCSdSDLc.c.

7.30.4.22 italir()

```
void TCSdrWIN__ italir (
            void  )
```

Definition at line 1937 of file TCSdSDLc.c.

7.30.4.23 lib_movc3()

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

Definition at line 2293 of file TCSdSDLc.c.

7.30.4.24 lincol()

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

Definition at line 1794 of file TCSdSDLc.c.

7.30.4.25 LoResX()

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

Definition at line 278 of file TCSdSDLc.c.

7.30.4.26 LoResY()

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

Definition at line 284 of file TCSdSDLc.c.

7.30.4.27 movabs()

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

Definition at line 1616 of file TCSdSDLc.c.

7.30.4.28 nrmsiz()

```
void TCSdrWIN__ nrmsiz (
            void  )
```

Definition at line 1993 of file TCSdSDLc.c.

7.30.4.29 outgtext()

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

Definition at line 1858 of file TCSdSDLc.c.

7.30.4.30 outtext()

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

Definition at line 2039 of file TCSdSDLc.c.

7.30.4.31 PlotText()

```
void PlotText (
            const char * outtxt )
```

Definition at line 425 of file TCSdSDLc.c.

7.30.4.32 pntabs()

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

Definition at line 1739 of file TCSdSDLc.c.

7.30.4.33 PointInWindow()

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

Definition at line 293 of file TCSdSDLc.c.

7.30.4.34 PresetProgPar()

```
void PresetProgPar ( )
```

Definition at line 1106 of file TCSdSDLc.c.

7.30.4.35 RepaintBuffer()

```
void RepaintBuffer ( )
```

Definition at line 452 of file TCSdSDLc.c.

7.30.4.36 sax_callback()

```
void sax_callback (
            mxml_node_t * node,
```

```
            mxml_sax_event_t event,
            void * usr )
```

Definition at line 767 of file TCSdSDLc.c.

7.30.4.37 sax_error_callback()

```
void sax_error_callback (
            char * mssg )
```

Definition at line 1061 of file TCSdSDLc.c.

7.30.4.38 sax_type_callback()

```
mxml_type_t sax_type_callback (
            mxml_node_t * node )
```

Definition at line 1041 of file TCSdSDLc.c.

7.30.4.39 swind1()

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

Definition at line 1549 of file TCSdSDLc.c.

7.30.4.40 TCSEventFilter()

```
int TCSEventFilter (
            void * UserData,
            SDL_Event * event )
```

Definition at line 700 of file TCSdSDLc.c.

7.30.4.41 TCSGraphicError()

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

Definition at line 648 of file TCSdSDLc.c.

7.30.4.42 txtcol()

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

Definition at line 1817 of file TCSdSDLc.c.

7.30.4.43 winlbl()

```
void TCSdrWIN__ winlbl (
            FTNSTRPAR * PloWinNam,
            FTNSTRPAR * StatWinNam,
            FTNSTRPAR *IniFilNam  FTNSTRPAR_TAILIniFilNam )
```

Definition at line 1185 of file TCSdSDLc.c.

7.30.4.44 XMLreadProgPar()

```
void XMLreadProgPar (
            const char * filname )
```

Definition at line 1079 of file TCSdSDLc.c.

7.30.5 Variable Documentation

7.30.5.1 AudioSample_nr

```
int AudioSample_nr = 0  [static]
```

Definition at line 254 of file TCSdSDLc.c.

7.30.5.2 ClippingNotActive

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

Definition at line 123 of file TCSdSDLc.c.

7.30.5.3 iHardcopyCount

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

Definition at line 145 of file TCSdSDLc.c.

7.30.5.4 PixFacX

```
float PixFacX  [static]
```

Definition at line 120 of file TCSdSDLc.c.

7.30.5.5 PixFacY

```
float PixFacY  [static]
```

Definition at line 120 of file TCSdSDLc.c.

7.30.5.6 SDL_AudioDev_optained

```
SDL_AudioSpec SDL_AudioDev_optained  [static]
```

Definition at line 251 of file TCSdSDLc.c.

7.30.5.7 SDL_AudioDev_wanted

```
SDL_AudioSpec SDL_AudioDev_wanted  [static]
```

Definition at line 252 of file TCSdSDLc.c.

7.30.5.8 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 214 of file TCSdSDLc.c.

7.30.5.9 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 154 of file TCSdSDLc.c.

7.30.5.10 szTCSGraphicFont

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

Definition at line 129 of file TCSdSDLc.c.

7.30.5.11 szTCSHardcopyFile

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

Definition at line 128 of file TCSdSDLc.c.

7.30.5.12 szTCSIniFile

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

Definition at line 127 of file TCSdSDLc.c.

7.30.5.13 szTCSsect0

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

Definition at line 131 of file TCSdSDLc.c.

7.30.5.14 szTCSstatWindowName

```
char szTCSstatWindowName[TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME  [static]
```
Definition at line 126 of file TCSdSDLc.c.

7.30.5.15 szTCSSysFont

```
char szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT  [static]
```
Definition at line 130 of file TCSdSDLc.c.

7.30.5.16 szTCSWindowName

```
char szTCSWindowName[TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME  [static]
```
Definition at line 125 of file TCSdSDLc.c.

7.30.5.17 TCSDefaultBckCol

```
int TCSDefaultBckCol = TCS_INIDEF_BCKCOL  [static]
```
Definition at line 144 of file TCSdSDLc.c.

7.30.5.18 TCSDefaultLinCol

```
int TCSDefaultLinCol = TCS_INIDEF_LINCOL  [static]
```
Definition at line 142 of file TCSdSDLc.c.

7.30.5.19 TCSDefaultTxtCol

```
int TCSDefaultTxtCol = TCS_INIDEF_TXTCOL  [static]
```
Definition at line 143 of file TCSdSDLc.c.

7.30.5.20 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 181 of file TCSdSDLc.c.

7.30.5.21 TCSEventFilterData

```
int TCSEventFilterData  [static]
```
Definition at line 118 of file TCSdSDLc.c.

7.30.5.22 TCSfont

```
TTF_Font* TCSfont = NULL  [static]
```
Definition at line 237 of file TCSdSDLc.c.

7.30.5.23 TCSinitialized

```
bool TCSinitialized = false  [static]
```
Definition at line 122 of file TCSdSDLc.c.

7.30.5.24 TCSrenderer

```
SDL_Renderer* TCSrenderer = NULL  [static]
```
Definition at line 236 of file TCSdSDLc.c.

7.30.5.25 TCSstatrenderer

```
SDL_Renderer* TCSstatrenderer = NULL  [static]
```
Definition at line 241 of file TCSdSDLc.c.

7.30.5.26 TCSstatusfont

```
TTF_Font* TCSstatusfont = NULL  [static]
```
Definition at line 238 of file TCSdSDLc.c.

7.30.5.27 TCSstatwindow

```
SDL_Window* TCSstatwindow = NULL  [static]
```
Definition at line 240 of file TCSdSDLc.c.

7.30.5.28 TCSstatWindowIniXrelpos

```
int TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX  [static]
```
Definition at line 137 of file TCSdSDLc.c.

7.30.5.29 TCSstatWindowIniXrelsiz

```
int TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX  [static]
```
Definition at line 139 of file TCSdSDLc.c.

7.30.5.30 TCSstatWindowIniYrelpos

```
int TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY  [static]
```
Definition at line 138 of file TCSdSDLc.c.

7.30.5.31 TCSstatWindowIniYrelsiz

```
int TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY  [static]
```
Definition at line 140 of file TCSdSDLc.c.

7.30.5.32 TCSwindow

```
SDL_Window* TCSwindow = NULL  [static]
```
Definition at line 235 of file TCSdSDLc.c.

7.30.5.33 TCSwindowIniXrelpos

```
int TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX  [static]
```
Definition at line 133 of file TCSdSDLc.c.

7.30.5.34 TCSwindowIniXrelsiz

```
int TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX  [static]
```
Definition at line 135 of file TCSdSDLc.c.

7.30.5.35 TCSwindowIniYrelpos

```
int TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY  [static]
```
Definition at line 134 of file TCSdSDLc.c.

7.30.5.36 TCSwindowIniYrelsiz

```
int TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY  [static]
```
Definition at line 136 of file TCSdSDLc.c.

7.30.5.37 TextLineHeight

```
int TextLineHeight  [static]
```
Definition at line 141 of file TCSdSDLc.c.

7.30.5.38 xTCSJournal

```
struct xJournalEntry_typ* xTCSJournal = NULL  [static]
```
Definition at line 247 of file TCSdSDLc.c.

7.31 TCSdSDLc.c

```
00001 /** ****************************************************************************
00002 \file       TCSdSDLc.c
00003 \brief      SDL Port: Low-Level Driver
00004 \version    1.3
00005 \author     (C) 2022 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 JOURNALTYP 3
00068 #define XMLSUPPORT
00069 #define AUDIOSUPPORT
00070 #define HIGHQUALCHAR
00071
00072
00073 /*
00074 ------------------- Debug Switches ---------------------------------------------
00075 */
00076
00077 #define LOGLEVEL   SDL_LOG_PRIORITY_ERROR
00078 // #define LOGLEVEL   SDL_LOG_PRIORITY_DEBUG
00079 // #define LOGLEVEL   SDL_LOG_PRIORITY_VERBOSE // Ausgaben < Error in Fehlerkanal
00080 // #define TRACE_CALLS // zusaetzliche Debugausgaben
00081
00082
00083 /*
00084 ------------------- Headerfiles ------------------------------------------------
00085 */
00086
00087 #include <stdlib.h>
00088 #include <string.h>
00089 #include <stdio.h> // Fuer HDCOPY: sprintf
00090
00091 #ifdef AUDIOSUPPORT
00092  #include <math.h>
00093 #endif
00094
00095 #include "SDL.h"
00096 #include "SDL_ttf.h"
00097
00098 #ifdef AUDIOSUPPORT
00099  #include "SDL_audio.h"
00100 #endif
00101
00102 #ifdef XMLSUPPORT
```

```
00103  #include  "mxml.h"
00104 #endif
00105
00106 #if (JOURNALTYP == 3)
00107  #include "sglib.h"
00108 #endif
00109
00110 #include "TCSdSDLc.h"
00111 #include "TKTRNX.h"
00112
00113
00114 /*
00115 ------------------- Globale Variablen ----------------------------------------
00116 */
00117
00118 static int      TCSEventFilterData; // Userdata, z.Zt. nicht verwendet
00119
00120 static  float   PixFacX, PixFacY; // Anpassung Bildschirmauflösung
00121
00122 static  bool    TCSinitialized = false,
00123                 ClippingNotActive = true;
00124
00125 static char     szTCSWindowName[TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME,
00126                 szTCSstatWindowName[TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME,
00127                 szTCSIniFile[TCS_FILE_NAMELEN] = "",
00128                 szTCSHardcopyFile[TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME,
00129                 szTCSGraphicFont[TCS_FILE_NAMELEN] = TCS_INIDEF_FONT,
00130                 szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT,
00131                 szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0;
00132
00133 static  int     TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX, // rel. Bildschirmpos.
00134                 TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY, // bei Init in %
00135                 TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX,
00136                 TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY,
00137                 TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX, // dito
00138                 TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY, // Statusfenster
00139                 TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX,
00140                 TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY,
00141                 TextLineHeight,
00142                 TCSDefaultLinCol = TCS_INIDEF_LINCOL,
00143                 TCSDefaultTxtCol = TCS_INIDEF_TXTCOL,
00144                 TCSDefaultBckCol = TCS_INIDEF_BCKCOL,
00145                 iHardcopyCount = 1;  // Zähler zur Erzeugung Filenamen
00146
00147
00148
00149 /*
00150   Zuordnung Fehlernummern zu Meldungen
00151 */
00152
00153 typedef char   ErrMsg[TCS_MESSAGELEN];
00154 static  ErrMsg  szTCSErrorMsg[(int) MSG_MAXERRNO+1] =
00155                 {"Element 0 unused","DOS",
00156                 TCS_INIDEF_UNKNGRAPHCARD, // Errno 2
00157                 TCS_INIDEF_NOFNTFIL,      // Errno 3
00158                 TCS_INIDEF_NOFNT,         // Errno 4
00159                 "DOS",
00160                 TCS_INIDEF_HDCOPN,        // Errno 6
00161                 TCS_INIDEF_HDCWRT,        // Errno 7
00162                 TCS_INIDEF_HDCINT,        // Errno 8
00163                 TCS_INIDEF_USR,           // Errno 9
00164                 TCS_INIDEF_HDCACT,        // Errno 10
00165                 TCS_INIDEF_USRWRN,        // Errno 11
00166                 TCS_INIDEF_EXIT,          // Errno 12
00167                 "Windows",
00168                 "Windows",
00169                 TCS_INIDEF_JOUCREATE,     // Errno 15
00170                 TCS_INIDEF_JOUENTRY,      // Errno 16
00171                 TCS_INIDEF_JOUADD,        // Errno 17
00172                 TCS_INIDEF_JOUCLR,        // Errno 18
00173                 TCS_INIDEF_JOUUNKWN,      // Errno 19
00174                 TCS_INIDEF_XMLPARSER,     // Errno 20
00175                 TCS_INIDEF_XMLOPEN,       // Errno 21
00176                 TCS_INIDEF_UNKNAUDIO,     // Errno 22
00177                 TCS_INIDEF_USR2,          // Errno 23
00178                 TCS_INIDEF_INI2,          // Errno 24
00179                 "Maxerr only for internal Use" };
00180
00181 static  int     TCSErrorLev[(int) MSG_MAXERRNO+1] =
00182                 {10,10,
00183                 TCS_INIDEF_UNKNGRAPHCARDL,// Errno 2
00184                 TCS_INIDEF_NOFNTFILL,     // Errno 3
00185                 TCS_INIDEF_NOFNTL,        // Errno 4
00186                 10,
00187                 TCS_INIDEF_HDCOPNL,       // Errno 6
00188                 TCS_INIDEF_HDCWRTL,       // Errno 7
00189                 TCS_INIDEF_HDCINTL,       // Errno 8
```

```
00190                 TCS_INIDEF_USRL,          // Errno 9
00191                 TCS_INIDEF_HDCACTL,       // Errno 10
00192                 TCS_INIDEF_USRWRNL,       // Errno 11
00193                 TCS_INIDEF_EXITL,         // Errno 12
00194                 10,
00195                 10,
00196                 TCS_INIDEF_JOUCREATEL,    // Errno 15
00197                 TCS_INIDEF_JOUENTRYL,     // Errno 16
00198                 TCS_INIDEF_JOUADDL,       // Errno 17
00199                 TCS_INIDEF_JOUCLRL,       // Errno 18
00200                 TCS_INIDEF_JOUUNKWNL,     // Errno 19
00201                 TCS_INIDEF_XMLPARSERL,    // Errno 20
00202                 TCS_INIDEF_XMLOPENL,      // Errno 21
00203                 TCS_INIDEF_UNKNAUDIOL,    // Errno 22
00204                 TCS_INIDEF_USR2L,         // Errno 23
00205                 TCS_INIDEF_INI2L,         // Errno 24
00206                 10};
00207
00208
00209
00210 /*
00211   Zuordnung der Farbennummern zur VGA-Palette
00212 */
00213
00214 static  SDL_Color sdlColorTable[] = {
00215                 {240,240,240,SDL_ALPHA_OPAQUE }, /* iCol= 00: weiss (DOS: 01) */
00216                 {  0,  0,  0,SDL_ALPHA_OPAQUE }, /* iCol= 01: schwarz(DOS:00) */
00217                 {240, 80, 80,SDL_ALPHA_OPAQUE }, /* iCol= 02: rot             */
00218                 { 80,240, 80,SDL_ALPHA_OPAQUE }, /* iCol= 03: gruen           */
00219                 { 80,240,240,SDL_ALPHA_OPAQUE }, /* iCol= 04: blau            */
00220                 { 80, 80,240,SDL_ALPHA_OPAQUE }, /* iCol= 05: lila            */
00221                 {240,240, 80,SDL_ALPHA_OPAQUE }, /* iCol= 06: gelb            */
00222                 {160,160,160,SDL_ALPHA_OPAQUE }, /* iCol= 07: grau            */
00223                 {240, 80,240,SDL_ALPHA_OPAQUE }, /* iCol= 08: violett         */
00224                 {160,  0,  0,SDL_ALPHA_OPAQUE }, /* iCol= 09: mattrot         */
00225                 {  0,160,  0,SDL_ALPHA_OPAQUE }, /* iCol= 10: mattgruen       */
00226                 {  0,  0,160,SDL_ALPHA_OPAQUE }, /* iCol= 11: mattblau        */
00227                 {  0,160,160,SDL_ALPHA_OPAQUE }, /* iCol= 12: mattlila        */
00228                 {160, 80,  0,SDL_ALPHA_OPAQUE }, /* iCol= 13: orange          */
00229                 { 80, 80, 80,SDL_ALPHA_OPAQUE }, /* iCol= 14: mattgrau        */
00230                 {160,  0,160,SDL_ALPHA_OPAQUE }  /* iCol= 15: mattviolett     */
00231               };
00232 #define MAX_COLOR_INDEX 15
00233
00234
00235 static  SDL_Window *TCSwindow = NULL;
00236 static  SDL_Renderer *TCSrenderer = NULL;
00237 static  TTF_Font* TCSfont = NULL;
00238 static  TTF_Font* TCSstatusfont = NULL;
00239
00240 static  SDL_Window *TCSstatwindow = NULL;
00241 static  SDL_Renderer *TCSstatrenderer = NULL;
00242
00243 #if (JOURNALTYP == 3)
00244  struct xJournalEntry_typ {struct xJournalEntry_typ * previous;
00245                            struct xJournalEntry_typ * next;
00246                            FTNINT action; FTNINT i1; FTNINT i2;};
00247  static struct xJournalEntry_typ* xTCSJournal = NULL;
00248 #endif
00249
00250 #ifdef AUDIOSUPPORT
00251  static SDL_AudioSpec      SDL_AudioDev_optained;
00252  static SDL_AudioSpec      SDL_AudioDev_wanted;
00253
00254  static int                AudioSample_nr = 0;
00255 #endif
00256
00257
00258
00259
00260
00261 // ---------------- interne Unterprogramme -------------------------------------
00262
00263
00264 /* --- Anpassung der Zeichenaufloesung an die Bildschirme --- */
00265
00266 int HiResX(FTNINT iX)
00267 {
00268     return (PixFacX*iX) +0.25f;
00269 }
00270
00271
00272 int HiResY(FTNINT iY)
00273 {
00274     return (PixFacY*iY)+0.25f;
00275 }
00276
```

```
00277
00278 int LoResX(FTNINT iX)
00279 {
00280     return (int)( ( (float)iX/PixFacX) +0.25f );
00281 }
00282
00283
00284 int LoResY(FTNINT iY)
00285 {
00286     return (int)( ((float)iY/PixFacY)+0.25f );
00287 }
00288
00289
00290
00291 /* --- Clippingroutinen --- */
00292
00293 bool PointInWindow (FTNINT ix1, FTNINT iy1)
00294 {
00295     if (ClippingNotActive ) return true;
00296     return ( (TKTRNX.kminsx <= ix1) && (TKTRNX.kmaxsx >= ix1) &&
00297                     (TKTRNX.kminsy <= iy1) && (TKTRNX.kmaxsy >= iy1));
00298 }
00299
00300
00301 bool ClipLineStart (FTNINT ix1, FTNINT iy1, FTNINT ix2, FTNINT iy2,
00302                                                 FTNINT *isx, FTNINT *isy)
00303 /* ClipLineStart=true: isx,isy Startpunkt; =false: Linie nicht zeichnen */
00304 {
00305     if (ClippingNotActive) {
00306      *isx= ix1; *isy= iy1;
00307      return true;
00308     }
00309
00310     if (ix1 < TKTRNX.kminsx) { /* Start links vom Fenster */
00311      if (ix2 < TKTRNX.kminsx) return false;
00312      *isy= iy1+((TKTRNX.kminsx-ix1) * (iy2-iy1)) / (ix2-ix1);
00313      if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
00314       *isx= TKTRNX.kminsx;
00315       return true;
00316      }
00317      if (iy1 == iy2) return false;
00318      if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
00319       *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00320       *isy= TKTRNX.kminsy;
00321      } else {
00322       *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00323       *isy= TKTRNX.kmaxsy;
00324      }
00325      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00326      return true;
00327
00328     } else if (ix1 > TKTRNX.kmaxsx) { /* Start rechts vom Fenster */
00329      if (ix2 > TKTRNX.kmaxsx) return false;
00330      *isy= iy1+((TKTRNX.kmaxsx-ix1) * (iy2-iy1)) / (ix2-ix1);
00331      if ((TKTRNX.kminsy <= *isy) && (TKTRNX.kmaxsy >= *isy)) {
00332       *isx= TKTRNX.kmaxsx;
00333       return true;
00334      }
00335      if (iy1 == iy2) return false;
00336      if (((ix2-ix1)*(iy2-iy1)) >= 0) { /* Steigung positiv */
00337       *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00338       *isy= TKTRNX.kmaxsy;
00339      } else {
00340       *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00341       *isy= TKTRNX.kminsy;
00342      }
00343      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00344      return true;
00345
00346     } else if (iy1 < TKTRNX.kminsy) { /* Start unter dem Fenster */
00347      if (iy2 < TKTRNX.kminsy) return false;
00348      *isx= ix1+ ((TKTRNX.kminsy-iy1)*(ix2-ix1))/(iy2-iy1);
00349      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00350      *isy= TKTRNX.kminsy;
00351      return true;
00352
00353     } else if (iy1 > TKTRNX.kmaxsy) { /* Start ueber dem Fenster */
00354      if (iy2 > TKTRNX.kmaxsy) return false;
00355      *isx= ix1+ ((TKTRNX.kmaxsy-iy1)*(ix2-ix1))/(iy2-iy1);
00356      if ((*isx > TKTRNX.kmaxsx) || (*isx < TKTRNX.kminsx)) return false;
00357      *isy= TKTRNX.kmaxsy;
00358      return true;
00359
00360     }
00361     *isx= ix1;                          /* Startpunkt liegt im Fenster */
00362     *isy= iy1;
00363     return true;
```

```
00364 }
00365
00366 /* Zeichnen einer gestrichelten Linie in den Backbuffer */
00367
00368 void DrawHiResDashLine (FTNINT ix,FTNINT iy, FTNINT ix2,FTNINT iy2,FTNINT *iMask)
00369 {
00370 FTNINT ixx,iyy, ixx2,iyy2;
00371 float xx,yy, dx,dy, dLin,dBlank;
00372
00373     if (*iMask <= 0) {
00374      dLin= 10., dBlank=0.; // solid
00375     } else if (*iMask == 1) {
00376      dLin= 1.; dBlank=1.; // dotted
00377     } else if (*iMask == 2) {
00378      dLin= 3.; dBlank=1.; //  substitute dashed-dotted
00379     } else if (*iMask == 3) {
00380      dLin= 3.; dBlank=3.; //  dashed
00381     } else {
00382      dLin= 3., dBlank=3.; // unrecognized -> dashed
00383     }
00384
00385     if (abs(ix2-ix) >= abs(iy2-iy)) {
00386      dx= ix2 >= ix ?  3. : -3.;
00387      dy= ((float)(iy2-iy))/((float)(ix2-ix))*dx;
00388
00389      xx= (float)ix; yy= (float)iy;
00390      while (dx != 0.) {
00391       ixx= (FTNINT) xx; iyy= (FTNINT) yy;
00392       ixx2=(FTNINT) (xx+dLin*dx); iyy2=(FTNINT) (yy+dLin*dy);
00393       xx+= (dLin+dBlank)*dx; yy+= (dLin+dBlank)*dy;
00394       if (    ((dx>=0.) && ((FTNINT)xx>=ix2) )
00395           || ((dx<=0.) && ((FTNINT)xx<=ix2) )    ) {
00396        ixx2= ix2; iyy2= iy2;
00397        dx= 0.;
00398       }
00399       SDL_RenderDrawLine(TCSrenderer, HiResX(ixx),HiResY(TEK_YMAX-iyy),
00400                                       HiResX(ixx2),HiResY(TEK_YMAX-iyy2));
00401      }
00402
00403     } else {
00404      dy= iy2 >= iy ?  3. : -3.;
00405      dx= ((float)(ix2-ix))/((float)(iy2-iy))*dy;
00406
00407      xx= (float)ix; yy= (float)iy;
00408      while (dy != 0.) {
00409       ixx= (FTNINT) xx; iyy= (FTNINT) yy;
00410       ixx2=(FTNINT) (xx+dLin*dx); iyy2=(FTNINT) (yy+dLin*dy);
00411       xx+= (dLin+dBlank)*dx; yy+= (dLin+dBlank)*dy;
00412       if (    ((dy>=0.) && ((FTNINT)yy>=iy2) )
00413           || ((dy<=0.) && ((FTNINT)yy<=iy2) )    ) {
00414        ixx2= ix2; iyy2= iy2;
00415        dy= 0.;
00416       }
00417       SDL_RenderDrawLine(TCSrenderer, HiResX(ixx), HiResY(TEK_YMAX-iyy),
00418                                       HiResX(ixx2), HiResY(TEK_YMAX-iyy2));
00419      }
00420     }
00421 }
00422
00423
00424
00425 void PlotText (const char *outtxt)
00426 {
00427 SDL_Rect dstrect;
00428 SDL_Surface* surface;
00429 SDL_Texture* texture;
00430
00431 #ifdef HIGHQUALCHAR
00432     surface = TTF_RenderUTF8_Blended(TCSfont, outtxt, sdlColorTable[TKTRNX.iTxtCol]);
00433 #else
00434     surface = TTF_RenderUTF8_Solid(TCSfont, outtxt, sdlColorTable[TKTRNX.iTxtCol]);
00435 #endif
00436     texture = SDL_CreateTextureFromSurface(TCSrenderer, surface);
00437
00438     SDL_QueryTexture(texture, NULL, NULL, &dstrect.w, &dstrect.h);
00439     dstrect.x= HiResX(TKTRNX.kBeamX);
00440     dstrect.y= HiResY(TEK_YMAX-TKTRNX.kBeamY)-dstrect.h;
00441
00442     SDL_RenderCopy(TCSrenderer, texture, NULL, &dstrect);
00443
00444     SDL_DestroyTexture(texture);
00445     SDL_FreeSurface(surface);
00446
00447     TKTRNX.kBeamX= TKTRNX.kBeamX + LoResX(dstrect.w);
00448 }
00449
00450
```

```
00451
00452 void RepaintBuffer () // Hier nicht GraphicError verwenden(Rekursionsschleifen)!
00453 {
00454 FTNINT DashStyle;
00455 int wx, wz, iStringLen, iStringActual;
00456 char szString [TCS_MESSAGELEN+1];
00457 #if (JOURNALTYP == 3)
00458  struct xJournalEntry_typ *xJournalEntry;
00459 #endif
00460
00461 #ifdef TRACE_CALLS
00462     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> called");
00463 #endif
00464
00465     DashStyle= 0; // Vorbesetzung nur notwendig bei fehlerhaftem Journal
00466     iStringActual= 0; // Zahler Einlesen String ueber XACTION_ASCII
00467     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
00468                                       , sdlColorTable[TKTRNX.iBckCol].g
00469                                       , sdlColorTable[TKTRNX.iBckCol].b
00470                                       , sdlColorTable[TKTRNX.iBckCol].a);
00471     SDL_RenderClear (TCSrenderer); // Backbuffer nach RenderPresent undefiniert
00472
00473 #if (JOURNALTYP == 3)
00474  #ifdef TRACE_CALLS
00475     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p", xTCSJournal);
00476  #endif
00477     SGLIB_DL_LIST_GET_LAST(struct xJournalEntry_typ, xTCSJournal, previous, next, xJournalEntry)
00478     while (xJournalEntry != NULL) {
00479  #ifdef TRACE_CALLS
00480      SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p", xTCSJournal);
00481      SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> Current Entry: Ptr= %p / previous: Ptr=
    %p / next: Ptr= %p",
00482                      xJournalEntry, xJournalEntry->previous, xJournalEntry->next);
00483      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_??? = %i (i1= %i, i2= %i)",
00484                    xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2 );
00485  #endif
00486      switch (xJournalEntry->action) {
00487        case XACTION_INITT: {
00488         TKTRNX.iLinCol= TCSDefaultLinCol;
00489         TKTRNX.iTxtCol= TCSDefaultTxtCol;
00490         TKTRNX.iBckCol= TCSDefaultBckCol;
00491
00492         INITT2(); // Reset TKTRNX (Margin, Scale...)
00493
00494         TKTRNX.ksizef = 0; // Reset FONT
00495         TKTRNX.kitalc = 0;
00496         if (!TCSfont)TTF_CloseFont(TCSfont);
00497         TCSfont = TTF_OpenFont(szTCSGraphicFont,
00498                                HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
00499         if (!TCSfont) {
00500          SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_INITT Error Opening Fontfile");
00501         } else {
00502          TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
00503          if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
00504           SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_INITT Fontsize?");
00505          } else {
00506           TKTRNX.khorsz= LoResX(wx);
00507           TKTRNX.kversz= LoResY(wz);
00508           TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
00509          }
00510         }
00511         TKTRNX.kBeamX= TKTRNX.klmrgn; // HOME
00512         TKTRNX.kBeamY= TKTRNX.khomey;
00513
00514        } // weiter mit Erase
00515        case XACTION_ERASE: {
00516         SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
00517                                           , sdlColorTable[TKTRNX.iBckCol].g
00518                                           , sdlColorTable[TKTRNX.iBckCol].b
00519                                           , sdlColorTable[TKTRNX.iBckCol].a);
00520         SDL_RenderClear (TCSrenderer);
00521         break; // Erase ohne Auswirkungen auf die Cursorposition!
00522        }
00523        case XACTION_MOVABS: {
00524         TKTRNX.kBeamX= xJournalEntry->i1;
00525         TKTRNX.kBeamY= xJournalEntry->i2;
00526         break;
00527        }
00528        case XACTION_DRWABS: {
00529         SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
00530                                           , sdlColorTable[TKTRNX.iLinCol].g
00531                                           , sdlColorTable[TKTRNX.iLinCol].b
00532                                           , sdlColorTable[TKTRNX.iLinCol].a );
00533         SDL_RenderDrawLine(TCSrenderer, HiResX(TKTRNX.kBeamX),
00534                                         HiResY(TEK_YMAX-TKTRNX.kBeamY),
00535                                         HiResX(xJournalEntry->i1),
00536                                         HiResY(TEK_YMAX-xJournalEntry->i2) );
```

```
00537         TKTRNX.kBeamX= xJournalEntry->i1;
00538         TKTRNX.kBeamY= xJournalEntry->i2;
00539         break;
00540        }
00541        case XACTION_DSHSTYLE: {
00542         DashStyle= xJournalEntry->i1;
00543         break;
00544        }
00545        case XACTION_DSHABS: {
00546         SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
00547                                           , sdlColorTable[TKTRNX.iLinCol].g
00548                                           , sdlColorTable[TKTRNX.iLinCol].b
00549                                           , sdlColorTable[TKTRNX.iLinCol].a );
00550         DrawHiResDashLine (TKTRNX.kBeamX,TKTRNX.kBeamY,
      xJournalEntry->i1,xJournalEntry->i2,&DashStyle);
00551         TKTRNX.kBeamX= xJournalEntry->i1;
00552         TKTRNX.kBeamY= xJournalEntry->i2;
00553         break;
00554        }
00555        case XACTION_PNTABS: {
00556         SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
00557                                           , sdlColorTable[TKTRNX.iLinCol].g
00558                                           , sdlColorTable[TKTRNX.iLinCol].b
00559                                           , sdlColorTable[TKTRNX.iLinCol].a );
00560         SDL_RenderDrawPoint(TCSrenderer, HiResX(xJournalEntry->i1),
00561                                          HiResY(TEK_YMAX-xJournalEntry->i2) );
00562         TKTRNX.kBeamX= xJournalEntry->i1;
00563         TKTRNX.kBeamY= xJournalEntry->i2;
00564         break;
00565        }
00566        case XACTION_BCKCOL: {
00567         TKTRNX.iBckCol= xJournalEntry->i1;
00568         break;
00569        }
00570        case XACTION_LINCOL: {
00571         TKTRNX.iLinCol= xJournalEntry->i1;
00572         break;
00573        }
00574       case XACTION_TXTCOL: {
00575         TKTRNX.iTxtCol= xJournalEntry->i1;
00576         break;
00577        }
00578         case XACTION_FONTATTR: {
00579         TKTRNX.kitalc= xJournalEntry->i1;
00580         if (TKTRNX.kitalc > 0) {
00581          TTF_SetFontStyle(TCSfont, TTF_STYLE_ITALIC);
00582         } else {
00583          TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
00584         }
00585
00586         if (TKTRNX.ksizef != xJournalEntry->i2) {
00587          TKTRNX.ksizef= xJournalEntry->i2;
00588          if (!TCSfont) TTF_CloseFont(TCSfont);
00589          TCSfont = TTF_OpenFont(szTCSGraphicFont,
00590                           HiResY((1+TKTRNX.ksizef)*TCS_REL_CHR_HEIGHT*TEK_YMAX));
00591          if (!TCSfont) {
00592           SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_FONTATTR");
00593          } else {
00594           if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
00595            SDL_LogError (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_FONTATTR Size");
00596           } else {
00597            TKTRNX.khorsz= LoResX(wx);
00598            TKTRNX.kversz= LoResY(wz);
00599            TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
00600           }
00601          }
00602         }
00603         break;
00604        }
00605        case XACTION_GTEXT: {
00606         iStringActual= 0;
00607         iStringLen= xJournalEntry->i1;
00608         if (iStringLen > TCS_MESSAGELEN) iStringLen= TCS_MESSAGELEN;
00609         if (iStringLen == 0) break;
00610         szString[iStringActual++]= xJournalEntry->i2;
00611         if (iStringLen == 1) {
00612          szString[iStringActual]= '\0';
00613          PlotText (szString);
00614         }
00615         break;
00616        }
00617        case XACTION_ASCII: {
00618         if (iStringActual < iStringLen) {
00619          szString[iStringActual++]= xJournalEntry->i1;
00620          if (iStringActual < iStringLen) szString[iStringActual++]= xJournalEntry->i2;
00621          if (iStringActual >= iStringLen ) {
00622           szString[iStringActual]= '\0';
```

```
00623           PlotText (szString);
00624          }
00625         }
00626         break;
00627        }
00628        case XACTION_NOOP: {
00629         break;
00630        }
00631        default: {
00632         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> XACTION_XXX");
00633         break;
00634        }
00635      }
00636      xJournalEntry= xJournalEntry -> previous;
00637     }
00638  #ifdef TRACE_CALLS
00639     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "RepaintBuffer> xTCSJournal: Ptr= %p / Last Entry: Ptr=
       %p", xTCSJournal, xJournalEntry);
00640  #endif
00641
00642 #endif
00643
00644 }
00645
00646
00647
00648 void TCSGraphicError (int iErr, const char* msg)
00649 {
00650 char cBuf[TCS_MESSAGELEN];
00651 FTNINT i; // Dummyparameter
00652
00653     snprintf( cBuf, TCS_MESSAGELEN, szTCSErrorMsg[iErr], msg );
00654     if (!TCSinitialized) { // Vor Systeminitalisierung nur Basismeldungen
00655      SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
00656      SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR,
00657                               szTCSstatWindowName, cBuf, TCSwindow);
00658     } else { // ab jetzt mit bell, outtext...
00659      SDL_RenderPresent (TCSrenderer);
00660      RepaintBuffer ();
00661      if (TCSErrorLev[iErr] > 0) {
00662       bell ();
00663       outtext (cBuf, strlen (cBuf) );
00664       if (TCSErrorLev[iErr] == 2) {
00665        SDL_LogInfo (SDL_LOG_CATEGORY_VIDEO, cBuf);
00666       }
00667       if (TCSErrorLev[iErr] == 3) {
00668        SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
00669       } else if (TCSErrorLev[iErr] < 10) {
00670        SDL_LogWarn (SDL_LOG_CATEGORY_VIDEO, cBuf);
00671        if (TCSErrorLev[iErr] == 5) {
00672         dcursr (&i,&i,&i); // Press Any Key
00673        } else if (TCSErrorLev[iErr]==8) {
00674         SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_INFORMATION,
00675                                  szTCSstatWindowName, cBuf, TCSwindow);
00676        }
00677       } else {
00678        if (TCSErrorLev[iErr] == 10) {
00679         dcursr (&i,&i,&i); // Press Any Key
00680        }
00681        if (TCSErrorLev[iErr] == 12) {
00682         SDL_ShowSimpleMessageBox(SDL_MESSAGEBOX_ERROR,
00683                                  szTCSstatWindowName, cBuf, TCSwindow);
00684        }
00685        if (iErr != ERR_EXIT) { // Error-Level von finitt durch XML veraenderbar
00686         SDL_LogError (SDL_LOG_CATEGORY_VIDEO, cBuf);
00687         finitt ();                  // Erzwungenes Beenden durch finitt
00688        }
00689       }
00690      }
00691     }
00692 }
00693
00694
00695
00696
00697
00698 /* Eventhandler zum Fensterhandling */
00699
00700 int TCSEventFilter(void* UserData, SDL_Event* event)
00701 {
00702 SDL_Point winsiz;
00703
00704     if (event->type == SDL_WINDOWEVENT) {
00705      switch (event->window.event) {
00706       case SDL_WINDOWEVENT_RESIZED:
00707       case SDL_WINDOWEVENT_MAXIMIZED:
00708       case SDL_WINDOWEVENT_RESTORED:
```

```
00709         if (event->window.windowID == SDL_GetWindowID(TCSwindow)) {
00710          if (SDL_GetRendererOutputSize(TCSrenderer, &winsiz.x, &winsiz.y) != 0) {
00711           TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
00712          } else {
00713           PixFacX= (float)(winsiz.x) / (float) TEK_XMAX;
00714           PixFacY= (float)(winsiz.y) / (float) TEK_YMAX;
00715           SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "WINSIZ> PixFac: x= %f, y= %f", PixFacX, PixFacY);
00716          }
00717         }
00718        case SDL_WINDOWEVENT_EXPOSED:
00719         if (event->window.windowID == SDL_GetWindowID(TCSwindow)) {
00720          SDL_RenderPresent (TCSrenderer);
00721          RepaintBuffer ();
00722         } else { if (event->window.windowID == SDL_GetWindowID(TCSstatwindow)) {
00723          SDL_RenderPresent (TCSstatrenderer);
00724         } }
00725         break;
00726        default:
00727         break;
00728       }
00729      }
00730      return 1;
00731 }
00732
00733
00734
00735 #ifdef AUDIOSUPPORT
00736  void audio_callback(void *sample_nr, Uint8 *raw_buffer, int bytes)
00737  {
00738  int i, length;
00739  float time, value;
00740  Sint16* buffer;
00741  SDL_AudioCVT cvt;
00742
00743     buffer= (Sint16*) raw_buffer;
00744     length = 8*bytes /SDL_AUDIO_BITSIZE(SDL_AudioDev_optained.format) /
       SDL_AudioDev_optained.channels; // Bytes = Variablenlänge (Bit/8) pro Kanal
00745     for(i=0; i < length; i++, *((int*)sample_nr)=*((int*)sample_nr)+1 ) {
00746      time = ((float)( *((int*)sample_nr)) / SAMPLE_RATE);
00747      value= BELL_AMPLITUDE * sin(2.0f * M_PI * BELL_FREQUENCY * time);
00748      buffer[i] = (Sint16)(value);
00749     }
00750     SDL_BuildAudioCVT(&cvt, AUDIO_S16SYS, 1, SAMPLE_RATE, SDL_AudioDev_optained.format,
       SDL_AudioDev_optained.channels, SDL_AudioDev_optained.freq);
00751     cvt.len = length*2;  // Sint16 = 2 Bytes
00752     cvt.buf = raw_buffer;
00753     SDL_ConvertAudio(&cvt); // Konvertiere in das Deviceformat
00754 #ifdef TRACE_CALLS
00755     SDL_LogVerbose (SDL_LOG_CATEGORY_AUDIO, "audio_callback» Number of Samples= %d Bytes allocated= %d
       ", length,bytes);
00756     SDL_LogVerbose (SDL_LOG_CATEGORY_AUDIO, "audio_callback» Bytes 16bit Audio= %d Bytes needed= %d",
       cvt.len,cvt.len_cvt);
00757 #endif
00758  }
00759 #endif
00760
00761
00762
00763 /* Eventhandler zum Parsen von XML-Dateien */
00764
00765 #ifdef XMLSUPPORT
00766
00767 void sax_callback (mxml_node_t *node, mxml_sax_event_t event, void *usr)
00768 {
00769 char * StorePtr;
00770
00771     switch (event) {
00772      case MXML_SAX_ELEMENT_OPEN: {
00773       switch (*(int*)usr ) {
00774        case -1: { // Statemachine: noch keine aktive Sektion
00775         if (strcmp(mxmlGetElement(node),szTCSsect0) == 0) {
00776          *(int*)usr= 0;  // Parsing active
00777          mxmlElementSetAttr (node,"typ","none");
00778         }
00779         break;
00780        }
00781        case 0: {
00782         if ((strcmp(mxmlGetElement(node),TCS_INISECT1) == 0)  ) {
00783          *(int*)usr= 1; // State: TCS_INISECT1
00784         } else if ((strcmp(mxmlGetElement(node),TCS_INISECT2) == 0)  ) {
00785          *(int*)usr= 2; // State: TCS_INISECT2
00786         } else if ((strcmp(mxmlGetElement(node),TCS_INISECT3) == 0)  ) {
00787          *(int*)usr= 3; // State: TCS_INISECT3
00788         }
00789         mxmlElementSetAttr (node,"typ","none");
00790         break;
00791        }
```

```
00792
00793         case 1: { // Section = Names
00794          if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINNAM) == 0)  ) {
00795           mxmlElementSetAttr (node,"typ","opaque");
00796           mxmlElementSetAttrf(node,"store","%p",&szTCSWindowName);
00797          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATNAM) == 0)  ) {
00798           mxmlElementSetAttr (node,"typ","opaque");
00799           mxmlElementSetAttrf(node,"store","%p",&szTCSstatWindowName);
00800          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCNAM) == 0)  ) {
00801           mxmlElementSetAttr (node,"typ","opaque");
00802           mxmlElementSetAttrf(node,"store","%p",&szTCSHardcopyFile);
00803          }
00804          break;
00805         }
00806
00807         case 2: { // Section = Layout
00808          if ((strcmp(mxmlGetElement(node),TCS_INIVAR_FONT) == 0)  ) {
00809           mxmlElementSetAttr (node,"typ","opaque");
00810           mxmlElementSetAttrf(node,"store","%p",&szTCSGraphicFont);
00811          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_SYSFONT) == 0)  ) {
00812           mxmlElementSetAttr (node,"typ","opaque");
00813           mxmlElementSetAttrf(node,"store","%p",&szTCSSysFont);
00814
00815          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSX) == 0)  ) {
00816           mxmlElementSetAttr (node,"typ","integer");
00817           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelpos);
00818          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINPOSY) == 0)  ) {
00819           mxmlElementSetAttr (node,"typ","integer");
00820           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelpos);
00821          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZX) == 0)  ) {
00822           mxmlElementSetAttr (node,"typ","integer");
00823           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniXrelsiz);
00824          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_WINSIZY) == 0)  ) {
00825           mxmlElementSetAttr (node,"typ","integer");
00826           mxmlElementSetAttrf(node,"store","%p",&TCSwindowIniYrelsiz);
00827
00828          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSX) == 0)  ) {
00829           mxmlElementSetAttr (node,"typ","integer");
00830           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelpos);
00831          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATPOSY) == 0)  ) {
00832           mxmlElementSetAttr (node,"typ","integer");
00833           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelpos);
00834          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZX) == 0)  ) {
00835           mxmlElementSetAttr (node,"typ","integer");
00836           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniXrelsiz);
00837          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_STATSIZY) == 0)  ) {
00838           mxmlElementSetAttr (node,"typ","integer");
00839           mxmlElementSetAttrf(node,"store","%p",&TCSstatWindowIniYrelsiz);
00840
00841          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_LINCOL) == 0)  ) {
00842           mxmlElementSetAttr (node,"typ","integer");
00843           mxmlElementSetAttrf(node,"store","%p",&TCSDefaultLinCol);
00844          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_TXTCOL) == 0)  ) {
00845           mxmlElementSetAttr (node,"typ","integer");
00846           mxmlElementSetAttrf(node,"store","%p",&TCSDefaultTxtCol);
00847          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_BCKCOL) == 0)  ) {
00848           mxmlElementSetAttr (node,"typ","integer");
00849           mxmlElementSetAttrf(node,"store","%p",&TCSDefaultBckCol);
00850          }
00851          break;
00852         }
00853
00854         case 3: { // Section = Messages
00855          if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNGRAPHCARD) == 0)  ) {
00856           mxmlElementSetAttr (node,"typ","opaque");
00857           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_UNKNGRAPHCARD]);
00858          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNGRAPHCARDL) == 0)  ) {
00859           mxmlElementSetAttr (node,"typ","integer");
00860           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_UNKNGRAPHCARD]);
00861
00862          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_NOFNTFIL) == 0)  ) {
00863           mxmlElementSetAttr (node,"typ","opaque");
00864           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_NOFNTFIL]);
00865          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_NOFNTFILL) == 0)  ) {
00866           mxmlElementSetAttr (node,"typ","integer");
00867           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_NOFNTFIL]);
00868
00869          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPN) == 0)  ) {
00870           mxmlElementSetAttr (node,"typ","opaque");
00871           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILOPN]);
00872          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCOPNL) == 0)  ) {
00873           mxmlElementSetAttr (node,"typ","integer");
00874           mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILOPN]);
00875
00876          } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRT) == 0)  ) {
00877           mxmlElementSetAttr (node,"typ","opaque");
00878           mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCFILWRT]);
```

```
00879         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCWRTL) == 0)  ) {
00880          mxmlElementSetAttr (node,"typ","integer");
00881          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCFILWRT]);
00882
00883         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINT) == 0)  ) {
00884          mxmlElementSetAttr (node,"typ","opaque");
00885          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_HDCINTERN]);
00886         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCINTL) == 0)  ) {
00887          mxmlElementSetAttr (node,"typ","integer");
00888          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_HDCINTERN]);
00889
00890         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR) == 0)  ) {
00891          mxmlElementSetAttr (node,"typ","opaque");
00892          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR]);
00893         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRL) == 0)  ) {
00894          mxmlElementSetAttr (node,"typ","integer");
00895          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR]);
00896
00897         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACT) == 0)  ) {
00898          mxmlElementSetAttr (node,"typ","opaque");
00899          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_HDCACT]);
00900         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_HDCACTL) == 0)  ) {
00901          mxmlElementSetAttr (node,"typ","integer");
00902          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_HDCACT]);
00903
00904         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRN) == 0)  ) {
00905          mxmlElementSetAttr (node,"typ","opaque");
00906          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_USRPRESSANY]);
00907         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USRWRNL) == 0)  ) {
00908          mxmlElementSetAttr (node,"typ","integer");
00909          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_USRPRESSANY]);
00910
00911         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXIT) == 0)  ) {
00912          mxmlElementSetAttr (node,"typ","opaque");
00913          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_EXIT]);
00914         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_EXITL) == 0)  ) {
00915          mxmlElementSetAttr (node,"typ","integer");
00916          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_EXIT]);
00917
00918         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATE) == 0)  ) {
00919          mxmlElementSetAttr (node,"typ","opaque");
00920          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCREATE]);
00921         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCREATEL) == 0)  ) {
00922          mxmlElementSetAttr (node,"typ","integer");
00923          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCREATE]);
00924
00925         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRY) == 0)  ) {
00926          mxmlElementSetAttr (node,"typ","opaque");
00927          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUENTRY]);
00928         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUENTRYL) == 0)  ) {
00929          mxmlElementSetAttr (node,"typ","integer");
00930          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUENTRY]);
00931
00932         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADD) == 0)  ) {
00933          mxmlElementSetAttr (node,"typ","opaque");
00934          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUADD]);
00935         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUADDL) == 0)  ) {
00936          mxmlElementSetAttr (node,"typ","integer");
00937          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUADD]);
00938
00939         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLR) == 0)  ) {
00940          mxmlElementSetAttr (node,"typ","opaque");
00941          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUCLR]);
00942         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUCLRL) == 0)  ) {
00943          mxmlElementSetAttr (node,"typ","integer");
00944          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUCLR]);
00945
00946         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWN) == 0)  ) {
00947          mxmlElementSetAttr (node,"typ","opaque");
00948          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_JOUUNKWN]);
00949         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_JOUUNKWNL) == 0)  ) {
00950          mxmlElementSetAttr (node,"typ","integer");
00951          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_JOUUNKWN]);
00952
00953         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSER) == 0)  ) {
00954          mxmlElementSetAttr (node,"typ","opaque");
00955          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLPARSER]);
00956         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLPARSERL) == 0)  ) {
00957          mxmlElementSetAttr (node,"typ","integer");
00958          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLPARSER]);
00959
00960         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPEN) == 0)  ) {
00961          mxmlElementSetAttr (node,"typ","opaque");
00962          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_XMLOPEN]);
00963         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_XMLOPENL) == 0)  ) {
00964          mxmlElementSetAttr (node,"typ","integer");
00965          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_XMLOPEN]);
```

```
00966
00967         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNAUDIO) == 0)  ) {
00968          mxmlElementSetAttr (node,"typ","opaque");
00969          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[ERR_UNKNAUDIO]);
00970         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_UNKNAUDIOL) == 0)  ) {
00971          mxmlElementSetAttr (node,"typ","integer");
00972          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[ERR_UNKNAUDIO]);
00973
00974         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2) == 0)  ) {
00975          mxmlElementSetAttr (node,"typ","opaque");
00976          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[MSG_USR2]);
00977         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_USR2L) == 0)  ) {
00978          mxmlElementSetAttr (node,"typ","integer");
00979          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[MSG_USR2]);
00980
00981         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2) == 0)  ) {
00982          mxmlElementSetAttr (node,"typ","opaque");
00983          mxmlElementSetAttrf(node,"store","%p",&szTCSErrorMsg[WRN_INI2]);
00984         } else if ((strcmp(mxmlGetElement(node),TCS_INIVAR_INI2L) == 0)  ) {
00985          mxmlElementSetAttr (node,"typ","integer");
00986          mxmlElementSetAttrf(node,"store","%p",&TCSErrorLev[WRN_INI2]);
00987
00988         }
00989         break;
00990        }
00991
00992       }
00993       break;
00994      }
00995
00996      case MXML_SAX_DATA: {
00997       switch (mxmlGetType(node)) {
00998        case MXML_INTEGER: {
00999         sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01000         (*(int*)StorePtr)= mxmlGetInteger(node);
01001         break;
01002        }
01003        case MXML_REAL: {
01004         sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01005         (*(float*)StorePtr)= mxmlGetReal(node);
01006         break;
01007        }
01008        case MXML_TEXT: {
01009         sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01010         strcpy (StorePtr, mxmlGetText(node, NULL));
01011         break;
01012        }
01013        case MXML_OPAQUE: {
01014         sscanf (mxmlElementGetAttr(mxmlGetParent(node), "store"),"%p",&StorePtr);
01015         strcpy (StorePtr, mxmlGetOpaque(node));
01016         break;
01017        }
01018       }
01019       break;
01020      }
01021
01022      case MXML_SAX_ELEMENT_CLOSE: {
01023       if ((*(int*)usr==0) && (strcmp(mxmlGetElement(node),szTCSsect0)==0)) {
01024        *(int*)usr= -1; // State: idle
01025       } else if (
01026              ((*(int*)usr==1) && (strcmp(mxmlGetElement(node),TCS_INISECT1)==0))
01027          || ((*(int*)usr==2) && (strcmp(mxmlGetElement(node),TCS_INISECT2)==0))
01028          || ((*(int*)usr==3) && (strcmp(mxmlGetElement(node),TCS_INISECT3)==0))
01029          ) {
01030        *(int*)usr= 0; // State: Parsing active
01031       }
01032       break;
01033      }
01034     }
01035 }
01036
01037
01038 /* ------------------------------------------------------------------------ */
01039
01040
01041 mxml_type_t  sax_type_callback(mxml_node_t  *node)
01042 {
01043 const char *type;
01044
01045     if ((type = mxmlElementGetAttr(node, "typ")) == NULL) type = "none";
01046     if (!strcmp(type, "integer"))
01047      return (MXML_INTEGER);
01048     else if (!strcmp(type, "opaque") || !strcmp(type, "pre"))
01049      return (MXML_OPAQUE);
01050     else if (!strcmp(type, "real"))
01051      return (MXML_REAL);
01052     else if (!strcmp(type, "text"))
```

```
01053       return (MXML_TEXT);
01054     else
01055       return (MXML_IGNORE);
01056 }
01057
01058 /* ------------------------------------------------------------------------ */
01059
01060
01061 void sax_error_callback (char *mssg)
01062 {
01063     TCSGraphicError (ERR_XMLPARSER, mssg);
01064     return;
01065 }
01066
01067 /* ------------------------------------------------------------------------ */
01068
01069 #endif   // Ende XML-Unterstützung
01070
01071
01072
01073 /*
01074 ------------------- Userroutinen: Initialisierung ------------------------------
01075 */
01076
01077 #ifdef XMLSUPPORT
01078
01079 void XMLreadProgPar (const char * filname)
01080 {
01081 int ParserState;
01082 FILE *fp;
01083 mxml_node_t *tree;
01084
01085     if (filname[0] != '\0') {
01086       fp = fopen(filname, "r");
01087       if (fp == NULL) {
01088        TCSGraphicError (ERR_XMLOPEN, filname);
01089       } else {
01090         ParserState= -1; // State= idle
01091         mxmlSetErrorCallback ((mxml_error_cb_t)sax_error_callback);
01092         tree = mxmlSAXLoadFile(NULL, fp, sax_type_callback, sax_callback, &ParserState);
01093         fclose(fp);
01094       }
01095     }
01096 }
01097
01098 #endif    // Ende XML-Unterstützung
01099
01100
01101
01102 /*
01103 Setzen der Defaultwerte vor dem Einlesen der Initialisierungsdaten
01104 */
01105
01106 void PresetProgPar ()
01107 {
01108     TCSDefaultLinCol= TCS_INIDEF_LINCOL;
01109     TCSDefaultTxtCol= TCS_INIDEF_TXTCOL;
01110     TCSDefaultBckCol= TCS_INIDEF_BCKCOL;
01111
01112     TCSwindowIniXrelpos= TCS_INIDEF_WINPOSX;
01113     TCSwindowIniYrelpos= TCS_INIDEF_WINPOSY;
01114     TCSwindowIniXrelsiz= TCS_INIDEF_WINSIZX;
01115     TCSwindowIniYrelsiz= TCS_INIDEF_WINSIZY;
01116
01117     TCSstatWindowIniXrelpos= TCS_INIDEF_STATPOSX;
01118     TCSstatWindowIniYrelpos= TCS_INIDEF_STATPOSY;
01119     TCSstatWindowIniXrelsiz= TCS_INIDEF_STATSIZX;
01120     TCSstatWindowIniYrelsiz= TCS_INIDEF_STATSIZY;
01121
01122     // Fensternamen werden nur durch winlbl vorher veraendert
01123
01124     // Hardcopyname und Zaehlerstand bleibt!
01125
01126     // Fehlermeldungen werden bei der Variablendefinition durch den Compiler initialisiert
01127 }
01128
01129
01130 /*
01131 Anpassung der Dateinamen an die Laufzeitumgebung
01132 */
01133
01134 void CustomizeProgPar ()
01135 {
01136 char        szTmpString[TCS_FILE_NAMELEN], szTmpString1[TCS_FILE_NAMELEN];
01137 FTNSTRDESC  ftn_WorkString, o, n;
01138
01139     ftn_WorkString.len= TCS_FILE_NAMELEN; // Ersatz %: durch Programmverzeichnis
```

```
01140     ftn_WorkString.addr= szTCSGraphicFont;
01141     n.addr= SDL_GetBasePath(); // Neuer Substring = Directory
01142     n.len= strlen(n.addr);
01143     o.addr= PROGDIRTOKEN; // Alter Substring
01144     o.len= strlen (o.addr);
01145     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01146                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01147                 CALLFTNSTRL(ftn_WorkString)
01148                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01149     strncpy(szTCSGraphicFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01150
01151     ftn_WorkString.addr= szTCSSysFont;
01152     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01153                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01154                 CALLFTNSTRL(ftn_WorkString)
01155                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01156     strncpy(szTCSSysFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01157
01158     SDL_free (n.addr); // SDL_BasePath nicht mehr benoetigt
01159
01160     n.addr= FNTFILEXT; // "Ersatz .% durch .TTF oder kein Punkt durch .TTF
01161     n.len= strlen(n.addr);
01162     o.addr= INIFILEXTTOKEN; // Alter Substring
01163     o.len= strlen (o.addr);
01164     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01165                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01166                 CALLFTNSTRL(ftn_WorkString)
01167                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01168     strncpy(szTCSSysFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01169     if (strchr(szTCSSysFont,'.') == 0) {
01170         strncat (szTCSSysFont, n.addr, TCS_FILE_NAMELEN-n.len);
01171     }
01172
01173     ftn_WorkString.addr= szTCSGraphicFont;
01174     SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01175                 CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01176                 CALLFTNSTRL(ftn_WorkString)
01177                 CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01178     strncpy(szTCSGraphicFont, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01179     if (strchr(szTCSGraphicFont,'.') == 0) {
01180         strncat (szTCSGraphicFont, n.addr, TCS_FILE_NAMELEN-n.len);
01181     }
01182 }
01183
01184
01185 extern void TCSdrWIN__ winlbl (FTNSTRPAR * PloWinNam, FTNSTRPAR * StatWinNam,
01186                                FTNSTRPAR *IniFilNam
01187                                FTNSTRPAR_TAIL(PloWinNam)
01188                                FTNSTRPAR_TAIL(StatWinNam)
01189                                FTNSTRPAR_TAIL(IniFilNam)      )
01190
01191 {
01192 // Absicherung der Definition der Programmparameter
01193 #if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN)
01194  #define TMPSTRLEN TCS_FILE_NAMELEN
01195 #else
01196  #define TMPSTRLEN TCS_WINDOW_NAMELEN
01197 #endif
01198
01199 int         i;
01200 FTNINT      iL;
01201 char        szTmpString[TMPSTRLEN], szTmpString1[TCS_FILE_NAMELEN];
01202 char *      iAt;
01203 FTNSTRDESC  ftn_WorkString, o, n;
01204
01205     iL= FTNSTRPARL(PloWinNam);                      // Name des Grahikfensters
01206     if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
01207     strncpy(szTmpString, FTNSTRPARA(PloWinNam),iL);
01208     szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
01209     iL= strlen (szTmpString);
01210     if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
01211     if (iL > 0) {
01212      strncpy( szTCSWindowName, szTmpString, iL);
01213      szTCSWindowName[iL]= '\0';
01214     }
01215
01216     iL= FTNSTRPARL(StatWinNam);                     // Name des Statusfensters
01217     if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
01218     strncpy(szTmpString, FTNSTRPARA(StatWinNam), iL);
01219     szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
01220     iL= strlen (szTmpString);
01221     if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
01222     if (iL > 0) {
01223      strncpy( szTCSstatWindowName, szTmpString, iL);
01224      szTCSstatWindowName[iL]= '\0';
01225     }
01226
```

```
01227     iL= FTNSTRPARL(IniFilNam);                // Name der Initialisierungsdatei
01228     if (iL > (TMPSTRLEN-1)) iL= TMPSTRLEN-1;
01229     strncpy(szTmpString, FTNSTRPARA(IniFilNam), iL);
01230     szTmpString[iL]= '\0'; // Fortranstring evtl. ohne \0
01231
01232     iL= strlen(szTmpString);
01233     if (iL > (TCS_FILE_NAMELEN-1)) iL= TCS_FILE_NAMELEN-1;
01234     if (iL > 0) {
01235      strncpy( szTCSIniFile, szTmpString, iL);
01236      szTCSIniFile[iL]= '\0';
01237
01238      iAt= strstr (szTCSIniFile, "@"); // Section Level0?
01239      if (iAt != 0) {
01240       strncpy (szTCSsect0, &iAt[1], iL);
01241       iAt[0]= '\0'; // Abschneiden von @Section0 in szTCSIniFile
01242      }
01243
01244      ftn_WorkString.len= TCS_FILE_NAMELEN;
01245      ftn_WorkString.addr= szTCSIniFile;
01246
01247      n.addr= SDL_GetBasePath(); // Neuer Substring = Directory
01248      n.len= strlen(n.addr);
01249      o.addr= PROGDIRTOKEN; // Alter Substring
01250      o.len= strlen (o.addr);
01251      SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01252                  CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01253                  CALLFTNSTRL(ftn_WorkString)
01254                  CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01255      SDL_free (n.addr);
01256
01257      n.addr= INIFILEXT; // Neuer Substring = Default Extension
01258      n.len= strlen (INIFILEXT);
01259      o.addr= INIFILEXTTOKEN; // Alter Substring
01260      o.len= strlen (o.addr);
01261      SUBSTITUTE( CALLFTNSTRA(ftn_WorkString),
01262                  CALLFTNSTRA(ftn_WorkString), CALLFTNSTRA(o), CALLFTNSTRA(n)
01263                  CALLFTNSTRL(ftn_WorkString)
01264                  CALLFTNSTRL(ftn_WorkString) CALLFTNSTRL(o) CALLFTNSTRL(n) );
01265      strncpy(szTCSIniFile, ftn_WorkString.addr, TCS_FILE_NAMELEN);
01266     }
01267
01268 #ifdef TRACE_CALLS
01269     SDL_LogSetAllPriority(LOGLEVEL); // Ausgabe in Fehlerkanal vor INIT moeglich
01270     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM,
01271             "WINLBL> Setting Windowname >%s< Statusname >%s< Inifile >%s<\n\r",
01272                           szTCSWindowName, szTCSstatWindowName, szTCSIniFile);
01273 #endif
01274
01275 // Absicherung TMPSTRLEN nicht mehr benoetigt
01276 #undef TMPSTRLEN
01277 }
01278
01279
01280
01281 extern void initt1 ()
01282 {
01283 int iD;
01284 Uint32 flags;
01285 SDL_Point winsiz;
01286 SDL_Rect rect;
01287
01288 #if (JOURNALTYP == 3)
01289  struct xJournalEntry_typ * xJournalEntry;
01290 #endif
01291
01292     if (TCSinitialized) return;   /* Bereits initialisiert */
01293
01294     SDL_LogSetAllPriority(LOGLEVEL); // Ausgabe in Fehlerkanal bereits moeglich
01295
01296     PresetProgPar(); // Compilerinitialisierung nach finitt() wiederherstellen
01297
01298     /*
01299         Falls Extension des Ini-Files .XML: XML-Parser -> hier immer XML
01300     */
01301 #if defined(XMLSUPPORT)
01302     XMLreadProgPar (szTCSIniFile);
01303 #endif
01304
01305     CustomizeProgPar (); // Ersatz %: durch Programmverzeichnis
01306
01307      /*
01308      Übernahme der durch den Nutzer angepassten Initialisierungsdaten
01309      */
01310
01311     TKTRNX.iLinCol= TCSDefaultLinCol;
01312     TKTRNX.iTxtCol= TCSDefaultTxtCol;
01313     TKTRNX.iBckCol= TCSDefaultBckCol;
```

```
01314
01315     /*
01316         Initialisierung des SDL2-Systems
01317     */
01318
01319     if (SDL_Init(SDL_INIT_VIDEO) != 0) {
01320      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01321     }
01322     if (TTF_Init() != 0) {
01323      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01324     }
01325 #ifdef AUDIOSUPPORT
01326     if (SDL_InitSubSystem(SDL_INIT_AUDIO) != 0) {
01327      TCSGraphicError (ERR_UNKNAUDIO, SDL_GetError());
01328     }
01329 #endif
01330
01331     /*
01332         Ermittlung allgemeiner systemspezifischer Parameter
01333     */
01334
01335     iD= SDL_GetNumVideoDisplays();
01336     if (iD <= 0) {
01337      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01338     } else {
01339      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> SDL_GetNumVideoDisplays = %i", iD);
01340     }
01341
01342     iD= iD-1;
01343     if (SDL_GetDisplayUsableBounds(iD, &rect) != 0) {
01344      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01345     } else {
01346      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> UsableDisplayBounds: x= %i, y= %i, w= %i, h= %i",
      rect.x,rect.y,rect.w,rect.h);
01347     }
01348
01349     SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "linear");
01350     SDL_SetEventFilter(TCSEventFilter,&TCSEventFilterData);
01351
01352     /*
01353         Erzeugung des Graphikfensters
01354     */
01355
01356     flags= SDL_WINDOW_RESIZABLE;
01357     if (szTCSWindowName[0] == '~') {
01358      flags= flags | SDL_WINDOW_BORDERLESS;
01359     }
01360     TCSwindow = SDL_CreateWindow(szTCSWindowName,
01361                                  TCSwindowIniXrelpos *rect.w / 100,
01362                                  TCSwindowIniYrelpos *rect.h / 100,
01363                                  TCSwindowIniXrelsiz *rect.w / 100,
01364                                  TCSwindowIniYrelsiz *rect.h / 100,
01365                                  flags );
01366     TCSrenderer = SDL_CreateRenderer(TCSwindow, -1, 0);
01367
01368
01369
01370     if (SDL_GetRendererOutputSize(TCSrenderer, &winsiz.x, &winsiz.y) != 0) {
01371      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01372     } else {
01373      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> RendererBounds: x= %i, y= %i", winsiz.x,winsiz.y);
01374      PixFacX= (float)(winsiz.x) / (float) TEK_XMAX;
01375      PixFacY= (float)(winsiz.y) / (float) TEK_YMAX;
01376      SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> PixFac: x= %f, y= %f", PixFacX, PixFacY);
01377     }
01378
01379     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
01380                                       , sdlColorTable[TKTRNX.iBckCol].g
01381                                       , sdlColorTable[TKTRNX.iBckCol].b
01382                                       , sdlColorTable[TKTRNX.iBckCol].a );
01383     SDL_RenderClear (TCSrenderer);
01384     SDL_RenderPresent (TCSrenderer);
01385
01386     TCSfont = TTF_OpenFont(szTCSGraphicFont,
01387                    HiResY(TCS_REL_CHR_HEIGHT*TEK_YMAX));
01388     if (!TCSfont) {
01389      TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01390     }          // TKTRNX wird durch INITT gesetzt
01391
01392     /*
01393         Erzeugung des Statusfensters
01394     */
01395
01396     if (TCSstatWindowIniYrelsiz > 0 ) {
01397      flags= SDL_WINDOW_RESIZABLE;
01398      if (szTCSstatWindowName[0] == '~') {
01399       flags= flags | SDL_WINDOW_BORDERLESS;
```

```
01400      }
01401      TCSstatwindow = SDL_CreateWindow(szTCSstatWindowName,
01402                                TCSstatWindowIniXrelpos *rect.w / 100,
01403                                TCSstatWindowIniYrelpos *rect.h / 100,
01404                                TCSstatWindowIniXrelsiz *rect.w / 100,
01405                                TCSstatWindowIniYrelsiz *rect.h / 100,
01406                                flags);
01407
01408      TCSstatrenderer = SDL_CreateRenderer(TCSstatwindow, -1, 0);
01409
01410      SDL_SetRenderDrawColor(TCSstatrenderer, sdlColorTable[TCSDefaultBckCol].r
01411                                        , sdlColorTable[TCSDefaultBckCol].g
01412                                        , sdlColorTable[TCSDefaultBckCol].b
01413                                        , sdlColorTable[TCSDefaultBckCol].a );
01414      SDL_RenderClear (TCSstatrenderer);
01415      SDL_RenderPresent (TCSstatrenderer);
01416
01417      TextLineHeight= HiResY(TCS_REL_CHR_HEIGHT*TEK_YMAX);
01418      TCSstatusfont = TTF_OpenFont(szTCSSysFont, TextLineHeight);
01419      if (!TCSstatusfont) {
01420       TCSGraphicError (ERR_UNKNGRAPHCARD, SDL_GetError());
01421      }
01422      TKTRNX.kStCol= 1; // Nur einzeilige Ausgabe
01423     }
01424
01425     /*
01426         Initialisierung des Audiosystems
01427     */
01428
01429 #ifdef AUDIOSUPPORT
01430
01431     SDL_AudioDev_wanted.freq = SAMPLE_RATE;
01432     SDL_AudioDev_wanted.format = AUDIO_S16SYS; // 16 bit integer
01433     SDL_AudioDev_wanted.channels = 1; // Mono
01434     SDL_AudioDev_wanted.samples = 2048; // buffer-size
01435     SDL_AudioDev_wanted.callback = audio_callback;
01436     SDL_AudioDev_wanted.userdata = &AudioSample_nr; // Zaehler zur Sinusberechnung
01437
01438     if(SDL_OpenAudio(&SDL_AudioDev_wanted, &SDL_AudioDev_optained) < 0) {
01439      TCSGraphicError (ERR_UNKNAUDIO, SDL_GetError());
01440     } else {
01441      if(SDL_AudioDev_wanted.format != SDL_AudioDev_optained.format) {
01442       SDL_LogInfo(SDL_LOG_CATEGORY_AUDIO, "INITT1> Failed to get the desired AudioSpec");
01443      }
01444     }
01445     SDL_LogDebug (SDL_LOG_CATEGORY_AUDIO, "INITT1> want.frequ= %i  want.channels= %i  want.samples= %i
       want.size= %i",
01446                   SDL_AudioDev_wanted.freq, SDL_AudioDev_wanted.channels, SDL_AudioDev_wanted.samples,
      SDL_AudioDev_wanted.size);
01447     SDL_LogDebug (SDL_LOG_CATEGORY_AUDIO, "INITT1> optained.frequ= %i  optained.channels= %i
      optained.samples= %i  optained.size= %i",
01448                   SDL_AudioDev_optained.freq, SDL_AudioDev_optained.channels,
      SDL_AudioDev_optained.samples, SDL_AudioDev_optained.size);
01449 #endif
01450
01451     /*
01452         Anlegen des Journals
01453     */
01454
01455 #if (JOURNALTYP == 3)
01456     xTCSJournal= NULL;
01457     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> xTCSJournal initialisiert: Ptr= %p", xTCSJournal);
01458
01459     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01460     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");
01461     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> Nach 1. malloc: xJournalEntry: Ptr= %p",
      xJournalEntry);
01462
01463     xJournalEntry->action=  XACTION_NOOP; // Erkennung Listenanfang: Wurzelelement ohne Funktion
01464     xJournalEntry->i1= 0;
01465     xJournalEntry->i2= 0;
01466     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01467     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> LIST_ADD=Create Journal: xTCSJournal: Ptr= %p /
      xJournalEntry: Ptr= %p", xTCSJournal, xJournalEntry);
01468     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "INITT1> previous: Ptr= %p / next: Ptr= %p", xJournalEntry
      -> previous, xJournalEntry -> next);
01469
01470     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01471     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
01472     xJournalEntry->action=  XACTION_INITT;
01473     xJournalEntry->i1= 0;
01474     xJournalEntry->i2= 0;
01475     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01476     SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "INITT1> Nach 2. LIST_ADD: xTCSJournal: Ptr= %p /
      xJournalEntry: Ptr= %p", xTCSJournal, xJournalEntry);
01477     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "INITT1> previous: Ptr= %p / next: Ptr= %p", xJournalEntry
      -> previous, xJournalEntry -> next);
```

```
01478 #endif
01479
01480     /*
01481         Initialisierung erfolgreich abgeschlossen
01482     */
01483
01484     TCSinitialized= true;
01485
01486     return;
01487 }
01488
01489
01490
01491 extern void finitt ()
01492 {
01493
01494 #if (JOURNALTYP == 3)
01495   struct xJournalEntry_typ    * xJournalEntry;
01496 #endif
01497
01498     if (!TCSinitialized) return; /* Graphiksystem nicht initialisiert */
01499
01500     TCSGraphicError (ERR_EXIT,"");
01501     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "finitt> Quit SDL");
01502
01503     TCSinitialized= false;       /* Ab jetzt nicht mehr funktionsfähig */
01504
01505 #if (JOURNALTYP == 3)
01506     SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, xTCSJournal,
01507           xJournalEntry,previous,next, { free (xJournalEntry);}); // free all
01508     xTCSJournal= NULL;
01509 #endif
01510
01511     TTF_CloseFont(TCSfont);
01512     TTF_CloseFont(TCSstatusfont);
01513
01514     SDL_DestroyRenderer(TCSrenderer);
01515     SDL_DestroyWindow(TCSwindow);
01516
01517     if (TCSstatWindowIniYrelsiz > 0 ) {
01518      SDL_DestroyRenderer(TCSstatrenderer);
01519      SDL_DestroyWindow(TCSstatwindow);
01520     }
01521
01522 #ifdef AUDIOSUPPORT
01523     SDL_CloseAudio();
01524 #endif
01525
01526     TTF_Quit();
01527     SDL_Quit();
01528
01529     if (TCSErrorLev[ERR_EXIT] >= 10) exit (EXIT_SUCCESS);
01530     return;
01531 }
01532
01533
01534
01535 extern void iowait (void)
01536 {
01537     SDL_RenderPresent (TCSrenderer);
01538     RepaintBuffer ();
01539 }
01540
01541
01542
01543 /*
01544 ------------------- Userroutinen: Zeichnen -------------------------------------
01545 */
01546
01547
01548
01549 extern void TCSdrWIN__ swind1 (FTNINT *ix1,FTNINT *iy1,FTNINT *ix2,FTNINT *iy2)
01550 {
01551     ClippingNotActive = (*ix1==0) && (*iy1==0) &&
01552                                         (*ix2==TEK_XMAX) && (*iy2==TEK_YMAX);
01553     /* Berechnung BOOL zur Wahrung der Programmstruktur der DOS-Version */
01554 }
01555
01556
01557
01558 extern void TCSdrWIN__ erase (void)
01559 {
01560
01561 #if (JOURNALTYP == 3)
01562  struct xJournalEntry_typ    * xJournalEntry;
01563 #endif
01564
```

```
01565     SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iBckCol].r
01566                                       , sdlColorTable[TKTRNX.iBckCol].g
01567                                       , sdlColorTable[TKTRNX.iBckCol].b
01568                                       , sdlColorTable[TKTRNX.iBckCol].a );
01569     SDL_RenderClear (TCSrenderer);
01570     SDL_RenderPresent (TCSrenderer);
01571
01572 #if (JOURNALTYP == 3)
01573      SGLIB_DL_LIST_MAP_ON_ELEMENTS (struct xJournalEntry_typ, xTCSJournal,
01574           xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
01575
01576      xTCSJournal= NULL; // create new journal
01577      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01578      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCLR,"");
01579      xJournalEntry->action=  XACTION_NOOP; // Wurzelelement ohne Vorgaenger
01580      xJournalEntry->i1= 0;
01581      xJournalEntry->i2= 0;
01582      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01583
01584      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01585      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01586      xJournalEntry->action=  XACTION_LINCOL;
01587      xJournalEntry->i1= TKTRNX.iLinCol;
01588      xJournalEntry->i2= 0;
01589      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01590
01591      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01592      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01593      xJournalEntry->action=  XACTION_TXTCOL;
01594      xJournalEntry->i1= TKTRNX.iTxtCol;
01595      xJournalEntry->i2= 0;
01596      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01597
01598      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01599      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01600      xJournalEntry->action=  XACTION_BCKCOL;
01601      xJournalEntry->i1= TKTRNX.iBckCol;
01602      xJournalEntry->i2= 0;
01603      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01604
01605      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ)); // New
       Plot
01606      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
01607      xJournalEntry->action=  XACTION_ERASE;
01608      xJournalEntry->i1= 0;
01609      xJournalEntry->i2= 0;
01610      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01611 #endif
01612 }
01613
01614
01615
01616 extern void TCSdrWIN__ movabs (FTNINT *ix,FTNINT *iy)
01617 {
01618
01619 #if (JOURNALTYP == 3)
01620  struct xJournalEntry_typ    * xJournalEntry;
01621 #endif
01622
01623
01624     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01625 #if (JOURNALTYP == 3)
01626     if (PointInWindow (*ix, *iy)) {
01627      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01628      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01629      xJournalEntry->action=  XACTION_MOVABS;
01630      xJournalEntry->i1= *ix;
01631      xJournalEntry->i2= *iy;
01632      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01633     }
01634 #endif
01635 }
01636
01637
01638
01639 extern void TCSdrWIN__ drwabs (FTNINT *ix,FTNINT *iy)
01640 {
01641 FTNINT iXClip, iYClip, iXClip2, iYClip2;
01642 #if (JOURNALTYP == 3)
01643  struct xJournalEntry_typ    * xJournalEntry;
01644 #endif
01645
01646     if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
01647      ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip2,&iYClip2); // geclippter Endpunkt
01648      SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
01649                                       , sdlColorTable[TKTRNX.iLinCol].g
01650                                       , sdlColorTable[TKTRNX.iLinCol].b
```

```
01651                                       , sdlColorTable[TKTRNX.iLinCol].a );
01652      SDL_RenderDrawLine(TCSrenderer, HiResX(iXClip),HiResY(TEK_YMAX-iYClip),
01653                                    HiResX(iXClip2),HiResY(TEK_YMAX-iYClip2));
01654
01655 #if (JOURNALTYP == 3)
01656      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01657      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01658      xJournalEntry->action=  XACTION_MOVABS;
01659      xJournalEntry->i1= iXClip;
01660      xJournalEntry->i2= iYClip;
01661      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01662
01663      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01664      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01665      xJournalEntry->action=  XACTION_DRWABS;
01666      xJournalEntry->i1= iXClip2;
01667      xJournalEntry->i2= iYClip2;
01668      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01669 #endif
01670     }
01671     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01672 #if (JOURNALTYP == 3)
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 #endif
01680 }
01681
01682
01683
01684 extern void TCSdrWIN__ dshabs (FTNINT *ix,FTNINT *iy, FTNINT *iMask)
01685 {
01686 FTNINT iXClip,iYClip, iXClip2, iYClip2;
01687 FTNINT ixx,iyy, ixx2,iyy2;
01688 float xx,yy, dx,dy, dLin,dBlank;
01689
01690
01691 #if (JOURNALTYP == 3)
01692  struct xJournalEntry_typ    * xJournalEntry;
01693 #endif
01694
01695     if (ClipLineStart(TKTRNX.kBeamX,TKTRNX.kBeamY, *ix,*iy, &iXClip,&iYClip)) {
01696      ClipLineStart(*ix,*iy, TKTRNX.kBeamX,TKTRNX.kBeamY, &iXClip2,&iYClip2); // Clip Endpunkt
01697      SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
01698                                       , sdlColorTable[TKTRNX.iLinCol].g
01699                                       , sdlColorTable[TKTRNX.iLinCol].b
01700                                       , sdlColorTable[TKTRNX.iLinCol].a );
01701      DrawHiResDashLine (iXClip,iYClip, iXClip2,iYClip2,iMask);
01702
01703 #if (JOURNALTYP == 3)
01704      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01705      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01706      xJournalEntry->action=  XACTION_MOVABS;
01707      xJournalEntry->i1= iXClip;
01708      xJournalEntry->i2= iYClip;
01709      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01710
01711      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01712      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01713      xJournalEntry->action=  XACTION_DSHSTYLE;
01714      xJournalEntry->i1= *iMask;
01715      xJournalEntry->i2= 0;
01716      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01717
01718      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01719      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01720      xJournalEntry->action=  XACTION_DSHABS;
01721      xJournalEntry->i1= iXClip2;
01722      xJournalEntry->i2= iYClip2;
01723      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01724 #endif
01725     }
01726     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01727 #if (JOURNALTYP == 3)
01728     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01729     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01730     xJournalEntry->action=  XACTION_MOVABS;
01731     xJournalEntry->i1= *ix;
01732     xJournalEntry->i2= *iy;
01733     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01734 #endif
01735 }
01736
01737
```

```
01738
01739 extern void TCSdrWIN__ pntabs (FTNINT *ix,FTNINT *iy)
01740 {
01741 #if (JOURNALTYP == 3)
01742  struct xJournalEntry_typ    * xJournalEntry;
01743  FTNINT ActPntMov;
01744 #endif
01745
01746     TKTRNX.kBeamX= *ix; TKTRNX.kBeamY= *iy;
01747     if (PointInWindow (*ix, *iy)) {
01748      SDL_SetRenderDrawColor(TCSrenderer, sdlColorTable[TKTRNX.iLinCol].r
01749                                        , sdlColorTable[TKTRNX.iLinCol].g
01750                                        , sdlColorTable[TKTRNX.iLinCol].b
01751                                        , sdlColorTable[TKTRNX.iLinCol].a );
01752      SDL_RenderDrawPoint(TCSrenderer, HiResX(*ix),HiResX(TEK_YMAX-*iy));
01753 #if (JOURNALTYP == 3)
01754      ActPntMov= XACTION_PNTABS;
01755     } else {
01756      ActPntMov= XACTION_MOVABS;
01757     }
01758     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01759     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01760     xJournalEntry->action= ActPntMov;
01761     xJournalEntry->i1= *ix;
01762     xJournalEntry->i2= *iy;
01763     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01764 #else
01765     }
01766 #endif
01767
01768 }
01769
01770
01771
01772 extern void TCSdrWIN__ bckcol (FTNINT *iCol)
01773 {
01774 #if (JOURNALTYP == 3)
01775  struct xJournalEntry_typ    * xJournalEntry;
01776 #endif
01777
01778     TKTRNX.iBckCol= *iCol;
01779     if (*iCol > MAX_COLOR_INDEX) TKTRNX.iBckCol= MAX_COLOR_INDEX;
01780
01781 #if (JOURNALTYP == 3)
01782     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01783     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01784     xJournalEntry->action=  XACTION_BCKCOL;
01785     xJournalEntry->i1= TKTRNX.iBckCol;
01786     xJournalEntry->i2= 0;
01787     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01788 #endif
01789
01790 }
01791
01792
01793
01794 extern void TCSdrWIN__ lincol (FTNINT *iCol)
01795 {
01796 #if (JOURNALTYP == 3)
01797  struct xJournalEntry_typ    * xJournalEntry;
01798 #endif
01799
01800     TKTRNX.iLinCol= *iCol;
01801     if (*iCol > MAX_COLOR_INDEX) TKTRNX.iLinCol= MAX_COLOR_INDEX;
01802
01803 #if (JOURNALTYP == 3)
01804     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01805     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01806     xJournalEntry->action=  XACTION_LINCOL;
01807     xJournalEntry->i1= TKTRNX.iLinCol;
01808     xJournalEntry->i2= 0;
01809     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01810 #endif
01811
01812 }
01813
01814
01815
01816
01817 extern void TCSdrWIN__ txtcol (FTNINT *iCol)
01818 {
01819 #if (JOURNALTYP == 3)
01820  struct xJournalEntry_typ    * xJournalEntry;
01821 #endif
01822
01823     TKTRNX.iTxtCol= *iCol;
01824     if (*iCol > MAX_COLOR_INDEX) TKTRNX.iTxtCol= MAX_COLOR_INDEX;
```

```
01825
01826 #if (JOURNALTYP == 3)
01827     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01828     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01829     xJournalEntry->action=  XACTION_TXTCOL;
01830     xJournalEntry->i1= TKTRNX.iTxtCol;
01831     xJournalEntry->i2= 0;
01832     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01833 #endif
01834
01835 }
01836
01837
01838
01839 extern void TCSdrWIN__ DefaultColour (void)
01840 {
01841     TKTRNX.iLinCol= TCSDefaultLinCol;
01842     TKTRNX.iTxtCol= TCSDefaultTxtCol;
01843     TKTRNX.iBckCol= TCSDefaultBckCol;
01844
01845     lincol (&TKTRNX.iLinCol);
01846     txtcol (&TKTRNX.iTxtCol);
01847     bckcol (&TKTRNX.iBckCol);
01848 }
01849
01850
01851
01852 /*
01853 ------------------- Userroutinen: Graphiktext ----------------------------------
01854 */
01855
01856
01857
01858 extern void TCSdrWIN__ outgtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
01859 {
01860 int iL;
01861 char outbuf [TCS_MESSAGELEN+1];
01862
01863
01864 #if (JOURNALTYP == 3)
01865  int i;
01866  struct xJournalEntry_typ    * xJournalEntry;
01867 #endif
01868
01869     if (FTNSTRPARA(ftn_string)[0] == '\0' ) return; // Leerstring char(0)
01870
01871     iL= 0; // Bei Bedarf String mit char(0) abschliessen -> Kopie in outbuf
01872     while ( (FTNSTRPARA(ftn_string)[iL] != '\0')  &&     // c-String bis \0
01873                     (iL < FTNSTRPARL(ftn_string)) &&     // String= Fortran Konstante
01874                     (iL < TCS_MESSAGELEN-1)       ) {    // Buffer Overflow
01875      outbuf[iL]= FTNSTRPARA(ftn_string)[iL];
01876      iL++;
01877     }
01878     outbuf[iL]= '\0'; //
01879
01880     PlotText (outbuf);
01881
01882 #if (JOURNALTYP == 3)
01883      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01884      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01885      xJournalEntry->action=  XACTION_GTEXT;
01886      xJournalEntry->i1= (FTNINT) iL;
01887      xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[0];
01888      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01889
01890      i= 1;
01891      while (i < iL) {
01892       xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01893       if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01894       xJournalEntry->action=  XACTION_ASCII;
01895       xJournalEntry->i1= (FTNINT) FTNSTRPARA(ftn_string)[i++];
01896       if ( i<iL ) {
01897        xJournalEntry->i2= (FTNINT) FTNSTRPARA(ftn_string)[i++];
01898       } else {
01899        xJournalEntry->i2= (FTNINT) 0;
01900       }
01901       SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01902      }
01903
01904      xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01905      if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01906      xJournalEntry->action=  XACTION_MOVABS;
01907      xJournalEntry->i1= TKTRNX.kBeamX;
01908      xJournalEntry->i2= TKTRNX.kBeamY;
01909      SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01910 #endif
01911
```

```
01912 }
01913
01914
01915
01916 extern void TCSdrWIN__ italic (void)
01917 {
01918 #if (JOURNALTYP == 3)
01919  struct xJournalEntry_typ    * xJournalEntry;
01920 #endif
01921
01922     TKTRNX.kitalc = 1;
01923     TTF_SetFontStyle(TCSfont, TTF_STYLE_ITALIC);
01924
01925 #if (JOURNALTYP == 3)
01926     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01927     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01928     xJournalEntry->action= XACTION_FONTATTR;
01929     xJournalEntry->i1= TKTRNX.kitalc;
01930     xJournalEntry->i2= TKTRNX.ksizef;
01931     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01932 #endif
01933 }
01934
01935
01936
01937 extern void TCSdrWIN__ italir (void)
01938 {
01939 #if (JOURNALTYP == 3)
01940  struct xJournalEntry_typ    * xJournalEntry;
01941 #endif
01942
01943     TKTRNX.kitalc = 0;
01944     TTF_SetFontStyle(TCSfont, TTF_STYLE_NORMAL);
01945
01946 #if (JOURNALTYP == 3)
01947     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01948     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01949     xJournalEntry->action= XACTION_FONTATTR;
01950     xJournalEntry->i1= TKTRNX.kitalc;
01951     xJournalEntry->i2= TKTRNX.ksizef;
01952     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01953 #endif
01954 }
01955
01956
01957
01958 extern void TCSdrWIN__ dblsiz (void)
01959 {
01960 int wx,wz;
01961 #if (JOURNALTYP == 3)
01962  struct xJournalEntry_typ    * xJournalEntry;
01963 #endif
01964
01965     TKTRNX.ksizef = 1;
01966
01967     if (!TCSfont)TTF_CloseFont(TCSfont);
01968     TCSfont = TTF_OpenFont(szTCSGraphicFont, 2*HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
01969     if (!TCSfont) {
01970      TCSGraphicError (ERR_NOFNT,TTF_GetError() );
01971     } else {
01972      if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
01973       TCSGraphicError (ERR_NOFNT,TTF_GetError() );
01974      } else {
01975       TKTRNX.khorsz= LoResX(wx);
01976       TKTRNX.kversz= LoResY(wz);
01977       TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
01978      }
01979     }
01980
01981     #if (JOURNALTYP == 3)
01982     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
01983     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01984     xJournalEntry->action= XACTION_FONTATTR;
01985     xJournalEntry->i1= TKTRNX.kitalc;
01986     xJournalEntry->i2= TKTRNX.ksizef;
01987     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
01988 #endif
01989 }
01990
01991
01992
01993 extern void TCSdrWIN__ nrmsiz (void)
01994 {
01995 int wx, wz;
01996 #if (JOURNALTYP == 3)
01997  struct xJournalEntry_typ    * xJournalEntry;
01998 #endif
```

```
01999
02000     TKTRNX.ksizef = 0;
02001
02002     if (!TCSfont)TTF_CloseFont(TCSfont);
02003     TCSfont = TTF_OpenFont(szTCSGraphicFont, HiResY(TEK_YMAX *TCS_REL_CHR_HEIGHT));
02004     if (!TCSfont) {
02005      TCSGraphicError (ERR_NOFNT,TTF_GetError() );
02006     } else {
02007      if(TTF_SizeText(TCSfont,"M",&wx,&wz)) {
02008       TCSGraphicError (ERR_NOFNT,TTF_GetError() );
02009      } else {
02010       TKTRNX.khorsz= LoResX(wx);
02011       TKTRNX.kversz= LoResY(wz);
02012       TKTRNX.khomey= TEK_YMAX - TKTRNX.kversz;
02013      }
02014     }
02015
02016 #if (JOURNALTYP == 3)
02017     xJournalEntry= (struct xJournalEntry_typ*) malloc (sizeof (struct xJournalEntry_typ));
02018     if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
02019     xJournalEntry->action= XACTION_FONTATTR;
02020     xJournalEntry->i1= TKTRNX.kitalc;
02021     xJournalEntry->i2= TKTRNX.ksizef;
02022     SGLIB_DL_LIST_ADD (xJournalEntry_typ, xTCSJournal, xJournalEntry, previous, next)
02023 #endif
02024 }
02025
02026
02027
02028
02029
02030
02031 extern void TCSdrWIN__ csize (FTNINT *ix,FTNINT *iy)
02032 {
02033      *ix=  TKTRNX.khorsz;
02034      *iy=  TKTRNX.kversz;
02035 }
02036
02037
02038
02039 extern void TCSdrWIN__ outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) )
02040 {
02041 int iL;
02042 char outbuf [TCS_MESSAGELEN+1];
02043 SDL_Rect dstrect;
02044 SDL_Surface* surface;
02045 SDL_Texture* texture;
02046
02047     if ( (FTNSTRPARA(ftn_string)[0] == '\0' ) // Leerstring char(0)
02048        || (TCSstatWindowIniYrelsiz <= 0 ) ) { // kein Statusfenster
02049      return;
02050     }
02051     SDL_RenderPresent (TCSrenderer);
02052     RepaintBuffer ();
02053
02054     iL= 0; // Bei Bedarf String mit char(0) abschliessen -> Kopie in outbuf
02055     while ( (FTNSTRPARA(ftn_string)[iL] != '\0')  &&     // c-String bis \0
02056                     (iL < FTNSTRPARL(ftn_string)) &&     // String= Fortran Konstante
02057                     (iL < TCS_MESSAGELEN-1)       ) {    // Buffer Overflow
02058      outbuf[iL]= FTNSTRPARA(ftn_string)[iL];
02059      iL++;
02060     }
02061     outbuf[iL]= '\0'; //
02062
02063     SDL_SetRenderDrawColor(TCSstatrenderer, sdlColorTable[TCSDefaultBckCol].r
02064                                          , sdlColorTable[TCSDefaultBckCol].g
02065                                          , sdlColorTable[TCSDefaultBckCol].b
02066                                          , sdlColorTable[TCSDefaultBckCol].a );
02067     SDL_RenderClear (TCSstatrenderer);
02068
02069 #ifdef HIGHQUALCHAR
02070     surface = TTF_RenderUTF8_Blended (TCSstatusfont, outbuf, sdlColorTable[TCSDefaultLinCol]);
02071 #else
02072     surface = TTF_RenderUTF8_Solid (TCSstatusfont, outbuf, sdlColorTable[TCSDefaultLinCol]);
02073 #endif
02074
02075     texture = SDL_CreateTextureFromSurface(TCSstatrenderer, surface);
02076
02077     dstrect.x= 0;
02078     dstrect.y= 0;
02079     SDL_QueryTexture(texture, NULL, NULL, &dstrect.w, &dstrect.h);
02080     SDL_RenderCopy(TCSstatrenderer, texture, NULL, &dstrect);
02081
02082     SDL_RenderPresent(TCSstatrenderer);
02083     SDL_DestroyTexture(texture);
02084     SDL_FreeSurface(surface);
02085 }
```

```
02086
02087
02088
02089 extern void TCSdrWIN__ bell (void)
02090 {
02091 #ifdef AUDIOSUPPORT
02092     AudioSample_nr= 0;
02093     SDL_PauseAudio(0); // start playing sound
02094     SDL_Delay(BELL_DURATION); // wait while sound is playing
02095     SDL_PauseAudio(1); // stop playing sound
02096 #endif
02097     return;
02098 }
02099
02100
02101 extern void TCSdrWIN__ GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
02102                                      FTNINT *iL  FTNSTRPAR_TAIL(ftn_string))
02103 {
02104     TCSGraphicError (*iErr, FTNSTRPARA(ftn_string));
02105
02106 }
02107
02108
02109
02110 /*
02111 ------------------- Userroutinen: Graphic Input---------------------------------
02112 */
02113
02114
02115
02116 extern void TCSdrWIN__ dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy)
02117 {
02118 SDL_Event  event;
02119
02120     if (!TCSinitialized) return;            /* Aufhängen vermeiden */
02121
02122     SDL_RenderPresent (TCSrenderer);
02123     RepaintBuffer ();
02124     SDL_RaiseWindow(TCSwindow); // Set input focus
02125
02126     *ic= 0;
02127     while (*ic == 0) {
02128      SDL_WaitEvent (&event);
02129      switch (event.type) {
02130       case SDL_KEYDOWN:
02131        if (event.key.keysym.sym < 256) {
02132         *ic= (FTNINT) event.key.keysym.sym;
02133        }
02134        break;
02135       case SDL_MOUSEBUTTONDOWN:
02136        if (ix == iy) break; // Aufruf TINPUT, nicht  DCURSR
02137        switch (event.button.button) { // Tastaturcode analog DOS
02138         case SDL_BUTTON_LEFT: *ic= 1; break;
02139         case SDL_BUTTON_RIGHT: *ic= 2; break;
02140         case SDL_BUTTON_MIDDLE: *ic= 4; break;
02141        }
02142        *ix= (FTNINT) (LoResX(event.button.x));
02143        *iy= (FTNINT) (TEK_YMAX-LoResY(event.button.y));
02144        break;
02145       default:
02146        TCSEventFilter(NULL, &event); // Weiterleitung Standardhandler, ic = Dummy
02147        break;
02148      }
02149     }
02150 }
02151
02152
02153
02154 /*
02155 ------------------- Userroutinen: Hardcopy ----------------------------------
02156 */
02157
02158
02159
02160 extern void TCSdrWIN__ hdcopy (void)
02161 {
02162
02163 FTNINT      iErr;
02164 FTNSTRDESC  ftnstrg;
02165 char        szTmpString[TCS_FILE_NAMELEN];
02166 SDL_RWops*  hFile;
02167
02168 #if (JOURNALTYP == 3)
02169  struct xJournalEntry_typ *xJournalEntry;
02170 #endif
02171
02172     snprintf( szTmpString,TCS_FILE_NAMELEN, szTCSHardcopyFile, iHardcopyCount++ );
```

```
02173     hFile = SDL_RWFromFile( szTmpString, "r" );
02174     while ((iHardcopyCount < MAX_HDCCOUNT) && (hFile != NULL) ) {
02175      SDL_RWclose (hFile);
02176      snprintf( szTmpString,TCS_FILE_NAMELEN, szTCSHardcopyFile, iHardcopyCount++ );
02177      hFile = SDL_RWFromFile( szTmpString, "r" );
02178     }
02179     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> iHardcopyCount Next= %i", iHardcopyCount);
02180     SDL_LogDebug (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Filnam= %s", szTmpString);
02181     if (hFile != NULL) { // iHardcopyCount zu klein
02182      SDL_RWclose (hFile);
02183      SDL_LogError (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Open HDC_File: kein freier Filename");
02184      return;
02185     }
02186
02187     hFile = SDL_RWFromFile( szTmpString, "wb" );
02188     if (hFile == NULL) {
02189      SDL_LogError (SDL_LOG_CATEGORY_SYSTEM, "HDCOPY> Error openening %s",szTmpString);
02190      return;
02191     }
02192
02193     TCSGraphicError (MSG_HDCACT, szTmpString);
02194
02195 #if (JOURNALTYP == 3)
02196     SGLIB_DL_LIST_GET_LAST (struct xJournalEntry_typ, xTCSJournal, previous, next, xJournalEntry)
02197 #ifdef TRACE_CALLS
02198     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xTCSJournal: Ptr= %p", xTCSJournal);
02199     SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> 1. Entry: Ptr= %p / previous: Ptr= %p / next:
      Ptr= %p", xJournalEntry, xJournalEntry -> previous, xJournalEntry -> next);
02200 #endif
02201     while (xJournalEntry != NULL) {
02202      snprintf( szTmpString,TCS_FILE_NAMELEN, "%02i#%04i-%03i\n", xJournalEntry->action,
     xJournalEntry->i1, xJournalEntry->i2 );
02203      SDL_RWwrite(hFile, szTmpString, 1, strlen(szTmpString));
02204 #ifdef TRACE_CALLS
02205      switch (xJournalEntry->action) {
02206        case XACTION_INITT: {
02207         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_INITT");
02208         break;
02209        }
02210        case XACTION_ERASE: {
02211         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_ERASE");
02212         break;
02213        }
02214        case XACTION_MOVABS: {
02215         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_MOVABS: x= %i, y= %i",
     xJournalEntry->i1, xJournalEntry->i2);
02216         break;
02217        }
02218        case XACTION_DRWABS: {
02219         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DRWABS: x= %i, y= %i",
     xJournalEntry->i1, xJournalEntry->i2);
02220         break;
02221        }
02222        case XACTION_DSHSTYLE: {
02223         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DSHSTYLE: x= %i", xJournalEntry->i1);
02224         break;
02225        }
02226        case XACTION_DSHABS: {
02227         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_DSHABS: x= %i, y= %i",
     xJournalEntry->i1, xJournalEntry->i2);
02228         break;
02229        }
02230        case XACTION_PNTABS: {
02231         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_PNTABS: x= %i, y= %i",
     xJournalEntry->i1, xJournalEntry->i2);
02232         break;
02233        }
02234        case XACTION_BCKCOL: {
02235         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_BCKCOL: x= %i", xJournalEntry->i1);
02236         break;
02237        }
02238        case XACTION_TXTCOL: {
02239         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_TXTCOL: x= %i", xJournalEntry->i1);
02240         break;
02241        }
02242        case XACTION_LINCOL: {
02243         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_LINCOL: x= %i", xJournalEntry->i1);
02244         break;
02245        }
02246        case XACTION_FONTATTR: {
02247         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_FONTATTR: x= %i, y= %i",
     xJournalEntry->i1, xJournalEntry->i2);
02248         break;
02249        }
02250        case XACTION_GTEXT: {
02251         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_GTEXT: Len= %i, Char[%i]= %c",
02252                       xJournalEntry->i1, xJournalEntry->i2, xJournalEntry->i2);
```

```
02253         break;
02254        }
02255        case XACTION_ASCII: {
02256         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_ASCII: Char1[%i]= %c, Char2[%i]= %c",
02257                       xJournalEntry->i1, xJournalEntry->i1, xJournalEntry->i2, xJournalEntry->i2);
02258         break;
02259        }
02260        case XACTION_NOOP: {
02261         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_NOOP");
02262         break;
02263        }
02264        default: {
02265         SDL_LogDebug (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> XACTION_XXX");
02266         break;
02267        }
02268      }
02269      SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xJournalEntry: Ptr= %i / previous: Ptr= %i /
       next: Ptr= %i", xJournalEntry, xJournalEntry -> previous, xJournalEntry -> next);
02270 #endif // TRACE_CALLS
02271      xJournalEntry= xJournalEntry -> previous;
02272     }
02273
02274    SDL_RWclose (hFile);
02275 #ifdef TRACE_CALLS
02276    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> xTCSJournal New Current Entry: Ptr= %p",
       xJournalEntry);
02277    SDL_LogVerbose (SDL_LOG_CATEGORY_VIDEO, "HDCOPY> Previous: Ptr= %p  Next: Ptr= %p",
       xJournalEntry->previous, xJournalEntry->next);
02278 #endif // TRACE_CALLS
02279
02280
02281 #endif // Journaltyp=3
02282
02283 }
02284
02285
02286
02287 /*
02288 ------------------- subroutine LIB_MOVC3 fuer Watcom- und GNU-Compiler ---------
02289 Hier nicht benoetigt, nur wg. Kompatibilitaet zur DOS-Version enthalten
02290 */
02291
02292
02293 extern void lib_movc3 (FTNINT *len,FTNSTRPAR *sou,FTNSTRPAR *dst
02294                                FTNSTRPAR_TAIL(sou)  FTNSTRPAR_TAIL(dst) )
02295
02296 {
02297 int n;
02298     if (FTNSTRPARA(dst) <= FTNSTRPARA(sou) ) {
02299      for (n=0; n<*len; n++) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
02300     } else {
02301      for (n= (*len)-1; n>=0; n--) FTNSTRPARA(dst)[n]= FTNSTRPARA(sou)[n];
02302     };
02303 }
```

7.32 TCSdSDLc.h File Reference

SDL Port: Low-Level Driver.

Classes

- struct FTNCOMPLEX
- struct FTNSTRDESC

Macros

- #define TEK_XMAX 1023
- #define TEK_YMAX 780
- #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
- #define TCSdrWIN__
- #define false 0
- #define true !false

Typedefs

- 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
- typedef int bool

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)

7.32.1 Detailed Description

SDL Port: Low-Level Driver.

Version

1.1

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

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

7.32.2 Macro Definition Documentation

7.32.2.1 bckcol

```
#define bckcol bckcol_
```

Definition at line 69 of file TCSdSDLc.h.

7.32.2.2 bell

```
void bell bell_
```

Definition at line 78 of file TCSdSDLc.h.

7.32.2.3 BELL_AMPLITUDE

```
#define BELL_AMPLITUDE 32000.0
```
Definition at line 132 of file TCSdSDLc.h.

7.32.2.4 BELL_DURATION

```
#define BELL_DURATION 200
```
Definition at line 134 of file TCSdSDLc.h.

7.32.2.5 BELL_FREQUENCY

```
#define BELL_FREQUENCY 441.0f
```
Definition at line 133 of file TCSdSDLc.h.

7.32.2.6 CALLFTNSTRA

```
#define CALLFTNSTRA(
              ftns ) ftns.addr
```
Definition at line 51 of file TCSdSDLc.h.

7.32.2.7 CALLFTNSTRL

```
#define CALLFTNSTRL(
              ftns ) , ftns.len
```
Definition at line 52 of file TCSdSDLc.h.

7.32.2.8 csize

```
#define csize csize_
```
Definition at line 82 of file TCSdSDLc.h.

7.32.2.9 dblsiz

```
#define dblsiz(
              void ) dblsiz_
```
Definition at line 76 of file TCSdSDLc.h.

7.32.2.10 dcursr

```
#define dcursr dcursr_
```
Definition at line 81 of file TCSdSDLc.h.

7.32.2.11 DefaultColour

```
#define DefaultColour(
              void ) defaultcolour_
```
Definition at line 72 of file TCSdSDLc.h.

7.32.2.12 drwabs

```
#define drwabs drwabs_
```
Definition at line 66 of file TCSdSDLc.h.

7.32.2.13 dshabs

```
#define dshabs dshabs_
```
Definition at line 67 of file TCSdSDLc.h.

7.32.2.14 erase

```
#define erase(
              void ) erase_
```
Definition at line 63 of file TCSdSDLc.h.

7.32.2.15 ERR_EXIT

```
#define ERR_EXIT 12
```
Definition at line 169 of file TCSdSDLc.h.

7.32.2.16 ERR_NOFNT

```
#define ERR_NOFNT 4
```
Definition at line 161 of file TCSdSDLc.h.

7.32.2.17 ERR_NOFNTFIL

```
#define ERR_NOFNTFIL 3
```
Definition at line 160 of file TCSdSDLc.h.

7.32.2.18 ERR_UNKNAUDIO

```
#define ERR_UNKNAUDIO 22
```
Definition at line 179 of file TCSdSDLc.h.

7.32.2.19 ERR_UNKNGRAPHCARD

```
#define ERR_UNKNGRAPHCARD 2
```
Definition at line 159 of file TCSdSDLc.h.

7.32.2.20 ERR_XMLOPEN

```
#define ERR_XMLOPEN 21
```
Definition at line 178 of file TCSdSDLc.h.

7.32.2.21 ERR_XMLPARSER

```
#define ERR_XMLPARSER 20
```
Definition at line 177 of file TCSdSDLc.h.

7.32.2.22 false

```
#define false 0
```
Definition at line 381 of file TCSdSDLc.h.

7.32.2.23 finitt

```
void finitt finitt_
```
Definition at line 59 of file TCSdSDLc.h.

7.32.2.24 FTNSTRPAR_TAIL

```
#define FTNSTRPAR_TAIL(
              ftns ) , FTNCHARLEN ftns##_len
```
Definition at line 48 of file TCSdSDLc.h.

7.32.2.25 FTNSTRPARA

```
#define FTNSTRPARA(
              ftns ) ftns
```
Definition at line 49 of file TCSdSDLc.h.

7.32.2.26 FTNSTRPARL

```
#define FTNSTRPARL(
              ftns ) ftns##_len
```
Definition at line 50 of file TCSdSDLc.h.

7.32.2.27 FWRDFTNSTRA

```
#define FWRDFTNSTRA(
              ftns ) ftns
```
Definition at line 53 of file TCSdSDLc.h.

7.32.2.28 FWRDFTNSTRL

```
#define FWRDFTNSTRL(
              ftns ) , ftns##_len
```
Definition at line 54 of file TCSdSDLc.h.

7.32.2.29 GETARG

```
#define GETARG getarg_
```
Definition at line 88 of file TCSdSDLc.h.

7.32.2.30 GraphicError

```
#define GraphicError graphicerror_
```
Definition at line 61 of file TCSdSDLc.h.

7.32.2.31 hdcopy

```
#define hdcopy(
              void ) hdcopy_
```
Definition at line 83 of file TCSdSDLc.h.

7.32.2.32 INIFILEXTTOKEN

```
#define INIFILEXTTOKEN ".%" /* Token fuer den Filenamenparser */
```
Definition at line 126 of file TCSdSDLc.h.

7.32.2.33 initt1

```
#define initt1 initt1_
```
Definition at line 58 of file TCSdSDLc.h.

7.32.2.34 INITT2

```
void INITT2 initt2_
```
Definition at line 91 of file TCSdSDLc.h.

7.32.2.35 iowait

```
#define iowait(
              void ) iowait_
```
Definition at line 60 of file TCSdSDLc.h.

7.32.2.36 italic

```
#define italic(
              void ) italic_
```
Definition at line 74 of file TCSdSDLc.h.

7.32.2.37 italir

```
#define italir(
              void ) italir_
```
Definition at line 75 of file TCSdSDLc.h.

7.32.2.38 lib_movc3

```
#define lib_movc3 lib_movc3_
```
Definition at line 84 of file TCSdSDLc.h.

7.32.2.39 lincol

```
#define lincol lincol_
```
Definition at line 70 of file TCSdSDLc.h.

7.32.2.40 MAX_HDCCOUNT

```
#define MAX_HDCCOUNT 1000 /* s.u.:  Format TCS_HDCFILE_NAME */
```
Definition at line 124 of file TCSdSDLc.h.

7.32.2.41 movabs

```
#define movabs movabs_
```
Definition at line 65 of file TCSdSDLc.h.

7.32.2.42 MSG_HDCACT

```
#define MSG_HDCACT 10
```
Definition at line 167 of file TCSdSDLc.h.

7.32.2.43 MSG_MAXERRNO

```
#define MSG_MAXERRNO 25
```
Definition at line 182 of file TCSdSDLc.h.

7.32.2.44 MSG_NOMOUSE

```
#define MSG_NOMOUSE 5
```
Definition at line 162 of file TCSdSDLc.h.

7.32.2.45 MSG_USR

```
#define MSG_USR 9
```
Definition at line 166 of file TCSdSDLc.h.

7.32.2.46 MSG_USR2

```
#define MSG_USR2 23
```
Definition at line 180 of file TCSdSDLc.h.

7.32.2.47 nrmsiz

```
#define nrmsiz(
              void ) nrmsiz_
```
Definition at line 77 of file TCSdSDLc.h.

7.32.2.48 outgtext

```
#define outgtext outgtext_
```
Definition at line 73 of file TCSdSDLc.h.

7.32.2.49 outtext

```
#define outtext outtext_
```
Definition at line 79 of file TCSdSDLc.h.

7.32.2.50 pntabs

```
#define pntabs pntabs_
```
Definition at line 68 of file TCSdSDLc.h.

7.32.2.51 PROGDIRTOKEN

```
#define PROGDIRTOKEN "%:"
```
Definition at line 127 of file TCSdSDLc.h.

7.32.2.52 SAMPLE_RATE

```
#define SAMPLE_RATE 41000
```
Definition at line 131 of file TCSdSDLc.h.

7.32.2.53 STAT_MAXROWS

```
#define STAT_MAXROWS 1 /* vorhandene Statuszeilen */
```
Definition at line 116 of file TCSdSDLc.h.

7.32.2.54 SUBSTITUTE

```
#define SUBSTITUTE substitute_
```
Definition at line 94 of file TCSdSDLc.h.

7.32.2.55 swind1

```
#define swind1 swind1_
```
Definition at line 64 of file TCSdSDLc.h.

7.32.2.56 TCS_FILE_NAMELEN

```
#define TCS_FILE_NAMELEN 128
```
Definition at line 121 of file TCSdSDLc.h.

7.32.2.57 TCS_HDCFILE_NAME

```
#define TCS_HDCFILE_NAME "HDC%03i.UNKNOWN"
```
Definition at line 207 of file TCSdSDLc.h.

7.32.2.58 TCS_INIDEF_BCKCOL

```
#define TCS_INIDEF_BCKCOL 0
```
Definition at line 239 of file TCSdSDLc.h.

7.32.2.59 TCS_INIDEF_COPLCK

```
#define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager:  ClipBoard locked."
```
Definition at line 287 of file TCSdSDLc.h.

7.32.2.60 TCS_INIDEF_COPLCKL

```
#define TCS_INIDEF_COPLCKL 1
```
Definition at line 289 of file TCSdSDLc.h.

7.32.2.61 TCS_INIDEF_COPMEM

```
#define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager:  Out of Memory."
```
Definition at line 283 of file TCSdSDLc.h.

7.32.2.62 TCS_INIDEF_COPMEML

```
#define TCS_INIDEF_COPMEML 1
```
Definition at line 285 of file TCSdSDLc.h.

7.32.2.63 TCS_INIDEF_COPMEN

```
#define TCS_INIDEF_COPMEN "Copy"
```
Definition at line 212 of file TCSdSDLc.h.

7.32.2.64 TCS_INIDEF_EXIT

```
#define TCS_INIDEF_EXIT "Press any key to exit program."
```
Definition at line 279 of file TCSdSDLc.h.

7.32.2.65 TCS_INIDEF_EXITL

```
#define TCS_INIDEF_EXITL 10
```
Definition at line 281 of file TCSdSDLc.h.

7.32.2.66 TCS_INIDEF_FONT

```
#define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
```
Definition at line 214 of file TCSdSDLc.h.

7.32.2.67 TCS_INIDEF_HDCACT

```
#define TCS_INIDEF_HDCACT "Hardcopy in progress:  File %s created."
```
Definition at line 271 of file TCSdSDLc.h.

7.32.2.68 TCS_INIDEF_HDCACTL

```
#define TCS_INIDEF_HDCACTL 1
```
Definition at line 273 of file TCSdSDLc.h.

7.32.2.69 TCS_INIDEF_HDCINT

```
#define TCS_INIDEF_HDCINT "GRAPH2D HARDCOPY: Internal Error."
```
Definition at line 263 of file TCSdSDLc.h.

7.32.2.70 TCS_INIDEF_HDCINTL

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

7.32.2.71 TCS_INIDEF_HDCOPN

```
#define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
```
Definition at line 255 of file TCSdSDLc.h.

7.32.2.72 TCS_INIDEF_HDCOPNL

```
#define TCS_INIDEF_HDCOPNL 5
```
Definition at line 257 of file TCSdSDLc.h.

7.32.2.73 TCS_INIDEF_HDCWRT

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

7.32.2.74 TCS_INIDEF_HDCWRTL

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

7.32.2.75 TCS_INIDEF_INI2

```
#define TCS_INIDEF_INI2 "%s"
```
Definition at line 327 of file TCSdSDLc.h.

7.32.2.76 TCS_INIDEF_INI2L

```
#define TCS_INIDEF_INI2L 5
```
Definition at line 329 of file TCSdSDLc.h.

7.32.2.77 TCS_INIDEF_JOUADD

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

7.32.2.78 TCS_INIDEF_JOUADDL

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

7.32.2.79 TCS_INIDEF_JOUCLR

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

7.32.2.80 TCS_INIDEF_JOUCLRL

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

7.32.2.81 TCS_INIDEF_JOUCREATE

```
#define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal.  Error-No:  %s."
```
Definition at line 291 of file TCSdSDLc.h.

7.32.2.82 TCS_INIDEF_JOUCREATEL

```
#define TCS_INIDEF_JOUCREATEL 5
```
Definition at line 293 of file TCSdSDLc.h.

7.32.2.83 TCS_INIDEF_JOUENTRY

```
#define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
```
Definition at line 295 of file TCSdSDLc.h.

7.32.2.84 TCS_INIDEF_JOUENTRYL

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

7.32.2.85 TCS_INIDEF_JOUUNKWN

```
#define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
```
Definition at line 307 of file TCSdSDLc.h.

7.32.2.86 TCS_INIDEF_JOUUNKWNL

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

7.32.2.87 TCS_INIDEF_LINCOL

```
#define TCS_INIDEF_LINCOL 1
```
Definition at line 235 of file TCSdSDLc.h.

7.32.2.88 TCS_INIDEF_NOFNT

```
#define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
```
Definition at line 251 of file TCSdSDLc.h.

7.32.2.89 TCS_INIDEF_NOFNTFIL

```
#define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
```
Definition at line 247 of file TCSdSDLc.h.

7.32.2.90 TCS_INIDEF_NOFNTFILL

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

7.32.2.91 TCS_INIDEF_NOFNTL

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

7.32.2.92 TCS_INIDEF_STATPOSX

```
#define TCS_INIDEF_STATPOSX 1
```
Definition at line 226 of file TCSdSDLc.h.

7.32.2.93 TCS_INIDEF_STATPOSY

```
#define TCS_INIDEF_STATPOSY 91
```
Definition at line 228 of file TCSdSDLc.h.

7.32.2.94 TCS_INIDEF_STATSIZX

```
#define TCS_INIDEF_STATSIZX 98
```
Definition at line 230 of file TCSdSDLc.h.

7.32.2.95 TCS_INIDEF_STATSIZY

```
#define TCS_INIDEF_STATSIZY 3
```
Definition at line 232 of file TCSdSDLc.h.

7.32.2.96 TCS_INIDEF_SYSFONT

```
#define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
```
Definition at line 216 of file TCSdSDLc.h.

7.32.2.97 TCS_INIDEF_TXTCOL

```
#define TCS_INIDEF_TXTCOL 1
```
Definition at line 237 of file TCSdSDLc.h.

7.32.2.98 TCS_INIDEF_UNKNAUDIO

```
#define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System:  Error %s."
```
Definition at line 319 of file TCSdSDLc.h.

7.32.2.99 TCS_INIDEF_UNKNAUDIOL

```
#define TCS_INIDEF_UNKNAUDIOL 5
```
Definition at line 321 of file TCSdSDLc.h.

7.32.2.100 TCS_INIDEF_UNKNGRAPHCARD

```
#define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System:  Error %s."
```
Definition at line 243 of file TCSdSDLc.h.

7.32.2.101 TCS_INIDEF_UNKNGRAPHCARDL

```
#define TCS_INIDEF_UNKNGRAPHCARDL 10
```
Definition at line 245 of file TCSdSDLc.h.

7.32.2.102 TCS_INIDEF_USR

```
#define TCS_INIDEF_USR "%s"
```
Definition at line 267 of file TCSdSDLc.h.

7.32.2.103 TCS_INIDEF_USR2

```
#define TCS_INIDEF_USR2 "%s"
```
Definition at line 323 of file TCSdSDLc.h.

7.32.2.104 TCS_INIDEF_USR2L

```
#define TCS_INIDEF_USR2L 5
```
Definition at line 325 of file TCSdSDLc.h.

7.32.2.105 TCS_INIDEF_USRL

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

7.32.2.106 TCS_INIDEF_USRWRN

```
#define TCS_INIDEF_USRWRN "Press any key to continue."
```
Definition at line 275 of file TCSdSDLc.h.

7.32.2.107 TCS_INIDEF_USRWRNL

```
#define TCS_INIDEF_USRWRNL 5
```
Definition at line 277 of file TCSdSDLc.h.

7.32.2.108 TCS_INIDEF_WINPOSX

```
#define TCS_INIDEF_WINPOSX 1
```
Definition at line 218 of file TCSdSDLc.h.

7.32.2.109 TCS_INIDEF_WINPOSY

```
#define TCS_INIDEF_WINPOSY 3
```
Definition at line 220 of file TCSdSDLc.h.

7.32.2.110 TCS_INIDEF_WINSIZX

```
#define TCS_INIDEF_WINSIZX 98
```
Definition at line 222 of file TCSdSDLc.h.

7.32.2.111 TCS_INIDEF_WINSIZY

```
#define TCS_INIDEF_WINSIZY 85
```
Definition at line 224 of file TCSdSDLc.h.

7.32.2.112 TCS_INIDEF_XMLOPEN

```
#define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
```
Definition at line 315 of file TCSdSDLc.h.

7.32.2.113 TCS_INIDEF_XMLOPENL

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

7.32.2.114 TCS_INIDEF_XMLPARSER

```
#define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File:  %s"
```
Definition at line 311 of file TCSdSDLc.h.

7.32.2.115 TCS_INIDEF_XMLPARSERL

```
#define TCS_INIDEF_XMLPARSERL 8
```
Definition at line 313 of file TCSdSDLc.h.

7.32.2.116 TCS_INIFILE_NAME

```
#define TCS_INIFILE_NAME "Graph2D"
```
Definition at line 129 of file TCSdSDLc.h.

7.32.2.117 TCS_INISECT0

```
#define TCS_INISECT0 "Graph2D"
```
Definition at line 192 of file TCSdSDLc.h.

7.32.2.118 TCS_INISECT1

```
#define TCS_INISECT1 "Names"
```
Definition at line 194 of file TCSdSDLc.h.

7.32.2.119 TCS_INISECT2

```
#define TCS_INISECT2 "Layout"
```
Definition at line 210 of file TCSdSDLc.h.

7.32.2.120 TCS_INISECT3

```
#define TCS_INISECT3 "Messages"
```
Definition at line 241 of file TCSdSDLc.h.

7.32.2.121 TCS_INIVAR_BCKCOL

```
#define TCS_INIVAR_BCKCOL "G2dBckCol"
```
Definition at line 238 of file TCSdSDLc.h.

7.32.2.122 TCS_INIVAR_COPLCK

```
#define TCS_INIVAR_COPLCK "G2dClipLock"
```
Definition at line 286 of file TCSdSDLc.h.

7.32.2.123 TCS_INIVAR_COPLCKL

```
#define TCS_INIVAR_COPLCKL "G2dClipLockL"
```
Definition at line 288 of file TCSdSDLc.h.

7.32.2.124 TCS_INIVAR_COPMEM

```
#define TCS_INIVAR_COPMEM "G2dNoMemory"
```
Definition at line 282 of file TCSdSDLc.h.

7.32.2.125 TCS_INIVAR_COPMEML

```
#define TCS_INIVAR_COPMEML "G2dNoMemoryL"
```
Definition at line 284 of file TCSdSDLc.h.

7.32.2.126 TCS_INIVAR_COPMEN

```
#define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
```
Definition at line 211 of file TCSdSDLc.h.

7.32.2.127 TCS_INIVAR_EXIT

```
#define TCS_INIVAR_EXIT "G2dExit"
```
Definition at line 278 of file TCSdSDLc.h.

7.32.2.128 TCS_INIVAR_EXITL

```
#define TCS_INIVAR_EXITL "G2dExitL"
```
Definition at line 280 of file TCSdSDLc.h.

7.32.2.129 TCS_INIVAR_FONT

```
#define TCS_INIVAR_FONT "G2dGraphicFont"
```
Definition at line 213 of file TCSdSDLc.h.

7.32.2.130 TCS_INIVAR_HDCACT

```
#define TCS_INIVAR_HDCACT "G2dHdcActive"
```
Definition at line 270 of file TCSdSDLc.h.

7.32.2.131 TCS_INIVAR_HDCACTL

```
#define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
```
Definition at line 272 of file TCSdSDLc.h.

7.32.2.132 TCS_INIVAR_HDCINT

```
#define TCS_INIVAR_HDCINT "G2dHdcIntern"
```
Definition at line 262 of file TCSdSDLc.h.

7.32.2.133 TCS_INIVAR_HDCINTL

```
#define TCS_INIVAR_HDCINTL "G2dHdcInternL"
```
Definition at line 264 of file TCSdSDLc.h.

7.32.2.134 TCS_INIVAR_HDCNAM

```
#define TCS_INIVAR_HDCNAM "G2dHardcopy"
```
Definition at line 199 of file TCSdSDLc.h.

7.32.2.135 TCS_INIVAR_HDCOPN

```
#define TCS_INIVAR_HDCOPN "G2dHdcOpen"
```
Definition at line 254 of file TCSdSDLc.h.

7.32.2.136 TCS_INIVAR_HDCOPNL

```
#define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
```
Definition at line 256 of file TCSdSDLc.h.

7.32.2.137 TCS_INIVAR_HDCWRT

```
#define TCS_INIVAR_HDCWRT "G2dHdcWrite"
```
Definition at line 258 of file TCSdSDLc.h.

7.32.2.138 TCS_INIVAR_HDCWRTL

```
#define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
```
Definition at line 260 of file TCSdSDLc.h.

7.32.2.139 TCS_INIVAR_INI2

```
#define TCS_INIVAR_INI2 "G2d2xInitt"
```
Definition at line 326 of file TCSdSDLc.h.

7.32.2.140 TCS_INIVAR_INI2L

```
#define TCS_INIVAR_INI2L "G2d2xInittL"
```
Definition at line 328 of file TCSdSDLc.h.

7.32.2.141 TCS_INIVAR_JOUADD

```
#define TCS_INIVAR_JOUADD "G2dJouAdd"
```
Definition at line 298 of file TCSdSDLc.h.

7.32.2.142 TCS_INIVAR_JOUADDL

```
#define TCS_INIVAR_JOUADDL "G2dJouAddL"
```
Definition at line 300 of file TCSdSDLc.h.

7.32.2.143 TCS_INIVAR_JOUCLR

```
#define TCS_INIVAR_JOUCLR "G2dJouClr"
```
Definition at line 302 of file TCSdSDLc.h.

7.32.2.144 TCS_INIVAR_JOUCLRL

```
#define TCS_INIVAR_JOUCLRL "G2dJouClrL"
```
Definition at line 304 of file TCSdSDLc.h.

7.32.2.145 TCS_INIVAR_JOUCREATE

```
#define TCS_INIVAR_JOUCREATE "G2dJouCreate"
```
Definition at line 290 of file TCSdSDLc.h.

7.32.2.146 TCS_INIVAR_JOUCREATEL

```
#define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
```
Definition at line 292 of file TCSdSDLc.h.

7.32.2.147 TCS_INIVAR_JOUENTRY

```
#define TCS_INIVAR_JOUENTRY "G2dJouEntry"
```
Definition at line 294 of file TCSdSDLc.h.

7.32.2.148 TCS_INIVAR_JOUENTRYL

```
#define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
```
Definition at line 296 of file TCSdSDLc.h.

7.32.2.149 TCS_INIVAR_JOUUNKWN

```
#define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
```
Definition at line 306 of file TCSdSDLc.h.

7.32.2.150 TCS_INIVAR_JOUUNKWNL

```
#define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
```
Definition at line 308 of file TCSdSDLc.h.

7.32.2.151 TCS_INIVAR_LINCOL

```
#define TCS_INIVAR_LINCOL "G2dLinCol"
```
Definition at line 234 of file TCSdSDLc.h.

7.32.2.152 TCS_INIVAR_NOFNT

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

7.32.2.153 TCS_INIVAR_NOFNTFIL

```
#define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
```
Definition at line 246 of file TCSdSDLc.h.

7.32.2.154 TCS_INIVAR_NOFNTFILL

```
#define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
```
Definition at line 248 of file TCSdSDLc.h.

7.32.2.155 TCS_INIVAR_NOFNTL

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

7.32.2.156 TCS_INIVAR_STATNAM

```
#define TCS_INIVAR_STATNAM "G2dStatus"
```
Definition at line 197 of file TCSdSDLc.h.

7.32.2.157 TCS_INIVAR_STATPOSX

```
#define TCS_INIVAR_STATPOSX "G2dStatusPosX"
```
Definition at line 225 of file TCSdSDLc.h.

7.32.2.158 TCS_INIVAR_STATPOSY

```
#define TCS_INIVAR_STATPOSY "G2dStatusPosY"
```
Definition at line 227 of file TCSdSDLc.h.

7.32.2.159 TCS_INIVAR_STATSIZX

```
#define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
```
Definition at line 229 of file TCSdSDLc.h.

7.32.2.160 TCS_INIVAR_STATSIZY

```
#define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
```
Definition at line 231 of file TCSdSDLc.h.

7.32.2.161 TCS_INIVAR_SYSFONT

```
#define TCS_INIVAR_SYSFONT "G2dSystemFont"
```
Definition at line 215 of file TCSdSDLc.h.

7.32.2.162 TCS_INIVAR_TXTCOL

```
#define TCS_INIVAR_TXTCOL "G2dTxtCol"
```
Definition at line 236 of file TCSdSDLc.h.

7.32.2.163 TCS_INIVAR_UNKNAUDIO

```
#define TCS_INIVAR_UNKNAUDIO "G2dAudio"
```
Definition at line 318 of file TCSdSDLc.h.

7.32.2.164 TCS_INIVAR_UNKNAUDIOL

```
#define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
```
Definition at line 320 of file TCSdSDLc.h.

7.32.2.165 TCS_INIVAR_UNKNGRAPHCARD

```
#define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
```
Definition at line 242 of file TCSdSDLc.h.

7.32.2.166 TCS_INIVAR_UNKNGRAPHCARDL

```
#define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
```
Definition at line 244 of file TCSdSDLc.h.

7.32.2.167 TCS_INIVAR_USR

```
#define TCS_INIVAR_USR "G2dUser"
```
Definition at line 266 of file TCSdSDLc.h.

7.32.2.168 TCS_INIVAR_USR2

```
#define TCS_INIVAR_USR2 "G2dUser2"
```
Definition at line 322 of file TCSdSDLc.h.

7.32.2.169 TCS_INIVAR_USR2L

```
#define TCS_INIVAR_USR2L "G2dUser2L"
```
Definition at line 324 of file TCSdSDLc.h.

7.32.2.170 TCS_INIVAR_USRL

```
#define TCS_INIVAR_USRL "G2dUserL"
```
Definition at line 268 of file TCSdSDLc.h.

7.32.2.171 TCS_INIVAR_USRWRN

```
#define TCS_INIVAR_USRWRN "G2dPressAny"
```
Definition at line 274 of file TCSdSDLc.h.

7.32.2.172 TCS_INIVAR_USRWRNL

```
#define TCS_INIVAR_USRWRNL "G2dPressAnyL"
```
Definition at line 276 of file TCSdSDLc.h.

7.32.2.173 TCS_INIVAR_WINNAM

```
#define TCS_INIVAR_WINNAM "G2dGraphic"
```
Definition at line 195 of file TCSdSDLc.h.

7.32.2.174 TCS_INIVAR_WINPOSX

```
#define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
```
Definition at line 217 of file TCSdSDLc.h.

7.32.2.175 TCS_INIVAR_WINPOSY

```
#define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
```
Definition at line 219 of file TCSdSDLc.h.

7.32.2.176 TCS_INIVAR_WINSIZX

```
#define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
```
Definition at line 221 of file TCSdSDLc.h.

7.32.2.177 TCS_INIVAR_WINSIZY

```
#define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
```
Definition at line 223 of file TCSdSDLc.h.

7.32.2.178 TCS_INIVAR_XMLOPEN

```
#define TCS_INIVAR_XMLOPEN "G2dXMLopen"
```
Definition at line 314 of file TCSdSDLc.h.

7.32.2.179 TCS_INIVAR_XMLOPENL

```
#define TCS_INIVAR_XMLOPENL "G2dXMLerrorL"
```
Definition at line 316 of file TCSdSDLc.h.

7.32.2.180 TCS_INIVAR_XMLPARSER

```
#define TCS_INIVAR_XMLPARSER "G2dXMLerror"
```
Definition at line 310 of file TCSdSDLc.h.

7.32.2.181 TCS_INIVAR_XMLPARSERL

```
#define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
```
Definition at line 312 of file TCSdSDLc.h.

7.32.2.182 TCS_MESSAGELEN

```
#define TCS_MESSAGELEN 132
```
Definition at line 122 of file TCSdSDLc.h.

7.32.2.183 TCS_REL_CHR_HEIGHT

```
#define TCS_REL_CHR_HEIGHT 0.023f
```
Definition at line 118 of file TCSdSDLc.h.

7.32.2.184 TCS_STATWINDOW_NAME

```
#define TCS_STATWINDOW_NAME "System Messages"
```
Definition at line 198 of file TCSdSDLc.h.

7.32.2.185 TCS_WINDOW_NAME

```
#define TCS_WINDOW_NAME "Graphics"
```
Definition at line 196 of file TCSdSDLc.h.

7.32.2.186 TCS_WINDOW_NAMELEN

```
#define TCS_WINDOW_NAMELEN 50
```
Definition at line 120 of file TCSdSDLc.h.

7.32.2.187 TCSdrWIN__

```
#define TCSdrWIN__
```
Definition at line 378 of file TCSdSDLc.h.

7.32.2.188 tcslev3

```
#define tcslev3 tcslev3_
```
Definition at line 57 of file TCSdSDLc.h.

7.32.2.189 TEK_XMAX

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

7.32.2.190 TEK_YMAX

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

7.32.2.191 tinput

```
#define tinput tinput_
```
Definition at line 80 of file TCSdSDLc.h.

7.32.2.192 TKTRNX

```
#define TKTRNX tktrnx_ /* Fortran Naming Convention */
```
Definition at line 56 of file TCSdSDLc.h.

7.32.2.193 true

```
#define true !false
```
Definition at line 382 of file TCSdSDLc.h.

7.32.2.194 txtcol

```
#define txtcol txtcol_
```
Definition at line 71 of file TCSdSDLc.h.

7.32.2.195 winlbl

```
#define winlbl winlbl_
```
Definition at line 62 of file TCSdSDLc.h.

7.32.2.196 WRN_COPYLOCK

```
#define WRN_COPYLOCK 14
```
Definition at line 171 of file TCSdSDLc.h.

7.32.2.197 WRN_COPYNOMEM

```
#define WRN_COPYNOMEM 13
```
Definition at line 170 of file TCSdSDLc.h.

7.32.2.198 WRN_HDCFILOPN

```
#define WRN_HDCFILOPN 6
```
Definition at line 163 of file TCSdSDLc.h.

7.32.2.199 WRN_HDCFILWRT

```
#define WRN_HDCFILWRT 7
```
Definition at line 164 of file TCSdSDLc.h.

7.32.2.200 WRN_HDCINTERN

```
#define WRN_HDCINTERN 8
```
Definition at line 165 of file TCSdSDLc.h.

7.32.2.201 WRN_INI2

```
#define WRN_INI2 24
```
Definition at line 181 of file TCSdSDLc.h.

7.32.2.202 WRN_JOUADD

```
#define WRN_JOUADD 17
```
Definition at line 174 of file TCSdSDLc.h.

7.32.2.203 WRN_JOUCLR

```
#define WRN_JOUCLR 18
```
Definition at line 175 of file TCSdSDLc.h.

7.32.2.204 WRN_JOUCREATE

```
#define WRN_JOUCREATE 15
```
Definition at line 172 of file TCSdSDLc.h.

7.32.2.205 WRN_JOUENTRY

```
#define WRN_JOUENTRY 16
```
Definition at line 173 of file TCSdSDLc.h.

7.32.2.206 WRN_JOUUNKWN

```
#define WRN_JOUUNKWN 19
```
Definition at line 176 of file TCSdSDLc.h.

7.32.2.207 WRN_NOMSG

```
#define WRN_NOMSG 1
```
Definition at line 158 of file TCSdSDLc.h.

7.32.2.208 WRN_USRPRESSANY

```
#define WRN_USRPRESSANY 11
```
Definition at line 168 of file TCSdSDLc.h.

7.32.2.209 XACTION_ASCII

```
#define XACTION_ASCII 9
```
Definition at line 147 of file TCSdSDLc.h.

7.32.2.210 XACTION_BCKCOL

```
#define XACTION_BCKCOL 10
```
Definition at line 148 of file TCSdSDLc.h.

7.32.2.211 XACTION_DRWABS

```
#define XACTION_DRWABS 4
```
Definition at line 142 of file TCSdSDLc.h.

7.32.2.212 XACTION_DSHABS

```
#define XACTION_DSHABS 6
```
Definition at line 144 of file TCSdSDLc.h.

7.32.2.213 XACTION_DSHSTYLE

```
#define XACTION_DSHSTYLE 5
```
Definition at line 143 of file TCSdSDLc.h.

7.32.2.214 XACTION_ERASE

```
#define XACTION_ERASE 2
```
Definition at line 140 of file TCSdSDLc.h.

7.32.2.215 XACTION_FONTATTR

```
#define XACTION_FONTATTR 13
```
Definition at line 151 of file TCSdSDLc.h.

7.32.2.216 XACTION_GTEXT

```
#define XACTION_GTEXT 8
```
Definition at line 146 of file TCSdSDLc.h.

7.32.2.217 XACTION_INITT

```
#define XACTION_INITT 1
```
Definition at line 139 of file TCSdSDLc.h.

7.32.2.218 XACTION_LINCOL

```
#define XACTION_LINCOL 11
```
Definition at line 149 of file TCSdSDLc.h.

7.32.2.219 XACTION_MOVABS

```
#define XACTION_MOVABS 3
```
Definition at line 141 of file TCSdSDLc.h.

7.32.2.220 XACTION_NOOP

```
#define XACTION_NOOP 14
```
Definition at line 152 of file TCSdSDLc.h.

7.32.2.221 XACTION_PNTABS

```
#define XACTION_PNTABS 7
```
Definition at line 145 of file TCSdSDLc.h.

7.32.2.222 XACTION_TXTCOL

```
#define XACTION_TXTCOL 12
```
Definition at line 150 of file TCSdSDLc.h.

7.32.3 Typedef Documentation

7.32.3.1 bool

```
typedef int bool
```
Definition at line 380 of file TCSdSDLc.h.

7.32.3.2 FTNCHAR

```
typedef char FTNCHAR
```
Definition at line 41 of file TCSdSDLc.h.

7.32.3.3 FTNCHARLEN

```
typedef size_t FTNCHARLEN
```
Definition at line 44 of file TCSdSDLc.h.

7.32.3.4 FTNDOUBLE

```
typedef double FTNDOUBLE
```
Definition at line 38 of file TCSdSDLc.h.

7.32.3.5 FTNINT

```
typedef integer FTNINT
```
Definition at line 36 of file TCSdSDLc.h.

7.32.3.6 ftnlen

```
typedef size_t ftnlen
```
Definition at line 43 of file TCSdSDLc.h.

7.32.3.7 FTNREAL

```
typedef float FTNREAL
```
Definition at line 37 of file TCSdSDLc.h.

7.32.3.8 FTNSTRPAR

```
typedef FTNCHAR FTNSTRPAR
```
Definition at line 47 of file TCSdSDLc.h.

7.32.3.9 integer

```
typedef long int integer
```
Definition at line 33 of file TCSdSDLc.h.

7.32.3.10 logical

```
typedef long int logical
```
Definition at line 32 of file TCSdSDLc.h.

7.32.3.11 LOGICAL

```
typedef logical LOGICAL
```
Definition at line 35 of file TCSdSDLc.h.

7.32.4 Function Documentation

7.32.4.1 dcursr()

```
void dcursr (
            FTNINT * ic,
            FTNINT * ix,
            FTNINT * iy )
```
Definition at line 2116 of file TCSdSDLc.c.

7.32.4.2 GETARG()

```
FTNINT GETARG (
            FTNINT * iNo,
            FTNCHAR * line,
            FTNCHARLEN line_len )
```

7.32.4.3 GraphicError()

```
void GraphicError (
            FTNINT * iErr,
            FTNSTRPAR * ftn_string,
            FTNINT *iL  FTNSTRPAR_TAILftn_string )
```
Definition at line 2101 of file TCSdSDLc.c.

7.32.4.4 outtext()

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

Definition at line 2039 of file TCSdSDLc.c.

7.32.4.5 SUBSTITUTE()

```
void SUBSTITUTE (
            FTNSTRPAR * Src,
            FTNSTRPAR * Dst,
            FTNSTRPAR * old,
            FTNSTRPAR *new  FTNSTRPAR_TAILSrc) FTNSTRPAR_TAIL(Dst) FTNSTRPAR_TAIL(old) FTNST↩
RPAR_TAIL(new )
```

7.33 TCSdSDLc.h

```
00001 /** ****************************************************************************
00002 \file    TCSdSDLc.h
00003 \brief   SDL Port: Low-Level Driver
00004 \version 1.1
00005 \author  (C) 2022 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 Parameteruebergabe Fortran <-> C */
00031
00032 typedef long int logical; // 3 plattformabhaengige Definitionen
00033 typedef long int integer; // evtl. ueberpruefen
00034
00035 typedef logical LOGICAL;
00036 typedef integer FTNINT;
00037 typedef float FTNREAL;
00038 typedef double FTNDOUBLE;
00039 typedef struct {float real, imag;} FTNCOMPLEX;
00040
00041 typedef char FTNCHAR;
00042
00043 typedef size_t ftnlen;  // Ersatz fuer g2c.h
00044 typedef size_t FTNCHARLEN;
00045
00046 typedef struct { FTNCHAR * addr; FTNCHARLEN len; } FTNSTRDESC;
00047 typedef FTNCHAR FTNSTRPAR;
00048 #define FTNSTRPAR_TAIL(ftns) , FTNCHARLEN ftns##_len
00049 #define FTNSTRPARA(ftns) ftns
00050 #define FTNSTRPARL(ftns) ftns##_len
00051 #define CALLFTNSTRA(ftns) ftns.addr
00052 #define CALLFTNSTRL(ftns) , ftns.len
00053 #define FWRDFTNSTRA(ftns) ftns
00054 #define FWRDFTNSTRL(ftns) , ftns##_len
00055
00056 #define TKTRNX tktrnx_ /* Fortran Naming Convention */
00057 #define tcslev3 tcslev3_
00058 #define initt1 initt1_
00059 #define finitt finitt_
00060 #define iowait iowait_
```

```
00061 #define GraphicError graphicerror_
00062 #define winlbl winlbl_
00063 #define erase erase_
00064 #define swindl swindl_
00065 #define movabs movabs_
00066 #define drwabs drwabs_
00067 #define dshabs dshabs_
00068 #define pntabs pntabs_
00069 #define bckcol bckcol_
00070 #define lincol lincol_
00071 #define txtcol txtcol_
00072 #define DefaultColour defaultcolour_
00073 #define outgtext outgtext_
00074 #define italic italic_
00075 #define italir italir_
00076 #define dblsiz dblsiz_
00077 #define nrmsiz nrmsiz_
00078 #define bell bell_
00079 #define outtext outtext_
00080 #define tinput tinput_
00081 #define dcursr dcursr_
00082 #define csize csize_
00083 #define hdcopy hdcopy_
00084 #define lib_movc3 lib_movc3_
00085
00086 /* Deklarationen von durch C aufgerufenen FTN77-Unterprogrammen */
00087
00088 #define GETARG getarg_       // aus GNU F77-Library
00089 FTNINT GETARG (FTNINT *iNo, FTNCHAR *line, FTNCHARLEN line_len);
00090
00091 #define INITT2 initt2_
00092 void INITT2 (void);
00093
00094 #define SUBSTITUTE substitute_
00095 void SUBSTITUTE (FTNSTRPAR *Src, FTNSTRPAR *Dst, FTNSTRPAR *old, FTNSTRPAR *new
00096                                          FTNSTRPAR_TAIL(Src) FTNSTRPAR_TAIL(Dst)
00097                                          FTNSTRPAR_TAIL(old) FTNSTRPAR_TAIL(new));
00098
00099 /* Forward Deklarationen: Codiert in C und auch in C verwendet */
00100
00101 void bell (void); //  -> Forward Deklaration
00102 void GraphicError (FTNINT *iErr, FTNSTRPAR *ftn_string,
00103                    FTNINT *iL  FTNSTRPAR_TAIL(ftn_string));
00104 void outtext(FTNSTRPAR * ftn_string FTNSTRPAR_TAIL(ftn_string) );
00105 void dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy);
00106 void finitt ();
00107
00108
00109 /* Systemparameter */
00110
00111
00112
00113 /* ------------ Programmparameter ---------------------------------------- */
00114
00115
00116 #define STAT_MAXROWS 1             /* vorhandene Statuszeilen */
00117
00118 #define TCS_REL_CHR_HEIGHT 0.023f
00119
00120 #define TCS_WINDOW_NAMELEN 50
00121 #define TCS_FILE_NAMELEN 128
00122 #define TCS_MESSAGELEN 132
00123
00124 #define MAX_HDCCOUNT 1000          /* s.u.: Format TCS_HDCFILE_NAME */
00125
00126 #define INIFILEXTTOKEN ".%"        /* Token fuer den Filenamenparser */
00127 #define PROGDIRTOKEN "%:"
00128
00129 #define TCS_INIFILE_NAME "Graph2D"
00130
00131 #define SAMPLE_RATE 41000 // fuer SDL-Audioausgabe
00132 #define BELL_AMPLITUDE 32000.0
00133 #define BELL_FREQUENCY 441.0f
00134 #define BELL_DURATION 200
00135
00136
00137 /* Actioncodes des Journalfiles */
00138
00139 #define XACTION_INITT       1
00140 #define XACTION_ERASE       2
00141 #define XACTION_MOVABS      3
00142 #define XACTION_DRWABS      4
00143 #define XACTION_DSHSTYLE    5
00144 #define XACTION_DSHABS      6
00145 #define XACTION_PNTABS      7
00146 #define XACTION_GTEXT       8
00147 #define XACTION_ASCII       9
```

```
00148 #define XACTION_BCKCOL      10
00149 #define XACTION_LINCOL      11
00150 #define XACTION_TXTCOL      12
00151 #define XACTION_FONTATTR    13
00152 #define XACTION_NOOP        14
00153
00154
00155
00156 /* Zuordnung Fehlernummern zu Meldungen */
00157
00158 #define WRN_NOMSG 1
00159 #define ERR_UNKNGRAPHCARD 2
00160 #define ERR_NOFNTFIL 3
00161 #define ERR_NOFNT 4
00162 #define MSG_NOMOUSE 5
00163 #define WRN_HDCFILOPN 6
00164 #define WRN_HDCFILWRT 7
00165 #define WRN_HDCINTERN 8
00166 #define MSG_USR 9
00167 #define MSG_HDCACT 10
00168 #define WRN_USRPRESSANY 11
00169 #define ERR_EXIT 12
00170 #define WRN_COPYNOMEM 13
00171 #define WRN_COPYLOCK 14
00172 #define WRN_JOUCREATE 15
00173 #define WRN_JOUENTRY 16
00174 #define WRN_JOUADD 17
00175 #define WRN_JOUCLR 18
00176 #define WRN_JOUUNKWN 19
00177 #define ERR_XMLPARSER 20
00178 #define ERR_XMLOPEN 21
00179 #define ERR_UNKNAUDIO 22
00180 #define MSG_USR2 23
00181 #define WRN_INI2 24
00182 #define MSG_MAXERRNO 25
00183
00184
00185
00186 /* Initialisierungskonstanten *.INI, werden sinngemaess auch bei der
00187     Registry und XML-Initialisierung verwendet.
00188     Bei Erweiterungen Variableninitialisierung szTCSErrorMsg und TCSErrorLev
00189     in TCSdWINc.c fuer Registry und XML-Initialisierung nicht vergessen und
00190     alle Parser (*.ini, Registry und *.xml) beruecksichtigen! */
00191
00192 #define TCS_INISECT0 "Graph2D" // Root-Section, derzeit nur bei XML verwendet
00193
00194 #define TCS_INISECT1 "Names"
00195  #define TCS_INIVAR_WINNAM "G2dGraphic"
00196     #define TCS_WINDOW_NAME "Graphics"
00197  #define TCS_INIVAR_STATNAM "G2dStatus"
00198     #define TCS_STATWINDOW_NAME "System Messages"
00199  #define TCS_INIVAR_HDCNAM "G2dHardcopy"
00200     #if (JOURNALTYP ==1)
00201        #define TCS_HDCFILE_NAME "HDC%03i.WMF"
00202     #elif (JOURNALTYP ==2)
00203        #define TCS_HDCFILE_NAME "HDC%03i.EMF"
00204     #elif (JOURNALTYP ==3)
00205        #define TCS_HDCFILE_NAME "HDC%03i.HDC"
00206     #else
00207        #define TCS_HDCFILE_NAME "HDC%03i.UNKNOWN"
00208     #endif
00209
00210 #define TCS_INISECT2 "Layout"
00211  #define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
00212     #define TCS_INIDEF_COPMEN "Copy"
00213  #define TCS_INIVAR_FONT "G2dGraphicFont"
00214     #define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
00215  #define TCS_INIVAR_SYSFONT "G2dSystemFont"
00216     #define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
00217  #define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
00218     #define TCS_INIDEF_WINPOSX 1
00219  #define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
00220     #define TCS_INIDEF_WINPOSY 3
00221  #define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
00222     #define TCS_INIDEF_WINSIZX 98
00223  #define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
00224     #define TCS_INIDEF_WINSIZY 85
00225  #define TCS_INIVAR_STATPOSX "G2dStatusPosX"
00226     #define TCS_INIDEF_STATPOSX 1
00227  #define TCS_INIVAR_STATPOSY "G2dStatusPosY"
00228     #define TCS_INIDEF_STATPOSY 91
00229  #define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
00230     #define TCS_INIDEF_STATSIZX 98
00231  #define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
00232     #define TCS_INIDEF_STATSIZY 3   // mit X11 o.k.
00233 //    #define TCS_INIDEF_STATSIZY 0 // sonst nur 1 Fenster
00234  #define TCS_INIVAR_LINCOL "G2dLinCol"
```

```
00235     #define TCS_INIDEF_LINCOL 1
00236  #define TCS_INIVAR_TXTCOL "G2dTxtCol"
00237     #define TCS_INIDEF_TXTCOL 1
00238  #define TCS_INIVAR_BCKCOL "G2dBckCol"
00239     #define TCS_INIDEF_BCKCOL 0
00240
00241 #define TCS_INISECT3 "Messages"
00242  #define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
00243     #define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System: Error %s."
00244     #define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
00245     #define TCS_INIDEF_UNKNGRAPHCARDL 10
00246  #define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
00247     #define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
00248     #define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
00249     #define TCS_INIDEF_NOFNTFILL 10
00250  #define TCS_INIVAR_NOFNT "G2dFntfilOpen"
00251     #define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
00252     #define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
00253     #define TCS_INIDEF_NOFNTL 10
00254  #define TCS_INIVAR_HDCOPN "G2dHdcOpen"
00255     #define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
00256     #define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
00257     #define TCS_INIDEF_HDCOPNL 5
00258  #define TCS_INIVAR_HDCWRT "G2dHdcWrite"
00259     #define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
00260     #define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
00261     #define TCS_INIDEF_HDCWRTL 5
00262  #define TCS_INIVAR_HDCINT "G2dHdcIntern"
00263     #define TCS_INIDEF_HDCINT "GRAPH2D HARDCOPY: Internal Error."
00264     #define TCS_INIVAR_HDCINTL "G2dHdcInternL"
00265     #define TCS_INIDEF_HDCINTL 5
00266  #define TCS_INIVAR_USR "G2dUser"
00267     #define TCS_INIDEF_USR "%s"
00268     #define TCS_INIVAR_USRL "G2dUserL"
00269     #define TCS_INIDEF_USRL 5
00270  #define TCS_INIVAR_HDCACT "G2dHdcActive"
00271     #define TCS_INIDEF_HDCACT "Hardcopy in progress: File %s created."
00272     #define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
00273     #define TCS_INIDEF_HDCACTL 1
00274  #define TCS_INIVAR_USRWRN "G2dPressAny"
00275     #define TCS_INIDEF_USRWRN "Press any key to continue."
00276     #define TCS_INIVAR_USRWRNL "G2dPressAnyL"
00277     #define TCS_INIDEF_USRWRNL 5
00278  #define TCS_INIVAR_EXIT "G2dExit"
00279     #define TCS_INIDEF_EXIT "Press any key to exit program."
00280     #define TCS_INIVAR_EXITL "G2dExitL"
00281     #define TCS_INIDEF_EXITL 10
00282  #define TCS_INIVAR_COPMEM "G2dNoMemory"
00283     #define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager: Out of Memory."
00284     #define TCS_INIVAR_COPMEML "G2dNoMemoryL"
00285     #define TCS_INIDEF_COPMEML 1
00286  #define TCS_INIVAR_COPLCK "G2dClipLock"
00287     #define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager: ClipBoard locked."
00288     #define TCS_INIVAR_COPLCKL "G2dClipLockL"
00289     #define TCS_INIDEF_COPLCKL 1
00290  #define TCS_INIVAR_JOUCREATE "G2dJouCreate"
00291     #define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal. Error-No: %s."
00292     #define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
00293     #define TCS_INIDEF_JOUCREATEL 5
00294  #define TCS_INIVAR_JOUENTRY "G2dJouEntry"
00295     #define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
00296     #define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
00297     #define TCS_INIDEF_JOUENTRYL 5
00298  #define TCS_INIVAR_JOUADD "G2dJouAdd"
00299     #define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
00300     #define TCS_INIVAR_JOUADDL "G2dJouAddL"
00301     #define TCS_INIDEF_JOUADDL 5
00302  #define TCS_INIVAR_JOUCLR "G2dJouClr"
00303     #define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
00304     #define TCS_INIVAR_JOUCLRL "G2dJouClrL"
00305     #define TCS_INIDEF_JOUCLRL 5
00306  #define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
00307     #define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
00308     #define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
00309     #define TCS_INIDEF_JOUUNKWNL 5
00310  #define TCS_INIVAR_XMLPARSER "G2dXMLerror"
00311     #define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File: %s"
00312     #define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
00313     #define TCS_INIDEF_XMLPARSERL 8
00314  #define TCS_INIVAR_XMLOPEN "G2dXMLopen"
00315     #define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
00316     #define TCS_INIVAR_XMLOPENL "G2dXMLerrorL"
00317     #define TCS_INIDEF_XMLOPENL 8
00318  #define TCS_INIVAR_UNKNAUDIO "G2dAudio"
00319     #define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System: Error %s."
00320     #define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
00321     #define TCS_INIDEF_UNKNAUDIOL 5
```

```
00322  #define TCS_INIVAR_USR2 "G2dUser2"
00323     #define TCS_INIDEF_USR2 "%s"
00324     #define TCS_INIVAR_USR2L "G2dUser2L"
00325     #define TCS_INIDEF_USR2L 5
00326  #define TCS_INIVAR_INI2 "G2d2xInitt"
00327     #define TCS_INIDEF_INI2 "%s"
00328     #define TCS_INIVAR_INI2L "G2d2xInittL"
00329     #define TCS_INIDEF_INI2L 5
00330
00331
00332 /* ------------ Steuerung C++: Klassendefinition / C: Unterprogramme ------ */
00333
00334 #ifdef __cplusplus
00335
00336 class TCSdrWIN
00337 {
00338 public:
00339             TCSdrWIN();
00340   virtual   ~TCSdrWIN();
00341
00342             tcslev3 (FTNINT *SysLev);
00343             winlbl (FTNSTRDESC * PloWinNam, FTNSTRDESC * StatWinNam,
00344                 FTNSTRDESC * IniFilNam, FTNINT *hIcon, FTNINT hIn, FTNINT hPrevIn);
00345
00346             initt1 (HINSTANCE *hParentInstance);
00347             finitt ();
00348             erase ();
00349             swindo (FTNINT *ix,FTNINT *iLx, FTNINT *iy,FTNINT *iLy);
00350             swind1 (FTNINT *ix,FTNINT *iLx, FTNINT *iy,FTNINT *iLy);
00351             movabs (FTNINT *ix,FTNINT *iy);
00352             drwabs (FTNINT *ix,FTNINT *iy);
00353             dshabs (FTNINT *ix,FTNINT *iy, FTNINT *iMask);
00354             pntabs (FTNINT *ix,FTNINT *iy);
00355             bckcol (FTNINT *iCol);
00356             lincol (FTNINT *iCol);
00357             txtcol (FTNINT *iCol);
00358             DefaultColour ();
00359             outgtext(FTNSTRDESC * ftn_string);
00360             italic ();
00361             italir ();
00362             dblsiz ();
00363             nrmsiz ();
00364   static    bell ();
00365   static    outtext (FTNSTRDESC * ftn_string);
00366             tinput (FTNINT *ic);
00367             dcursr (FTNINT *ic,FTNINT *ix,FTNINT *iy);
00368             GraphicErrorMsg (FTNINT *iErr, FTNSTRDESC *ftn_string, FTNINT *iL);
00369             csize (FTNINT *ix,FTNINT *iy);
00370             hdcopy ();
00371             lib_movc3 (FTNINT *len,FTNSTRDESC *sou,FTNSTRDESC *dst);
00372 };
00373
00374  #define TCSdrWIN__ TCSdrWIN:: /* zur Vereinheitlichung C++ und C */
00375
00376 #else /* __cplusplus */
00377
00378  #define TCSdrWIN__
00379
00380  typedef int bool;
00381  #define false 0
00382  #define true !false
00383
00384 #endif /* not __cplusplus */
00385
```

7.34 Tktrnx.fd File Reference

SDL Port: TCS Common Block TKTRNX.

7.34.1 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.

7.35 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
```

7.36 TKTRNX.h File Reference

SDL Port: TCS Common Block TKTRNX.

Classes

- struct TKTRNXcommonBlock

Variables

- struct TKTRNXcommonBlock TKTRNX

7.36.1 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.

7.36.2 Variable Documentation

7.36.2.1 TKTRNX

```
struct TKTRNXcommonBlock TKTRNX
```

7.37 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