Graph2D Library --- wxWidgets ---

Generated by Doxygen 1.8.19

Chapter 1

Plot10 & Advanced Graphing II

Graph2D is written in Fortran2008/FTN77 and ANSI C++11/C90. Compilation instructions are available for Windows (MinGW) under "Additional Information".

1.0.0.1 How to build the library:

After copying the source files by "$getfiles.bat wx" into the /build subdirectory there are also the project files for CodeBlocks (Windows IDE) AND A LINUX BASHSCRIPT.

1.0.0.2 Using the library:

The main properties can be adjusted as follows:

- Initialization: By the WINLBL subroutine and/or ∗.xml files.
- Internationalization by GNU gettext
- Icons (Windows only): By linking a resource

1.0.0.3 Hardcopies

Default are proprietary ASCII-journalfiles with the default extension ∗.hdc. By choosing an other file extension bitmaps (∗.bmp) and jpgs (∗.jpg) are supported too.

Chapter 2

Application Programming Interface

2.0.1 Possible applications

The different use cases, from porting a DOS program to using several wX drawing windows, could be found in the example programs.

2.0.2 Using the initt1() subroutine for initialization

This subroutine is a pure C subroutine and is only available under wX. It is used to create one ore more drawing windows. If the WINLBL subroutine is used, it must be called before initt1(). The order is thus:

WINLBL() -> initt1() -> INITT()

2.0.2.1 Call

initt1 (int iMode, wxFrame* parent, wxFrame* FrameToUse, wxStatusBar* StatusBarToUse);

2.0.2.2 Parameters

2.0.2.2.1 iMode iMode= 0: From INITT(iDummy) with (0, nullptr, nullptr, nullptr) -> reset TCS and return

iMode= 1: from wxDemoFrame.cpp with (1, this, nullptr, nullptr) -> this is the parent window -> New window with status bar and title, size and position from TCS initialization

iMode= 2: from wxTCSmain.cpp with (2, nullptr, wxAppframe, nullptr) -> uses existing frame without parents and with new status bar.

iMode= 3: from wxDemoFrame.cpp with (3, this, (wxFrame*) Panel2, StatusBar1) -> uses existing panel and status bar. Passing a panel as a frame is allowed in Mode 3 because it only takes up another drawing panel and no specific frame methods are applied.

2.0.2.2.2 parent Parent window or NULL

2.0.2.2.3 FrameToUse existing frame or panel as a drawing area. Default: NULL

2.0.2.2.4 StatusBarToUse Existing status bar or NULL

Chapter 3

Compilersettings for Windows (MinGW)

3.0.1 Setup of the Windows IDE (TDM and CodeBlocks)

Install the TDM-Toolchain 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

Fortran2008 modules are stored in the CodeBlocks IDE via the preferences in the object directory. A recompilation cleans up this directory and deletes all $*$.mod files. -> Problematic when splitting the subroutine library to be developed (Graph2D) and the test program (wxDemo). Therefore, the IDE setting is changed so that the $*$.mod files are created next to the source files (GCC parameter -J):
Compiler Settings -> GlobalCompilerSettings -> SelectedCompiler:GNU FortranCompiler -> OtherSettings -> AdvancedOptions -> Command:CompileSingleFileToObjectFile -> -J $objects_output_dir -> DELETE!

For the test program located in a different folder, the parameter -J has to be set accordingly via the project settings. Without the prior deletion from the IDE settings, gfortran would be called with two contradictory -J parameters -> gfortran would then abort the compilation with an error message.

3.0.2 Settings for your own user programs

3.0.2.1 Linking of wX main programs

see example wxDemo, order is main program, teklib, wx, windows, gfortran:

```
<Add library="../libgraph2d.a" />\n


<Add library="libwxmsw31u.a" />\n
<Add library="libwxpng.a" />\n
<Add library="libwxjpeg.a" />\n
<Add library="libwxtiff.a" />\n
<Add library="libwxzlib.a" />\n
<Add library="libwxexpat.a" />\n


<Add library="libkernel32.a" />\n
<Add library="libuser32.a" />\n
<Add library="libgdi32.a" />\n
<Add library="libwinspool.a" />\n
<Add library="libcomdlg32.a" />\n
<Add library="libadvapi32.a" />\n
<Add library="libshell32.a" />\n
<Add library="libole32.a" />\n
<Add library="liboleaut32.a" />\n
<Add library="libuuid.a" />\n
<Add library="libcomctl32.a" />\n
<Add library="libwsock32.a" />\n
<Add library="libodbc32.a" />\n
<Add library="libshlwapi.a" />\n
<Add library="libversion.a" />\n
<Add library="liboleacc.a" />\n
<Add library="libuxtheme.a" />\n


<Add library="libgfortran.a" />\n
```

3.0.2.2 Linking Fortran Main Programs

The usual toolchain creates a sequential program without an event loop by calling the (automatically generated) start routine "main". The use of wX is not possible in this way, because all wX actions are based on event handlers. In addition, an overlay by a C++ program by IMPLEMENT_APP is impossible in case the object file of the main program does contain a definition of the symbol "main".

Possible approaches:

- If it is possible to change the main program in the source code (see project D2):

1. Change (or add if "PROGRAM" is not present) the statement "PROGRAM [name]" to "SUBROUTINE Ftn↩ Main2sub".
2. Change the main program end from "STOP / END" to "RETURN / END" or just "END" (see above)
3. Link with wxTCSmain2sub.cpp as the main program

- If changing the main program to a subroutine is not possible/desired and the main program is only closed with "END" and not "STOP / END" (see D1):

1. Separate compilation of the main program and then editing the object file: gcc -c ag2demo1.for
2. Delete the main symbol with the GNU binutil "STRIP -Nmain ag2demo1.o"
3. The entry into the main program "MAIN__" has to be globally visible: objcopy –globalize-symbol=MAIN__ obj/ag2demo1.o
4. Step 2 and 3 summarized: objcopy –strip-symbol=main –globalize-symbol=MAIN__ obj/ag2demo1.o

5. Link with wxTCSmain.cpp as the main program
6. Note: wxTCSmain is GNU compiler specific, other compilers have not been considered.

- If the main program is not changed, but terminated with "STOP/END" (see D3):

1. Generation of assembly code of the main program:

   From the Fortran code: gcc -S -m64 ag2demo3.for
   From the object code: objdump -D -Mx86-64 –no-addresses –no-show-raw obj/ag2demo3.o > ag2demo3.s

2. Changes in. ag2demo3.s (see ag2demo3changed.s):
   Add: . .globl MAIN__
   Delete: call _gfortran_stop_string and surrounding
   Add: addq $152, rsp ; $152 size from seh_stackalloc
   Add: popq rbx, popq rbp, ret ; see prologue
   Delete: .def __main -> seh_endproc main ; main and surrounding

3. Linking with wxTCSmain.cpp as the main program and MAIN__ from ag2demo3changed.s and not ag2demo3.for

4. Note: all adaptations are GNU compiler specific, other compilers have not been considered.

Chapter 4

Compilersettings for Linux

4.0.1 tbd.

Chapter 5

Data Type Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Data Type Index

6.1 Data Types List

Here are the data types with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Data Type Documentation

8.1 cTCScanvas Class Reference

Public Member Functions

- cTCScanvas (int iMode, wxFrame *parent, wxFrame *FrameToUse, wxStatusBar *StatusBarToUse)
- virtual ~cTCScanvas ()

Public Attributes

- wxFrame * TCSframe
- wxPanel * TCSpanel
- wxLogWindow * logWindow
- wxStatusBar * TCSstatusBar
- wxWindowID ID_TCSframe
- wxWindowID ID_TCSpanel
- wxWindowID ID_TCSstatus
- wxPen TCSpen
- wxBrush TCSbrush
- wxFont TCSfont
- bool ClippingNotActive = true
- int TCSpanelKeyPressed
- int TCSmouseButtonDown
- int TCSmouseX
- int TCSmouseY
- xJournalEntry_typ * xTCSJournal = NULL
- struct TKTRNX TekSav
- struct G2dAG2 AG2Sav
- int DefaultLinColSav
- int DefaultTxtColSav
- int DefaultBckColSav
- char HardcopyFileSav [TCS_FILE_NAMELEN]
- char sect0Sav [TCS_FILE_NAMELEN]

8.1.1 Detailed Description

Definition at line 83 of file TCSdrWXcpp.cpp.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 cTCScanvas()

```
cTCScanvas::cTCScanvas (
            int iMode,
            wxFrame * parent,
            wxFrame * FrameToUse,
            wxStatusBar * StatusBarToUse )
```

Definition at line 848 of file TCSdrWXcpp.cpp.

8.1.2.2 ~cTCScanvas()

```
cTCScanvas::~cTCScanvas ( )  [virtual]
```

Definition at line 929 of file TCSdrWXcpp.cpp.

8.1.3 Member Data Documentation

8.1.3.1 AG2Sav

```
struct G2dAG2 cTCScanvas::AG2Sav
```

Definition at line 104 of file TCSdrWXcpp.cpp.

8.1.3.2 ClippingNotActive

```
bool cTCScanvas::ClippingNotActive = true
```

Definition at line 100 of file TCSdrWXcpp.cpp.

8.1.3.3 DefaultBckColSav

```
int cTCScanvas::DefaultBckColSav
```

Definition at line 108 of file TCSdrWXcpp.cpp.

8.1.3.4 DefaultLinColSav

```
int cTCScanvas::DefaultLinColSav
```

Definition at line 108 of file TCSdrWXcpp.cpp.

8.1.3.5 DefaultTxtColSav

```
int cTCScanvas::DefaultTxtColSav
```

Definition at line 108 of file TCSdrWXcpp.cpp.

8.1.3.6 HardcopyFileSav

```
char cTCScanvas::HardcopyFileSav[TCS_FILE_NAMELEN]
```

Definition at line 109 of file TCSdrWXcpp.cpp.

8.1.3.7 ID_TCSframe

```
wxWindowID cTCScanvas::ID_TCSframe
```

Definition at line 92 of file TCSdrWXcpp.cpp.

8.1.3.8 ID_TCSpanel

```
wxWindowID cTCScanvas::ID_TCSpanel
```

Definition at line 93 of file TCSdrWXcpp.cpp.

8.1.3.9 ID_TCSstatus

```
wxWindowID cTCScanvas::ID_TCSstatus
```

Definition at line 94 of file TCSdrWXcpp.cpp.

8.1.3.10 logWindow

```
wxLogWindow* cTCScanvas::logWindow
```

Definition at line 89 of file TCSdrWXcpp.cpp.

8.1.3.11 sect0Sav

```
char cTCScanvas::sect0Sav[TCS_FILE_NAMELEN]
```

Definition at line 109 of file TCSdrWXcpp.cpp.

8.1.3.12 TCSbrush

```
wxBrush cTCScanvas::TCSbrush
```

Definition at line 97 of file TCSdrWXcpp.cpp.

8.1.3.13 TCSfont

```
wxFont cTCScanvas::TCSfont
```

Definition at line 98 of file TCSdrWXcpp.cpp.

8.1.3.14 TCSframe

```
wxFrame* cTCScanvas::TCSframe
```

Definition at line 87 of file TCSdrWXcpp.cpp.

8.1.3.15 TCSmouseButtonDown

```
int cTCScanvas::TCSmouseButtonDown
```

Definition at line 102 of file TCSdrWXcpp.cpp.

8.1.3.16 TCSmouseX

```
int cTCScanvas::TCSmouseX
```

Definition at line 102 of file TCSdrWXcpp.cpp.

8.1.3.17 TCSmouseY

```
int cTCScanvas::TCSmouseY
```

Definition at line 102 of file TCSdrWXcpp.cpp.

8.1.3.18 TCSpanel

```
wxPanel* cTCScanvas::TCSpanel
```

Definition at line 88 of file TCSdrWXcpp.cpp.

8.1.3.19 TCSpanelKeyPressed

```
int cTCScanvas::TCSpanelKeyPressed
```

Definition at line 101 of file TCSdrWXcpp.cpp.

8.1.3.20 TCSpen

```
wxPen cTCScanvas::TCSpen
```

Definition at line 96 of file TCSdrWXcpp.cpp.

8.1.3.21 TCSstatusBar

```
wxStatusBar* cTCScanvas::TCSstatusBar
```

Definition at line 90 of file TCSdrWXcpp.cpp.

8.1.3.22 TekSav

```
struct TKTRNX cTCScanvas::TekSav
```

Definition at line 104 of file TCSdrWXcpp.cpp.

8.1.3.23 xTCSJournal

```
xJournalEntry_typ* cTCScanvas::xTCSJournal = NULL
```

Definition at line 104 of file TCSdrWXcpp.cpp.

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

- TCSdrWXcpp.cpp

8.2 TKTRNX Struct Reference

```
#include <TKTRNX.hpp>
```

Public Attributes

- int khomey
- int khorsz
- int kversz
- int kitalc
- int ksizef
- int klmrgn
- int krmrgn
- int kScrX
- int kScrY
- int kbeamx
- int kbeamy
- int kminsx
- int kminsy
- int kmaxsx
- int kmaxsy
- float tminvx
- float tminvy
- float tmaxvx
- float tmaxvy
- float trcosf
- float trsinf
- float trscal
- float xfac
- float yfac
- float xlog
- float ylog
- int kStCol
- int iLinCol
- int iBckCol
- int iTxtCol

8.2.1 Detailed Description

Definition at line 18 of file TKTRNX.hpp.

8.2.2 Member Data Documentation

8.2.2.1 iBckCol

```
int TKTRNX::iBckCol
```

Definition at line 33 of file TKTRNX.hpp.

8.2.2.2 iLinCol

```
int TKTRNX::iLinCol
```

Definition at line 33 of file TKTRNX.hpp.

8.2.2.3 iTxtCol

```
int TKTRNX::iTxtCol
```

Definition at line 33 of file TKTRNX.hpp.

8.2.2.4 kbeamx

```
int TKTRNX::kbeamx
```

Definition at line 24 of file TKTRNX.hpp.

8.2.2.5 kbeamy

```
int TKTRNX::kbeamy
```

Definition at line 24 of file TKTRNX.hpp.

8.2.2.6 khomey

```
int TKTRNX::khomey
```

Definition at line 20 of file TKTRNX.hpp.

8.2.2.7 khorsz

```
int TKTRNX::khorsz
```

Definition at line 21 of file TKTRNX.hpp.

8.2.2.8 kitalc

```
int TKTRNX::kitalc
```

Definition at line 22 of file TKTRNX.hpp.

8.2.2.9 klmrgn

```
int TKTRNX::klmrgn
```

Definition at line 23 of file TKTRNX.hpp.

8.2.2.10 kmaxsx

```
int TKTRNX::kmaxsx
```

Definition at line 25 of file TKTRNX.hpp.

8.2.2.11 kmaxsy

```
int TKTRNX::kmaxsy
```

Definition at line 25 of file TKTRNX.hpp.

8.2.2.12 kminsx

```
int TKTRNX::kminsx
```

Definition at line 25 of file TKTRNX.hpp.

8.2.2.13 kminsy

```
int TKTRNX::kminsy
```

Definition at line 25 of file TKTRNX.hpp.

8.2.2.14 krmrgn

```
int TKTRNX::krmrgn
```

Definition at line 23 of file TKTRNX.hpp.

8.2.2.15 kScrX

```
int TKTRNX::kScrX
```

Definition at line 23 of file TKTRNX.hpp.

8.2.2.16 kScrY

```
int TKTRNX::kScrY
```

Definition at line 23 of file TKTRNX.hpp.

8.2.2.17 ksizef

```
int TKTRNX::ksizef
```

Definition at line 22 of file TKTRNX.hpp.

8.2.2.18 kStCol

```
int TKTRNX::kStCol
```

Definition at line 32 of file TKTRNX.hpp.

8.2.2.19 kversz

```
int TKTRNX::kversz
```

Definition at line 21 of file TKTRNX.hpp.

8.2.2.20 tmaxvx

```
float TKTRNX::tmaxvx
```

Definition at line 28 of file TKTRNX.hpp.

8.2.2.21 tmaxvy

```
float TKTRNX::tmaxvy
```

Definition at line 28 of file TKTRNX.hpp.

8.2.2.22 tminvx

```
float TKTRNX::tminvx
```

Definition at line 28 of file TKTRNX.hpp.

8.2.2.23 tminvy

```
float TKTRNX::tminvy
```

Definition at line 28 of file TKTRNX.hpp.

8.2.2.24 trcosf

```
float TKTRNX::trcosf
```

Definition at line 29 of file TKTRNX.hpp.

8.2.2.25 trscal

```
float TKTRNX::trscal
```

Definition at line 29 of file TKTRNX.hpp.

8.2.2.26 trsinf

```
float TKTRNX::trsinf
```

Definition at line 29 of file TKTRNX.hpp.

8.2.2.27 xfac

```
float TKTRNX::xfac
```

Definition at line 30 of file TKTRNX.hpp.

8.2.2.28 xlog

```
float TKTRNX::xlog
```

Definition at line 30 of file TKTRNX.hpp.

8.2.2.29 yfac

```
float TKTRNX::yfac
```

Definition at line 30 of file TKTRNX.hpp.

8.2.2.30 ylog

```
float TKTRNX::ylog
```

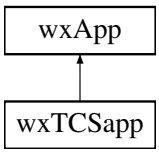
Definition at line 30 of file TKTRNX.hpp.

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

- TKTRNX.hpp

8.3 wxTCSapp Class Reference

Inheritance diagram for wxTCSapp:



Public Member Functions

- virtual bool OnInit ()
- virtual void OnIdle ()

8.3.1 Detailed Description

Definition at line 39 of file wxTCSmain.cpp.

8.3.2 Member Function Documentation

8.3.2.1 OnIdle()

```
void wxTCSapp::OnIdle ( )  [virtual]
```

Definition at line 75 of file wxTCSmain.cpp.

8.3.2.2 OnInit()

```
bool wxTCSapp::OnInit ( )  [virtual]
```

Definition at line 51 of file wxTCSmain.cpp.

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

- wxTCSmain.cpp

8.4 xJournalEntry_typ Struct Reference

Public Attributes

- struct xJournalEntry_typ * previous
- struct xJournalEntry_typ * next
- int action
- int i1
- int i2

8.4.1 Detailed Description

Definition at line 77 of file TCSdrWXcpp.cpp.

8.4.2 Member Data Documentation

8.4.2.1 action

```
int xJournalEntry_typ::action
```

Definition at line 79 of file TCSdrWXcpp.cpp.

8.4.2.2 i1

```
int xJournalEntry_typ::i1
```

Definition at line 79 of file TCSdrWXcpp.cpp.

8.4.2.3 i2

```
int xJournalEntry_typ::i2
```

Definition at line 79 of file TCSdrWXcpp.cpp.

8.4.2.4 next

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

Definition at line 78 of file TCSdrWXcpp.cpp.

8.4.2.5 previous

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

Definition at line 77 of file TCSdrWXcpp.cpp.

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

- TCSdrWXcpp.cpp

Chapter 9

File Documentation

9.1 AG2.for File Reference

Graph2D: Tektronix Advanced Graphing II Emulation.

Functions/Subroutines

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

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

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

9.1.1 Detailed Description

Graph2D: Tektronix Advanced Graphing II Emulation.

Version

(2025,347, x)

Author

(C) 2025 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Layer 2: scientific 2-D graphic subroutines

Note

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

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

Definition in file AG2.for.

9.1.2 Function/Subroutine Documentation

9.1.2.1 ag2infin()

```
real function ag2infin
```

Definition at line 155 of file AG2.for.

9.1.2.2 ag2lev()

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

Definition at line 94 of file AG2.for.

9.1.2.3 alfsetc()

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

Definition at line 2574 of file AG2.for.

9.1.2.4 bar()

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

Definition at line 1698 of file AG2.for.

9.1.2.5 binitt()

```
subroutine binitt
```

Definition at line 724 of file AG2.for.

9.1.2.6 bsyms()

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

Definition at line 1850 of file AG2.for.

9.1.2.7 calcon()

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

Definition at line 1336 of file AG2.for.

9.1.2.8 calpnt()

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

Definition at line 1281 of file AG2.for.

9.1.2.9 check()

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

Definition at line 808 of file AG2.for.

9.1.2.10 cmnmx()

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

Definition at line 930 of file AG2.for.

9.1.2.11 coptim()

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

Definition at line 1125 of file AG2.for.

9.1.2.12 cplot()

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

Definition at line 1548 of file AG2.for.

9.1.2.13 datget()

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

Definition at line 1670 of file AG2.for.

9.1.2.14 dinitx()

```
subroutine dinitx
```

Definition at line 654 of file AG2.for.

9.1.2.15 dinity()

```
subroutine dinity
```

Definition at line 668 of file AG2.for.

9.1.2.16 dlimx()

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

Definition at line 474 of file AG2.for.

9.1.2.17 dlimy()

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

Definition at line 486 of file AG2.for.

9.1.2.18 dsplay()

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

Definition at line 1534 of file AG2.for.

9.1.2.19 eformc()

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

Definition at line 2445 of file AG2.for.

9.1.2.20 esplit()

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

Definition at line 2478 of file AG2.for.

9.1.2.21 expoutc()

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

Definition at line 2498 of file AG2.for.

9.1.2.22 fformc()

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

Definition at line 2385 of file AG2.for.

9.1.2.23 filbox()

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

Definition at line 1765 of file AG2.for.

9.1.2.24 findge()

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

Definition at line 2933 of file AG2.for.

9.1.2.25 findle()

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

Definition at line 2952 of file AG2.for.

9.1.2.26 fonlyc()

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

Definition at line 2414 of file AG2.for.

9.1.2.27 frame()

```
subroutine frame
```

Definition at line 1520 of file AG2.for.

9.1.2.28 gline()

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

Definition at line 2183 of file AG2.for.

9.1.2.29 grid()

```
subroutine grid
```

Definition at line 1966 of file AG2.for.

9.1.2.30 hbarst()

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

Definition at line 682 of file AG2.for.

9.1.2.31 iformc()

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

Definition at line 2353 of file AG2.for.

9.1.2.32 infin()

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

Definition at line 142 of file AG2.for.

9.1.2.33 iother()

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

Definition at line 3077 of file AG2.for.

9.1.2.34 iubgc()

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

Definition at line 1483 of file AG2.for.

9.1.2.35 justerc()

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

Definition at line 2677 of file AG2.for.

9.1.2.36 keyset()

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

Definition at line 1644 of file AG2.for.

9.1.2.37 label()

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

Definition at line 2210 of file AG2.for.

9.1.2.38 leap()

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

Definition at line 1469 of file AG2.for.

9.1.2.39 line()

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

Definition at line 109 of file AG2.for.

9.1.2.40 locge()

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

Definition at line 2974 of file AG2.for.

9.1.2.41 locle()

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

Definition at line 2992 of file AG2.for.

9.1.2.42 logtix()

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

Definition at line 2052 of file AG2.for.

9.1.2.43 loptim()

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

Definition at line 998 of file AG2.for.

9.1.2.44 lwidth()

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

Definition at line 2743 of file AG2.for.

9.1.2.45 mnmx()

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

Definition at line 891 of file AG2.for.

9.1.2.46 monpos()

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

Definition at line 2169 of file AG2.for.

9.1.2.47 notatec()

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

Definition at line 2629 of file AG2.for.

9.1.2.48 npts()

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

Definition at line 165 of file AG2.for.

9.1.2.49 numsetc()

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

Definition at line 2326 of file AG2.for.

9.1.2.50 optim()

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

Definition at line 981 of file AG2.for.

9.1.2.51 oubgc()

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

Definition at line 1497 of file AG2.for.

9.1.2.52 place()

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

Definition at line 522 of file AG2.for.

9.1.2.53 remlab()

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

Definition at line 2818 of file AG2.for.

9.1.2.54 rescom()

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

Definition at line 3061 of file AG2.for.

9.1.2.55 rgchek()

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

Definition at line 864 of file AG2.for.

9.1.2.56 roundd()

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

Definition at line 3010 of file AG2.for.

9.1.2.57 roundu()

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

Definition at line 3026 of file AG2.for.

9.1.2.58 savcom()

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

Definition at line 3045 of file AG2.for.

9.1.2.59 setwin()

```
subroutine setwin
```

Definition at line 632 of file AG2.for.

9.1.2.60 sizel()

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

Definition at line 198 of file AG2.for.

9.1.2.61 sizes()

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

Definition at line 187 of file AG2.for.

9.1.2.62 slimx()

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

Definition at line 498 of file AG2.for.

9.1.2.63 slimy()

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

Definition at line 510 of file AG2.for.

9.1.2.64 spread()

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

Definition at line 2881 of file AG2.for.

9.1.2.65 stepl()

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

Definition at line 176 of file AG2.for.

9.1.2.66 steps()

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

Definition at line 131 of file AG2.for.

9.1.2.67 symbl()

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

Definition at line 120 of file AG2.for.

9.1.2.68 symout()

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

Definition at line 1867 of file AG2.for.

9.1.2.69 teksym()

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

Definition at line 1892 of file AG2.for.

9.1.2.70 teksym1()

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

Definition at line 1940 of file AG2.for.

9.1.2.71 tset()

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

Definition at line 2099 of file AG2.for.

9.1.2.72 tset2()

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

Definition at line 2137 of file AG2.for.

9.1.2.73 typck()

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

Definition at line 833 of file AG2.for.

9.1.2.74 vbarst()

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

Definition at line 702 of file AG2.for.

9.1.2.75 vlablc()

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

Definition at line 2654 of file AG2.for.

9.1.2.76 width()

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

Definition at line 2702 of file AG2.for.

9.1.2.77 xden()

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

Definition at line 322 of file AG2.for.

9.1.2.78 xetyp()

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

Definition at line 606 of file AG2.for.

9.1.2.79 xfrm()

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

Definition at line 400 of file AG2.for.

9.1.2.80 xlab()

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

Definition at line 300 of file AG2.for.

9.1.2.81 xlen()

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

Definition at line 374 of file AG2.for.

9.1.2.82 xloc()

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

Definition at line 256 of file AG2.for.

9.1.2.83 xloctp()

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

Definition at line 278 of file AG2.for.

9.1.2.84 xmfrm()

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

Definition at line 448 of file AG2.for.

9.1.2.85 xmtcs()

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

Definition at line 426 of file AG2.for.

9.1.2.86 xneat()

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

Definition at line 212 of file AG2.for.

9.1.2.87 xtics()

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

Definition at line 352 of file AG2.for.

9.1.2.88 xtype()

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

Definition at line 554 of file AG2.for.

9.1.2.89 xwdth()

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

Definition at line 580 of file AG2.for.

9.1.2.90 xzero()

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

Definition at line 234 of file AG2.for.

9.1.2.91 yden()

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

Definition at line 337 of file AG2.for.

9.1.2.92 yetyp()

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

Definition at line 619 of file AG2.for.

9.1.2.93 yfrm()

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

Definition at line 413 of file AG2.for.

9.1.2.94 ylab()

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

Definition at line 311 of file AG2.for.

9.1.2.95 ylen()

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

Definition at line 387 of file AG2.for.

9.1.2.96 yloc()

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

Definition at line 267 of file AG2.for.

9.1.2.97 ylocrt()

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

Definition at line 289 of file AG2.for.

9.1.2.98 ymdyd()

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

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

Definition at line 1414 of file AG2.for.

9.1.2.99 ymfrm()

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

Definition at line 461 of file AG2.for.

9.1.2.100 ymtcs()

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

Definition at line 437 of file AG2.for.

9.1.2.101 yneat()

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

Definition at line 223 of file AG2.for.

9.1.2.102 ytics()

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

Definition at line 363 of file AG2.for.

9.1.2.103 ytype()

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

Definition at line 567 of file AG2.for.

9.1.2.104 ywdth()

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

Definition at line 593 of file AG2.for.

9.1.2.105 yzero()

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

Definition at line 245 of file AG2.for.

9.2 AG2.for

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.3 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

9.3.1 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.

9.3.2 Function/Subroutine Documentation

9.3.2.1 alfset()

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

Definition at line 45 of file AG2Holerith.for.

9.3.2.2 comdmp()

```
subroutine comdmp
```

Definition at line 328 of file AG2Holerith.for.

9.3.2.3 comget()

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

Definition at line 271 of file AG2Holerith.for.

9.3.2.4 comset()

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

Definition at line 299 of file AG2Holerith.for.

9.3.2.5 eform()

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

Definition at line 173 of file AG2Holerith.for.

9.3.2.6 expout()

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

Definition at line 90 of file AG2Holerith.for.

9.3.2.7 fform()

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

Definition at line 189 of file AG2Holerith.for.

9.3.2.8 fonly()

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

Definition at line 205 of file AG2Holerith.for.

9.3.2.9 hlabel()

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

Definition at line 121 of file AG2Holerith.for.

9.3.2.10 hstrin()

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

Definition at line 112 of file AG2Holerith.for.

9.3.2.11 ibasec()

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

Definition at line 241 of file AG2Holerith.for.

9.3.2.12 ibasex()

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

Definition at line 251 of file AG2Holerith.for.

9.3.2.13 ibasey()

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

Definition at line 261 of file AG2Holerith.for.

9.3.2.14 iform()

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

Definition at line 221 of file AG2Holerith.for.

9.3.2.15 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.

9.3.2.16 notate()

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

Definition at line 30 of file AG2Holerith.for.

9.3.2.17 numset()

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

Definition at line 67 of file AG2Holerith.for.

9.3.2.18 vlabel()

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

Definition at line 139 of file AG2Holerith.for.

9.3.2.19 vstrin()

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

Definition at line 130 of file AG2Holerith.for.

9.4 AG2Holerith.for

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

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

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

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

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

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

9.5 AG2uline.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine uline (x, y, i)

9.5.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2uline.for.

9.5.2 Function/Subroutine Documentation

9.5.2.1 uline()

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

Definition at line 10 of file AG2uline.for.

9.6 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
```

9.7 AG2umnmx.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine umnmx (array, amin, amax)

9.7.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2umnmx.for.

9.7.2 Function/Subroutine Documentation

9.7.2.1 umnmx()

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

Definition at line 9 of file AG2umnmx.for.

9.8 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
```

9.9 AG2upoint.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- real function upoint (arr, ii, oldone)

9.9.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2upoint.for.

9.9.2 Function/Subroutine Documentation

9.9.2.1 upoint()

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

Definition at line 9 of file AG2upoint.for.

9.10 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
```

9.11 AG2users.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine users (x, y, i)

9.11.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2users.for.

9.11.2 Function/Subroutine Documentation

9.11.2.1 users()

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

Definition at line 9 of file AG2users.for.

9.12 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
```

9.13 AG2useset.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

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

9.13.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2useset.for.

9.13.2 Function/Subroutine Documentation

9.13.2.1 useset()

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

Definition at line 9 of file AG2useset.for.

9.14 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
```

9.15 AG2usesetC.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

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

9.15.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2usesetC.for.

9.15.2 Function/Subroutine Documentation

9.15.2.1 usesetc()

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

Definition at line 9 of file AG2usesetC.for.

9.16 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
```

9.17 AG2UsrSoftek.for File Reference

Graph2D: Dummy User Routine.

Functions/Subroutines

- subroutine softek (isym)

9.17.1 Detailed Description

Graph2D: Dummy User Routine.

Definition in file AG2UsrSoftek.for.

9.17.2 Function/Subroutine Documentation

9.17.2.1 softek()

```
subroutine softek (
              isym )
```

Definition at line 9 of file AG2UsrSoftek.for.

9.18 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
```

9.19 G2dAG2.fd File Reference

Graph2D: AG2 Common Block G2dAG2.

9.19.1 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.

9.20 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
```

9.21 GetHDC.for File Reference

Restore Hardcopies.

Functions/Subroutines

- logical function gethdc (Filnam)

9.21.1 Detailed Description

Restore Hardcopies.

Version

1.2

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Read and plot hardcopies
Temporary input unit: 41. If already used, an other channel will be searched.

Definition in file GetHDC.for.

9.21.2 Function/Subroutine Documentation

9.21.2.1 gethdc()

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

Parameters

FilNam	Hardcopyfie

Returns

(optional) .true. -> Error

Definition at line 15 of file GetHDC.for.

9.22 GetHDC.for

```
00001 C> \file       GetHDC.for
00002 C> \brief      Restore Hardcopies
00003 C> \version    1.2
00004 C> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Einlesen und Zeichnen von Hardcopydateien\n
00008 C> Verwendete temporaeres Ein/Ausgabeunit: 41. Falls bereits belegt, wird ein freier Kanal gesucht
00009 C> \~english
00010 C> Read and plot hardcopies\n
00011 C> Temporary input unit: 41. If already used, an other channel will be searched.
00012 C> \~
00013 C
00014
00015       logical function gethdc (Filnam)
00016 C> \param FilNam: Hardcopyfie
00017 C> \result (optional) .true. -> Error
00018       include 'Tktrnx.fd'
00019       integer tcs_messagelen, iunit
00020       parameter(tcs_messagelen=132)
00021       character *(*) filnam
00022       logical iunitused
00023       character *(TCS_MESSAGELEN+1) txtstring
```

```
00024
00025       integer ios, idash, iprntlen, iactlen
00026       integer action, i1, i2
00027
00028       iunit= 40
00029       gethdc= .true.
00030
00031   5   continue ! repeat
00032         iunit= iunit+1
00033         inquire (unit=iunit, opened= iunitused)
00034       if (iunitused) goto 5
00035
00036       open (iunit,file=filnam,status='old',iostat=ios,form='formatted')
00037       if (ios.ne.0) then
00038         call graphicerror (6, ' ')
00039         return
00040       end if
00041
00042   10  continue ! repeat
00043         read (iunit, fmt='(i2,1x,i4,1x,i3)', iostat=ios)action, i1, i2
00044         if (ios.gt.0) then ! Error, not EOF
00045          call graphicerror (8, ' ')
00046          return
00047         end if
00048         if (action.eq.1) then ! XACTION_INITT
00049           call defaultcolour()
00050           call erase ()
00051         else if (action.eq.2) then ! XACTION_ERASE
00052           call erase ()
00053         else if (action.eq.3) then ! XACTION_MOVABS
00054           call movabs (i1,i2)
00055         else if (action.eq.4) then ! XACTION_DRWABS
00056           call drwabs (i1,i2)
00057         else if (action.eq.5) then ! XACTION_DSHSTYLE
00058           idash= i1
00059         else if (action.eq.6) then ! XACTION_DSHABS
00060           call dshabs (i1,i2,idash)
00061         else if (action.eq.7) then ! XACTION_PNTABS
00062           call pntabs (i1,i2)
00063         else if (action.eq.8) then ! XACTION_GTEXT
00064           iprntlen= i1
00065           if (iprntlen.gt.tcs_messagelen) iprntlen= tcs_messagelen
00066           txtstring(1:1)= char(i2)
00067           if (iprntlen.eq.1) then
00068             txtstring= txtstring(1:1) // char(0)
00069             call toutstc (txtstring)
00070           else
00071             iactlen= 1
00072           end if
00073         else if (action.eq.9) then ! XACTION_ASCII
00074           if (iactlen.lt.iprntlen) then
00075             iactlen= iactlen+1
00076             txtstring(iactlen:iactlen)= char(i1)
00077           end if
00078           if (iactlen.lt.iprntlen) then
00079             iactlen= iactlen+1
00080             txtstring(iactlen:iactlen)= char(i2)
00081           end if
00082           if (iactlen.ge.iprntlen) then
00083             txtstring(iactlen+1:iactlen+1) = char(0)
00084             call toutstc (txtstring)
00085           end if
00086         else if (action.eq.10) then ! XACTION_BCKCOL
00087           call bckcol(i1)
00088         else if (action.eq.11) then ! XACTION_LINCOL
00089           call lincol (i1)
00090         else if (action.eq.12) then ! XACTION_TXTCOL
00091           call txtcol (i1)
00092         else if (action.eq.13) then ! XACTION_FONTATTR
00093           if (i1.eq.0) call italir()
00094           if (i1.eq.1) call italic()
00095           if (i2.eq.0) call nrmsiz()
00096           if (i2.eq.1) call dblsiz()
00097         else if (action.eq.14) then ! XACTION_NOOP
00098           continue
00099         else if (action.eq.15) then ! XACTION_CLIP
00100           if (i1.eq.0) then ! clipping not active
00101             kminsx= 0
00102             kminsy= 0
00103             kmaxsx= 1023 ! TEK_XMAX
00104             kmaxsy= 780 ! TEK_YMAX
00105             call swind1(kminsx,kminsy,kmaxsx,kmaxsy) ! Set bool ClippingNotActive
00106           end if
00107         else if (action.eq.16) then ! XACTION_CLIP1
00108           kminsx= i1
00109           kminsy= i2
00110           call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
```

```
00111         else if (action.eq.17) then ! XACTION_CLIP2
00112           kmaxsx= i1
00113           kmaxsy= i2
00114           call swind1(kminsx,kminsy,kmaxsx,kmaxsy)
00115         else ! unknown
00116           continue
00117         end if
00118       if (ios.eq.0) goto 10 ! until EOF
00119
00120       close (iunit)
00121       gethdc= .false.
00122       return
00123       end
```

9.23 Mainpage.dox File Reference

9.24 PlotHDC.f03 File Reference

Utility: Plot Journalfiles.

Functions/Subroutines

- program plothdc

9.24.1 Detailed Description

Utility: Plot Journalfiles.

Version

1.0-GCC

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Utility to draw journal-hardcopies from SDL2 and wX programs. With cut/paste they could be used by other MS-win programs. Program parameters are optained by calling ISO Fortran 2003 intrinsic procedures.

Note

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

Definition in file PlotHDC.f03.

9.24.2 Function/Subroutine Documentation

9.24.2.1 plothdc()

```
program plothdc
```

Definition at line 26 of file PlotHDC.f03.

9.25 PlotHDC.f03

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

9.26 Strings.for File Reference

TCS: String functions.

Functions/Subroutines

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

9.26.1 Detailed Description

TCS: String functions.

Version

1.26

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

Fortran utility functions for string processing

Definition in file Strings.for.

9.26.2 Function/Subroutine Documentation

9.26.2.1 istringlen()

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

Definition at line 94 of file Strings.for.

9.26.2.2 itrimlen()

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

Definition at line 133 of file Strings.for.

9.26.2.3 printstring()

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

Definition at line 114 of file Strings.for.

9.26.2.4 substitute()

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

Definition at line 30 of file Strings.for.

9.27 Strings.for

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

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

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

9.28 TCS.for File Reference

TCS: Tektronix Plot 10 Emulation.

Functions/Subroutines

- subroutine vcursr (IC, X, Y)
- subroutine drawr (X, Y)
- subroutine mover (X, Y)
- subroutine pointr (X, Y)
- subroutine dashr (X, Y, iL)
- subroutine rel2ab (Xrel, Yrel, Xabs, Yabs)
- subroutine drawa (X, Y)
- subroutine movea (X, Y)
- subroutine pointa (X, Y)
- subroutine dasha (X, Y, iL)
- subroutine wincot (X, Y, IX, IY)
- subroutine revcot (IX, IY, X, Y)
- subroutine anstr (NChar, IStrin)
- subroutine ancho (ichar)
- subroutine newlin
- subroutine cartn
- subroutine linef
- subroutine baksp
- subroutine newpag
- function linhgt (Numlin)
- function linwdt (NumChr)
- subroutine lintrn
- subroutine logtrn (IMODE)
- subroutine twindo (IX1, IX2, IY1, IY2)
- subroutine swindo (IX, LX, IY, LY)
- subroutine dwindo (X1, X2, Y1, Y2)
- subroutine vwindo (X, XL, Y, YL)
- subroutine rescal
- subroutine rrotat (Grad)
- subroutine rscale (Faktor)
- subroutine home
- subroutine setmrg (Mlinks, Mrecht)
- subroutine seetrm (IBaud, Iterm, ICSize, MaxScr)
- subroutine seetrn (xf, yf, key)
- logical function genflg (ITEM)

9.28.1 Detailed Description

TCS: Tektronix Plot 10 Emulation.

Version

4.1

Author

(C) 2022 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

System independent subroutines

Definition in file TCS.for.

9.28.2 Function/Subroutine Documentation

9.28.2.1 ancho()

```
subroutine ancho (
              ichar )
```

Definition at line 339 of file TCS.for.

9.28.2.2 anstr()

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

Definition at line 329 of file TCS.for.

9.28.2.3 baksp()

```
subroutine baksp
```

Definition at line 384 of file TCS.for.

9.28.2.4 cartn()

```
subroutine cartn
```

Definition at line 365 of file TCS.for.

9.28.2.5 dasha()

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

Definition at line 290 of file TCS.for.

9.28.2.6 dashr()

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

Definition at line 236 of file TCS.for.

9.28.2.7 drawa()

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

Definition at line 257 of file TCS.for.

9.28.2.8 drawr()

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

Definition at line 212 of file TCS.for.

9.28.2.9 dwindo()

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

Definition at line 462 of file TCS.for.

9.28.2.10 genflg()

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

Definition at line 558 of file TCS.for.

9.28.2.11 home()

```
subroutine home
```

Definition at line 518 of file TCS.for.

9.28.2.12 linef()

```
subroutine linef
```

Definition at line 374 of file TCS.for.

9.28.2.13 linhgt()

```
function linhgt (
              Numlin )
```

Definition at line 400 of file TCS.for.

9.28.2.14 lintrn()

```
subroutine lintrn
```

Definition at line 418 of file TCS.for.

9.28.2.15 linwdt()

```
function linwdt (
              NumChr )
```

Definition at line 408 of file TCS.for.

9.28.2.16 logtrn()

```
subroutine logtrn (
              IMODE )
```

Definition at line 428 of file TCS.for.

9.28.2.17 movea()

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

Definition at line 268 of file TCS.for.

9.28.2.18 mover()

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

Definition at line 220 of file TCS.for.

9.28.2.19 newlin()

```
subroutine newlin
```

Definition at line 357 of file TCS.for.

9.28.2.20 newpag()

```
subroutine newpag
```

Definition at line 392 of file TCS.for.

9.28.2.21 pointa()

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

Definition at line 279 of file TCS.for.

9.28.2.22 pointr()

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

Definition at line 228 of file TCS.for.

9.28.2.23 rel2ab()

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

Definition at line 244 of file TCS.for.

9.28.2.24 rescal()

```
subroutine rescal
```

Definition at line 481 of file TCS.for.

9.28.2.25 revcot()

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

Definition at line 314 of file TCS.for.

9.28.2.26 rrotat()

```
subroutine rrotat (
              Grad )
```

Definition at line 501 of file TCS.for.

9.28.2.27 rscale()

```
subroutine rscale (
              Faktor )
```

Definition at line 510 of file TCS.for.

9.28.2.28 seetrm()

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

Definition at line 536 of file TCS.for.

9.28.2.29 seetrn()

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

Definition at line 547 of file TCS.for.

9.28.2.30 setmrg()

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

Definition at line 527 of file TCS.for.

9.28.2.31 swindo()

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

Definition at line 450 of file TCS.for.

9.28.2.32 twindo()

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

Definition at line 443 of file TCS.for.

9.28.2.33 vcursr()

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

Definition at line 202 of file TCS.for.

9.28.2.34 vwindo()

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

Definition at line 469 of file TCS.for.

9.28.2.35 wincot()

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

Definition at line 301 of file TCS.for.

9.29 TCS.for

```
00001 C> \file       TCS.for
00002 C> \brief      TCS: Tektronix Plot 10 Emulation
00003 C> \version    4.1
00004 C> \author     (C) 2022 Dr.-Ing. Klaus Friedewald
00005 C> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 C> \~german
00007 C> Systemübergreifende TCS-Routinen
00008 C> \~english
00009 C> System independent subroutines
00010 C> \~
00011 C
00012 C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC   Changelog   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00013 C
00014 C      26.07.23 Version 5.0:
00015 C               Einheitliche Version CPM/DOS/Windows/SDL2/wX
00016 C
00017 C      27.11.20 Version 4.0:
00018 C               Einheitliche Version CPM/DOS/Windows/SDL2
00019 C
00020 C      17.08.20 Version 3.2
00021 C               Harmonisierung der Verwendung des Commonblocks TKTRNX
00022 C               Variable KHOMEY wird jetzt (analog alter DOS-Version) verwendet.
00023 C               Da KHOMEY nicht in der CP/M Version vorhanden ist, muss ab dieser
00024 C               Version fuer eine Complilation unter CP/M die entsprechende Zeile
00025 C               in der SUBROUTINE HOME geändert werden.
00026 C
00027 C      13.11.17 Version 3.1
00028 C               Anpassung an OpenWatcom 2.0
00029 C               Bugfix: Unterscheidung Aufrufe ueber windowsx.h (win16) und GDI (win32)
00030 C                - SelectPen -> SelectObject
00031 C                - DeletePen -> DeleteObject
00032 C                - DeleteBrush -> DeleteObject
00033 C                - GetStockBrush -> GetStockObject
00034 C                - DeleteRgn -> DeleteObject
00035 C                - SelectFont -> SelectObject
00036 C                - DeleteFont -> DeleteObject
00037 C
00038 C      27.03.13 Version 3.0
00039 C               Anpassung an Windows 7 und OpenWatcom 1.9
00040 C               Anpassung an gfortran anstelle von g77 der GCC
00041 C
00042 C      22.12.05 Version 2.19
00043 C               Elimination berechnetes GOTO in LOGTRN
00044 C
00045 C      18.10.05 Version 2.18
00046 C               Anpassung der Windowsversionen zur gemeinsamen Verwendung SDL2:
```

```
C                 TCSdrWIN.for
C                 TCSdWINc.h
C                 - Überfuehrung der Deklaration aus TCSdWIN.c nach *.h:
C                   GraphicError und CreateMainWindow_IfNecessary
C                 - Definition der Fehlernummern als Konstante statt enum
C               Abhaengigkeit Watcom-Defaultwindowsystem eliminiert
C               - TCSdWINc.c: Kein Abbruch bei OpenWatcom > 1.3 und
C                 definiertem Symbol trace_calls
C
C      26.10.04 Version 2.17
C               Bugfix Windows-System: Größe und Defaultposition des Status-
C                fensters wird bei der Erzeugung berechnet -> 1. RESTORE nach
C                Verkleinern des Graphikfensters entspricht dem vorherigen
C                Bild. 2. Angleichung des Verhaltens von 16- und 32bit Windows
C               Bei Definition des Symbols STAT_WINDOW_PRIVATE erhält das
C                Statusfenster einen privaten Devicekontext.
C               Zusammenfuehrung Initialisierung der Windows-Library und
C                Windows-DLL -> zusaetzliche Sourcefiles
C                TCSinitt.for, CreateMainWindow.c, GetMainInstance.c
C
C      23.06.04 Version 2.16:
C               Anpassungen an GNU-Compiler fuer Win32. Zusätzliches Sourcefile
C                fuer die GNU-Version: WinMain.c
C               CSIZE in Windows-Version: Korrektur Rundungsfehler
C
C      08.06.04 Version 2.15:
C               Umbenennung lib$movc3 in lib_movc3 (entsprechend ANSI-Fortran)
C               Modul STRINGS.FOR: Version 1.24
C
C      27.06.03 Version 2.14:
C               Verarbeitung Steuerzeichen in ANCHO
C
C      21.10.02 Version 2.13:
C               Einheitliche Version CPM/DOS/Windows
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Grundversion fuer C128 / Version 1.0:
C
C      Zugehoerige Module:
C              TKTRNX.FOR    Common-Block TKTRNX
C              TCSBASIC.ASM  Low-Level Routinen in Bank 0, C128 spezifisch
C              TCSDRIVR.ASM  Treiber fuer TCSBASIC
C              TCSGIN.ASM    Treiber des Gin-Cursors
C
C      20.4.88         Dr.-Ing. K. Friedewald
C                      4000 Duesseldorf 1
C                      Gerresheimerstr. 84
C
C      21.10.02 Version 2.13:
C               Vereinheitlichung CPM/DOS/Windowsversion
C               Zusätzliches Modul: TCSdrCPM.FOR: früher Teil von TCS.FOR
C               Ausschließliche Verwendung von durch grosses "C" eingeleiteten
C                Kommentaren zur Kompatibilität mit FORTRAN 4
C               Umbenennung des Includefiles in Tktrnx.fd. So kann unter CP/M
C                das als Teil des Filenamens interpretierte "/" der INCLUDE-
C                Anweisung entsprechend der 8.3 Filenamen umgesetzt werden.
C               Implementierung Unterprogramm TCSLEV
C               Bugfix: Kommentar in Tktrnx.fd wurde falsch gekennzeichnet
C                       (c statt C) -> SVSTAT und RESTAT fehlerhaft, da nicht
C                       erkannte Kommentare zusaetzliche Variablen erzeugten.
C
C      TBD: Implementierung vertikale Auflösung von 400 Pixeln
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Anpassung an DOS:
C
C      Aenderungen gegenueber CP/M-Version:
C               SEELOC, DCURSR, SVSTAT, RESTAT, CSIZE in TCSdrDOS.FOR
C      Bugfix:  DASHA, DASHR - Korrektur Parameterliste
C               SEETRM - ibaud statt ibaudr
C
C      Zugehoerige Module:
C               TKTRNX.FOR    Common-Block TKTRNX
C               TCSdrDOS.FOR  Bildschirmtreiber
C               TCSdDOSa.ASM  Betriebssystemspezifische Low-Level Routinen
C               HDCOPY.FOR    Hardcopyroutine
C               STRINGS.FOR   Hilfsroutinen zur Stringverarbeitung
C               OUTTEXT.FOR   nur für WATCOM-Compiler
C
C      25.10.01 Version 2.00:  Dr.-Ing. K. Friedewald
C
C      07.02.02 Version 2.10:
C               Implementierung multilinguale Fehlermeldungen
C
C      11.10.02 Version 2.12:
```

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



C
C Graphic Input
C

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

C
C  Virtuelle Graphik, relativ
C

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



      subroutine mover (X,Y)
```

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

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

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

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

9.30 TCSdrWXcpp.cpp File Reference

wX Port: Low-Level Driver

```
#include <wx/string.h>
#include <wx/frame.h>
#include <wx/panel.h>
#include <wx/sizer.h>
#include <wx/dc.h>
#include <wx/dcclient.h>
#include <wx/dcsvg.h>
#include <wx/image.h>
#include <wx/dcmemory.h>
#include <wx/log.h>
#include <wx/msgdlg.h>
#include <wx/stdpaths.h>
#include <wx/filename.h>
#include <wx/xml/xml.h>
#include <wx/file.h>
#include "sglib.h"
#include "TCSdrWXcpp.hpp"
#include "TKTRNX.hpp"
#include "G2dAG2.hpp"
#include "graph2d.h"
```

Classes

- struct xJournalEntry_typ
- class cTCScanvas

Macros

- #define wxDEBUG_LEVEL 2
- #define MAX_COLOR_INDEX 15
- #define TMPSTRLEN TCS_FILE_NAMELEN
- #define TMPSTRLEN TCS_FILE_NAMELEN

Typedefs

- typedef struct xJournalEntry_typ xJournalEntry_typ
- typedef char ErrMsg[TCS_MESSAGELEN]

Functions

- void initt0 ()
- wxWindowID getCanvasID (wxWindowID win2search)
- void RepaintBuffer (wxDC &dc)
- void PresetProgPar ()
- void CustomizeProgPar ()
- void XMLreadProgPar (const char *filname)
- void winlbl0 (const char PloWinNam[ ], const char StatWinNam[ ], const char IniFilNam[ ])
- bool WINSELECT (wxWindowID *iD)
- void initt1 (int iMode, wxFrame *parent, wxFrame *FrameToUse, wxStatusBar *StatusBarToUse)
- void FINITT (int *ix, int *iy)
- void IOWAIT (int *iWait)
- void swind1_ (int *ix1, int *iy1, int *ix2, int *iy2)
- void ERASE (void)
- void MOVABS (int *ix, int *iy)
- void DRWABS (int *ix, int *iy)
- void DSHABS (int *ix, int *iy, int *iMask)
- void PNTABS (int *ix, int *iy)
- void BCKCOL (int *iCol)
- void LINCOL (int *iCol)
- void TXTCOL (int *iCol)
- void DEFAULTCOLOUR (void)
- void outgtext_ (char strng[ ])
- void ITALIC (void)
- void ITALIR (void)
- void DBLSIZ (void)
- void NRMSIZ (void)
- void BELL (void)
- void outtext_ (char strng[ ])
- void TCSGraphicError (int iErr, const char *msg)
- void DCURSR (int *ic, int *ix, int *iy)
- void TINPUT (int *ic)
- void HDCOPY (void)
- void SVSTAT (char dst[ ])
- void RESTAT (char src[ ])
- void lib_movc3_ (int *len, char sou[ ], char dst[ ])

Variables

- static char szTCSWindowName [TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME
- static char szTCSstatWindowName [TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME
- static char szTCSIniFile [TCS_FILE_NAMELEN] = TCS_INIFILE_NAME
- static char szTCSHardcopyFile [TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME
- 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 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 wxColour TCSColorTable [MAX_COLOR_INDEX+1]
- static cTCScanvas * ActiveCanvas = NULL
- static wxWindowID ActiveCanvasID = 0
- static cTCScanvas * OpenCanvases [MAX_OPEN_CANVAS] = {}

9.30.1 Detailed Description

wX Port: Low-Level Driver

Version

1.1

Author

(C) 2024 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

system-specific subroutines of the Tektronix emulation

Note

```
Under wX several drawing windows can be used at the same time,
see the example wxDemo.
```

Definition in file TCSdrWXcpp.cpp.

9.30.2 Macro Definition Documentation

9.30.2.1 MAX_COLOR_INDEX

```
#define MAX_COLOR_INDEX 15
```

Definition at line 225 of file TCSdrWXcpp.cpp.

9.30.2.2 TMPSTRLEN [1/2]

```
#define TMPSTRLEN TCS_FILE_NAMELEN
```

9.30.2.3 TMPSTRLEN [2/2]

```
#define TMPSTRLEN TCS_FILE_NAMELEN
```

9.30.2.4 wxDEBUG_LEVEL

```
#define wxDEBUG_LEVEL 2
```

Definition at line 28 of file TCSdrWXcpp.cpp.

9.30.3 Typedef Documentation

9.30.3.1 ErrMsg

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

Definition at line 164 of file TCSdrWXcpp.cpp.

9.30.3.2 xJournalEntry_typ

```
typedef struct xJournalEntry_typ xJournalEntry_typ
```

9.30.4 Function Documentation

9.30.4.1 BCKCOL()

```
void BCKCOL (
            int * iCol )
```

Definition at line 1442 of file TCSdrWXcpp.cpp.

9.30.4.2 BELL()

```
void BELL (
            void  )
```

Definition at line 1645 of file TCSdrWXcpp.cpp.

9.30.4.3 CustomizeProgPar()

```
void CustomizeProgPar ( )
```

Definition at line 546 of file TCSdrWXcpp.cpp.

9.30.4.4 DBLSIZ()

```
void DBLSIZ (
            void  )
```

Definition at line 1592 of file TCSdrWXcpp.cpp.

9.30.4.5 DCURSR()

```
void DCURSR (
            int * ic,
            int * ix,
            int * iy )
```

Definition at line 1709 of file TCSdrWXcpp.cpp.

9.30.4.6 DEFAULTCOLOUR()

```
void DEFAULTCOLOUR (
            void  )
```

Definition at line 1498 of file TCSdrWXcpp.cpp.

9.30.4.7 DRWABS()

```
void DRWABS (
            int * ix,
            int * iy )
```

Definition at line 1378 of file TCSdrWXcpp.cpp.

9.30.4.8 DSHABS()

```
void DSHABS (
            int * ix,
```

```
            int * iy,
            int * iMask )
```
Definition at line 1397 of file TCSdrWXcpp.cpp.

9.30.4.9 ERASE()

```
void ERASE (
            void  )
```
Definition at line 1311 of file TCSdrWXcpp.cpp.

9.30.4.10 FINITT()

```
void FINITT (
            int * ix,
            int * iy )
```
Definition at line 1225 of file TCSdrWXcpp.cpp.

9.30.4.11 getCanvasID()

```
wxWindowID getCanvasID (
            wxWindowID win2search )
```
Definition at line 292 of file TCSdrWXcpp.cpp.

9.30.4.12 HDCOPY()

```
void HDCOPY (
            void  )
```
Definition at line 1753 of file TCSdrWXcpp.cpp.

9.30.4.13 initt0()

```
void initt0 ( )
```
Definition at line 262 of file TCSdrWXcpp.cpp.

9.30.4.14 initt1()

```
void initt1 (
            int iMode,
            wxFrame * parent,
            wxFrame * FrameToUse,
            wxStatusBar * StatusBarToUse )
```
Definition at line 1130 of file TCSdrWXcpp.cpp.

9.30.4.15 IOWAIT()

```
void IOWAIT (
            int * iWait )
```
Definition at line 1255 of file TCSdrWXcpp.cpp.

9.30.4.16 ITALIC()

```
void ITALIC (
            void  )
```

Definition at line 1556 of file TCSdrWXcpp.cpp.

9.30.4.17 ITALIR()

```
void ITALIR (
            void  )
```

Definition at line 1574 of file TCSdrWXcpp.cpp.

9.30.4.18 lib_movc3_()

```
void lib_movc3_ (
            int * len,
            char sou[],
            char dst[] )
```

Definition at line 1856 of file TCSdrWXcpp.cpp.

9.30.4.19 LINCOL()

```
void LINCOL (
            int * iCol )
```

Definition at line 1461 of file TCSdrWXcpp.cpp.

9.30.4.20 MOVABS()

```
void MOVABS (
            int * ix,
            int * iy )
```

Definition at line 1359 of file TCSdrWXcpp.cpp.

9.30.4.21 NRMSIZ()

```
void NRMSIZ (
            void  )
```

Definition at line 1615 of file TCSdrWXcpp.cpp.

9.30.4.22 outgtext_()

```
void outgtext_ (
            char strng[] )
```

Definition at line 1515 of file TCSdrWXcpp.cpp.

9.30.4.23 outtext_()

```
void outtext_ (
            char strng[] )
```

Definition at line 1654 of file TCSdrWXcpp.cpp.

9.30.4.24 PNTABS()

```
void PNTABS (
            int * ix,
            int * iy )
```

Definition at line 1423 of file TCSdrWXcpp.cpp.

9.30.4.25 PresetProgPar()

```
void PresetProgPar ( )
```

Definition at line 525 of file TCSdrWXcpp.cpp.

9.30.4.26 RepaintBuffer()

```
void RepaintBuffer (
            wxDC & dc )
```

Definition at line 309 of file TCSdrWXcpp.cpp.

9.30.4.27 RESTAT()

```
void RESTAT (
            char src[] )
```

Definition at line 1839 of file TCSdrWXcpp.cpp.

9.30.4.28 SVSTAT()

```
void SVSTAT (
            char dst[] )
```

Definition at line 1828 of file TCSdrWXcpp.cpp.

9.30.4.29 swind1_()

```
void swind1_ (
            int * ix1,
            int * iy1,
            int * ix2,
            int * iy2 )
```

Definition at line 1271 of file TCSdrWXcpp.cpp.

9.30.4.30 TCSGraphicError()

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

Definition at line 1667 of file TCSdrWXcpp.cpp.

9.30.4.31 TINPUT()

```
void TINPUT (
            int * ic )
```

Definition at line 1731 of file TCSdrWXcpp.cpp.

9.30.4.32 TXTCOL()

```
void TXTCOL (
            int * iCol )
```

Definition at line 1480 of file TCSdrWXcpp.cpp.

9.30.4.33 winlbl0()

```
void winlbl0 (
            const char PloWinNam[],
            const char StatWinNam[],
            const char IniFilNam[] )
```

Definition at line 1022 of file TCSdrWXcpp.cpp.

9.30.4.34 WINSELECT()

```
bool WINSELECT (
            wxWindowID * iD )
```

Definition at line 1089 of file TCSdrWXcpp.cpp.

9.30.4.35 XMLreadProgPar()

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

Definition at line 579 of file TCSdrWXcpp.cpp.

9.30.5 Variable Documentation

9.30.5.1 ActiveCanvas

```
cTCScanvas* ActiveCanvas = NULL  [static]
```

Definition at line 249 of file TCSdrWXcpp.cpp.

9.30.5.2 ActiveCanvasID

```
wxWindowID ActiveCanvasID = 0  [static]
```

Definition at line 250 of file TCSdrWXcpp.cpp.

9.30.5.3 iHardcopyCount

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

Definition at line 156 of file TCSdrWXcpp.cpp.

9.30.5.4 OpenCanvases

```
cTCScanvas* OpenCanvases[MAX_OPEN_CANVAS] = {}  [static]
```

Definition at line 251 of file TCSdrWXcpp.cpp.

9.30.5.5 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,
                "DOS",
                TCS_INIDEF_USR,
                TCS_INIDEF_HDCACT,
                TCS_INIDEF_USRWRN,
                TCS_INIDEF_EXIT,
                "Windows",
                "Windows",
                TCS_INIDEF_JOUCREATE,
                TCS_INIDEF_JOUENTRY,
                TCS_INIDEF_JOUADD,
                "JOUCLR unused",
                "JOUUNKWN unused",
                TCS_INIDEF_XMLPARSER,
                TCS_INIDEF_XMLOPEN,
                TCS_INIDEF_UNKNAUDIO,
                TCS_INIDEF_USR2,
                TCS_INIDEF_INI2,
                "Maxerr only for internal Use" }
```

Definition at line 165 of file TCSdrWXcpp.cpp.

9.30.5.6 szTCSHardcopyFile

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

Definition at line 139 of file TCSdrWXcpp.cpp.

9.30.5.7 szTCSIniFile

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

Definition at line 138 of file TCSdrWXcpp.cpp.

9.30.5.8 szTCSsect0

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

Definition at line 142 of file TCSdrWXcpp.cpp.

9.30.5.9 szTCSstatWindowName

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

Definition at line 137 of file TCSdrWXcpp.cpp.

9.30.5.10 szTCSWindowName

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

Definition at line 136 of file TCSdrWXcpp.cpp.

9.30.5.11 TCSColorTable

```
wxColour TCSColorTable[MAX_COLOR_INDEX+1]  [static]
```

Initial value:

```
= {
                    {240,240,240,wxALPHA_OPAQUE },
```

```
                  {  0,  0,  0,wxALPHA_OPAQUE },
                  {240, 80, 80,wxALPHA_OPAQUE },
                  { 80,240, 80,wxALPHA_OPAQUE },
                  { 80,240,240,wxALPHA_OPAQUE },
                  { 80, 80,240,wxALPHA_OPAQUE },
                  {240,240, 80,wxALPHA_OPAQUE },
                  {160,160,160,wxALPHA_OPAQUE },
                  {240, 80,240,wxALPHA_OPAQUE },
                  {160,  0,  0,wxALPHA_OPAQUE },
                  {  0,160,  0,wxALPHA_OPAQUE },
                  {  0,  0,160,wxALPHA_OPAQUE },
                  {  0,160,160,wxALPHA_OPAQUE },
                  {160, 80,  0,wxALPHA_OPAQUE },
                  { 80, 80, 80,wxALPHA_OPAQUE },
                  {160,  0,160,wxALPHA_OPAQUE }
                }
```

Definition at line 227 of file TCSdrWXcpp.cpp.

9.30.5.12 TCSDefaultBckCol

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

Definition at line 155 of file TCSdrWXcpp.cpp.

9.30.5.13 TCSDefaultLinCol

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

Definition at line 153 of file TCSdrWXcpp.cpp.

9.30.5.14 TCSDefaultTxtCol

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

Definition at line 154 of file TCSdrWXcpp.cpp.

9.30.5.15 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,
                10,
                TCS_INIDEF_USRL,
                TCS_INIDEF_HDCACTL,
                TCS_INIDEF_USRWRNL,
                TCS_INIDEF_EXITL,
                10,
                10,
                TCS_INIDEF_JOUCREATEL,
                TCS_INIDEF_JOUENTRYL,
                TCS_INIDEF_JOUADDL,
                10,
                10,
                TCS_INIDEF_XMLPARSERL,
                TCS_INIDEF_XMLOPENL,
                TCS_INIDEF_UNKNAUDIOL,
                TCS_INIDEF_USR2L,
                TCS_INIDEF_INI2L,
                10}
```

Definition at line 192 of file TCSdrWXcpp.cpp.

9.30.5.16 TCSwindowIniXrelpos

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

Definition at line 145 of file TCSdrWXcpp.cpp.

9.30.5.17 TCSwindowIniXrelsiz

```
int TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX  [static]
```
Definition at line 147 of file TCSdrWXcpp.cpp.

9.30.5.18 TCSwindowIniYrelpos

```
int TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY  [static]
```
Definition at line 146 of file TCSdrWXcpp.cpp.

9.30.5.19 TCSwindowIniYrelsiz

```
int TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY  [static]
```
Definition at line 148 of file TCSdrWXcpp.cpp.

9.31 TCSdrWXcpp.cpp

```
00001 /** ****************************************************************************
00002 \file       TCSdrWXcpp.cpp
00003 \brief      wX Port: Low-Level Driver
00004 \version    1.1
00005 \author     (C) 2024 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           Unter wX können mehrere Zeichenfenster gleichzeitig verwendet werden,
00011           siehe das Beispiel wxDemo.
00012 \endverbatim
00013 \~english
00014          system-specific subroutines of the Tektronix emulation
00015 \note \verbatim
00016          Under wX several drawing windows can be used at the same time,
00017          see the example wxDemo.
00018 \endverbatim
00019 \~
00020 ***************************************************************************** */
00021
00022
00023 /*
00024 ------------------- Debug Switches ------------------------------------------
00025 */
00026
00027 // #define wxDEBUG_LEVEL 0
00028 #define wxDEBUG_LEVEL 2 // Debug: Output into the status window
00029 // #define TRACE_CALLS // additional debug output: journalpointer
00030
00031 /*
00032 ------------------- Headerfiles ---------------------------------------------
00033 */
00034
00035 // #include <wx/intl.h>
00036 #include <wx/string.h>
00037
00038 #include <wx/frame.h>      // needed for: class cTSCcanvas
00039 #include <wx/panel.h>
00040 #include <wx/sizer.h>
00041 // #include <wx/display.h>
00042 // #include <wx/gdicmn.h>
00043
00044 #include <wx/dc.h>         // needed for: subroutine RepaintBuffer
00045 #include <wx/dcclient.h>
00046
00047 #include <wx/dcsvg.h>
00048
00049 #include <wx/image.h>     // needed for bitmap hardcopies (not for *.bmp)
00050 #include <wx/dcmemory.h>
00051
00052 // #include <wx/metafile.h>
00053
00054 #include <wx/log.h>        // needed for: subroutine TCSGraphicError
```

```
00055 #include <wx/msgdlg.h>
00056
00057 #include <wx/stdpaths.h>   // needed for: winlbl
00058 #include <wx/filename.h>
00059
00060 #include <wx/xml/xml.h>    // Read inifiles
00061
00062 #include <wx/file.h>
00063
00064 #include "sglib.h"         // Journal for repaint / hardcopy
00065
00066 #include "TCSdrWXcpp.hpp"  // program configuration
00067 #include "TKTRNX.hpp"      // common block TCS
00068 #include "G2dAG2.hpp"      // common block AG2
00069 #include "graph2d.h"       // contains forward declarations
00070
00071
00072
00073 /*
00074 ------------------- Declarations ---------------------------------------------
00075 */
00076
00077 typedef struct xJournalEntry_typ {struct xJournalEntry_typ * previous;
00078                                   struct xJournalEntry_typ * next;
00079                                   int action; int i1; int i2;}
00080               xJournalEntry_typ;
00081
00082
00083 class cTCScanvas
00084 {
00085     public:
00086
00087         wxFrame* TCSframe; // windows
00088         wxPanel* TCSpanel;
00089         wxLogWindow* logWindow;
00090         wxStatusBar* TCSstatusBar;
00091
00092         wxWindowID ID_TCSframe;
00093         wxWindowID ID_TCSpanel;
00094         wxWindowID ID_TCSstatus;
00095
00096         wxPen      TCSpen; //resources
00097         wxBrush    TCSbrush;
00098         wxFont     TCSfont;
00099
00100         bool ClippingNotActive = true; // drawing status
00101         int TCSpanelKeyPressed;
00102         int TCSmouseButtonDown, TCSmouseX, TCSmouseY;
00103
00104         xJournalEntry_typ* xTCSJournal = NULL; // journal used as a drawing metafile
00105
00106         struct TKTRNX TekSav; // notepad for changing instances
00107         struct G2dAG2 AG2Sav;
00108         int    DefaultLinColSav, DefaultTxtColSav, DefaultBckColSav;
00109         char   HardcopyFileSav[TCS_FILE_NAMELEN], sect0Sav[TCS_FILE_NAMELEN];
00110
00111         cTCScanvas(int iMode, wxFrame* parent, wxFrame* FrameToUse, wxStatusBar* StatusBarToUse);
00112         virtual ~cTCScanvas();
00113
00114     protected:
00115
00116     private:
00117
00118         void CompleteCanvas (wxSize UseScreen, wxPoint PosScreen, wxSize MinScreen); // Add sizers,
      menues etc.
00119
00120         void OnTCSClose(wxCloseEvent& event); // event handlers
00121         void OnTCSpanelPaint(wxPaintEvent& event);
00122         void OnTCSpanelResize(wxSizeEvent& event);
00123         void OnTCSpanelKey(wxKeyEvent& event);
00124         void OnTCSmouseLeft(wxMouseEvent& event);
00125         void OnTCSmouseMiddle(wxMouseEvent& event);
00126         void OnTCSmouseRight(wxMouseEvent& event);
00127
00128 };
00129
00130
00131
00132 /*
00133 -------------------- Global Variables -------------------------------------------
00134 */
00135
00136 static char     szTCSWindowName[TCS_WINDOW_NAMELEN] = TCS_WINDOW_NAME,
00137                 szTCSstatWindowName[TCS_WINDOW_NAMELEN] = TCS_STATWINDOW_NAME,
00138                 szTCSIniFile[TCS_FILE_NAMELEN] = TCS_INIFILE_NAME,
00139                 szTCSHardcopyFile[TCS_FILE_NAMELEN] = TCS_HDCFILE_NAME,
00140 //                szTCSGraphicFont[TCS_FILE_NAMELEN] = TCS_INIDEF_FONT,
```

```
00141 //                szTCSSysFont[TCS_FILE_NAMELEN] = TCS_INIDEF_SYSFONT,
00142                 szTCSsect0[TCS_FILE_NAMELEN] = TCS_INISECT0;
00143
00144
00145 static  int     TCSwindowIniXrelpos = TCS_INIDEF_WINPOSX, // window size/position
00146                 TCSwindowIniYrelpos = TCS_INIDEF_WINPOSY, // at initt in % of Screen
00147                 TCSwindowIniXrelsiz = TCS_INIDEF_WINSIZX,
00148                 TCSwindowIniYrelsiz = TCS_INIDEF_WINSIZY,
00149 //                TCSstatWindowIniXrelpos = TCS_INIDEF_STATPOSX, // dito
00150 //                TCSstatWindowIniYrelpos = TCS_INIDEF_STATPOSY, // Statusfenster
00151 //                TCSstatWindowIniXrelsiz = TCS_INIDEF_STATSIZX,
00152 //                TCSstatWindowIniYrelsiz = TCS_INIDEF_STATSIZY,
00153                 TCSDefaultLinCol = TCS_INIDEF_LINCOL,
00154                 TCSDefaultTxtCol = TCS_INIDEF_TXTCOL,
00155                 TCSDefaultBckCol = TCS_INIDEF_BCKCOL,
00156                 iHardcopyCount = 1;  // Zähler zur Erzeugung Filenamen
00157
00158
00159
00160 /*
00161    Assign error numbers to error messages
00162 */
00163
00164 typedef char   ErrMsg[TCS_MESSAGELEN];
00165 static  ErrMsg  szTCSErrorMsg[(int) MSG_MAXERRNO+1] =
00166                  {"Element 0 unused","DOS",
00167                  TCS_INIDEF_UNKNGRAPHCARD, // Errno 2
00168                  TCS_INIDEF_NOFNTFIL,      // Errno 3
00169                  TCS_INIDEF_NOFNT,         // Errno 4
00170                  "DOS",
00171                  TCS_INIDEF_HDCOPN,        // Errno 6
00172                  TCS_INIDEF_HDCWRT,        // Errno 7
00173                  "DOS",
00174                  TCS_INIDEF_USR,           // Errno 9
00175                  TCS_INIDEF_HDCACT,        // Errno 10
00176                  TCS_INIDEF_USRWRN,        // Errno 11
00177                  TCS_INIDEF_EXIT,          // Errno 12
00178                  "Windows",
00179                  "Windows",
00180                  TCS_INIDEF_JOUCREATE,     // Errno 15
00181                  TCS_INIDEF_JOUENTRY,      // Errno 16
00182                  TCS_INIDEF_JOUADD,        // Errno 17
00183                  "JOUCLR unused",          // Errno 18
00184                  "JOUUNKWN unused",        // Errno 19
00185                  TCS_INIDEF_XMLPARSER,     // Errno 20
00186                  TCS_INIDEF_XMLOPEN,       // Errno 21
00187                  TCS_INIDEF_UNKNAUDIO,     // Errno 22
00188                  TCS_INIDEF_USR2,          // Errno 23
00189                  TCS_INIDEF_INI2,          // Errno 24
00190                  "Maxerr only for internal Use" };
00191
00192  static  int     TCSErrorLev[(int) MSG_MAXERRNO+1] =
00193                  {10,10,
00194                  TCS_INIDEF_UNKNGRAPHCARDL,// Errno 2
00195                  TCS_INIDEF_NOFNTFILL,     // Errno 3
00196                  TCS_INIDEF_NOFNTL,        // Errno 4
00197                  10,
00198                  TCS_INIDEF_HDCOPNL,       // Errno 6
00199                  TCS_INIDEF_HDCWRTL,       // Errno 7
00200                  10,
00201                  TCS_INIDEF_USRL,          // Errno 9
00202                  TCS_INIDEF_HDCACTL,       // Errno 10
00203                  TCS_INIDEF_USRWRNL,       // Errno 11
00204                  TCS_INIDEF_EXITL,         // Errno 12
00205                  10,
00206                  10,
00207                  TCS_INIDEF_JOUCREATEL,    // Errno 15
00208                  TCS_INIDEF_JOUENTRYL,     // Errno 16
00209                  TCS_INIDEF_JOUADDL,       // Errno 17
00210                  10,                       // Errno 18
00211                  10,                       // Errno 19
00212                  TCS_INIDEF_XMLPARSERL,    // Errno 20
00213                  TCS_INIDEF_XMLOPENL,      // Errno 21
00214                  TCS_INIDEF_UNKNAUDIOL,    // Errno 22
00215                  TCS_INIDEF_USR2L,         // Errno 23
00216                  TCS_INIDEF_INI2L,         // Errno 24
00217                  10};
00218
00219
00220 /*
00221    Assign colour numbers VGA palette coordinates
00222 */
00223
00224
00225 #define MAX_COLOR_INDEX 15
00226
00227  static wxColour TCSColorTable[MAX_COLOR_INDEX+1] = {
```

```
00228                   {240,240,240,wxALPHA_OPAQUE }, /* iCol= 00: weiss (DOS: 01) */
00229                   {  0,  0,  0,wxALPHA_OPAQUE }, /* iCol= 01: schwarz(DOS:00) */
00230                   {240, 80, 80,wxALPHA_OPAQUE }, /* iCol= 02: rot             */
00231                   { 80,240, 80,wxALPHA_OPAQUE }, /* iCol= 03: gruen           */
00232                   { 80,240,240,wxALPHA_OPAQUE }, /* iCol= 04: blau            */
00233                   { 80, 80,240,wxALPHA_OPAQUE }, /* iCol= 05: lila            */
00234                   {240,240, 80,wxALPHA_OPAQUE }, /* iCol= 06: gelb            */
00235                   {160,160,160,wxALPHA_OPAQUE }, /* iCol= 07: grau            */
00236                   {240, 80,240,wxALPHA_OPAQUE }, /* iCol= 08: violett         */
00237                   {160,  0,  0,wxALPHA_OPAQUE }, /* iCol= 09: mattrot         */
00238                   {  0,160,  0,wxALPHA_OPAQUE }, /* iCol= 10: mattgruen       */
00239                   {  0,  0,160,wxALPHA_OPAQUE }, /* iCol= 11: mattblau        */
00240                   {  0,160,160,wxALPHA_OPAQUE }, /* iCol= 12: mattlila        */
00241                   {160, 80,  0,wxALPHA_OPAQUE }, /* iCol= 13: orange          */
00242                   { 80, 80, 80,wxALPHA_OPAQUE }, /* iCol= 14: mattgrau        */
00243                   {160,  0,160,wxALPHA_OPAQUE }  /* iCol= 15: mattviolett     */
00244                };
00245
00246
00247 // static int      TCSEventFilterData; // Userdata, z.Zt. nicht verwendet
00248
00249 static cTCScanvas*     ActiveCanvas = NULL;
00250 static wxWindowID      ActiveCanvasID = 0;
00251 static cTCScanvas*     OpenCanvases[MAX_OPEN_CANVAS] = {};
00252
00253
00254
00255 // ---------------- Internal subroutines ------------------------------------
00256
00257
00258 /*
00259   Initialization COMMON TKTRNX before creating new object of class  cTCScanvas
00260 */
00261
00262 void initt0 ()
00263 {
00264   tktrnx_.iLinCol= TCSDefaultLinCol; // reset colours
00265   tktrnx_.iTxtCol= TCSDefaultTxtCol;
00266   tktrnx_.iBckCol= TCSDefaultBckCol;
00267
00268   tktrnx_.ksizef = 0; // Reset FONT
00269   tktrnx_.kitalc = 0;
00270
00271   tktrnx_.xlog= 255.; // call LINTRN
00272   tktrnx_.ylog= 255.;
00273   tktrnx_.kminsx= 0; // call SWINDO (0,1023,0,780)
00274   tktrnx_.kmaxsx= (int) TEK_XMAX;
00275   tktrnx_.kminsy= 0;
00276   tktrnx_.kmaxsy= (int) TEK_YMAX;
00277   tktrnx_.tminvx= 0.; // call VWINDO (0.,1023.,0.,780.)
00278   tktrnx_.tmaxvx= TEK_XMAX;
00279   tktrnx_.tminvy= 0.;
00280   tktrnx_.tmaxvy= TEK_YMAX;
00281   tktrnx_.xfac= 1.; // subroutine RESCAL, called from LINTRN...VWINDO
00282   tktrnx_.yfac= 1.;
00283   tktrnx_.trsinf= 0.; // call RROTAT (0.)
00284   tktrnx_.trcosf= 1.;
00285   tktrnx_.trscal= 1.; // call RSCALE (1.)
00286
00287   tktrnx_.klmrgn= 0; //  call SETMRG (0,1023)
00288   tktrnx_.krmrgn= (int) TEK_XMAX;
00289 }
00290
00291
00292 wxWindowID getCanvasID (wxWindowID win2search)
00293 {
00294   int i;
00295
00296     i= MAX_OPEN_CANVAS-1;
00297     while (i >= 0) {
00298       if (OpenCanvases[i] != nullptr) {
00299         if ( (OpenCanvases[i]->ID_TCSframe == win2search) ||
00300              (OpenCanvases[i]->ID_TCSpanel == win2search) ) return i;
00301       }
00302       i--;
00303     }
00304     return i; // i<0 -> window is not a member of any canvas
00305 }
00306
00307
00308
00309 void RepaintBuffer (wxDC& dc)
00310 {
00311   xJournalEntry_typ * xJournalEntry;
00312   int DashStyle;
00313   wxCoord w,h;
00314   int iStringLen, iStringActual;
```

```
00315   char szString [TCS_MESSAGELEN+1];
00316
00317     wxLogDebug ( wxT("RepaintBuffer> called"));
00318 #ifdef TRACE_CALLS
00319     wxLogDebug ( wxT("RepaintBuffer> xTCSJournal: Ptr= %p / Current Entry: Ptr= %p"),
       ActiveCanvas->xTCSJournal, xJournalEntry);
00320 #endif // TRACE_CALLS
00321
00322     SGLIB_DL_LIST_GET_LAST(xJournalEntry_typ, ActiveCanvas->xTCSJournal, previous, next,
       xJournalEntry)
00323     while (xJournalEntry != NULL) {
00324
00325 #ifdef TRACE_CALLS
00326      wxLogDebug ( wxT("RepaintBuffer> xTCSJournal: Ptr= %p"), ActiveCanvas->xTCSJournal);
00327      wxLogDebug ( wxT("RepaintBuffer> Current Entry: Ptr= %p / previous: Ptr= %p / next: Ptr= %p"),
00328                   xJournalEntry, xJournalEntry->previous, xJournalEntry->next);
00329      wxLogDebug ( wxT("RepaintBuffer> XACTION_??? = %i (i1= %i, i2= %i)"),
00330                   xJournalEntry->action, xJournalEntry->i1, xJournalEntry->i2 );
00331 #endif // TRACE_CALLS
00332
00333       switch (xJournalEntry->action) {
00334         case XACTION_INITT: {
00335           initt0 ();
00336
00337           ActiveCanvas->TCSpen.SetColour (TCSColorTable[tktrnx_.iLinCol]);
00338           ActiveCanvas->TCSpen.SetStyle (wxPENSTYLE_SOLID);
00339           dc.SetPen(ActiveCanvas->TCSpen); // Umbedingt Initialstift setzen !!!
00340
00341           tktrnx_.kbeamx = tktrnx_.klmrgn;  //  call HOME, first guess khomey in INITT1()
00342           tktrnx_.kbeamy = tktrnx_.khomey;
00343         } // continue with Erase
00344         case XACTION_ERASE: {
00345           ActiveCanvas->TCSbrush.SetColour (TCSColorTable[tktrnx_.iBckCol]);
00346           dc.SetBrush (ActiveCanvas->TCSbrush);
00347           dc.SetBackground (ActiveCanvas->TCSbrush);
00348           dc.Clear();
00349
00350           ActiveCanvas->TCSfont = wxFont(wxFONTSIZE_MEDIUM, wxFONTFAMILY_TELETYPE,
00351                                          wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL, false);
00352           ActiveCanvas->TCSfont.SetFractionalPointSize
       (TEK_YMAX*TCS_REL_CHR_HEIGHT*(1+tktrnx_.ksizef));
00353           dc.SetFont(ActiveCanvas->TCSfont);
00354           dc.SetTextForeground (TCSColorTable[tktrnx_.iTxtCol]);
00355
00356           dc.GetTextExtent ("MMMMMMMMMM", &w, &h);
00357           tktrnx_.khorsz = (int) (w*0.1+0.5);
00358           tktrnx_.kversz = h;
00359           tktrnx_.khomey= (int) TEK_YMAX - tktrnx_.kversz;
00360
00361           break; // Erase don't change the cursor position
00362         }
00363         case XACTION_MOVABS: {
00364           tktrnx_.kbeamx= xJournalEntry->i1;
00365           tktrnx_.kbeamy= xJournalEntry->i2;
00366           break;
00367         }
00368         case XACTION_DRWABS: {
00369           if (!ActiveCanvas->ClippingNotActive) {
00370             dc.SetClippingRegion(tktrnx_.kminsx, tktrnx_.kminsy,
00371                tktrnx_.kmaxsx-tktrnx_.kminsx, tktrnx_.kmaxsy-tktrnx_.kminsy);
00372           }
00373           dc.DrawLine (tktrnx_.kbeamx,tktrnx_.kbeamy ,
00374                        xJournalEntry->i1, xJournalEntry->i2);
00375           tktrnx_.kbeamx= xJournalEntry->i1;
00376           tktrnx_.kbeamy= xJournalEntry->i2;
00377           dc.DrawPoint (tktrnx_.kbeamx, tktrnx_.kbeamy); // Set last point of line
00378           if (!ActiveCanvas->ClippingNotActive) dc.DestroyClippingRegion();
00379           break;
00380         }
00381         case XACTION_DSHSTYLE: {
00382           switch (xJournalEntry->i1) {
00383             case 0:  DashStyle= wxPENSTYLE_SOLID;
00384                      break;
00385             case 1:  DashStyle= wxPENSTYLE_DOT;
00386                      break;
00387             case 2:  DashStyle= wxPENSTYLE_DOT_DASH;
00388                      break;
00389             case 3:  DashStyle= wxPENSTYLE_LONG_DASH;
00390                      break;
00391             default: DashStyle= wxPENSTYLE_SOLID;
00392           }
00393           break;
00394         }
00395         case XACTION_DSHABS: {
00396           ActiveCanvas->TCSpen.SetStyle (DashStyle);
00397           dc.SetPen(ActiveCanvas->TCSpen);
00398           if (!ActiveCanvas->ClippingNotActive) {
```

```
00399             dc.SetClippingRegion(tktrnx_.kminsx, tktrnx_.kminsy,
00400                tktrnx_.kmaxsx-tktrnx_.kminsx, tktrnx_.kmaxsy-tktrnx_.kminsy);
00401           }
00402           dc.DrawLine (tktrnx_.kbeamx,tktrnx_.kbeamy ,
00403                        xJournalEntry->i1, xJournalEntry->i2);
00404           if (!ActiveCanvas->ClippingNotActive) dc.DestroyClippingRegion();
00405           ActiveCanvas->TCSpen.SetStyle (wxPENSTYLE_SOLID);
00406           dc.SetPen(ActiveCanvas->TCSpen); // reset to SOLID
00407 
00408           tktrnx_.kbeamx= xJournalEntry->i1;
00409           tktrnx_.kbeamy= xJournalEntry->i2;
00410           break;
00411         }
00412         case XACTION_PNTABS: {
00413           tktrnx_.kbeamx= xJournalEntry->i1;
00414           tktrnx_.kbeamy= xJournalEntry->i2;
00415           if (!ActiveCanvas->ClippingNotActive) {
00416             dc.SetClippingRegion(tktrnx_.kminsx, tktrnx_.kminsy,
00417                tktrnx_.kmaxsx-tktrnx_.kminsx, tktrnx_.kmaxsy-tktrnx_.kminsy);
00418           }
00419           dc.DrawPoint (tktrnx_.kbeamx, tktrnx_.kbeamy);
00420           if (!ActiveCanvas->ClippingNotActive) dc.DestroyClippingRegion();
00421           break;
00422         }
00423         case XACTION_BCKCOL: {
00424           tktrnx_.iBckCol= xJournalEntry->i1;
00425           ActiveCanvas->TCSbrush.SetColour (TCSColorTable[tktrnx_.iBckCol]);
00426           dc.SetBrush (ActiveCanvas->TCSbrush);
00427           dc.SetBackground (ActiveCanvas->TCSbrush);
00428           break;
00429         }
00430         case XACTION_LINCOL: {
00431           tktrnx_.iLinCol= xJournalEntry->i1;
00432           ActiveCanvas->TCSpen.SetColour (TCSColorTable[tktrnx_.iLinCol]);
00433           dc.SetPen(ActiveCanvas->TCSpen);
00434           break;
00435         }
00436         case XACTION_TXTCOL: {
00437           tktrnx_.iTxtCol= xJournalEntry->i1;
00438           dc.SetTextForeground (TCSColorTable[tktrnx_.iTxtCol]);
00439           break;
00440         }
00441         case XACTION_FONTATTR: {
00442           tktrnx_.kitalc= xJournalEntry->i1;
00443           if (tktrnx_.kitalc > 0) {
00444             ActiveCanvas->TCSfont.SetStyle (wxFONTSTYLE_ITALIC);
00445           } else {
00446             ActiveCanvas->TCSfont.SetStyle (wxFONTSTYLE_NORMAL);
00447           }
00448 
00449           if (tktrnx_.ksizef != xJournalEntry->i2) {
00450             tktrnx_.ksizef= xJournalEntry->i2;
00451             if (tktrnx_.ksizef > 0) {
00452               ActiveCanvas->TCSfont.SetFractionalPointSize (2.0* TEK_YMAX*TCS_REL_CHR_HEIGHT);
00453             } else {
00454               ActiveCanvas->TCSfont.SetFractionalPointSize (TEK_YMAX *TCS_REL_CHR_HEIGHT);
00455             }
00456           }
00457           dc.SetFont(ActiveCanvas->TCSfont);
00458           dc.GetTextExtent ("MMMMMMMMM", &w, &h);
00459           tktrnx_.khorsz = (int) (w*0.1+0.5);
00460           tktrnx_.kversz = h;
00461           tktrnx_.khomey= TEK_YMAX - tktrnx_.kversz;
00462           break;
00463         }
00464         case XACTION_GTEXT: {
00465             iStringActual= 0;
00466             iStringLen= xJournalEntry->i1;
00467             if (iStringLen > TCS_MESSAGELEN) iStringLen= TCS_MESSAGELEN;
00468             if (iStringLen == 0) break;
00469             szString[iStringActual++]= xJournalEntry->i2;
00470             if (iStringLen == 1) {
00471               szString[iStringActual]= '\0';
00472               dc.GetTextExtent (szString, &w, &h);
00473               dc.DrawText (szString, tktrnx_.kbeamx, tktrnx_.kbeamy+ TCS_REL_CHR_SPACING*h); // +h:
      Plot text from UPPER left corner
00474               tktrnx_.kbeamx += w; // move cursor to End of String
00475             }
00476         break;
00477         }
00478         case XACTION_ASCII: {
00479           if (iStringActual < iStringLen) {
00480             szString[iStringActual++]= xJournalEntry->i1;
00481             if (iStringActual < iStringLen) szString[iStringActual++]= xJournalEntry->i2;
00482             if (iStringActual >= iStringLen ) {
00483               szString[iStringActual]= '\0';
00484               dc.GetTextExtent (szString, &w, &h);
```

```
00485               dc.DrawText (szString, tktrnx_.kbeamx, tktrnx_.kbeamy+ TCS_REL_CHR_SPACING*h);
00486               tktrnx_.kbeamx += w;
00487             }
00488           }
00489           break;
00490         }
00491         case XACTION_NOOP: {
00492           break;
00493         }
00494         case XACTION_CLIP: {
00495           ActiveCanvas->ClippingNotActive= (xJournalEntry->i1 == 0);
00496           break;
00497         }
00498         case XACTION_CLIP1: {
00499           tktrnx_.kminsx= xJournalEntry->i1;
00500           tktrnx_.kminsy= xJournalEntry->i2;
00501           break;
00502         }
00503         case XACTION_CLIP2: {
00504           tktrnx_.kmaxsx= xJournalEntry->i1;
00505           tktrnx_.kmaxsy= xJournalEntry->i2;
00506           break;
00507         }
00508         default: {
00509           wxLogDebug (wxT("RepaintBuffer> XACTION_XXX"));
00510           break;
00511         }
00512       }
00513     xJournalEntry= xJournalEntry -> previous;
00514     }
00515 #ifdef TRACE_CALLS
00516     wxLogDebug ( wxT("RepaintBuffer> xTCSJournal: Ptr= %p / Last Entry: Ptr= %p"),
      ActiveCanvas->xTCSJournal, xJournalEntry);
00517 #endif // TRACE_CALLS
00518  }
00519
00520
00521 /*
00522   Setting default values before reading the initialization files
00523 */
00524
00525 void PresetProgPar ()
00526 {
00527     TCSDefaultLinCol= TCS_INIDEF_LINCOL;
00528     TCSDefaultTxtCol= TCS_INIDEF_TXTCOL;
00529     TCSDefaultBckCol= TCS_INIDEF_BCKCOL;
00530
00531     TCSwindowIniXrelpos= TCS_INIDEF_WINPOSX;
00532     TCSwindowIniYrelpos= TCS_INIDEF_WINPOSY;
00533     TCSwindowIniXrelsiz= TCS_INIDEF_WINSIZX;
00534     TCSwindowIniYrelsiz= TCS_INIDEF_WINSIZY;
00535
00536     // No reset of windownames and initialisation files
00537
00538     // No reset of hardcopyname and counter
00539
00540     // Error messages should be changed only once
00541
00542 }
00543
00544
00545
00546 void CustomizeProgPar ()
00547 #if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN) // Get a safe buffer
00548   #define TMPSTRLEN TCS_FILE_NAMELEN
00549 #else
00550   #define TMPSTRLEN TCS_WINDOW_NAMELEN
00551 #endif
00552 {
00553   size_t iL;
00554   char* szTemp;
00555   char TmpStr[TMPSTRLEN];
00556   wxString wxTmpStr;
00557   wxFileName wxTmpFilNam;
00558
00559     szTemp= strstr (szTCSWindowName, PROGDIRTOKEN); // Default ProgDir?
00560     if (szTemp != NULL) {
00561       strncpy (TmpStr, szTCSWindowName, TMPSTRLEN);
00562       wxTmpFilNam= wxStandardPaths::Get().GetExecutablePath();
00563       wxTmpStr= wxTmpFilNam.GetFullName();
00564       iL= szTemp-szTCSWindowName+1;
00565       if ((TCS_WINDOW_NAMELEN-iL) > 1) {
00566         strncpy (szTemp, wxTmpStr, TCS_WINDOW_NAMELEN-iL);
00567         if ((TCS_WINDOW_NAMELEN-iL-wxTmpStr.length()) > 1) {
00568           strncpy (&szTCSWindowName[iL+wxTmpStr.length()-1],
00569                    &TmpStr[iL+strlen(PROGDIRTOKEN)-1], TCS_WINDOW_NAMELEN-iL-wxTmpStr.length());
00570         }
```

```
00571       }
00572       szTCSWindowName[TCS_WINDOW_NAMELEN-1]= '\0'; // just in case...
00573     }
00574 #undef TMPSTRLEN
00575 }
00576
00577
00578
00579 void XMLreadProgPar (const char * filname)
00580 {
00581   wxXmlDocument xmlDoc;
00582   wxXmlNode *node, *node1, *NodeSect0;
00583
00584   size_t iL;
00585
00586   long longTmp;
00587   wxString wxTmpStr;
00588
00589
00590     if (filname[0] != '\0') {
00591       if (!wxFileExists(filname)) {
00592          TCSGraphicError (ERR_XMLOPEN, filname); // No input file
00593          return; // give warning and continue with defaults
00594       }
00595       if (!xmlDoc.Load(filname)) {
00596         TCSGraphicError (ERR_XMLOPEN, filname); // Unknown file error
00597         return; // unexpected file error -> handle error in any case
00598       }
00599       if (xmlDoc.GetRoot() == nullptr) {
00600         TCSGraphicError (ERR_XMLOPEN, filname); // No root node
00601         return;
00602       }
00603       NodeSect0=  nullptr;
00604       if (xmlDoc.GetRoot()->GetName().IsSameAs(szTCSsect0)) {
00605         NodeSect0= xmlDoc.GetRoot() ;
00606       } else {
00607         node= xmlDoc.GetRoot()->GetChildren();
00608         while (node != nullptr) {
00609           if (node->GetName().IsSameAs(szTCSsect0)) {
00610             NodeSect0= node;
00611             break;
00612           }
00613           node= node->GetNext();
00614         }
00615       }
00616       if (NodeSect0 != nullptr) {
00617         node1= NodeSect0->GetChildren();
00618         while (node1 != nullptr) {
00619           if (node1->GetName().IsSameAs(TCS_INISECT1)) { // TCS_INISECT1: Names
00620             node= node1->GetChildren();
00621             while (node != nullptr) {
00622               if (node->GetName().IsSameAs(TCS_INIVAR_WINNAM)) {
00623                 iL= node->GetNodeContent().length();
00624                 if (iL > 0) {
00625                   wxTmpStr= node->GetNodeContent().Truncate(TCS_WINDOW_NAMELEN);
00626                   strncpy (szTCSWindowName, wxTmpStr.c_str(), TCS_WINDOW_NAMELEN);
00627                 }
00628               } else if (node->GetName().IsSameAs(TCS_INIVAR_STATNAM)) {
00629                 iL= node->GetNodeContent().length();
00630                 if (iL > 0) {
00631                   wxTmpStr= node->GetNodeContent().Truncate(TCS_WINDOW_NAMELEN);
00632                   strncpy (szTCSstatWindowName, wxTmpStr.c_str(), TCS_WINDOW_NAMELEN);
00633                 }
00634               } else if (node->GetName().IsSameAs(TCS_INIVAR_HDCNAM)) {
00635                 iL= node->GetNodeContent().length();
00636                 if (iL > 0) {
00637                   wxTmpStr= node->GetNodeContent().Truncate(TCS_FILE_NAMELEN);
00638                   strncpy (szTCSHardcopyFile, wxTmpStr.c_str(), TCS_FILE_NAMELEN);
00639                 }
00640               }
00641               node= node->GetNext();
00642             } // end dataloop TCS_INISECT1
00643
00644           } else if (node1->GetName().IsSameAs(TCS_INISECT2)) { // TCS_INISECT2: Layout
00645             node= node1->GetChildren();
00646             while (node != nullptr) {
00647               wxTmpStr= node->GetNodeContent();
00648               if (node->GetName().IsSameAs(TCS_INIVAR_WINPOSX)) {
00649                 if (wxTmpStr.IsNumber()) {
00650                   TCSwindowIniXrelpos= wxAtoi(wxTmpStr);
00651                 }
00652               } else if (node->GetName().IsSameAs(TCS_INIVAR_WINPOSY)) {
00653                 if (wxTmpStr.IsNumber()) {
00654                   TCSwindowIniYrelpos= wxAtoi(wxTmpStr);
00655                 }
00656               } else if (node->GetName().IsSameAs(TCS_INIVAR_WINSIZX)) {
00657                 if (wxTmpStr.IsNumber()) {
```

```
00658                   TCSwindowIniXrelsiz= wxAtoi(wxTmpStr);
00659                 }
00660               } else if (node->GetName().IsSameAs(TCS_INIVAR_WINSIZY)) {
00661                 if (wxTmpStr.IsNumber()) {
00662                   TCSwindowIniYrelsiz= wxAtoi(wxTmpStr);
00663                 }
00664 /*
00665               } else if (node->GetName().IsSameAs(TCS_INIVAR_STATPOSX)) {
00666                 if (wxTmpStr.IsNumber()) {
00667                   TCSstatWindowIniXrelpos= wxAtoi(wxTmpStr);
00668                 }
00669               } else if (node->GetName().IsSameAs(TCS_INIVAR_STATPOSY)) {
00670                 if (wxTmpStr.IsNumber()) {
00671                   TCSstatWindowIniYrelpos= wxAtoi(wxTmpStr);
00672                 }
00673               } else if (node->GetName().IsSameAs(TCS_INIVAR_STATSIZX)) {
00674                 if (wxTmpStr.IsNumber()) {
00675                   TCSstatWindowIniXrelsiz= wxAtoi(wxTmpStr);
00676                 }
00677               } else if (node->GetName().IsSameAs(TCS_INIVAR_STATSIZY)) {
00678                 if (wxTmpStr.IsNumber()) {
00679                   TCSstatWindowIniYrelsiz= wxAtoi(wxTmpStr);
00680                 }
00681 */
00682               } else if (node->GetName().IsSameAs(TCS_INIVAR_LINCOL)) {
00683                 if (wxTmpStr.IsNumber()) {
00684                   TCSDefaultLinCol= wxAtoi(wxTmpStr);
00685                 }
00686               } else if (node->GetName().IsSameAs(TCS_INIVAR_TXTCOL)) {
00687                 if (wxTmpStr.IsNumber()) {
00688                   TCSDefaultTxtCol= wxAtoi(wxTmpStr);
00689                 }
00690               } else if (node->GetName().IsSameAs(TCS_INIVAR_BCKCOL)) {
00691                 if (wxTmpStr.IsNumber()) {
00692                   TCSDefaultBckCol= wxAtoi(wxTmpStr);
00693                 }
00694               }
00695               node= node->GetNext();
00696             } // end dataloop TCS_INISECT2
00697           } else if (node1->GetName().IsSameAs(TCS_INISECT3)) { // TCS_INISECT3: Messages
00698             node= node1->GetChildren();
00699             while (node != nullptr) {
00700               wxTmpStr= node->GetNodeContent();
00701               if (node->GetName().IsSameAs(TCS_INIVAR_HDCOPN)) {
00702                 iL= node->GetNodeContent().length();
00703                 if (iL > 0) {
00704                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00705                   strncpy (szTCSErrorMsg[WRN_HDCFILOPN], wxTmpStr.c_str(), TCS_MESSAGELEN);
00706                 }
00707               } else if (node->GetName().IsSameAs(TCS_INIVAR_HDCOPNL)) {
00708                 if (wxTmpStr.IsNumber()) {
00709                   TCSErrorLev[WRN_HDCFILOPN]= wxAtoi(wxTmpStr);
00710                 }
00711
00712               } else if (node->GetName().IsSameAs(TCS_INIVAR_HDCWRT)) {
00713                 iL= node->GetNodeContent().length();
00714                 if (iL > 0) {
00715                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00716                   strncpy (szTCSErrorMsg[WRN_HDCFILWRT], wxTmpStr.c_str(), TCS_MESSAGELEN);
00717                 }
00718               } else if (node->GetName().IsSameAs(TCS_INIVAR_HDCWRTL)) {
00719                 if (wxTmpStr.IsNumber()) {
00720                   TCSErrorLev[WRN_HDCFILWRT]= wxAtoi(wxTmpStr);
00721                 }
00722
00723               } else if (node->GetName().IsSameAs(TCS_INIVAR_USR)) {
00724                 iL= node->GetNodeContent().length();
00725                 if (iL > 0) {
00726                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00727                   strncpy (szTCSErrorMsg[MSG_USR], wxTmpStr.c_str(), TCS_MESSAGELEN);
00728                 }
00729               } else if (node->GetName().IsSameAs(TCS_INIVAR_USRL)) {
00730                 if (wxTmpStr.IsNumber()) {
00731                   TCSErrorLev[MSG_USR]= wxAtoi(wxTmpStr);
00732                 }
00733
00734               } else if (node->GetName().IsSameAs(TCS_INIVAR_HDCACT)) {
00735                 iL= node->GetNodeContent().length();
00736                 if (iL > 0) {
00737                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00738                   strncpy (szTCSErrorMsg[MSG_HDCACT], wxTmpStr.c_str(), TCS_MESSAGELEN);
00739                 }
00740               } else if (node->GetName().IsSameAs(TCS_INIVAR_HDCACTL)) {
00741                 if (wxTmpStr.IsNumber()) {
00742                   TCSErrorLev[MSG_HDCACT]= wxAtoi(wxTmpStr);
00743                 }
00744
```

```
00745               } else if (node->GetName().IsSameAs(TCS_INIVAR_USRWRN)) {
00746                 iL= node->GetNodeContent().length();
00747                 if (iL > 0) {
00748                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00749                   strncpy (szTCSErrorMsg[WRN_USRPRESSANY], wxTmpStr.c_str(), TCS_MESSAGELEN);
00750                 }
00751               } else if (node->GetName().IsSameAs(TCS_INIVAR_USRWRNL)) {
00752                 if (wxTmpStr.IsNumber()) {
00753                   TCSErrorLev[WRN_USRPRESSANY]= wxAtoi(wxTmpStr);
00754                 }
00755
00756               } else if (node->GetName().IsSameAs(TCS_INIVAR_EXIT)) {
00757                 iL= node->GetNodeContent().length();
00758                 if (iL > 0) {
00759                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00760                   strncpy (szTCSErrorMsg[ERR_EXIT], wxTmpStr.c_str(), TCS_MESSAGELEN);
00761                 }
00762               } else if (node->GetName().IsSameAs(TCS_INIVAR_EXITL)) {
00763                 if (wxTmpStr.IsNumber()) {
00764                   TCSErrorLev[ERR_EXIT]= wxAtoi(wxTmpStr);
00765                 }
00766
00767               } else if (node->GetName().IsSameAs(TCS_INIVAR_JOUCREATE)) {
00768                 iL= node->GetNodeContent().length();
00769                 if (iL > 0) {
00770                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00771                   strncpy (szTCSErrorMsg[WRN_JOUCREATE], wxTmpStr.c_str(), TCS_MESSAGELEN);
00772                 }
00773               } else if (node->GetName().IsSameAs(TCS_INIVAR_JOUCREATEL)) {
00774                 if (wxTmpStr.IsNumber()) {
00775                   TCSErrorLev[WRN_JOUCREATE]= wxAtoi(wxTmpStr);
00776                 }
00777
00778               } else if (node->GetName().IsSameAs(TCS_INIVAR_JOUENTRY)) {
00779                 iL= node->GetNodeContent().length();
00780                 if (iL > 0) {
00781                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00782                   strncpy (szTCSErrorMsg[WRN_JOUENTRY], wxTmpStr.c_str(), TCS_MESSAGELEN);
00783                 }
00784               } else if (node->GetName().IsSameAs(TCS_INIVAR_JOUENTRYL)) {
00785                 if (wxTmpStr.IsNumber()) {
00786                   TCSErrorLev[WRN_JOUENTRY]= wxAtoi(wxTmpStr);
00787                 }
00788
00789               } else if (node->GetName().IsSameAs(TCS_INIVAR_JOUADD)) {
00790                 iL= node->GetNodeContent().length();
00791                 if (iL > 0) {
00792                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00793                   strncpy (szTCSErrorMsg[WRN_JOUADD], wxTmpStr.c_str(), TCS_MESSAGELEN);
00794                 }
00795               } else if (node->GetName().IsSameAs(TCS_INIVAR_JOUADDL)) {
00796                 if (wxTmpStr.IsNumber()) {
00797                   TCSErrorLev[WRN_JOUADD]= wxAtoi(wxTmpStr);
00798                 }
00799
00800               } else if (node->GetName().IsSameAs(TCS_INIVAR_XMLOPEN)) {
00801                 iL= node->GetNodeContent().length();
00802                 if (iL > 0) {
00803                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00804                   strncpy (szTCSErrorMsg[ERR_XMLOPEN], wxTmpStr.c_str(), TCS_MESSAGELEN);
00805                 }
00806               } else if (node->GetName().IsSameAs(TCS_INIVAR_XMLOPENL)) {
00807                 if (wxTmpStr.IsNumber()) {
00808                   TCSErrorLev[ERR_XMLOPEN]= wxAtoi(wxTmpStr);
00809                 }
00810
00811               } else if (node->GetName().IsSameAs(TCS_INIVAR_USR2)) {
00812                 iL= node->GetNodeContent().length();
00813                 if (iL > 0) {
00814                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00815                   strncpy (szTCSErrorMsg[MSG_USR2], wxTmpStr.c_str(), TCS_MESSAGELEN);
00816                 }
00817               } else if (node->GetName().IsSameAs(TCS_INIVAR_USR2L)) {
00818                 if (wxTmpStr.IsNumber()) {
00819                   TCSErrorLev[MSG_USR2]= wxAtoi(wxTmpStr);
00820                 }
00821
00822               } else if (node->GetName().IsSameAs(TCS_INIVAR_INI2)) {
00823                 iL= node->GetNodeContent().length();
00824                 if (iL > 0) {
00825                   wxTmpStr= node->GetNodeContent().Truncate(TCS_MESSAGELEN);
00826                   strncpy (szTCSErrorMsg[WRN_INI2], wxTmpStr.c_str(), TCS_MESSAGELEN);
00827                 }
00828               } else if (node->GetName().IsSameAs(TCS_INIVAR_INI2L)) {
00829                 if (wxTmpStr.IsNumber()) {
00830                   TCSErrorLev[WRN_INI2]= wxAtoi(wxTmpStr);
00831                 }
```

```
00832
00833               }
00834               node= node->GetNext();
00835             } // end dataloop TCS_INISECT3
00836           }
00837           node1= node1->GetNext();
00838         }
00839       }
00840     }
00841  }
00842
00843
00844
00845 /* --------- Object cTCScanvas ------ */
00846
00847
00848 cTCScanvas::cTCScanvas(int iMode, wxFrame* parent, wxFrame* FrameToUse, wxStatusBar* StatusBarToUse)
00849 {
00850   wxRect Screen;
00851   wxSize UseScreen, MinScreen;
00852   wxPoint PosScreen;
00853
00854     if (iMode == 0) return;
00855
00856     if (FrameToUse == nullptr) {
00857       ID_TCSframe = wxNewId();  // TCSframe->GetID()
00858       TCSframe= new wxFrame(parent, ID_TCSframe, szTCSWindowName, wxDefaultPosition, wxDefaultSize,
      wxDEFAULT_FRAME_STYLE, wxString::Format(wxT("%i"),ID_TCSframe));
00859       TCSstatusBar= TCSframe->GetStatusBar();
00860     } else {
00861       TCSframe= FrameToUse; // Use given plot frame
00862       ID_TCSframe = FrameToUse->GetId();
00863     }
00864
00865     TCSstatusBar= StatusBarToUse;
00866     if ( StatusBarToUse != nullptr ) {
00867       ID_TCSstatus = TCSstatusBar->GetId();
00868     } else {
00869       ID_TCSstatus = wxID_NONE;
00870     }
00871
00872     if (iMode <= 2) { // New window: use screensize/title from TCS initialization
00873       Screen = wxGetClientDisplayRect (); // usable screen size
00874       if (TCSwindowIniYrelsiz > 0) {
00875         UseScreen.x = TCSwindowIniXrelsiz * Screen.width / 100;
00876         UseScreen.y = TCSwindowIniYrelsiz * Screen.height / 100;  // TCSframe->GetMaxClientSize()
00877         PosScreen.x = TCSwindowIniXrelpos * Screen.width / 100;
00878         PosScreen.y = TCSwindowIniYrelpos * Screen.height / 100;  // TCSframe->GetScreenPosition()
00879         MinScreen = wxSize(-1,-1); // No restriction
00880       }
00881       if (strlen(szTCSWindowName)>0) TCSframe->SetLabel(szTCSWindowName); // only for iMode=2 relevant
00882
00883       if (TCSstatusBar == nullptr) {
00884         ID_TCSstatus = wxNewId();
00885         TCSstatusBar = new wxStatusBar(TCSframe, ID_TCSstatus, wxSTB_DEFAULT_STYLE,
      wxString::Format(wxT("%i"),ID_TCSstatus));
00886         TCSstatusBar->SetFieldsCount(1);
00887         TCSframe->SetStatusBar(TCSstatusBar);
00888       }
00889     } else { // keep current screensize and title
00890       UseScreen = TCSframe->GetClientSize ();
00891       PosScreen = wxPoint(-1,-1); // x < 0 -> don't touch position
00892       MinScreen = UseScreen; // don't allow screensize 0
00893     }
00894     CompleteCanvas(UseScreen, PosScreen, MinScreen);
00895 }
00896
00897
00898
00899 void cTCScanvas::CompleteCanvas (wxSize UseScreen, wxPoint PosScreen, wxSize MinScreen)
00900 {
00901   wxBoxSizer* TCSBoxSizer;
00902     ID_TCSpanel = wxNewId();
00903     TCSpanel = new wxPanel(TCSframe, ID_TCSpanel, wxDefaultPosition, UseScreen, wxTAB_TRAVERSAL,
      wxString::Format(wxT("%i"),ID_TCSpanel));
00904     TCSpanel->SetMinSize(MinScreen);
00905     TCSpanel->SetMaxSize(wxSize(-1,-1));
00906     TCSBoxSizer = new wxBoxSizer(wxHORIZONTAL);
00907     TCSBoxSizer->Add(TCSpanel, 1, wxALL|wxEXPAND, 5);
00908     TCSframe->SetSizer(TCSBoxSizer);
00909     TCSBoxSizer->Fit(TCSframe);
00910     TCSBoxSizer->SetSizeHints(TCSframe);
00911
00912     TCSframe->SetClientSize (UseScreen);
00913     if (PosScreen.x > 0) {
00914      TCSframe->Move (PosScreen);
00915     }
```

```
00916
00917     TCSframe->Connect(wxID_ANY,wxEVT_CLOSE_WINDOW,(wxObjectEventFunction)&cTCScanvas::OnTCSClose);
00918
00919
      TCSpanel->Connect(wxEVT_PAINT,(wxObjectEventFunction)&cTCScanvas::OnTCSpanelPaint,0,this->TCSframe);
00920     TCSpanel->Connect(wxEVT_SIZE,
     (wxObjectEventFunction)&cTCScanvas::OnTCSpanelResize,0,this->TCSframe);
00921     TCSpanel->Connect(wxEVT_KEY_DOWN,(wxObjectEventFunction)&cTCScanvas::OnTCSpanelKey);
00922     TCSpanel->Connect(wxEVT_LEFT_DOWN ,(wxObjectEventFunction)&cTCScanvas::OnTCSmouseLeft);
00923     TCSpanel->Connect(wxEVT_MIDDLE_DOWN ,(wxObjectEventFunction)&cTCScanvas::OnTCSmouseMiddle);
00924     TCSpanel->Connect(wxEVT_RIGHT_DOWN ,(wxObjectEventFunction)&cTCScanvas::OnTCSmouseRight);
00925 }
00926
00927
00928
00929 cTCScanvas::~cTCScanvas()
00930 {
00931     finitt_ (NULL, NULL); // -> Destroy ();
00932 }
00933
00934
00935 void cTCScanvas::OnTCSClose(wxCloseEvent& event)
00936 {
00937     if ((event.GetId() == ActiveCanvas->ID_TCSframe) ||
00938                         (event.GetId() == ActiveCanvas->ID_TCSpanel)) {
00939       finitt_ (NULL, NULL); // -> Destroy ();
00940     }
00941 }
00942
00943
00944 void cTCScanvas::OnTCSpanelPaint(wxPaintEvent& event)
00945 {
00946   wxWindowID RequestingWindowID, WorkWindowID;
00947
00948     WorkWindowID = ActiveCanvasID; // store for further plotting
00949     RequestingWindowID = getCanvasID (event.GetId());
00950     if (RequestingWindowID >= 0) { // requested window belongs to a TCScanvas
00951       if (RequestingWindowID != WorkWindowID) WINSELECT (&RequestingWindowID);
00952       wxPaintDC dc (ActiveCanvas->TCSpanel);
00953       dc.GetSize (&tktrnx_.kScrX, &tktrnx_.kScrY);
00954       dc.SetAxisOrientation (true, true);       // y-axis bottom->up
00955       dc.SetDeviceOrigin (0., tktrnx_.kScrY);  // (0,0) lower left corner
00956       dc.SetLogicalScale (tktrnx_.kScrX/TEK_XMAX, tktrnx_.kScrY/TEK_YMAX);
00957       RepaintBuffer (dc);
00958       if (RequestingWindowID != WorkWindowID) WINSELECT (&WorkWindowID);
00959     }
00960 }
00961
00962
00963
00964 void cTCScanvas::OnTCSpanelResize(wxSizeEvent& event)
00965 {
00966   wxWindowID RequestingWindowID;
00967
00968     RequestingWindowID = getCanvasID (event.GetId());
00969     if (RequestingWindowID >= 0) { // requesting window belongs to a TCScanvas
00970       OpenCanvases[RequestingWindowID]->TCSpanel->Refresh (); // Redraw with new scale -> wxEVT_PAINT
00971     } // Only OnTCSpanelPaint() switches windows
00972 }
00973
00974
00975
00976 void cTCScanvas::OnTCSpanelKey(wxKeyEvent& event)
00977 {
00978     ActiveCanvas->TCSpanelKeyPressed= event.GetKeyCode();
00979     if ((!event.m_shiftDown) && (ActiveCanvas->TCSpanelKeyPressed > 0x40)
00980                              && (ActiveCanvas->TCSpanelKeyPressed < 0x5b) ) {
00981       ActiveCanvas->TCSpanelKeyPressed+= 0x20; // lower case ASCII
00982     }
00983 }
00984
00985
00986
00987 void cTCScanvas::OnTCSmouseLeft(wxMouseEvent& event)
00988 {
00989     ActiveCanvas->TCSmouseButtonDown= 1;
00990     event.GetPosition(&ActiveCanvas->TCSmouseX, &ActiveCanvas->TCSmouseY);
00991     ActiveCanvas->TCSmouseX= ActiveCanvas->TCSmouseX * TEK_XMAX/tktrnx_.kScrX;
00992     ActiveCanvas->TCSmouseY= TEK_YMAX - (ActiveCanvas->TCSmouseY * TEK_YMAX/tktrnx_.kScrY);
00993 }
00994
00995
00996
00997 void cTCScanvas::OnTCSmouseMiddle(wxMouseEvent& event)
00998 {
00999     ActiveCanvas->TCSmouseButtonDown= 4; // same as in DOS-port
01000     event.GetPosition(&ActiveCanvas->TCSmouseX, &ActiveCanvas->TCSmouseY);
```

```
01001     ActiveCanvas->TCSmouseX= ActiveCanvas->TCSmouseX * TEK_XMAX/tktrnx_.kScrX;
01002     ActiveCanvas->TCSmouseY= TEK_YMAX - (ActiveCanvas->TCSmouseY * TEK_YMAX/tktrnx_.kScrY);
01003 }
01004
01005
01006 void cTCScanvas::OnTCSmouseRight(wxMouseEvent& event)
01007 {
01008     ActiveCanvas->TCSmouseButtonDown= 2;
01009     event.GetPosition(&ActiveCanvas->TCSmouseX, &ActiveCanvas->TCSmouseY);
01010     ActiveCanvas->TCSmouseX= ActiveCanvas->TCSmouseX * TEK_XMAX/tktrnx_.kScrX;
01011     ActiveCanvas->TCSmouseY= TEK_YMAX - (ActiveCanvas->TCSmouseY * TEK_YMAX/tktrnx_.kScrY);
01012 }
01013
01014
01015
01016 /*
01017 ------------------- Userroutinen: Initialization -----------------------------
01018 */
01019
01020
01021 extern "C" {
01022     void winlbl0 (const char PloWinNam[], const char StatWinNam[], const char IniFilNam[])
01023 #if (TCS_WINDOW_NAMELEN <= TCS_FILE_NAMELEN) // Get a safe buffer
01024   #define TMPSTRLEN TCS_FILE_NAMELEN
01025 #else
01026   #define TMPSTRLEN TCS_WINDOW_NAMELEN
01027 #endif
01028     {
01029       size_t iL;
01030       char* szTemp;
01031       char  tmpstr[TMPSTRLEN], PathSeparator[2];
01032
01033         iL= strlen(PloWinNam);
01034         if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
01035         if (iL > 0) {
01036           strncpy( szTCSWindowName, PloWinNam, iL); // Destination is zero-padded
01037           szTCSWindowName[iL]= '\0'; // just in case iL>= TCS_WINDOW_NAMELEN
01038         }
01039
01040         iL= strlen(StatWinNam);
01041         if (iL > (TCS_WINDOW_NAMELEN-1)) iL= TCS_WINDOW_NAMELEN-1;
01042         if (iL > 0) {
01043           strncpy( szTCSstatWindowName, StatWinNam, iL);
01044           szTCSstatWindowName[iL]= '\0';
01045         }
01046
01047         iL= strlen(IniFilNam);
01048         if (iL > (TCS_FILE_NAMELEN-1)) iL= TCS_FILE_NAMELEN-1;
01049         if (iL > 0) {
01050           strncpy( szTCSIniFile, IniFilNam, iL);
01051           szTCSIniFile[iL]= '\0';
01052           szTemp= strstr (szTCSIniFile, "@"); // section Level0?
01053           if (szTemp != 0) {
01054             strncpy (szTCSsect0, &szTemp[1], iL); // len(szSect0)=TCS_FILE_NAMELEN --> iL o.k.
01055             szTemp[0]= '\0'; // cut of @Section0 in szTCSIniFile
01056           }
01057         }
01058         iL= strlen(szTCSIniFile); // perhaps shortened by @ processing
01059         if (iL > 0) {
01060           szTemp= strstr (szTCSIniFile, INIFILEXTTOKEN); // Default extension?
01061           if (szTemp != 0) {
01062             iL= TCS_FILE_NAMELEN + szTCSIniFile-szTemp;
01063             strncpy (szTemp, INIFILEXT, iL); // Sideeffect: szTCSIniFile with extension
01064             szTCSIniFile[TCS_FILE_NAMELEN-1]= '\0'; // just in case...
01065           }
01066         }
01067         iL= strlen(szTCSIniFile); // perhaps extended by .% processing
01068         if (iL > 0) {
01069           szTemp= strstr (szTCSIniFile, PROGDIRTOKEN); // Default ProgDir?
01070           if (szTemp == szTCSIniFile) {
01071             strncpy (tmpstr, szTCSIniFile, TCS_FILE_NAMELEN);
01072             strncpy (szTCSIniFile, wxStandardPaths::Get().GetDataDir(), TCS_FILE_NAMELEN);
01073             iL= strlen(szTCSIniFile);
01074             PathSeparator[0]= wxFileName::GetPathSeparator();
01075             PathSeparator[1]= char (0);
01076             strncpy (&szTCSIniFile[iL], PathSeparator, TCS_FILE_NAMELEN-iL-2); // -2: length Path
      separator
01077             iL= strlen(szTCSIniFile);
01078             strncpy (&szTCSIniFile[iL], &tmpstr[strlen(PROGDIRTOKEN)], TCS_FILE_NAMELEN-iL);
01079             szTCSIniFile[TCS_FILE_NAMELEN-1]= '\0'; // just in case...
01080           }
01081         }
01082      }
01083 #undef TMPSTRLEN
01084 }
01085
01086
```

```
01087
01088 extern "C" {
01089     bool WINSELECT (wxWindowID* iD)
01090     {
01091       size_t numbytes;
01092
01093         if (*iD >= MAX_OPEN_CANVAS) {
01094           TCSGraphicError (WRN_INI2," ");
01095           return true; // Error handling !?
01096         } else {
01097           if (ActiveCanvas != nullptr) { // already active -> save status
01098             numbytes= sizeof (struct TKTRNX); // save TKTRNX
01099             memmove (&ActiveCanvas->TekSav.khomey, &tktrnx_.khomey, numbytes);
01100             numbytes= sizeof (struct G2dAG2); // save AG2
01101             memmove (&ActiveCanvas->AG2Sav.cline, &g2dag2_.cline, numbytes);
01102
01103             ActiveCanvas->DefaultLinColSav = TCSDefaultLinCol;
01104             ActiveCanvas->DefaultTxtColSav = TCSDefaultTxtCol;
01105             ActiveCanvas->DefaultBckColSav = TCSDefaultBckCol;
01106             memmove (ActiveCanvas->HardcopyFileSav, szTCSHardcopyFile, TCS_FILE_NAMELEN);
01107             memmove (ActiveCanvas->sect0Sav, szTCSsect0, TCS_FILE_NAMELEN);
01108           }
01109           if (OpenCanvases[*iD] != nullptr) { // restore TKTRNX
01110             numbytes= sizeof (struct G2dAG2);
01111             memmove (&tktrnx_.khomey, &OpenCanvases[*iD]->TekSav.khomey, numbytes);
01112             numbytes= sizeof (struct G2dAG2);
01113             memmove (&g2dag2_.cline, &OpenCanvases[*iD]->AG2Sav.cline, numbytes);
01114
01115             TCSDefaultLinCol = OpenCanvases[*iD]->DefaultLinColSav;
01116             TCSDefaultTxtCol = OpenCanvases[*iD]->DefaultTxtColSav;
01117             TCSDefaultBckCol = OpenCanvases[*iD]->DefaultBckColSav;
01118             memmove (szTCSHardcopyFile,&OpenCanvases[*iD]->HardcopyFileSav, TCS_FILE_NAMELEN);
01119             memmove (szTCSsect0, &OpenCanvases[*iD]->sect0Sav, TCS_FILE_NAMELEN);
01120           }
01121           ActiveCanvasID= *iD;
01122           ActiveCanvas= OpenCanvases[*iD];
01123       }
01124        return (OpenCanvases[*iD] == nullptr);
01125     }
01126 }
01127
01128
01129 extern "C" {
01130     void initt1 (int iMode, wxFrame* parent, wxFrame* FrameToUse, wxStatusBar* StatusBarToUse)
01131     {
01132        wxSize UseScreen;
01133        xJournalEntry_typ * xJournalEntry;
01134
01135          PresetProgPar(); // restore initialization after finitt()
01136          XMLreadProgPar (szTCSIniFile);
01137          CustomizeProgPar (); // substitute %: with program directory
01138          initt0(); // initialize COMMON TKTRNX
01139
01140          if (ActiveCanvas != NULL) { // Reset journal
01141            SGLIB_DL_LIST_MAP_ON_ELEMENTS (xJournalEntry_typ, ActiveCanvas->xTCSJournal,
01142                          xJournalEntry,previous,next, { free (xJournalEntry);}); // free all
01143            ActiveCanvas->xTCSJournal= NULL;
01144            xJournalEntry= (xJournalEntry_typ *)  malloc (sizeof (xJournalEntry_typ));
01145            if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");
01146            xJournalEntry->action=  XACTION_NOOP; // mark  beginning of the list with NOOP
01147            xJournalEntry->i1= 0;
01148            xJournalEntry->i2= 0;
01149            SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01150            xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01151            if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
01152            xJournalEntry->action=  XACTION_INITT;
01153            xJournalEntry->i1= 0;
01154            xJournalEntry->i2= 0;
01155            SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01156            return; // Remaining reset will be done during redraw due to XACTION_INITT
01157          }
01158
01159          ActiveCanvas = new cTCScanvas (iMode, parent, FrameToUse, StatusBarToUse);
01160          OpenCanvases[ActiveCanvasID] = ActiveCanvas;
01161
01162          ActiveCanvas->TCSpen = wxPen(TCSColorTable[tktrnx_.iLinCol], TCS_LINEWIDTH,
       wxPENSTYLE_SOLID);
01163          ActiveCanvas->TCSbrush = wxBrush (TCSColorTable[tktrnx_.iBckCol], wxBRUSHSTYLE_SOLID);
01164          ActiveCanvas->TCSfont = wxFont(wxFONTSIZE_MEDIUM, wxFONTFAMILY_TELETYPE,
01165                                         wxFONTSTYLE_NORMAL, wxFONTWEIGHT_NORMAL, false);
01166
01167          UseScreen = ActiveCanvas->TCSpanel->GetClientSize ();
01168          tktrnx_.kversz = (int) (TEK_YMAX *TCS_REL_CHR_HEIGHT +0.5); // first guess
01169          tktrnx_.khorsz = (int) ((float)UseScreen.y/(float)UseScreen.x*(float)tktrnx_.kversz +0.5);
01170          ActiveCanvas->TCSfont.SetFractionalPointSize (TEK_YMAX *TCS_REL_CHR_HEIGHT);
```

```
01171
01172           tktrnx_.khomey= TEK_YMAX - tktrnx_.kversz;
01173           tktrnx_.kbeamx = tktrnx_.klmrgn;  //  call HOME
01174           tktrnx_.kbeamy = tktrnx_.khomey;
01175
01176           ActiveCanvas->TCSframe->Show();
01177
01178           //  Logging Window
01179
01180           ActiveCanvas->logWindow = new wxLogWindow(ActiveCanvas->TCSframe, szTCSstatWindowName, false,
       false);
01181           wxLog::SetActiveTarget(ActiveCanvas->logWindow);
01182           wxLog::SetTimestamp(""); // don't write timestamps before messages
01183           wxLogStatus ("); // without a first message wxLog::show() will crash
01184
01185           //  Create journal
01186
01187           ActiveCanvas->xTCSJournal = (xJournalEntry_typ*) NULL;
01188           wxLogDebug ( wxT("INITT1> xTCSJournal initialisiert: Ptr= %p"), ActiveCanvas->xTCSJournal);
01189
01190           xJournalEntry= (xJournalEntry_typ *)  malloc (sizeof (xJournalEntry_typ));
01191           if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUCREATE,"");
01192 #ifdef TRACE_CALLS
01193           wxLogDebug ( wxT("INITT1> Nach 1. malloc: xJournalEntry: Ptr= %p"), xJournalEntry);
01194 #endif // TRACE_CALLS
01195
01196           xJournalEntry->action=  XACTION_NOOP; // mark  beginning of the list with NOOP
01197           xJournalEntry->i1= 0;
01198           xJournalEntry->i2= 0;
01199           SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01200 #ifdef TRACE_CALLS
01201           wxLogDebug ( wxT("INITT1> LIST_ADD=Create Journal: xTCSJournal: Ptr= %p / xJournalEntry: Ptr=
       %p"), ActiveCanvas->xTCSJournal, xJournalEntry);
01202           wxLogDebug ( wxT("INITT1> previous: Ptr= %p / next: Ptr= %p"), xJournalEntry -> previous,
       xJournalEntry -> next);
01203           wxLogDebug ( wxT("INITT1> XACTION_??? = %i (i1= %i, i2= %i)"), xJournalEntry->action,
       xJournalEntry->i1, xJournalEntry->i2 );
01204 #endif // TRACE_CALLS
01205
01206           xJournalEntry= (xJournalEntry_typ *)  malloc (sizeof (xJournalEntry_typ));
01207           if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUENTRY,"");
01208           xJournalEntry->action=  XACTION_INITT;
01209           xJournalEntry->i1= 0;
01210           xJournalEntry->i2= 0;
01211           SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01212 #ifdef TRACE_CALLS
01213           wxLogDebug ( wxT("INITT1> Nach 2. LIST_ADD=Create Journal: xTCSJournal: Ptr= %p /
       xJournalEntry: Ptr= %p"), ActiveCanvas->xTCSJournal, xJournalEntry);
01214           wxLogDebug ( wxT("INITT1> previous: Ptr= %p / next: Ptr= %p"), xJournalEntry -> previous,
       xJournalEntry -> next);
01215           wxLogDebug ( wxT("INITT1> XACTION_??? = %i (i1= %i, i2= %i)"), xJournalEntry->action,
       xJournalEntry->i1, xJournalEntry->i2 );
01216 #endif // TRACE_CALLS
01217
01218           return;
01219       }
01220 }
01221
01222
01223
01224 extern "C" {
01225     void FINITT (int* ix, int* iy)
01226     {
01227       cTCScanvas* CanvasToKill;
01228       xJournalEntry_typ * xJournalEntry;
01229
01230         if (ActiveCanvas == NULL) return;
01231         CanvasToKill = ActiveCanvas; // Window could be changed due to user action
01232         do {
01233           if (ActiveCanvas == CanvasToKill) {
01234             TCSGraphicError (ERR_EXIT,""); // User can accept or change window here
01235           } else {
01236             wxYield(); // Allow processing in case of a changed window
01237           }
01238         } while (ActiveCanvas != CanvasToKill); // Don't kill a wrong window
01239
01240         SGLIB_DL_LIST_MAP_ON_ELEMENTS (xJournalEntry_typ, ActiveCanvas->xTCSJournal,
01241                         xJournalEntry,previous,next, { free (xJournalEntry);}); // free all
01242         ActiveCanvas->xTCSJournal= nullptr;
01243
01244         ActiveCanvas->TCSframe->Destroy();
01245         ActiveCanvas = nullptr;
01246         OpenCanvases[ActiveCanvasID] = nullptr;
01247
01248         return;
```

```
01249     }
01250 }
01251
01252
01253
01254 extern "C" {
01255     void IOWAIT (int* iWait)
01256     {
01257         ActiveCanvas->TCSpanel->Refresh(); // wxEVT_PAINT will be executed after wxYield()
01258         wxYield();                   // process event loop -> be aware of recursive loops!
01259     }
01260 }
01261
01262
01263
01264 /*
01265 ------------------- TCS API: Drawing -----------------------------------
01266 */
01267
01268
01269
01270 extern "C" {
01271     void swindl_ (int* ix1, int* iy1, int* ix2, int* iy2)
01272     {
01273       xJournalEntry_typ * xJournalEntry;
01274
01275         ActiveCanvas->ClippingNotActive = (*ix1==0) && (*iy1==0) &&
01276                                           (*ix2==TEK_XMAX) && (*iy2==TEK_YMAX);
01277         /* Same meaning of bool variable in DOS, SDL2 ... */
01278
01279         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01280         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01281         xJournalEntry->action=  XACTION_CLIP;
01282         if (ActiveCanvas->ClippingNotActive) {
01283           xJournalEntry->i1= 0;
01284         } else {
01285           xJournalEntry->i1= 1;
01286         }
01287         xJournalEntry->i2= 0;
01288         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01289
01290         if (!ActiveCanvas->ClippingNotActive) {
01291           xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01292           if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01293           xJournalEntry->action=  XACTION_CLIP1;
01294           xJournalEntry->i1= *ix1;
01295           xJournalEntry->i2= *iy1;
01296           SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01297
01298           xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01299           if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01300           xJournalEntry->action=  XACTION_CLIP2;
01301           xJournalEntry->i1= *ix2;
01302           xJournalEntry->i2= *iy2;
01303           SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01304         }
01305     }
01306 }
01307
01308
01309
01310 extern "C" {
01311     void ERASE (void)
01312     {
01313       xJournalEntry_typ * xJournalEntry;
01314
01315         SGLIB_DL_LIST_MAP_ON_ELEMENTS (xJournalEntry_typ, ActiveCanvas->xTCSJournal,
01316                         xJournalEntry,previous,next, {free (xJournalEntry);}); // free all
01317         ActiveCanvas->xTCSJournal= NULL; // create new journal
01318
01319         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01320         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01321         xJournalEntry->action=  XACTION_NOOP; // root element without predecessor;
01322         xJournalEntry->i1= 0;
01323         xJournalEntry->i2= 0;
01324         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01325
01326         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01327         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01328         xJournalEntry->action=  XACTION_LINCOL;
01329         xJournalEntry->i1= tktrnx_.iLinCol;
01330         xJournalEntry->i2= 0;
01331         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
```

```
      next)
01332
01333         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01334         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01335         xJournalEntry->action=  XACTION_TXTCOL;
01336         xJournalEntry->i1= tktrnx_.iTxtCol;
01337         xJournalEntry->i2= 0;
01338         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01339
01340         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01341         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01342         xJournalEntry->action=  XACTION_BCKCOL;
01343         xJournalEntry->i1= tktrnx_.iBckCol;
01344         xJournalEntry->i2= 0;
01345         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01346
01347         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01348         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01349         xJournalEntry->action=  XACTION_ERASE;
01350         xJournalEntry->i1= 0;
01351         xJournalEntry->i2= 0;
01352         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01353     }
01354 }
01355
01356
01357
01358 extern "C" {
01359     void MOVABS (int* ix,int* iy)
01360     {
01361       xJournalEntry_typ * xJournalEntry;
01362
01363         tktrnx_.kbeamx= *ix;
01364         tktrnx_.kbeamy= *iy;
01365
01366         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01367         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01368         xJournalEntry->action=  XACTION_MOVABS;
01369         xJournalEntry->i1= *ix;
01370         xJournalEntry->i2= *iy;
01371         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01372     }
01373 }
01374
01375
01376
01377 extern "C" {
01378     void DRWABS (int* ix,int* iy)
01379     {
01380       xJournalEntry_typ * xJournalEntry;
01381
01382         tktrnx_.kbeamx= *ix;
01383         tktrnx_.kbeamy= *iy;
01384
01385         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01386         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01387         xJournalEntry->action=  XACTION_DRWABS;
01388         xJournalEntry->i1= *ix;
01389         xJournalEntry->i2= *iy;
01390         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01391     }
01392 }
01393
01394
01395
01396 extern "C" {
01397     void DSHABS (int* ix,int* iy, int* iMask)
01398     {
01399       xJournalEntry_typ * xJournalEntry;
01400
01401         tktrnx_.kbeamx= *ix;
01402         tktrnx_.kbeamy= *iy;
01403
01404         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01405         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01406         xJournalEntry->action=  XACTION_DSHSTYLE;
01407         xJournalEntry->i1= *iMask;
01408         xJournalEntry->i2= 0;
01409         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01410
01411         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
```

```
01412         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01413         xJournalEntry->action=  XACTION_DSHABS;
01414         xJournalEntry->i1= *ix;
01415         xJournalEntry->i2= *iy;
01416         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01417     }
01418 }
01419
01420
01421
01422 extern "C" {
01423     void PNTABS (int* ix,int* iy)
01424     {
01425       xJournalEntry_typ * xJournalEntry;
01426
01427         tktrnx_.kbeamx= *ix;
01428         tktrnx_.kbeamy= *iy;
01429
01430         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01431         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01432         xJournalEntry->action=  XACTION_PNTABS;
01433         xJournalEntry->i1= *ix;
01434         xJournalEntry->i2= *iy;
01435         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01436     }
01437 }
01438
01439
01440
01441 extern "C" {
01442     void BCKCOL (int* iCol)
01443     {
01444       xJournalEntry_typ * xJournalEntry;
01445
01446         tktrnx_.iBckCol= *iCol;
01447         if (*iCol > MAX_COLOR_INDEX) tktrnx_.iBckCol= MAX_COLOR_INDEX;
01448
01449         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01450         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01451         xJournalEntry->action=  XACTION_BCKCOL;
01452         xJournalEntry->i1= tktrnx_.iBckCol;
01453         xJournalEntry->i2= 0;
01454         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01455     }
01456 }
01457
01458
01459
01460 extern "C" {
01461     void LINCOL (int* iCol)
01462     {
01463       xJournalEntry_typ * xJournalEntry;
01464
01465         tktrnx_.iLinCol= *iCol;
01466         if (*iCol > MAX_COLOR_INDEX) tktrnx_.iLinCol= MAX_COLOR_INDEX;
01467
01468         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01469         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01470         xJournalEntry->action=  XACTION_LINCOL;
01471         xJournalEntry->i1= tktrnx_.iLinCol;
01472         xJournalEntry->i2= 0;
01473         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01474     }
01475 }
01476
01477
01478
01479 extern "C" {
01480     void TXTCOL (int* iCol)
01481     {
01482       xJournalEntry_typ * xJournalEntry;
01483
01484         tktrnx_.iTxtCol= *iCol;
01485         if (*iCol > MAX_COLOR_INDEX) tktrnx_.iTxtCol= MAX_COLOR_INDEX;
01486
01487         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01488         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01489         xJournalEntry->action=  XACTION_TXTCOL;
01490         xJournalEntry->i1= tktrnx_.iTxtCol;
01491         xJournalEntry->i2= 0;
01492         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
       next)
01493     }
```

```
01494 }
01495
01496
01497 extern "C" {
01498     void DEFAULTCOLOUR (void)
01499     {
01500       LINCOL (&TCSDefaultLinCol);
01501       TXTCOL (&TCSDefaultTxtCol);
01502       BCKCOL (&TCSDefaultBckCol);
01503     }
01504 }
01505
01506
01507
01508 /*
01509 ------------------- TCS API: Graphic text output ----------------------------------
01510 */
01511
01512
01513
01514 extern "C" {
01515     void outgtext_ (char strng[] )
01516     {
01517       int i, iL;
01518       struct xJournalEntry_typ    * xJournalEntry;
01519
01520         iL= strlen(strng);
01521         tktrnx_.kbeamx+= iL*tktrnx_.khorsz;
01522
01523         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01524         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01525         xJournalEntry->action=  XACTION_GTEXT;
01526         xJournalEntry->i1= iL;
01527         xJournalEntry->i2= strng[0];
01528         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01529
01530         i= 1;
01531         while (i < iL) {
01532           xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01533           if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01534           xJournalEntry->action=  XACTION_ASCII;
01535           xJournalEntry->i1= strng [i++];
01536           if ( i<iL ) {
01537             xJournalEntry->i2= strng[i++];
01538           } else {
01539             xJournalEntry->i2= 0;
01540           }
01541           SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01542         }
01543
01544         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01545         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01546         xJournalEntry->action=  XACTION_MOVABS;
01547         xJournalEntry->i1= tktrnx_.kbeamx;
01548         xJournalEntry->i2= tktrnx_.kbeamy;
01549         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01550     }
01551 }
01552
01553
01554
01555 extern "C" {
01556     void ITALIC (void)
01557     {
01558       struct xJournalEntry_typ    * xJournalEntry;
01559
01560         tktrnx_.kitalc = 1;
01561
01562         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01563         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01564         xJournalEntry->action=  XACTION_FONTATTR;
01565         xJournalEntry->i1= tktrnx_.kitalc;
01566         xJournalEntry->i2= tktrnx_.ksizef;
01567         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01568     }
01569 }
01570
01571
01572
01573 extern "C" {
01574     void ITALIR (void)
01575     {
01576       struct xJournalEntry_typ    * xJournalEntry;
```

```
01577
01578         tktrnx_.kitalc = 0;
01579
01580         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01581         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01582         xJournalEntry->action=  XACTION_FONTATTR;
01583         xJournalEntry->i1= tktrnx_.kitalc;
01584         xJournalEntry->i2= tktrnx_.ksizef;
01585         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01586     }
01587 }
01588
01589
01590
01591 extern "C" {
01592     void DBLSIZ (void)
01593     {
01594       struct xJournalEntry_typ    * xJournalEntry;
01595
01596         if (tktrnx_.ksizef == 0) {
01597           tktrnx_.khorsz = tktrnx_.khorsz * 2;
01598           tktrnx_.kversz = tktrnx_.kversz * 2;
01599           tktrnx_.khomey= TEK_YMAX - tktrnx_.kversz;
01600         }
01601         tktrnx_.ksizef = 1;
01602
01603         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01604         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01605         xJournalEntry->action=  XACTION_FONTATTR;
01606         xJournalEntry->i1= tktrnx_.kitalc;
01607         xJournalEntry->i2= tktrnx_.ksizef;
01608         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01609     }
01610 }
01611
01612
01613
01614 extern "C" {
01615     void NRMSIZ (void)
01616     {
01617       struct xJournalEntry_typ    * xJournalEntry;
01618
01619
01620         if (tktrnx_.ksizef == 1) {
01621           tktrnx_.khorsz = tktrnx_.khorsz / 2;
01622           tktrnx_.kversz = tktrnx_.kversz / 2;
01623           tktrnx_.khomey= TEK_YMAX - tktrnx_.kversz;
01624         }
01625         tktrnx_.ksizef = 0;
01626
01627         xJournalEntry= (xJournalEntry_typ *) malloc (sizeof (xJournalEntry_typ));
01628         if (xJournalEntry == NULL) TCSGraphicError (WRN_JOUADD,"");
01629         xJournalEntry->action=  XACTION_FONTATTR;
01630         xJournalEntry->i1= tktrnx_.kitalc;
01631         xJournalEntry->i2= tktrnx_.ksizef;
01632         SGLIB_DL_LIST_ADD (xJournalEntry_typ, ActiveCanvas->xTCSJournal, xJournalEntry, previous,
      next)
01633     }
01634 }
01635
01636
01637
01638 /*
01639 ------------------- TCS API: Messages ---------------------------------
01640 */
01641
01642
01643
01644 extern "C" {
01645     void BELL (void)
01646     {
01647         wxBell();
01648     }
01649 }
01650
01651
01652
01653 extern "C" {
01654     void outtext_ (char strng[] )
01655     {
01656       if (ActiveCanvas != nullptr) {
01657         if (ActiveCanvas->TCSstatusBar != nullptr) {
01658           ActiveCanvas->TCSstatusBar->SetStatusText(strng);
01659         }
01660       }
```

```
01661     }
01662 }
01663
01664
01665
01666 extern "C" {
01667     void TCSGraphicError (int iErr, const char* msg)
01668     {
01669       char cBuf[TCS_MESSAGELEN];
01670       int i; // Dummyparameter
01671
01672         snprintf( cBuf, TCS_MESSAGELEN, szTCSErrorMsg[iErr], msg );
01673         if (ActiveCanvas == nullptr) { // TCS not initialized
01674           if  (TCSErrorLev[iErr] > 0) wxLogStatus (cBuf);
01675           return;
01676         } else {
01677             if ((ActiveCanvas->TCSstatusBar == nullptr) && (TCSErrorLev[iErr] > 0))  {
01678               wxLogStatus (cBuf); // no own space for logging
01679             } else {
01680               if (TCSErrorLev[iErr] > 0) {
01681                 wxBell ();
01682                 ActiveCanvas->TCSstatusBar->SetStatusText(cBuf);
01683                 if (TCSErrorLev[iErr] < 5) return;
01684                 if ((TCSErrorLev[iErr] == 5) || (TCSErrorLev[iErr] == 10)) {
01685                   tinput_ (&i); // Press Any Key
01686                   ActiveCanvas->TCSstatusBar->SetStatusText("");
01687                 } else if ((TCSErrorLev[iErr]==8) || (TCSErrorLev[iErr]==12)) {
01688                   wxMessageBox (cBuf, szTCSstatWindowName, wxOK||wxICON_ERROR,
      ActiveCanvas->TCSpanel,wxDefaultCoord);
01689                 }
01690                 if (TCSErrorLev[iErr] < 10) return;
01691                 if (iErr != ERR_EXIT) { // Error-Level of finitt() can be changed by XML-Initfile
01692                   finitt_ (&i,&i);        // Forced exit for all Levels >= 10 over finitt()
01693                 }
01694               }
01695             }
01696         }
01697     }
01698 }
01699
01700
01701
01702 /*
01703  ------------------- TCS API: User Input --------------------------------
01704 */
01705
01706
01707
01708  extern "C" {
01709     void DCURSR (int *ic,int* ix,int* iy)
01710     {
01711         ActiveCanvas->TCSmouseButtonDown= 0; // don't use old mouseclicks
01712         ActiveCanvas->TCSpanelKeyPressed= 0; // or old keystrokes
01713         ActiveCanvas->TCSpanel->Refresh();   // wxEVT_PAINT will be executed after wxYield()
01714         ActiveCanvas->TCSpanel->SetFocus();
01715         do {
01716           wxYield(); // process event loop -> be aware of recursive loops!
01717           wxMilliSleep(100); // wait for MOUSE_BUTTON_DOWN event
01718         } while ((ActiveCanvas->TCSmouseButtonDown == 0) && (ActiveCanvas->TCSpanelKeyPressed == 0));
01719         *ic= ActiveCanvas->TCSmouseButtonDown;
01720         if (*ic == 0) {
01721           *ic= ActiveCanvas->TCSpanelKeyPressed;
01722         }
01723         *ix= ActiveCanvas->TCSmouseX;
01724         *iy= ActiveCanvas->TCSmouseY;
01725     }
01726  }
01727
01728
01729
01730  extern "C" {
01731     void TINPUT (int *ic)
01732     {
01733         ActiveCanvas->TCSpanelKeyPressed= 0; // don't use old keystrokes
01734         ActiveCanvas->TCSpanel->Refresh();   // wxEVT_PAINT will be executed after wxYield()
01735         ActiveCanvas->TCSpanel->SetFocus();
01736         do {
01737           wxYield(); // process event loop -> be aware of recursive loops!
01738           wxMilliSleep(100); // wait for KEY_DOWN event
01739         } while (ActiveCanvas->TCSpanelKeyPressed == 0);
01740         *ic= ActiveCanvas->TCSpanelKeyPressed;
01741     }
01742  }
01743
01744
01745
01746 /*
```

```
01747 ------------------- TCS API: Hardcopy -----------------------------------
01748 */
01749
01750
01751
01752 extern "C" {
01753    void HDCOPY (void)
01754    {
01755       wxString FilNam, TmpString;
01756       wxFile HDCfile;
01757       struct xJournalEntry_typ *xJournalEntry;
01758
01759         do {
01760           FilNam.Printf(szTCSHardcopyFile,iHardcopyCount++);
01761         } while ((iHardcopyCount < MAX_HDCCOUNT) && (wxFileExists(FilNam)) );
01762         if (iHardcopyCount >= MAX_HDCCOUNT) {
01763           TCSGraphicError (WRN_HDCFILOPN, "???"); // no unused filename
01764         }
01765         TCSGraphicError (MSG_HDCACT, FilNam.c_str());
01766
01767         if (FilNam.Lower().EndsWith(".hdc")) { // ------ *.hdc -----> Journal File
01768           if (!HDCfile.Open (FilNam, wxFile::write, wxS_DEFAULT) ) {
01769             TCSGraphicError (WRN_HDCFILOPN, FilNam.c_str()); // error during open
01770           };
01771
01772           SGLIB_DL_LIST_GET_LAST(xJournalEntry_typ, ActiveCanvas->xTCSJournal, previous, next,
      xJournalEntry)
01773           while (xJournalEntry != NULL) {
01774             TmpString.Printf("%02i#%04i-%03i\n", xJournalEntry->action, xJournalEntry->i1,
      xJournalEntry->i2);
01775             if (!HDCfile.Write (TmpString) ) {
01776               TCSGraphicError (WRN_HDCFILWRT, FilNam.c_str());
01777             }
01778             xJournalEntry= xJournalEntry -> previous;
01779           }
01780           HDCfile.Close();
01781
01782
01783         } else if (false) { // ------ *.svg -----> Vector Hardcopy
01784           wxSVGFileDC dc(FilNam, TEK_XMAX, TEK_YMAX);
01785           dc.SetAxisOrientation (true, true);      // y-axis bottom->up
01786           dc.SetDeviceOrigin (0., -TEK_YMAX);      // (0,0) lower left corner
01787           RepaintBuffer (dc); // Bug in wx V3.1.5: Text will plotted upside down !!!
01788
01789 //      } else if (false) { // ------ *.wmf -----> Windows Metafile
01790 //        wxMetafileDC dc(FilNam, TEK_XMAX, TEK_YMAX);
01791 //        dc.SetAxisOrientation (true, true);      // y-axis bottom->up
01792 //        dc.SetDeviceOrigin (0., -TEK_YMAX);  // (0,0) lower left corner
01793 //           dc.SetBrush (*wxWHITE_BRUSH); // Testplot works
01794 //           dc.Clear();
01795 //           dc.SetPen (*wxBLACK_PEN);
01796 //           dc.DrawRectangle (10,10,40,40);
01797 //        RepaintBuffer (dc); // Doesn't work: textmeasure.cpp must not be used with non-native wxDCs
01798 //        dc.Close();
01799
01800         } else if (FilNam.Lower().EndsWith(".bmp") ||
01801                    FilNam.Lower().EndsWith(".jpg") ) { // ------ *.??? -----> Bitmaps
01802           wxBitmap *PixelMap= new wxBitmap (TEK_XMAX, TEK_YMAX, wxBITMAP_SCREEN_DEPTH);
01803           wxMemoryDC dc;
01804           dc.SelectObject (*PixelMap);
01805
01806           dc.SetAxisOrientation (true, true); // y-axis bottom->up
01807           dc.SetDeviceOrigin (0., TEK_YMAX);  // Origin moved in unmodified axis orientation!
01808           RepaintBuffer (dc);
01809           dc.SelectObject (wxNullBitmap); // unlock bitmap
01810
01811           if (FilNam.Lower().EndsWith(".bmp")) {
01812             PixelMap->SaveFile (FilNam, wxBITMAP_TYPE_BMP, (wxPalette*)NULL);
01813           } else if (FilNam.Lower().EndsWith(".jpg")) {
01814             if (wxImage::FindHandler(wxBITMAP_TYPE_JPEG) == nullptr) {
01815               wxImage::AddHandler(new wxJPEGHandler);
01816             }
01817             PixelMap->SaveFile (FilNam, wxBITMAP_TYPE_JPEG , (wxPalette*)NULL);
01818           }
01819           delete PixelMap;
01820
01821         } // Last format of hardcopies
01822     } // End of subroutine
01823 } // End of extern "C"
01824
01825
01826
01827 extern "C" {
01828    void SVSTAT (char dst[])
01829    {
01830      size_t numbytes;
01831        numbytes= sizeof (struct TKTRNX);
```

```
01832         memmove (dst, &tktrnx_.khomey, numbytes);
01833     }
01834 }
01835
01836
01837
01838 extern "C" {
01839     void RESTAT (char src[])
01840     {
01841       size_t numbytes;
01842         numbytes= sizeof (struct TKTRNX);
01843         memmove (&tktrnx_.khomey, src, numbytes);
01844         movabs_ (&tktrnx_.kbeamx, &tktrnx_.kbeamy);
01845     }
01846 }
01847
01848
01849
01850 /*
01851 ------------------- subroutine LIB_MOVC3 -----------------------------
01852       Subroutine is not used here, for downward compatibility only
01853 */
01854
01855 extern "C" {
01856     void lib_movc3_ (int *len,char sou[],char dst[])
01857     {
01858         memmove (dst, sou, (size_t) *len);
01859     }
01860 }
```

9.32 TCSdrWXcpp.hpp File Reference

WX Port: Headerfile.

Macros

- #define TEK_XMAX 1023.0
- #define TEK_YMAX 780.0
- #define TCS_LINEWIDTH 1
- #define MAX_OPEN_CANVAS 20
- #define STAT_MAXROWS 1
- #define TCS_REL_CHR_HEIGHT 0.018f
- #define TCS_REL_CHR_SPACING 0.7f
- #define TCS_WINDOW_NAMELEN 50
- #define TCS_FILE_NAMELEN 132
- #define TCS_MESSAGELEN 132
- #define MAX_HDCCOUNT 1000
- #define TCS_INIFILE_NAME ""
- #define INIFILEXT ".XML"
- #define INIFILEXTTOKEN ".%"
- #define PROGDIRTOKEN "%:"
- #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 XACTION_CLIP 15
- #define XACTION_CLIP1 16
- #define XACTION_CLIP2 17
- #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.HDC"
- #define TCS_INISECT2 "Layout"
- #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_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_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 "G2dXMLopenL"
- #define TCS_INIDEF_XMLOPENL 0
- #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 "G2dInitt"
- #define TCS_INIDEF_INI2 "Error creating windows in subroutine INITT"
- #define TCS_INIVAR_INI2L "G2dInittL"
- #define TCS_INIDEF_INI2L 1

9.32.1 Detailed Description

WX Port: Headerfile.

Version

1.0

Author

Dr.-Ing. Klaus Friedewald

Headerfile for TCSdrWXcpp.cpp

Note

- Configuration of the library
- Defining default values

Definition in file TCSdrWXcpp.hpp.

9.32.2 Macro Definition Documentation

9.32.2.1 ERR_EXIT

```
#define ERR_EXIT 12
```

Definition at line 87 of file TCSdrWXcpp.hpp.

9.32.2.2 ERR_NOFNT

```
#define ERR_NOFNT 4
```
Definition at line 79 of file TCSdrWXcpp.hpp.

9.32.2.3 ERR_NOFNTFIL

```
#define ERR_NOFNTFIL 3
```
Definition at line 78 of file TCSdrWXcpp.hpp.

9.32.2.4 ERR_UNKNAUDIO

```
#define ERR_UNKNAUDIO 22
```
Definition at line 97 of file TCSdrWXcpp.hpp.

9.32.2.5 ERR_UNKNGRAPHCARD

```
#define ERR_UNKNGRAPHCARD 2
```
Definition at line 77 of file TCSdrWXcpp.hpp.

9.32.2.6 ERR_XMLOPEN

```
#define ERR_XMLOPEN 21
```
Definition at line 96 of file TCSdrWXcpp.hpp.

9.32.2.7 ERR_XMLPARSER

```
#define ERR_XMLPARSER 20
```
Definition at line 95 of file TCSdrWXcpp.hpp.

9.32.2.8 INIFILEXT

```
#define INIFILEXT ".XML"
```
Definition at line 46 of file TCSdrWXcpp.hpp.

9.32.2.9 INIFILEXTTOKEN

```
#define INIFILEXTTOKEN ".%"
```
Definition at line 47 of file TCSdrWXcpp.hpp.

9.32.2.10 MAX_HDCCOUNT

```
#define MAX_HDCCOUNT 1000
```
Definition at line 43 of file TCSdrWXcpp.hpp.

9.32.2.11 MAX_OPEN_CANVAS

```
#define MAX_OPEN_CANVAS 20
```
Definition at line 32 of file TCSdrWXcpp.hpp.

9.32.2.12 MSG_HDCACT

```
#define MSG_HDCACT 10
```
Definition at line 85 of file TCSdrWXcpp.hpp.

9.32.2.13 MSG_MAXERRNO

```
#define MSG_MAXERRNO 25
```
Definition at line 100 of file TCSdrWXcpp.hpp.

9.32.2.14 MSG_NOMOUSE

```
#define MSG_NOMOUSE 5
```
Definition at line 80 of file TCSdrWXcpp.hpp.

9.32.2.15 MSG_USR

```
#define MSG_USR 9
```
Definition at line 84 of file TCSdrWXcpp.hpp.

9.32.2.16 MSG_USR2

```
#define MSG_USR2 23
```
Definition at line 98 of file TCSdrWXcpp.hpp.

9.32.2.17 PROGDIRTOKEN

```
#define PROGDIRTOKEN "%:"
```
Definition at line 48 of file TCSdrWXcpp.hpp.

9.32.2.18 STAT_MAXROWS

```
#define STAT_MAXROWS 1
```
Definition at line 34 of file TCSdrWXcpp.hpp.

9.32.2.19 TCS_FILE_NAMELEN

```
#define TCS_FILE_NAMELEN 132
```
Definition at line 40 of file TCSdrWXcpp.hpp.

9.32.2.20 TCS_HDCFILE_NAME

```
#define TCS_HDCFILE_NAME "HDC%03i.HDC"
```
Definition at line 114 of file TCSdrWXcpp.hpp.

9.32.2.21 TCS_INIDEF_BCKCOL

```
#define TCS_INIDEF_BCKCOL 0
```
Definition at line 148 of file TCSdrWXcpp.hpp.

9.32.2.22 TCS_INIDEF_COPLCK

```
#define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager:  ClipBoard locked."
```
Definition at line 192 of file TCSdrWXcpp.hpp.

9.32.2.23 TCS_INIDEF_COPLCKL

```
#define TCS_INIDEF_COPLCKL 1
```
Definition at line 194 of file TCSdrWXcpp.hpp.

9.32.2.24 TCS_INIDEF_COPMEM

```
#define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager:  Out of Memory."
```
Definition at line 188 of file TCSdrWXcpp.hpp.

9.32.2.25 TCS_INIDEF_COPMEML

```
#define TCS_INIDEF_COPMEML 1
```
Definition at line 190 of file TCSdrWXcpp.hpp.

9.32.2.26 TCS_INIDEF_EXIT

```
#define TCS_INIDEF_EXIT "Press any key to exit program."
```
Definition at line 184 of file TCSdrWXcpp.hpp.

9.32.2.27 TCS_INIDEF_EXITL

```
#define TCS_INIDEF_EXITL 10
```
Definition at line 186 of file TCSdrWXcpp.hpp.

9.32.2.28 TCS_INIDEF_HDCACT

```
#define TCS_INIDEF_HDCACT "Hardcopy in progress:  File %s created."
```
Definition at line 176 of file TCSdrWXcpp.hpp.

9.32.2.29 TCS_INIDEF_HDCACTL

```
#define TCS_INIDEF_HDCACTL 1
```
Definition at line 178 of file TCSdrWXcpp.hpp.

9.32.2.30 TCS_INIDEF_HDCOPN

```
#define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
```
Definition at line 164 of file TCSdrWXcpp.hpp.

9.32.2.31 TCS_INIDEF_HDCOPNL

```
#define TCS_INIDEF_HDCOPNL 5
```
Definition at line 166 of file TCSdrWXcpp.hpp.

9.32.2.32 TCS_INIDEF_HDCWRT

```
#define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
```
Definition at line 168 of file TCSdrWXcpp.hpp.

9.32.2.33 TCS_INIDEF_HDCWRTL

```
#define TCS_INIDEF_HDCWRTL 5
```
Definition at line 170 of file TCSdrWXcpp.hpp.

9.32.2.34 TCS_INIDEF_INI2

```
#define TCS_INIDEF_INI2 "Error creating windows in subroutine INITT"
```
Definition at line 232 of file TCSdrWXcpp.hpp.

9.32.2.35 TCS_INIDEF_INI2L

```
#define TCS_INIDEF_INI2L 1
```
Definition at line 234 of file TCSdrWXcpp.hpp.

9.32.2.36 TCS_INIDEF_JOUADD

```
#define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
```
Definition at line 204 of file TCSdrWXcpp.hpp.

9.32.2.37 TCS_INIDEF_JOUADDL

```
#define TCS_INIDEF_JOUADDL 5
```
Definition at line 206 of file TCSdrWXcpp.hpp.

9.32.2.38 TCS_INIDEF_JOUCLR

```
#define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
```
Definition at line 208 of file TCSdrWXcpp.hpp.

9.32.2.39 TCS_INIDEF_JOUCLRL

```
#define TCS_INIDEF_JOUCLRL 5
```
Definition at line 210 of file TCSdrWXcpp.hpp.

9.32.2.40 TCS_INIDEF_JOUCREATE

```
#define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal.  Error-No:  %s."
```
Definition at line 196 of file TCSdrWXcpp.hpp.

9.32.2.41 TCS_INIDEF_JOUCREATEL

```
#define TCS_INIDEF_JOUCREATEL 5
```
Definition at line 198 of file TCSdrWXcpp.hpp.

9.32.2.42 TCS_INIDEF_JOUENTRY

```
#define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
```
Definition at line 200 of file TCSdrWXcpp.hpp.

9.32.2.43 TCS_INIDEF_JOUENTRYL

```
#define TCS_INIDEF_JOUENTRYL 5
```
Definition at line 202 of file TCSdrWXcpp.hpp.

9.32.2.44 TCS_INIDEF_JOUUNKWN

```
#define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
```
Definition at line 212 of file TCSdrWXcpp.hpp.

9.32.2.45 TCS_INIDEF_JOUUNKWNL

```
#define TCS_INIDEF_JOUUNKWNL 5
```
Definition at line 214 of file TCSdrWXcpp.hpp.

9.32.2.46 TCS_INIDEF_LINCOL

```
#define TCS_INIDEF_LINCOL 1
```
Definition at line 144 of file TCSdrWXcpp.hpp.

9.32.2.47 TCS_INIDEF_NOFNT

```
#define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
```
Definition at line 160 of file TCSdrWXcpp.hpp.

9.32.2.48 TCS_INIDEF_NOFNTFIL

```
#define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
```
Definition at line 156 of file TCSdrWXcpp.hpp.

9.32.2.49 TCS_INIDEF_NOFNTFILL

```
#define TCS_INIDEF_NOFNTFILL 10
```
Definition at line 158 of file TCSdrWXcpp.hpp.

9.32.2.50 TCS_INIDEF_NOFNTL

```
#define TCS_INIDEF_NOFNTL 10
```
Definition at line 162 of file TCSdrWXcpp.hpp.

9.32.2.51 TCS_INIDEF_TXTCOL

```
#define TCS_INIDEF_TXTCOL 1
```
Definition at line 146 of file TCSdrWXcpp.hpp.

9.32.2.52 TCS_INIDEF_UNKNAUDIO

```
#define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System:  Error %s."
```
Definition at line 224 of file TCSdrWXcpp.hpp.

9.32.2.53 TCS_INIDEF_UNKNAUDIOL

```
#define TCS_INIDEF_UNKNAUDIOL 5
```
Definition at line 226 of file TCSdrWXcpp.hpp.

9.32.2.54 TCS_INIDEF_UNKNGRAPHCARD

```
#define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System:  Error %s."
```
Definition at line 152 of file TCSdrWXcpp.hpp.

9.32.2.55 TCS_INIDEF_UNKNGRAPHCARDL

```
#define TCS_INIDEF_UNKNGRAPHCARDL 10
```
Definition at line 154 of file TCSdrWXcpp.hpp.

9.32.2.56 TCS_INIDEF_USR

```
#define TCS_INIDEF_USR "%s"
```
Definition at line 172 of file TCSdrWXcpp.hpp.

9.32.2.57 TCS_INIDEF_USR2

```
#define TCS_INIDEF_USR2 "%s"
```
Definition at line 228 of file TCSdrWXcpp.hpp.

9.32.2.58 TCS_INIDEF_USR2L

```
#define TCS_INIDEF_USR2L 5
```
Definition at line 230 of file TCSdrWXcpp.hpp.

9.32.2.59 TCS_INIDEF_USRL

```
#define TCS_INIDEF_USRL 5
```
Definition at line 174 of file TCSdrWXcpp.hpp.

9.32.2.60 TCS_INIDEF_USRWRN

```
#define TCS_INIDEF_USRWRN "Press any key to continue."
```
Definition at line 180 of file TCSdrWXcpp.hpp.

9.32.2.61 TCS_INIDEF_USRWRNL

```
#define TCS_INIDEF_USRWRNL 5
```
Definition at line 182 of file TCSdrWXcpp.hpp.

9.32.2.62 TCS_INIDEF_WINPOSX

```
#define TCS_INIDEF_WINPOSX 1
```
Definition at line 127 of file TCSdrWXcpp.hpp.

9.32.2.63 TCS_INIDEF_WINPOSY

```
#define TCS_INIDEF_WINPOSY 3
```
Definition at line 129 of file TCSdrWXcpp.hpp.

9.32.2.64 TCS_INIDEF_WINSIZX

```
#define TCS_INIDEF_WINSIZX 98
```
Definition at line 131 of file TCSdrWXcpp.hpp.

9.32.2.65 TCS_INIDEF_WINSIZY

```
#define TCS_INIDEF_WINSIZY 85
```
Definition at line 133 of file TCSdrWXcpp.hpp.

9.32.2.66 TCS_INIDEF_XMLOPEN

```
#define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
```
Definition at line 220 of file TCSdrWXcpp.hpp.

9.32.2.67 TCS_INIDEF_XMLOPENL

```
#define TCS_INIDEF_XMLOPENL 0
```
Definition at line 222 of file TCSdrWXcpp.hpp.

9.32.2.68 TCS_INIDEF_XMLPARSER

```
#define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File:  %s"
```
Definition at line 216 of file TCSdrWXcpp.hpp.

9.32.2.69 TCS_INIDEF_XMLPARSERL

```
#define TCS_INIDEF_XMLPARSERL 8
```
Definition at line 218 of file TCSdrWXcpp.hpp.

9.32.2.70 TCS_INIFILE_NAME

```
#define TCS_INIFILE_NAME ""
```
Definition at line 45 of file TCSdrWXcpp.hpp.

9.32.2.71 TCS_INISECT0

```
#define TCS_INISECT0 "Graph2D"
```
Definition at line 106 of file TCSdrWXcpp.hpp.

9.32.2.72 TCS_INISECT1

```
#define TCS_INISECT1 "Names"
```
Definition at line 108 of file TCSdrWXcpp.hpp.

9.32.2.73 TCS_INISECT2

```
#define TCS_INISECT2 "Layout"
```
Definition at line 118 of file TCSdrWXcpp.hpp.

9.32.2.74 TCS_INISECT3

```
#define TCS_INISECT3 "Messages"
```
Definition at line 150 of file TCSdrWXcpp.hpp.

9.32.2.75 TCS_INIVAR_BCKCOL

```
#define TCS_INIVAR_BCKCOL "G2dBckCol"
```
Definition at line 147 of file TCSdrWXcpp.hpp.

9.32.2.76 TCS_INIVAR_COPLCK

```
#define TCS_INIVAR_COPLCK "G2dClipLock"
```
Definition at line 191 of file TCSdrWXcpp.hpp.

9.32.2.77 TCS_INIVAR_COPLCKL

```
#define TCS_INIVAR_COPLCKL "G2dClipLockL"
```
Definition at line 193 of file TCSdrWXcpp.hpp.

9.32.2.78 TCS_INIVAR_COPMEM

```
#define TCS_INIVAR_COPMEM "G2dNoMemory"
```
Definition at line 187 of file TCSdrWXcpp.hpp.

9.32.2.79 TCS_INIVAR_COPMEML

```
#define TCS_INIVAR_COPMEML "G2dNoMemoryL"
```
Definition at line 189 of file TCSdrWXcpp.hpp.

9.32.2.80 TCS_INIVAR_EXIT

```
#define TCS_INIVAR_EXIT "G2dExit"
```
Definition at line 183 of file TCSdrWXcpp.hpp.

9.32.2.81 TCS_INIVAR_EXITL

```
#define TCS_INIVAR_EXITL "G2dExitL"
```
Definition at line 185 of file TCSdrWXcpp.hpp.

9.32.2.82 TCS_INIVAR_HDCACT

```
#define TCS_INIVAR_HDCACT "G2dHdcActive"
```
Definition at line 175 of file TCSdrWXcpp.hpp.

9.32.2.83 TCS_INIVAR_HDCACTL

```
#define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
```
Definition at line 177 of file TCSdrWXcpp.hpp.

9.32.2.84 TCS_INIVAR_HDCNAM

```
#define TCS_INIVAR_HDCNAM "G2dHardcopy"
```
Definition at line 113 of file TCSdrWXcpp.hpp.

9.32.2.85 TCS_INIVAR_HDCOPN

```
#define TCS_INIVAR_HDCOPN "G2dHdcOpen"
```
Definition at line 163 of file TCSdrWXcpp.hpp.

9.32.2.86 TCS_INIVAR_HDCOPNL

```
#define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
```
Definition at line 165 of file TCSdrWXcpp.hpp.

9.32.2.87 TCS_INIVAR_HDCWRT

```
#define TCS_INIVAR_HDCWRT "G2dHdcWrite"
```
Definition at line 167 of file TCSdrWXcpp.hpp.

9.32.2.88 TCS_INIVAR_HDCWRTL

```
#define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
```
Definition at line 169 of file TCSdrWXcpp.hpp.

9.32.2.89 TCS_INIVAR_INI2

```
#define TCS_INIVAR_INI2 "G2dInitt"
```
Definition at line 231 of file TCSdrWXcpp.hpp.

9.32.2.90 TCS_INIVAR_INI2L

```
#define TCS_INIVAR_INI2L "G2dInittL"
```
Definition at line 233 of file TCSdrWXcpp.hpp.

9.32.2.91 TCS_INIVAR_JOUADD

```
#define TCS_INIVAR_JOUADD "G2dJouAdd"
```
Definition at line 203 of file TCSdrWXcpp.hpp.

9.32.2.92 TCS_INIVAR_JOUADDL

```
#define TCS_INIVAR_JOUADDL "G2dJouAddL"
```
Definition at line 205 of file TCSdrWXcpp.hpp.

9.32.2.93 TCS_INIVAR_JOUCLR

```
#define TCS_INIVAR_JOUCLR "G2dJouClr"
```
Definition at line 207 of file TCSdrWXcpp.hpp.

9.32.2.94 TCS_INIVAR_JOUCLRL

```
#define TCS_INIVAR_JOUCLRL "G2dJouClrL"
```
Definition at line 209 of file TCSdrWXcpp.hpp.

9.32.2.95 TCS_INIVAR_JOUCREATE

```
#define TCS_INIVAR_JOUCREATE "G2dJouCreate"
```
Definition at line 195 of file TCSdrWXcpp.hpp.

9.32.2.96 TCS_INIVAR_JOUCREATEL

```
#define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
```
Definition at line 197 of file TCSdrWXcpp.hpp.

9.32.2.97 TCS_INIVAR_JOUENTRY

```
#define TCS_INIVAR_JOUENTRY "G2dJouEntry"
```
Definition at line 199 of file TCSdrWXcpp.hpp.

9.32.2.98 TCS_INIVAR_JOUENTRYL

```
#define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
```
Definition at line 201 of file TCSdrWXcpp.hpp.

9.32.2.99 TCS_INIVAR_JOUUNKWN

```
#define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
```
Definition at line 211 of file TCSdrWXcpp.hpp.

9.32.2.100 TCS_INIVAR_JOUUNKWNL

```
#define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
```
Definition at line 213 of file TCSdrWXcpp.hpp.

9.32.2.101 TCS_INIVAR_LINCOL

```
#define TCS_INIVAR_LINCOL "G2dLinCol"
```
Definition at line 143 of file TCSdrWXcpp.hpp.

9.32.2.102 TCS_INIVAR_NOFNT

```
#define TCS_INIVAR_NOFNT "G2dFntfilOpen"
```
Definition at line 159 of file TCSdrWXcpp.hpp.

9.32.2.103 TCS_INIVAR_NOFNTFIL

```
#define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
```
Definition at line 155 of file TCSdrWXcpp.hpp.

9.32.2.104 TCS_INIVAR_NOFNTFILL

```
#define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
```
Definition at line 157 of file TCSdrWXcpp.hpp.

9.32.2.105 TCS_INIVAR_NOFNTL

```
#define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
```
Definition at line 161 of file TCSdrWXcpp.hpp.

9.32.2.106 TCS_INIVAR_STATNAM

```
#define TCS_INIVAR_STATNAM "G2dStatus"
```
Definition at line 111 of file TCSdrWXcpp.hpp.

9.32.2.107 TCS_INIVAR_TXTCOL

```
#define TCS_INIVAR_TXTCOL "G2dTxtCol"
```
Definition at line 145 of file TCSdrWXcpp.hpp.

9.32.2.108 TCS_INIVAR_UNKNAUDIO

```
#define TCS_INIVAR_UNKNAUDIO "G2dAudio"
```
Definition at line 223 of file TCSdrWXcpp.hpp.

9.32.2.109 TCS_INIVAR_UNKNAUDIOL

```
#define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
```
Definition at line 225 of file TCSdrWXcpp.hpp.

9.32.2.110 TCS_INIVAR_UNKNGRAPHCARD

```
#define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
```
Definition at line 151 of file TCSdrWXcpp.hpp.

9.32.2.111 TCS_INIVAR_UNKNGRAPHCARDL

```
#define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
```
Definition at line 153 of file TCSdrWXcpp.hpp.

9.32.2.112 TCS_INIVAR_USR

```
#define TCS_INIVAR_USR "G2dUser"
```
Definition at line 171 of file TCSdrWXcpp.hpp.

9.32.2.113 TCS_INIVAR_USR2

```
#define TCS_INIVAR_USR2 "G2dUser2"
```
Definition at line 227 of file TCSdrWXcpp.hpp.

9.32.2.114 TCS_INIVAR_USR2L

```
#define TCS_INIVAR_USR2L "G2dUser2L"
```
Definition at line 229 of file TCSdrWXcpp.hpp.

9.32.2.115 TCS_INIVAR_USRL

```
#define TCS_INIVAR_USRL "G2dUserL"
```
Definition at line 173 of file TCSdrWXcpp.hpp.

9.32.2.116 TCS_INIVAR_USRWRN

```
#define TCS_INIVAR_USRWRN "G2dPressAny"
```
Definition at line 179 of file TCSdrWXcpp.hpp.

9.32.2.117 TCS_INIVAR_USRWRNL

```
#define TCS_INIVAR_USRWRNL "G2dPressAnyL"
```
Definition at line 181 of file TCSdrWXcpp.hpp.

9.32.2.118 TCS_INIVAR_WINNAM

```
#define TCS_INIVAR_WINNAM "G2dGraphic"
```
Definition at line 109 of file TCSdrWXcpp.hpp.

9.32.2.119 TCS_INIVAR_WINPOSX

```
#define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
```
Definition at line 126 of file TCSdrWXcpp.hpp.

9.32.2.120 TCS_INIVAR_WINPOSY

```
#define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
```
Definition at line 128 of file TCSdrWXcpp.hpp.

9.32.2.121 TCS_INIVAR_WINSIZX

```
#define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
```
Definition at line 130 of file TCSdrWXcpp.hpp.

9.32.2.122 TCS_INIVAR_WINSIZY

```
#define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
```
Definition at line 132 of file TCSdrWXcpp.hpp.

9.32.2.123 TCS_INIVAR_XMLOPEN

```
#define TCS_INIVAR_XMLOPEN "G2dXMLopen"
```
Definition at line 219 of file TCSdrWXcpp.hpp.

9.32.2.124 TCS_INIVAR_XMLOPENL

```
#define TCS_INIVAR_XMLOPENL "G2dXMLopenL"
```
Definition at line 221 of file TCSdrWXcpp.hpp.

9.32.2.125 TCS_INIVAR_XMLPARSER

```
#define TCS_INIVAR_XMLPARSER "G2dXMLerror"
```
Definition at line 215 of file TCSdrWXcpp.hpp.

9.32.2.126 TCS_INIVAR_XMLPARSERL

```
#define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
```
Definition at line 217 of file TCSdrWXcpp.hpp.

9.32.2.127 TCS_LINEWIDTH

```
#define TCS_LINEWIDTH 1
```
Definition at line 31 of file TCSdrWXcpp.hpp.

9.32.2.128 TCS_MESSAGELEN

```
#define TCS_MESSAGELEN 132
```
Definition at line 42 of file TCSdrWXcpp.hpp.

9.32.2.129 TCS_REL_CHR_HEIGHT

```
#define TCS_REL_CHR_HEIGHT 0.018f
```
Definition at line 36 of file TCSdrWXcpp.hpp.

9.32.2.130 TCS_REL_CHR_SPACING

```
#define TCS_REL_CHR_SPACING 0.7f
```
Definition at line 37 of file TCSdrWXcpp.hpp.

9.32.2.131 TCS_STATWINDOW_NAME

```
#define TCS_STATWINDOW_NAME "System Messages"
```
Definition at line 112 of file TCSdrWXcpp.hpp.

9.32.2.132 TCS_WINDOW_NAME

```
#define TCS_WINDOW_NAME "Graphics"
```
Definition at line 110 of file TCSdrWXcpp.hpp.

9.32.2.133 TCS_WINDOW_NAMELEN

```
#define TCS_WINDOW_NAMELEN 50
```
Definition at line 39 of file TCSdrWXcpp.hpp.

9.32.2.134 TEK_XMAX

```
#define TEK_XMAX 1023.0
```
Definition at line 24 of file TCSdrWXcpp.hpp.

9.32.2.135 TEK_YMAX

```
#define TEK_YMAX 780.0
```
Definition at line 25 of file TCSdrWXcpp.hpp.

9.32.2.136 WRN_COPYLOCK

```
#define WRN_COPYLOCK 14
```
Definition at line 89 of file TCSdrWXcpp.hpp.

9.32.2.137 WRN_COPYNOMEM

```
#define WRN_COPYNOMEM 13
```
Definition at line 88 of file TCSdrWXcpp.hpp.

9.32.2.138 WRN_HDCFILOPN

```
#define WRN_HDCFILOPN 6
```
Definition at line 81 of file TCSdrWXcpp.hpp.

9.32.2.139 WRN_HDCFILWRT

```
#define WRN_HDCFILWRT 7
```
Definition at line 82 of file TCSdrWXcpp.hpp.

9.32.2.140 WRN_HDCINTERN

```
#define WRN_HDCINTERN 8
```
Definition at line 83 of file TCSdrWXcpp.hpp.

9.32.2.141 WRN_INI2

```
#define WRN_INI2 24
```
Definition at line 99 of file TCSdrWXcpp.hpp.

9.32.2.142 WRN_JOUADD

```
#define WRN_JOUADD 17
```
Definition at line 92 of file TCSdrWXcpp.hpp.

9.32.2.143 WRN_JOUCLR

```
#define WRN_JOUCLR 18
```
Definition at line 93 of file TCSdrWXcpp.hpp.

9.32.2.144 WRN_JOUCREATE

```
#define WRN_JOUCREATE 15
```
Definition at line 90 of file TCSdrWXcpp.hpp.

9.32.2.145 WRN_JOUENTRY

```
#define WRN_JOUENTRY 16
```
Definition at line 91 of file TCSdrWXcpp.hpp.

9.32.2.146 WRN_JOUUNKWN

```
#define WRN_JOUUNKWN 19
```
Definition at line 94 of file TCSdrWXcpp.hpp.

9.32.2.147 WRN_NOMSG

```
#define WRN_NOMSG 1
```
Definition at line 76 of file TCSdrWXcpp.hpp.

9.32.2.148 WRN_USRPRESSANY

```
#define WRN_USRPRESSANY 11
```
Definition at line 86 of file TCSdrWXcpp.hpp.

9.32.2.149 XACTION_ASCII

```
#define XACTION_ASCII 9
```
Definition at line 62 of file TCSdrWXcpp.hpp.

9.32.2.150 XACTION_BCKCOL

```
#define XACTION_BCKCOL 10
```
Definition at line 63 of file TCSdrWXcpp.hpp.

9.32.2.151 XACTION_CLIP

```
#define XACTION_CLIP 15
```
Definition at line 68 of file TCSdrWXcpp.hpp.

9.32.2.152 XACTION_CLIP1

```
#define XACTION_CLIP1 16
```
Definition at line 69 of file TCSdrWXcpp.hpp.

9.32.2.153 XACTION_CLIP2

```
#define XACTION_CLIP2 17
```
Definition at line 70 of file TCSdrWXcpp.hpp.

9.32.2.154 XACTION_DRWABS

```
#define XACTION_DRWABS 4
```
Definition at line 57 of file TCSdrWXcpp.hpp.

9.32.2.155 XACTION_DSHABS

```
#define XACTION_DSHABS 6
```
Definition at line 59 of file TCSdrWXcpp.hpp.

9.32.2.156 XACTION_DSHSTYLE

```
#define XACTION_DSHSTYLE 5
```
Definition at line 58 of file TCSdrWXcpp.hpp.

9.32.2.157 XACTION_ERASE

```
#define XACTION_ERASE 2
```
Definition at line 55 of file TCSdrWXcpp.hpp.

9.32.2.158 XACTION_FONTATTR

```
#define XACTION_FONTATTR 13
```
Definition at line 66 of file TCSdrWXcpp.hpp.

9.32.2.159 XACTION_GTEXT

```
#define XACTION_GTEXT 8
```
Definition at line 61 of file TCSdrWXcpp.hpp.

9.32.2.160 XACTION_INITT

```
#define XACTION_INITT 1
```
Definition at line 54 of file TCSdrWXcpp.hpp.

9.32.2.161 XACTION_LINCOL

```
#define XACTION_LINCOL 11
```
Definition at line 64 of file TCSdrWXcpp.hpp.

9.32.2.162 XACTION_MOVABS

```
#define XACTION_MOVABS 3
```
Definition at line 56 of file TCSdrWXcpp.hpp.

9.32.2.163 XACTION_NOOP

```
#define XACTION_NOOP 14
```
Definition at line 67 of file TCSdrWXcpp.hpp.

9.32.2.164 XACTION_PNTABS

```
#define XACTION_PNTABS 7
```
Definition at line 60 of file TCSdrWXcpp.hpp.

9.32.2.165 XACTION_TXTCOL

```
#define XACTION_TXTCOL 12
```
Definition at line 65 of file TCSdrWXcpp.hpp.

9.33 TCSdrWXcpp.hpp

```
00001 /** ****************************************************************************
00002 \file    TCSdrWXcpp.hpp
00003 \brief   WX Port: Headerfile
00004 \version 1.0
00005 \author  Dr.-Ing. Klaus Friedewald
00006 \~german
00007          Headerfile zu TCSdrWXcpp.cpp
00008 \note
00009           - Konfiguration der Bibliothek
00010           - Definition der Defaultwerte
00011 \~english
00012          Headerfile for TCSdrWXcpp.cpp
00013 \note
00014           - Configuration of the library
00015           - Defining default values
00016 \~
00017
00018 ******************************************************************************* **/
00019
00020
00021
00022 /* -------------- Drawing area in Tektronix coordinates --------------------- */
00023
00024 #define TEK_XMAX 1023.0  // Double precision because of
00025 #define TEK_YMAX 780.0   // use in wx::SetLogicalScale ()
00026
00027
00028
00029 /* -------------- Program parameters ----------------------------------------- */
00030
00031 #define TCS_LINEWIDTH 1
00032 #define MAX_OPEN_CANVAS 20         // Maximum number of used canvases
00033
00034 #define STAT_MAXROWS 1             // Analogue to the other ports, not used here
00035
00036 #define TCS_REL_CHR_HEIGHT 0.018f  // Define size / vertical spacing of graphic text
00037 #define TCS_REL_CHR_SPACING 0.7f
00038
00039 #define TCS_WINDOW_NAMELEN 50
00040 #define TCS_FILE_NAMELEN 132
00041
00042 #define TCS_MESSAGELEN 132
00043 #define MAX_HDCCOUNT 1000          // parameter is bound to TCS_HDCFILE_NAME
00044
00045 #define TCS_INIFILE_NAME ""
00046 #define INIFILEXT ".XML"
00047 #define INIFILEXTTOKEN ".%"        // Token for parsing filenames
00048 #define PROGDIRTOKEN "%:"
00049
00050
```

```
00051
00052 /* Actioncodes of the journalfiles */
00053
00054 #define XACTION_INITT       1
00055 #define XACTION_ERASE       2
00056 #define XACTION_MOVABS      3
00057 #define XACTION_DRWABS      4
00058 #define XACTION_DSHSTYLE    5
00059 #define XACTION_DSHABS      6
00060 #define XACTION_PNTABS      7
00061 #define XACTION_GTEXT       8
00062 #define XACTION_ASCII       9
00063 #define XACTION_BCKCOL      10
00064 #define XACTION_LINCOL      11
00065 #define XACTION_TXTCOL      12
00066 #define XACTION_FONTATTR    13
00067 #define XACTION_NOOP        14
00068 #define XACTION_CLIP        15
00069 #define XACTION_CLIP1       16
00070 #define XACTION_CLIP2       17
00071
00072
00073
00074 /* Assign errornumbers */
00075
00076 #define WRN_NOMSG 1
00077 #define ERR_UNKNGRAPHCARD 2
00078 #define ERR_NOFNTFIL 3
00079 #define ERR_NOFNT 4
00080 #define MSG_NOMOUSE 5
00081 #define WRN_HDCFILOPN 6
00082 #define WRN_HDCFILWRT 7
00083 #define WRN_HDCINTERN 8
00084 #define MSG_USR 9
00085 #define MSG_HDCACT 10
00086 #define WRN_USRPRESSANY 11
00087 #define ERR_EXIT 12
00088 #define WRN_COPYNOMEM 13
00089 #define WRN_COPYLOCK 14
00090 #define WRN_JOUCREATE 15
00091 #define WRN_JOUENTRY 16
00092 #define WRN_JOUADD 17
00093 #define WRN_JOUCLR 18
00094 #define WRN_JOUUNKWN 19
00095 #define ERR_XMLPARSER 20
00096 #define ERR_XMLOPEN 21
00097 #define ERR_UNKNAUDIO 22
00098 #define MSG_USR2 23
00099 #define WRN_INI2 24
00100 #define MSG_MAXERRNO 25
00101
00102
00103
00104 /* Default initialization, can be changed by the ini-XML file */
00105
00106 #define TCS_INISECT0 "Graph2D" // Root-Section for XML, change with WINLBL()
00107
00108 #define TCS_INISECT1 "Names"
00109  #define TCS_INIVAR_WINNAM "G2dGraphic"
00110     #define TCS_WINDOW_NAME "Graphics"
00111  #define TCS_INIVAR_STATNAM "G2dStatus"
00112     #define TCS_STATWINDOW_NAME "System Messages"
00113  #define TCS_INIVAR_HDCNAM "G2dHardcopy"
00114     #define TCS_HDCFILE_NAME "HDC%03i.HDC"
00115
00116
00117
00118 #define TCS_INISECT2 "Layout"
00119 /* #define TCS_INIVAR_COPMEN "G2dSysMenuCopy"
00120     #define TCS_INIDEF_COPMEN "Copy"
00121  #define TCS_INIVAR_FONT "G2dGraphicFont"
00122     #define TCS_INIDEF_FONT PROGDIRTOKEN "graph2d"
00123  #define TCS_INIVAR_SYSFONT "G2dSystemFont"
00124     #define TCS_INIDEF_SYSFONT PROGDIRTOKEN "graph2d"
00125 */
00126  #define TCS_INIVAR_WINPOSX "G2dGraphicPosX"
00127     #define TCS_INIDEF_WINPOSX 1
00128  #define TCS_INIVAR_WINPOSY "G2dGraphicPosY"
00129     #define TCS_INIDEF_WINPOSY 3
00130  #define TCS_INIVAR_WINSIZX "G2dGraphicSizeX"
00131     #define TCS_INIDEF_WINSIZX 98
00132  #define TCS_INIVAR_WINSIZY "G2dGraphicSizeY"
00133     #define TCS_INIDEF_WINSIZY 85
00134 /* #define TCS_INIVAR_STATPOSX "G2dStatusPosX"
00135     #define TCS_INIDEF_STATPOSX 1
00136  #define TCS_INIVAR_STATPOSY "G2dStatusPosY"
00137     #define TCS_INIDEF_STATPOSY 91
```

```
00138  #define TCS_INIVAR_STATSIZX "G2dStatusSizeX"
00139     #define TCS_INIDEF_STATSIZX 98
00140  #define TCS_INIVAR_STATSIZY "G2dStatusSizeY"
00141     #define TCS_INIDEF_STATSIZY 3
00142  */
00143  #define TCS_INIVAR_LINCOL "G2dLinCol"
00144     #define TCS_INIDEF_LINCOL 1
00145  #define TCS_INIVAR_TXTCOL "G2dTxtCol"
00146     #define TCS_INIDEF_TXTCOL 1
00147  #define TCS_INIVAR_BCKCOL "G2dBckCol"
00148     #define TCS_INIDEF_BCKCOL 0
00149
00150 #define TCS_INISECT3 "Messages"
00151  #define TCS_INIVAR_UNKNGRAPHCARD "G2dGraphCard"
00152     #define TCS_INIDEF_UNKNGRAPHCARD "GRAPH2D Video System: Error %s."
00153     #define TCS_INIVAR_UNKNGRAPHCARDL "G2dGraphCardL"
00154     #define TCS_INIDEF_UNKNGRAPHCARDL 10
00155  #define TCS_INIVAR_NOFNTFIL "G2dFntfilOpen"
00156     #define TCS_INIDEF_NOFNTFIL "GRAPH2D SDLTTF: Error opening Fontfile %s."
00157     #define TCS_INIVAR_NOFNTFILL "G2dFntfilOpenL"
00158     #define TCS_INIDEF_NOFNTFILL 10
00159  #define TCS_INIVAR_NOFNT "G2dFntfilOpen"
00160     #define TCS_INIDEF_NOFNT "GRAPH2D SDLTTF: Error -> %s."
00161     #define TCS_INIVAR_NOFNTL "G2dFntfilOpenL"
00162     #define TCS_INIDEF_NOFNTL 10
00163  #define TCS_INIVAR_HDCOPN "G2dHdcOpen"
00164     #define TCS_INIDEF_HDCOPN "GRAPH2D HARDCOPY: Error during OPEN."
00165     #define TCS_INIVAR_HDCOPNL "G2dHdcOpenL"
00166     #define TCS_INIDEF_HDCOPNL 5
00167  #define TCS_INIVAR_HDCWRT "G2dHdcWrite"
00168     #define TCS_INIDEF_HDCWRT "GRAPH2D HARDCOPY: Error during WRITE."
00169     #define TCS_INIVAR_HDCWRTL "G2dHdcWriteL"
00170     #define TCS_INIDEF_HDCWRTL 5
00171  #define TCS_INIVAR_USR "G2dUser"
00172     #define TCS_INIDEF_USR "%s"
00173     #define TCS_INIVAR_USRL "G2dUserL"
00174     #define TCS_INIDEF_USRL 5
00175  #define TCS_INIVAR_HDCACT "G2dHdcActive"
00176     #define TCS_INIDEF_HDCACT "Hardcopy in progress: File %s created."
00177     #define TCS_INIVAR_HDCACTL "G2dHdcActiveL"
00178     #define TCS_INIDEF_HDCACTL 1
00179  #define TCS_INIVAR_USRWRN "G2dPressAny"
00180     #define TCS_INIDEF_USRWRN "Press any key to continue."
00181     #define TCS_INIVAR_USRWRNL "G2dPressAnyL"
00182     #define TCS_INIDEF_USRWRNL 5
00183  #define TCS_INIVAR_EXIT "G2dExit"
00184     #define TCS_INIDEF_EXIT "Press any key to exit program."
00185     #define TCS_INIVAR_EXITL "G2dExitL"
00186     #define TCS_INIDEF_EXITL 10
00187  #define TCS_INIVAR_COPMEM "G2dNoMemory"
00188     #define TCS_INIDEF_COPMEM "GRAPH2D Clipboard Manager: Out of Memory."
00189     #define TCS_INIVAR_COPMEML "G2dNoMemoryL"
00190     #define TCS_INIDEF_COPMEML 1
00191  #define TCS_INIVAR_COPLCK "G2dClipLock"
00192     #define TCS_INIDEF_COPLCK "GRAPH2D Clipboard Manager: ClipBoard locked."
00193     #define TCS_INIVAR_COPLCKL "G2dClipLockL"
00194     #define TCS_INIDEF_COPLCKL 1
00195  #define TCS_INIVAR_JOUCREATE "G2dJouCreate"
00196     #define TCS_INIDEF_JOUCREATE "GRAPH2D Error Creating Journal. Error-No: %s."
00197     #define TCS_INIVAR_JOUCREATEL "G2dJouCreateL"
00198     #define TCS_INIDEF_JOUCREATEL 5
00199  #define TCS_INIVAR_JOUENTRY "G2dJouEntry"
00200     #define TCS_INIDEF_JOUENTRY "GRAPH2D Error Creating Journal Entry."
00201     #define TCS_INIVAR_JOUENTRYL "G2dJouEntryL"
00202     #define TCS_INIDEF_JOUENTRYL 5
00203  #define TCS_INIVAR_JOUADD "G2dJouAdd"
00204     #define TCS_INIDEF_JOUADD "GRAPH2D Error Appending Journal Entry."
00205     #define TCS_INIVAR_JOUADDL "G2dJouAddL"
00206     #define TCS_INIDEF_JOUADDL 5
00207  #define TCS_INIVAR_JOUCLR "G2dJouClr"
00208     #define TCS_INIDEF_JOUCLR "GRAPH2D Error Clearing Journal Entry."
00209     #define TCS_INIVAR_JOUCLRL "G2dJouClrL"
00210     #define TCS_INIDEF_JOUCLRL 5
00211  #define TCS_INIVAR_JOUUNKWN "G2dJouEntryUnknwn"
00212     #define TCS_INIDEF_JOUUNKWN "GRAPH2D Unknown Journal Entry."
00213     #define TCS_INIVAR_JOUUNKWNL "G2dJouEntryUnknwnL"
00214     #define TCS_INIDEF_JOUUNKWNL 5
00215  #define TCS_INIVAR_XMLPARSER "G2dXMLerror"
00216     #define TCS_INIDEF_XMLPARSER "GRAPH2D Error parsing XML-File: %s"
00217     #define TCS_INIVAR_XMLPARSERL "G2dXMLerrorL"
00218     #define TCS_INIDEF_XMLPARSERL 8
00219  #define TCS_INIVAR_XMLOPEN "G2dXMLopen"
00220     #define TCS_INIDEF_XMLOPEN "GRAPH2D Error opening %s"
00221     #define TCS_INIVAR_XMLOPENL "G2dXMLopenL"
00222     #define TCS_INIDEF_XMLOPENL 0 // no errormessage due to wxTCSmain.cpp
00223  #define TCS_INIVAR_UNKNAUDIO "G2dAudio"
00224     #define TCS_INIDEF_UNKNAUDIO "GRAPH2D Audio System: Error %s."
```

```
00225     #define TCS_INIVAR_UNKNAUDIOL "G2dAudioL"
00226     #define TCS_INIDEF_UNKNAUDIOL 5
00227  #define TCS_INIVAR_USR2 "G2dUser2"
00228     #define TCS_INIDEF_USR2 "%s"
00229     #define TCS_INIVAR_USR2L "G2dUser2L"
00230     #define TCS_INIDEF_USR2L 5
00231  #define TCS_INIVAR_INI2 "G2dInitt"
00232     #define TCS_INIDEF_INI2 "Error creating windows in subroutine INITT"
00233     #define TCS_INIVAR_INI2L "G2dInittL"
00234     #define TCS_INIDEF_INI2L 1
```

9.34 TCSdrWXfor.f08 File Reference

wX Port: High-Level Driver

Functions/Subroutines

- subroutine tcslev (LEVEL)
- subroutine winlbl (PloWinNam, StatWinNam, IniFilNam)
- subroutine initt (iDummy)
- 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 csize (ixlen, iylen)
- subroutine statst (String)
- subroutine graphicerror (iErr, Mssg)
- subroutine anmode

  Entry dummy routines.

9.34.1 Detailed Description

wX Port: High-Level Driver

Version

(2025,64,8)

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

wX 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 TCSdrWXfor.f08.

9.34.2 Function/Subroutine Documentation

9.34.2.1 anmode()

```
subroutine anmode
```

Entry dummy routines.
AlfMod
pClipt
alpha
Definition at line 247 of file TCSdrWXfor.f08.

9.34.2.2 csize()

```
subroutine csize (
              ixlen,
              iylen )
```

Definition at line 197 of file TCSdrWXfor.f08.

9.34.2.3 drwrel()

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

Definition at line 114 of file TCSdrWXfor.f08.

9.34.2.4 dshrel()

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

Definition at line 124 of file TCSdrWXfor.f08.

9.34.2.5 graphicerror()

```
subroutine graphicerror (
             integer iErr,
             character *(*) Mssg )
```

Definition at line 224 of file TCSdrWXfor.f08.

9.34.2.6 initt()

```
subroutine initt (
             integer iDummy )
```

Definition at line 70 of file TCSdrWXfor.f08.

9.34.2.7 movrel()

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

Definition at line 94 of file TCSdrWXfor.f08.

9.34.2.8 pntrel()

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

Definition at line 104 of file TCSdrWXfor.f08.

9.34.2.9 seeloc()

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

Definition at line 138 of file TCSdrWXfor.f08.

9.34.2.10 statst()

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

Definition at line 206 of file TCSdrWXfor.f08.

9.34.2.11 tcslev()

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

Definition at line 39 of file TCSdrWXfor.f08.

9.34.2.12 toutpt()

```
subroutine toutpt (
             integer iChr )
```

Definition at line 151 of file TCSdrWXfor.f08.

9.34.2.13 toutst()

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

Definition at line 169 of file TCSdrWXfor.f08.

9.34.2.14 toutstc()

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

Definition at line 180 of file TCSdrWXfor.f08.

9.34.2.15 winlbl()

```
subroutine winlbl (
             character*(*) PloWinNam,
             character*(*) StatWinNam,
             character*(*) IniFilNam )
```

Definition at line 53 of file TCSdrWXfor.f08.

9.35 TCSdrWXfor.f08

```
00001 !> \file       TCSdrWXfor.f08
00002 !> \brief      wX Port: High-Level Driver
00003 !> \version    (2025,64,8)
00004 !> \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00005 !> \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00006 !>
00007 !> \~german
00008 !> wX-spezifische TCS-Routinen
00009 !> \note \verbatim
00010 !>    Erweiterungen gegenüber Tektronix:
00011 !>     subroutine TOUTSTC (String): Ausgabe Fortran-String
00012 !>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
00013 !>     subroutine TXTCOL (iCol): Setzen Textfarbe
00014 !>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00015 !>     subroutine DefaultColour: Wiederherstellung Defaultfarben
00016 !> \endverbatim
00017 !>
00018 !>
00019 !> \~english
00020 !> wX specific subroutines
00021 !> \note \verbatim
00022 !>    Supplement to Tektronix:
00023 !>     subroutine TOUTSTC (String): Ausgabe Fortran-String
00024 !>     subroutine LINCOL (iCol): Setzen Linienfarbe (iCol=0..15)
00025 !>     subroutine TXTCOL (iCol): Setzen Textfarbe
00026 !>     subroutine BCKCOL (iCol): Hintergrundfarbe (nach ERASE sichtbar)
00027 !>     subroutine DefaultColour: Wiederherstellung Defaultfarben
00028 !> \endverbatim
00029 !> \~
00030 !>
00031
00032
00033 ! FTN 77 linkbare Unterprogramme / Wrapper
00034
00035 !
00036 !  Ausgabe der Softwareversion
00037 !
00038
00039       subroutine tcslev(LEVEL)
00040       integer LEVEL(3)
00041       level(1)=2025     ! Aenderungsjahr
00042       level(2)=   64     ! Aenderungstag
00043       level(3)=    8     ! System= wX
00044       return
00045       end
00046
00047
00048
00049 !
00050 !  Initialization
00051 !
00052
00053       subroutine winlbl (PloWinNam, StatWinNam, IniFilNam)
00054       use, intrinsic :: iso_c_binding
00055       implicit none
00056
00057       character*(*) PloWinNam, StatWinNam, IniFilNam
00058       interface
00059         subroutine winlbl0 (PloWinNam0, StatWinNam0, IniFilNam0) bind(C, name='winlbl0')
00060         use, intrinsic                        :: iso_c_binding, only: c_char
00061         character(kind= c_char), dimension(*) :: PloWinNam0, StatWinNam0, IniFilNam0
00062         end subroutine winlbl0
00063       end interface
00064
00065       call winlbl0 (plowinnam//c_null_char, statwinnam//c_null_char, inifilnam//c_null_char)
00066       end
00067
00068
00069
00070       subroutine initt (iDummy)
00071       use, intrinsic :: iso_c_binding
00072       implicit none
00073
00074       integer iDummy
00075       integer (c_intptr_t), parameter :: NULLPTR = 0
00076       interface
00077         subroutine initt1 (iMode, iParent, iFrame, iStatus) bind(C)
00078           use, intrinsic              :: iso_c_binding
00079           integer (c_int), value      :: iMode
00080           integer (c_intptr_t), value :: iParent, iFrame, iStatus
00081         end subroutine initt1
```

```
00082       end interface
00083
00084       call initt1 (0, nullptr, nullptr, nullptr) ! 0 => no Parent Window
00085       return
00086       end
00087
00088
00089
00090 !
00091 !  Relative drawing
00092 !
00093
00094       subroutine movrel (iX, iY)
00095       include 'Tktrnx.fd'
00096       ixx= kbeamx + ix
00097       iyy= kbeamy + iy
00098       call movabs (ixx, iyy)
00099       return
00100       end
00101
00102
00103
00104       subroutine pntrel (iX, iY)
00105       include 'Tktrnx.fd'
00106       ixx= kbeamx + ix
00107       iyy= kbeamy + iy
00108       call pntabs (ixx, iyy)
00109       return
00110       end
00111
00112
00113
00114       subroutine drwrel (iX, iY)
00115       include 'Tktrnx.fd'
00116       ixx= kbeamx + ix
00117       iyy= kbeamy + iy
00118       call drwabs (ixx, iyy)
00119       return
00120       end
00121
00122
00123
00124       subroutine dshrel (iX, iY, iMask)
00125       include 'Tktrnx.fd'
00126       ixx= kbeamx + ix
00127       iyy= kbeamy + iy
00128       call dshabs (ixx, iyy, imask)
00129       return
00130       end
00131
00132
00133
00134  !
00135  !  Ersatz SEELOC der CP/M-Version (wie MS Windows, DOS)
00136  !
00137
00138        subroutine seeloc (IX,IY)
00139        include 'Tktrnx.fd'
00140        ix= kbeamx
00141        iy= kbeamy
00142        return
00143        end
00144
00145
00146
00147 !
00148 !  Graphic text output
00149 !
00150
00151       subroutine toutpt (iChr)
00152       use, intrinsic :: iso_c_binding
00153       implicit none
00154       integer iChr
00155
00156       interface
00157         subroutine outgtext (strng) bind(C, name='outgtext_')
00158         use, intrinsic                        :: iso_c_binding, only: c_char
00159         character(kind= c_char), dimension(*) :: strng
00160         end subroutine outgtext
00161       end interface
00162
00163       call outgtext (char(ichr)//c_null_char)
00164       return
00165       end
00166
00167
00168
```

```
00169       subroutine toutst (nChr, iChrArr)
00170       integer iChrArr (1)
00171       if (nchr.eq.0) return
00172       do 10 i=1,nchr
00173        call toutpt (ichrarr(i))
00174 10    continue
00175       return
00176       end
00177
00178
00179
00180       subroutine toutstc (String)
00181       implicit none
00182
00183       character *(*) String
00184       interface
00185         subroutine outgtext (strng) bind(C, name='outgtext_')
00186         use, intrinsic                        :: iso_c_binding, only: c_char
00187         character(kind= c_char), dimension(*) :: strng
00188         end subroutine outgtext
00189       end interface
00190
00191       call outgtext (string//char(0))
00192       return
00193       end
00194
00195
00196
00197       subroutine csize (ixlen,iylen)
00198       include 'Tktrnx.fd'
00199       ixlen= khorsz
00200       iylen= kversz
00201       return
00202       end
00203
00204
00205
00206       subroutine statst (String)
00207       use, intrinsic :: iso_c_binding
00208       implicit none
00209
00210       character *(*) String
00211       interface
00212         subroutine outtext (cString) bind(C, name='outtext_')
00213         use, intrinsic                        :: iso_c_binding, only: c_char
00214         character(kind= c_char), dimension(*) :: cString
00215         end subroutine outtext
00216       end interface
00217
00218       call outtext (string//c_null_char)
00219       return
00220       end
00221
00222
00223
00224       subroutine graphicerror (iErr,Mssg) ! Bis jetzt genutzt: TCSGraphicError in Cpp
00225       use, intrinsic :: iso_c_binding
00226       implicit none
00227
00228       integer iErr
00229       character *(*) Mssg
00230       interface
00231         subroutine tcsgraphicerror (i, cString) bind(C, name='TCSGraphicError')
00232         use, intrinsic                        :: iso_c_binding
00233         integer(kind=c_int), value            :: i
00234         character(kind= c_char), dimension(*) :: cString
00235         end subroutine tcsgraphicerror
00236       end interface
00237
00238       call tcsgraphicerror (ierr,mssg//c_null_char)
00239       return
00240       end
00241
00242
00243
00244 !
00245 !>  Entry dummy routines
00246 !
00247       subroutine  anmode
00248 !> AlfMod
00249       entry       alfmod
00250 !> pClipt
00251       entry       pclipt
00252 !> alpha
00253       entry       alpha
00254       return
00255       end
```

9.36 Tktrnx.fd File Reference

wX Port: TCS Common Block TKTRNX

9.36.1 Detailed Description

wX Port: TCS Common Block TKTRNX

Version

1.0

Author

Dr.-Ing. Klaus Friedewald

Header belonging to TKTRNX.hpp. The Source Format complies to the requirements of FTN77 Fixed Formar as well as Fortran08 Free Form.

Note

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

Definition in file Tktrnx.fd.

9.37 Tktrnx.fd

```
00001 !> \file Tktrnx.fd
00002 !> \brief   wX Port: TCS Common Block TKTRNX
00003 !> \version 1.0
00004 !> \author  Dr.-Ing. Klaus Friedewald
00005 !> \~german
00006 !> Header passend zu TKTRNX.hpp. Das Quelltextformat ist sowohl zum FTN77 Fixed
00007 !> Format als auch zum Ftn08 Free Format kompatibel.
00008 !> \note
00009 !> Da die folgende Definition kein Bestandteil eines Moduls
00010 !> ist, versagt der DOXYGEN-Parser bei der Kombination von
00011 !> COMMON und INTEGER. Workaraound: \\cond ... \\endcond.
00012 !> \~english
00013 !> Header belonging to TKTRNX.hpp. The Source Format complies to the
00014 !> requirements of FTN77 Fixed Formar as well as Fortran08 Free Form.
00015 !> \note
00016 !> Because the following definition not being part of a module, the
00017 !> DOXYGEN parser is not able to handle the combination of COMMON
00018 !> and INTEGER declarations.  Workaround: \\cond ... \\endcond.
00019 !> \~
00020 !> \cond
00021
00022       use iso_c_binding, only: c_int, c_float, c_sizeof
00023
00024       integer (c_int)                                                  &
00025      & khomey,                                                         &
00026      & khorsz,kversz,                                                  &
00027      & kitalc,ksizef,                                                  &
00028      & klmrgn,krmrgn, kScrX,kScrY,                                     &
00029      & kbeamx,kbeamy,                                                  &
00030      & kminsx,kminsy,kmaxsx,kmaxsy
00031       real (c_float)                                                   &
00032      & tminvx,tminvy,tmaxvx,tmaxvy,                                    &
00033      & trcosf,trsinf,trscal,                                           &
00034      & xfac,yfac,xlog,ylog
00035       integer (c_int)                                                  &
00036      & kStCol,                                                         &
00037      & iLinCol, iBckCol, iTxtCol
00038
00039
00040        COMMON /tktrnx/                                                 &
00041      & khomey,                                                         &
00042      & khorsz,kversz,                                                  &
00043      & kitalc,ksizef,                                                  &
00044      & klmrgn,krmrgn, kscrx,kscry,                                     &
00045      & kbeamx,kbeamy,                                                  &
00046      & kminsx,kminsy,kmaxsx,kmaxsy,tminvx,tminvy,tmaxvx,tmaxvy,        &
00047      & trcosf,trsinf,trscal,                                           &
00048      & xfac,yfac,xlog,ylog,kstcol,                                     &
00049      & ilincol, ibckcol, itxtcol
```

```
00050
00051       SAVE /tktrnx/
00052       bind(c, name='tktrnx_') :: /tktrnx/
00053
00054 !> \endcond
00055
00056
```

9.38 TKTRNX.hpp File Reference

wX Port: TCS Common Block TKTRNX

Classes

- struct TKTRNX

Variables

- struct TKTRNX tktrnx_

9.38.1 Detailed Description

wX Port: TCS Common Block TKTRNX

Version

1.0

Author

Dr.-Ing. Klaus Friedewald

C header belonging to TKTRNX.fd

Note

wX-Version auf Basis der SDL-Version 1.2

Definition in file TKTRNX.hpp.

9.38.2 Variable Documentation

9.38.2.1 tktrnx_

```
struct TKTRNX tktrnx_
```

9.39 TKTRNX.hpp

```
00001 /** ****************************************************************************
00002 \file    TKTRNX.hpp
00003 \brief   wX Port: TCS Common Block TKTRNX
00004 \version 1.0
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    wX-Version auf Basis der SDL-Version 1.2
00014
00015 ******************************************************************************* */
00016
00017 extern "C" {
00018   extern struct TKTRNX {
```

```
00019     int
00020      khomey,
00021      khorsz,kversz,
00022      kitalc,ksizef,
00023      klmrgn,krmrgn, kScrX,kScrY,
00024      kbeamx,kbeamy,
00025      kminsx,kminsy,kmaxsx,kmaxsy;
00026
00027     float
00028      tminvx,tminvy,tmaxvx,tmaxvy,
00029      trcosf,trsinf,trscal
00030      ,xfac,yfac,xlog,ylog;
00031     int
00032      kStCol,
00033      iLinCol, iBckCol, iTxtCol;
00034   } tktrnx_; // use gfortran FTN77 name mangling
00035 }
00036
```

9.40 wxTCSmain.cpp File Reference

Initialization of wxWidgets.

```
#include <wx/wx.h>
#include <wx/filename.h>
#include <wx/stdpaths.h>
#include "graph2d.h"
```

Classes

- class wxTCSapp

Macros

- #define MainProgram MAIN__

Functions

- void _gfortran_set_args (int argc, char ∗argv[ ])

9.40.1 Detailed Description

Initialization of wxWidgets.

Version

1.0

Author

(C) 2023 Dr.-Ing. Klaus Friedewald

Copyright

GNU LESSER GENERAL PUBLIC LICENSE Version 3

wxTCSapp for executing Fortran console programs Since the windows are created before the Fortran program is executed (and thus before a call to WINLBL), an initialization file with the name of the main program is used.
Definition in file wxTCSmain.cpp.

9.40.2 Macro Definition Documentation

9.40.2.1 MainProgram

```
void MainProgram MAIN__
```

Definition at line 20 of file wxTCSmain.cpp.

9.40.3 Function Documentation

9.40.3.1 _gfortran_set_args()

```
void _gfortran_set_args (
            int argc,
            char * argv[] )
```

9.41 wxTCSmain.cpp

```
00001 /** ****************************************************************************
00002 \file       wxTCSmain.cpp
00003 \brief      Initialization of wxWidgets
00004 \version    1.0
00005 \author     (C) 2023 Dr.-Ing. Klaus Friedewald
00006 \copyright  GNU LESSER GENERAL PUBLIC LICENSE Version 3
00007 \~german
00008          wxTCSapp zur Ausführung von Fortran-Konsolenprogrammen
00009          Da die Fenster vor dem Ausführen des Fortranprogrammes (und somit vor
00010          einem Aufruf von WINLBL) erstellt werden, wird eine Initialisierungsdatei
00011          mit dem Namen des Hauptprogrammes verwendet.
00012 \~english
00013          wxTCSapp for executing Fortran console programs
00014          Since the windows are created before the Fortran program is executed
00015          (and thus before a call to WINLBL), an initialization file with the
00016          name of the main program is used.
00017 \~
00018 ******************************************************************************* */
00019
00020 #define MainProgram MAIN__
00021 // #define MainProgram ftnmain2sub_
00022
00023 #include <wx/wx.h>
00024 #include <wx/filename.h>
00025 #include <wx/stdpaths.h>
00026 #include "graph2d.h"
00027
00028
00029 extern "C" {
00030     void MainProgram (); // subroutine plot f1
00031 }
00032
00033 extern "C" {
00034     void _gfortran_set_args (int argc, char *argv[]);
00035 }
00036
00037
00038
00039 class wxTCSapp : public wxApp
00040 {
00041 public:
00042     virtual bool OnInit();
00043     virtual void OnIdle();
00044 private:
00045     bool MainStarted = false;
00046     wxFrame* wxAppframe;
00047 };
00048
00049 IMPLEMENT_APP(wxTCSapp)
00050
00051 bool wxTCSapp::OnInit() // Build wx Event Loop
00052 {
00053   wxString wxTmpStr;
00054   wxFileName wxTmpFilNam;
00055
00056
00057     wxAppframe = new wxFrame((wxFrame*) NULL, -1, GetAppDisplayName(),
      wxDefaultPosition,wxDefaultSize, wxDEFAULT_FRAME_STYLE);
00058     wxAppframe->Show(true);
00059     SetTopWindow(wxAppframe);
00060
```

```
00061     _gfortran_set_args (wxAppConsole::argc, wxAppConsole::argv); // Initialize FTN command-line
      intrinsics
00062
00063     Connect(wxEVT_IDLE,(wxObjectEventFunction)&wxTCSapp::OnIdle);
00064
00065     wxTmpFilNam= wxStandardPaths::Get().GetExecutablePath();
00066     wxTmpStr= wxTmpFilNam.GetName();
00067     wxTmpStr.Prepend("%:");  wxTmpStr.Append(".%");
00068
00069     winlbl0 ("","", wxTmpStr.c_str() ); // read default inifile before creating windows
00070     initt1 (2, nullptr, wxAppframe, nullptr); // use wxAppframe for plotting
00071
00072     return true;
00073 }
00074
00075 void wxTCSapp::OnIdle()
00076 {
00077     if (!MainStarted) {
00078       MainStarted= true; // 1st statement to avoid recursive invocation, e.g. due to wxYield() in
      tinput
00079       MainProgram();
00080       wxAppframe->Refresh();
00081     }
00082     return;
00083 }
```

Index