```
//
// Interna main program
// 15.jan 2004/sc
//

#include <tuxbase.h>
#include "use_tuxvars.h"
#include "interna.h"
#include <sql2tuxbase.h>


extern Widget global_IconContainer;
extern int global_containercounter;
extern Widget global_ContainerShell[];
extern Boolean global_containershell_aktiv[];
extern Widget global_ContainerToggle[];
extern void *program_end_routine;

// Prototypen
// ----------
void dialog_loadfile_Create (Widget, XtPointer, XtPointer);
void artikel_import (Widget, XtPointer, XtPointer);
void artikel_export (Widget, XtPointer, XtPointer);
void artikel_misc_import (Widget, XtPointer, XtPointer);
void adressen_misc_import (Widget, XtPointer, XtPointer);
void adressen_vcf_export (Widget, XtPointer, XtPointer);
void Print_Hotline (Widget, XtPointer, XtPointer);

//
// Dokumentation
// -------------
void interna_print_handbook_html (Widget parent, XtPointer data, XtPointer cbs);
void interna_print_tutor_html (Widget parent, XtPointer data, XtPointer cbs);
void interna_print_reference_html (Widget parent, XtPointer data, XtPointer cbs);



struct PMdata *
txPm_CreatePulldown (Widget parent)
{

  static struct PMdata *PM = NULL;

  // Hole den nötigen Speicher
  // -------------------------
  if (txPm_AllocMem (&PM, __FUNCTION__))
    {

      if (txPm_Init (parent, PM))
            {

              // Bedienen Menue
              // --------------
              txPm_OpenMain ("1111", tx_LangTrans ("Bearbeiten"), 'B', "", PM, NULL, tx_TipMsgTrans("\
Dieses Menü stellt Ihnen verschiedene Funktionen zur Verfügung.\n"));
              {
                if (txPm_BeginHtmlDokumentation (parent, PM))
                  {
                        txHtm_Print (tx_DocMsgTrans ("\n"));
                        txPm_FinishHtmlDokumentation (parent, PM);
                  }
              }
              {
                txPm_AddItem ("1111", tx_LangTrans ("Login"), 'L', "", "", NULL, NULL, "", PM, NULL, tx_TipMsgTrans("\
Hiermit verbinden Sie sich mit einer Datenbank auf einem entfernten Server.\n"));
                {
                  if (txPm_BeginHtmlDokumentation (parent, PM))
                        {
```

```
                    txHtm_Print (tx_DocMsgTrans ("\n"));
                    txPm_FinishHtmlDokumentation (parent, PM);
                  }
            }
            txPm_AddItem ("1111", tx_LangTrans ("Logout"), 'O', "", "", NULL, NULL, "", PM, NULL, tx_TipMsgTrans("\
Sie trennen die bestehende Verbindung zu einem Datenbank-Server.\n"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                  {
                    txHtm_Print (tx_DocMsgTrans ("\n"));
                    txPm_FinishHtmlDokumentation (parent, PM);
                  }
            }
            txPm_AddSep (PM);
            txPm_AddItem ("1111", tx_LangTrans ("Mandanten auswählen"), 0, "Ctrl<Key>0", "Strg-0",
tx_ChooseMandant, NULL, "", PM, NULL, tx_TipMsgTrans ("\
Wählen Sie den Mandanten aus, für den Sie Ihre Arbeiten durchführen.\n"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                  {
                    txHtm_Print (tx_DocMsgTrans ("Der Mandant ist der Auftraggeber, für den Arbeiten durchgeführt
werden.\n\
Wählen Sie an dieser Stelle den Auftraggeber aus.\n"));
                    txPm_FinishHtmlDokumentation (parent, PM);
                  }
            }
            txPm_AddItem ("1111", tx_LangTrans ("Mandanten entfernen"), 0, "", "", Func_EraseMandant, NULL, "", PM,
NULL, tx_TipMsgTrans ("\
Wählen Sie den Mandanten aus, den Sie mit allen Daten löschen wollen.\n"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                  {
                    txHtm_Print (tx_DocMsgTrans ("\
Diese Funktion entfernt alle Daten, die zu einem ausgewählten Mandanten gehören.\n\
Bitte gehen Sie mit dieser Funktion besonders vorsichtig um, da dieser Vorgang absolut nicht rückgängig zu machen
ist.\n\
Zuerst wählen Sie den Mandanten aus, den Sie löschen wollen.\n\
Danach werden Sie wiederholt gefragt, ob Sie auch tatsächlich diesen Mandanten endgültig löschen möchten.\n\
Erst dann wird der Löschvorgang durchgeführt."));
                    txPm_FinishHtmlDokumentation (parent, PM);
                  }
            }
            txPm_AddSep (PM);
            txPm_AddItem ("1111", "Datenbank aus Version 86.2 importieren", 'L', "", "", tx_DatabaseImport, NULL, "",
PM, NULL, tx_TipMsgTrans ("\
Sie können bestehende Daten aus einer Vorgängerversion übernehmen.\n\
Dieses Programm führt das nicht automatisch durch.\n\
Sinnvollerweise sollten Sie diesen Vorgang nur einmal durchführen.\n\
Bereits bestehende neue Einträge werden durch den Import aber nicht überschrieben.\n"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                  {
                    txHtm_Print (tx_DocMsgTrans ("Der Import aus einer Datenbank der älteren Softwareversionen
86.2 dient dem Zweck, Ihre bereits erfassten Datenbestände in dieses neue System zu integrieren.\n\
Dabei werden alle Felder übernommen. Sofern bereits Einträge in Ihrer neuen Datenbank vorhanden sind, werden
diese aber nicht überschrieben.\n\
Arbeitsweise:\n\
Es werden alle Tabelle der fremden Datenbank durchsucht.\n\
Sofern eine Tabelle den gleichen Namen hat, wie eine Tabelle dieses Programms, \
wird diese in die erste Untersuchung genommen.\n\
Dann werden alle Felder dieser Tabelle auf Namensgleichheit mit der Tabelle dieses Programms verglichen.\n\
Sofern die Namen übereinstimmen, wird der Inhalt übernommen. Dabei spielen die Eigenschaften des Feldes keine
Rolle.\n\
Der komplette Eintrag wird als Zeichenfolge kopiert."));
                    txPm_FinishHtmlDokumentation (parent, PM);
                  }
            }
            txPm_AddSep (PM);
```

```
            txPm_AddItem ("1111", tx_LangTrans ("Einstellung"), 'V', "", "", (XtCallbackProc)
dialog_ARCADSetup_Create, (XtPointer) NULL, "hi_5", PM, NULL, tx_LangTrans ("\
Sie stellen die Größe der Fonts, Ikons ein. Zusätzlich können Sie die Hintergrundfarbe des Bildschirms wählen.\n"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                    {
                     txHtm_Print (tx_DocMsgTrans ("\n"));
                     txPm_FinishHtmlDokumentation (parent, PM);
                    }
            }
            txPm_AddSep (PM);
            txPm_AddItem ("1111", tx_LangTrans ("Ende"), 'n', "Alt<Key>F4", "Alt+F4", EndProgramm, NULL, "", PM,
NULL, tx_TipMsgTrans("Hier verlassen Sie das Programm.\nAlle Daten sind bereits in Ihrer Datenbank
gespeichert.\n"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                    {
                     txHtm_Print (tx_DocMsgTrans ("\n"));
                     txPm_FinishHtmlDokumentation (parent, PM);
                    }
            }
           }
           txPm_CloseMain (PM);

           // Makroeditor Menue
           // -----------------
           txPm_OpenMain ("1111", tx_LangTrans ("Fakturierung"), 'F', "", PM, NULL, tx_TipMsgTrans("\
Dieses Menü ermöglicht Ihnen die Erstellung und Bearbeitung diverser kaufmännischer Vorgänge. Ferner finden Sie\
hier die Unterpunkte Versandpapiere und verschiedene Zusammenfassungen.\n"));
           {
            if (txPm_BeginHtmlDokumentation (parent, PM))
              {
                    txHtm_Print (tx_DocMsgTrans ("\n"));
                    txPm_FinishHtmlDokumentation (parent, PM);
              }
           }


.....snip.....

.....snap.....


            txPm_AddItem ("1111", tx_LangTrans ("Füge Text ein"), 'F', "Ctrl<Key>F", "Strg F", NULL, NULL, "", PM,
NULL, tx_TipMsgTrans(""));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                    {
                     txHtm_Print (tx_DocMsgTrans ("\n"));
                     txPm_FinishHtmlDokumentation (parent, PM);
                    }
            }
           }
           txPm_CloseMain (PM);


           // Hilfe Menue
           // -----------------
           txPm_OpenMainH ("1111", "Hilfe", 'H', "hilfe_menu", PM, NULL, "Aufruf der Hilfe-Funktion");
           {
            if (txPm_BeginHtmlDokumentation (parent, PM))
              {
                    txHtm_Print (tx_DocMsgTrans ("\n"));
                    txPm_FinishHtmlDokumentation (parent, PM);
              }
           }
           {
            txPm_AddItem ("1111", tx_LangTrans ("Handbuch Grundlagen"), 'R', "", "", (XtCallbackProc)
```

```
interna_print_handbook_html, (XtPointer) NULL, "hi_4", PM, NULL, tx_LangTrans ("Zeigt das ARCAD Grundlagen-
Handbuch an"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                    {
                     txHtm_Print ("Erzeugt ein HTML Grundlagen-Handbuch, das direkt von Ihrem gewählten Browser
aufgerufen wird.\n");
                     txPm_FinishHtmlDokumentation (parent, PM);
                    }
            }

            txPm_AddItem ("1111", tx_LangTrans ("Handbuch Referenz"), 'R', "", "", (XtCallbackProc)
interna_print_reference_html, (XtPointer) NULL, "hi_4", PM, NULL, tx_LangTrans ("Zeigt die ARCAD Kommandos an"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                    {
                     txHtm_Print ("Erzeugt ein HTML Handbuch, das direkt von Ihrem gewählten Browser aufgerufen
wird.\n");
                     txPm_FinishHtmlDokumentation (parent, PM);
                    }
            }
            txPm_AddSep (PM);
            txPm_AddItem ("1111", tx_LangTrans ("Über..."), 'V', "", "", (XtCallbackProc) tx_DisplayLicenceVersion,
(XtPointer) NULL, HELP_LICENCE, PM, NULL, tx_LangTrans ("Informationen zur CAD-Lizenz"));
            {
             if (txPm_BeginHtmlDokumentation (parent, PM))
                    {
                     txHtm_Print ("Zeigt die aktuelle Version Ihrer Software an.\n");
                     txPm_FinishHtmlDokumentation (parent, PM);
                    }
            }
           }
           txPm_CloseMain (PM);


           // Programm Eigenarten werden an dieser Stelle berücksichtigt
           // -------------------------------------------------------
           txPm_DisableItem4Evaluation ("INTERNA_PROVISION", PM);


           // Schliesse jetzt das Pulldown ab
           // ------------------------------
           txPm_FinishPulldown (parent, PM);

          }
   }

 return PM;

}



//
##########################################################################
#######################
//
##########################################################################
#######################
// Hauptprogramm EINSPRUNG
//
##########################################################################
#######################
//
##########################################################################
#######################

int
```

```
main (int argc, char *argv[])
{

  // Zeige auf die Routine, die am Ende des Programmes
  // aufgerufen werden kann.
  // Optional.
  // ------------------------------------------------
  // program_end_routine = ### routine muss an dieser Stelle eingesetzt werden ###;


  // Setze die interne Nummer des Produktes
  // Diese Nummer bezeichnet das Erzeugnis
  // und wird mit der Lizenznummer verglichen
  // ----------------------------------------
  setProduktNumber (PRODID_INTERNA);

// Wenn Sie wollen, können Sie auch eine Funtion auf das Startbild legen
// Wenn Sie mit der Maus diese Fläche anklicken, wird Ihre Funktion aufgerufen
// Diese Funktion muss die Syntax 'void func (Widget, XtPointer, XtPointer)' haben.
// ------------------------------------------------------------------------------
  tx_SetAboutFunction (tx_DisplayLicenceVersion);



  // ------------------------------------------------------------
  // Initialisiere die Hauptfläche
  // INIT_AS_TERMINAL   Initialisiert als normales Terminal-Fenster
  // INIT_AS_CADAREA    Initialisiert mit kompletter CAD Oberfläche
  // ------------------------------------------------------------
  if (txVw_InitToplevelForm (argc, argv, "Interna", NULL, INIT_AS_TERMINAL))
    {

      // Baue jetzt die Verbindung zum SQL Server auf
      // Die Informationen dafür holt diese Routine
      // aus der interna.ini Datei
      // ---------------------------------------------
      if (txSql_InitConnection ())
            {

              // **********************************
              // An dieser Stelle können die
              // Vorbedingungen für den Aufbau der
              // Datenbank gesetzt werden.
              // **********************************


              ctrace (MSG_INFO, "Create AliasTables");
              // ------------------------------------------------
              // Erstelle die Aliasing Tabellen
              // Diese können komplett unabhängig im Tuxbase-System
              // verwendet werden, obgleich Sie eine gemeinsame
              // Mutter-Tabelle haben.
              // Beispielsweise können Kunden, Lieferanten unabhängig
              // bearbeitet werden, ohne sich zu stören, obgleich
              // Sie alle zur Tabelle Adressen gehören.
              // ------------------------------------------------
              txSql_BuildAliases ("backadre", "adressen");
              txSql_BuildAliases ("backarti", "artikel");
              txSql_BuildAliases ("backgewerke", "gewerke");
              txSql_BuildAliases ("bauzstatus_ist", "bauzstatus");
              txSql_BuildAliases ("backordh", "orderhea");

              // ------------------------------------------------
              // Zusätzlich setze noch den Default-Wert der
              // Alias-Tabelle auf einen neuen Wert.
              // ------------------------------------------------
              if (txSql_BuildAliases ("kunde", "adressen"))
                txSql_DBField2Default ("kunde", adressen_TYP, "1");
```

```
            if (txSql_BuildAliases ("bauherr", "adressen"))
              txSql_DBField2Default ("bauherr", adressen_TYP, "1");

            if (txSql_BuildAliases ("bestellt", "adressen"))
              txSql_DBField2Default ("bestellt", adressen_TYP, "1");

            if (txSql_BuildAliases ("lieferant", "adressen"))
              txSql_DBField2Default ("lieferant", adressen_TYP, "2");

            if (txSql_BuildAliases ("mitarbeiter", "adressen"))
              txSql_DBField2Default ("mitarbeiter", adressen_TYP, "3");

            if (txSql_BuildAliases ("kurier", "adressen"))
              txSql_DBField2Default ("kurier", adressen_TYP, "4");

            if (txSql_BuildAliases ("sachkonto", "adressen"))
              txSql_DBField2Default ("sachkonto", adressen_TYP, "9");


            // STARTE jetzt die Programm(EVENT)schleife
            // ----------------------------------------
            txVw_RunProgram ();

          }
    }

  return 0;

}
```

```
#include <tuxbase.h>
#include "use_tuxvars.h"

extern char byref[];


// Include Datei für Ihre Projektabhängigen Daten
// ----------------------------------------------
#include <sql2tuxbase.h>


// PROTOTYPES
// ----------
void adressen_sl (Widget parent, XtPointer data, XmPushButtonCallbackStruct * cbs);
void adressentyp_sl (Widget parent, XtPointer data, XmPushButtonCallbackStruct * cbs);
void gruppen_sl (Widget parent, XtPointer data, XmPushButtonCallbackStruct * cbs);
void anrede_sl (Widget parent, XtPointer data, XmPushButtonCallbackStruct * cbs);
void steuer_sl (Widget parent, XtPointer data, XmPushButtonCallbackStruct * cbs);


// Wollen Sie in dieser View editieren erlauben?
// ---------------------------------------------
Boolean UseNavigatebuttons = True;




Widget
View_Adressen (Widget parent, XtPointer data, XtPointer cbs)
{

  // Misc Variablen, die im Verlauf der Routine
  // gebraucht werden
  // ----------------------------------------
  Widget feldform, manager, tabbed_frame, mainrc;

  // Verankere an dieser Stelle die Tuxbase Hauptstruktur
  // ----------------------------------------------------
  static struct CBdata *CB = NULL;


  // ----------------------------
  // Hole den nötigen Speicher
  // Sofern bereits diese View
  // Aufgerufen wurde, wird
  // nicht weiter verzweigt,
  // sondern direkt an das
  // Ende der Routine gesprungen
  // und die bereits vorhandene
  // View eingeblendet
  // ----------------------------
  if (tx_AllocCBMemory (&CB, __FUNCTION__))
    {

      // Wenn die Shell noch nicht erstellt worden ist,
      // wird jetzt die Shell durch die folgenden
      // Angaben gebildet
      // ---------------------------------------------
      if (CB->Shell == NULL)
            {

              // Erzeuge die Shell
              // -----------------
              mainrc = txVw_ContainerInit (parent, tx_LangTrans ("Adressen"), CB, ICONIFY_TRUE);


              // Zeiger auf die Master-DB
```

```
            // ------------------------
            CB->dbnr = dbnr ("adressen");


            // Definition der verwendeten Datenbänke, die in dieser View
            // vorkommen.
            // Es werden Kopien für den Gebrauch in dieser View angefertigt
            // damit Änderungen in den Feldern nur in dieser View
            // aktiv sind.
            // Erst nach Speichern stehen diese Änderungen auch in der
            // Datenbank.
            // ---------------------------------------------------------
            txSql_CloneDBTbl (CB->db, db, "adressen");
            txSql_CloneDBTbl (CB->db, db, "backadre");
            txSql_CloneDBTbl (CB->db, db, "gruppe");
            txSql_CloneDBTbl (CB->db, db, "adressentyp");
            txSql_CloneDBTbl (CB->db, db, "anrede");
            txSql_CloneDBTbl (CB->db, db, "steuer");
            txSql_CloneDBTbl (CB->db, db, "orderterms");


            // NEU:
            // Achtung; diese View darf beim Blättern mit den Pfeiltasten
            // NUR die Personenkonten anzeigen. Daher muss an diese Stelle
            // ein Vorfilter definiert werden.
            // Syntax ist; where ... AND 'diese angabe'
            // ---------------------------------------------------------------------------
            txVw_SetConstrain (CB, "and (typ>=\"1\" and typ<\"9\")");  // nur adressdaten
            // zwischen 1 und kleiner
            // 9 werden angezeigt


            // Setze jetzt die Navigationsbuttons
            // Die Buttons werden dort plaziert, wo sie in
            // der Reihenfolge des Aufbaus dieser Seite
            // defininiert werden.
            // Wenn Sie am Ende der View gesetzt werden sollen,
            // muß diese Angabe am Ende dieses Programmes stehen.
            // ------------------------------------------------
            if (UseNavigatebuttons)
              txVw_NavigationEnable (parent, mainrc, CB);



// Feldform 1
//
#######################################################################
#####################################


            // Neuer Rahmen
            // Containerbox für die folgenden Felder
            // -------------------------------------
            txVw_Form (mainrc, &feldform, "", FORM_FRAME);
            {

              int z = 16;
              int y = -z;

              CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
              CB->Vf[CB->vwmax].field = adressen_SUCHBEGRIFF;
              CB->Vf[CB->vwmax].LookUp = (XtCallbackProc) adressen_sl;
              CB->Vf[CB->vwmax].Func = (XtCallbackProc) __bloub;      // Individuelle Funktion
              CB->Vf[CB->vwmax].behavior = TXVW_AUTOLOOKUP | TXVW_AUTOFIND;    // Blendet LookUp bei Enter
und leerem Feld ein
              txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Suchbegriff"), "A", "U", 64, y += z, 306, 1,
tx_LangTrans ("Hilfe/nTip"));
              // ------------------------------------------------
              // Die routinen können so oft angegeben werden, wie
```

```
		// CB->vwmax Einträge vorhanden sind.
		// Dabei spielt es keine Rolle, wo diese stehen
		// Es wird immer die gesamte view durchsucht und
		// nicht etwa nur der jeweilige eintrag.
		// ------------------------------------------------
		CB->Vf[CB->vwmax].routine[0][PROOF_FOR_DELETE] = adressen_delete_ok;		// Diese Routine wird
aufgerufen, bevor gespeichert wird
		CB->Vf[CB->vwmax].routine[0][PROOF_FOR_SAVING] = adressen_save_ok; // Diese Routine wird
aufgerufen, bevor gespeichert wird
		CB->Vf[CB->vwmax].routine[0][PROOF_FOR_UPDATE] = adressen_update_ok;		// Diese Routine wird
aufgerufen, bevor gespeichert wird
		CB->vwmax++;


		CB->Vf[CB->vwmax].rel_dbnr = dbnr ("adressen");
		CB->Vf[CB->vwmax].rel_field = adressen_KENNUNG;
		CB->Vf[CB->vwmax].dbnr = dbnr ("gruppe");
		CB->Vf[CB->vwmax].field = gruppe_GRUPPE;
		CB->Vf[CB->vwmax].LookUp = (XtCallbackProc) gruppen_sl;
		CB->Vf[CB->vwmax].behavior = TXVW_AUTOLOOKUP;	// Blendet LookUp bei Enter und leerem Feld ein
		CB->Vf[CB->vwmax].Zoom = View_Gruppen;	// Blendet bei einem Zoom die Adressen ein!
		txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Gruppe"), "A", "", 407, y, 126, 1, tx_LangTrans
("Alle Adressen, die zu einer gemeinsamen Gruppe gehören, können im eMail-Rundschreiben in einer Aussendung
verschickt werden"));
		CB->vwmax++;


		CB->Vf[CB->vwmax].rel_dbnr = dbnr ("adressen");
		CB->Vf[CB->vwmax].rel_field = adressen_TYP;
		CB->Vf[CB->vwmax].dbnr = dbnr ("adressentyp");
		CB->Vf[CB->vwmax].field = adressentyp_TYP;
		CB->Vf[CB->vwmax].LookUp = (XtCallbackProc) adressentyp_sl;
		CB->Vf[CB->vwmax].behavior = TXVW_AUTOLOOKUP;	// Blendet LookUp bei Enter und leerem Feld ein
		CB->Vf[CB->vwmax].LookUpGroup = "ATyp";
		CB->Vf[CB->vwmax].lowrange = 1;
		CB->Vf[CB->vwmax].highrange = 4;
		txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Adressentyp"), "N", "", 64, y += z, 30, 1, "");
		CB->vwmax++;

		txVw_CreateTextfield (CB, feldform, False, "", "A", "", 99, 16, 238, 1, tx_LangTrans ("Adressen-Typ-
Bezeichnung"));
		CB->Vf[CB->vwmax].dbnr = dbnr ("adressentyp");
		CB->Vf[CB->vwmax].field = adressentyp_NAME;
		CB->Vf[CB->vwmax].LookUpGroup = "ATyp";
		CB->vwmax++;


		CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
		CB->Vf[CB->vwmax].field = adressen_KONTENNUMMER;
		txVw_CreateTextfield (CB, feldform, False, tx_LangTrans ("Kontennummer"), "N", "", 407, y, 42, 1, "");
		CB->vwmax++;


		CB->Vf[CB->vwmax].LookUp = (XtCallbackProc) anrede_sl;
		CB->Vf[CB->vwmax].behavior = TXVW_AUTOLOOKUP;	// Blendet LookUp bei Enter und leerem Feld ein
		CB->Vf[CB->vwmax].rel_dbnr = dbnr ("adressen");
		CB->Vf[CB->vwmax].rel_field = adressen_ANREDE;
		CB->Vf[CB->vwmax].dbnr = dbnr ("anrede");
		CB->Vf[CB->vwmax].field = anrede_NAME;
		CB->Vf[CB->vwmax].Zoom = View_Anrede;	// Blendet bei einem Zoom die Adressen ein!
		txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Anrede"), "A", "", 64, y += z, 150, 1,
tx_LangTrans ("Wenn eine Anrede angegeben ist, \
wird diese als erste Zeile \
in einem Adressfeld eines \
Anschreiben gedruckt"));
		CB->vwmax++;
```

..... snip .....

..... snap .....

```
                // 4. Spalte
                // ---------
                CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
                CB->Vf[CB->vwmax].field = adressen_KARENZTAGE;
                txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Karenztage"), "N", "", 420, y += z, 25, 1, "");
                CB->vwmax++;

                CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
                CB->Vf[CB->vwmax].field = adressen_MAHNGEBUEHR1;
                txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Mahngebühr1"), "C", "", 420, y += z, 42, 1, "");
                CB->vwmax++;

                CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
                CB->Vf[CB->vwmax].field = adressen_MAHNGEBUEHR2;
                txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Mahngebühr2"), "C", "", 420, y += z, 42, 1, "");
                CB->vwmax++;

                CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
                CB->Vf[CB->vwmax].field = adressen_MAHNGEBUEHR3;
                txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Mahngebühr3"), "C", "", 420, y += z, 42, 1, "");
                CB->vwmax++;

                CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");
                CB->Vf[CB->vwmax].field = adressen_PROVISION;
                txVw_CreateTextfield (CB, feldform, True, tx_LangTrans ("Provision"), "A", "U", 420, y += z, 30, 1,
tx_LangTrans ("Kann diese Adresse \
Provision aus einer \
Fakturierung erhalten?"));
                CB->vwmax++;

               }
               tx_ManageChild (feldform);
               tx_ManageChild (manager);
               tx_ManageChild (mainrc);

               // Schliesse die Shell ab
               // ----------------------
               txVw_ContainerFinish (parent, mainrc, CB);

               // Fülle die Views mit den Defaults auf
               // Wenn die View das erste Mal dargestellt wird,
               // wird auch die View mit den entsprechenden
               // Defaultwerten passend zur Master Database
               // aufgefüllt.
               // ---------------------------------------------
               txVw_ZeroWithDefaults (parent, (XtPointer) CB, cbs);
              }
    }

  // Zeige an
  // Wenn diese View geschlossen wird, wird beim zweiten
  // Aufruf diese nur wieder dargestellt, nicht aber neu
  // konstruiert.
  // ---------------------------------------------------
  txVw_RaiseView (parent, CB, data);

  return CB->Shell;

}
```

```
#include <tuxbase.h>
#include "use_tuxvars.h"

// Include Datei für Ihre Projektabhängigen Daten
// ----------------------------------------------
#include <sql2tuxbase.h>



void
adressen_sl (Widget parent, XtPointer data, XmPushButtonCallbackStruct * cbs)
{

  // Verankere an dieser Stelle die Tuxbase Hauptstruktur
  // ----------------------------------------------------
  static struct CBdata *CB = NULL;

  // ----------------------------
  // Hole den nötigen Speicher
  // Sofern bereits diese Selektion
  // Aufgerufen wurde, wird
  // nicht weiter verzweigt,
  // sondern direkt an das
  // Ende der Routine gesprungen
  // und die bereits vorhandene
  // View eingeblendet
  // ----------------------------
  if (tx_AllocCBMemory (&CB, __FUNCTION__))
    {

      // Setze den Zeiger auf die aufrufende VIEW
      // Es ist ein Pointer in struct CBdata Form
      // ----------------------------------------
      CB->Parent = (struct CBdata *) data;

      // parent ist der AddSuchButton, der zuerst diese
      // Routine in der View aufgerufen hat.
      // -----------------------------------------------
      CB->parentWidget = parent;


      // Wenn die Shell noch nicht erstellt worden ist,
      // wird jetzt die Shell durch die folgenden
      // Angaben gebildet
      // ----------------------------------------------
      if (CB->Shell == NULL)
             {

               Widget feldform, mainrc;

               // Erzeuge die Shell
               // -----------------
               mainrc = txVw_ContainerInit (parent, tx_LangTrans("Adressen"), CB, ICONIFY_FALSE);


               // Zeiger auf die Master-DB
               // ------------------------
               CB->dbnr = dbnr ("adressen");


               // Definition der verwendeten Datenbänke, die in dieser View
               // vorkommen.
               // Es werden Kopien für den Gebrauch in dieser View angefertigt
               // damit Änderungen in den Feldern nur in dieser View
               // aktiv sind.
               // Erst nach Speichern stehen diese Änderungen auch in der
               // Datenbank.
               // -------------------------------------------------------
```

```
	txSql_CloneDBTbl (CB->db, db, "adressen");


	// #######################
	// Adressen Tabelle
	// #######################

	// Neuer Rahmen
	// Containerbox für die folgenden Felder
	// -----------------------------------
	txVw_Form (mainrc, &feldform, tx_LangTrans("Adressenauswahl"), FORM_FRAME);


	// Hier werden alle Einträge der Datenbank gefunden und in der
	// Tabelle dargestellt.
	// Startselektion zur Darstellung aller benötigten Database-Einträge
	// TABLEFIELDS_REPLACEMENT Platzhalter für alle Felder der Tabelle.
	// ----------------------------------------------------------------
	strcpy (CB->Vf[CB->vwmax].Ta_List_Select_Value, "%");
	sprintf (CB->Vf[CB->vwmax].Ta_List_Select,
			"select %s from adressen where ((suchbegriff like \"%%s\") and (typ!=\"9\")) order by
suchbegriff;",
			TABLEFIELDS_REPLACEMENT);


	// Suchfeld für die Tabelle
	// ------------------------
	strcpy (CB->Vf[CB->vwmax].Qu_title, tx_LangTrans("Adressen Suche"));		// Titel der Suchroutine
	strcpy (CB->Vf[CB->vwmax].Qu_helpresource, tx_LangTrans("Auswahl"));		// Hilfe der Suchroutine (im
Resourcefile)
	strcpy (CB->Vf[CB->vwmax].Qu_label, tx_LangTrans("Suchbegriff"));			// Was zeigt die 1-
zeilige Suchroutine im Label an


	// Definiere die dargestellten Felder in der Tabellenanzeige
	// ---------------------------------------------------------
	CB->Vf[CB->vwmax].Ta_felder = 0;
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].dbnr = dbnr ("adressen");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].field = adressen_SUCHBEGRIFF;
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].label, tx_LangTrans("Suchbegriff"));
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].type, "A");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].width = 200;
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].upper = True;
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].state = True;
	CB->Vf[CB->vwmax].Ta_felder++;

	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].dbnr = dbnr ("adressen");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].field = adressen_KONTENNUMMER;
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].label, tx_LangTrans("Konto"));
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].type, "N");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].width = 50;
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].state = False;
	CB->Vf[CB->vwmax].Ta_felder++;

	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].dbnr = dbnr ("adressen");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].field = adressen_NAME;
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].label, tx_LangTrans("Name"));
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].type, "A");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].width = 100;
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].state = False;
	CB->Vf[CB->vwmax].Ta_felder++;

	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].dbnr = dbnr ("adressen");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].field = adressen_ORT;
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].label, tx_LangTrans("Ort"));
	strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].type, "A");
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].width = 100;
	CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].state = False;
```

```
            CB->Vf[CB->vwmax].Ta_felder++;

            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].dbnr = dbnr ("adressen");
            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].field = adressen_EMAIL;
            strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].label, tx_LangTrans("eMail"));
            strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].type, "A");
            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].width = 100;
            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].state = False;
            CB->Vf[CB->vwmax].Ta_felder++;

            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].dbnr = dbnr ("adressen");
            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].field = adressen_KENNUNG;
            strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].label, tx_LangTrans("Gruppe"));
            strcpy (CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].type, "A");
            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].width = 80;
            CB->Vf[CB->vwmax].Ta[CB->Vf[CB->vwmax].Ta_felder].state = False;
            CB->Vf[CB->vwmax].Ta_felder++;


            // CreateTabelle
            // -------------
            strcpy (CB->Vf[CB->vwmax].Grid_Backcolors, "lightblue,grey90");
            CB->Vf[CB->vwmax].dbnr = dbnr ("adressen");     // Dieses ist die Hauptdatenbank des Grids
            CB->Vf[CB->vwmax].field = adressen_SUCHBEGRIFF;         // Dieses Feld hat eine Verknüpfung zur View
            CB->Vf[CB->vwmax].parent_dbnr = dbnr ("adressen");       // Verweist auf die View
            CB->Vf[CB->vwmax].parent_field = adressen_SUCHBEGRIFF;          // da dieses Grid das Hauptgrid ist
            txVw_CreateGrid (CB, feldform, True, False, False, tx_LangTrans("Adressen Auswahl"), 0, 0, 0, 0, 12, dbnr
("adressen"), tx_LangTrans("Auswahl"),
                        tx_LangTrans("Adressen Tabelle"));
            CB->vwmax++;


            // Abschluss und zeige Feld an
            // ---------------------------
            tx_ManageChild (feldform);


            // Setze die buttons
            // -----------------
            txSl_NavigationEnable (parent, mainrc, CB);


            // Schliesse die Shell ab
            // ----------------------
            txVw_ContainerFinish (parent, mainrc, CB);

           }

            txSl_RaiseSelection (parent, (XtPointer) CB, NULL);

    }
}
```

```
// ----------------------

#include <tuxbase.h>
#include "use_tuxvars.h"


// Include Datei für Ihre Projektabhängigen Daten
// ----------------------------------------------
#include <sql2tuxbase.h>


// Interne Deklarationen
// ---------------------
static void Run_Report (Widget parent, XtPointer, XtPointer);
static void Report_Header (Widget parent, XtPointer, XtPointer);
static void Report_SubHeader (Widget parent, XtPointer, XtPointer);
static void Report_Section_Top (Widget parent, XtPointer, XtPointer);
static void Report_Section_Bottom (Widget parent, XtPointer, XtPointer);
static void Report_Body (Widget parent, XtPointer, XtPointer);
static void Report_Bottom (Widget parent, XtPointer, XtPointer);

static int counter = 0;
static short AusgabeForm = 0;
#define OrderByKonto 0
#define OrderByName  1
#define OrderByLandName 2
#define OrderByLandOrtName 3
#define OrderByEmail 4
#define OrderByGroup 5


extern char byref[];
static Boolean NurWennUmsatz = False;


// Formatierungsanweisung:
// \\l000       linksbündige Ausgabe mit angegebener Länge
// \\r000       rechtsbündige Ausgabe mit angegebener Länge
// -------------------------------------------------------
//                          KONTENNUMMER    NAME    LAND    PLZ     ORT     TELEFON1  EMAIL LFNR
static char form1[MAXCHLEN] = "\\r010%s\\l012%s\\l055%s\\r070%s\\l072%s\\l110%s\\l140\\L026%s\\r175%s\0";




void
Print_AdressenShortOrderByKonto (Widget parent, XtPointer data, XtPointer cbs)
{

  static struct CBdata *CB = NULL;

  // Hole den nötigen Speicher
  // -------------------------
  if (tx_AllocCBMemory (&CB, __FUNCTION__))
    {

      // gesetzt werden, die den Report betreffen
      // ----------------------------------------
      AusgabeForm = OrderByKonto;

      if (CB->Shell == NULL)
            {

              Widget feldform, mainrc, rc;

              // Erzeuge die Shell
              // -----------------
```

```
            mainrc = txVw_ContainerInit (parent, tx_LangTrans ("Personen-Konten Druck"), CB, ICONIFY_FALSE);

            // Neuer Rahmen
            // Containerbox für die folgenden Felder
            // -------------------------------------
            txVw_Form (mainrc, &feldform, "", FORM_ONLY);

            // txVw_CreateTogglebox...
            // -------------------------------
            txVw_AddToToggleItem (CB, tx_LangTrans ("Kontoumsatz"), (XtPointer) NULL, (XtPointer) &
NurWennUmsatz, tx_LangTrans ("Druck nur die Konten, die auch einen Umsatz haben"));
            txVw_CreateTogglebox (feldform, tx_LangTrans ("Druckauswahl"), CB, 310, 0, True, tx_LangTrans
("Druckart"));
            CB->vwmax++;

            tx_ManageChild (feldform);


            // -------------------------------------------
            // Setze die Buttons
            // -------------------------------------------
            CB->DialogMask = DIALOG_MASK_OK | DIALOG_MASK_CANCEL;
            CB->Dialog_OK.Proc = Run_Report;
            txVw_DialogNavigation (parent, mainrc, CB);



            tx_ManageChild (mainrc);

            // Schliesse die Shell ab
            // ----------------------
            txVw_ContainerFinish (parent, mainrc, CB);

           }
     else
           {
            // Zeige an
            // --------
            XMapRaised (XtDisplay (CB->Shell), XtWindow (CB->Shell));
           }
   }
}



void
Print_AdressenShortOrderByName (Widget parent, XtPointer data, XtPointer cbs)
{
  AusgabeForm = OrderByName;
  Run_Report (parent, data, cbs);
}

void
Print_AdressenShortOrderByGroup (Widget parent, XtPointer data, XtPointer cbs)
{
  AusgabeForm = OrderByGroup;
  Run_Report (parent, data, cbs);
}



static void
Run_Report (Widget parent, XtPointer data, XtPointer cbs)
{


  // Setze die Tuxbase Struktur für diesen Report an
```

```
// dieser Stelle fest
// ---------------------------------------------
struct CBdata *CB = NULL;


// Hole den nötigen Speicher
// Sofern kein Speicher mehr zur
// Verfügung steht, läuft das Programm
// direkt bis an das Ende
// und verzweigt nicht weiter
// ----------------------------------
if (tx_AllocCBMemory (&CB, __FUNCTION__))
  {

    // Initialisiere den Report und bestimme
    // den Drucker.
    // ----------------------------------
    if (txRp_Init (parent, CB, cbs))
          {


            // An dieser Stelle werden die ersten Parameter auf die
            // PostScript-Seite geschrieben.
            // Alle Ausgabe die durch TuxBase erzeugt werden, sind
            // zuerst IMMER Postscript Dateien.
            // ----------------------------------------------------
            txRp_InitPage (parent, CB, NULL);

              {

                MYSQL_RES *db_res = NULL;
                int reihen = 0;
                int felder = 0;


                // HauptDateinummer (Master)
                // -------------------------
                CB->dbnr = dbnr ("adressen");

                // Definition der verwendeten Datenbänke, die in diesem Report
                // vorkommen.
                // ----------------------------------------------------------
                txSql_CloneDBTbl (CB->db, db, "adressen");
                txSql_CloneDBTbl (CB->db, db, "steuer");


                // Gliederungen
                // Bei welchen Umbruch soll welche
                // Routine aufgerufen werden?
                // ------------------------------
                CB->druckjob.report.proc[RepProc_Page][RepProc_Top] = Report_Header;
                CB->druckjob.report.proc[RepProc_Page2][RepProc_Top] = Report_SubHeader;
                CB->druckjob.report.proc[RepProc_Body][RepProc_Top] = Report_Body;
                CB->druckjob.report.proc[RepProc_Page][RepProc_Foot] = Report_Bottom;


                // Selektion
                // Interne Besonderheiten
                // ----------------------
                        strcpy (CB->druckjob.titel, tx_LangTrans ("Adressenliste sortiert nach Land, Ort, Name"));
                        sprintf (byref,
                                "select * from adressen where ((suchbegriff between \"%s\" and \"%
szzzzzzzzzzzzzzzzzzzzzzz\")and(Typ!=\"9\")) order by land,ort,suchbegriff;",
                                CB->druckjob.min, CB->druckjob.max);


                        // Breaks, die die jeweiligen Sectionen auslösen
                        // ---------------------------------------------------
```

```
                    CB->druckjob.report.proc[RepProc_Section][RepProc_Top] = Report_Section_Top;
                    CB->druckjob.report.proc[RepProc_Section][RepProc_Foot] = Report_Section_Bottom;
                    txRp_SetBreaks (parent, RepProc_Section, 0, dbnr ("adressen"), adressen_KENNUNG, CB);


                  // Reset
            counter = 0;

            // Starte die Hauptschleife, die durch den
            // gesamten Report läuft.
            // ---------------------------------------
            if (txSql_FindSet (parent, byref, &mysql, &db_res , CB->db[dbnr ("adressen")], dbnr ("adressen"),
&reihen, &felder, 1))
                  {
                    while (txSql_FindSet (parent, "", &mysql, &db_res , CB->db[dbnr ("adressen")], dbnr ("adressen"),
&reihen, &felder, 3))
                      {
                              txRp_Trigger (parent, CB, cbs);
                      };
                    txSql_FindSet (parent, "", &mysql, &db_res , CB->db[dbnr ("adressen")], dbnr ("adressen"),
&reihen, &felder, 5);
                  }
          }

          // Und Abschluss
          // -------------
          txRp_Finish (parent, CB, cbs);

         }

    // Speicher wieder freigeben
    // -------------------------
    tx_FreeCBMemory (&CB);

  }
}




static void
Report_Header (Widget parent, XtPointer data, XtPointer cbs)
{
 struct CBdata *CB = (struct CBdata *) data;

 // Dieser Header bewirkt KEINEN Seitenumbruch
 // Eine PAGE Anweisung, die intern immer läuft, führt den Seitenumbruch durch
 // ---------------------------------------------------------------------------
 txRp_PageHeader (parent, CB, JOBINFO_TRUE, SEITENZAHL_TRUE, IMAGE_FALSE);

 // Testausgabe eines Positionierungslineales
 // -----------------------------------------
 // txRp_Lineal (parent, CB);

 // Setze die Vertikale Position
 // für die folgenden Absätze
 // Entweder: Anschrift oder auf folgenden Seiten Artikelheader
 // -----------------------------------------------------------
 txRp_OneLineForward (parent, CB);
 txRp_OneLineForward (parent, CB);
}



static void
Report_SubHeader (Widget parent, XtPointer data, XtPointer cbs)
{
```

```
  struct CBdata *CB = (struct CBdata *) data;

  // Setze den Font und die Größe
  // ----------------------------
  txRp_Fontmode (_k_PS_KURSIVDRUCKAN, CB);
  txRp_Fontsize (8, CB);

  // Gebe den Drucktext aus
  // ----------------------
  sprintf (byref, form1, tx_LangTrans ("Konto"), tx_LangTrans ("Name"), tx_LangTrans ("Land"), tx_LangTrans ("PLZ"),
              tx_LangTrans ("Ort"), tx_LangTrans ("Tel"), tx_LangTrans ("email"), tx_LangTrans ("ldnr"));
  txRp_String (parent, byref, CB);

  // Setze den Font auf die Vorgabewerte zurück
  // ------------------------------------------
  txRp_Fontsize (DefaultFontsize, CB);
  txRp_Fontmode (_k_PS_KURSIVDRUCKAUS, CB);

  // Gebe eine Trennlinie aus
  // ------------------------
  txRp_Line (parent, 0, RightPaperBorder, 0.25, MODE_MIDDLELINE, CB);

}


static void
Report_Section_Top (Widget parent, XtPointer data, XtPointer cbs)
{
  struct CBdata *CB = (struct CBdata *) data;

  // Setze den Font und die Größe
  // ----------------------------
  txRp_Fontmode (_k_PS_FETTDRUCKAN, CB);
  txRp_Fontsize (8, CB);

  // Gebe den Drucktext aus
  // ----------------------
  txRp_String (parent, "", CB);
  sprintf (byref, form1, tx_LangTrans ("Gruppe"), CB->db[dbnr ("adressen")]->fd[adressen_KENNUNG]->value, "", "",
"", "", "", "");
  txRp_String (parent, byref, CB);

  // Setze den Font auf die Vorgabewerte zurück
  // ------------------------------------------
  txRp_Fontsize (DefaultFontsize, CB);
  txRp_Fontmode (_k_PS_FETTDRUCKAUS, CB);

  // Gebe eine Trennlinie aus
  // ------------------------
  txRp_Line (parent, 0, RightPaperBorder, 0.25, MODE_MIDDLELINE, CB);


// Packe sofort auch auf die gruppen datei
// netter service, sofern gruppendatei leer
// ----------------------------------------
  txSql_SetTblValues2Defaults (db[dbnr ("gruppe")]->fd, dbnr ("gruppe"));
  tx_valcpy (db, dbnr ("gruppe"), gruppe_GRUPPE, CB->db[dbnr ("adressen")]->fd[adressen_KENNUNG]->value);
  ladeoderlegeneuan_gruppe (txWidget_TopLevel);


}


static void
Report_Section_Bottom (Widget parent, XtPointer data, XtPointer cbs)
{
  struct CBdata *CB = (struct CBdata *) data;
```

```
  // Erzwinge einen Seitenumbruch bei der nächsten Zeile
  // ---------------------------------------------------
  txRp_ForcePagebreak (parent, CB, cbs);

  // Reset
  counter = 0;

}



static void
Report_Body (Widget parent, XtPointer data, XtPointer cbs)
{
  struct CBdata *CB = (struct CBdata *) data;
  char byref2[16];

  // Da in den folgenden Zeilen eine Variable verwendet wird,
  // die aber zeitversetzt angezeigt wird, machen wir hier
  // einen Aufruf, um die Header, etc. bereist abzuspulen
  // Immer, wenn eine Laufvariable direkt am Anfang einer Body-Section
  // verwendet wird, ist es ratsam, diesen Vorcheck durchzuführen.
  // ---------------------------------------------------------------
  (void) txRp_Pagecheck (parent, CB);

  // Formatiere und gebe die Zeile aus
  // ---------------------------------
  sprintf (byref2, "%d", ++counter);
  sprintf (byref, form1,
              CB->db[dbnr ("adressen")]->fd[adressen_KONTENNUMMER]->value,
              CB->db[dbnr ("adressen")]->fd[adressen_NAME]->value,
              CB->db[dbnr ("adressen")]->fd[adressen_LAND]->value,
              CB->db[dbnr ("adressen")]->fd[adressen_PLZ]->value,
              CB->db[dbnr ("adressen")]->fd[adressen_ORT]->value,
              CB->db[dbnr ("adressen")]->fd[adressen_TELEFON1]->value, CB->db[dbnr ("adressen")]->fd
[adressen_EMAIL]->value, byref2);
  txRp_String (parent, byref, CB);
}



static void
Report_Bottom (Widget parent, XtPointer data, XtPointer cbs)
{
  struct CBdata *CB = (struct CBdata *) data;

  // Dieser Header bewirkt KEINEN Seitenumbruch
  // Eine PAGE Anweisung, die intern immer läuft, führt den Seitenumbruch durch
  // ------------------------------------------------------------------------
  txRp_PageBottom (parent, CB, SEITENZAHL_FALSE);
}
```