CF-Lib

Christian Felsch

COLLABORATORS			
	TITLE : CF-Lib		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Christian Felsch	April 6, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

CF-Lib

1.1 CF-Lib

Titel CF-Lib

Einleitung

Entwicklungsgeschichte

Variablen

Alerts	App	Asciitab	Comm	Cookie
Debug	Dragdrop	File	Filesel	Fontsel
Form_do	Globals	Magx	Mdial	Menu
Misc	Nkcc	Objc	Popup	Scrap
Userdef	Vdi	Wdial		

Tips zur RSC-Gestaltung

Programmliste

1.2 CF-Lib: Einleitung

Einleitung CF-Lib

Im Laufe meiner C-Programmiertttigkeit auf dem Atari habe ich
verschiedene Bibiliotheken benutzt. qed nutzte frher die MyDials
von Olaf Meisiek fr die Dialog-Darstellung. In Programmen, die
mit GNU-C programmiert sind (z.B. TosWin2), konnte ich die MyDials
nicht benutzen, da es mir nicht mglich war, die doch etwas
unbersichlichen Quellen auf GNU umzusetzen.
Da die MyDials einige Schwchen ausweisen (z.B. haben sie mit prop.
Systemzeichenstzen arge Probleme), kam die Idee, eine gnzlich
neue Bibiliothek zu schaffen. Herausgekommen ist eine Lib, die sich

in folgenden wesentlichen Punkten von den MyDials unterscheidet:

- ù Nutzung der neuen MagiC-Objekte in Dialogen. Falls MagiC nicht aktiv ist, werden diese Objekte nahezu identisch emuliert.
- ù Portabel programmiert, so da die Lib sowohl mit PureC als auch mit GNU C nutzbar ist.
- ù Die Lib kann sowohl mit den frei verfgbaren MiNT-Lib / GEM-Lib sowie den Pure-Bibliotheken / MT_AES verwendet werden.

Ansonsten stellt die Lib ein Sammelsurium von allen mglichen Routinen dar, die man immer wieder bentigt. Zentrale Funktionalitt drften die Dialogverwaltung sein, die unter Systemen ohne MagiC dessen Objekte und deren Bedienung (Stichwort Shortcuts) emuliert sowie die Bereitstellung von Dialogen und in Fenster.

Die CF-Lib wird als Freeware mit allen Quelltexten vertrieben.
Jeder kann sie benutzen und fr seine Anwendungen verndern.

Selbstverstndlich sind in der Lib noch jede Menge Fehler enthalten, die darauf warten, entdeckt zu werden :-)

Christian Felsch
Hamburg im September 1999

PS: Hier der bliche Hinweis fr die Juristen :-)

> Die Haftung fr unmittelbare und mittelbare Schden, Folgeschden und Drittschden durch die Benutzung der Bibliothek sind ausgeschlossen. Fr die Vollstndigkeit und Richtigkeit der gemachten Angaben wird keinerlei Gewhr bernommen.

1.3 CF-Lib: Entwicklungsgeschichte

Entwicklungsgeschichte CF-Lib

PL 11, 19.09.99

> *
> Quelltext freigegeben!
>
> userdef.c
> Kleine nderungen am Gruppenrahmen.
> Das Vorhandensein der MagiC-Objekte wird jetzt ber appl_getinfo(13) erfragt, so da die Lib die Objekte unter N.AES 2.0 nicht mehr emuliert.
>
> *
> Bisher wurden fehlgeschlagene malloc() nur teilweise beachtet. Jetzt wird das zentral berwacht und es gibt eine Meldung, wenn kein Speicher mehr frei ist.
> Auerdem sollte es nun keine Probleme mehr mit MP geben.
>
> mdial.c

wdial.c
Unter MagiC kann der Cursor innerhalb eines Editfelds mit der Maus positioniert werden.

PL 10, 24.02.99

cicon.c
Korrektur an der Farbicon-Darstellung fr die 32bit-int-Version (GNU). Nun sollten unter TrueColor keine bunten Radio/Check-Buttons mehr auftreten.

userdef.c
Korrektur bei der Ausgabe von disabledten Texten.

mdial.c
wdial.c
Das Edit-Objekt "-1" wird wie "0" behandelt, also, als ob der Dialog kein Editfeld hat.

magx.c
get_magx_typ() hat Check- und Radio-Buttons nicht korrekt ausgewertet. Jetzt wird zwischen MagiC-Buttons mit und ohne Shortcut unterschieden.

app.c
Die Abmessungen des Systemfonts (sys_[wh]char und sys_[wh]box) wurde nur initialisiert, wenn ein GDOS da war :-((

mdial.c
wdial.c
Neue Funktionen change_mdedit(), change_wdedit().
Sollte man einmal aufrufen, nachdem man man den Status von Editfeldern in offenen Dialogen gendert (z.B. DISABLED) hat, damit die Cursor-Verwaltung sich ein neues gltiges Editfeld sucht bzw. den Cursor korrekt versteckt.

userdef.c
Bei fix_menu/popup wurde der alte Objekttyp nicht erhalten, so da get_obtype() falsche Werte lieferte.

mdial.c
wdial.c
Korrekturem am Cursor-Redraw.

userdef.c
Neue Funktion fix_popup() als Alternative zu fix_menu(). Es ist mglich, die Trennlinien in dnne Linien zu wandeln. Auerdem nimmt die Funktion die Wandlung des Objekttyps von G_STRING in G_SHORTCUT vor, wenn MagiC aktiv ist. Wer die neuen Funktionen nicht bentigt, kann weiterhin fix_menu() fr die Popusp benutzen.

objc.c
Der Typ G_SHORTCUT wird nun korrekt behandelt.
Auerdem habe ich ein paar Panik-Alerts in diverse Funktionen eingebaut, damit es bei unbekannten Objekttypen nicht mehr zum Crash kommt!
Falls ihr solche Alerts zu Gesicht bekommt, mich bitte ber die Umstnde informieren!

file.c
Work-Around fr path_exists("X:") unter MagiCPC, da 'stat("X:")'
dort nicht funkt, wenn X: ein WinXFS ist.

userdef.c
Bei Titel-Objekten ist die Unterstreichung jetzt so lang, wie das
Objekt breit ist. Bisher wurde die effektive Textlnge benutzt.
Das Verhalten entspricht jetzt dem von MagiC.
Das Objekt MX_UNDERLINE2 (berschrift, kleine Schrift) wurde entfernt.

alerts.c
Jetzt sind insgesammt fnf Zeilen Text mglich.

app.c
Neue globale Variable gl_planes, die die Anzahl der verfgbaren
Farbebenen enthlt.

mdial.c
Die mdial-Funktionen arbeiten nicht mehr mit void-Parameter sondern
mit dem Typ MDIAL, der in cflib.h definiert ist. Somit ist ein
vernftiges Type-Checking mglich.

PL 9, 08.11.98
wdial.c
Wenn es kein Titel-Objekt in einem Dialog gibt, wurde eine Variable
nicht initialisiert, so da der Dialog ggf. nicht korrekt geffnet
wurde.

mdial.c:
open_mdial() macht direkt nach dem ffnen des Fensters eine
Messageschleife, um den Redraw noch auszufhren. Dadurch ist der
Dialog bereits komplett sichtbar, wenn open_mdial() zurckkehrt.
Andere eventuell auflaufende Nachrichten werden ber den Callback
herausgereicht.

form_do.c
mdial.c
Es wird verhindert, da beim ffnen des Dialogs der Edit-Cursor in einem
ungltigen (DISABLED, HIDETREE, !EDITABLE) Editfeld steht.

popup.c
Bei Auswahl eines Popups-Eintrags wird solange gewartet, bis die
Maustaste wieder losgelassen wird. Es werden also keine Dialog-
Objekte mehr ausgewhlt, die direkt unter dem Popupeintrag liegen.

*
Die 'ntzlichen' Typen bool, uint, uchar, ulong und ushort entfernt,
damit es keine Probleme mit anderen Bibliotheken gibt (z.B. C++).
bool wurde durch int ersetzt.

alert.c
Statt der GEM-Icons werden die der MyDials benutzt.

userdef.c
In monochrom wird bei Editfeldern nicht mehr OUTLINED gesetzt, die

Felder haben also keinen Rahmen mehr.

alert.c
Es gibt jetzt ein viertes Icon (Nummer 0, "Info").

popup.c
Korrektur am Popup-Handling: Eintrge mssen G_STRING, !DISABLED und SELECTABLE sein, damit sie whlbar sind.

mdial.c
wdial.c
Zentrierung der Dialoge beim ersten ffnen korrigiert.

file.c, fs_case_sens()
VFAT von MagiCPC behauptet zwar, da es Groklein echt unterscheidet, was aber gelogen ist -> nun wird HALF_CASE geliefert.

mdial.c
form_do.c
wdial.c
Wird ein Exit-Button mit Doppelklick ausgelst, ist Bit15 im Exit-Objekt gesetzt.

inline.rsc
Ein fehlendes 'static' bei der RSC fhrte dazu, da es bei einer Applikation mit eigener Inline-RSC zwei 'rs_trindex' gab, was natrlich zum Crash fhrt.

userdef.c
Farbicons (Radio/Check-Buttons) werden nun auch bei mehr als 256 Farben korrekt gezeichnet.

filesel.c
Bei Mehrfachauswahl mu der Callback (open_cb()) jetzt TRUE oder FALSE zurckgeben. Bei FALSE wird der Callback von select_file() nicht mehr aufgerufen.

mdial.c
Beim Versuch einen Dialog zu 'backdrop'en gibts jetzt einen Ping.

fontsel.c
Beim FontProtokoll wird geprft, ob der Fontserver auch wirklich da ist. Es gibt immer noch Leute, die in FONTSELECT eine Fontauswahl anstatt eines Font-Servers eintragen!

Nun auch mit PureLib/MT_AES benutzbar.
Die verschiedenen Libs erkennt man an einem 'm' fr MiNT-Lib bzw. einen 'p' fr PureLibs im PatchLevel.

PL 8, 31.08.98
userdef.c
Gruppenrahmen und berschrift jetzt in 3D (Dank an Jo Even Skarstein!)

fontsel.c
MAGCUFSL wird ignoriert, da dieser xFSL->MagiC-Cookie-Verbieger nicht mit der Auswahl im Fenster klar kommt. Der Anwender hat dadurch aber

```
   keinen Nachteil, weil die MagiC-Auswahl direkt, ohne Umweg ber den
   xFSL-Cookie benutzt wird!

   popupc.c
   handle_popup() verschiebt das Popup ggf. so, da es immer komplett
   auf dem Bildschirm ist.

   misc.c
   background-Funktionen berarbeitet. Sie clippen jetzt auf den Bildschirm.

   objc.c
   set_obspec() neu
   set_ulong() neu
   tree_state() neu

   popups.c
   cf_menu_popup() hat neuen Parameter
   handle_popup() sucht sich nun G_STRING-Eintrge, so da z.B. Farb-
   Auswahlpopups (vorne BOX_CHAR, dahinter String) mglich sind.

   misc.c
   Neue Funktion get_patchlev()

   fontsel.c
   Schwerer Fehler in do_magx(), der zum Verstellen des Fonts auf der
   phys. WS 1 fhrte.

   cookie.c
   Getcookie() der MiNT-Lib wird benutzt.

   file.c
   make_normalpath() liefert das Ergebnis von path_exists()


 PL 7, 12.07.98
   Erste ffnetliche Version.


 PL 6, 01.03.98
 PL 5, 19.02.98
 PL 4, 06.11.97
 PL 3, 22.10.97
```

1.4 CF-Lib: Variablen

```
Globale Variablen                                                    CF-Lib

Alle Variablen werden von init_app() gesetzt und auf sie sollte nur lesend
zugegriffen werden.

   #define CFLIB_PATCHLEVEL   Enthlt den PatchLevel als String
   char __Ident_cflib[];      Ident-String

   int   gl_apid              Applikations-Nummer
   int   gl_phys_handle       Physikalische WS
```

```
GRECT gl_desk              Gre des Desktops

int   sys_big_id           ID und Hhe der Systmezeichenstze
int   sys_big_height
int   sys_big_pts
int   sys_sml_id
int   sys_sml_height
int   sys_sml_pts

int   sys_wchar            Zeichenbox-Gren des groen Systemzeichensatz
int   sys_hchar
int   sys_wbox
int   sys_hbox

int   gl_wchar             Zeichenbox-Gren von graf_handle()
int   gl_hchar
int   gl_wbox
int   gl_hbox

int   gl_gdos              Ist 1, falls ein GDOS installiert ist
int   gl_planes            Anzahl der Farbebenen

char  gl_appdir[]          Der Pfad, von der das Programm gestartet wurde

int   gl_gem               Enthlt die GEM-Version
int   gl_mint              Versionsnummer von MiNT oder 0
int   gl_naes              Versionsnummer von N.AES oder 0
int   gl_magx              Versionsnummer von MagiC oder 0
int   gl_nvdi              Versionsnummer von NVDI oder 0
                           Alle Versionsnummern liegen als HEX-Werte
                           vor (z.B. 0x601 -> 6.01)
```

1.5 CF-Lib: Alerts

```
Alert-Boxen                                                    CF-Lib

Grundstzlich drfen Alertboxen maximal fnf Zeilen Text (~ 40 Zeichen)
und drei Buttons (~ 14 Zeichen) enthalten. Als Icons stehen die vier
bekannten zur Verfgung (0: Info, 1: Ausrufezeichen, 2: Warndreieck,
3: Stopschild).
Zustzlich zu den Shortcuts, Default- und UNDO-Button lsen F1, F2 und F3
MagiC-konform die Buttons 1 bis 3 aus. Als Shortcut wird grundstzlich
das erste Zeichen im Button verwendet.

   int do_alert(int def, int undo, char *str);
      Zeigt eine herkmmliche (modale) Alert-Box an. Parameter wie beim
      form_alert(), allerdings kann zustzlich ein Button definiert
      werden, der mit UNDO ausgelst wird.

   int do_walert(int def, int undo, char *str, char *win_title);
      Wie do_alert(), allerdings als app-modaler Fensterdialog.
      Fr das Fenster kann ein Titel angegeben werden.
      Ein Fenster-Alert verhlt sich wie ein MDial, d.h. fr den
      korrekten Redraw hintenliegender Fenster mu ein
```

eingeklinkt werden!

1.6 "CF-Lib: App

Applikation CF-Lib

Einige ntzlichen Funktionen, die jede Applikation bentigt.

void init_app(char *rsc)
Meldet die Applikation beim GEM an. Setzt alle globalen Variable und schaltet Maus auf 'Pfeil'. Wird der Name einer RSC bergeben, wird diese geladen.

void exit_gem(void)
Meldet die Applikation beim GEM ab, gibt eine eventuell geladene Resource frei.

void exit_app(int ret)
Beendet eine Applikation mit dem Returnwert <ret>. Ruft dazu exit_gem() und anschlieend exit(ret) auf.

void hide_mouse(void)
Schaltet die Maus ab.

void hide_mouse_if_needed(GRECT *rect)
Schaltet Maus aus, wenn sie innerhalb des angegebenen Rechtecks liegt.

void show_mouse(void)
Schaltet die Maus an.

int appl_xgetinfo(int type, int *out1, int *out2, int *out3, int *out4)
Ruft appl_getinfo() auf, prft aber zunchst auf dessen Vorhandensein.

1.7 CF-Lib: Asciitab

ASCII-Tabelle CF-Lib

Die ASCII-Tabelle stell alle nicht ber die Tastatur erreichbaren Zeichen zum Einfgen zur Verfgung.

int ascii_table(int id, int pts);
Ruft die Tabelle als modalen Dialog auf. bergeben werden mu der Font, der zur Darstellung der Zeichen verwendet werden soll. Font-Hhe in Point!!
Fr normalen Systemfont: 1, 10.
Die Tabelle ist vollstndig mit Maus und Tastatur bedienbar:

- ù Mit den Cursortasten kann man ein Zeichen auswhlen.
- ù HOME springt zum ersten, Shift-Home zum letzten Zeichen.

```
            ù Return bzw. Mausklick auf ein Zeichen whlt dieses aus.
            ù UNDO lst den Abbruch-Button aus.

         Die ASCII-Tabelle kann auch in jedem Editfeld durch Drcken
         von INSERT aufgerufen werden. So knnen dort auch Sonder-
         zeichen eingegeben werden, sofern die Maske des Feldes
         diese zult.


   void set_asciitable_strings(char *title, char *button)
         ndert berschrift und Button-Text der Tabelle.
```

1.8 CF-Lib: Comm

```
Kommunikation                                                    CF-Lib

   void send_vastart(int id, char *str);
         Schickt ein VA_START an die Applikation <id>.

   void send_m_special(int mode, int app);
         Schickt eine Spezial-Nachricht an den ScreenManager von MagiC.

   void send_scchanged(void);
         Verschickt ein SC_CHANGED an alle anderen Applikationen per
         Broardcast. Sollte man verwenden, wenn man das Klemmbrett
         verndert hat.
```

1.9 CF-Lib: Cookie

```
Cookie-Auslesen                                                  CF-Lib

   int getcookie(char *cookie, long *value)
         Prft auf den angegebenen Cookie.
         Eingabe:
            cookie   Name des zu berprfenden Cookies.
         Rckgabe:
            value    Inhalt des Cookie (falls vorhanden!). Wenn
                     der Inhalt nicht interessiert, kann NULL
                     bergeben werden.
         Die Routine liefert 1, wenn der Cookie vorhanden ist, sonst 0.
```

1.10 CF-Lib: Debug

```
Debugausgaben                                                    CF-Lib


   typedef enum {null, Con, TCon, Datei, Terminal,
                 Modem1, Modem2, Seriell1, Seriell2, Prn} DEVICETYP;
         Die mglichen Ausgabekanle. Bei TCON gehen die Ausgaben
         wie bei Con auf die Console, allerdings wird vorher berprft,
```

```
      ob T-Con luft.

   int gl_debug
      Zeigt an, ob debug_init() erfolgt ist.

   void debug_init(char *prgName, DEVICETYP dev, char *file);
      Initialisiert das Debugdevice.
      Eingabe:
        prgName  Wird jeder Debug-Meldung vorangestellt.
        dev      Typ des Devices
        file     Wenn dev = Datei, der Dateiname der Log-Datei,
                 sonst NULL.

   void debug_exit(void)
      Schliet das Device.

   void debug(char *FormatString, ...);
      Die eigentliche Ausgabefunktion.
      Parameter so wie bei printf().
```

1.11 CF-Lib: Dragdrop

```
Drag&Drop Protokoll                                              CF-Lib

   #define DD_OK
   #define DD_NAK
   #define DD_EXT
   #define DD_LEN
   #define DD_TIMEOUT
   #define DD_NUMEXTS
   #define DD_EXTSIZE
   #define DD_NAMEMAX
   #define DD_HDRMAX

   Funktionen fr den Sender:

   int dd_create(int apid, int winid, int msx, int msy, int kstate,
                 char *exts);
      Legt eine Pipe fr D&D an und informiert Empfnger mittels
      AP_DRAGDROP.
         apid        ID der Ziel-App
         winid       ID des Ziel-Fensters (0 fr Desktop)
         msx, msy    aktuelle Maus-Position
         kstate      Shift-Status

         exts        32 Byte groer Puffer, in dem die gltigen
                     Extensions des Empfngers geliefert werden
         Rckgabe    Entweder positives GEMDOS Dateihandle oder:
                        -1 Empfnger schickt DD_NAK
                        -2 Fehler beim appl_write(AP_DRAGDROP)

   int dd_stry(int fd, char *ext, char *name, long size);
      Prft, ob der Empnger einen bestimmten Datentyp akzeptiert.
         fd          Pipe-Handle
         ext         4 Byte Extension
```

```
          name        Name der Daten
          size        Anzahl der Bytes, die bertragen werden
                      sollen

          Rckgabe    DD_OK    wird akzeptiert
                      DD_EXT   Extension abgelehnt
                      DD_LEN   Lnge der Daten abgelehnt
                      DD_NAK   Kommunikation abgebrochen


   Funktionen fr den Empnger:

   int dd_open(int ddnam, char *ext);
         ffnet eine Pipe.

          ddnam       Nummer der Pipe (msg[7] vom AP_DRAGDROP)
          ext         Vom Empfnger verstandene Extensions. Jeweils
                      4 Bytes, insgesamt DD_NUMEXT

          Rckgabe    Entweder ein positives GEMDOS Dateihandle,
                      -1 bei Abruch durch den Sender oder einen
                      negativen GEMDOS-Fehlercode

   int dd_rtry(int fd, char *name, char *ext, long *size);
         Einen D&D-Header empfangen.
          fd          Handle der Pipe

          name        Name der Daten, max. DD_NAMEMAX Bytes lang
          ext         Typ der Daten als 4Byte-Extension
          size        Gre der Daten

          Rckgabe    1, wenn O.K.

   int dd_reply(int fd, int ack);
         Schickt 1-Byte-Antwort an den Sender.
          fd          Handle der Pipe
          ack         Antwort: (DD_OK, DD_NAK, ...)

          Rckgabe    1, wenn alles OK


   Funktionen fr beide:

   void dd_close(int fd);
         Schliet die bergeben Pipe wieder.
```

1.12 CF-Lib: File

```
Aktionen auf Dateinamen und Pfade                                  CF-Lib

   int file_exists(char *filename);
   int path_exists(char *pathname);
         Prft das Vorhandensein.

   int get_path(char *path, char drive);
```

```
int set_path(char *path);
      Liefer/Setzt den aktuellen Pfad. (drive = 0 fr Defaultdrive)

void split_filename(char *fullname, char *path, char *name);
      Spaltet einen kompletten Dateipfad in Pfad und Name auf.
      Wenn Pfad oder Name nicht interessieren, kann NULL angegeben
      werden.

void set_extension(char *filename, char *new_ext);
      ndert die Datei-Extension in einem Namen.

int make_normalpath(char *path);
      Wandelt Pfad in einen gltigen um (ggf. mit unx2dos).
      Liefert das Ergebnis von path_exists(path) zurck.

void make_shortpath(char *path, char *shortpath, int maxlen);
      Krzt <path> auf <maxlen>. Falls <path> zulang ist, werden
      '..' eingefgt.

int fs_long_name(char *filename);
      Prft, ob das Dateisystem, auf dem sich <filename> befindet,
      lange Dateinamen kann. 'Lang' bedeutet in diesem Fall mehr als
      8+3 Zeichen auf TOS-Partitionen.

int fs_case_sens(char *filename);
      Prft, ob das Dateisystem, auf dem sich <filename> befindet,
      Gro/Kleinschrift unterscheidet. Ergebnis ist eine drei-wertige
      Logik:
         NO_CASE     Kein Unterschied (TOS-FS)
         HALF_CASE   Kein echter Unterschied. Es kann zwar Gro/Klein
                     benutzt werden, beim Zugriff ist es aber egal
                     (VFAT, Mac HFS)
         FULL_CASE   Echte Unterscheidung (Minix)
```

1.13 CF-Lib: Filesel

```
Dateiauswahl                                                    CF-Lib

typedef void (* FSEL_CB)(char *path, char *name);
#define FSCB_NULL

int select_file(char *path, char *name, char *mask, char *title,
                  FSEL_CB open_cb);

   Fhrt eine Dateiauswahl durch (Einfach- oder Mehrfachauswahl)(*)
   Dabei wird eine ggf. vorhandene Mehrfachauswahl untersttzt,
   d.h. sobald Selectric-Cookie (Selectrics, Freedom, Boxkites)
   oder die MagiC-Dateiauswahl vorhanden sind, ist es mglich,
   mehrere Dateien mit einem Aufruf auszuwhlen.
      <path>      Pfad, auf dem die Auswahl geffnet wird, oder ""
      <name>      Datei, die vorselektiert wird, oder ""
      <mask>      Dateimaske, bei leerem "" wird *.* benutzt
      <title>     berschift der Auswahl
      <open_cb>   Wird in <open_cb> eine Routine angegeben, wird
                  eine Mehrfachauswahl durchgefhrt, wobei die
```

```
                  angegebene Funktion mit jedem ausgewhlten
                  Dateinamen einmal aufgerufen wird.

      Rckgabe:
         1           wenn die Auswahl erfolgreich war
         <path>      Ausgewhlter Pfad
         <name>      Ausgwhlte Datei

      Da bei der Mehrfachauswahl der Callback mit jedem Namen
      aufgerufen wird, liefert select_file() zwar TRUE, allerdings
      werden <path> und <name> leer zurckgegeben.

   Anmerkung zu Selectric:
     Bei diesem Protokoll mu man vorher festlegen, wieviele Dateien
     gleichzeitig markiert werden drfen. Derzeit sind mit der CF-Lib
     maximal 10 Dateien mglich!

   Anmerkung zu MagiC:
     Falls mit set_mdial_wincb() eine Message-Funktion angemeldet
     wurde, erscheint die Dateiauswahl im Fenster, an sonsten als
     modaler Dialog.
     Bei der MagiC-Dateiauswahl knnen im Gegensatz zu Selectric
     bei der Mehrfachauswahl beliebig viele Dateien ausgewhlt
     werden.
     In die Popups der Auswahl werden folgende Dinge eingestellt:
        Pfad-History   Mit dem bergebenen Pfad
        Maske          '*.*' wird immer an <mask> angehngt
        Sortierung     Nach Name
```

1.14 CF-Lib: Fontsel

```
Fontauswahl                                                    CF-Lib

   int do_fontsel(int flags, char *title, int *id, int *pts);
         Ruft eine Fontauswahl auf.
         ber verschiedene in <flags> gesetzte Bits kann die Font-
         auswahl konfiguriert werden:

            - Art der Auswahl:
               FS_M_XFSL      ber den xFSL-Cookie (Calvino, HuGo)
               FS_M_FPROT     ber Fontprotokoll (*)
               FS_M_MAGX      ber AES-Calls von MagiC (*)
               FS_M_ALL       Alle o.g. nacheinander, bis ein Modus
                              erfolgreich war.
               Falls mehrere Fontauswahlen im System vorhanden sind,
               werden die drei Mglichkeiten nach einander geprft.
               Also als erstes wird der xFSL-Cookie, dann $FONTSERVER
               und zuletzt MagiC probiert.

            - Sonstiges:
               FS_F_MONO      Statt aller Zeichenstze werden nur die
                              quidistanten (monospaced) angeboten.

         Desweiteren wird in <title> die berschrift fr die Font-
         auswahl bergeben.
```

```
        Im Falle, da die Auswahl erfolgreich war, wird 1 geliefert
        und die Fontdaten zurckgegeben.
        Sollte keine Fontauswahl verfgbar sein, wird 0 geliefert
        und <id> und <pts> enthalten -1.

   Anmerkung zu MagiC/xFSL:
     Falls mit set_mdial_wincb() eine Message-Funktion angemeldet
     wurde, erscheint die Fontauswahl im Fenster, an sonsten als
     modaler Dialog.

   Anmerkung zu FS_M_FPROT:
     Beim Fontprotokoll wird in der Environmentvariable 'FONTSERVER'
     ein Programm eingetragen, da dieses Protokoll beherrscht (z.B.
     der Desktop Thing).
     Der Fontserver teilt den ausgewhlten Zeichensatz mit einer
     speziellen Nachricht mit, die auerhalb von do_fontsel() in der
     globalen Eventschleife der Applikation aufluft; do_fontsel()
     liefert in diesem Fall immer FALSE zurck. Dieser Modus sollte
     also nur dann benutzt werden, wenn die Applikation diese Nach-
     richt versteht!
```

1.15 CF-Lib: Form_do

```
Dialogroutinen                                                  CF-Lib

   int  cf_form_do(OBJECT *tree, int *ed_start);
         form_do()-Ersatz mit Sondertastenauswertung. Das Start-Objekt
         mu als Variable bergeben werden, da der cf_form_do() am Ende
         dort das aktuelle Editfeld eintrgt.

   int  simple_dial(OBJECT *tree, int start_edit);
         Fhrt einen simplen Dialog durch.

   typedef int (* KEY_CB )(OBJECT *tree, int edit_obj, int kstate,
                           int *kreturn, int *next_obj);
         Typ der Tasten-Routine fr set_formdo_keycb().
         Wenn ein Exit-Objekt simuliert werden soll, mu die Routine
         den Objekt-Index in next_obj eintragen und 0 zurckliefern.
         Soll der Tasten-Event nicht auch noch durch die CF-Lib aus-
         gewertet werden, mu kreturn auf 0 gesetzt werden.

   KEY_CB set_formdo_keycb(KEY_CB new);
         Klinkt eine Routine in die Tasten-Auswertung des form_do() ein.
         Sie wird bei smtlichen Dialogen (modal/a-modal/u-modal)
         aufgerufen, bevor die CF-Lib den Tasten-Event auswertet.
         Die vorher eingestellte Funktion wird zurckgeliefert.
```

1.16 CF-Lib: Magx

```
MagiC Kontrollfunktionen                                        CF-Lib

   int  get_magx_version(void);
```

```
      Liefert die MagiX-Version zurck.
      Man kann statt dessen auch die Variable gl_magx verwenden, da
      die Funktion von init_app() aufgerufen wird.

int  get_magx_obj(OBJECT *tree, int obj);
      Prft, ob <obj> ein spezielles MagiC-Objekt ist.
      Folgende Typen werden erkannt (anhand des Status WHITEBAK):

         MX_NOTXOBJ      0     Kein MagiC-Objekt (ohne WHITEBAK)
         MX_UNDERLINE    1     String als berschrift
         MX_RADIO        2     Radiobutton
         MX_SCRADIO      3     Radiobutton mit Shortcut
         MX_CHECK        4     Checkbutton
         MX_SCCHECK      5     Checkbutton mit Shortcut
         MX_SCEXIT       6     Exitbutton mit Shortcut
         MX_SCSTRING     7     String mit Shortcut
         MX_GROUPBOX     8     Gruppenrahmen
         MX_EDIT3D       9     3D-Editfeld

         MX_GROUPBOX2   10     Gruppenrahmen, kleine Schrift (*)

         MX_UNKNOWN     -1     Unbekanntes MagiC-Objekt

      Die mit (*) markierten Objekte stellen eine Erweiterung der CF-Lib
      dar! (siehe auch Userdef)

int  get_magx_shortcut(OBJECT *tree, int obj, char *c);
      Liefert die Position des Shortcuts eines Buttons zurck, oder -1.
      In <c> wird das Shortcut-Zeichen eingetragen.
```

1.17 CF-Lib: Mdial

```
Modaler Fensterdialog                                          CF-Lib

Mit diesen Funktionen sind applikations-modale Fensterdialge mglich.
Whrend der Dialogverarbeitung, steht die Applikation still, es ist
aber mglich, zu anderen Programmen umzuschalten.
Die Funktionen sind ist re-entrant, d.h. wenn ein Dialoge offen ist,
darf ein weiterer geffnet werden (z.B. Fenster-Alert).

typedef (* MDIAL_WCB )(int *msg);
      Callback fr die Fenster-Events der anderen Fenster. MDial
      hat eine eigene Event-Schleife, die bestimmte WM_-Nachrichten
      (z.B. WM_REDRAW, WM_MOVED) ber diesen Callback herausreicht,
      damit hinter dem modalen Dialog liegende Fenster der Applikation
      entsprechend reagieren knnen.
      ACHTUNG: die Routine reicht keine Message weiter, die irgendwelche
      Aktionen in dem Programm auslsen knnten, z.B. VA_START usw!!

void set_mdial_wincb(MDIAL_WCB new);
      Meldet die Prozedur fr Fensterevents an.
      Dieser Message-Handler wird unter anderem auch bei Datei- und
      Fontauswahl sowie Alerts in Fenstern benutzt.
```

```
   MDIAL *open_mdial(OBJECT *tree, int edit_start);
         Legt einen neuen Dialog an und ffnet das Fenster. Zurckgegeben
         wird das Dialog-Handle oder NULL. Zustzlich zum Dialogbaum
         mu das Editfeld angegeben werden, auf dem der Cursor beim
         ffnen gesetzt werden soll. Existiert kein Editfeld, so it 0
         anzugeben.

   int do_mdial(MDIAL *mdial);
         Event-Schleife des Dialogs. Kehrt erst, wenn ein Exit-Objekt
         gedrckt wurde, mit dessen Index zurck.

   void close_mdial(MDIAL *mdial);
         Schliet und entfernt Dialog.
         Wenn einen Eventprozedur angemeldet ist, kehrt close_mdial()
         erst zurck, wenn alle nach dem Schlieen des Dialogs
         auflaufenden Message-Events verarbeitet wurden (untenliegende
         Fenster werden also erst noch neu gezeichnet).

   int simple_mdial(OBJECT *tree, int edit_start);
         Fhrt einen modalen Fensterdialog komplett durch.
         Ruft dazu open_mdial(), do_mdial() und close_mdial() nacheinander
         auf und liefert das Exit-Objekt zurck.

   void redraw_mdobj(MDIAL *mdial, int obj);
         Zeichnet ein Objekt im Dialog neu.

   void change_mdedit(MDIAL *wd, int new);
         Sollte man einmal aufrufen, nachdem man man den Status von Edit-
         feldern in offenen Dialogen gendert (z.B. DISABLED) hat, damit
         die Cursor-Verwaltung sich ein neues gltiges Editfeld sucht bzw.
         den Cursor korrekt versteckt.
```

1.18 CF-Lib: Menu

```
Funktionen fr die Menleiste                                        CF-Lib

Verwaltung die Menleiste der Applikation.
ACHTUNG: Nicht re-entrant, d.h. jede Applikation nur ein Kreuz, h Men!

   int create_menu(OBJECT *tree)
         Neues Men anlegen und anzeigen.
         Liefer 1, wenn alles OK.

   void delete_menu(void)
         Men wieder abmelden.

   void disable_menu(void)
   void enable_menu(void)
         Menzeile (About und alle Titel bis auf Desk) ab- bzw. an-
         schalten. Die Routinen zhlen mit, d.h. man mu genau so oft
         enablen, wie man vorher disabled hat!

   int is_menu_key(int kreturn, int kstate, int *title, int *item)
         Wenn es sich bei der Taste um einen Men-Shortcut handelt,
```

```
        wird 1 sonst 0 geliefert.
```

1.19 CF-Lib: Misc

```
Sonstiger Kleinkram                                                CF-Lib

   long ts2ol(short i1, short i2);
   void ol2ts(long l, short *i1, short *i2);
         "two shorts to one long"
         und
         "one long to two shorts"

   void save_background      (GRECT *box, MFDB *buffer);
   void restore_background   (GRECT *box, MFDB *buffer);
         Bildschirmhintergrund puffern.

   void *malloc_global(long size)
         Alloziert einen global freien Speicherbereich mittels Mxalloc()
         (Modus 35), wenn MiNT (Speicherschutz) luft, sonst mit Malloc().

   int get_patchlev(char *id_str, char *pl);
         Ermittelt aus einem Ident-String den Patchlevel. Jede Lib
         sollte einen Ident-String enthalten, der so aussieht:
            $PatchLevel: CF library: 7 $
         Liefert 1, wenn der PL ermittelt werden konnte.
```

1.20 CF-Lib: NKCC

```
Normalized Keycode Converter                                       CF-Lib

Die CF-Lib enthlt einige Funktionen aus der NKCC-Lib von Harald Siegmund.
Das sind im wesentlichen die beiden Funktionen, um Tastencodes zwischen
dem TOS- und dem normalisierten Format zu konvertieren.

   /* NKCC key code flags */
   #define NKF_FUNC     0x8000         /* function          */
   #define NKF_RESVD    0x4000         /* resvd, ignore it! */
   #define NKF_NUM      0x2000         /* numeric pad       */
   #define NKF_CAPS     0x1000         /* CapsLock          */
   #define NKF_ALT      0x0800         /* Alternate         */
   #define NKF_CTRL     0x0400         /* Control           */
   #define NKF_SHIFT    0x0300         /* any Shift key     */
   #define NKF_LSH      0x0200         /* left Shift key    */
   #define NKF_RSH      0x0100         /* right Shift key   */

   /* special key codes for keys performing a function */
   #define NK_INVALID   0x00           /* invalid key code  */
   #define NK_UP        0x01           /* cursor up         */
   #define NK_DOWN      0x02           /* cursor down       */
   #define NK_RIGHT     0x03           /* cursor right      */
   #define NK_LEFT      0x04           /* cursor left       */
   #define NK_M_PGUP    0x05           /* Mac: page up      */
```

```
#define NK_M_PGDOWN  0x06          /* Mac: page down    */
#define NK_M_END     0x07          /* Mac: end          */
#define NK_BS        0x08          /* Backspace         */
#define NK_TAB       0x09          /* Tab               */
#define NK_ENTER     0x0a          /* Enter             */
#define NK_INS       0x0b          /* Insert            */
#define NK_CLRHOME   0x0c          /* Clr/Home          */
#define NK_RET       0x0d          /* Return            */
#define NK_HELP      0x0e          /* Help              */
#define NK_UNDO      0x0f          /* Undo              */
#define NK_F1        0x10          /* function key #1   */
#define NK_F2        0x11          /* function key #2   */
#define NK_F3        0x12          /* function key #3   */
#define NK_F4        0x13          /* function key #4   */
#define NK_F5        0x14          /* function key #5   */
#define NK_F6        0x15          /* function key #6   */
#define NK_F7        0x16          /* function key #7   */
#define NK_F8        0x17          /* function key #8   */
#define NK_F9        0x18          /* function key #9   */
#define NK_F10       0x19          /* function key #10  */
#define NK_M_F11     0x1a          /* Mac: func key #11 */
#define NK_ESC       0x1b          /* Esc               */
#define NK_M_F12     0x1c          /* Mac: func key #12 */
#define NK_M_F14     0x1d          /* Mac: func key #14 */
#define NK_RVD1E     0x1e          /* reserved!         */
#define NK_DEL       0x1f          /* Delete            */


int nkc_init(void);
      Initialisiert die NKCC. Wird nur bentigt, wenn nkc_tos2n(),
      nkc_n2tos(), norm_to_gem() oder gem_to_norm() benutzt werden
      sollen.

unsigned short nkc_tos2n(long toskey);
long nkc_n2tos(unsigned short nkcode);
      BIOS-Taste (Bit 0..7 Ascii, Bit 16..23 Scan, Bit 24..31 Kbshift)
      ins normalisierte Format bzw. umgekehrt wandeln.

unsigned short gem_to_norm(int ks, int kr);
void norm_to_gem(unsigned int norm, int *ks, int *kr);
      Wandelt GEM-Taste (z.B. vom evnt_multi) in normalisierten Code
      um.


unsigned char nkc_toupper(unsigned char chr);
unsigned char nkc_tolower(unsigned char chr);
      Einzelnes Zeichen in Gro- bzw. Kleinbuchstaben wandeln.
      Nationale Sonderzeichen werden korrekt gewandelt.

void str_toupper(char *str);
void str_tolower(char *str);
      Zeichenkette wandeln.
```

1.21 CF-Lib: Objc

Objektmanipulation CF-Lib

```
int  get_obtype(OBJECT *tree, int obj, short *ud);
      Gibt den Objekttyp zurck. Bei umgewandelten Objekten
      (USERDEFs) wird jeweils der Basistyp zurckgegeben und
      <ud> auf 1 gesetzt (nur, falls <ud> != NULL).

long get_obspec(OBJECT *tree, int obj);
void set_obspec(OBJECT *tree, int obj, long spec)
      Abfragen/ndern der ob_spec.
      Bei umgewandelten Objekten (USERDEFs) wird der Original-
      wert zurckgegeben/gendert.

void set_string(OBJECT *tree, int obj, char *text);
void get_string(OBJECT *tree, int obj, char *text);
      Objekttext setzen/auslesen. Funkt fr G_BUTTON, G_STRING, G_TITLE,
      G_CICON, G_ICON, G_TEXT, G_BOXTEXT, G_FTEXT, G_FBOXTEXT.

void set_int(OBJECT *tree, int obj, int i);
int  get_int(OBJECT *tree, int obj);
void set_long(OBJECT *tree, int obj, long l);
long get_long(OBJECT *tree, int obj);
void set_ulong(OBJECT *tree, int obj, unsigned long l);
      Zahl in Text-Objekt setzen/lesen.

void set_state(OBJECT *tree, int obj, int state, int set);
int get_state(OBJECT *tree, int obj, int state);
      Objektstatus setzen (set = 1) oder lschen (set = 0)
      bzw. abfragen.

void tree_state(OBJECT *tree, int start_obj, int state, int set);
      Bei allen Elementen eines Sub-Baums den Status ndern
      (nicht rekursiv!!)

void set_flag(OBJECT *tree, int obj, int flag, int set);
int get_flag(OBJECT *tree, int obj, int flag);
      Objektflag setzen (set = 1) oder lschen (set = 0)
      bzw. abfragen.

int  find_flag(OBJECT *tree, int flag);
      Sucht das erste Object im Baum, bei dem <flag> gesetzt ist.

void get_objframe(OBJECT *tree, int obj, GRECT *r);
      Ermittelt die absoluten Ausmae eines Objekts. Beachtet Rahmen,
      3D und alle anderen Flags.

void redraw_obj(OBJECT *tree, int obj);
      Objekt neu zeichnen.
```

1.22 CF-Lib: Popup

Popupverwaltung CF-Lib

Neben in der Resource definierte Popups kann man mit den Funktionen auch

dynamisch zur Laufzeit Popups erzeugen.

```
typedef struct _popup
{
   OBJECT   *tree;      /* der Objektbaum */
   int      max_item;   /* maximal mgliche Anzahl */
   int      akt_item;   /* aktuelle Anzahl */
   int      item_len;   /* Lnge eines Eintrages */
} POPUP;
```

Kontrollstruktur fr die dynamischen Popups.

```
int create_popup(POPUP *p, int anz, int maxlen, char *item);
```

Neues Popup anlegen.

- p — die Popup Variable
- anz — die Anzahl der maximal mglichen Eintrge
- maxlen — maximale Lnge der Eintrge. Wird -1 bergeben, wird strlen(item) als maximale Lnge angesehen.
- item — der erste Eintrag, der die Lnge fr alle weiteren festlegt!!

Ein Rckgabewert von 0 zeigt an, da der Objekt-Speicher nicht angefordert werden konnte.

```
int free_popup(POPUP *p);
```

Gibt den Speicher wieder frei.

```
int append_popup(POPUP *p, char *item);
```

Eintrag am Ende anhngen.

- p — das Popup
- item — Eintrag, der am Ende angehngt werden soll. Wenn ein Eintrag aus '-' besteht, wird er 'disabled' angezeigt.

Ein Rckgabewert von 0 zeigt an, da der Objekt-Speicher nicht angefordert werden konnte.

```
int do_popup(POPUP *p, int button);
```

Popup auf den Bildschirm bringen und ausfhren. Es erscheint an der aktuellen Mausposition.

Rckgabe:

Der ausgewhlte Eintrag (beginnend bei 1) oder null.

```
int cf_menu_popup(MENU *m1, int x, int y, MENU *m2, int button, int off);
```

Kopie von menu_popup(), die enweder menu_popup() aufruft oder diesen Call emuliert.
Die Emulation untersttzt keine scrollende Mens!!
Zustzliche Parameter:
Falls <button> = 0 und das System menu_popup() hat, wird dieses benutzt. Ist <button> = 1/2 (linke/rechte Maustaste), wird das Popup selbst verwaltet. Es geht dann z.B. sofort auf (nicht wie bei menu_popup() erst beim Loslassen der Taste).
<off> gibt den Offset zwischen Root-Objekt (Rahmen) und dem ersten Popup-String an (i.d.R. ist offset 0, bei einem Farb-Popup aber z.B. nicht).

```
int handle_popup(OBJECT *dial, int dial_obj, OBJECT *pop, int pop_obj,
                 int mode);
```

Hi-Level Popupverwaltung fr Dialoge.

- dial — der Dialog, in dem das Popup angeklickt wurde
- dial_obj — Nummer des Objekts im Dialog

```
            pop      der Baum, in dem das Popup enthalten ist
            pop_obj  Nummer des Popups im Baum
            mode     Aktion, die ausgefhrt werden soll:
                     POP_OPEN (1)   Popup ffnen
                     POP_CYCLE (2)  Nchsten Eintrag selektieren

         Infos zur Popup-Gestaltung in der RSC gibts .
```

1.23 CF-Lib: Scrap

```
Scrap                                                              CF-Lib

Simple Klemmbrettverwaltung (alle Aktionen auf scrap.txt).

   int get_scrapdir(char *scrap);
   void scrap_clear(void);
   char *scrap_rtxt(char *buf, long *len, long maxlen);
   void scrap_wtxt(char *buf);
```

1.24 CF-Lib: Userdef

```
Userdef-Verwaltung                                                 CF-LIB

Die CF-Lib emuliert die erweiterten Objekt-Typen von MagiC. Emuliert werden
Radio- und Checkbuttons, Gruppenrahmen, berschift sowie Shortscuts zur
Tastaturauslsung der Objekte.
Zustzlich zu den MagiC-Objekten gibt es noch Gruppenrahmen und berschrift
mit kleinem Systemfont. Die kleine Schrift wird verwendet, wenn zu den
normalen MagiC-Flags noch der Status CHECKED aktiviert wird.

   void fix_dial(OBJECT *tree);
         Wandelt die Objekte eines Baumes ggf. in USERDEFs um.

   void fix_menu(OBJECT *tree);
         Ersetzt die '--' in Mens durch echte Linie. Kann auch fr die
         Trenner in Popups benutzt werden.

   void fix_popup(OBJECT *tree, int thin_line);
         Alternative zu fix_menu().
         Unter MagiC werden die Eintrge in G_SHORTCUT gewandelt.
         Auerdem ist es mglich, eine dnnere Linie als Trennlinien,
         als sie bei normalen Mens zum Einsatz kommen, zu erhalten.

ACHTUNG: Wenn ein Programm neben den von der CF-Lib verwalteten auch
         noch eigene USERDEFs verwendet, funktioniert get_obtype() fr
         diese nicht unbedingt korrekt!
         get_obtype() geht davon aus, da der alte Objekttyp im oberen
         Byte von ob_type liegt (im unteren liegt G_USERDEF). Ist dem
         nicht so, liefert get_obtype() falsche Werte zurck und alle
         Teile der CF-Lib, die auf diese Funktion aufbauen, werden sich
         nicht korrekt verhalten!
```

1.25 CF-Lib: Vdi

```
Ntzliches rumd um das VDI                                          CF-Lib

   int open_vwork(int *w_out);
         Virtuelle WS ffnen. bergeben werden mu das 57 int groe
         work_out Array.

   void set_clipping(int handle, int x, int y, int w, int h, int on);
         Clipping an- bzw. ausschalten.

   int height2pts(int handle, int f_id, int f_height);
         Ermittelt aus einer Pixel-Hhe die zugehrige Point-Hhe.
```

1.26 CF-Lib: Wdial

```
Fensterdialoge                                                     CF-Lib

Mit diesen Funkionen sind unmodale Fensterdialge mglich.
Die geffneten Dialoge laufen parallel zu den anderen Fenstern der
Applikation.

   #define WD_CLOSER
      exit_obj fr Closer, fall kein UNDO-Obj vorhanden ist.

   #define WD_OPEN
   #define WD_ICON
   #define WD_SHADE
      Zustand des Fenster (wd->mode).

   typedef void (* WDIAL_OCB )(WDIALOG *dial);
   typedef int  (* WDIAL_XCB )(WDIALOG *dial, int exit_obj);

   #define WOCB_NULL
   #define WXCB_NULL
         NULL-Pointer fr den Fall, da man kein Callback einklinken
         will.

   typedef struct _wdial
   {
      struct _wdial  *next;

      OBJECT      *tree;           /* Objektbaum */
      OBJECT      *icon;           /* Icon fr Iconify */
      int         mode;            /* aktueller Status */
      int         win_handle;      /* Fensterhandle */
      char        win_name[80];    /* Fenstertitel */
      int         win_kind;        /* Elemente */
      GRECT       work;            /* Fenstergre */
      int         title_obj;       /* Objektnummer des Titelobjektes */
      int         cancel_obj;      /*       "      des Abbruchbuttons */
      int         delta_y;         /* Offset bis zum Titelobjekt */
      int         edit_idx,        /* Objektnummern fr die Editfelder */
                  next_obj,
```

```
                edit_obj;
   WDIAL_OCB    open_cb;
   WDIAL_XCB    exit_cb;
} WDIALOG;


WDIALOG *create_wdial(OBJECT *tree, OBJECT *icon, int edit_obj,
                              WDIAL_OCB open_cb, WDIAL_XCB exit_cb);
      Neuen Fensterdialog anlegen. Dazu wird der Objektbaum, der
      Baum fr das Icon-Fenster, das erste Editfeld (oder 0) sowie
      zwei Callbacks (oder NULL) bergeben.
      <open_cb> wird aufgerufen, bevor das Fenster geffnet wird.
      <exit_cb> wird fr Exit-Objekte aufgerufen.
      Der Dialog wird geschlossen (aber nicht gelscht), sobald der
      Exit-Callback TRUE zurckgeliefert.
      Wurde der Closer bettigt, wird der Exit-Callback mit dem
      Index des UNDO-Buttons benutzt. Ist kein UNDO-Button vorhanden,
      wird WD_CLOSER (s.o.) geliefert.
      Der Exit-Button wird beim Close deselektiert!

void delete_wdial(WDIALOG *wd);
      Fensterdialog wieder abmelden.

void open_wdial(WDIALOG *wd, int pos_x, int pos_y);
      Dialog an der bergebenen Position geffnet. Bei (-1,-1) wird
      er zentriert.

void close_wdial(WDIALOG *wd);
      Schliet das Fenster.

void redraw_wdobj(WDIALOG *wd, int obj);
      Zeichnet Objekt neu.

void redraw_wdicon(WDIALOG *wd, int obj);
      Zeichnet den iconifizierten Dialog neu.

void change_wdedit(WDIALOG *wd, int new);
      Sollte man einmal aufrufen, nachdem man man den Status von Edit-
      feldern in offenen Dialogen gendert (z.B. DISABLED) hat, damit
      die Cursor-Verwaltung sich ein neues gltiges Editfeld sucht bzw.
      den Cursor korrekt versteckt.

int message_wdial(int *msg);
int click_wdial(int clicks, int x, int y, int kshift, int mbutton);
int key_wdial(int kreturn, int kstate);
      Diese drei Funktionen men von der der Haupt-Eventschleife der
      Applikation bei den entsprechenden Events aufgerufen werden.
      Wurde ein Event verarbeitet, wird 1 zurckgeliefert, sonst 0.
```

1.27 Tips zur RSC-Gestaltung

```
Tips zur RSC-Gestaltung                                          CF-Lib

   Erweiterte Objekttypen von MagiC
```

```
   Aufbau einiger besonderer RSC-Elemente

   Sondertasten-Belegung

 Fr Lese-Faule:
   Im ExtObjFix.prg fr InterFace ist eine Kurzhilfe vorhanden, in
   der die wichtigsten Objekttypen aufgelistet sind.
   An diese Hilfe gelangt man, wenn man in den InterFace-Objekt-
   editoren den entsprechenden Button bzw. die HELP-Taste auslst!
```

1.28 Erweiterte Objekttypen von MagiC

```
Erweiterte Objekttypen von MagiC                                    CF-Lib
MagiC bietet neben den standard GEM-Objekten einige Erweiterungen.
Dabei handelt es sich um jeweils einen Standardtyp mit einem besonderen
Status bzw. Flag.
Falls das Objekt einen Shortcut erlaubt, wird dessen Position ber eine
4bit Zahl in den Status-Bits 8 - 11 festgelegt.

   Titel (unterstrichener Text)
         Basistyp:      G_STRING
         Status:        WHITEBACK sowie Bits 8 - 15
         Sonstiges:     Es wird empfolen, die Objekthhe um zwei bis
                        drei Pixel zu vergern.

   Gruppenrahmen
         Basistyp:      G_BUTTON
         Status:        WHITEBACK sowie Bits 9 - 15
         Sonstiges:     CF-Lib-Erweiterung: Bei Status CHECKED wird
                        der Text im kleinen Systemfont gesetzt.

   EXIT-Button
         Basistyp:      G_BUTTON
         Status:        EXIT, 3D-Activator
                        mit Shortcut: WHITEBACK, Position ber Bits 8 - 11.

   Kreuz-Button
         Basistyp:      G_BUTTON
         Status:        WHITEBACK, 3D-Background
                        mit Shortcut: Bit 15, Position ber Bits 8 - 11.
                        ohne Shortcut: Bits 8 - 15

   Radio-Button
         Basistyp:      G_BUTTON
         Status:        RADIOBUTTON, WHITEBACK, 3D-Background
                        mit Shortcut: Bit 15, Position ber Bits 8 - 11.
                        ohne Shortcut: Bits 8 - 15

   tast. String
         Basistyp:      G_STRING
         Status:        WHITEBACK, Shortcut-Position ber Bits 8 - 11.

   Edit-Objekt
         Basistyp:      G_FTEXT
         Status:        EDITABLE, 3D-Background
```

```
         Sonstiges:     Rand auen 2 Pixel
                        Wird ab 16 Farben eingesunken, in Monochrom
                        umrahmt gezeichnet.
                        Durch den Rand sollte zwischen zwei ber-
                        einanderliegenden Edit-Objekten 1,5 Zeilen
                        Platz gelassen werden, da sonst die Rahmen
                        kollidieren.
```

1.29 spezielle rsc-elemente

```
Aufbau spezieller RSC-Elemente                                  CF-Lib


Popups
Die Routine handle_popup() setzt einen der beiden folgenden Popup-
Strukturen im Dialog voraus:

               String: [Popup-Box]
     oder
               String: [z.B. Editfeld] [?]

Bei [Popup-Box] handelt es sich um einen BOXTEXT (SHADOWED,
TOUCHEXIT, 3D-Background), der die Lnger der Strings im Popup plus
einem Blank vorne und hinten haben sollte. handle_popup() liet den
Text aus diesem Objekt aus und positioniert das Popup entsprechend.
Nach Auswahl eines Eintrags wird dessen Text automatisch in die Popup-
Box eingetragen.

Bei [?] handelt es sich um ein BOXCHAR (SHADOWED, TOUCHEXIT, 3D-
Background), an dem das Popup rechtsbndig nach unten aufgeht. Da die
CF-Lib nicht wissen kann, in welches Objekt der Text des ausgewhlten
Eintrags geschrieben werden soll, mu man das bei diesem Popup-Typ
selbst erledigen!

Bei <String> knnte es sich um ein tastaturbedienbaren STRING handeln,
ber den das Popup per Tastatur gesteuert wird. Die Dialog-Routinen
mte dann handle_popup() mit Modus POP_CYCLE aufrufen, um den nchsten
Eintrag einzustellen. Auch diese Funktion funktioniert nur mit dem ersten
Popup-Typ, da nur dort die CF-Lib durch den Text in der Popup-Box wei,
welches der nchste Eintrag ist.

Das eigentliche Popup sollte aus einer BOX (SHADOWED, 3D-Background)
bestehen, in der die Eintrge als STRING (SELECTABEL, !DISABLED)
realisiert werden. Trennlinien knnen durch disabled'te '--'-Eintrge
erzeugt werden. Damit die '--' durch eine richtige Linie ersetzt wird,
mu fr das Popup dial_fix() aufgerufen werden.
```

1.30 sondertasten-belegung

```
Sondertasten in Dialogen                                        CF-Lib

Einige Sondertasten sind in den Dialogen der CF-Lib mit Funktionen
```

belegt:

- Ein Exit-Button bei dem das Flag Bit 11 gesetzt ist, kann mit UNDO ausgelst werden.

- Ein Exit-Button bei dem das Flag Bit 12 gesetzt ist, kann mit HELP ausgelst werden.

- In Alertboxen knnen die Buttons mit F1 bis F3 ausgelst werden (wie in MagiC).

- Die INSERT-Taste innerhalb eines Editfelds ffnet die ASCII-Tabelle.

- ber ^X, ^C und ^V kann der Text vom GEM-Klemmbrett in Editfeldern verarbeitet werden.

- Mit TAB bzw. Shift-TAB kann zum nchsten bzw. vorherigen Editfeld gesprungen werden. Vom Letzten wird wieder zum Ersten gesprungen und umgekehrt.

1.31 CF-Lib: Programmliste

Programmliste CF-Lib

Hier eine Liste der Programme, die die CF-Lib benutzen:

Programm	Autor	Programm-Version
gemgs	CF	ab 1.0
GEM-Setup	Joachim Fornallaz	ab 1.20
HD-Driver	Uwe Seimet	ab 7.70
qed	CF	ab 4.5
TosWin2	CF	ab 2.2