Technisches Handbuch

Inhalt

Anhang C: Betreuerhandbuch

Installation und Platzbedarf des Programms

Das Programm ist auf dem Rechner im Unterverzeichnis PRAKTI.KUM installiert. Die benötigten Dateien befinden sich alle dort.

Das Programm benötigt 512 kByte freien Speicher und ebensoviel EMS-Speicher, sowie eine Maus (mit installiertem Treiber).

Das Programm wird direkt aus der AUTOEXEC.BAT gestartet. Dabei werden einige Kommandozeilenparameter übergeben, die unten erklärt werden.

Benutzerberechtigungen

Nach dem Start des Programms muß eine Benutzeridentifikation erfolge. Dazu wird der Gruppenname und eine Abkürzung erfragt. Die Abkürzung kennzeichnet die Dateien, die diese Gruppe erstellt. Wenn man das Programm wieder startet und eine andere Gruppenabkürzung eingibt, so sind die alten Dateien nicht mehr zugänglich. Gibt man allerdings dieselbe Abkürzung ein, so werden die Dateien wieder im Inhaltsverzeichnis des Programms angezeigt.

Gibt man als Gruppenabkürzung das Paßwort für den Praktikumsbetreuer ein (zur Zeit "Ossau *93*", so erhält man bestimmte Vorrechte: Das Programm startet ohne die Abfrage ("Lernen/Anleitung/nichts"), es steht sofort das vollständige Menü zur Verfügung, man kann alle Dateien bearbeiten (und löschen) und kann das Programm einfach beenden.

Für den Betreuer des Physikpraktikums kann es sinnvoll sein, das Programm zu verlassen. Der Befehl "Datei/Beenden" steht nur im vollständigen Menü (siehe Anleitung) zur Verfügung. Hat man beim Programmstart die Betreuerberechtigungen bekommen, so kann man den Befehl einfach aufrufen und er wird ausgeführt. Läuft das Programm für Praktikanten, so ist vor der Ausführung des Befehls die Eingabe des obigen Paßwortes notwendig.

Die Kommandozeilenparameter

Es gibt folgende Parameter, die alle optional sind und in beliebiger Reihenfolge stehen können:

- MASTER startet ohne Abfrage in den Betreuermodus; empfehlenswert für die Entwicklungsumgebung von TurboPascal.
- NOEXIT verhindert das Verlassen des Programms für Praktikanten. Hat man die Betreuerberechtigungen, so ist dieser Parameter wirkungslos
- NOSOUND verhindert, daß die Soundkarte angesprochen wird und, bei ihrem Fehlen, die entsprechenden Fehlermeldungen.
- NOMESS entsprechend für die Messkarte. Damit sind aber auch keine Messungen möglich. Sinnvoll z.B. bei einer Änderung des Programms auf einem anderen Rechner.
- SVGA benutzt dann den 800*600-Modus einiger SVGA-Karte mit 256 Farben (ohne diese auszunutzen, die Grafik ist allerdings schneller). Dies verbessert u.a. die Qualität des Ausdrucks (das dieser eine Bildschirmkopie ist). Nur bei 15-Zoll-Monitor und z.B. ET4000-Grafikkarte zu empfehlen. Die Bildwiederholfrequenz sollte bei 70Hz. liegen (leider nicht selbstverständlich).

Compilerbefehl und Dokumentationshilfe

Beim Kompilieren der Quelldateien kann bei der Compileroption "Conditional defines" der Parameter `extras` angegeben werden. Dann stehen mit den Tasten F8-F10 zusätzliche Möglichkeiten zur Verfügung.

F8 fragt nach einer Beschreibung des aktuellen Bildschirminhalts. Diese wird in einem mit der Maus positionierbaren Fenster auf den Bildschirm geschrieben. An der gewünschten Stelle drückt man BEIDE Maustasten, der Rechner piept dann kurz. Jetzt werden die Mausroutinen nicht benutzt, und man kann ohne großes Absturzrisiko den Bildschirminhalt sichern, z.B. mit BPL-CAPT.EXE. Der nächste Tatendruck (ESC) rekonstruiert der ursprünglichen Bildschirminhalt. Mit dieser Methode wurde die Bildershow des Lernprogramms erstellt.

F9 und F10 setzen den Bildschirm auf schwarz-weiß um. Das dauert etwas, da die Routinen alles andere als optimiert sind. Benötigt wurde dies nur, um die Menübefehle etc. als Schwarzweißgrafiken in die Anleitung übernehmen zu können.

Anhang D: Programmbeschreibung

Zur Schriftsetzung (Typographie):

Es werden im folgenden verschiedene Schriften benutzt:

Für Programmbefehle, Listings etc. wird `CourierNew` verwendet, da es eine schreibmaschinenähnliche Schrift ist und dem Aussehen von Listings in Zeitschriften und in den Handbüchern von Borland gleicht. Variablen werden in *`Courier kursiv`* angegeben, um sie von den Befehlen abzusetzen. Die Namen von Units und anderen Dateien sind generell in GROßSCHRIFT und Objekttypendefinitionen in Normalschrift (da am Namen ..._Button erkennbar) angegeben.

D.1. Allgemeine Funktion

Das Programm basiert grundsätzlich auf einer Benutzeroberfläche, die mit Prozeduren 'zum Leben erweckt' wird. Das soll heißen, die Benutzeroberfläche ist für sich funktionsfähig ohne eine Funktion durchzuführen. Sie besteht hauptsächlich aus Menüs und aktivierbaren Feldern (genannt Buttons [Knöpfe, da man 'draufdrücken' kann]), aus denen übrigens auch die Menüs intern aufgebaut sind. Diese werden komplett im Rahmen der GUI-Units (GraphicUserInterface) objektorientiert verwaltet. Die dazugehörige Unit GUI_DEF.PAS stellt die benötigten Objekttypen zur Verfügung. Die anderen Units müssen nun die Objekte anlegen und initialisieren sowie mindestens ein Menüobjekt aktivieren, das dann die eigentliche Verwaltung übernimmt, doch dazu später (→1.1.).

Die weiteren Units haben Aufgaben, die sich in ihren Namen wiederspiegeln (sollen):

- F_MAIN ist keine Unit sondern das eigentliche Programm und ruft alle anderen Units (bzw. deren Initialisierungen) auf.
- F_MENU baut das Hauptmenü auf und verwaltet dessen Umschalten sowie die Benutzerkontrolle (Name/Identifikation).
- F_MOUSE verwaltet die Mausabfragen und Darstellung verschiedener Mauszeiger.
- F_HELP gibt Hilfstexte, die in der Datei HELPDAT.PAS stehen, aus.
- F_GRAPH stellt die Graphikschnittstelle dar, um Graphen verschieden darzustellen, wobei primär die Objekte aus F_GR_DEF benutzt werden.
- F_GR_DEF enthält die Objektdefinitionen und Konstanten, die allgemein zur Graphendarstellung benutzt werden.
- F_CALC führt die Berechnungen zwischen Kurve und Spektrum durch.
- F_CHANGE dient dem Verändern der Daten sowie deren Auswahl des Arbeitsbereichs nach Messung / Laden
- F_DATEI führt Laden und Speichern sowie Löschen der Daten auf Festplatte durch.
- F_MESS führt die Messungen mit der M42-Karte durch
- F_SOUND ermöglicht Ein/Ausgaben mit einer Soundblaster- (kompatiblen)-Karte
- F_TEXT dient nur der Darstellung von Text und ersetzt, außer bei EGA-Grafik, Borlands OuttextXY.
- F_GLOBAL enthält die globalen Variablen, außer den in GUI_DEF definierten.

D.2. Der Aufbau des Programms

Es ist für das Verständnis von Vorteil, zuerst den Ablauf des Programms zu betrachten:

Der Ablauf des Programms wird von F_MAIN gesteuert. Es läßt F_MENU die Struktur des Hauptmenüs (*`main_menu`*) mit seinen Untermenüs (*`sub_menu`*[..]) aufbauen. Benutzt wird dabei der Objekttyp Menu_Button, dem Unterobjekte zugeordnet werden. Außerdem wird ein adäquates Umfeld für die einzelnen Programmteile geschaffen, also auf Graphik umgeschaltet und eine Initialisierung der Programmteile in der benötigten Reihenfolge durchgeführt. Man könnte sagen, die Units werden eingeschaltet.

Nun gibt das Hauptprogramm die Kontrolle an die Methode (Funktion) *`main_menu`*`.direct_call` ab. Das ist also die Methode `direct_call` des Objekts *`main_menu`*. Von dieser aus wird das ganze Programm gesteuert. Wenn etwas geschehen soll, so muß der Anwender ein Objekt (vom Prototyp Button), innerhalb der Menüstruktur die abgebildet wird, auswählen. Dies geschieht durch Anklicken oder Eingabe des "Hotkeys". Die Objekte können Befehle sein die evtl. wieder Menüs sind (also z.B. die Titel in der Menüleiste mit dazugehörigen herunterfallenden Befehlslisten) oder Bereiche auf dem Bildschirm (z.B. die Fenster in denen Daten graphisch dargestellt werden) sein. Jedes Anklicken eines Objekts hat den Aufruf der Methode `handle_klick` dieses Objekts zur Folge.

Nun muß das Objekt selber entscheiden, wie es darauf reagiert (was natürlich durch den Typ des Objekts vorgegeben ist, aber mit `handle_klick` erscheinen alle Objekte nur als Button-Objekte; das ist der Sinn und Vorteil der objektorientierten Programmierung): Ein Submenü (ein Menu_Button) stellt sich dar (also werden die Befehlslisten 'heruntergeklappt'), ein Befehl (Order_Button) ruft eine Routine auf, deren Adresse er beim Initialisieren erhielt und ein Fenster (vom Objekttyp Graph_Button) ruft ebenfalls eine Routine auf, die aber auch nach der Initialisierung geändert werden kann. Die verschiedenen Darstellungen in den Fenstern werden durch jeweils verschiedene Graph_Buttons realisiert. Diese können (wie alle Buttons) abgeschaltet werden, so daß sie auf Aufrufe durch den Menüverwalter nicht reagieren und unsichtbar sind. Dadurch kann die eigentliche Menüstruktur statisch bleiben, was eine Vereinfachung in der Programmierung und eine Erhöhung der Zuverlässigkeit des Programms bedeutet.

Alle Funktionen des Programms werden nun durch Aufrufe aus `handle_klick` aufgerufen. Für spezielle Zwecke werden dann neue Menüs aufgebaut, bearbeitet und dann wieder entfernt, z.B. in den Auswahlboxen beim Laden oder Löschen von Dateien. Aber immer wird die Kontrolle nach Abschluß der Arbeit an den Menüverwalter zurückgegeben. Dieser kehrt erst dann zurück (er wurde ja auch aufgerufen) wenn er das Flag *`dont_exit`* nicht mehr gesetzt hat. Alle Menüs haben es erst einmal nicht gesetzt, d.h. sie kehren nach der ersten ausgewählten Funktion und deren Abarbeitung zurück. Das Hauptmenü soll dies aber nicht. Also wird das Flag explizit gesetzt und erst von der Funktion `quit` zurückgesetzt! Das mache übrigens andere Menüs auch, die mehrere Funktionen aufrufen sollen, z.B. die Auswahl des Arbeitsbereichs nach einer Messung. Dort werden von einem neuen Menü 2 Order_buttons (Ende/Hilfe) und ein Graph_Button verwaltet. Hierbei ist übrigens das mess_window in zwei Menüstrukturen (`main_menu` und `select_menu`) gleichzeitig eingebunden, wovon natürlich nur eine aktiv ist; Zeiger machen dies möglich...

Die Units

Die Units teilen sich die Arbeit, greifen aber duch globale Variablen und Aufrufe stark ineinander ein. Die globalen Variablen sind möglichst in F_GLOBAL.PAS untergebracht. Nur die globalen Variablen, die die GUI-Reihe hauptsächlich für eigene Zwecke bereitstellt, sind getrennt in GUI_DEF.PAS, da GUI... getrennt vom eigentlichen Programm laufen soll.

D.3. Die Unit GUI_DEF.PAS

Das Arbeitsprinzip und Aufgaben

Die Unit GUI_DEF ist einer der GUI-Units. Diese sollen das GraphicUserInterface, also die graphische Schnittstelle zwischen Benutzer und Programm, zur Verfügung stellen. Die Aufgabe dieser Unit ist die Bereitstellung geeigneter allgemeiner Objekte, die auf dem Bildschirm erscheinen und auf Maus und Tastatur reagieren können.

Prototyping

Die vorhanden Objekte sind einfache Grundtypen, die immer wieder verwendet werden. Da ist zum einen der Objekttyp Button. Seine Bedeutung liegt darin, daß er der Prototyp für die weiteren Objekttypen ist. Alle weiteren Typen werden von ihm abgeleitet. Schon er hat alle allgemeinen Fähigkeiten (d.h. Methoden), die später benötigt werden. Das wären: `init`, `test_n_handle_touch`, `handle_klick` und weitere. `init` ist ein Constructor, der Dinge wie Koordinaten, Farbe(n), Nummer eines Hilfstextes sowie den Text für die Statuszeile übergeben bekommt und dann festhält. `test_n_handle_touch` meldet nach Übergabe der Mauskoordinaten, ob das Objekt berührt wurde und reagiert evtl. darauf.

`handle_klick` übernimmt die Arbeit wenn das Objekt angeklickt wurde. Ob es getroffen wurde, muß aber vorher mit `test_n_handle_touch` oder `test_hotkey` geprüft werden. `handle_klick` wird von Button noch nicht ausgeführt, es wird einfach zurückgekehrt. Der Vorteil liegt aber darin, daß andere Objekte beliebige Methoden hierfür definieren können (Virtuelles Überladen), und sie von vornherein vorgesehen sind. Wie auch weitere Methoden wie `select`, `unselect` und insbesondere `display` sind alle Methoden virtuell angelegt, um sie nach Bedarf überladen zu können. Z.B. arbeitet die Methode `select` indem sie u.a. `display` aufruft. Wenn ein neuer Typ keine eigene `select`-Methode definiert, wird die des Vorgängers übernommen. Soll diese aber die neu definierte `display`-Methode nutzen, so muß diese virtuell sein. Nur so wird immer die aktuell gültige Methode benutzt!

Kinder des Prototyps

Das nächste Objekt, Text_Button, ist auch noch ein Prototyp, der aber selbst von Button abgeleitet wurde. Er enthält nur zusätzlich einen Text der linksbündig und in halber Höhe des Objekts dargestellt wird. Er kann nun auch beim Berühren (mittels `test_n_handle_touch` sowie `select` bzw. `unselect`) invertiert und wieder normal dargestellt werden.

Order_Button soll nun ein Text_Button sein, der bei `handle_klick` eine Prozedur aufruft. Diese wird einfach in einem Zeiger gespeichert, der durch `init` gesetzt wurde (Stichwort Prozedurvariable).

Menu_Button ist auch von Text_Button abgeleitet, kann aber viel mehr. Er ist mit zusätzlichen Methoden ausgestattet, die es ermöglichen ihm Zeiger auf Buttons zu übergeben, die in einer Liste verwaltet werden. Wenn nun `handle_klick` aufgerufen wird[1], so werden alle Buttons in der Liste dargestellt (mit `show`) und verwaltet (Schleife mit Mauskoordinatenabfragen und alle Buttons auf Hotkey/Berührung/Anklicken testen). Dies wird erst abgebrochen, wenn ein Button ausgewählt oder die rechte Maustaste bzw. die ESC-Taste (beide für Abbruch) betätigt wurde. Wenn einer der Buttons vom Typ Menu_Button ist, so wird also ein Submenü aufgerufen und gehandhabt, bis irgendwann ein Abbruch oder eine Befehlsauswahl durch einen Order_Button (und dessen `handle_klick`) erfolgt. Danach werden gewöhnlich (genauer: wenn *`dont_exit`* nicht gesetzt ist) die Buttons wieder gelöscht, der gewählte Befehl wird ausgeführt und es wird von `handle_klick` zurückgekehrt.

Die Methoden der Unit GUI_DEF

`CONSTRUCTOR Button.init`
Bekommt die Daten übergeben die der Button immer braucht: Koordinaten, Farben, Hilfstextnummer...

`DESTRUCTOR Button.done;`
Reiner Dummy um überladen zu werden.

`PROCEDURE Button.show;`
Stellt Button dar (mittels display)

`PROCEDURE Button.select;`
Hiermit erfährt der Button, daß er selektiert (berührt) wurde.

`PROCEDURE Button.unselect;`
Entsprechend...

`PROCEDURE Button.display;`
Stellt Button als Rechteck dar.

`PROCEDURE Button.hide;`
Überladbare, potentielle Routine zum Löschen des Button. (Z.B. ab Text_Button durch Zurückschreiben des Hintergrundes, bei GraphButton mit XOR-Überschreiben des Graphen)

`FUNCTION Button.test_hot_key ( ch: CHAR):BOOLEAN;`
Meldet mit TRUE, wenn der übergebene Char der HotKey war und der Button aktiv ist (beachtet also *`enabled`*-Flag).

`FUNCTION Button.test_n_handle_touch( mx, my: WORD): BOOLEAN ;`
Meldet mit TRUE wenn die übergebenen Maus/Bildschirm-Koordinaten im Button liegen und der Button aktiv ist (mittels *`enabled`*-Flag). Ruft, wenn ja, select auf und stellt mit `show_text` (➔ gui) seinen *`touch_text`* dar.

---

[1] Das kann über `direct_call` oder durch den Aufruf aus einem Menü erfolgen.

`PROCEDURE Button.enable;`
`PROCEDURE Button.disable;`
Diese beiden Methoden schalten den Button an/aus. Dazu wird er dargestellt/entfernt (wenn nötig, dies wird in shown nachgeschaut) und der entsprechende Zustand in *`enabled`* vermerkt. Nun reagiert der Button nicht mehr auf `test_n_handle_klick` und `test_hot_key`.

`PROCEDURE Button.force_enabled_flag(value : BOOLEAN);`
Setzt das enabled-Flag auf einen beliebigen Wert. Da die Bildschirmverwaltung dadurch getäuscht wird, ist der Befehl nur mit Vorsicht zu genießen. Er wird nur benutzt um bei unnötigen Löschaktionen des Meßfensters Abgeschaltetsein vorzutäuschen.

`FUNCTION Button.handle_klick( mx, my, nr: WORD):BOOLEAN;`
Aufforderung an den Button auf ein Klicken bzw. HotKey zu reagieren, `test_n_handle_touch` muß vorangegangen sein um Ort zu überprüfen. Der Button fühlt sich also bei dieser Routine immer gemeint. Gibt i.A. immer TRUE zurück,

`PROCEDURE Button.set_selectable( sable: BOOLEAN);`
Ein dargestellter Button kann trotzdem `test_n_handle_klick` ignorieren und `select/unselect` abwehren, wenn *`selectable`* auf FALSE gesetzt ist. Dies wird in Menüs für die Zwischenstriche benutzt und kann hiermit gesetzt werden.

`PROCEDURE Button.identity:Object_Type;`
Liefert immer Button zurück. Wird bei Menu_Button überladen um Menüs hintereinander darstellen zu können und erst nach der letzten Auswahl alle forderen zu löschen.

`CONSTRUCTOR Text_Button.init;`
übergibt die meisten Werte an Button.init, bekommt zusätzlich den anzuzeigenden Text.

`PROCEDURE Text_Button.show;`
Stellt den Button mitsamt dem bei `init` übergebenen Text dar (durch Aufruf von `display`). Vorher wird der Hintergrund in einem dynamisch allokierten Bereich abgelegt. Ein Fehler bei `GetMem` führt zu einem NIL-Pointer und keiner Fehlermeldung. (Bedeutung: Wenn Text_Button illegal groß, so daß reservierter Speicher > 64kByte, aber evtl. übernimmt Aufrufer die Hintergrundverwaltung...z.B. bei create_select als Rahmen!)

`PROCEDURE Text_Button.hide;`
Der Hintergrund, der vor `show` zu sehen war, wird rekonstruiert, der bei `show` reservierte Bereich wird wieder freigegeben. Wenn der Pointer auf den Bereich NIL ist, wird nichts gemacht.

`PROCEDURE Text_Button.display;`
Stellt je nach Zustand des *`selected`*-Flags den Text und den Hintergrund in den zwei bei `init` übergebenen Farben dar.

`CONSTRUCTOR Order_Button.init`
wie Text_Button.init, bekommt noch Zeiger auf Routine, die bei Anklicken auszuführen ist.

`FUNCTION Order_Button.handle_klick( mx, my, nr: WORD): BOOLEAN;`
Ruft eine Funktion auf, deren Adresse als Prozedurvariable bei `init` übergeben wurde.

`CONSTRUCTOR Menu_Button.init`
wie Order_Button.init

`DESTRUCTOR Menu_Button.done;`
Löscht den Button selbst und, je nach *`all_dyna`*-Flag, alle untergeordneten Objekte durch deren Destructor `done`.

`PROCEDURE Menu_Button.add_Button(bptr:Button_PTR;sable:BOOLEAN)`
Es wird in der Liste ein neuer Button hinzugefügt mit der Eigenschaft sable, die angibt ob er selektierbar sein soll (es wird intern `set_selectable` des Buttons aufgerufen).

`FUNCTION Menu_Button.handle_klick( mx, my, nr: WORD): BOOLEAN;`
Ein Aufruf dieser Methode bedeutet meist, daß die Darstellung des Menüs (als Text_Button) angeklickt wurde. Nun wird ein neues Menü dargestellt, indem alle Buttons in der Liste (die durch Aufrufe von `add_button` entstand) mit `show` zur Darstellung aufgefordert werden (weigern werden sich dabei alle, bei denen `disable` aufgerufen wurde). Dann folgt eine Schleife mit Mausabfrage, Keyboard-Abfrage, dann `test_hot_key` bei allen Objekten, `text_n_handle_touch` bei allen, bis einer der Buttons das Angebot an Mausereignissen/Tastenereignissen akzeptiert und TRUE meldet. Dann werden alle Buttons entfernt (mittels `hide`) und dieser Button mittels `handle_klick` aktiviert. Ausnahme: Wenn der neue Button mittels `identity` meldet, daß er menu_button ist, so wird erst das neue Menu (und evtl. weitere) dargestellt, dann (nach endgültiger Auswahl eines Befehls) alle gelöscht und dann der Befehl ausgeführt.

`PROCEDURE Menu_Button.set_leave(flag: BOOLEAN);`
Hiermit kann das in 2.1.1. beschriebene *`dont_leave`*-Flag bearbeitet werden.

`FUNCTION Menu_Button.identity:Object_Type;`
liefert `menu` zurück; siehe `Menu_Button.handle_klick`

`PROCEDURE Menu_Button.direct_call;`
dient dem direkten Aufruf eines Menu_Buttons. Der Aufruf mittels `handle_klick` scheitert, da die Menüverwaltung die Adresse des Befehls nur zurückliefert, und er dann von `direct_call` erst aufgerufen wird.

D.4. Die Unit GUI

`PROCEDURE init_ems;`
Die EMS-Verwaltung wird aufgebaut, Mit ihrer Hilfe werden später Bildschirmhintergründe abgespeichert ohne den Heap zu belasten. Dies ist gerade bei höheren Bildschirm- und Farbauflösungen wegen des hohen Datenaufkommens unverzichtbar. Bei einem Fehler wird das Programm beendet.

`PROCEDURE gui_exit;`
Wird bei Verlassen des Programms automatisch aufgerufen (augenblicklich leer).

`PROCEDURE init_gui;`
Die notwendigen Initialisierungen wie `init_ems`.

`PROCEDURE hide_status;`
Löscht die Statuszeile.

`PROCEDURE show_status(status: STRING);`
Stellt den übergebenen Text in der (halben) Statuszeile dar.

`PROCEDURE hide_info_line;`

`PROCEDURE show_info_line(info: STRING);`
Zwei entsprechende Funktionen für die Informationszeile.

`FUNCTION save_big_screen(x0,y0,dx,dy: WORD): INTEGER;`
Der Bildschirminhalt innerhalb der angegebenen Koordinaten (von *x0* bis *x0*+*dx*, von *y0* bis *y0*+*dy*) wird mit EMS gesichert. Dabei werden u.a. auch die Koordinaten gesichert, so daß das EMS-Handle zum Rekonstruieren des Bildschirms reicht. Dieses wird daher zurückgegeben.

`PROCEDURE restore_big_screen(ems_handle: INTEGER);`
Das wird hier gemacht.

D.5. Die Unit GUI_TOOL

Zweck der Unit

Die GUI-Units soll alle Aufgaben übernehmen, die einer Benutzeroberfläche (GUI=GraphicUserInterface) zukommen. Die Menüverwaltung ist schon in GUI_DEF untergebracht, GUI übernimmt die globalen Variablen und ständige Aufgaben, wie Status- und Info-Zeile verwalten, Bildschirmspeichern etc. Die nicht permanent benutzten Funktionen sind in dieser zu Overlay fähigen Unit untergebracht. Da ihr Aufruf nicht übermäßig zeitkritisch ist, ist eine eventuelle Nachladezeit als Overlay (bei EMS-Benutzung minimal) tolerierbar. Dabei geht es primär um Eingabe- und Ausgabemöglichkeiten. Unter Alert sollte man sich ein Feld vorstellen, mit dessen Hilfe eine Meldung ausgegeben und evtl. eine Eingabe (im Range von einer Auswahl aus max. 3 Optionen) entgegen genommen wird. Der Begriff kommt übrigens aus der Programmierung des Atari ST, woher auch die Anleihen zum Format der `alert_select`-Funktion stammen (genaugenommen aus GFA-Basic). So einfach war dort nämlich der Dialog über 'Alertboxen'.

Die Prozeduren von GUI_TOOL

Zu den folgenden 4 Funktionen ist zu sagen, daß sie sich gegenseitig in der angegebenen Reihenfolge aufrufen. Die eigentliche Arbeit macht also `show_big_alertxy`.

`FUNCTION show_alert(var s1,s2,s3: STRING): INTEGER;`
Zeigt 3 Strings zentriert untereinander auf dem Bildschirm nachdem der Hintergrund gespeichert wurde. Das EMS-Handle, unter dem die Daten stehen, wird zurückgegeben.

`FUNCTION show_alerty(y_ko:WORD;vars1,s2,s3:STRING; touchable1, touchable2, touchable3: Button_PTR): INTEGER;`
Es wird x-zentriert und um den übergebenen y-Wert 'zentriert' folgendes ausgegeben:

Die drei Strings, darunter wird soviel Platz gelassen, wie das größte der übergebenen Objekte in y-Ausdehnung beansprucht. Die Objekte *touchable*$_j$ werden auf ihre jeweilige linke obere Ecke hin überprüft. Liegt sie bei (0,0) so werden sie als verschiebbar (bzw. noch nicht lokalisiert) angenommen. Ihr jeweiliger x,y-Wert wird nun so gelegt, daß sie nebeneinander unter den Ausgabetext passen. Wenn ein Objektzeiger den Wert NIL hat, so wird er ignoriert (und nachfolgende auch). Dargestellt werden die Objekte aber nicht. Dies bleibt dem Aufrufer (für danach, wg. Koordinatenänderung) überlassen. (→Zur Verwendung: siehe z.B. Listing zu `f_change.do_select_work_area`.) Wieder wird das EMS-Handle des Hintergrunds übergeben.

`FUNCTION show_big_alert(y_ko:WORD;VAR tr:text_rec; touchable1, touchable2, touchable3: Button_PTR): INTEGER;`
Macht dieselbe Arbeit wie `show_alerty`, wird aber mit einer größeren Zeilenanzahl (entsprechend *tr*) aufgerufen. In Wirklichkeit ruft `show_alerty` diese Funktion auf, diese gibt denn Ball an `show_big_alertxy` weiter!

`FUNCTION show_big_alertxy(x_ko, y_ko:WORD;VAR tr:text_rec; touchable1, touchable2, touchable3: Button_PTR): INTEGER;`
Nun wird der in *tr* übergebene Text um die Koordinaten x_ko und y_ko dargestellt, wobei vorher die bis zu drei Buttons in die untersten Zeile des Textes eingepasst werden. Dies geschieht aber nur mit denen, die die Koordinate (0,0) haben. Zurückgeliefert (und über die Aufrufer zurückgeleitet) wird das EMS-Handle der Hintergrundverwaltung.

`PROCEDURE hide_alert(handle: INTEGER);`
Das von `show_alert/y` gelieferte Handle wird hier übergeben, um den Hintergrund zu rekonstruieren, also den alert zu löschen. Genausogut könnte man gleich `restore_big_screen` aufrufen

`FUNCTION save_number( x,y: LongInt ; nr: WORD): BOOLEAN;`
Wenn ein Menü nur eine Auswahl unter verschieden Optionen bewirken soll, so kann allen beteiligten Order_Buttons die Funktion `save_number` übergeben werden. Sie speichert die Nummer des angeklickten Buttons in einer globalen Variablen (die Nummer gibt die Stellung in der Liste des aufrufenden Menu_Buttons an. Sei wird durch die Reihenfolge bei `add_button` bestimmt. Der erste hat die Nummer 1 etc.

`PROCEDURE create_select(var menptr:Menu_Button_PTR;var strg....`
Es wird mit den übergebenen Strings ein Menü im übergebenen Menu_button *men_ptr* aufgebaut. Alle Strings ergeben jeweils einen Order_Button der `save_number` aufruft. Sie liegen brav untereinander mit festen Farben.

`FUNCTION handle_select(menptr: Menu_Button_PTR): WORD;`
verwaltet eine mit `create_select` aufgebaute Auswahlliste, das Ergebnis ist entweder die Nummer der ausgewählten Option oder (bei Cancel) die Null.

`FUNCTION do_edit(var strg ; str_len, x, y: WORD; ed_mode: ed_m_type): EDEXCODE;`
Der String *strg* wird an der Position *x*,*y* dargestellt. Er kann nun editiert (durch Eingaben verändert oder ersetzt) werden wobei *ed_mode* bestimmt, welche Tasten akzeptiert werden. Die Länge ist durch *str_len* beschränkt. Je nach Ende des Editierens wird ein entsprechender Code zurückgegeben. Er kann auf Cursortasten o.ä. hinweisen.

`PROCEDURE input(var what_to_do; var strg ; str_len:WORD);`
Der String what_to_to wird ausgegeben, dann der String strg editiert, wobei die Länge auf str_len begrenzt bleibt.

`PROCEDURE input_number(var what_to_do; var number: LONGINT);`
Wie oben, jedoch wird die übergebene Zahl in einen String verwandelt und nach dem Editieren wieder zurückübersetzt.

`PROCEDURE input2_numbers(var what_to_do, wtd2; var number1, number2: LONGINT);`
Wie oben, aber zwei mal. Der Wechsel zwischen den Eingabefeldern ist mit TAB oder den Cursortasten möglich.

`FUNCTION handle_input(var ed: input_rec): EDEXCODE;`
Ein durch Setzen von Werten in dem `input_rec` *ed* vorbereiteter Input wird durchgeführt. Diese Funktion wird von den anderen Inputfunktionen aufgerufen.

`FUNCTION alert_get_number( rel_x,rel_y: LONGINT; nr: WORD): BOOLEAN;`
Liefert die Nummer des angeklickten Buttons in einer globalen Variablen zurück.

`FUNCTION alert_select(text_strg,select_strg: STRING):INTEGER;`
Diese Funktion interpretiert den ersten String (*text_strg*) und trennt bis zu vier darin enthaltene und durch '|' getrennt Textzeilen auf. Diese werden untereinander dargestellt. Der zweite String wird entsprechend interpretiert und in max. drei Order_Buttons gesteckt. Dann wird eine Auswahl zwischen den dreien ermöglicht. Der erste Button ist dabei außerdem durch die Eingabetaste auslösbar, der letzte (ab 2 Stück) durch ESC. Zurückgegeben wird die Nummer des angeklickten Buttons.

`FUNCTION big_alert(VAR tr:text_rec;select_strg:STRING):INTEGER;`
Wie `alert_select`, aber der Text wird in dem Text-Record *tr* übergeben und kann entsprechend länger sein. `alert_select` ruft diese Funktion auf. Diese Funktion wiederum ruft u.a. `show_big_alert` auf.

`PROCEDURE test_big_save;`
Diese Prozedur dient nur bei der Programmentwicklung dem Test der EMS-Verwaltung (Test durch Strapazieren). Bei Umstellung der Verwaltung auf Auslagerungsdatei o.ä. wäre sie evtl. wieder ganz praktisch. Solange sie nirgends aufgerufen wird, nimmt sie der intelligente Linker auch garnicht auf (Borland sei Dank).

`PROCEDURE edit_document;`
Wird von der Menüverwaltung aufgerufen wenn 1. F8 gedrückt wurde und 2. das Compilerflag "extras" gesetzt war. Die Prozedur dient dem Beschriften von Bildschirminhalten, und war nur zur Erstellung der Bildershow im Lernprogramm gedacht. Dabei gibt man erst den Text (incl. Zeilenanzahl) ein, dieser wird bei Mausklick um die Maus zentriert dargestellt, wenn einem die Position gefällt drückt man beide Maustasten, es piept, nun kann man mittels Shift-Druck das BPLCAPT-Programm (wenn es resident ist) aufrufen. Weiterer Tastendruck löscht Beschriftung, das Programm geht weiter. Wenn man am Schluß allerdings ESC drückt, kann man den Text neu editieren. Ohne das Compilerflag bleibt die Routine nicht im Programm, da sie nie aufgerufen wird.

D.6. Die Unit F_ANALYS

Hier sind die Analysefunktionen zusammengefasst.

`FUNCTION set_higru(rel_x, rel_y:LONGINT; nr:WORD):BOOLEAN;`
Hiermit wird der Inhalt des Meßfensters in einen Speicher geschrieben, der in Zukunft (bis zum Abschalten mittels `toggle_higru`) in Grün "hinter" der Meßkurve dargestellt wird.

`FUNCTION toggle_higru(rel_x, rel_y:LONGINT; nr:WORD):BOOLEAN;`
Schaltet den Hintergrund an und aus.

`PROCEDURE init_higru;`
Legt den Speicher.. für den Hintergrund an.

`FUNCTION show_parameter(rel_x, rel_y:LONGINT; nr:WORD):BOOLEAN;`
Gibt die wichtigsten globalen Daten der Messung/Rechnung in einer Liste aus.

`FUNCTION calc_spec( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Ruft die Berechnung in F_CALC auf, mit der aus der (Meß-)Kurve ein Spektrum berechnet wird.

`FUNCTION calc_kurve( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Ruft Berechnung in F_CALC auf, die aus einem gegebenen/veränderten Spektrum wieder eine 'Meß'-Kurve, also den zeitlichen Amplitudenverlauf, berechnet.

D.7. Die Unit F_CALC

`FUNCTION time_to_calc:LONGINT;`
Liefert die Zeit zurück, die für die letzte Berechnung nötig war. Dient primär dem Test von Performanceoptimierungen in den Rechenroutinen.

`FUNCTION max_spec_amplitude:INTEGER;`
Liefert die maximale Amplitude zurück, die errechnet wurde.

```
FUNCTION FMSin( x: EXTENDED): EXTENDED;   { aus c't 4/92 }
FUNCTION FMCos( x: EXTENDED): EXTENDED;
```
Die `sin`/`cos`-Funktionen von Pascal rufen nur Funktionen des 8087 auf. Da aber ein 387 oder 486 (mit integriertem 487) installiert sein kann, ist es möglich direkte `sin`/`cos`-Berechnungen des Prozessors aufzurufen. Dies übernehmen diese von der Computerfachzeitschrift c't übernommenen Assembleraufrufe. In den Schleifen zur Berechnung ist einfach die innere Schleife, also die eigentlich zeitkritische, doppelt vorhanden und ruft entweder `sin`/`cos` von Turbo-Pascal oder `FMsin` oder `FMcos` auf. Die Entscheidung wird nach der Auskunft von Turbo-Pascal bzgl. der Koproausstattung getroffen.

`FUNCTION sign ( x:Extended):EXTENDED;`
Liefert das Vorzeichen von *x*.

`PROCEDURE calculate_spec;`
Berechnet Spektrum, legt Ergebnisse als Spektrum, Phasen, Sin- und Cos-Werte ab.

`PROCEDURE calculate_kurve;`

Berechnet aus dem Spektrum und den Phasen wieder eine Kurve die den Meßwerten, also einer Spannung als Funktion der Zeit, entspricht. Die alten Werte werden überschrieben

`PROCEDURE calc_init;`

Setzt Werte für Zeit und maximale Amplitude auf Null.

`PROCEDURE scale_mess;`

Die Meßwerte werden nach dem Maximum & Minimum durchsucht. Dem Ergebnis nach werden `mess_window, zoom_window, higru_window` und `back_-zoom_window` skaliert.

`PROCEDURE scale_spec;`

Entsprechend `scale_mess` wird hier das Spektrum in den entsprechenden Fenstern skaliert (das sind `spec_ sin_` und `cos_window`).

D.8. Die Unit F_CHANGE

`FUNCTION global_hilfe(rel_x, rel_y: LONGINT; nr:WORD):BOOLEAN`

Ruft für Spektrum-Verändern bzw. Meßwerte-Verändern die Hilfe auf, wenn der entsprechende Button im Dialogfenster angeklickt wurde; d.h. die Addresse dieser Fkt. wird dem Order_Button übergeben.

`FUNCTION set_spec_used( rel_x, rel_y:LONGINT;nr:WORD): BOOLEAN;`

Das Spektrum wird immer von der Frequenz null ab berechnet. Die Obergrenze kann aber relativ frei gewählt werden. Nur 'relativ frei', da ihre Anzahl auf 999 beschränkt ist und sie maximal halb so groß sein darf wie die Anzahl der Meßwerte zwischen den markierten Marken. Eine größere Anzahl hätte aber eine unverhältnismäßig lange Rechenzeit (486 ist sowieso schon ausgelastet...) und eine problematische, da unübersichtliche, Anzeige zur Folge. Die Wahl wird jedenfalls durch diese Funktion durchgeführt.

`FUNCTION select_zoom_factor( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`

In dem Lupenfenster wird eine Vergrößerung eines Ausschnitts der Meßdaten dargestellt. Dabei wurde ursprünglich in jeder Bildschirmspalte ein Meßwert dargestellt. Es stellte sich aber bei starken Amplitudenschwankungen als übersichtlicher heraus, wenn man nur jede zweite oder (bei evtl. höherer Bildschirmauflösung ohne größeren Monitor) gar vierte Spalte einen Meßwert darstellt. Diese Wahl wird hier per Auswahlbox durchgeführt.

`FUNCTION select_work_area_end( rel_x, rel_y:LONGINT; nr:WORD): BOOLEAN;`

Wird vom *`mess_window`* aufgerufen. Diese Funktion wählt das Ende des Arbeitsbereichs und setzt das *`select_menü`* so, daß es danach verlassen wird (→`set_leave`, → `do_select_work_area`).

`FUNCTION select_work_area_start(rel_x, rel_y:LONGINT; nr:WORD): BOOLEAN;`

Wird vom mess_window bei Anklicken aufgerufen. Hiermit wird der Anfang des Arbeitsbereiches gewählt. Dann wird die Funktion, die durch `mess_window.handle_klick` aufrufen soll, auf `select_work_area_end` gesetzt. (→`do_select_work_area`).

`FUNCTION end_select (rel_x, rel_y: LONGINT;nr: WORD): BOOLEAN;`

Diese Funktion wird vom Abbruch-Knopf bei der Auswahl des Arbeitsbereiches aufgerufen. Dann wird das ganze abgebrochen (→`do_select_work_area`).

`PROCEDURE do_select_work_area;`

Nach einer Messung sind gewöhnlich zu viele Meßwerte vorhanden um sinnvoll damit arbeiten zu können. Auch sind nicht alle Signale so periodisch, wie man es sich wünschen würde. Also ist es sinnvoller nur einen Ausschnitt zu wählen. Dieser wird direkt nach einer Messung oder nach dem Laden eines unveränderten Meßfeldes mit dieser Funktion gewählt.

Dabei wird ein Menu_Button aufgebaut, der *`mess_window`* sowie zwei Order_Buttons enthält. Diese sind für Abbruch & Hilfe zuständig. Das *`mess_window`* wird (via `set_reaction`) beauftragt beim Anklicken (mittels `handle_klick`) `select_work_area_start` aufzurufen. Diese Routine setzt dann `select_work_area_ende` als aufzurufende Funktion. Letztere beendet dann, wenn sie aufgerufen wird, die Auswahl.

`FUNCTION new_tab_eingabe(rel_x,rel_y:LONGINT;nr: WORD):BOOLEAN;`

Fragt nach, ob Funktion gewünscht, fragt nach neuer Grundfrequenz, löscht Spektrums- und Phaseneinträge und ruft dann die Tabellenbearbeitung der gerade gelöschten Werte auf.

`FUNCTION tab_eingabe( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;`

Eine Tabelle mit 20 Amplituden und Phasen des Spektrums wird dargestellt und kann editiert werden (mittels `gui_tool.do_edit`). Nach der Eingabe werden die Werte außerdem umgerechnet und in die `sin`/`cos`-Array eingetragen um auch bei der Koeffizientendarstellung zu erscheinen.

`FUNCTION octave_up( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;`
`FUNCTION octave_down( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;`

Das Spektrum wird eine Oktave höher/tiefer gesetzt.

`FUNCTION do_change_kurve(rel_x,rel_y:LONGINT; nr:WORD):BOOLEAN;`

Entsprechend der Struktur von `do_select_work_area` wird ein Menu_Button aufgebaut, der zwei Button für Ende/Hilfe enthält. Außerdem ist das Spektrumsfenster (*`spec_window`*) eingebunden, so daß bei Anklicken dieses Fensters `change_kurve` aufgerufen wird. Das Flag *`dont_leave`* im Menü bleibt gesetzt bis `end_change` aufgerufen wird.

`FUNCTION end_change(rel_x, rel_y: LONGINT;nr: WORD): BOOLEAN;`

Beendet die Veränderung mittels zurücksetzen des *`dont_leave`*-Flags (→`do_change_kurve`).

`FUNCTION change_kurve(rel_x,rel_y: LONGINT; nr: WORD): BOOLEAN;`

Verändert den Meßwert, der gerade berührt wurde. Er wird so gesetzt, daß er dargestellt an der Stelle erscheint, die gerade angeklickt wurde. Dadurch ist ein Zeichnen der Meßkurve mit der Maus möglich (→`do_change_kurve`).

`FUNCTION do_change_spec( rel_x,rel_y:LONGINT;nr:WORD): BOOLEAN;`

Dasselbe Verfahren wie bei →`do_change_kurve` wird nun beim Spektrum (anstatt bei den Meßwerten) angewandt.

`FUNCTION change_spec( rel_x, rel_y:LONGINT; nr: WORD): BOOLEAN;`

Führt die eigentliche Veränderung des Spektrums durch. Der neue Wert wird wieder mit der Maus duch Anklicken 'eingegeben'.(→`do_change_kurve`,→`do_change_spec`).

D.9. Die Unit F_DATEI

`PROCEDURE save( name: String );`

Sichert die Meßdaten unter dem übergebenen Namen. Dabei werden außer den eigentlichen Meßwerten (die direkt aus dem Speicher kopiert werden) noch Angaben zu Anzahl der Meßwerte, Lage der Marken, Meßfrequenz etc. in einem Header abgespeichert. Die vollständige Auflistung ist der Definition des Dateiheaders zu entnehmen.

Es werden auch Angaben zum Benutzer abgespeichert, indem eine Benutzeridentifikation abgespeichert wird. Im Praktikumsbetrieb wird so das Löschen und Laden von Files anderer Gruppen, sowie das Löschen der Beispieldateien (die jedoch jeder Laden darf) verhindert. Dies geschieht folgendermaßen: Die Benutzeridentifikation "Master" ist nur über ein Codewort einzugeben. Damit darf man alle Dateien bearbeiten & löschen, die erzeugten Files bekommen diese BenutzerID. Wenn man eine andere BenutzerID hat, so bekommt man immer seine und die "Master"-Dateien angezeigt, gelöscht werden dürfen aber nur die eigenen.

`PROCEDURE load( name: String );`

Die Datei mit dem angegebenen Namen wird geladen, die zusätzlichen Informationen werden ausgewertet und benutzt (→ save).

`FUNCTION file_select( match, why: string ): STRING;`

Es wird eine Fileselectbox gezeigt, in der alle Dateien gezeigt werden, die dem Pattern (Muster) *`match`* entsprechen. Dabei wird der Text 'Auswahl zum '+*`why`* ausgegeben. Außerdem müssen die Dateien über einen korrekten Header verfügen. In der Box werden nicht die Dateinamen, sondern die Dateibschreibungen gezeigt. Es wird ein String mit dem gewählten Dateinamen zurückgegeben, der leer ist, wenn die Wahl abgebrochen wurde.

`FUNCTION FileExists(FileName: string);`

Ein von der Borland-Hilfe übernommener Trick um festzustellen, ob eine Datei vorhanden ist. Wenn ja wird (womit niemand rechnen würde) TRUE zurückgegeben, sonst FALSE.

`FUNCTION save_data( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`

Es wird nach einer Dateibeschreibung gefragt. Wenn diese eingegeben wurde (also kein Abbruch oder unveränderter Textvorschlag) so wird ein recht zufälliger Name erzeugt, dessen bisheriges Vorkommen ausgeschlossen und dann `save(`*`name`*`)` aufgerufen.

`FUNCTION load_data( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`

Mit `file_select` wird eine Datei ausgewählt und bei Erfolg geladen. Dabei stehen nur Dateien mit derselben Benutzeridentifikation (~kennung) oder mit der BenutzerID "Master" zur Verfügung.

`PROCEDURE save_messung;`

Die aktuelle Messung wird ohne Rückfrage mit der Beschreibung 'Letzte Messung, unbearbeitet' gespeichert. Dies wird nach jeder Messung automatisch durchgeführt!

`PROCEDURE load_messung;`

Die mit `save_messung` abgespeicherten Werten werden hiermit wiedergeholt (wird nur beim Programmstart ausgeführt). Dies setzt aber eine korrekte Benutzerkennung voraus.

`PROCEDURE save_temp;`
`PROCEDURE load_temp;`

Diese Routinen speichern die Daten temporär und holen sie wieder. Dies geschieht nur bei Speichermangel, um für das Lernprogramm Platz zu schaffen. Nach dem Laden wird das File sofort gelöscht.

`PROCEDURE save_all(name);`
`PROCEDURE load_all(name);`

Diese Routinen speichern die Daten und holen sie wieder. Dazu werden ihnen von `save_messung/temp` bzw. `load_messung/temp` die entsprechenden (unterschiedlichen) Namen übergeben.

`PROCEDURE do_delete(stg: STRING);`

Löscht die Datei mit dem Namen *`stg`*.

`FUNCTION delete_file(rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`

Läßt mittels `file_select` eine Datei zum Löschen auswählen und entfernt diese dann.

D.10. Die Unit F_GLOBAL

Hier sind hauptsächlich die globalen Variablen untergebracht (wenn sie nicht von GUI.. verwaltet werden).

`FUNCTION set_standards(rel_x, rel_y :LONGINT; nr:WORD):BOOLEAN;`

Setzt die Anfangswerte des Programms für Oberwellenanzahl usw.

D.11. Die Unit F_GR_DEF

Bedeutung

Diese Unit übernimmt nur die Definitionen der Daten- und Objekttypen, die das Programm verwendet. Dabei wird aber die ganze Arbeit des Darstellens der Daten übernommen.

`CONSTRUCTOR Graph_Button.init...`

Erledigt das Anlegen des Objektes mit Koordinaten, Farben...

`PROCEDURE Graph_Button.ignore_first_time;`

Da alle Graph_Buttons den Hintergrund als farbige Fläche setzen können (vom Ur-Button geerbt), muß dies verhindert werden, damit mehr als einer gleichzeitig in einem Bereich darstellbar ist (z.B. Vor/Hintergrund bei der Markenwahl, oder a- und b-Koeffizienten). Daher kann jeder nur einmal den Hintergrund auf Braun setzen, was mit dem *`first_time`*-Flag vermerkt wird. Um einem Button die Hintergrundmalerei von vorneherein zu verbieten kann `ignore_first_time` aufgerufen werden.

`PROCEDURE Graph_Button.set_girwidz_color( col: WORD);`

Herr R. Girwidz schlug vor an den seitlichen Enden des Zoomfensters farbige Balken zu setzen um die Vergrößerung aus dem Meßfenster zu verdeutlichen. Dies macht das Fenster selbständig wenn die Variable *`girwidz_color`* einen Wert ungleich 0 hat. Der Wert ist dann gleich für die Farbe veantwortlich. Er kann mit dieser Funktion gesetzt werden.

`PROCEDURE Graph_Button.set_data( var dta ; fd,ld: WORD);`
Die Daten die ein Graph_Button darstellen soll müssen ihm hiermit übergeben werden. Dabei ist *dta* das Datenarray (ARRAY[0..32000] OF data_type) und *fd* und *ld* der erste/letzte aus diesem Feld darzustellende Wert.

`PROCEDURE Graph_Button.get_data( var f_d,l_d: WORD);`
Liefert die Nummern des ersten und letzten darzustellenden Wertes zurück.

`PROCEDURE Graph_Button.set_scale( bottom,top: data_type; dmode: display_mode_type);`
Hiermit werden die maximal/minimal darzustellenden Werte dem Button mitgeteilt. Diese werden dann zur Umrechnung zwischen Wert im Array und y-Koordinate herangezogen. Der Wert *dmode* gibt an, ob Punkte (points), Linien (lines), nur jeder 2. Punkt (dotted), Uhren (clocks) oder alle Punkte (all_points) dargestellt werden sollen. Alle Ausgaben erfolgen mittels XOR-Verknüpfung mit dem Hintergrund. Dieselbe Ausgabe würde also wieder löschen. Um bei nahe beieinanderliegenden Punkten kein eher zufälliges Löschen/Wiederschreiben hervorzurufen indem diese Punkte aufeinander abgebildet werden, wird jede Bildschirmposition nur gezeichnet wenn sie nicht direkt vorher schon verändert wurde. Ausnahme: Bei all_points werden doch alle gesetzt, um eine Veränderung mit der Maus mittels einzelner Ausgabe der Punkte mit XOR zu ermöglichen. (→`Graph_Button.change_value`).

`PROCEDURE Graph_Button.set_mark( nr: WORD; value: INTEGER; active: BOOLEAN);`
Eine der 4 möglichen Marken wird gesetzt. Die Marken 1&2 werden rot, 3&4 in grün dargestellt. Wenn active=TRUE ist, so wird die Marke heller gesetzt. Eine evtl. andere helle Marke wird dunkler.

`PROCEDURE Graph_Button.set_analytic(a:BOOLEAN;tt_changer:gtt);`
a ist ein Flag, das angibt, ob bei einer Berührung des Buttons mit der Maus ein *touch_text* auszugeben ist, der ständig aktualisert wird. Dazu wird (bei TRUE) vor jeder Ausgabe die Funktion aufgerufen, deren Adresse mit *tt_changer* übergeben wurde. Diese bekommt außerdem die relativen Mauskoordinaten übergeben (relativ zu den Buttonausmaßen als LongInt in Promille).

`FUNCTION Graph_Button.scr_to_data(koord: WORD): INTEGER;`
Rechnet die Bildschirm-x-Koordinate in die Nummer des in dieser Spalte dargestellten Funktionswertes um. Dabei wird keine Überprüfung vorgenommen!!

`FUNCTION Graph_Button.data_to_scr(number: WORD): WORD;`
Macht umgekehrte Berechnung ohne Überprüfung.

`FUNCTION Graph_Button.rel_to_data(rel_x: LONGINT):INTEGER;`
Rechnet die übergebene relative x-Koordinate (in Promille) in Datennummer um.

`FUNCTION Graph_Button.y_to_value( koord: WORD): data_type;`
Rechnet Bildschirm-y-Koordinate in Datenwert um; keinTest auf Sinn des Wertes und Überlauf, sollte also nur mit y∈[Oberkante,Unterkante des Fensters] aufgerufen werden!

`PROCEDURE Graph_Button.update_ttext;`
Hier kann der *touch_text* direkt geändert werden.

`FUNCTION Graph_Button.test_n_handle_touch( mx, my: WORD): BOOLEAN;`
Um die erweiterten Fähigkeiten bzgl. aktualisiertem *touch_text* realiseren zu können wurde diese Fkt. überladen, sie ruft aber weiterhin `Button`.test_... auf.

`FUNCTION Graph_Button.handle_klick( mx, my, nr: WORD): BOOLEAN;`
Wie bei Order_Button.handle_klick wird eine per übergebener Adresse bestimmte Funktion aufgerufen.

`PROCEDURE Graph_Button.select;`
`PROCEDURE Graph_Button.unselect;`
Schalten durch Überladen mit Untätigkeit diese Funktionen ab.

`PROCEDURE Graph_Button.show;`
Ruft, wenn noch nichts angezeigt wurde, `display` auf.

`PROCEDURE Graph_Button.hide;`
Ruft, wenn schon etwas dargestellt wurde, `display` auf, was dank XOR-Darstellung, ein Löschen bedeutet.

`PROCEDURE Graph_Button.display;`
Stellt den Funktionsgraph entsprechend der Einstellungen von `set_scale` und `set_data` dar. Außerdem werden die Marken dargestellt.

`PROCEDURE Graph_Button.change_value(nr:WORD; value: data_type);`
Ändert den Wert mit der Nummer *nr* auf den Wert *value*, wobei die Darstellung korrigiert wird. Dies ist nur bei all_points- oder lines-Darstellung (→`Graph_Button.set_scale`) möglich und ruft sonst eine Fehlermeldung hervor.

`PROCEDURE Graph_Button.draw_mark( nr: WORD);`
Bildet eine Marke (mit XOR, wie gehabt) auf dem Bildschirm ab.

`PROCEDURE Graph_Button.set_reaction( ptr: Do_It_Func; t_text: String; m_style: WORD );`
Es werden die bei `init` übergebenen Werte für die bei `handle_touch` aufzurufende Funktion, den *touch_text* und die Art des Mauszeigers, wie er sich bei Berühren darzustellen hat, überschrieben.

`PROCEDURE My_Circle(x,y,r: WORD);`
Diese Prozedur malt einen (weniger schönen) Kreis, der mit XOR-geschrieben und gelöscht wird! Er wird nur bei `display` mit dem Parameter *clocks* beim *display_mode* benutzt.

D.12. Die Unit F_GRAPH

`FUNCTION not_implement( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Ein Ersatz für noch nicht programmierte Routinen, die aber schon in der Menüstruktur aufrufbar sein sollen. Wenn er nicht mehr benötigt wird, wirft ihn der Linker aus dem Programm.

`FUNCTION dummy ( rel_X, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Routine die von einem Order_Button aufgerufen werden kann, um garnichts zu tun (Leerzeilen in Menüs o.ä.)

`PROCEDURE change_touch_text( var touch_text: touch_text_type; nr: WORD)`
Wird vom Graph_Button *spec_window* (innerhalb dessen `test_n_handle_touch`) aufgerufen um einen neuen *touch_text* zu kreieren. Dieser enthält dann eine Beschreibung der Frequenz (Amplitude & Phase) die gerade mit der Maus berührt wird.

`PROCEDURE change_sin_touch_text(var touch_text:touch_text_type; nr: WORD);`
Entsprechende Routine, die für *sin_window* den *touch_text* berechnet (andere Darstellung, nämlich Koeffizienten).

`FUNCTION play_sample( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Ruft nur `put_sound` aus F_SOUND auf.

`FUNCTION do_zoom( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Wird bei Anklicken des *mess_windows* aufgerufen, um die Lupe um den Punkt zu setzen, der angeklickt wurde.

`PROCEDURE do_shift(delta: INTEGER);`
Verschiebt den dargestellten vergrößerten Ausschnitt um *delta* Werte nach links oder rechts, je nach Vorzeichen. Dabei werden Marken gesetzt (sowohl in den Fenstern als auch *data_mark[..]*) und neu dargestellt (mittels `display_zoom`).

`FUNCTION do_shift_left( rel_x,rel_y:LONGINT; nr:WORD):BOOLEAN;`

`FUNCTION do_shift_right(rel_x,rel_y:LONGINT; nr:WORD):BOOLEAN;`
Rufen je nach *rel_x* und Richtung die Prozedur `do_shift` mit verschiedenen Werten auf. Sie gehören zu den Verschiebebuttons zwischen den Fenstern. Hiermit wird der Ausschnitt, der vergrößert dargestellt werden soll, nach links/rechts verschoben.

`FUNCTION set_spec_koeff_mode( x_rel, y_rel: LONGINT; nr: WORD): BOOLEAN;`
Schaltet die Darstellung des unteren Fensters um auf Ausgabe der Koeffizienten des Spektrums.

`FUNCTION set_spec_phase_mode( x_rel, y_rel: LONGINT; nr: WORD): BOOLEAN;`
Schaltet die Darstellung des unteren Fensters um auf Ausgabe der Amplituden und Phasen des Spektrums.

`PROCEDURE hide_all;`
Schaltet alle Anzeigen im unteren Fenster ab. Außerdem werden die Verschiebe-Buttons und der Marken-wechsel-Button zwischen den Fenstern abgeschaltet (wenn sie überhaupt an waren).

`FUNCTION display_spec(rel_x,rel_y: LONGINT; nr: WORD): BOOLEAN;`
Setzt Lupen(Zoom)Fenster inaktiv, aktiviert Spektrumsdarstellung.

`FUNCTION display_only_zoom( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Stellt nur Lupenfenster dar, löscht alle anderen unteren, keine Wahlmöglichkeiten außer dem Lupenbereich im Meßfenster/rote Balken Versetzen.

`FUNCTION display_zoom( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Inaktiviert evtl. Spektrumsfenster, stellt Lupenfenster/Hintergrundsfenster dar, nachdem diese mit neuen Daten (Auschnitt, Marken) versehen wurden, aktualisiert auch Marken im Meßfenster.

`FUNCTION set_first_mark(rel_x,rel_y:LONGINT;nr:WORD): BOOLEAN;`

`FUNCTION set_second_mark(rel_x,rel_y:LONGINT;nr:WORD): BOOLEAN;`
Werden aufgerufen um die Marken zu setzen. Dabei wird die jeweils andere in den Hintergrund der Zoomfensters gebracht, im Meßfenster in dunklerer Farbe angezeigt. Die aktuelle wird in den Vordergrund gebracht und heller dargestellt. Außerdem wird natürlich überhaupt auf Lupendarstellung umgeschaltet.

`FUNCTION do_swap_marks( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Schaltet zwischen den zu setzenden Marken um. Wird vom 'Andere-Marke'-Button (*swap_marks*) aufgerufen, wenn dieser angeklickt wird.

`PROCEDURE fg_get_mem;`
Reserviert für die Meß- & Spektrumsfelder den Speicher. Dieser kann wieder freigegeben werden, muß aber über diese Funktion später wieder zur Verfügung gestellt werden →siehe F_LERN

`PROCEDURE fg_setup;`
Setzt interne Daten (wie z.B. Default-Meßwerte wenn keine Meßkarte angeschlossen ist).

`PROCEDURE fg_init( x, y, xx, yy, dy, frame_color, graph_color, area_color, back_color: Word);`
Setzt interne Daten wie Farben, Koordinaten und initialisiert u.a. die Graph_Buttons.

D.13. Die Unit F_HELP

`FUNCTION explain_help( rel_x, rel_y : LONGINT; nr : WORD);BOOL.`
ruft bestimmte Hilfstexte über die Hilfe auf.

`PROCEDURE init_help;`
Bereitet die Unit vor.

`procedure show_help( nr: Integer);`
Holt Hilfstext mit der angegebenen Nummer aus Hilfsfile HELPDAT.PAS und stellt ihn mit obigen Routinen dar.

D.14. Die Unit F_LERN

Zweck

Um die Bedienung eines Programms zu erlernen ist eine abstrakte Anleitung etwas problematisch. Daher soll dem Benutzer die Bedienung einmal in Form einer Bildershow vorgeführt werden. Ausgelöst wird dies durch den Menübefehl Extras/Lernen oder durch die Dialogbox zur Begrüßung des Benutzers nach dem Programmstart.

Probleme

Die Bildershow soll schnell gehen, d.h. der Benutzer soll nicht auf den Aufbau der Bilder warten müssen. Also müssen die Bilder im Speicher bereitliegen, wenn der Benutzer sie sehen will. Das verbraucht viel Speicher.

Lösungen

Zum schnellen Laden war die Lösung folgende: Herr R.Girwidz (Uni Würzburg / Physikdidaktik) hatte dasselbe Problem und entwarf ein Fileformat (BPL), das den Bildschirmspeicher wiedergibt, außerdem sind Farbwerte enthalten. Die Bilder werden in jeweils einem File untergebracht. Erzeugt werden diese Files von dem residenten Programm BPLCAPT.EXE durch Abspeichern des Bildschirminhaltes. Dieses Programm und die Routinen zum Laden des Files stellte er mir freundlicherweise zur Verfügung.

Die Routinen wurden aufgespalten, so daß man Bilder im voraus laden kann. Normalerweise wird der Benutzer die Bilder der Reihe nach sehen wollen, man kann also bei Anzeige des Bildes 20 schon das 21. in den Speicher des Rechners laden. Wenn dann der Tastendruck für das nächste Bild erfolgt, muß es nur noch in den Bildschirmspeicher kopiert werden, was schnell genug geht. Wenn die Taste für das vorhergehende Bild gedrückt wurde, so ist dieses erst von Festplatte zu laden. Da es aber wahrscheinlich noch im Speicher des Festplattencaches ist, ist die Ladezeit ebenfalls unproblematisch kurz.

Das Speicherproblem (das Bild muß in VGA-Auflösung 640*480*16Farben komplett in den Speicher passen) tritt eigentlich nur unter der Entwicklungsumgebung auf. Dann werden die Meßdaten ausgelagert, der Speicher freigegeben, neu belegt und benutzt. Nach dem Ende des Lernprogrammes, bzw. der Bildershow, wird der Speicher wieder freigegeben und für die Meßdaten neu reserviert.

Die Funktionen von F_LERN

`PROCEDURE PCXHeader_auswerten;`
Von R.Girwidz.

`PROCEDURE VGAPalette_laden;`
Von R.Girwidz.

`PROCEDURE init_bpl;`
Legt den Speicher an, schaltet auf VGA-Graphik um.

`PROCEDURE load_bpl( Bpl_bild : STRING);`
Das angegebene Bild wird in den Puffer geladen.

`PROCEDURE show_bpl;`
Das Bild aus dem Puffer erscheint nun auf dem Bildschirm.

`PROCEDURE disable_bpl;`
Gibt den Speicher wieder frei.

`PROCEDURE do_learn( learn_file : file_name );`
Führt das Lernen für ein gegebenes File (mit einer Liste der zu zeigenden Bilder) durch.

`FUNCTION lernen (rel_x, rel_y:LONGINT; nr : WORD);`
Fragt nach dem Bereich, über den man etwas sehen will. Organisiert evtl. den Speicher um (lagert also evtl. die Meßdaten aus). Dann wird BGI abgeschaltet um sicher auf VGA-Auflösung zu kommen, es könnte ja 800*600 gewählt sein. Im EGA-Fall ist das Lernen nicht möglich. Dann wird do_learn aufgerufen, danach der Graphikmodus und der Speicher in Ordnung gebracht.

D.15. Das Programm F_MAIN

Es übernimmt die Verbindung zwischen den Units. Es fordert sie mittels `uses` an. Dann werden elementare Initialisierungen vorgenommen, wie Grafik auswählen und einschalten, sowie die jeweiligen Initialisierungen der Units (die nicht automatisch geschehen soll, da sonst keine Overlayverwaltung möglich wäre, bzw. die wichtige Reihenfolge der Initialisierungen in der Implementierung von `uses` versteckt wäre).

Die Kontrolle wird dann mittels `handle_klick` an den Menübaum abgegeben. Alle auf dem Bildschirm erscheinenden Objekte sind in dem Menübaum integriert, aber teilweise via `disable` abgeschaltet. Erst in entsprechenden Situationen werden sie eingeschaltet, andere abgeschaltet. Die Situationen sind z.B.: Umschalten zwischen der Lupendarstellung, Marken setzen, Spektrumsdarstellung in Koeffizienten oder Phasendarstellung! Nur aktivierte Objekte reagieren auf die Befehle show und `test_n_handle_touch` (u.a.) und sind also sichtbar und reagieren.

D.16 Die Unit F_MENU

`FUNCTION quit(rel_x, rel_y : LONGINT; nr:WORD);`
Setzt in dem Hauptmenü das exit_flag zurück, um diesem das Verlassen zu ermöglichen.

`FUNCTION ts(ptr : Button_PTR):Button_PTR;`
(Bedeutet to_switch). Der übergebene Pointer wird in eine Liste für umzuschaltende Menüeinträge geschrieben. Der Pointer wird zurückgegeben, damit man den Befehl in Funktionsaufrufe "einschleifen" kann.

`FUNCTION ts_inv(ptr : Button_PTR):Button_PTR;`
Merkt sich hiermit den einzigen Eintrag, der abgeschaltet wird, wenn die zusätzlichen aus der ts_liste eingeschaltet werden.

`FUNCTION switch_menu_on(rel_x, rel_y : LONGINT; nr:WORD);`
`FUNCTION switch_menu_off(rel_x, rel_y : LONGINT; nr:WORD);`
Über diese beiden Funktionen wird der Wechsel zwischen Voll/Einfach-Menü vorgenommen.

```
PROCEDURE show_menu;
```
Die Menüleiste wird gezeigt.

```
PROCEDURE init_menu;
```
Es wird der Menübaum erstellt (init_menu). Man beachte die Unterschiede zwischen statisch und dynamisch vorhandenen Objekten, sowie die Einbindung der Unter- bzw. Submenüs. Menüpunkte, die keine Funktion haben, bekommen die Routinen dummy oder not_implement zugewiesen.

```
PROCEDURE get_user_name;
```
Der Name und die Abkürzung des Benutzers werden abgefragt. Die Informationen werden beim Ausdruck und beim Abspeichern benutzt.

```
PROCEDURE say_hello;
```
Begrüßt und fragt, ob Anleitung oder Lernen notwendig sind. Ruft Entsprechendes auf.

```
PROCEDURE init_global_params;
```
Die Parameterliste des Programms wird ausgewertet um globale Flags entsprechend zu setzen. Dabei ist die Reihenfolge der Parameter egal.

D.17. Die Unit F_MESS

Die Prozeduren dieser Unit sind hauptsächlich der Diskette, die bei der Meßkarte mitgeliefert war, entnommen. Dabei wurde etwas gekürzt (nicht alles ist notwendig für dieses Programm), ein Fehler korrigiert (SystemCheck versagte bei einem zu schnellen Rechner, da das Meßfeld zu klein gewählt war) und die Routinen für die Menüaufrufe hinzugefügt. Nur diese sind hier nennenswert dokumentiert:

```
PROCEDURE beep;
```
Piept.

```
PROCEDURE Enable_INT(Int: BYTE);
PROCEDURE Disable_INT (Int: BYTE);
```
Schaltet Interrupt an/aus.

```
Procedure Int_off; {Switches all Interrupts off ...
Procedure Int_on;   {Switches all above interrups on again}

PROCEDURE Set_AD_s_rate (VAR AD_s_rate:REAL);
```
Wählt Wandlerrate (Meßfrequenz)

```
PROCEDURE SCSpeedw (VAR sample;Nos:INTEGER);

PROCEDURE SCSpeedo (VAR sample;Nos:INTEGER);

PROCEDURE MCSpeed (VAR sample;NoS:INTEGER);

Procedure System_check;
```
Testet den Rechner auf Geschwindigkeit. Wurde von mir erweitert um auf Anwesenheit der Karte zu testen!

Die folgenden Prozeduren sind nun von mir:

```
FUNCTION do_mess( mx,my: LONGINT; nr: WORD): BOOLEAN;
```
Führt Messung direkt durch.

```
FUNCTION get_mess_parameters( mx, my: LONGINT; nr: WORD): BOOLEAN;
```
Es werden Meßparameter erfragt und gesetzt.

```
PROCEDURE init_mess;
```
Es werden Flags gesetzt und `system_check` durchgeführt.

D.18. Die Unit F_MOUSE

```
procedure emulate( mr: mouse_rec);
```
Emuliert einen Mauszeiger. Dies wurde notwendig, um auch bei SuperVga-Auflösung (mittels SVGA.BGI) einen Mauszeiger zu sehen.

```
procedure mouse_reset(auto_mouse_off:Boolean;x_max,y_max:Word);
```
Setzt den Maustreiber zurück und initialisiert die Emulationsdatenbereiche, wenn das übergebene *auto_mouse_off*-Flag TRUE ist.

```
procedure show_mouse;
```
Zeigt Mauszeiger (wenn Maus aktiv ist).

```
procedure hide_mouse;
```
Versteckt (löscht) Mauszeiger (wenn Maus aktiv ist).

```
procedure get_mouse( mr: mouse_rec);
```
Holt vom Maustreiber die Koordinaten & Tastenzustände der Maus und setzt diese neu. Dann werden die Koordinaten & Tasten dem Aufrufer in dem übergebenen Record übergeben.

```
procedure set_mouse;
```
Um die Maus an eine Position zu setzen.

```
procedure set_mouse_style(nr: Integer);
```
Ändert das Aussehen der Maus indem eine der z.Z. 3 Darstellungen gewählt wird.

```
procedure set_mouse_activ( activ: BOOLEAN);
```
Ändert die Aktivität des Mauszeigers um für 'längere' Zeit die Maus auszuschalten, z.B. zum Darstellen eines Menüs.

```
procedure pacman;
```
Bildschirmschoner der von `get_mouse` aufgerufen wird, wenn zu lange keine Bewegung erfolgte. Mit einer Mausbewegung oder einem Tastendruck wird der Bildschirm rekonstruiert und `pacman` verlassen.

D.19 Die Unit F_SOUND

`PROCEDURE init_blaster`
Führt den Reset der Soundkarte durch.

`PROCEDURE init_sound;`
Es wird nach der Soundkarte gesucht und diese dann initialisiert (ihr Reset ausgelöst). Bei Erfolg wird ein Flag gesetzt und getestet ob genug Speicher für einen Ein/Ausgabe-Puffer vorhanden ist.

`PROCEDURE play_buffer;`
Gibt die Daten im Puffer aus. Den Takt liefert `handle_timer`.

`PROCEDURE handle_timer; INTERRUPT;`
Wird vom Timerinterrupt aufgerufen und gibt über eine globale Variable das Fortschreiten der Zeit an.

`PROCEDURE init_timer;`
Sorgt für den Aufruf von `handle_timer` durch den Interrupt der vom Timerchip ausgelöst wird.

`PROCEDURE put_sound(var buffer; acount, to_repeat: WORD);`
Der Puffer wird *`to_repeat`*-mal über die Soundkarte ausgegeben

D.20. Die Unit F_TEXT

`PROCEDURE f_text_init;`
Es wird ein Font im eigenen Format geladen. Er ist einfach aus dem Grafikbildschirm ausgelesen worden, nachdem mit Hilfe des BIOS alle Zeichen eingetragen wurden. Der Font ist als 8*16-Font organisiert, wobei die Bitmuster spaltenweise als Word eingetragen wurden. Die Darstellung der Zeichen erfolgt nun mit Hilfe der Prozedur `line`, so daß jede Spalte eines Zeichens als 'user defined line style' senkrecht ausgegeben wird. Das macht pro Zeichen acht Spalten gegenüber 16 Zeilen (bei waagrechter, byteweise Organisation), die dafür jeweils schneller wären; in der Summe dürfte diese Methode schneller sein.

`PROCEDURE f_outtextXY( x, y, horiz, vert:WORD; strg: STRING);`
Ein direkter Ersatz für `OutTextXY` und `SetTextJustify` von BGI. Der Text und seine Orientierung werden zugleich übergeben.

Wenn EGA-Grafik verwendet wird, so wird Borlands Original verwendet (wegen des kleineren Standard-Fonts).

D.21. Die Alphabetische Auflistung der Prozeduren

Die Reihenfolge ist: Constructoren, Destructoren, Funktionen dann Prozeduren.

`CONSTRUCTOR Button.init (bx,by,bdx,bdy...`
Unit GUI_DEF

`CONSTRUCTOR Graph_Button.init (bx,by,bdx,bdy...`
Unit GRAPH_DEF (!!)

`CONSTRUCTOR Menu_Button.init (bx,by,bdx,bdy...`
Unit GUI_DEF

`CONSTRUCTOR Order_Button.init (bx,by,bdx,bdy...`
Unit GUI_DEF

`CONSTRUCTOR Text_Button.init (bx,by,bdx,bdy...`
Unit GUI_DEF

`DESTRUCTOR Button.done;`
Unit GUI_DEF

`DESTRUCTOR Menu_Button.done;`
Unit GUI_DEF

`FUNCTION alert_get_number( rel_x,rel_y: LONGINT; nr: WORD):BOOLEAN;`
Unit GUI_TOOL

`FUNCTION alert_select(text_strg,select_strg: STRING):INTEGER;`
Unit GUI_TOOL

`FUNCTION Button.handle_klick( mx, my, nr: WORD):BOOLEAN;`
Unit GUI_DEF

`FUNCTION Button.identity:Object_Type;`
Unit GUI_DEF

`FUNCTION Button.test_hot_key ( ch: CHAR):BOOLEAN;`
Unit GUI_DEF

`FUNCTION Button.test_n_handle_touch( mx, my: WORD): BOOLEAN ;`
Unit GUI_DEF

`FUNCTION big_alert(VAR tr : text_rec; select_strg : STRING; first_bold : BOOLEAN):INTEGER;`
Unit GUI_TOOL

`FUNCTION calc_kurve( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Unit F_CALC

`FUNCTION calc_spec( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Unit F_CALC

`FUNCTION change_kurve( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Unit F_CHANGE

`FUNCTION change_spec( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Unit F_CHANGE

`FUNCTION delete_file( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Unit F_DATEI

`FUNCTION display_only_zoom(rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;`
Unit F_GRAPH

`FUNCTION display_spec( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;`
Unit F_GRAPH

```
FUNCTION display_zoom( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION do_change_kurve( rel_x, rel_y:LONGINT; nr:WORD): BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION do_change_spec( rel_x, rel_y:LONGINT; nr:WORD): BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION do_edit(varstrg;str_len,x,y:WORD;ed_mode:ed_m_type):EDEXCODE;
```
Unit GUI_TOOL

```
FUNCTION do_mess( mx,my: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_MESS

```
FUNCTION do_select_work_area(rel_x,rel_y:LONGINT; nr: WORD) : BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION do_shift_left( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION do_shift_right( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION do_swap_marks( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION do_zoom( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION dummy ( rel_X, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION end_change (rel_x, rel_y: LONGINT;nr: WORD): BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION end_select (rel_x, rel_y: LONGINT;nr: WORD): BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION explain_help( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
```
Unit F_HELP

```
FUNCTION FileExists(FileName: string)
```
Unit F_DATEI

```
FUNCTION FMCos( x: EXTENDED): EXTENDED;
```
Unit F_CALC

```
FUNCTION FMSin( x: EXTENDED): EXTENDED;
```
Unit F_CALC

```
FUNCTION file_select( match, why: string ): STRING;
```
Unit F_DATEI

Ab hier.....

```
FUNCTION Graph_Button.data_to_scr(number: WORD): WORD;
FUNCTION Graph_Button.handle_klick( mx, my, nr: WORD): BOOLEAN;
FUNCTION Graph_Button.rel_to_data(rel_x: LONGINT):INTEGER;
FUNCTION Graph_Button.scr_to_data(koord: WORD): INTEGER;  { ohne
     Range_Check !!}
FUNCTION Graph_Button.test_n_handle_touch( mx, my: WORD): BOOLEAN;
FUNCTION Graph_Button.y_to_value( koord: WORD): data_type;
```
... bis hier sind alle aus der Unit F_GR_DEF

```
FUNCTION get_mess_parameters( mx, my: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_MESS

```
FUNCTION global_hilfe (rel_r, rel_y : LONGINT; nr : WORD):BOOLEAN;
```
Unit F_CHANGE


```
FUNCTION handle_input(var ed: input_rec): EDEXCODE;
```
Unit GUI_TOOL

```
FUNCTION handle_select(menptr: Menu_Button_PTR): WORD;
```
Unit GUI_TOOL

```
FUNCTION load_data( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_DATEI

```
FUNCTION Menu_Button.handle_klick( mx, my, nr: WORD): BOOLEAN;
```
Unit GUI_DEF

```
FUNCTION Menu_Button.identity: Object_Type;
```
Unit GUI_DEF

```
FUNCTION max_spec_amplitude:INTEGER;
```
Unit F_CALC

```
FUNCTION new_tab_eingabe( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION not_implement( mx, my: LONGINT; nr: WORD): BOOLEAN;
```
Program F_MAIN

```
FUNCTION not_implement( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION Order_Button.handle_klick( mx, my, nr: WORD): BOOLEAN;
```
Unit GUI_DEF

```
FUNCTION octave_down( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION octave_up( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION play_sample( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION quit( mx, my:LONGINT; nr: WORD): BOOLEAN;
```
Program F_MAIN

```
FUNCTION save_big_screen(x0,y0,dx,dy: WORD): INTEGER;
```
Unit GUI

```
FUNCTION save_data( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_DATEI

```
FUNCTION save_number( x,y: LongInt ; nr: WORD): BOOLEAN;
```
Unit GUI_TOOL

```
FUNCTION select_work_area_end(rel_x, rel_y:LONGINT; nr:WORD): BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION select_work_area_start(rel_x,rel_y:LONGINT; nr:WORD):BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION select_zoom_factor( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION set_first_mark( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION set_higru(x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
```
Unit F_ANALAS

```
FUNCTION set_second_mark( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION set_spec_koeff_mode( x_rel, y_rel: LONGINT; nr:WORD):BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION set_spec_phase_mode( x_rel, y_rel: LONGINT; nr:WORD):BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION set_spec_used( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION set_standards(x,y : LONGINT; nr : WORD):BOOLEAN;
```
Unit F_GLOBALS

```
FUNCTION show_alert(var s1,s2,s3: STRING): INTEGER;
FUNCTION show_alerty( y_ko : WORD; var s1,s2,s3 : STRING; touchable1,
    touchable2, touchable3 : Button_PTR): INTEGER;
FUNCTION show_big_alert( y_ko : WORD; VAR tr : text_rec;
FUNCTION show_big_alertxy(x_ko, y_ko : WORD; VAR tr : text_rec;
```
Unit GUI_TOOL (alle 4)

```
FUNCTION show_parameter( rel_x, rel_y: LONGINT; nr: WORD): BOOLEAN;
```
Unit F_GRAPH

```
FUNCTION sign ( x:Extended):EXTENDED;
```
Unit F_CALC

```
FUNCTION switch_menu_off(rel_x, rel_y : LONGINT;nr:WORD):BOOLEAN;
FUNCTION switch_menu_on(rel_x, rel_y : LONGINT;nr:WORD):BOOLEAN;
```
Unit F_MENU (beide)

```
FUNCTION tab_eingabe( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;
```
Unit F_CHANGE

```
FUNCTION time_to_calc:LONGINT;
```
Unit F_CALC

```
FUNCTION toggle_higru(x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
```
Unit F_ANALYS

```
FUNCTION ts(ptr : Button_PTR):Button_PTR;
FUNCTION ts_inv(ptr : Button_PTR):Button_PTR;
```
Unit F_MENU (2x)

```
FUNCTION versch_wahl( rel_x, rel_y: LONGINT; nr: WORD):BOOLEAN;
```
Unit F_CHANGE

```
PROCEDURE Button.disable;
PROCEDURE Button.display;
PROCEDURE Button.enable;
PROCEDURE Button.force_enabled_flag(value : BOOLEAN);
PROCEDURE Button.hide;
PROCEDURE Button.select;
PROCEDURE Button.set_selectable( sable: BOOLEAN);
PROCEDURE Button.show;
PROCEDURE Button.unselect;
```
Unit GUI_DEF

```
PROCEDURE beep;
```
Unit F_MESS

```
PROCEDURE calculate_kurve;
PROCEDURE calculate_spec;
```
Unit F_CALC

```
PROCEDURE calc_init;
```
Unit F_CALC

```
PROCEDURE change_sin_touch_text(var touch_txt:touch_text_type;nr:WORD)
```
Unit F_GRAPH

```
PROCEDURE change_touch_text( var touch_text:touch_text_type;nr:WORD);
```
Unit F_GRAPH


```
PROCEDURE create_select( var menptr: Menu_Button_PTR ;  var strg ;
```
Unit GUI_TOOL

```
PROCEDURE Disable_INT (Int: BYTE);
```
Unit F_MESS

```
PROCEDURE disable_bpl;
```
Unit F_LERN

```
PROCEDURE do_delete(stg: STRING);
```
Unit F_DATEI

```
PROCEDURE do_learn(learn_file : file_name);
```
Unit F_LERN

```
PROCEDURE do_shift(delta : INTEGER);
```
Unit F_GRAPH

```
PROCEDURE edit_document;
```
Unit GUI_TOOL

```
PROCEDURE emulate( mr: mouse_rec);
```
Unit F_MOUSE

```
PROCEDURE Enable_INT(Int: BYTE);
```
Unit F_MESS

```
PROCEDURE fg_get_mem;
PROCEDURE fg_init(x,y,xx,yy,dy,frame_color,graph_color, area_color,
    back_color: Word);
```
Unit F_GRAPH

```
PROCEDURE fg_setup;
```
Unit F_GRAPH

```
PROCEDURE f_outtextXY( x, y, horiz, vert:WORD; strg: STRING);
```
Unit F_TEXT

```
PROCEDURE f_text_init;
```
Unit F_TEXT

```
PROCEDURE get_mouse;
```
Unit F_MOUSE

Ab hier.....

```
PROCEDURE Graph_Button.change_value(nr: WORD; value: data_type);
PROCEDURE Graph_Button.display;
PROCEDURE Graph_Button.draw_mark( nr: WORD);
PROCEDURE Graph_Button.get_data( var f_d,l_d: WORD);
PROCEDURE Graph_Button.hide;
PROCEDURE Graph_Button.ignore_first_time;
PROCEDURE Graph_Button.select;
PROCEDURE Graph_Button.set_analytic( analyse: BOOLEAN; tt_changer: gtt);
PROCEDURE Graph_Button.set_data( var dta ; fd,ld: WORD);
PROCEDURE Graph_Button.set_girwidz_color( col: WORD);
PROCEDURE Graph_Button.set_mark( nr: WORD; value: INTEGER; active:
    BOOLEAN);
PROCEDURE Graph_Button.set_reaction( ptr: Do_It_Func; t_text: String;
    m_style: WORD );
PROCEDURE Graph_Button.set_scale( bottom,top: data_type; dmode:
    display_mode_type);
PROCEDURE Graph_Button.show;
PROCEDURE Graph_Button.unselect;
PROCEDURE Graph_Button.update_ttext;
```
... bis hier sind alle aus der Unit F_GR_DEF

```
PROCEDURE gebe_byte_aus( bt : BYTE);
PROCEDURE get_sound(var buffer; acount: WORD);
```
Unit F_SOUND

```
PROCEDURE get_user_name;
```
Unit F_MENU

```
PROCEDURE gui_exit;
```
Unit GUI

```
PROCEDURE handle_timer; Interrupt;
```
Unit F_SOUND

```
PROCEDURE hide_alert(handle: INTEGER);
```
Unit GUI_TOOL

```
PROCEDURE hide_all;
```
Unit F_GRAPH

```
PROCEDURE hide_info_line;
```
Unit GUI

```
PROCEDURE hide_mouse;
```
Unit F_MOUSE

```
PROCEDURE hide_status;
```
Unit GUI

```
PROCEDURE init_blaster;
```
Unit F_SOUND

```
PROCEDURE init_bpl;
```
Unit F_LERN

```
PROCEDURE init_ems;
```
Unit GUI

```
PROCEDURE init_global_params;
```
Unit F_GLOBALS

```
PROCEDURE init_gui;
```
Unit GUI

```
PROCEDURE init_help;
```
Unit F_HELP

```
PROCEDURE init_higru;
```
Unit F_ANALYS

```
PROCEDURE init_menu;
```
Unit F_MENU

```
PROCEDURE init_mess;
```
Unit F_MESS

```
PROCEDURE init_sound;
PROCEDURE init_timer;
```
Unit F_SOUND

```
PROCEDURE input(var what_to_do; var strg ; str_len:WORD);
```
Unit GUI_TOOL

```
PROCEDURE input2_numbers(var what_to_do, wtd2; var number1, number2:
    LONGINT);
```
Unit GUI_TOOL

```
PROCEDURE input_number(var what_to_do; var number: LONGINT);
```
Unit GUI_TOOL

```
PROCEDURE Int_off;   {Switches all Interrupts off which otherwise
```
Unit F_MESS

```
PROCEDURE Int_on;   {Switches all above interrups on again}
```
Unit F_MESS


```
PROCEDURE load( name: String );
PROCEDURE load_all(filename:STRING);
```
Unit F_DATEI

```
PROCEDURE load_bpl( Bpl_Bild : String );
```
Unit F_LERN

```
PROCEDURE load_messung;
PROCEDURE load_temp;
```
Unit F_DATEI

```
PROCEDURE MCSpeed (VAR sample;NoS:INTEGER);
```
Unit F_MESS

```
PROCEDURE Menu_Button.add_Button( bptr: Button_PTR; sable: BOOLEAN);
PROCEDURE Menu_Button.direct_call;
PROCEDURE Menu_Button.set_leave(flag: BOOLEAN);
```
Unit GUI_DEF

```
PROCEDURE mouse_reset( auto_mouse_off: Boolean; x_max,y_max: Word);
```
Unit F_MOUSE

```
PROCEDURE My_Circle(x,y,r: WORD);
```
Unit GUI_DEF

```
PROCEDURE PCXHeader_auswerten;
```
Unit F_LERN

```
PROCEDURE pacman;
```
Unit F_MOUSE

```
PROCEDURE put_buffer;
PROCEDURE put_sound(var buffer; acount, to_repeat: WORD);
```
Unit F_SOUND

```
PROCEDURE restore_big_screen(ems_handle: INTEGER);
```
Unit GUI

```
PROCEDURE SCSpeedo (VAR sample;Nos:INTEGER);
```
Unit F_MESS

```
PROCEDURE SCSpeedw (VAR sample;Nos:INTEGER);
```
Unit F_MESS

```
PROCEDURE save( name: String );
PROCEDURE save_all(filename:STRING);
PROCEDURE save_messung;
PROCEDURE save_temp;
```
Unit F_DATEI

```
PROCEDURE say_hello;
```
Unit F_MENU

```
PROCEDURE scale_mess;    { hier werden alle Werte durchgescannt...
PROCEDURE scale_spec;
```
Unit F_CALC

```
PROCEDURE Set_AD_s_rate (VAR AD_s_rate:REAL);
```
Unit F_MESS

```
PROCEDURE set_mouse;
```
Unit F_MOUSE

```
PROCEDURE set_mouse_activ( activ: BOOLEAN);
```
Unit F_MOUSE

```
PROCEDURE set_mouse_style(nr: Integer);
```
Unit F_MOUSE

```
PROCEDURE set_sound(frequency: WORD);
```
Unit F_SOUND

```
PROCEDURE show_bpl;
```
Unit F_LERN

```
PROCEDURE show_help( nr: Integer);
```
Unit F_HELP

```
PROCEDURE show_info_line(info: STRING);
```
Unit GUI

```
PROCEDURE show_menu;
```
Program F_MAIN

```
PROCEDURE show_mouse;
```
Unit F_MOUSE

```
PROCEDURE show_status(status: STRING);
```
Unit GUI

```
PROCEDURE System_check;
```
Unit F_MESS

```
PROCEDURE test_big_save;
```
Unit GUI

```
PROCEDURE Text_Button.display;
```
Unit GUI_DEF

```
PROCEDURE Text_Button.hide;
```
Unit GUI_DEF

```
PROCEDURE Text_Button.show;
```
Unit GUI_DEF

Anhang E: Listing des Programms

E.1. Die Unit GUI_DEF

```
UNIT gui_def;        { GraphicUserInterface Teil 1}
             { Objektorientiert programmiert mit TP 6.0 / Dos }
             { Autor: C.Bienmüller Uni Würzburg }
             { Beginn: 22.08.92 }
             { Dieser Teil stellt die Objekte und globalen Variablen }
             {   & Konstanten zur Verfügung }
             { Es werden (vorläufig) vier ObjektTypen angeboten: }
             { Button, ein einfaches Objekt, das nur ein gerahmtes }
             {   Rechteck darstellt }
             { Text_Button: dito, aber zusätzlich mit Ausgabe eines }
             {   Strings,}
             { Order_Button: ..und bei Anklicken Aufruf eines Routine }
             {   mittels Pointer, mit Übergabe relativer Mauskoord.}
             { Menu_Button: Selber ein Text_Button, löst bei Anklicken das}
             {   Handling   eines   ButtonArrays   aus,   das   im   Objekt
      gespeichert}
             {   ist}

INTERFACE

USES Graph, CRT;
VAR
    char_height,char_width           : WORD;
    al_bk,
    al_col : WORD;
    col_select_titel : WORD;
CONST
    col_main = LightGray;              {Farbe für Ränder...}
    col_status = LightCyan;            {F. für Statuszeilentext}
    col_stat_back = DarkGray;          {F. f. Stat.Zeil.Hintergrund }
    col_info = White;
    col_info_back = DarkGray;

    col_help = Yellow;
    col_help_back = Blue;

    col_input =  Black;
    col_input_descr=Blue;
    col_input_back =LightGray;
    col_cs_frame =   LightBlue;

    Menu_Help = 90;
    max_touch_text   = 40;              {anzahl zeichen in touch_text}
    max_buttons      = 20;               {anzahl buttons in menu (max 26)
      }
    no_hot_key = CHAR(0);

TYPE
    Select_Array = ARRAY[1..max_buttons] OF STRING[40];
    Do_It_Func = FUNCTION ( rel_x, rel_y: LONGINT; nr : WORD) : BOOLEAN;
    text_rec = record
      strg       : Array[1..24] of string[75];
      last_string : Word;
    end;
    Object_Type = (button_t,menu_t,user_t);

    Button = OBJECT
      x, y, dx, dy        : INTEGER;
      color, frame_color,
```

```
  select_color        : WORD;
  touch_text          : STRING[max_touch_text];
  hot_key                 : CHAR;
  selectable,
  selected, enabled,
  invertable          : BOOLEAN;
  help_page           : WORD;

  CONSTRUCTORinit(    bx, by, bdx, bdy      : INTEGER;
                      bcolor, bframe_color,
                      bselect_color,bh_page : WORD;
                      hkey                  : CHAR;
                      btouch_text           : STRING);
  DESTRUCTOR  done; VIRTUAL;
  PROCEDURE   show ;VIRTUAL;
  PROCEDURE   select ;VIRTUAL;
  PROCEDURE   unselect ;VIRTUAL;
  PROCEDURE    display; VIRTUAL;
  PROCEDURE   hide; VIRTUAL;
  PROCEDURE    enable; VIRTUAL;
  PROCEDURE    disable; VIRTUAL;
  FUNCTION    test_n_handle_touch( mx, my : WORD) : BOOLEAN ;VIRTUAL;
  FUNCTION     test_hot_key( ch : CHAR):BOOLEAN; VIRTUAL;
  FUNCTION     handle_klick( mx, my, nr : WORD) : BOOLEAN ;VIRTUAL;
  PROCEDURE    set_selectable( sable : BOOLEAN); VIRTUAL;
  FUNCTION     identity: Object_Type; VIRTUAL;
  PROCEDURE    force_enabled_flag(value : BOOLEAN);
END;

Button_PTR = ^Button;

Text_Button = OBJECT( Button)
  sign_text : STRING[40];
  ptr          : POINTER;
  Size         : WORD;
  shown     : BOOLEAN;
  CONSTRUCTORinit(    bx, by, bdx, bdy      : INTEGER;
                      bcolor, bframe_color,
                      btext_color,bh_page         : WORD;
                      hkey                  : CHAR;
                      btouch_text,
                      bsign_text                  : STRING );
  PROCEDURE   display ;VIRTUAL;
  PROCEDURE   show ; VIRTUAL;
  PROCEDURE   hide ; VIRTUAL;
END;

Order_Button = OBJECT( Text_Button)
  do_it  : Do_It_Func;

  CONSTRUCTORinit(    bx, by, bdx, bdy      : INTEGER;
                      bcolor, bframe_color,
                      btext_color,bh_page         : WORD;
                      hkey                  : CHAR;
                      btouch_text,
                      bsign_text                  : STRING;
                      bdo_it                      : Do_It_Func );
  FUNCTION     handle_klick( mx, my, nr : WORD) : BOOLEAN ;VIRTUAL;
END;

Menu_Button = OBJECT( Text_Button)
  button_list     : ARRAY[1..max_buttons] OF Button_PTR;
  last_button     : WORD;
  dont_leave      : BOOLEAN;
  all_dynam       : BOOLEAN; { gibt an, ob alle Button in Liste
                           dynamisch sind und autom. gelöscht werden }
```



```
      CONSTRUCTORinit(    bx, by, bdx, bdy      : INTEGER;
                          bcolor, bframe_color,
                          btext_color,bh_page         : WORD;
                          hkey                  : CHAR;
                          btouch_text,
                          bsign_text                  : STRING);
      DESTRUCTOR  done; VIRTUAL;
      PROCEDURE   add_Button(bptr : Button_PTR; sable : BOOLEAN);
      FUNCTION     handle_klick( mx, my, nr : WORD) : BOOLEAN ;VIRTUAL;
      PROCEDURE   set_leave(flag : BOOLEAN);
      FUNCTION     identity: Object_Type; VIRTUAL;
      PROCEDURE    direct_call; VIRTUAL;
    END;

    Text_Button_PTR  = ^Text_Button;
    Order_Button_PTR = ^Order_Button;
    Menu_Button_PTR  = ^Menu_Button;

IMPLEMENTATION    { ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ }
USES f_help, dos, f_mouse, f_text, gui, gui_tool;
VAR
    menu_give_back  : Button_PTR;   { immer NIL; ausser ein Menü gibt an
      vorheriges
                                    den   Auftrag,   eine   Fkt.   durchzuführen
      zurück }
    mgbx,mgby,mgbnr : WORD;
{-------------------------------------------------------------------------------
      ----}
{------------------ Methoden von Button ---------------------------------------
      ----}
{-------------------------------------------------------------------------------
      ----}
CONSTRUCTOR  Button.init( bx, by, bdx, bdy      : INTEGER;
                          bcolor, bframe_color,
                          bselect_color,bh_page : WORD;
                          hkey                  : CHAR;
                          btouch_text           : STRING);
BEGIN
    x:=bx;       y:=by;
    dx:=bdx; dy:=bdy;
    color:=bcolor;
    frame_color:=bframe_color;
    select_color:=bselect_color;
    touch_text:=btouch_text;
    help_page:=bh_page;
    selected:=FALSE;
    selectable:=TRUE;
    enabled:=TRUE;
    invertable:=TRUE;
    hot_key :=hkey;
END;

DESTRUCTOR  Button.done;
BEGIN
    hide;
END;

PROCEDURE Button.show;
BEGIN
IF enabled THEN BEGIN
    hide_mouse;
    SetLineStyle( SolidLn, 0, NormWidth);
    SetColor( frame_color);
    Rectangle( x, y, x+dx, y+dy);
    show_mouse;
    selected:=FALSE;
```

```
    display;
END;
END;

PROCEDURE Button.select;
BEGIN
    IF (NOT selected) AND selectable THEN BEGIN
      selected:=TRUE;
      IF invertable THEN display;
    END;
END;

PROCEDURE Button.unselect;
BEGIN
    IF selected THEN BEGIN
      selected:=FALSE;
      IF invertable THEN display;
    END;
END;

PROCEDURE Button.display;
BEGIN
    hide_mouse;
    IF selected THEN
      SetFillStyle( SolidFill, select_color)
    ELSE
      SetFillStyle( SolidFill, color);
    Bar( x+1, y+1, x+dx-1, y+dy-1);
    show_mouse;
END;

PROCEDURE Button.hide;
BEGIN
IF enabled THEN BEGIN
    hide_mouse;
    {SetFillStyle( back_style, back_color);
    Bar( x, y, x+dx, y+dy);}
    selected:=FALSE;
    show_mouse;
END;
END;

FUNCTION Button.test_hot_key ( ch : CHAR):BOOLEAN;
BEGIN
    IF (ch<>CHAR(0)) AND (ch=hot_key) AND  (ch<>CHAR(0)) THEN
      test_hot_key:=test_n_handle_touch(x,y)        { immer ok }
    ELSE
      test_hot_key:=FALSE;
END;

FUNCTION Button.test_n_handle_touch( mx, my : WORD) : BOOLEAN ;
BEGIN
    IF enabled THEN BEGIN
      if ( mx >= x) AND ( mx <= x+dx) AND
         ( my >= y) AND ( my <= y+dy) THEN BEGIN
        IF NOT selected THEN BEGIN
            show_status(touch_text);
        END;
        select;
        test_n_handle_touch:=TRUE;
      END
      ELSE BEGIN
        unselect;
        test_n_handle_touch:=FALSE;
      END;
    END
```



```
    ELSE
      test_n_handle_touch:=FALSE;
END;

PROCEDURE Button.enable;
BEGIN
    IF NOT enabled THEN BEGIN
      enabled:=TRUE;
      show;
    END;
END;

PROCEDURE Button.disable;
BEGIN
    IF enabled THEN BEGIN
      hide;
      enabled:=FALSE;
    END;
END;

{ die folgende Procedure ist nur mit Vorsicht zu benutzen, da sie in die
  Verwaltung   des   Gezeichnetseins/Verwalten   des   Darstellungszustands
      eingreift }
PROCEDURE  Button.force_enabled_flag(value : BOOLEAN);
BEGIN
    enabled:=value;
    selected:=FALSE;     { nur sicherheitshalber (der Button zum Umschalten
      blieb selectiert)}
END;

FUNCTION Button.handle_klick( mx, my, nr : WORD) :BOOLEAN;
BEGIN                                       { in dieser Form eher eine}
    show_status(touch_text);                    { Dummy-funktion }
    handle_klick:=TRUE;
END;

PROCEDURE Button.set_selectable( sable : BOOLEAN);
BEGIN
    selectable:=sable;
END;

FUNCTION Button.identity: Object_Type;
BEGIN
    identity:=button_t;
END;

{-------------------------------------------------------------------------
      ----}
{------------------ Methoden von Text_Button ------------------------------
      -----}
{-------------------------------------------------------------------------
      ----}


CONSTRUCTOR  Text_Button.init(    bx, by, bdx, bdy         : INTEGER;
                                  bcolor, bframe_color,
                                  btext_color,bh_page          : WORD;
                                  hkey                     : CHAR;
                                  btouch_text,
                                  bsign_text               : STRING);
BEGIN
    Button.init( bx, by, bdx, bdy, bcolor,
             bframe_color, btext_color,bh_page, hkey , btouch_text);
    sign_text:=bsign_text;
    ptr:=NIL;
    shown:=FALSE;
```

```
END;


PROCEDURE Text_Button.show;
BEGIN
    IF enabled AND NOT shown THEN BEGIN
      hide_mouse;
      Size := ImageSize( x, y, x+dx, y+dy);
      IF Maxavail<=Size THEN BEGIN show_help(103); halt; END;
      GetMem( ptr, Size);                      { Get memory from heap }
      IF ptr<>NIL THEN
        GetImage( x, y, x+dx, y+dy, ptr^);
      show_mouse;
      shown:=TRUE;
      Button.show;
    END;
END;

PROCEDURE Text_Button.hide;
BEGIN
    IF (ptr<>NIL) AND enabled AND shown THEN BEGIN
      hide_mouse;
      PutImage(x, y, ptr^, NormalPut);
      FreeMem(ptr,size);
      ptr:=NIL;
      shown:=FALSE;
      show_mouse;
    END;
END;


PROCEDURE Text_Button.display;
BEGIN
    Button.display;
    IF NOT selected THEN
      SetColor(select_color)
    ELSE
      SetColor(color);
    hide_mouse;
    f_OutTextXY( x+8, y+ (dy div 2)+1, LeftText, CenterText, sign_text);
    show_mouse;
END;

{------------------------------------------------------------------------
      ----}
{------------------ Methoden von Order_Button ----------------------------
      ----}
{------------------------------------------------------------------------
      ----}


CONSTRUCTOR  Order_Button.init(   bx, by, bdx, bdy         : INTEGER;
                              bcolor, bframe_color,
                              btext_color,bh_page     : WORD;
                              hkey                    : CHAR;
                              btouch_text,
                              bsign_text              : STRING;
                              bdo_it                       : Do_It_Func );
BEGIN
    Text_Button.init( bx, by, bdx, bdy, bcolor,
             bframe_color,   btext_color,bh_page,   hkey   ,   btouch_text,
      bsign_text);
    do_it:=bdo_it;
END;

FUNCTION Order_Button.handle_klick( mx, my, nr : WORD) : BOOLEAN;
```



```
VAR
    rel_x_pm,rel_y_pm  :  LONGINT;  {  für  Angabe  der  relativen  Koord.  in
      Promille}
BEGIN
    show_status(touch_text);
    rel_x_pm:=(1000*LONGINT(mx-x)) DIV (dx);
    rel_y_pm:=(1000*LONGINT(my-y)) DIV (dy);
    handle_klick:=do_it( rel_x_pm, rel_y_pm, nr);
END;



{------------------------------------------------------------------------
      ----}
{------------------ Methoden von Menu_Button -----------------------------
      ----}
{------------------------------------------------------------------------
      ----}

CONSTRUCTOR  Menu_Button.init(    bx, by, bdx, bdy         : INTEGER;
                              bcolor, bframe_color,
                              btext_color,bh_page     : WORD;
                              hkey                    : CHAR;
                              btouch_text,
                              bsign_text              : STRING);
BEGIN
    Text_Button.init( bx, by, bdx, bdy, bcolor,
             bframe_color,   btext_color,   bh_page,   hkey,   btouch_text,
      bsign_text);
    last_button:=0;
    dont_leave :=FALSE;
    all_dynam:=FALSE;
END;

DESTRUCTOR Menu_Button.done;
VAR
    result : BOOLEAN;
    bptr : ^Button;
BEGIN
    IF menu_give_back<>NIL THEN
      result:=menu_give_back^.handle_klick(mgbx,mgby,mgbnr);
    WHILE (last_button >= 1) AND (all_dynam) DO BEGIN
      bptr:=button_list[last_button];
      dispose(bptr,done);
      DEC(last_button);
    END;
    hide;
END;

PROCEDURE Menu_Button.direct_call;
VAR
    dummy : BOOLEAN;
BEGIN
    dummy:=handle_klick(0,0,0);
    IF menu_give_back<>NIL THEN
      dummy:=menu_give_back^.handle_klick(mgbx,mgby,mgbnr);
    menu_give_back:=NIL;
END;

PROCEDURE Menu_Button.add_Button( bptr : Button_PTR; sable : BOOLEAN);
BEGIN
    IF (last_button<max_buttons) THEN BEGIN
      Inc(last_button);
      button_list[last_button]:=bptr;
      bptr^.set_selectable(sable);
    END;
```

```
END;

{ Menu_Button.handle steuert die meisten Benutzereingaben durch Verwalten
    von einem Array of Buttons, die wiederum auf touch und klick zu
      reagieren
    haben. Jetzt auch mit der Möglichkeit die Maus gedrückt zu halten
      (hold_down)
    und die Auswahl durch Loslassen zu treffen sowie kontextsensitive
      Hilfe}
FUNCTION Menu_Button.handle_klick( mx, my, nr : WORD) : BOOLEAN;
CONST
    no_leave_text    : STRING =
      'Wählen Sie mit der Maus. Etwas mit ihr berühren,dann F1 drücken gibt
      Hilfe dazu';
    leave_text       : STRING ='Auswahl durch Anklicken mit linker, Abbruch
      mit rechter Maustaste';
    drag_text        :  STRING  ='Auswahl  durch  Loslassen  über  Befehl,
      anderswo: Abbruch';
    tabelle          :          ARRAY[0..15]          OF          WORD
      =(0,0,15,0,0,0,0,15,0,15,15,15,15,15,15,15);
VAR
    mr   : mouse_rec;
    d : WORD;
    tx,ty : WORD;
    ch   : CHAR;
    strg: STRING;
    touched_any,     { wird gesetzt, wenn irgeneines je berührt wurde (Maus
      o. HotKey) }
    to_handle,           {  merkt  sich  ob  handle_klick  nach  Verlassen
      auszuführen ist }
    touched,           { wenn gerade ein Button berührt wird }
    hot_keyed,           { wenn der touched durch hotkey entstand }
    outside,         { wenn Maus ausserhalb ist }
    hold_down,           { merkt Maustastenzustand (für drag/drop) }
    result,              { merkt nur Ergebnisse, diese werden ignoriert }
    help_wanted : BOOLEAN;

BEGIN
    d:=1;                                {   der   Reihe   nach   Buttons
      darstellen }
    set_mouse_activ(FALSE);
    menu_give_back:=NIL;
    WHILE (d<=last_button) DO BEGIN
      button_list[d]^.show;
      Inc(d);
    END;
    hide_info_line;
    CASE dont_leave OF
      TRUE: show_info_line(no_leave_text);
      FALSE: show_info_line(leave_text);
    END;
    set_mouse_activ(TRUE);
    outside:=FALSE;
    hold_down:=TRUE;
    touched_any:=FALSE;
    to_handle:=FALSE;
    get_mouse(mr);
    REPEAT                               { Eigentliches Handling }
      ch:=char(0);
      help_wanted:=FALSE;
      IF NOT mr.left THEN begin
         IF hold_down THEN    CASE dont_leave OF
             TRUE: show_info_line(no_leave_text);
             FALSE: show_info_line(leave_text);
         END;
         hold_down:=FALSE;
```



```
         IF KeyPressed THEN BEGIN    { Tastenabfrage }
             ch:=ReadKey;
             IF ch=CHAR(0) THEN BEGIN       { Funktionstaste ? }
                 ch:=ReadKey;
                 IF ch=CHAR(59) THEN BEGIN
                     help_wanted:=TRUE;
                     ch:=CHAR(0);
                 END;
{$IFDEF extras}
                 IF ch=CHAR(68) THEN BEGIN
                     d:=save_big_screen(0,0,GetMaxX,GetMaxY);
                     FOR tx:=0 TO GetMaxX DO BEGIN
                         FOR ty:=0 TO GetMaxY DO BEGIN
                             PutPixel(tx,ty,tabelle[GetPixel(tx,ty)]);
                     END;END;
                     REPEAT
                     UNTIL ReadKey=CHAR(27);
                     restore_big_screen(d);
                     ch:=CHAR(0);
                 END;
                 IF ch=CHAR(67) THEN BEGIN
                     d:=save_big_screen(0,0,GetMaxX,GetMaxY);
                     FOR ty:=0 TO GetMaxY DO BEGIN
                         FOR tx:=0 TO GetMaxX DO BEGIN
                             PutPixel(tx,ty,15-tabelle[GetPixel(tx,ty)]);
                     END;END;
                     REPEAT
                     UNTIL ReadKey=CHAR(27);
                     restore_big_screen(d);
                     ch:=CHAR(0);
                 END;
                 IF ch=CHAR(66) THEN
                     edit_document;
{$ENDIF }
             END;
         END
         ELSE
             ch:=CHAR(0);
      END;
      IF hold_down THEN show_info_line(drag_text);
      get_mouse(mr);
      d:=1;
      touched:=FALSE;
      hot_keyed:=FALSE;
      WHILE (d<=last_button) AND (touched=FALSE) DO BEGIN
        touched:=button_list[d]^.test_hot_key( ch );
        IF touched THEN
            hot_keyed:=TRUE
        ELSE IF ch=CHAR(0) THEN
            touched:=button_list[d]^.test_n_handle_touch(mr.x,mr.y);
        Inc(d);
      END;
      IF ch=CHAR(27) THEN mr.right:=TRUE;
      ch:=CHAR(0);
      IF touched THEN BEGIN
        touched_any:=TRUE;
        outside:=FALSE;
      END
      ELSE BEGIN
        IF NOT outside THEN BEGIN
            outside:=TRUE;
            show_status('Keine Auswahl getroffen');
        END;
      END;
      nr:=d-1;
      set_mouse_activ(FALSE);
```

```
      WHILE (d<=last_button) DO BEGIN
        button_list[d]^.unselect;
        Inc(d);
      END;
      set_mouse_activ(TRUE);
      IF (help_wanted AND touched) THEN
        show_help(button_list[nr]^.help_page);
      IF (help_wanted AND NOT touched) THEN
        show_help(help_page);
      IF help_wanted THEN CASE dont_leave OF
        TRUE: show_info_line(no_leave_text);
        FALSE: show_info_line(leave_text);
      END;


      IF ((mr.left<>hold_down) AND touched) OR hot_keyed THEN BEGIN
        result:=TRUE;
        IF dont_leave OR (button_list[nr]^.identity=menu_t) THEN BEGIN
            result:=button_list[nr]^.handle_klick(mr.x,mr.y,nr);
            show_info_line(no_leave_text);
        END
        ELSE
            to_handle:=TRUE;
        IF dont_leave AND (menu_give_back<>NIL) THEN BEGIN
            result:=menu_give_back^.handle_klick(mgbx,mgby,mgbnr);
            menu_give_back:=NIL;
            show_info_line(no_leave_text);
        END;
      END
      ELSE IF (NOT mr.left) AND hold_down AND outside
                                          AND touched_any THEN
        result:=TRUE
      ELSE IF mr.left AND (NOT hold_down) AND outside THEN
        result :=TRUE
      ELSE
        result:=FALSE;
    UNTIL ( result OR mr.right ) AND NOT dont_leave;
    d:=last_button;
    set_mouse_activ(FALSE);
    WHILE (d>=1) DO BEGIN
      button_list[d]^.hide;
      Dec(d);
    END;
    set_mouse_activ(TRUE);
    IF to_handle THEN BEGIN
      menu_give_back:=button_list[nr];
      mgbnr:=nr;
      mgbx:=mr.x;
      mgby:=mr.y;
    END;

    handle_klick:=TRUE;
END;                    { of Menu_Button.handle_klick }

PROCEDURE Menu_Button.set_leave(flag : BOOLEAN);
BEGIN
    dont_leave:=flag;
END;

FUNCTION Menu_Button.identity: Object_Type;
BEGIN
    identity:=menu_t;
END;


END.
```

E.2. Die Unit GUI

```
UNIT gui;        { GraphicUserInterface Teil 2}
             { Objektorientiert programmiert mit TP 6.0 / Dos }
             { Autor: Christian Bienmüller Uni Würzburg }
             { Beginn: 22.08.92 }
             { Dieser Teil stellt die allgemeinen Routinen zur Bildschirm-
                 gestaltung zur Verfügung }

INTERFACE

USES Graph, CRT, gui_def;

    PROCEDURE init_gui;
    PROCEDURE show_status( status: STRING);
    PROCEDURE hide_status;
    PROCEDURE show_info_line( info: STRING);
    PROCEDURE hide_info_line;
    FUNCTION  save_big_screen( x0,y0,dx,dy : WORD):INTEGER;
    PROCEDURE restore_big_screen(ems_handle : INTEGER);

IMPLEMENTATION    { ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ }
USES f_help, dos, f_mouse, f_text, gui_tool;

{-------------------------------------------------------------------------
      ----}
{------------------ Allgemeine Prozeduren ---------------------------------
      ----}
{-------------------------------------------------------------------------
      ----}
VAR
    ems_ptr : pointer;
    old_exit : pointer;
    old_info : STRING;

PROCEDURE init_ems;
TYPE
    IDType = ARRAY[1..8] OF CHAR;
CONST
    Ems = $67;
    ID : IDType = 'EMMXXXX0';
VAR
    regs : Registers;
BEGIN
    show_status('Teste EMS Treiber');
    GetIntVec(Ems,ems_ptr);
    IF ID <> IDType(Ptr(Seg(ems_ptr^),10)^) THEN BEGIN
      ems_ptr:=NIL;
      show_help(86);
      halt(67);
    END;

    regs.ah:=$41;
    Intr(Ems,regs);
    hide_status;
    IF regs.ah<>0 THEN BEGIN
      ems_ptr:=NIL;     { es wird kein ems zugriff mehr durchgeführt }
      show_help(87);
      halt(67);
    END
    ELSE
      ems_ptr:= Ptr(regs.bx,0);
    {test_big_save;} { nur zum Testen ob big_save korrekt funktioniert &
      Tempo}
END;
```



```
PROCEDURE gui_exit;
BEGIN
    ExitProc:=old_exit;
END;

PROCEDURE init_gui;
BEGIN
    old_exit:=ExitProc;
    ExitProc:=@gui_exit;
    char_height := 16;
    IF EGA_flag THEN char_height:=10;
    char_width := 8;
    al_bk := col_help_back;
    al_col:= col_help;
    old_info:='';
    show_info_line(' Das Programm wird gestartet, bitte warten sie ');
    init_ems;
    col_select_titel:=Yellow;
END;

PROCEDURE hide_status;
VAR
    mr : mouse_rec;
BEGIN
    get_mouse(mr);
    IF mr.y<2*char_height+16 THEN
      hide_mouse;
    SetFillStyle( SolidFill , col_stat_back );
    SetColor(col_main);
    SetWriteMode( CopyPut);
    Bar ( (GetMaxX div 2) +1, 1, GetMaxX -2, char_height +5 );
    Rectangle (GetMaxX div 2, 0, GetMaxX -0, char_height +6 );
    IF mr.y<2*char_height+16 THEN
      show_mouse;
END;

PROCEDURE show_status(status : STRING);
VAR
    mr : mouse_rec;
BEGIN
    hide_status;
    get_mouse(mr);
    SetColor( col_status);
    IF mr.y<2*char_height+16 THEN
      hide_mouse;
    f_OutTextXY (GetMaxX - GetMaxX div 4, 3, CenterText, TopText, status);
    IF mr.y<2*char_height+16 THEN
      show_mouse;
END;


PROCEDURE hide_info_line;
BEGIN
    hide_mouse;
    SetFillStyle( SolidFill , col_info_back );
    SetColor(col_main);
    Bar ( 1, GetMaxY-char_height-3, GetMaxX -1, GetMaxY-1 );
    Rectangle ( 0, GetMaxY-char_height-4, GetMaxX , GetMaxY-0 );
    show_mouse;
    old_info:='';
END;

PROCEDURE show_info_line(info : STRING);
BEGIN
    IF old_info<>info THEN BEGIN
```

```
      hide_info_line;
      SetColor( col_info);
      hide_mouse;
      f_OutTextXY ( GetMaxX div 2, GetMaxY-2, centertext,bottomtext, info);
      show_mouse;
      old_info:=info;
    END;
END;

TYPE
    koord = RECORD
            last_page, x: WORD;
            y           : ARRAY[0..60] OF WORD;
    END;
    koord_array = ARRAY[1..127] OF koord;

FUNCTION save_big_screen(x0,y0,dx,dy : WORD): INTEGER;
CONST
    Ems = $67;
VAR
    k               : koord;
    d,f,
    line_size,
    last_page_lines,
    pages_needed    : WORD;
    lines_pro_page  : INTEGER;
    regs            : Registers;
    ems_handle      : INTEGER;

BEGIN
IF ems_ptr<>NIL THEN BEGIN
    IF (dy=0) THEN dy:=1;
    line_size       :=ImageSize(x0,y0,x0+dx,y0);
    lines_pro_page  :=(16000 div line_size)-1;
    pages_needed    :=dy DIV lines_pro_page;
    IF dy MOD lines_pro_page <> 0 THEN INC(pages_needed);
    last_page_lines:=dy-(pages_needed-1)*lines_pro_page;

    regs.ah:=$43;
    regs.bx:=pages_needed;          { belege EMS-Seiten }
    Intr(Ems,regs);
    IF regs.ah<>0 THEN BEGIN
      ems_ptr:=NIL;
      show_help(87);
      halt(regs.ah);
    END;
    ems_handle:=regs.dx;

    FOR d:=0 TO pages_needed-1 DO BEGIN
      k.y[d]:=y0+d*lines_pro_page;
    END;
    k.x:=x0;
    {IF (y0+dy-lines_pro_page > 0) AND (pages_needed>1) THEN
      k.y[pages_needed-1]:=y0+dy-lines_pro_page;}
    k.last_page:=pages_needed-1;
    hide_mouse;
    FOR f:=0 TO pages_needed-1 DO BEGIN
      regs.ah:=$44;       { Lege Fenster über Seiten (Mapping)}
      regs.al:=0;         { FensterSeite }
      regs.bx:=f;         { SpeicherSeite }
      regs.dx:=ems_handle;
      Intr(Ems,regs);
      IF regs.ah<>0 THEN BEGIN
        ems_ptr:=NIL;
        show_help(87);
        halt(regs.ah);
```



```
      END;
      IF (f<>pages_needed-1) THEN
        GetImage(k.x,k.y[f],k.x+dx,k.y[f]+lines_pro_page,ems_ptr^)
      ELSE
        GetImage(k.x,k.y[f],k.x+dx,k.y[f]+last_page_lines,ems_ptr^)
      ;
      koord_array(ems_ptr^)[127]:=k;   { großzügig überallhin schreiben }
    END;
    show_mouse;
END;
save_big_screen:=ems_handle;
END;

PROCEDURE restore_big_screen(ems_handle : INTEGER);
CONST
    Ems = $67;
VAR
    k       : koord;
    d,f : WORD;
    regs: Registers;
BEGIN
hide_mouse;
IF ems_ptr<>NIL THEN BEGIN
    regs.ah:=$44;
    regs.al:=0;          { FensterSeite }
    regs.bx:=0;          { SpeicherSeite }
    regs.dx:=ems_handle;
    Intr(Ems,regs);
    IF regs.ah<>0 THEN BEGIN
      ems_ptr:=NIL;
      show_help(87);
      halt(regs.ah);
    END;
    k:= koord_array(ems_ptr^)[127];
    FOR f:=0 TO k.last_page DO BEGIN
      regs.ah:=$44;
      regs.al:=0;         { FensterSeite }
      regs.bx:=f;         { SpeicherSeite }
      regs.dx:=ems_handle;
      Intr(Ems,regs);
      IF regs.ah<>0 THEN BEGIN
        ems_ptr:=NIL;
        show_help(87);
        halt(regs.ah);
      END;
      PutImage(k.x,k.y[f],ems_ptr^,NormalPut);
    END;
    show_mouse;
    regs.ah:=$45;           { Abmelden der Seiten }
    regs.dx:=ems_handle;
    Intr(Ems,regs);
    IF regs.ah<>0 THEN BEGIN
      ems_ptr:=NIL;
      show_help(87);
      halt(regs.ah);
    END;
END;
END;
END.
```

E.3. Die Unit GUI_TOOL

```
UNIT gui_tool;  { GraphicUserInterface Teil 3}
                        { Objektorientiert programmiert mit TP 6.0 / Dos }
                        { Autor: C.Bienmüller Uni Würzburg }
                        { Beginn: 22.08.92 }
                        { Dieser Teil stellt die allgemeinen Routinen zur Ein/Ausgabe
                              von Alertboxen, Input... zur Verfügung }
{$F+} {$O+}     { Overlay erlaubt, da Aufruf der Routinen nicht zeitkritisch}

INTERFACE

USES Graph, CRT, gui_def;

CONST
        max_edit = 20;
TYPE
        ed_m_type = (ascii, numbers, allchars);
        EDEXCODE = (cancel,ok,tab,cleft,cright,cup,cdown,mouse_clicked);
        input_rec = Record
                titel:          STRING[20];
                head_line:      STRING[40];
                base_line:      STRING[40];
                wtd :           ARRAY[1..max_edit] OF STRING[30];
                data:           ARRAY[1..max_edit] OF STRING[38];
                data2:          ARRAY[1..max_edit] OF STRING[19];
                dtyp:           ARRAY[1..max_edit] OF ed_m_type;
                dlen:           ARRAY[1..max_edit] OF WORD;
                double_flag : BOOLEAN;
                last_edit : WORD;
        END;

        PROCEDURE create_select(var menptr : Menu_Button_PTR ;  var strg ; last_strg,
x0 ,y0 : WORD; var titel);
        FUNCTION  handle_select(menptr : Menu_Button_PTR): WORD;
        FUNCTION  show_alert(var s1,s2,s3 : STRING): INTEGER;
```



```
        FUNCTION  show_alerty( y_ko : WORD; var s1,s2,s3 : STRING; touchable1,
touchable2, touchable3 : Button_PTR): INTEGER;
        PROCEDURE hide_alert(handle : INTEGER);
        PROCEDURE input(var what_to_do; var strg ; str_len:WORD);
        PROCEDURE input_number(var what_to_do; var number : LONGINT);
        PROCEDURE input2_numbers(var what_to_do, wtd2; var number1, number2 :
LONGINT);
        FUNCTION  handle_input(var ed : input_rec): EDEXCODE;
        FUNCTION  alert_select(text_strg,select_strg : STRING):INTEGER;
        FUNCTION  big_alert(VAR tr : text_rec; select_strg : STRING; first_bold :
BOOLEAN):INTEGER;
        FUNCTION  show_big_alert( y_ko : WORD; VAR tr : text_rec;
                                touch1, touch2, touch3 : Button_PTR; first_bold :
BOOLEAN): INTEGER;
        FUNCTION  show_big_alertxy(x_ko, y_ko : WORD; VAR tr : text_rec;
                                touch1, touch2, touch3 : Button_PTR; first_bold :
BOOLEAN): INTEGER;
        PROCEDURE edit_document;

IMPLEMENTATION    { ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ }
USES f_help, dos, f_mouse, f_text, gui;

{-------------------------------------------------------------------------}
{------------------ Allgemeine Prozeduren ----------------------------------}
{-------------------------------------------------------------------------}
VAR
        selected_number : WORD;

procedure test_big_save;
VAR
        x,y,xx,yy : WORD;
        handle : INTEGER;
BEGIN
While (readkey<>CHAR(27)) DO BEGIN
        x:=1+Random(GetMaxX-1);
        xx:=Random(GetMaxX-x-1);
        y:=1+Random(GetMaxy-1);
        yy:=Random(GetMaxy-y-1);
```

```
        handle:=save_big_screen(x,y,xx,yy);
        SetFillStyle(SolidFill,al_col);
        bar(x-1,y-1,xx+x+1,yy+y+1);
        Restore_big_screen(handle);
END;END;

{-------------------------------------------------}
{ Nun die Funktionen die eine Alertbox erstellen: }
{-------------------------------------------------}

FUNCTION show_alert(var s1,s2,s3 : STRING): INTEGER;
BEGIN
        show_alert:=show_alerty( GetMaxY div 2,s1,s2,s3,NIL,NIL,NIL);
END;

FUNCTION  show_alerty( y_ko : WORD; var s1,s2,s3 : STRING; touchable1, touchable2,
touchable3 : Button_PTR): INTEGER;
VAR
        tr : text_rec;
BEGIN
        tr.strg[1]:=s1;
        tr.strg[2]:=s2;
        tr.strg[3]:=s3;
        tr.last_string:=3;
        show_alerty:=show_big_alert(y_ko, tr, touchable1, touchable2, touchable3,FALSE);
END;

FUNCTION  show_big_alertxy(x_ko, y_ko : WORD; VAR tr : text_rec;
                touch1, touch2, touch3 : Button_PTR; first_bold : BOOLEAN): INTEGER;
VAR
        x, y, dx, dy, d             : INTEGER;
BEGIN
        dx:=0;
        IF first_bold THEN dx:=Length(tr.strg[1])*2;
        IF tr.last_string>24 THEN tr.last_string:=24; { milder Schutz }
        FOR d:=1 TO tr.last_string DO BEGIN
                IF dx<Length(tr.strg[d]) THEN dx:=Length(tr.strg[d]);
```



```
        END;
        dx:=(dx +4)*char_width;
        x:= x_ko-(dx div 2);
        IF x <= 1                       THEN x:=2;
        IF x+dx >= GetMaxX          THEN x:=GetMaxX-1-dx;
        IF dx >= GetMaxX    THEN BEGIN show_help(106); dx:=GetMaxX-5; END;

        dy:=(tr.last_string+2)*char_height;
        IF touch1<>NIL THEN
                INC(dy,touch1^.dy+10);
        y:= (2*y_ko - dy) div 2;
        IF y<=0                                 THEN y:=2;
        IF y+dy >= GetMaxY THEN y:=GetMaxY-2-dy;
        IF touch1<>NIL THEN BEGIN
                IF touch2 = NIL THEN BEGIN
                        touch1^.x:=x+ ((dx-touch1^.dx) div 2);
                        touch1^.y:=y+dy-touch1^.dy-10;
                END
                ELSE IF touch3 = NIL THEN BEGIN
                        touch1^.x:=x+ (dx div 2)-touch1^.dx-2;
                        touch1^.y:=y+dy-touch1^.dy-10;
                        touch2^.x:=x+ (dx div 2)+2;
                        touch2^.y:=y+dy-touch2^.dy-10;
                END
                ELSE BEGIN
                        touch2^.x:=x+ ((dx-touch2^.dx) div 2);
                        touch2^.y:=y+dy-touch2^.dy-10;
                        touch1^.x:=x+ (dx div 2)-touch1^.dx - 4 - touch2^.dx div 2;
                        touch1^.y:=y+dy-touch1^.dy-10;
                        touch3^.x:=x+ (dx div 2) + 4 + touch2^.dx div 2;
                        touch3^.y:=y+dy-touch2^.dy-10;
                END;
        END;

        show_big_alertxy:=save_big_screen(x,y,dx,dy);
        hide_mouse;
```

```
        SetFillStyle(SolidFill, al_bk);
        Bar(x,y,x+dx,y+dy);
        SetColor(al_col);
        Rectangle(x+1,y+1,x+dx-1,y+dy-1);
        f_text_bold:=first_bold;
        FOR d:=1 TO tr.last_string DO BEGIN
                Inc (y, char_height);
                IF first_bold AND (d>1) THEN
                        f_OutTextXY( x+2*char_width,y, LeftText, TopText,tr.strg[d])
                ELSE
                        f_OutTextXY( x+(dx div 2),y, CenterText, TopText,tr.strg[d]);
        END;
        show_mouse;
END;

FUNCTION  show_big_alert( y_ko : WORD; VAR tr : text_rec;
                          touch1, touch2, touch3 : Button_PTR; first_bold :
BOOLEAN): INTEGER;
BEGIN
        show_big_alert:=show_big_alertxy(GetMaxX DIV 2, y_ko, tr, touch1, touch2,
touch3, first_bold);
END;

PROCEDURE hide_alert(handle : INTEGER);
BEGIN
        restore_big_screen(handle);
END;

VAR
        alert_number:WORD;
        select_menu : Menu_Button_PTR;

FUNCTION alert_get_number( rel_x,rel_y : LONGINT; nr : WORD):BOOLEAN;
BEGIN
        alert_number:=nr;
        select_menu^.set_leave(FALSE);
        alert_get_number:=TRUE;
```



```
END;


{---------- Die Funktion analysiert die beiden Strings. Dabei ist der
        erste nur für max. 3 Kommentare die mit dem senkrechten | getrennt sind.
        der zweite String beinhaltet genauso getrennte Befehle/Auswahlmöglichkeiten.
        Die Idee ist von Ataris GEM bzw. GFA-Basic übernommen. Zurückgegeben wird
        die Nummer des angeklickten Befehls.
        Standardmäßig wird der erste Button auch mit RETURN, der letzte mit ESC
        ausgelöst }

FUNCTION  alert_select(text_strg,select_strg : STRING):INTEGER;
VAR
        tr : text_rec;
        dummy : BOOLEAN;
        strg : STRING;
        d,nr,f : WORD;
        handle : INTEGER;
BEGIN
        d:=1;
        f:=1;
        FOR d:=1 TO 4 DO
                tr.strg[d]:='';
        d:=1;
        WHILE (d<=Length(text_strg)) AND (f<5) DO BEGIN
                IF (text_strg[d]='|') THEN
                        INC(f)
                ELSE
                        tr.strg[f]:=tr.strg[f]+text_strg[d];
                INC(d);
        END;
        tr.last_string:=f;
        alert_select:=big_alert(tr, select_strg, FALSE);
END;

{---------- Die Funktion analysiert den String.
```

```
        Er beinhaltet durch '|' getrennte Befehle/Auswahlmöglichkeiten.
        Zurückgegeben wird  die Nummer des angeklickten Befehls. Standardmäßig
        wird der erste Button auch mit RETURN, der letzte (ab zwei) mit ESC
        ausgelöst. Vorher werden alle strings in text_rec ausgegeben!
        Das Flag first_bold gibt an, ob der erste String Fett dargestellt wird.
        In Abhängigkeit davon (wenn TRUE) werden die nächsten Zeilen linksbündig
        oder (wenn False) zentriert dargestellt. }

FUNCTION    big_alert(VAR tr : text_rec; select_strg : STRING; first_bold :
BOOLEAN):INTEGER;
VAR
        dummy : BOOLEAN;
        s : ARRAY[1..3] OF STRING[20];
        b : ARRAY[1..3] OF Order_Button;
        old_menu : Menu_Button_PTR;
        strg : STRING;
        d,nr,f : WORD;
        handle : INTEGER;
BEGIN
        FOR d:=1 TO 3 DO
                s[d]:='';
        d:=1;
        nr:=1;
        WHILE (d<=Length(select_strg)) AND (nr<4) DO BEGIN
                IF (select_strg[d]='|') THEN
                        INC(nr)
                ELSE
                        s[nr]:=s[nr]+select_strg[d];
                INC(d);
        END;
        b[1].init(0,0,(2+Length(s[1]))*char_width,char_height,al_bk,White,al_col,117,CHA
R(13),'Ihre Wahl',s[1],alert_get_number);
        IF nr=2 THEN

        b[2].init(0,0,(2+Length(s[2]))*char_width,char_height,al_bk,al_col,al_col,117,CHA
R(27),'Ihre Wahl',s[2],alert_get_number);
        IF nr=3 THEN BEGIN
```



```
        b[2].init(0,0,(2+Length(s[2]))*char_width,char_height,al_bk,al_col,al_col,117,no_h
ot_key,'Ihre Wahl',s[2],alert_get_number);

        b[3].init(0,0,(2+Length(s[3]))*char_width,char_height,al_bk,al_col,al_col,117,CHA
R(27),'Ihre Wahl',s[3],alert_get_number);
        END;
        d:=1;
        f:=1;
        old_menu:=select_menu;    { damit rekursiver alert-Aufruf mögl. (wg.Hilfe) }
        select_menu:=New(Menu_Button_PTR,init(0,0,0,0,0,0,0,117,no_hot_key,","));
        CASE nr OF
                1:      BEGIN
                                select_menu^.add_button(@b[1],TRUE);
                                handle:=show_big_alert(GetMaxY                  div
2,tr,@b[1],NIL,NIL,first_bold);
                        END;
                2:      BEGIN
                                select_menu^.add_button(@b[1],TRUE);
                                select_menu^.add_button(@b[2],TRUE);
                                handle:=show_big_alert(GetMaxY                  div
2,tr,@b[1],@b[2],NIL,first_bold);
                        END;
                3:      BEGIN
                                select_menu^.add_button(@b[1],TRUE);
                                select_menu^.add_button(@b[2],TRUE);
                                select_menu^.add_button(@b[3],TRUE);
                                handle:=show_big_alert(GetMaxY                  div
2,tr,@b[1],@b[2],@b[3],first_bold);
                        END;
                ELSE  handle:=show_big_alert(GetMaxY div 2,tr,NIL,NIL,NIL,first_bold);
        END;
        IF nr>=1 THEN BEGIN
                select_menu^.set_leave(TRUE);
                select_menu^.direct_call;
        END;
        hide_alert(handle);
        Dispose(select_menu);
```

```
        select_menu:=old_menu;
        big_alert:=alert_number;
END;



{-----------------------------------------------------}
{ Nun die Funktionen die eine Auswahlbox erstellen: }
{-----------------------------------------------------}

FUNCTION save_number( x,y : LongInt ; nr : WORD): BOOLEAN;
BEGIN
        selected_number:=nr;
        save_number:=TRUE;
END;

PROCEDURE create_select( var menptr : Menu_Button_PTR ;  var strg ;
                                                last_strg, x0, y0 : WORD; var titel);
VAR
        anz_char,
        dx, dy,
        d, temp : WORD;
        hot_key : char;
BEGIN
        anz_char:=Length(STRING(titel))+1;
        FOR d:=1 TO last_strg DO BEGIN
                temp:=Length( Select_Array( strg )[d] )+1;
                IF anz_char < temp THEN anz_char := temp;
        END;
        dx:=(anz_char+3)*char_width;
        dy:=char_height;
        IF x0>GetMaxX-dx-40 THEN
                x0:=GetMaxX-dx-40;
        IF y0>GetMaxy-(last_strg+4)*dy THEN
                y0:=GetMaxy-(last_strg+4)*dy;
```



```
        IF MaxAvail < (SizeOf(Menu_Button)+last_strg*SizeOf(Order_Button)) THEN
BEGIN
                show_help(105);
                halt(1);
        END;


        menptr:=New(Menu_Button_PTR,init(x0-8,y0-8,dx+16,16+(last_strg+2)*dy,

        col_main,col_cs_frame,col_main,88,CHAR(0),'Wählen Sie',STRING(titel)));
        FOR d:=1 TO last_strg DO BEGIN
                hot_key:=char(integer('a')-1+d);
                menptr^.add_button(New(                    Order_Button_PTR,
Init(x0,y0+(d+1)*dy,dx,dy,al_col,al_col,al_bk,88,
                        hot_key,'Bitte wählen sie',hot_key+': '+Select_Array( strg
)[d],save_number)),TRUE);
        END;
        menptr^.all_dynam:=TRUE;   { damit wird der MenButton und Anhänger löschbar
}
END;

FUNCTION  handle_select(menptr : Menu_Button_PTR): WORD;
VAR
        depp            : BOOLEAN;
        ems_handle      : INTEGER;
BEGIN
        selected_number:=0;
        WITH menptr^ DO BEGIN
                ems_handle:=save_big_screen(x,y,dx,dy);
                show;
                hide_mouse;
                SetColor(col_select_titel);
                f_OutTextXY(x+dx div 2,y+char_height, CenterText, CenterText,sign_text
);
                show_mouse;
                direct_call;
                hide;
                done;
        END;
```

```
        restore_big_screen(ems_handle);
        handle_select:=selected_number;
END;


FUNCTION do_edit(var strg ; str_len, x, y : WORD; ed_mode : ed_m_type) : EDEXCODE;
CONST
        tabelle : ARRAY[0..15] OF WORD =(0,0,15,0,0,0,0,15,0,15,15,15,15,15,15,15);
VAR
        d,f, dummy,
        hundreds, old_hunny, tx, ty   : WORD;
        s         : STRING;
        temp, cursor,
        c         : CHAR;
        cx : INTEGER;
        continue,
        escape : BOOLEAN;
        mr        : mouse_rec;


BEGIN
        s:=STRING(strg);
        FOR d:= str_len DOWNTO length(s) DO
                s:=s+' ';
        show_status('Bitte geben sie das Gewünschte ein');
        show_info_line('Korrektur mit Backspace <--, Abbruch mit ESC, Beenden mit RETURN <-+');
        SetFillStyle(SolidFill, col_input_back);
        SetColor(col_input);
        f_OutTextXY(x,y,LeftText,TopText, s);
        continue:=TRUE;
        escape:=FALSE;
        WHILE continue DO BEGIN
                get_mouse(mr);
                SetFillStyle(SolidFill, col_input_back);
                SetColor(Black);
                SetTextJustify(LeftText,TopText);
                GetTime(dummy,dummy,dummy,hundreds);
```



```
                IF (hundreds div 50) <> (old_hunny div 50) THEN BEGIN
                        old_hunny:=hundreds;
                        hide_mouse;
                        SetWriteMode(XORPut);
                        SetColor(col_input_back);
                        SetLineStyle(SolidLn,0,ThickWidth);
                        cx:=x+d*char_width;
                        Line(cx+char_width,y+char_height-3,cx,y+char_height-3);
                        SetLineStyle(SolidLn,0,NormWidth);
                        SetColor(col_input);
                        SetWriteMode(NormalPut);
                        show_mouse;
                END;
                IF KeyPressed THEN BEGIN
                        c:=ReadKey;
                        hide_mouse;
                        CASE c OF
                                CHAR(0): BEGIN
                                                c:=ReadKey;
                                                CASE c OF
                                                        CHAR(59) : BEGIN
                                                                show_help(116);
                                                                SetColor(col_input);
                                                                SetWriteMode(NormalPut);
                                                        END;
{$IFDEF extras}                                         CHAR(66) : BEGIN
                                                                edit_document;
                                                        END;
                                                        CHAR(68) : BEGIN

        d:=save_big_screen(0,0,GetMaxX,GetMaxY);
                                                                FOR tx:=0 TO GetMaxX DO
BEGIN
                                                                        FOR ty:=0 TO GetMaxY
DO BEGIN

        PutPixel(tx,ty,tabelle[GetPixel(tx,ty)]);
                                                                END;END;
```

```
                                        REPEAT
                                        UNTIL ReadKey=CHAR(27);
                                        restore_big_screen(d);
                                        c:=CHAR(0);
                                    END;
                                    CHAR(67): BEGIN

      d:=save_big_screen(0,0,GetMaxX,GetMaxY);
                                        FOR tx:=0 TO GetMaxX DO
BEGIN
                                            FOR ty:=0 TO GetMaxY
DO BEGIN
                                                PutPixel(tx,ty,15-
tabelle[GetPixel(tx,ty)]);
                                        END;END;
                                        REPEAT
                                        UNTIL ReadKey=CHAR(27);
                                        restore_big_screen(d);
                                        c:=CHAR(0);
                                    END;
{$ENDIF}                            CHAR(80) : BEGIN
                                        do_edit:=cdown;
                                        continue:=FALSE;
                                    END;
                                    CHAR(72) : BEGIN
                                        do_edit:=cup;
                                        continue:=FALSE;
                                    END;
                                    CHAR(77) : BEGIN
                                        do_edit:=cright;
                                        continue:=FALSE;
                                    END;
                                    CHAR(75) : BEGIN
                                        do_edit:=cleft;
                                        continue:=FALSE;
                                END;END;
                            END;
                        CHAR(27): BEGIN
```



```
                                        continue:=FALSE;
                                        escape:=TRUE;
                                        do_edit:=cancel;
                                    END;
                                CHAR(8): BEGIN
                                        IF(d>0) THEN BEGIN
                                            s[d]:=' ';
                                            DEC(d);

      Bar(x,y,x+(str_len+1)*char_width,y+char_height);
                                        END;
                                    END;
                                CHAR(13): BEGIN
                                        continue:=FALSE;
                                        escape:=FALSE;
                                        do_edit:=ok;
                                    END;
                                CHAR(9): BEGIN
                                        continue:=FALSE;
                                        escape:=FALSE;
                                        do_edit:=tab;
                                    END;
                                ELSE BEGIN
                                    CASE ed_mode OF
                                    ascii :
                                        IF (c >= Char(32) ) AND ( c <= char(255))
                                                                        AND
(d<=str_len-1) THEN BEGIN
                                            INC(d);
                                            s[d]:=c;

      Bar(x,y,x+(str_len+1)*char_width,y+char_height);
                                        END;
                                    numbers :
                                        IF (c >= '0' ) AND ( c <= '9')
                                                                        AND
(d<=str_len-1) THEN BEGIN
                                            INC(d);
```

```
                                        s[d]:=c;

Bar(x,y,x+(str_len+1)*char_width,y+char_height);
                                    END;
                                allchars :
                                    IF (d<=str_len-1) THEN BEGIN
                                        INC(d);
                                        s[d]:=c;

Bar(x,y,x+(str_len+1)*char_width,y+char_height);
                                    END;
                                END;


                        END;
                    END;
                    IF NOT continue THEN BEGIN
                        Bar(x,y,x+(str_len+1)*char_width,y+char_height);
                    END;

                    f_OutTextXY(x,y,Lefttext,toptext,s);
                    show_mouse;
            END;
            IF mr.right THEN BEGIN
                    escape:=TRUE;
                    continue:=FALSE;
                    do_edit:=cancel;
            END;
            IF mr.left THEN BEGIN
                    escape:=FALSE;
                    continue:=FALSE;
                    do_edit:=mouse_clicked;
                    hide_mouse;
                    Bar(x,y,x+(str_len+1)*char_width,y+char_height);
                    f_OutTextXY(x,y,Lefttext,toptext,s);
                    show_mouse;
            END;
END;
```



```
        IF NOT escape THEN BEGIN
                STRING(strg):='';
                FOR f:=1 TO d DO
                        STRING(strg):=STRING(strg)+s[f];
        END;
END;

PROCEDURE input(var what_to_do; var strg ; str_len:WORD);
VAR
        x,y,dx,dy, cx, len : WORD;
        ems_handle     : INTEGER;
        exit_code : EDEXCODE;
        ed : input_rec;
BEGIN
        ed.double_flag:=FALSE;
        ed.titel:='Eingabe';
        ed.head_line:='';
        ed.base_line:='';
        ed.wtd[1]:=STRING(what_to_do);
        ed.data[1]:=STRING(strg);
        ed.dlen[1]:=str_len;
        ed.dtyp[1]:=ascii;
        ed.last_edit:=1;
        exit_code:=handle_input(ed);
        IF exit_code=ok THEN
                STRING(strg):=ed.data[1];
END;

PROCEDURE input_number(var what_to_do; var number : LONGINT);
VAR
        strg : STRING[10];
        d : WORD;
        value : INTEGER;
        exit_code : EDEXCODE;
        ed : input_rec;
BEGIN
```

```
        ed.double_flag:=FALSE;
        STR( number, strg );
        ed.titel:='Eingabe';
        ed.head_line:='';
        ed.base_line:='';
        ed.wtd[1]:=STRING(what_to_do);
        ed.data[1]:=strg;
        ed.dlen[1]:=6;
        ed.dtyp[1]:=numbers;
        ed.last_edit:=1;
        exit_code:=handle_input(ed);
        IF exit_code=ok THEN BEGIN
                strg:=ed.data[1];
                VAL( ed.data[1], value, d);
                IF d=0 THEN number:=value;
        END;
END;

PROCEDURE input2_numbers(var what_to_do, wtd2; var number1, number2 :
LONGINT);
VAR
        strg : STRING[10];
        d : WORD;
        value : LONGINT;
        exit_code : EDEXCODE;
        ed : input_rec;
BEGIN
        ed.double_flag:=FALSE;
        ed.titel:='Eingabe';
        ed.head_line:='';
        ed.base_line:='';
        ed.wtd[1]:=STRING(what_to_do);
        STR( number1, strg );
        ed.data[1]:=strg;
        ed.dlen[1]:=6;
        ed.dtyp[1]:=numbers;
        ed.wtd[2]:=STRING(wtd2);
```



```
        STR( number2, strg );
        ed.data[2]:=strg;
        ed.dlen[2]:=6;
        ed.dtyp[2]:=numbers;
        ed.last_edit:=2;
        exit_code:=handle_input(ed);
        IF exit_code=ok THEN BEGIN
                VAL( ed.data[1], value, d);
                IF d=0 THEN number1:=value;
                VAL( ed.data[2], value, d);
                IF d=0 THEN number2:=value;
        END;
END;

FUNCTION handle_input(var ed : input_rec): EDEXCODE;
VAR
        d,f,x,y,dx,dy, len, len2, lent : WORD;
        ems_handle,
        val_code        : INTEGER;
        temp : REAL;
        exit_code       : EDEXCODE;
        strg, strg2     : STRING;
        dbl, second     : BOOLEAN;
        mr                      : mouse_rec;
BEGIN
        dbl:=ed.double_flag;
        lent:=Length(ed.titel)*2;                       { wegen bold = Fettschrift }
        IF lent < Length(ed.head_line) THEN len := Length(ed.head_line);
        IF lent < Length(ed.base_line) THEN len := Length(ed.base_line);
        len:=0;
        len2:=0;
        FOR d:=1 TO ed.last_edit DO BEGIN
                IF (len<Length(ed.wtd[d])) THEN len:=Length(ed.wtd[d]);
                IF (len2 < ed.dlen[d]) THEN len2:=ed.dlen[d];
        END;
        IF len>30 THEN len :=30;
```

```
    IF dbl THEN len2:=2*len2+2;
    IF len2>38 THEN len2:=38;
    dy:=(ed.last_edit+5)*char_height;
    IF Length(ed.base_line)>0 THEN INC(dy,(3*char_height)DIV 2);
    IF lent<(len+len2+4) THEN lent:=len+len2+4;
    dx:=( lent+2 ) * char_width;
    x:=(GetMaxX - dx) div 2 ;
    y:=(GetMaxy - dy) div 2;
    ems_handle:=save_big_screen( x , y , dx , dy);
    SetFillStyle(SolidFill, col_input_back);
    hide_mouse;
    Bar(x,y,x+dx,y+dy);
    SetColor(al_bk);
    Rectangle(x+1,y+1,x+dx-1,y+dy-1);
    f_text_bold:=TRUE;              { wirkt nur einmal }
    f_OutTextXY(x+ dx div 2, y+ char_height,
                                    CenterText, TopText, ed.titel);
    f_OutTextXY(x+ (char_width*5) div 2, y+ 2*char_height,
                                    LeftText, TopText, ed.head_line);
    IF Length(ed.base_line)>0 THEN
        f_OutTextXY(x+ dx div 2, y+ dy- 2*char_height,
                                    CenterText, CenterText, ed.base_line);
    FOR d:=1 TO ed.last_edit DO BEGIN
        f_OutTextXY(x+char_width*(len+2), y+ char_height*(d+3),
                                    RightText, TopText, ed.wtd[d]);
    END;
    SetColor(Black);
    FOR d:=1 TO ed.last_edit DO BEGIN
        f_OutTextXY(x+char_width*(len+3), y+ char_height*(d+3),
                                    LeftText, TopText, ed.data[d]);
        IF dbl THEN f_OutTextXY(x+char_width*(len+4+(len2 div 2)), y+
char_height*(d+3),
                                    LeftText, TopText, ed.data2[d]);

    END;
    show_mouse;
    d:=1;
```



```
    second:=FALSE;
    REPEAT
        IF NOT second THEN
            exit_code:=do_edit( ed.data[d], ed.dlen[d], x + char_width*(len+3),
                                    y + (d+3)* char_height,ed.dtyp[d])
        ELSE
            exit_code:=do_edit( ed.data2[d], ed.dlen[d], x +
char_width*(len+4+len2 div 2),
                                    y + (d+3)* char_height,ed.dtyp[d]);
        IF dbl THEN BEGIN
            IF (exit_code=cleft) THEN BEGIN
                second:=NOT second;
                IF second THEN exit_code:=cup;
            END;
            IF (exit_code=cright) THEN BEGIN
                second:=NOT second;
                IF NOT second THEN exit_code:=cdown;
            END;
        END;
        IF ((exit_code=tab) OR (exit_code=cdown)) THEN BEGIN
            IF d<ed.last_edit THEN INC(d)
            ELSE d:=1;
        END;
        IF (exit_code=cup) THEN
            IF d>1 THEN DEC(d)
            ELSE d:=ed.last_edit;

        IF (exit_code=mouse_clicked) THEN BEGIN
            get_mouse(mr);
            FOR f:=1 TO ed.last_edit DO BEGIN
                IF (mr.y>=y + (f+3)*char_height) AND
                        (mr.y<=y + (f+4)*char_height) THEN BEGIN
                    d:=f;
                    second:=FALSE;
                    IF dbl AND (mr.x >= x + char_width*(len+4+len2 div
2)) THEN
                        second:=TRUE;
```

```
                        END; {if}
                    END; {for}
                END;
        UNTIL (exit_code=ok) OR (exit_code=cancel);
        restore_big_screen(ems_handle);
        handle_input:=exit_code;
END;

PROCEDURE edit_document;
VAR
        handle, full_handle :INTEGER;
        d, code :WORD;
        ir : input_rec;
        tr : text_rec;
        mr : mouse_rec;
        x : EDEXCODE;
        strg : STRING;
BEGIN
        ir.titel:='Beschriftung';
        ir.head_line:='Für dieses Bild';
        ir.base_line:='RETURN/Mauspos./beide Tasten/prt/r oder nix';
        ir.wtd[1]:='Wieviele Zeilen';
        ir.data[1]:='5';
        ir.dtyp[1]:=numbers;
        ir.dlen[1]:=2;
        FOR d:=2 TO max_edit DO BEGIN
                Str(d-1,strg);
                ir.wtd[d]:=strg+'. Text';
                ir.data[d]:='';
                ir.dtyp[d]:=allchars;
                ir.dlen[d]:=36;
        END;
        ir.double_flag:=FALSE;
        ir.last_edit:=15;
        REPEAT
```



```
                full_handle:=save_big_screen(0,0,GetMaxX,GetMaxY);
                x:=handle_input(ir);
                restore_big_screen(full_handle);
                Val(ir.data[1],d,code);
                IF d<=0 THEN d:=1;
                IF d>14 THEN d:=14;
                tr.last_string:=d+2;
                REPEAT
                        tr.strg[d+2]:=ir.data[d+1];
                        DEC(d);
                UNTIL d=0;
                tr.strg[1]:='Lernen...';
                tr.strg[2]:='';
                al_bk:=col_input_back;
                al_col:=Black;
                get_mouse(mr);
                handle:=show_big_alertxy(mr.x, mr.y, tr, NIL,NIL,NIL, TRUE);
                REPEAT
                        REPEAT
                                get_mouse(mr);
                        UNTIL mr.left;
                        IF NOT mr.right THEN BEGIN
                                IF handle>0 THEN hide_alert(handle);
                                handle:=show_big_alertxy(mr.x,  mr.y,  tr,  NIL,NIL,NIL,
TRUE);
                        END;
                        REPEAT
                                get_mouse(mr);
                        UNTIL NOT mr.left;
                UNTIL mr.right;
                Sound(400);
                delay(1000);
                NoSound;
                Repeat
                Until KeyPressed;
                d:=INTEGER(ReadKey);
                hide_alert(handle);
```

```
                al_bk:=col_help_back;
                al_col:=col_help;
        UNTIL d<>INTEGER('r');
END;

END.
```


E.4. Die Unit F_ANALYS

```
unit f_analyse;
{$F+}{$O+}
INTERFACE
USES gui_def;
        FUNCTION set_higru(x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
        FUNCTION toggle_higru(x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
        PROCEDURE init_higru;
        FUNCTION calc_spec( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION calc_kurve( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION show_parameter( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION differenziere( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION integriere( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION alles_tabelle( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        PROCEDURE koeff_to_tr(VAR tr : text_rec);

IMPLEMENTATION
USES f_global, f_gr_def, f_graph, f_help, f_calc, f_mess,

        gui, gui_tool, Graph;



FUNCTION set_higru(x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
VAR
        d, delta : LONGINT;
BEGIN
        delta:=higru_window.dx;
        higru_window.disable;
        FOR d:=0 TO prec DO BEGIN
                higru_datas^[d]:=work_data^[(d*(used_data-1)) DIV prec];
        END;
        higru_window.set_data(higru_datas,1,prec);
        higru_window.enable;
        higru_set:=TRUE;
        higru_shown:=TRUE;
        set_higru:=TRUE;
```

```
END;

FUNCTION toggle_higru(x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
BEGIN
        IF higru_set THEN BEGIN
                IF higru_shown THEN
                        higru_window.disable
                ELSE
                        higru_window.enable;
                higru_shown:=NOT higru_shown;
                toggle_higru:=TRUE;
        END ELSE
                toggle_higru:=set_higru(0,0,0);
END;

PROCEDURE init_higru;
BEGIN
        New(higru_datas);
        higru_set:=FALSE;
        higru_shown:=FALSE;
END;

FUNCTION calc_spec( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        yes_he_is : BOOLEAN;
BEGIN
        mark_to_set:=0;
        IF (data_mark[1]=0) AND (data_mark[2]=0) THEN BEGIN
                show_help(85);
                show_help(60);
        END
        ELSE BEGIN
                hide_all;
                mess_window.set_mark(3,data_mark[1],FALSE); { nur wg TRUE }
                mess_window.set_mark(4,data_mark[2],FALSE);
                calculate_spec;
```



```
                calc_intervall:=Abs(data_mark[2]-data_mark[1]);
                yes_he_is:=display_spec(0,0,0);
        END;
        calc_spec:=TRUE;
END;

FUNCTION calc_kurve( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        egal : BOOLEAN;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        mark_to_set:=0;
        mess_window.hide;
        egal:=display_spec(0,0,0);
        calculate_kurve;
        mess_window.show;
END;
        calc_kurve:=TRUE;
END;

FUNCTION show_parameter( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
CONST
        mille : EXTENDED = 1000;
VAR
        tr : text_rec;
        strg,temp : STRING;
        calc : LONGINT;
        calc2 : EXTENDED;
BEGIN
        tr.strg[1]:='Information';
        tr.strg[2]:='';
        tr.strg[3]:='Messung:';
        Str(used_data:5,strg);
        tr.strg[4]:='    Anzahl der Messwerte: '+strg;
        Str(Round(mess_freq):5,strg);
        tr.strg[5]:='       bei Messfrequenz: '+strg+' Hz';
```

```
IF Round(mess_freq)>0 THEN
        Str(Round(1000.0*used_data/mess_freq):5,strg)
ELSE
        strg:=' unbekannt';
tr.strg[6]:='       und bei Messdauer: '+strg+' ms';
tr.strg[7]:=' Extremwerte:';
Str(mess_max:5,strg);
Str(mess_min:5,temp);
tr.strg[8]:='  Maximum: '+strg+' Minimum: '+temp+' (in mV)';
tr.strg[9]:='';
tr.strg[10]:='Markierung:';
Str(data_mark[1]:5,strg);
Str(data_mark[2]:5,temp);
tr.strg[11]:='  Marke 1: '+strg+'   Marke 2: '+temp;
IF Round(mess_freq)>0 THEN BEGIN
        calc:=Round(((data_mark[2]-data_mark[1])*mille) / mess_freq);
        Str(calc:5,strg);
        tr.strg[12]:='  Intervallänge zw.den Marken: '+strg+' ms';
        calc2:=mess_freq / (data_mark[2]-data_mark[1]);
        Str(calc2:8:2,strg);
        tr.strg[13]:='  Grundfrequenz zw. Marken: '+strg+' Hz';
END ELSE BEGIN
        tr.strg[12]:='  Intervallänge zw.den Marken: unbekannt';
        tr.strg[13]:='  Grundfrequenz zw. Marken:    unbekannt';
END;
tr.strg[14]:='';
tr.strg[15]:='Spektrum:';
Str(used_spec:5,strg);
tr.strg[16]:='  Oberwellen: '+strg;
calc2:=time_to_calc;
calc2:=calc2 / 100;
Str(calc2:5:2,strg);
tr.strg[17]:='  Berechnungszeit: '+strg+' sek.';
calc:=spec_step;
Str(calc:3,strg);
tr.strg[18]:='  Anzahl der markierten Perioden: '+strg;
```



```
        calc:=MaxAvail;
        Str(calc:3,strg);
        tr.strg[19]:='Freier Speicher: '+strg+' kByte';
        tr.last_string:=19;
        show_info_line('Um die Information zu verlassen: [ Gelesen ] anklicken oder
RETURN drücken');
        calc:=big_alert(tr,' Gelesen ',TRUE);
        show_parameter:=TRUE;
END;

FUNCTION integriere( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        max, temp, faktor : EXTENDED;
        d         : INTEGER;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        hide_all;
        max:=0;
        FOR d:=1 TO used_spec DO BEGIN
                temp:=spektrum^[d];
                temp:=temp / d;
                IF temp>max THEN max:=temp;
        END; {for}
        faktor:=10000.0 / max;
        spektrum^[0] := 0;
        sin_koeff^[0]:= 0;
        cos_koeff^[0]:= 0;
        FOR d:=1 TO used_spec DO BEGIN
                spektrum^[d] := Round(faktor * spektrum^[d]  / d);
                temp:= cos_koeff^[d];
                cos_koeff^[d]:=-Round(faktor * sin_koeff^[d] / d);
                sin_koeff^[d]:= Round(faktor * temp / d);
                IF (spektrum^[d]>=10) THEN BEGIN
                        IF (sin_koeff^[d]<>0)  THEN  BEGIN
                                temp:=-ArcTan(Abs(cos_koeff^[d]/sin_koeff^[d]))*180/Pi;
                                IF sin_koeff^[d]<0 THEN temp:=180-temp;
                                IF cos_koeff^[d]<0 THEN temp:=-temp;
```

```
                END
                ELSE
                        temp:=90*sign(cos_koeff^[d]);
                WHILE temp<0 DO
                        temp:=360+temp;
        END
        ELSE
                temp:=-1;
        IF temp*10>maxint THEN
                temp:=maxint div 10;  { reiner Schutz hoffentlich ohne Anwendung
(for test only) }
        phase^[d]:=Round(temp*10);
    END; {for}
    scale_spec;
    integriere:=display_spec(0,0,0);
END;
    integriere:=TRUE;
END; {integriere}

FUNCTION differenziere( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
    max, temp, faktor : EXTENDED;
    d       : INTEGER;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
    hide_all;
    max:=0;
    FOR d:=1 TO used_spec DO BEGIN
        temp:=spektrum^[d];
        temp:=temp * d;
        IF temp>max THEN max:=temp;
    END; {for}
    faktor:=5000.0 / max;
    spektrum^[0] := 0;
    sin_koeff^[0]:= 0;
    cos_koeff^[0]:= 0;
    FOR d:=1 TO used_spec DO BEGIN
```



```
        spektrum^[d] := Round(faktor * spektrum^[d]  * d);
        temp:= cos_koeff^[d];
        cos_koeff^[d]:= Round(faktor * sin_koeff^[d] * d);
        sin_koeff^[d]:=-Round(faktor * temp * d);
        IF (spektrum^[d]>=10) THEN BEGIN
            IF (sin_koeff^[d]<>0)  THEN  BEGIN
                temp:=-ArcTan(Abs(cos_koeff^[d]/sin_koeff^[d]))*180/Pi;
                IF sin_koeff^[d]<0 THEN temp:=180-temp;
                IF cos_koeff^[d]<0 THEN temp:=-temp;
            END
            ELSE
                temp:=90*sign(cos_koeff^[d]);
            WHILE temp<0 DO
                temp:=360+temp;
        END
        ELSE
            temp:=-1;
        IF temp*10>maxint THEN
            temp:=maxint div 10;  { reiner Schutz hoffentlich ohne Anwendung
(for test only) }
        phase^[d]:=Round(temp*10);
    END; {for}
    scale_spec;
    differenziere:=display_spec(0,0,0);
END;
    differenziere:=TRUE;
END;

FUNCTION alles_tabelle( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
    tr : text_rec;
    d  : WORD;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
    koeff_to_tr(tr);
    d:=big_alert(tr,'Gelesen',TRUE);
END;
```

```
        alles_tabelle:=TRUE;
END;

PROCEDURE koeff_to_tr(VAR tr : Text_rec);
VAR
        d,delta : WORD;
        strg : STRING[80];
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        tr.strg[1]:='Tabelle der Koeffizienten';
        tr.strg[2]:='';
        tr.strg[3]:=' Nr: Ampl., Phase -Cosinus, Sinus    | '+
                                ' Nr: Ampl., Phase -Cosinus, Sinus';
        FOR d:=0 TO 19 DO BEGIN
                tr.strg[d+4]:='';
                delta:=0;
                WHILE delta<=20 DO BEGIN
                        Str(d+delta:3,strg);
                        tr.strg[d+4]:=tr.strg[d+4]+strg+':';
                        IF (phase^[d+delta]>=0) AND (d+delta <used_spec) THEN BEGIN
                                Str(spektrum^[d+delta]:6,strg);
                                tr.strg[d+4]:=tr.strg[d+4]+strg+',';
                                Str((phase^[d+delta] DIV 10):5,strg);
                                tr.strg[d+4]:=tr.strg[d+4]+strg+'° - ';

                                Str(sin_koeff^[d+delta]:6,strg);
                                tr.strg[d+4]:=tr.strg[d+4]+strg+',';
                                Str(cos_koeff^[d+delta]:6,strg);
                                tr.strg[d+4]:=tr.strg[d+4]+strg+'    | ';
                        END         ELSE         IF         (d+delta<used_spec)         THEN
tr.strg[d+4]:=tr.strg[d+4]+'    0,    -° -      0,     0    | '
                                        ELSE tr.strg[d+4]:=tr.strg[d+4]+'                              |
';

                        INC(delta,20)
        END;END;
        tr.last_string:=23;
```



```
END;
END;

END.
```

E.5. Die Unit F_CALC

```
UNIT f_calc;

{$F+} {$O+}
INTERFACE
USES f_gr_def;
     PROCEDURE calculate_spec;
     PROCEDURE calculate_kurve;
     PROCEDURE calc_init;
     PROCEDURE scale_mess;
     PROCEDURE scale_spec;
     FUNCTION time_to_calc:LONGINT;
     FUNCTION max_spec_amplitude:INTEGER;
     FUNCTION sign ( x:Extended):EXTENDED;

IMPLEMENTATION
USES f_global, f_graph, gui, gui_tool, f_mouse, CRT, dos, f_help;
VAR
     time_needed : LONGINT;
     max_spec_ampl : INTEGER;


FUNCTION time_to_calc:LONGINT;
BEGIN
     time_to_calc:=time_needed;
END;

FUNCTION max_spec_amplitude:INTEGER;
BEGIN
     max_spec_amplitude:=max_spec_ampl;
END;

FUNCTION FMSin( x: EXTENDED): EXTENDED;              { übernommen aus c't
      4/92 }
BEGIN                                               { Seite 261 }
     ASM fld x        END;                                          { Die
       Sinus(Cosinusfunktion }
     InLine ($d9 / $fe );                             { wird ohne Umwege
       aufgerufen }
     ASM fstp x       END;                               { aber nur bei
       387/486 }
     FMSin := x;
END;

FUNCTION FMCos( x: EXTENDED): EXTENDED;
BEGIN
     ASM fld x        END;
     InLine ($d9 / $ff );
     ASM fstp x       END;
     FMCos := x;
END;

FUNCTION sign ( x:Extended):EXTENDED;
BEGIN
     IF x<0 THEN sign:=-1;
     IF x=0 THEN sign:=0;
     IF x>0 THEN sign:=1;
END;

PROCEDURE calculate_spec;
VAR
     ch : CHAR;
     t, faktor, s, c, sin_summe, cos_summe, wert1, wert2 : EXTENDED;
     hour,min,sec,hundreds,
     harmon, intervall,
     von, bis, d : WORD;
```



```
     temp : EXTENDED;
     maximum : data_type;
     mc : mouse_rec;
     text : STRING[40];
     t1,t2,t3 : STRING;
     alert_handle : INTEGER;
     calc_hour : LONGINT;
     aktuell,gesamt : string[10];
     data : data_array_ptr;
     flag : BOOLEAN;
BEGIN
     hide_all;
     mess_window.set_mark(3,data_mark[1],FALSE); { nur wg TRUE }
     mess_window.set_mark(4,data_mark[2],FALSE);
     IF data_mark[1] <= data_mark[2] THEN BEGIN
       von:=data_mark[1];
       bis:=data_mark[2];
     END
     ELSE BEGIN
       von:=data_mark[2];
       bis:=data_mark[1];
     END;
     intervall:=bis-von;
     FOR d:=0 TO max_spect DO BEGIN
       spektrum^[d]:=0;
       phase^[d]:=0;
     END;
     maximum:=0;
     IF intervall >= 10*spec_step THEN BEGIN
       IF (used_spec > intervall div 2 div spec_step) THEN
          used_spec:=intervall div 2 div spec_step;
       GetTime(hour,min,sec,hundreds);
       calc_hour:=hour;
       time_needed:=hundreds+100*(sec+60*(min+60*calc_hour));
       t1:='Berechnung des Spektrums, bitte etwas Geduld';
       Str(intervall,t2);
       Str(used_spec,t3);
       t3:='Intervall: '+t2+'  Oberwellen: '+t3;
       t2:='Information:';
       alert_handle:=show_alert(t1,t2,t3);
       show_info_line('Abbruch mit ESC');
       Str(used_spec,gesamt);
       gesamt:='/'+gesamt;

       data:=work_data;
       harmon:=0;
       WHILE harmon <= used_spec DO BEGIN
          IF (harmon mod 10)=0 THEN BEGIN
               Str(harmon,aktuell);
               text:='Berechnung des Fourierkoeff.:'+aktuell+gesamt;
               show_status(text);
               IF KeyPressed THEN BEGIN
                    ch:=ReadKey;
                    IF ch=CHAR(27) THEN BEGIN
                         d:=used_spec;
                         used_spec:=harmon;
                         harmon:=d;
                    END;
               END;
          END;

          faktor:= 2 * Pi * harmon / intervall * spec_step;
          sin_summe:= 0;
          cos_summe:= 0;
          t:=0;
          IF Test8087>=3 THEN    { Bedingung für 387 }
               FOR d:= 0 TO intervall-1 DO BEGIN
                    t:= d * faktor;
```

```
                wert1:=data^[von+d];
                sin_summe:= sin_summe + FMsin(t)*wert1;
                cos_summe:= cos_summe + FMcos(t)*wert1;
            END
        ELSE
            FOR d:= 0 TO intervall-1 DO BEGIN
                t:= d * faktor;
                wert1:=data^[von+d];
                sin_summe:= sin_summe + sin(t)*wert1;
                cos_summe:= cos_summe + cos(t)*wert1;
            END;

        sin_koeff^[harmon]:=Round(sin_summe / intervall * 2);
        cos_koeff^[harmon]:=Round(cos_summe / intervall * 2);
        temp:=sqrt(sin_summe*sin_summe+cos_summe*cos_summe)  /  intervall
      *2;
        IF harmon=0 THEN BEGIN
            cos_koeff^[0]:=Round(cos_summe/ intervall);
        END;
        IF (temp>maximum) THEN maximum:=Round(temp);
        spektrum^[harmon]:=Round(temp);
        IF (temp>=10) THEN BEGIN
            IF (sin_summe<>0)  THEN  BEGIN
                temp:=-ArcTan(Abs(cos_summe/sin_summe))*180/Pi;
                IF sin_summe<0 THEN temp:=180-temp;
                IF cos_summe<0 THEN temp:=-temp;
            END
            ELSE
                temp:=90*sign(cos_summe);
            WHILE temp<0 DO
                temp:=360+temp;
        END
        ELSE BEGIN
            temp:=-1;
            sin_koeff^[harmon]:=0;
            cos_koeff^[harmon]:=0;
            spektrum^[harmon]:=0;
        END;
        IF temp*10>maxint THEN
            temp:=maxint  div  10;    {  reiner  Schutz  hoffentlich  ohne
      Anwendung (for test only) }
        phase^[harmon]:=Round(temp*10);

        INC(harmon); {!!!}
      END;
      hide_alert(alert_handle);
      GetTime(hour,min,sec,hundreds);
      calc_hour:=hour;
      time_needed:=(hundreds+100*(sec+60*(min+60*calc_hour)))-
      time_needed;
    END
    ELSE
      show_help(120);
    scale_spec;
    max_spec_ampl:=Round(maximum*1.1);
    spec_calculated:=TRUE;                          { geschafft }
    flag:=set_spec_koeff_mode(0,0,0);
END;


PROCEDURE calculate_kurve;
VAR
    d,f,z,anfang : WORD;
    delta, x,
    temp,faktor,
    to_rad, korrektur : EXTENDED;
    min,max : EXTENDED;
    ch : CHAR;
```



```
    aktuell, gesamt,text : STRING[40];
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
    min:=0;
    max:=0;
    korrektur:=1;
    anfang:=data_mark[1];
    delta:=anfang;
    faktor:=2*Pi/calc_intervall*spec_step;
    to_rad:=PI/1800;
    Str(calc_intervall,gesamt);
    max:=0;
    { eigentliche Berechnung }
    FOR d:=0 TO calc_intervall DO BEGIN
      temp:=0;
      x:=d*faktor;
      IF (d mod 200)=0 THEN BEGIN
        Str(d,aktuell);
        text:='Berechnung des Wertes:'+aktuell+'/'+gesamt;
        show_status(text);
        IF KeyPressed THEN BEGIN
            ch:=ReadKey;
            IF ch=CHAR(27) THEN BEGIN
                d:=used_data;
                used_data:=d;
            END;
        END;
      END;
      IF test8087>=3 THEN
        FOR f:=0 TO used_spec DO BEGIN
            temp:=temp+sin_koeff^[f]*FMsin(x*f);
            temp:=temp+cos_koeff^[f]*FMcos(x*f);
        END
      ELSE
        FOR f:=0 TO used_spec DO BEGIN
            temp:=temp+sin_koeff^[f]*sin(x*f);
            temp:=temp+cos_koeff^[f]*cos(x*f);
            {temp:=temp+spektrum^[f]*sin(x*f+phase^[f]*to_rad);}
        END;
      IF (temp>14000.0) THEN temp:=14000.0;
      IF (temp<-14000.0) THEN temp:=-14000.0;
      z:=anfang+d;
      WHILE z>=calc_intervall DO DEC(z,calc_intervall);
      REPEAT
        work_data^[z]:=Round(temp);
        INC(z,calc_intervall);
      UNTIL z>used_data;
    END;
    scale_mess;
    mess_window.set_data(work_data,0,used_data);
END;
END;


PROCEDURE calc_init;
BEGIN
    time_needed:=0;
    spec_calculated:=FALSE;
END;

PROCEDURE scale_mess;    { hier werden alle Werte durchgescannt und die
                           Darstellung entsprechend scaliert }
VAR
    min,max : EXTENDED;
    temp, mini, maxi : data_type;
    d : WORD;
BEGIN
    min:=0;
```

```
    max:=0;
    FOR d:=0 TO used_data-1 DO BEGIN
      temp:=work_data^[d];
      IF min>temp THEN min:=temp;
      IF max<temp THEN max:=temp;
      IF higru_set AND higru_shown AND (d<prec) THEN BEGIN
         temp:=higru_datas^[d];
         IF min>temp THEN min:=temp;
         IF max<temp THEN max:=temp;
      END;
    END;
    mess_max:=Round(max);
    mess_min:=Round(min);

    mess_window.set_scale(Round(min*1.1),Round(max*1.1),points);
    higru_window.hide;
    higru_window.set_scale(Round(min*1.1),Round(max*1.1),second_points);
    higru_window.show;
    zoom_window.set_scale(Round(min*1.1),Round(max*1.1),points);
    back_zoom_window.set_scale(Round(min*1.1),Round(max*1.1),dotted);
END;

PROCEDURE scale_spec;    { hier werden alle Werte durchgescannt und die
                           Darstellung entsprechend scaliert }
VAR
    temp,min,max : EXTENDED;
    d : WORD;
BEGIN
    min:=0;
    max:=0;
    FOR d:=0 TO used_spec-1 DO BEGIN
      temp:=spektrum^[d];
      IF max<temp THEN max:=temp;
    END;
    spec_window.set_scale(Round(-0.01*max),Round(max*1.1),lines);
    sin_window.set_scale(Round(-max*1.1),Round(max*1.1),lines);
    cos_window.set_scale(Round(-max*1.1),Round(max*1.1),second_lines);
END;

END.
```


E.6. Die Unit F_CHANGE

```
unit f_change;

{$F+} {$O+}

INTERFACE
    FUNCTION tab_eingabe( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
    FUNCTION new_tab_eingabe( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
    FUNCTION octave_up( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
    FUNCTION octave_down( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
    FUNCTION change_kurve( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
    FUNCTION change_spec( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
    FUNCTION set_spec_used( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
    FUNCTION select_zoom_factor( rel_x, rel_y: LONGINT; nr: WORD) :
BOOLEAN;
    FUNCTION do_select_work_area( rel_x, rel_y: LONGINT; nr: WORD) :
BOOLEAN;

IMPLEMENTATION
uses f_global,f_gr_def, f_graph, gui, gui_def, gui_tool, f_help, Graph,
                                                          f_mouse, f_calc,
f_analyse, f_datei;
VAR
    global_help_number : WORD;

FUNCTION global_hilfe (rel_r, rel_y : LONGINT; nr : WORD):BOOLEAN;
BEGIN
    show_help(global_help_number);
    global_hilfe:=TRUE;
END;



FUNCTION set_spec_used( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
    komment1, komment2, komment3 : STRING;
    d,f,old,olf : LONGINT;
```

```
        jessy : BOOLEAN;
BEGIN
        komment1:='Anzahl der Oberwellen (<1000):';
        d:=used_spec;
        old:=d;
        komment2:='Wieviel markierte Perioden:';
        f:=spec_step;
        olf:=f;
        input2_numbers(komment1,komment2,d,f);
        IF (d> max_spect) THEN d:=max_spect;
        IF (d<20) THEN d:=20;
        used_spec:=d;
        IF (f > 99) THEN f:=99;
        IF (f < 1)  THEN f:=1;
        spec_step:=f;
        IF (olf<>f) OR (old<d) OR (NOT spec_calculated) THEN jessy:=calc_spec(0,0,0);
        hide_all;
        spec_window.set_data(spektrum,0,used_spec-1);
        phases_window.set_data(phase,0,used_spec-1);
        set_spec_used:=display_spec(0,0,0);
END;

FUNCTION select_zoom_factor( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        mr : mouse_rec;
        txt : Select_Array;
        d : WORD;
        menptr : Menu_Button_PTR;
        titel : STRING;
BEGIN
        get_mouse(mr);     { SelectBox erscheint an Mausposition, Abfrage bei create_select
auf Bildschirmränder }
        titel:='Wählen Sie die Lupenvergrösserung';
        txt[1]:='jede Bildschirmspalte ein Meßwert';
        txt[2]:='alle 2 Spalten ein Wert';
        txt[3]:='alle 4 Spalte ein Wert';
        create_select(menptr, txt, 3, mr.x, mr.y,titel);
```



```
        d:=handle_select(menptr);
        dispose(menptr,done);
        CASE d OF
                1: zoom_faktor:=1;
                2: zoom_faktor:=2;
                3: zoom_faktor:=3;
        END;
        select_zoom_factor:=display_zoom(0,0,0);
END;


VAR
        select_menu : Menu_Button_PTR;

FUNCTION end_select (rel_x, rel_y : LONGINT;nr : WORD): BOOLEAN;
BEGIN
        select_menu^.set_leave(FALSE);
        mess_window.force_enabled_flag(FALSE);      {verhindert      Löschen      und
wiederschreiben }
        end_select:=TRUE;
END;

FUNCTION select_work_area_end( rel_x, rel_y :LONGINT; nr:WORD) : BOOLEAN;
VAR
        d_center : INTEGER;
        d : WORD;
        dumm : BOOLEAN;
BEGIN
        d_center:= mess_window.rel_to_data(rel_x);
        IF (used_data>d_center) AND (d_center>800) THEN BEGIN
                mess_window.hide;
                used_data:=d_center;
                mess_window.unselect;
                mess_window.set_reaction(dummy,'Kurve der Messwerte',0);
                mess_window.set_data(work_data,0,used_data);
                mess_window.show;
                select_work_area_end:=end_select(0,0,0);
        END
```

```
        ELSE
                select_work_area_end:=TRUE;
END;

FUNCTION select_work_area_start( rel_x, rel_y :LONGINT; nr:WORD) : BOOLEAN;
VAR
        d_center : INTEGER;
        d : WORD;
BEGIN
        mess_window.hide;
        d_center:= (used_data*rel_x) div 1000;
        IF (used_data>d_center+800) AND (d_center>0) THEN BEGIN
                Dec(data_mark[1],d_center);
                Dec(data_mark[2],d_center);
                used_data:=used_data-d_center;
                FOR d:= 0 to used_data-1 DO BEGIN
                        work_data^[d]:=work_data^[d+d_center];
                END;
                mess_window.set_reaction(select_work_area_end,'Wählen sie
Datenende',1);
                mess_window.unselect;
                mess_window.set_data(work_data,0,used_data);
        END;
        mess_window.show;
        select_work_area_start:=TRUE;
END;

FUNCTION do_select_work_area( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        ch : CHAR;
        d : WORD;
        handle : INTEGER;
        continue ,touched, touched_button,
        old_touched,none : BOOLEAN;
        mr : mouse_rec;
        t1,t2,t3 : STRING;
        helper, exitor : Order_Button_PTR;
```



```
        old_help : WORD;
BEGIN
        mark_to_set:=0;
        old_help:=global_help_number;
        global_help_number:=110;
        exitor:=New(
Order_Button_PTR,init(0,0,9*char_width,char_height,col_help_back,col_help,col_help,11
0,
                                char(27),'Abbruch der
Bereichsauswahl','Abbruch',end_select));
        helper:=New(
Order_Button_PTR,init(0,0,9*char_width,char_height,col_help_back,col_help,col_help,11
0,
                                'h','Auswahl erklären lassen','
Hilfe',global_hilfe));
        select_menu:=New(menu_Button_ptr,init(0,0,0,0,col_help,col_help,col_help_back,
110,
                                no_hot_key,'',''));
        select_menu^.add_button(exitor,TRUE);
        select_menu^.add_button(helper,TRUE);
        select_menu^.add_button(@mess_window,TRUE);

        t1:='Die neuen Daten sind zu unübersichtlich. Wählen Sie daher';
        t2:='aus den Daten mit der Maus den für Sie interessanten Bereich aus.';
        t3:='Klicken Sie dazu im Messfenster erst für Anfang,dann für Ende des Bereichs.';
        hide_all;
        show_info_line('Bitte wählen sie Anfang/Ende nacheinander durch Anklicken,
Abbruch mit ESC');
        handle:=show_alertY(GetMaxY *2 div 3,t1,t2,t3,exitor,helper,NIL);
        mess_window.set_reaction(select_work_area_start,'Wählen sie Datenanfang',1);
        FOR d:=1 TO 4 DO BEGIN
                mess_window.set_mark(d,-1,FALSE);
        END;
        select_menu^.set_leave(TRUE);

        select_menu^.direct_call;

        hide_alert(handle);
        dispose(exitor,done);
```

```
        dispose(helper,done);
        dispose(select_menu,done);
        hide_info_line;
        mess_window.force_enabled_flag(TRUE);
        global_help_number:=old_help;
        IF data_mark[1]>used_data THEN data_mark[1]:=used_data div 2;
        IF data_mark[2]>used_data THEN data_mark[2]:=(used_data*2) div 3;
        data_mark[0]:=used_data div 2;
        save_choix;
        do_select_work_area:=set_first_mark(0,0,0);
END;

FUNCTION new_tab_eingabe( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
VAR
        frequenz : LONGINT;
        tr : text_rec;
        result, d : WORD;
        s : STRING;
BEGIN
        tr.strg[1]:='Information';
        tr.strg[2]:='';
        tr.strg[3]:='Sie haben die Eingabe eines neuen Spektrums gewählt!';
        tr.strg[4]:='Dazu sollten Sie folgendes beachten:';
        tr.strg[5]:='-Die alten Werte des Spektrums werden gelöscht';
        tr.strg[6]:='-Sie können je 20 Werte für Amplitude und Phase eingeben';
        tr.strg[7]:='-Um beliebige Signale produzieren zu können, dürfen Sie';
        tr.strg[8]:='  gleich die Frequenz der Grundschwingung eingeben';
        tr.strg[9]:='Wenn Sie dies alles doch nicht durchführen wollen so ';
        tr.strg[10]:='  wählen Sie jetzt ABBRUCH (auch mit ESC)';
        tr.strg[11]:='';
        tr.strg[12]:='';
        tr.strg[13]:='';
        tr.last_string:=10;
        result:=big_alert(tr,'Durchführen|  Abbruch  ',TRUE);
        IF result=1 THEN BEGIN
                frequenz:=0;
```



```
                s:='Grundfrequenz:';
                input_number(s,frequenz);
                IF (frequenz<>0) AND (mess_freq/frequenz<30000)
                                                        AND
(mess_freq/frequenz>50) THEN BEGIN
                        hide_all;
                        used_spec:=20;
                        FOR d:=0 TO used_spec DO BEGIN
                                spektrum^[d]:=0;
                                phase^[d]:=0;
                                sin_koeff^[d]:=0;
                                cos_koeff^[d]:=0;
                        END; {for}
                        calc_intervall:=Round(mess_freq/frequenz);
                        new_tab_eingabe:=tab_eingabe(0,0,0);
                        IF calc_intervall<used_data/2 THEN BEGIN
                                data_mark[1]:=(used_data-calc_intervall) div 2;
                                data_mark[2]:=used_data-data_mark[1];
                                mess_window.set_mark(3,data_mark[1],FALSE);  {  nur  wg
TRUE }
                                mess_window.set_mark(4,data_mark[2],FALSE);

                        END;
                END ELSE
                        show_help(145);
        END;
        new_tab_eingabe:=TRUE;
END;


FUNCTION tab_eingabe( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
VAR
        jojo : BOOLEAN;
        d, acode, pcode : WORD;
        strg : STRING[10];
        ed : Input_rec;
        exit_code : EDEXCODE;
```

```
        awert, pwert, max : LONGINT;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        ed.titel:='Spektrum';
        ed.head_line:='  Amlitude | Phase';
        ed.base_line:='';
        FOR d:=1 TO 20 DO BEGIN
                Str((d-1):2,strg);
                ed.wtd[d]:='Nr. '+strg+':';
                Str(spektrum^[d-1+spec_begin],strg);
                ed.data[d]:=strg;
                IF phase^[d-1+spec_begin]<0 THEN
                        strg:='-'
                ELSE
                        Str(Round(phase^[d-1+spec_begin] / 10.0),strg);
                ed.data2[d]:=strg;
                ed.dtyp[d]:=numbers;
                ed.dlen[d]:=4;
        END;
        ed.last_edit:=20;
        ed.double_flag:=TRUE;
        exit_code:=handle_input(ed);
        IF exit_code=ok THEN BEGIN
                hide_all;
                FOR d:=1 TO 20 DO BEGIN
                        VAL(ed.data[d],awert,acode);
                        VAL(ed.data2[d],pwert,pcode);
                        IF (pcode=0) AND (pwert<360) THEN BEGIN
                                phase^[d-1+spec_begin]:=pwert*10;
                                spektrum^[d-1+spec_begin]:=awert;
                                sin_koeff^[d-
1+spec_begin]:=Round(awert*cos(pwert*PI/180));
                                cos_koeff^[d-
1+spec_begin]:=Round(awert*sin(pwert*PI/180));
                        END;
                END;
                max:=0;
```



```
                scale_spec;
                {jojo:=set_spec_phase_mode(0,0,0);}
                jojo:=display_spec(0,0,0);
                jojo:=calc_kurve(0,0,0);
        END;
END;
        tab_eingabe:=TRUE;
END;

FUNCTION octave_up( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        calc_intervall:=calc_intervall div 2;
        octave_up:=display_spec(0,0,0);
END;
        octave_up:=TRUE;
END;

FUNCTION octave_down( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        calc_intervall:=calc_intervall*2;
        octave_down:=display_spec(0,0,0);
END;
        octave_down:=TRUE;
END;

VAR
        change_menu : Menu_Button_PTR;


FUNCTION do_change_kurve( rel_x, rel_y :LONGINT; nr:WORD) : BOOLEAN;
VAR
        d_center, d_next, d_old, d_start, d_end : INTEGER;
        d : WORD;
        mr : mouse_rec;
```

```
        value : data_type;
        temp : EXTENDED;
BEGIN
        d_old:=-1;
        d_center:= mess_window.rel_to_data(rel_x);
        get_mouse(mr);
        REPEAT
                IF d_center>=0 THEN BEGIN
                        d_next :=
mess_window.scr_to_data(1+mess_window.data_to_scr(d_center))-1;
                        value:=mess_window.y_to_value(mr.y);
                END;
                IF (d_center>=0) AND (d_next<used_data) THEN BEGIN
                        d_start:=d_center;
                        d_end  :=d_next;
                        IF (d_old>d_next) AND (d_old<used_data) AND (d_old>=0) THEN
BEGIN
                                d_end:=d_old;
                        END;
                        IF   (d_old<d_center)   AND   (d_old<used_data)   AND   (d_old>=0)
THEN BEGIN
                                d_start:=d_old;
                        END;
                        hide_mouse;
                        FOR d:=d_start TO d_end DO
                                mess_window.change_value(d,value);
                        show_mouse;
                END;
                get_mouse(mr);
                d_old:=d_center;
                IF mr.left THEN BEGIN
                        mr.left:=mess_window.test_n_handle_touch(mr.x,mr.y);
                        d_center:=mess_window.scr_to_data(mr.x);
                END;
        UNTIL NOT mr.left;

        do_change_kurve:=TRUE;
```



```
END;

FUNCTION end_change (rel_x, rel_y : LONGINT;nr : WORD): BOOLEAN;
BEGIN
        change_menu^.set_leave(FALSE);
        end_change:=TRUE;
END;

FUNCTION change_kurve( rel_x, rel_y : LONGINT; nr : WORD): BOOLEAN;
VAR
        ch : CHAR;
        d, old_help : WORD;
        handle : INTEGER;
        continue ,touched, touched_button,
        old_touched,none : BOOLEAN;
{       mr : mouse_rec;}
        t1,t2,t3 : STRING;
        helper, exitor : Order_Button_PTR;

BEGIN
        old_help:=global_help_number;
        global_help_number:=112;
        hide_all;
        mess_window.set_mark(3,data_mark[1],FALSE); { nur wg TRUE }
        mess_window.set_mark(4,data_mark[2],FALSE);
        mess_window.hide;
        mess_window.set_scale(0,0,all_points);
        mess_window.show;
        exitor:=New(
Order_Button_PTR,init(0,0,9*char_width,char_height,col_help_back,col_help,col_help,11
2,
                                char(27),'Ende                                des
Veränderns','Fertig',end_change));
        helper:=New(
Order_Button_PTR,init(0,0,9*char_width,char_height,col_help_back,col_help,col_help,11
2,
                                'h','Ändern                                erklären
lassen','Hilfe',global_hilfe));
```

```
        change_menu:=New(menu_Button_ptr,init(0,0,0,0,col_help,col_help,col_help_back
,112,
                                              no_hot_key,","));
        change_menu^.add_button(exitor,TRUE);
        change_menu^.add_button(helper,TRUE);
        change_menu^.add_button(@mess_window,TRUE);

        t1:='Sie wollen die Kurve ändern! Dazu klicken Sie bitte in dem Kurvenfenster';
        t2:=' an die Stellen, an denen Sie Änderungen wünschen. Mit gedrückter';
        t3:=' Maustaste und LANGSAMEM Bewegen wird eine ganz neue Kurve gezogen!';
        show_info_line('Bitte ändern oder wählen Sie durch Anklicken, Ende mit ESC');
        handle:=show_alertY(GetMaxY *2 div 3,t1,t2,t3,exitor,helper,NIL);
        mess_window.set_reaction(do_change_kurve,'Formen Sie die Kurve hier',1);
        change_menu^.set_leave(TRUE);
        change_menu^.direct_call;
        hide_alert(handle);
        dispose(exitor,done);
        dispose(helper,done);
        dispose(change_menu,done);
        hide_info_line;
        mess_window.hide;
        mess_window.set_reaction(dummy,'geänderte Kurve der Messwerte',0);
        mess_window.set_scale(0,0,points);
        mess_window.show;
        change_kurve:=display_zoom(0,0,0);
        global_help_number:=old_help;
END;


FUNCTION do_change_spec( rel_x, rel_y :LONGINT; nr:WORD) : BOOLEAN;
VAR
        d_center, d_next, d_old, d_start, d_end : INTEGER;
        d : WORD;
        mr : mouse_rec;
        value : data_type;
        temp, ph, am : EXTENDED;
BEGIN
```



```
        d_old:=-1;
        d_center:= spec_window.rel_to_data(rel_x);
        get_mouse(mr);
        hide_mouse;
        IF (d_center>=0) AND (d_center<used_spec) THEN BEGIN
                value:=spec_window.y_to_value(mr.y);
                IF value < max_spec_amplitude THEN BEGIN
                        IF value < 0 THEN
                                value:=0;
                        spec_window.change_value(d_center,value);
                        spec_window.update_ttext;

                        ph:=phase^[d_center];
                        am:=spektrum^[d_center];
                        sin_koeff^[d_center]:=Round(am*cos(ph*PI/180));
                        cos_koeff^[d_center]:=Round(am*sin(ph*PI/180));

        END;END;
        show_mouse;
        do_change_spec:=TRUE;
END;

FUNCTION change_spec( rel_x, rel_y : LONGINT; nr : WORD): BOOLEAN;
VAR
        ch : CHAR;
        d, old_help : WORD;
        handle : INTEGER;
        continue ,touched, touched_button,
        old_touched,none : BOOLEAN;
        t1,t2,t3 : STRING;
        helper, exitor : Order_Button_PTR;

BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        old_help:=global_help_number;
        global_help_number:=111;
```

```
        none:=set_spec_phase_mode(0,0,0);
        phases_window.disable;
        exitor:=New(
Order_Button_PTR,init(0,0,9*char_width,char_height,col_help_back,col_help,col_help,11
1,
                                        char(27),'Ende                                  des
Veränderns','Fertig',end_change));
        helper:=New(
Order_Button_PTR,init(0,0,9*char_width,char_height,col_help_back,col_help,col_help,11
1,
                                        'h','Ändern                                     erklären
lassen','Hilfe',global_hilfe));
        change_menu:=New(menu_Button_ptr,init(0,0,0,0,col_help,col_help,col_help_back
,111,
                                        no_hot_key,","));
        change_menu^.add_button(exitor,TRUE);
        change_menu^.add_button(helper,TRUE);
        change_menu^.add_button(@spec_window,TRUE);

        t1:='Sie wollen das Spektrum ändern! Dazu klicken Sie bitte in dem
Spektrumfenster';
        t2:=' an die Stellen, an denen Sie Änderungen wünschen und ziehen Sie dann';
        t3:=' einen neuen Balken als neue/ geänderte Amplitude einer Frequenz hoch';
        show_info_line('Bitte ändern oder wählen Sie durch Anklicken, Ende mit ESC');
        handle:=show_alertY(GetMaxY div 3,t1,t2,t3,exitor,helper,NIL);
        spec_window.set_reaction(do_change_spec,'Ändern',1);
        change_menu^.set_leave(TRUE);
        change_menu^.direct_call;                       { jetzt passiert's }

        hide_alert(handle);
        dispose(exitor,done);
        dispose(helper,done);
        dispose(change_menu,done);
        hide_info_line;
        spec_window.set_reaction(dummy,'Spektrum',0);
        global_help_number:=old_help;
        change_spec:=calc_kurve(0,0,0);
END;
        change_spec:=display_spec(0,0,0);
```



```
END;

END.
```

E.7. Die Unit F_DATEI

```
UNIT f_datei;           { übernimmt alle Dateioperationen }

{$F+} {$O+}

INTERFACE

        FUNCTION load_data( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION save_data( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION delete_file( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION print( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        PROCEDURE save_messung;
        PROCEDURE load_messung;
        PROCEDURE save_temp;
        PROCEDURE load_temp;
        PROCEDURE save_choix;
        function FileExists(FileName: string): Boolean;

IMPLEMENTATION
USES f_global,Dos, gui, gui_def, gui_tool, f_help, f_graph, f_gr_def,
                                f_mouse, f_change, f_calc, Graph, Printer, f_analyse;
TYPE
        file_header = RECORD
                identification          : String[96];
                datei_beschreibung      : String[40];
                user_ID                                 : String[10];
                marke1,marke2,
                datas_used                      : WORD;
                frequenz        : LONGINT;
                file_version            : WORD;
                reserved                        : ARRAY[1..43] OF WORD;
                ende                            : STRING[8];
        END;
VAR
        header : file_header;
        ident_text : STRING[96];
```



```
CONST
        identif_text = 'DatenFile zum Fourierprogramm '+
                                'Version 0.5 (c) 1992 C.Bienmüller '+
                                'Uni Würzburg'+chr(96)+chr(26)+
                                '                               ';

PROCEDURE save( name : String );
VAR
        d : WORD;
        datei : FILE;
BEGIN
        Assign( datei, name);
        Rewrite(datei,1);
        header.ende:='1248';
        FOR d:=1 TO 43 DO header.reserved[d]:=0;
        header.marke1:=data_mark[1];
        header.marke2:=data_mark[2];
        header.identification:=ident_text;
        header.frequenz:=Round(mess_freq);
        header.file_version:=1;         { Aktuelle Versionsnummmer }
        header.datas_used:=used_data;
        header.user_id:=user_id;
        BlockWrite(datei,header,SizeOf(header));
        BlockWrite(datei,work_data^,SizeOf(data_type)*(used_data+1));
        Close(datei);
END;

PROCEDURE load( name : String );
VAR
        txt1, txt2, txt3 : STRING;
        datei : FILE;
        a,b, t, min, max : data_type;
        d : WORD;
        handle : INTEGER;
        temp : Extended;
BEGIN
```

```
        txt1:='Die Datei mit den Daten';
        txt3:='wird geladen...';
        show_info_line('Bitte warten sie einen Moment');
        Assign( datei, name);
        Reset(datei,1);
        BlockRead(datei,header,SizeOf(header));
        IF (header.identification<>ident_text) OR (header.ende <>'1248') THEN
show_help(89)
        ELSE IF ((header.user_id=user_id) OR (header.user_id='Master')
                                                                    OR
(user_id='Master')) THEN BEGIN
                txt2:=header.datei_beschreibung;
                handle:=0;
                IF (name<>'fourmess.dat') AND (name<>'fourtemp.dat') THEN
                        handle:=show_alert(txt1,txt2,txt3);
                BlockRead(datei,work_data^,SizeOf(data_type)*(header.datas_used+1));
                IF handle>0 THEN hide_alert(handle);
                used_data:=header.datas_used;
                data_mark[0]:=used_data div 2;
                data_mark[1]:=header.marke1;
                data_mark[2]:=header.marke2;
                calc_intervall:=data_mark[2]-data_mark[1];
                CASE header.file_version OF
                        0: mess_freq:=0;                { nicht bekannt }
                        1: BEGIN
                                mess_freq:=header.frequenz; { ab jetzt aber... }
                                temp:=1000.0*used_data;
                                mess_intervall:=Round(temp / mess_freq);
                           END;
                END;
                scale_mess;
                mess_window.set_data(work_data,0,used_data-1);
                zoom_window.set_data(work_data,data_mark[1],data_mark[2]);
                back_zoom_window.set_data(work_data,0,1);
                FOR d:=0 TO used_spec DO BEGIN
                        spektrum^[d]:=0;
                        phase^[d]:=0;
```



```
                        sin_koeff^[d]:=0;
                        cos_koeff^[d]:=0;
                END;

        END;
        Close(datei);
END;

FUNCTION file_select( match, why : string; all_files : BOOLEAN; color : WORD ):
STRING;
VAR
        names : Select_Array;       { in GUI_def definiert, zum String-Übergeben an
create_select}
        d, lastfile, nr : WORD;
        search_file : FILE;
        DirInfo: SearchRec;
        txt : Select_Array; {ARRAY[1..20] OF STRING[40];}
        last_string : WORD;
        header : file_header;
        menptr : Menu_Button_PTR;
        mr      : mouse_rec;
        fault : BOOLEAN;
        titel : STRING[25];
begin
        show_status('Festplatte wird durchsucht');
        FindFirst(match, AnyFile, DirInfo);
        d:=1;
{$I-}
        WHILE (DosError = 0) AND (d<20) DO BEGIN
                names[d]:=DirInfo.Name;
                fault:=FALSE;
                Assign( search_file, DirInfo.Name);
                Reset(search_file,1);
                IF IOResult<>0 THEN
                        fault:=TRUE
                ELSE BEGIN
                        BlockRead(search_file, header, SizeOf(header));
```

```
                    IF IOResult<>0 THEN fault:=TRUE;
                    Close(search_file);
                END;
                IF    (header.identification=ident_text)
                  AND (header.ende='1248')
                  AND (         (header.user_id=user_id)
                                OR      ((header.user_id='Master')      AND      all_files)      OR
param_master)
                  AND (NOT fault) THEN BEGIN
                    txt[d]:=header.datei_beschreibung;
                    INC(d);
                END;
                FindNext(DirInfo);
        END;
{$I+}
        last_string:=d-1;
        get_mouse(mr);     { SelectBox erscheint an Mausposition, Abfrage bei create_select
auf Bildschirmränder }
        titel:='Dateiauswahl zum '+why;
        col_select_titel:=color;
        create_select(menptr, txt, last_string, mr.x, mr.y,titel);
        d:=handle_select(menptr);
        col_select_titel:=Yellow;
        dispose(menptr,done);
        IF (d>0) THEN
                file_select:=names[d]
        ELSE
                file_select:='';
END;

{ Die folgende Funktion wurde einfach von der Borland Hilfe für Reset übernommen }
function FileExists(FileName: string)
                        : Boolean;
{ Returns True if file exists; otherwise,
  it returns False. Closes the file if
  it exists. }
var
```



```
  f: file;
begin
  {$I-}
  Assign(f, FileName);
  Reset(f);
  Close(f);
  {$I+}
  FileExists := (IOResult = 0) and (FileName <> '');
end;  { FileExists }



FUNCTION save_data( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        y,m,d,dow,
        h,min,s,s1 : WORD;
        strg,
        file_name : STRING[40];
        summe : WORD;
        to_do : STRING[40];
BEGIN
        IDENT_text:=identif_text;
        to_do:='Eine ausführliche Beschreibung der Daten bitte';
        header.datei_beschreibung:='Messung von';
        input(to_do,header.datei_beschreibung,40);
        if header.datei_beschreibung <> 'Messung von' THEN BEGIN
                Randomize;
                REPEAT
                        Str(Random(9000)+1000,strg);
                        file_name:='four' + strg+'.dat';
                UNTIL NOT FileExists(file_name);
                save(file_name);
        END;
        save_data:=TRUE;
END;
```

```
PROCEDURE save_all(filename:STRING);
BEGIN
        IDENT_text:=identif_text;
        header.datei_beschreibung:='Letzte Messung, unbearbeitet';
        save(filename);
END;

PROCEDURE load_all(filename:STRING);
VAR
        d : WORD;
        none : BOOLEAN;
BEGIN
        IDENT_text:=identif_text;
        IF FileExists(filename) THEN BEGIN
                FOR d:=1 TO 4 DO BEGIN
                        mess_window.set_mark(d,-1,FALSE);
                END;
                mess_window.hide;
                hide_all;
                load(filename);
                mess_window.show;
                none:=display_zoom(0,0,0);
        END;
END;



FUNCTION load_data( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        d : WORD;
        strg : string[40];
BEGIN
        IDENT_text:=identif_text;
        strg:=file_select('four*.dat','Laden',TRUE,LightGreen);
        IF strg<>'' THEN BEGIN
                FOR d:=1 TO 4 DO BEGIN
                        mess_window.set_mark(d,-1,FALSE);
```



```
                END;
                hide_all;
                mess_window.hide;
                load(strg);
                mess_window.show;
                IF used_data>=30000 THEN load_data:=do_select_work_area(0,0,0);
                load_data:=     display_zoom(0,0,0);
        END ELSE
                load_data:=TRUE;
END;

PROCEDURE do_delete(stg : STRING);
VAR
        strg : STRING;
        regs : REGISTERS;
BEGIN
        strg:=stg+char(0);
        regs.ah:=$41;
        regs.ds:= Seg(strg);
        regs.dx:= Ofs(strg[1]);  { da C-String, endet mit 0 }
        Intr($21,regs);
        IF regs.ax=5 THEN show_help(131);
END;

FUNCTION delete_file( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        strg : string[40];
        txt : STRING[160];
        flag : WORD;
        datei : FILE;
BEGIN
        IDENT_text:=identif_text;
        REPEAT
                strg:=file_select('*.dat','Löschen',FALSE,LightRed);

                IF strg<>'' THEN BEGIN
```

```
                    Assign( datei, strg);
                    Reset(datei,1);
                    BlockRead(datei,header,SizeOf(header));
                    Close(datei);
                    IF (header.user_id='Master') AND (NOT param_master) THEN
show_help(130)
                    ELSE BEGIN
                        txt:='  Sind Sie sicher, daß Sie die Datei  |"'+
                                                    header.datei_beschreibung+'"
|dauerhaft löschen wollen?';
                        IF param_master THEN txt:='  Sind Sie sicher, daß Sie die
Datei  |"'+
                                                    header.datei_beschreibung+'"  |
von "'+header.user_id+'" dauerhaft löschen wollen?';
                        flag:=alert_select(txt,' Ja,löschen |Nein, abbrechen');
                        IF flag=1 THEN
                            do_delete(strg);
                    END;
                END;
        UNTIL strg='';
        delete_file:=TRUE;
END;

PROCEDURE save_messung;
BEGIN
        save_all('fourmess.dat');
END;

PROCEDURE load_messung;
BEGIN
        load_all('fourmess.dat');
END;

PROCEDURE save_temp;
BEGIN
        save_all('fourtemp.dat');
END;
```



```
PROCEDURE load_temp;
BEGIN
        load_all('fourtemp.dat');
        do_delete('fourtemp.dat');
END;

PROCEDURE save_choix;
BEGIN
        IDENT_text:=identif_text;
        header.datei_beschreibung:='Letzte Auswahl';
        save('fourchoi.dat');
END;



PROCEDURE hard_copy;
CONST
        esc : STRING ='□*';
TYPE
        c = CHAR;

        PROCEDURE p(chr : CHAR);
        BEGIN
        {$I-}
                Write(Lst,chr);
        {$I+}
                IF IOResult<>0 THEN BEGIN
                        show_help(333);
                        Write(Lst,chr);
                END;
        END;

        PROCEDURE prt_strg( strg : STRING);
        VAR d : INTEGER;
        BEGIN
                FOR d:=1 TO Length(strg) DO
                        p(strg[d]);
```

```
    END;


VAR
    d,f,x,y,bytes, rand : INTEGER;
    strg, auflsgstrg : STRING[15];
    field : ARRAY[0..255] OF BYTE;
CONST
    tabelle : ARRAY[0..15] OF BOOLEAN =
                    (FALSE,TRUE,TRUE,FALSE,
FALSE,FALSE,FALSE,TRUE,
                    FALSE,TRUE,TRUE,TRUE,
TRUE,TRUE,TRUE,TRUE);

BEGIN
    hide_mouse;

    IF GetMaxX<650 THEN BEGIN
        rand:=6;
        auflsgstrg:='100';
    END ELSE BEGIN
        rand:=9;
        auflsgstrg:='150';
    END;
    prt_strg(esc+'t'+auflsgstrg+'R');                {auflösung}
    prt_strg(esc+'r1A');        {linker Rand}

    Str((GetMaxX+1+rand*8),strg);
    prt_strg(esc+'r'+strg+'S');      {Punktzahl}

    bytes:=(GetMaxX+1) div 8;
    Str(bytes+rand,strg);
    strg:=esc+'b'+strg+'W';

    prt_strg(strg);                  {Bytezahl}
    FOR d:=1 TO rand DO p(c(0));
    FOR d:=0 TO bytes-1 DO      { oberer Rand }
        p(c(255));
```



```
    FOR f:=1 TO 5 DO BEGIN
        prt_strg(strg);              {Bytezahl}
        FOR d:=1 TO rand DO p(c(0));
        p(c(128));
        FOR d:=0 TO bytes-3 DO      { oberer Rand }
            p(c(0));
        p(c(1));
    END;
    FOR y:=2*char_height-3 TO GetMaxY-char_height-5 DO BEGIN
        prt_strg(strg);              {Bytezahl}

        FOR d:=1 TO rand DO p(c(0));

        FOR d:=0 TO bytes DO
            field[d]:=0;
        field[0]:=128;          { linker}
        field[bytes-1]:=1;          {und rechter Rand}

        FOR x:=2 TO GetMaxX-2 DO BEGIN
            d:=GetPixel(x,y);
            IF tabelle[d] THEN
                INC( field[x div 8],128 shr (x mod 8));
        END; {for x}



        FOR d:=0 TO bytes-1 DO
            p(c(field[d]));
    END; {for y}

    prt_strg(strg);              {Bytezahl}
    FOR d:=1 TO rand DO p(c(0));
    FOR d:=0 TO bytes-1 DO                  { unterer Rand }
        p(c(255));

    prt_strg(esc+'rbC');
    show_mouse;
```

```
END;

FUNCTION print( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        j,m,t,d,f,mx : WORD;
        strg : STRING[20];
        line : STRING[120];
        ex : EXTENDED;
        tr : text_rec;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        GetDate(j,m,t,d);
        WriteLn(Lst);
        WriteLn(Lst,Char(27)+'(s2QPraktikumsversuch 36 - Ausdruck des
Fourierprogramms von Chr. Bienmüller');  { NLQ }
        WriteLn(Lst,'Benutzer: '+user_name+' ('+user_id+')      am: ',t,'.',m,'.',j);
        WriteLn(Lst);

        hard_copy;

        mx:=119;
        IF mx>used_spec THEN mx:=used_spec;
        WriteLn(Lst);
        WriteLn(Lst);
        WriteLn(Lst);
        WriteLn(Lst);
        WriteLn(Lst);
        WriteLn(Lst,Char(27)+'(s2Q');   { s1Q für Draft }

        Koeff_to_tr(tr);

        FOR f:=1 TO tr.last_string DO BEGIN
                WriteLn(Lst,'   '+tr.strg[f]);
        END;

        WriteLn(Lst);
        WriteLn(Lst,'Messfrequenz:      ',Round(mess_freq):5,'  Hz');
```



```
        WriteLn(Lst,'Meßwerte (gesamt): ',used_data:5);
        d:=data_mark[2]-data_mark[1];
        IF d<0 THEN d:=-d;
        WriteLn(Lst,'    Periodenlänge:',d:5,'   Messwerte');
        ex:=d*1000.0/mess_freq*10;
        ex:=Round(ex)/10;
        Str(ex:8:2,strg);
        WriteLn(Lst,'        das sind:',strg,' Millisekunden');
        ex:=mess_freq/d*spec_step;
        Str(ex:8:2,strg);
        WriteLn(Lst,'    Grundfrequenz:',strg,' Hz');
        ex:=spec_step;
        Str(ex:8:2,strg);
        WriteLn(Lst,'Markierte Perioden:',strg);



        GetDate(j,m,t,d);
        {Write(Lst,Char(27)+
                '(s2Q                   Fourierprogramm von C.Bienmüller `93'
                +Char(12));} { NLQ-Text }
        Write(Lst,CHAR(12)); {formfeed}
END;
        print:=TRUE;
END;




END.
```

E.8. Die Unit F_GLOBAL

```
unit f_global;

INTERFACE
uses gui_def, f_gr_def, Graph;

{------------------------- Globale Variablen -----------------------------}
CONST

        col_mess =          White;
        col_higru =         LightGreen;
        col_spec =          White;
        col_phase =         Yellow;
        col_sin =           LightGreen;
        col_cos =           Yellow;
        col_graph_back= DarkGray;

        col_menu =          Blue;
        col_menu_back = Yellow;
        col_sub_menu =      Black;

        prec = 3000;

TYPE
        higrus = ARRAY[0..prec] OF data_type;
        higru_ptr = ^higrus;
VAR
        higru_datas : higru_ptr;
        higru_shown,
        higru_set : BOOLEAN;

TYPE
        spec_mode_type = (koeff,phases);
VAR
        no_mess : BOOLEAN;
```



```
        mess_freq : Real;
        mess_max, mess_min : INTEGER;
        mess_intervall : Real;  { cb }

        work_data : data_array_PTR;
        sin_koeff,
        cos_koeff,
        phase,
        spektrum  : spec_array_PTR;
        mess_window,                    { Oberes Fenster mit
Messwerten }
        higru_window,
        zoom_window,                    { Lupenfenster, nur bei
Bedarf aktiv }
        back_zoom_window,
        sin_window,
        cos_window,
        phases_window,
        spec_window : Graph_Button;     { Spektrum, nur bei Bedarf... }
        swap_marks,
        shift_left,
        shift_right: Order_Button;
        data_mark     : ARRAY[0..2] OF WORD;  { hier werden die Marken abgelegt}
        used_data     : WORD;           { wieviele data bei der Messung
belegt wurden }
        used_spec     : WORD;
        spec_step,                      { wieviele Perioden
markiert }
        calc_intervall : WORD;
        spec_display_mode : spec_mode_type;
        spec_begin : WORD;
        zoom_faktor : WORD;
        mark_to_set : WORD;             { welche Marke zu setzen ist }
        param_svga : BOOLEAN;           { ist SVGA.BGI-Treiber zu nehmen?}
        param_master : BOOLEAN;         { master als parastring?}
        param_sound : BOOLEAN;          { no_sound nicht als parastrg?}
        param_mess : BOOLEAN;           { no_mess     -"-     }
        param_noexit : BOOLEAN;
```

```
        user_name : STRING[38];
        user_id : STRING[10];
        spec_calculated : BOOLEAN;                { wurde überhaupt schon errechnet? }

FUNCTION set_standards(x,y : LONGINT; nr : WORD):BOOLEAN;

IMPLEMENTATION
uses f_mess;

FUNCTION set_standards(x,y : LONGINT; nr : WORD):BOOLEAN;
BEGIN
        IF NOT no_mess THEN BEGIN
                mess_freq:=SC_FMax;
                IF mess_freq>60000.0 THEN mess_freq:=60000.0;
                mess_intervall:=500;
        END;
        used_spec       :=100;
        spec_step   :=1;
        spec_begin  :=0;
        set_standards:=TRUE;
END;

END.
```


E.9. Die Unit F_GR_DEF

```
{ in dieser Unit werden die Graphen dargestellt sowie Zoom o.ä. }
{ Es wird insbesondere ein neues ButtonObjekt kreiert, Graph_Button }
UNIT f_gr_def;

INTERFACE
        USES gui, gui_def;

        CONST
                innenrand               = 5;                    { der Abstand zw. Graph
& Rahmen }
                max_datas = 32000;    { maximal 32000 }
                max_value = 2028;
                max_spect = 1000;

        TYPE
                data_type = INTEGER;
                data_array = array[0..Max_Datas] of data_type;
                spec_array = array[0..Max_spect] of data_type;
                data_array_ptr = ^data_array;
                spec_array_ptr = ^spec_array;
                name_pos_type = ( hidden, left, right );
                touch_text_type = STRING[max_touch_text];
                gtt = PROCEDURE( var touch_text : touch_text_type; nr :WORD);
                display_mode_type = ( points, lines, dotted, clocks, second_lines,
second_points, all_points );
                                                                { points: Punkte, xor-
sicher gesetzt,
                                                                 lines: dito Linien zur
Nullinie,
                                                                 dotted: Nur geradzalige
points,
                                                                 clocks: Uhren die
Winkel 0...3600 zeigen, bei<0 ohne Zeiger!,
                                                                 second_lines: um ein
Pixel n. rechts gesetzte Linie,
                                                                 second_points: ohne
Nullinie...
```

```
                                                    all_points:     ignoriert
xor-problem für change_value! }

        Graph_Button = Object(Button)

                CONSTRUCTOR     init(   bx, by, bdx, bdy        : INTEGER;
                                        bcolor, bframe_color,
                                        bgraph_color,bh_page,
                                        bm_style                :
WORD;
                                        hkey                    :
CHAR;
                                        btouch_text             : STRING;
                                        bdo_it                  :
Do_It_Func;
                                        bname                   :
STRING;
                                        bn_pos                  :
name_pos_type );
                FUNCTION    test_n_handle_touch( mx, my : WORD): BOOLEAN;
VIRTUAL;
                FUNCTION     handle_klick( mx, my, nr : WORD) : BOOLEAN
;VIRTUAL;
                PROCEDURE           select; VIRTUAL;
                PROCEDURE           unselect; VIRTUAL;
                PROCEDURE   show; VIRTUAL;
                PROCEDURE   hide; VIRTUAL;
                PROCEDURE           set_data( var dta; fd,ld : WORD); VIRTUAL;
                PROCEDURE   get_data( var f_d,l_d : WORD);
                PROCEDURE           set_scale( bottom, top : data_type; dmode :
display_mode_type); VIRTUAL;
                PROCEDURE     set_reaction( ptr : Do_It_Func; t_text : String;
m_style : WORD ); VIRTUAL;
                PROCEDURE    set_mark( nr : WORD; value : INTEGER; active :
BOOLEAN);
                PROCEDURE   set_analytic( analyse : BOOLEAN; tt_changer : gtt);
                PROCEDURE   draw_mark( nr : WORD);
                PROCEDURE           display; VIRTUAL;
                PROCEDURE   ignore_first_time;
                PROCEDURE   set_girwidz_color( col : WORD);
```



```
                FUNCTION    scr_to_data(koord : WORD): INTEGER;
                FUNCTION    data_to_scr(number : WORD): WORD;
                FUNCTION    rel_to_data(rel_x : LONGINT):INTEGER;
                FUNCTION    y_to_value( koord : WORD): data_type;
                PROCEDURE   change_value(nr : WORD; value : data_type); { nur
mit all_points verwenden }
                PROCEDURE   update_ttext;
        PRIVATE
                do_it   : Do_It_Func;       { wird bei klick aufgerufen }
                data    : data_array_ptr;   { zeigt auf darzustellende Daten }
                display_mode: display_mode_type;  { punkte, linien, gepunktet... }
                mouse_style,                { wie Maus bei Berühren
aussieht }
                first_data,                 { wählt }
                last_data : WORD;           {  Ausschnitt aus den Daten }
                old_data_analyt : INTEGER;  { nur Zwischenspeicher }
                mark    : ARRAY[1..4] OF INTEGER; { nimmt die Marken auf }
                active_mark : WORD;         {  für  hellere
Markensetzung }
                analytic,                   {  Werden  Koordinaten
eingeblendet }
                shown,                      { ist was dargestellt }
                first_time: BOOLEAN;        { schon mal dargestellt?f_t=true:nein}
                inv_data_intervall : EXTENDED;{ 1/(last_data-first_data)  }
                scale_top, scale_bottom, scale_delta : data_type; {für Maßstab}
                girwidz_color : WORD;       {  Farbe  für  seitliche
Begrenzung }
                name_button : Text_Button_PTR;  { zeigt Namen an... }
                get_touch_text : gtt;   { Procedure die den Touchtext verändert }
                y_factor: EXTENDED;
                y0 : INTEGER;
        END;


    PROCEDURE My_Circle(x,y,r : WORD);

implementation
uses Graph, f_help,f_mouse;
```

```
{------------------------------------------------------------------------}
{---------------------- Methoden von Graph_Button -----------------------}
{------------------------------------------------------------------------}

CONSTRUCTOR Graph_Button.init(      bx, by, bdx, bdy                : INTEGER;
                                                    bcolor, bframe_color,
                                                    bgraph_color,bh_page,
                                                    bm_style
        : WORD;
                                                    hkey
        : CHAR;
                                                    btouch_text         : STRING;
                                                    bdo_it
        : Do_It_Func;
                                                    bname
        : STRING;
                                                    bn_pos
        : name_pos_type);
VAR
        d : WORD;
BEGIN
        Button.init(    bx, by, bdx, bdy,
                                bcolor, bframe_color, bgraph_color, bh_page, hkey,
                                btouch_text );
        do_it:=bdo_it;
        first_data:=0;
        last_data:=0;
        first_time:=TRUE;
        shown:=FALSE;
        mouse_style:=bm_style;
        FOR d:=1 TO 4 DO
                mark[d]:=-1;  { disabled }
        analytic:=FALSE;
        active_mark:=0; { none }
        set_scale(-max_value,max_value, points);
        girwidz_color:=0; { disabled }
        IF MaxAvail < SizeOf(Text_Button) THEN BEGIN
                show_help(103);
```



```
                halt(1);
        END;
        CASE bn_pos OF
                hidden :        name_button:=NIL;
                left:           name_button:=New(Text_Button_PTR,init(x,y-char_height-2,

        (Length(bname)+2)*char_width,char_height+2,bcolor,
                                                        bframe_color,
bgraph_color XOR bcolor, bh_page,
                                                        hkey,       btouch_text,
bname));
                right:          name_button:=New(Text_Button_PTR,init(x+dx-
(Length(bname)+2)*char_width,y-char_height-2,

        (Length(bname)+2)*char_width,char_height+2,bcolor,
                                                        bframe_color,
bgraph_color XOR bcolor, bh_page,
                                                        hkey,       btouch_text,
bname));
        END;
END;

PROCEDURE Graph_Button.ignore_first_time;
BEGIN
        first_time:=FALSE;
END;

PROCEDURE Graph_Button.set_girwidz_color( col : WORD);
BEGIN
        girwidz_color:=col;
END;

PROCEDURE Graph_Button.set_data( var dta ; fd,ld : WORD);
VAR
        only_set_data: BOOLEAN;
BEGIN
        IF ( (first_data <> fd) OR
                (last_data <> ld) OR
```

```
                (data <> data_array_ptr(dta)) ) THEN BEGIN
                only_set_data:=NOT shown;
                IF (last_data<>first_data) AND NOT only_set_data THEN
                        display;        { löscht wieder wg. XOR }
                data:=data_array_ptr(dta);
                first_data:=fd;
                last_data:=ld;
                inv_data_intervall:=1/(last_data-first_data);
                IF NOT only_set_data THEN
                        display;
        END;
END;

PROCEDURE Graph_Button.get_data( var f_d,l_d : WORD);
BEGIN
        f_d:=first_data;
        l_d:=last_data;
END;

PROCEDURE Graph_Button.set_scale( bottom,top : data_type; dmode :
display_mode_type);
VAR
        temp : LONGINT;
BEGIN
        IF top<>bottom THEN BEGIN
                scale_top:=top;
                scale_bottom:=bottom;
                temp:=top-bottom+1;
                IF (temp>32000) THEN temp:=32000;  { schwacher Schutz }
                scale_delta:=temp;
        END;
        display_mode:=dmode;
END;

PROCEDURE Graph_Button.set_mark( nr : WORD; value : INTEGER; active :
BOOLEAN);
VAR
```



```
        old_active : WORD;
BEGIN
        draw_mark(nr);  {löschen...}
        mark[nr]:=value;
        IF active THEN BEGIN
                old_active:=active_mark;
                IF (old_active<>0) AND ( old_active<> nr) THEN BEGIN
                        draw_mark(old_active);
                        active_mark:=0;
                        draw_mark(old_active);
                END;
                active_mark:=nr;
        END
        ELSE
                IF active_mark=nr THEN
                        active_mark:=0;
        draw_mark(nr);  {wieder malen  XOR sei Dank}
END;

PROCEDURE Graph_Button.set_analytic( analyse : BOOLEAN; tt_changer : gtt);
BEGIN
        analytic:=analyse;
        get_touch_text:=tt_changer;
END;

FUNCTION Graph_Button.scr_to_data(koord : WORD): INTEGER;  { ohne Range_Check
!!}
BEGIN
        scr_to_data:= first_data + Round( (koord - (x+innenrand)) / ((dx-2*innenrand) *
inv_data_intervall) );
END;

FUNCTION Graph_Button.data_to_scr(number : WORD): WORD;
BEGIN
        data_to_scr:=(x+innenrand) + Round( (dx-2*innenrand) * (number-first_data) *
inv_data_intervall);
END;
```

```
FUNCTION Graph_Button.rel_to_data(rel_x : LONGINT):INTEGER;
BEGIN
        rel_to_data:= scr_to_data(x+(rel_x*dx) div 1000);
END;

FUNCTION Graph_Button.y_to_value( koord : WORD): data_type;
BEGIN
        IF (koord>=y) AND (koord<=y+dy) THEN
                y_to_value:=round((koord-y0)/y_factor)
        ELSE
                y_to_value:=0;
END;

PROCEDURE Graph_Button.update_ttext;
BEGIN
        old_data_analyt:=-1;
END;

FUNCTION Graph_Button.test_n_handle_touch( mx, my : WORD): BOOLEAN;
VAR
        d, calc,temp : LONGINT;
        wort : word;
        strg    : STRING[40];

BEGIN
        IF  enabled THEN BEGIN
                IF analytic THEN BEGIN
                        d:=scr_to_data(mx);
                        IF ((d<>old_data_analyt) OR NOT selected)
                                                AND (d>=first_data)
                                                AND (d<=last_data) THEN BEGIN
                                wort:=d;
                                get_touch_text(touch_text_type(touch_text),wort);
                                selected:=FALSE;  { zum Neuschreiben }
                                old_data_analyt:=d;
```



```
                        END;
                END;
                test_n_handle_touch:=Button.test_n_handle_touch( mx, my);
        END
        ELSE
                test_n_handle_touch:=FALSE;
END;

FUNCTION Graph_Button.handle_klick( mx, my, nr : WORD) : BOOLEAN;
VAR
        rel_x_pm,rel_y_pm : LONGINT; { für Angabe der relativen Koord. in Promille}
BEGIN                   { Funktion wie bei Order_Button }
        rel_x_pm:=(1000*LONGINT(mx-x)) DIV (dx);
        rel_y_pm:=(1000*LONGINT(my-y)) DIV (dy);
        handle_klick:=do_it( rel_x_pm, rel_y_pm, nr);
END;

PROCEDURE Graph_Button.select;
BEGIN
        selected:=TRUE;
        set_mouse_style(mouse_style);
END;

PROCEDURE Graph_Button.unselect;
BEGIN
        IF selected THEN BEGIN
                set_mouse_style(0);
                selected:=FALSE;
        END;
END;

PROCEDURE Graph_Button.show;
BEGIN
        IF enabled AND first_time THEN BEGIN
                first_time:=FALSE;
                Button.display;
```

```
            Button.show;
        END
        ELSE IF not shown AND enabled THEN display;
END;

PROCEDURE Graph_Button.hide;
VAR
        d : WORD;
BEGIN
        IF shown AND enabled THEN display;
END;

PROCEDURE Graph_Button.display;
VAR
        d,col,sx,sy, last_x, last_y : Word;
        temp : EXTENDED;
BEGIN
IF NOT first_time THEN BEGIN
        shown:=NOT shown;                       {shown XOR 1, wir zählen
Aufrufe mod 2}

        y_factor:=- (dy-2) / scale_delta ;
        y0:=y+ dy-1- Round(scale_bottom * y_factor);

        hide_mouse;
        col:=select_color;
        SetWriteMode( XORPut );
        SetColor ( color XOR LightBlue);
        SetLineStyle(UserBitLn,$FFFF,NormWidth);
        IF (display_mode=lines) OR (display_mode=points) THEN
            Line(x+1,y0,x+dx-1,y0);                 { die primären
Anzeigen }
        SetLineStyle(SolidLn,0,NormWidth);
        SetColor( select_color);
        last_x:=maxint;
        last_y:=maxint;
        for d:=first_data to last_data do BEGIN
```



```
            sx:=data_to_scr(d);
            temp:=y0+ y_factor * data^[d];
            IF (display_mode<>clocks) AND (temp<maxint) AND (temp>0) THEN
                sy:=round(temp)
            ELSE
                sy:=y;                                  { dummyWert für
nächste Zeile }
            IF (sy>=y) AND (sy<=y+dy) THEN
             CASE display_mode OF
                points,all_points,second_points: BEGIN
                    IF (sx<>last_x) OR (last_y <> sy) OR
(display_mode=all_points)
                        THEN BEGIN    { verhindert wiederlöschen
benachbarter Punkte }
                        col:=GetPixel(sx,sy) XOR select_color;
                        PutPixel(sx,sy,col);
                        last_x:=sx;
                        last_y:=sy;
                    END;
                END;
                dotted: BEGIN
                    IF ((sx<>last_x) OR (last_y <> sy)) AND (d MOD 2 = 0)
THEN BEGIN  { verhindert wiederlöschen benachbarter Punkte }
                        col:=GetPixel(sx,sy) XOR select_color;
                        PutPixel(sx,sy,col);
                        last_x:=sx;
                        last_y:=sy;
                    END;
                END;
                lines: BEGIN
                    Line(sx, sy, sx, y0);
                END;
                second_lines: BEGIN
                    Line(sx+1, sy, sx+1, y0);
                END;
                clocks: IF (d<>First_data) AND (d<>last_data) THEN BEGIN
                    temp:=data^[d]*PI/1800;  { da in 10*Grad }
                    My_Circle(sx,y+15,10);
```

```
                        IF temp>=0 THEN BEGIN
                            Line(sx,y+15,Round(sx+10*cos(temp)),Round(y+15-
10*sin(temp)));
                        END;
                    END;
                END;
        END;
        IF girwidz_color>0 THEN BEGIN
            SetLineStyle(SolidLn,0,ThickWidth);
            SetColor(girwidz_color);
            Line(x+2,y+1,x+2,y+dy-1);
            Line(x+dx-2,y+1,x+dx-2,y+dy-1);
            SetLineStyle( SolidLn, 0, NormWidth);
        END;
        SetWriteMode( NormalPut );
        show_mouse;
        IF name_button<>NIL THEN CASE shown OF
            TRUE : name_button^.show;
            FALSE: name_button^.hide;
        END;
        FOR d:=1 TO 4 DO
            draw_mark(d);
END;
END;

PROCEDURE Graph_Button.change_value(nr : WORD; value : data_type);
VAR
        col, sx, sy : WORD;
        temp        : EXTENDED;
BEGIN
IF (nr>=first_data) AND (nr<=last_data) THEN BEGIN
        sx:=data_to_scr(nr);
        temp:=y0+ y_factor * data^[nr];
        sy:=round(temp);
        CASE display_mode OF
            all_points: BEGIN
                col:=GetPixel(sx,sy) XOR select_color;
```



```
                PutPixel(sx,sy,col);
            END;
            lines: BEGIN
                SetWriteMode(XORPut);
                SetColor(select_color);
                sy:=round(temp);
                Line(sx, sy, sx, y0);
            END;
            ELSE
                show_help(999);
        END;
        data^[nr]:=value;
        temp:=y0+ y_factor * data^[nr];
        sy:=round(temp);
        CASE display_mode OF
            all_points: BEGIN
                col:=GetPixel(sx,sy) XOR select_color;
                PutPixel(sx,sy,col);
            END;
            lines: BEGIN
                sy:=round(temp);
                Line(sx, sy, sx, y0);
            END;
            ELSE
                show_help(999);
        END;

END; END;

PROCEDURE Graph_Button.draw_mark( nr : WORD);
VAR
        col, sx, d : WORD;
        temp : EXTENDED;
        triangle : ARRAY[1..3] OF PointType;
BEGIN
        IF (mark[nr] >= first_data) AND (mark[nr] <= last_data) AND enabled THEN
BEGIN
```

```
            IF nr<3 THEN
                col := LightRed XOR color
            ELSE BEGIN
                col := LightGreen XOR color;
                IF (nr<> active_mark) THEN col:=Green XOR color;
            END;
            hide_mouse;
            SetColor(col);
            SetLineStyle( SolidLn, 0, NormWidth);
            sx:=data_to_scr(mark[nr]);
            SetWriteMode(XORPut);
            Line(sx,y+1,sx,y+dy-1);
            IF nr = 3 THEN FOR d:=1 TO 5 DO BEGIN
                Line(sx-d,y+1,sx-d,y+7-d);
                Line(sx-d,y+dy-1,sx-d,y+dy-7+d);
            END;
            IF nr = 4 THEN FOR d:=1 TO 5 DO BEGIN
                Line(sx+d,y+1,sx+d,y+7-d);
                Line(sx+d,y+dy-1,sx+d,y+dy-7+d);
            END;
            SetWriteMode(NormalPut);
            show_mouse;
        END;
END;

PROCEDURE   Graph_Button.set_reaction( ptr : Do_It_Func; t_text : String; m_style :
WORD );
BEGIN
        do_it := ptr;               { diese Procedure ist die Folge davon, }
        mouse_style := m_style;                 { daß die Graphen immer wieder
unterschiedlich }
        touch_text := t_text;                   { reagieren müssen }
END;

PROCEDURE My_Circle(x,y,r : WORD);
VAR
        d : WORD;
```



```
        circle : array[0..12] of PointType ;
BEGIN
        FOR d:=0 TO 12 DO BEGIN
            circle[d].x:=x+Round(r*cos(2*PI/12*d));
            circle[d].y:=y+Round(r*sin(2*PI/12*d));
        END;
        DrawPoly(13,circle);
END;
END.
```

E.10. Die Unit F_GRAPH

```
{ in dieser Unit (f_graph) werden die Graphen dargestellt sowie Zoom o.ä. }
UNIT f_graph;

INTERFACE
        USES gui_def, f_gr_def;
        PROCEDURE fg_init(x,y,xx,yy,dy,frame_color,graph_color,area_color : Word);
        FUNCTION display_zoom( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION display_only_zoom( rel_x, rel_y: LONGINT; nr: WORD) :
BOOLEAN;
        FUNCTION display_spec( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION set_spec_phase_mode( x_rel, y_rel : LONGINT; nr : WORD):
BOOLEAN;
        FUNCTION set_spec_koeff_mode( x_rel, y_rel : LONGINT; nr : WORD):
BOOLEAN;
        FUNCTION set_first_mark( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        FUNCTION set_second_mark( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        PROCEDURE hide_all;
        FUNCTION dummy ( rel_x, rel_y : LONGINT; nr : WORD) : BOOLEAN;
        FUNCTION play_sample( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
        PROCEDURE fg_get_mem;

implementation
uses f_global,Graph,f_mouse,f_calc, f_datei, f_help, f_mess, CRT, gui, gui_tool, f_sound;
{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}
VAR
        fg_x1,fg_x2,  fg_m1,fg_m2, fg_zx,
        fg_y1,fg_y2 : Word;
   fg_delta,fg_frame_col,fg_graph_col : Word;
        fg_area_col,fg_back_col : Word;
        {fg_is_standard : Boolean;}

FUNCTION not_implement( rel_x, rel_y: LONGINT; nr : WORD) : BOOLEAN;
VAR
        zahl, text: STRING[40];
```



```
BEGIN
        Str(nr,zahl);
        text:='Funk. '+zahl+' ist nicht impl. ';
        Str(rel_x,zahl);
        text:=text+zahl+' ';
        Str(rel_y,zahl);
        text:=text+zahl;
        show_status(text);
        show_help(84);
        not_implement:=True;
END;

FUNCTION dummy ( rel_X, rel_y : LONGINT; nr : WORD) : BOOLEAN;
BEGIN
        dummy:=TRUE;
END;

PROCEDURE  change_touch_text( var touch_text : touch_text_type; nr : WORD);
VAR
        calc : LONGINT;
        calc2 :EXTENDED;
        temp : data_type;
        strg : STRING[max_touch_text];
BEGIN
        str(nr:3,strg);
        touch_text:='Nr:'+strg;
        calc:=Round(mess_freq * nr * spec_step / calc_intervall);
        IF calc>99 THEN BEGIN
                Str(calc:5,strg);
                IF Length(strg)<6 THEN
                        strg:=','+strg
                ELSE
                        strg[1]:=',';
        END ELSE BEGIN
                calc2:=mess_freq * nr * spec_step / calc_intervall;
                Str(calc2:5:1,strg);
```

```
            strg:=','+strg;
        END;

        touch_text:=touch_text+strg;
        temp:=spektrum^[nr];
        str(temp:5,strg);
        touch_text:=touch_text+'Hz, A:'+strg+'mV ';
        temp:=phase^[nr] div 10;
        IF temp>=0 THEN
            str(temp:4,strg)
        ELSE
            strg:='  - ';
        touch_text:=touch_text+'Phi:'+strg+'°';
END;

PROCEDURE  change_sin_touch_text( var touch_text : touch_text_type; nr : WORD);
VAR
        calc : LONGINT;
        calc2 :EXTENDED;
        temp : data_type;
        strg : STRING[max_touch_text];
BEGIN
        str(nr:3,strg);
        touch_text:='Nr:'+strg;
        calc:=Round(mess_freq * nr * spec_step / calc_intervall);
        IF calc>99 THEN BEGIN
            Str(calc:5,strg);
            IF Length(strg)<6 THEN
                strg:=','+strg
            ELSE
                strg[1]:=',';
        END ELSE BEGIN
            calc2:=mess_freq * nr * spec_step / calc_intervall;
            Str(calc2:5:1,strg);
            strg:=','+strg;
        END;
```



```
        touch_text:=touch_text+strg;
        temp:=sin_koeff^[nr];
        str(temp:5,strg);
        touch_text:=touch_text+'Hz, sin:'+strg+'mV ';
        temp:=cos_koeff^[nr];
        str(temp:5,strg);
        touch_text:=touch_text+'cos:'+strg+'mV';
END;

FUNCTION play_sample( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        delta : WORD;
BEGIN
        IF data_mark[2]> data_mark[1] THEN
            delta:=data_mark[2]-data_mark[1]
        ELSE
            delta:=data_mark[1]-data_mark[1];
        put_sound(@work_data^[data_mark[1]],delta,mess_freq);
        play_sample:=TRUE;
END;




FUNCTION do_zoom( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        d_first, d_last,
        d_center, d_delta : WORD;
BEGIN
        d_delta:=(fg_x2-fg_x1-2*innenrand) DIV (2*zoom_faktor) ;      { nur jede vierte
spalte }
        d_center:=(used_data * rel_x) DIV 1000; { /1000 da rel_x in Promille }
        IF d_center > used_data-d_delta THEN
            d_center:=used_data-d_delta;
        IF d_center < d_delta THEN
            d_center:=d_delta;
        data_mark[mark_to_set]:=d_center;
```

```
        do_zoom:=display_zoom(0,0,0);
END;

PROCEDURE do_shift(delta : INTEGER);
VAR
        first_data, last_data: WORD;
        middle : WORD;
BEGIN
        zoom_window.get_data(first_data,last_data);
        IF ( first_data+delta >= 0 ) AND ( last_data+delta <=used_data )
                        THEN BEGIN
                zoom_window.set_mark( 2+mark_to_set, -1,TRUE);
                zoom_window.set_data(work_data,first_data+delta,last_data+delta);
                mess_window.set_mark( 1, first_data,FALSE);
                mess_window.set_mark( 2, last_data,FALSE);
                middle:=(first_data+last_data) div 2 +delta;
                mess_window.set_mark( 2+mark_to_set, middle,TRUE);
                zoom_window.set_mark( 2+mark_to_set, middle,TRUE);
                data_mark[mark_to_set]:=middle;
        END;
END;

FUNCTION do_shift_left( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        mr: mouse_rec;
        delta : INTEGER;
BEGIN
        delta:=-1;
        if rel_x<250 THEN delta:=-4;
        do_shift(delta);
        IF (rel_x>500) THEN BEGIN
                get_mouse(mr);
                WHILE mr.left DO
                        get_mouse(mr);
        END;
        do_shift_left:=TRUE;
```



```
END;

FUNCTION do_shift_right( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        mr: mouse_rec;
        delta : INTEGER;
BEGIN
        delta:=1;
        if rel_x>750 THEN delta:=4;
        do_shift(delta);
        IF (rel_x<500) THEN BEGIN
                get_mouse(mr);
                WHILE mr.left DO
                        get_mouse(mr);
        END;
        do_shift_right:=TRUE;
END;

FUNCTION set_spec_koeff_mode( x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        hide_all;
        spec_display_mode:=koeff;
        set_spec_koeff_mode:=display_spec(0,0,0);
END;
        set_spec_koeff_mode:=TRUE;
END;

FUNCTION set_spec_phase_mode( x_rel, y_rel : LONGINT; nr : WORD): BOOLEAN;
BEGIN
IF NOT spec_calculated THEN show_help(321) ELSE BEGIN
        hide_all;
        spec_display_mode:=phases;
        set_spec_phase_mode:=display_spec(0,0,0);
END;
        set_spec_phase_mode:=TRUE;
```

```
END;

PROCEDURE hide_all;
BEGIN
        mark_to_set:=0;
        zoom_window.disable;
        back_zoom_window.disable;
        spec_window.disable;
        phases_window.disable;
        shift_left.disable;
        shift_right.disable;
        swap_marks.disable;
        sin_window.disable;
        cos_window.disable;
        mess_window.set_mark(1,-1,FALSE);
        mess_window.set_mark(2,-1,FALSE);
        mess_window.set_mark(3,-1,FALSE);
        mess_window.set_mark(4,-1,FALSE);
END;

FUNCTION display_spec( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
BEGIN
        mark_to_set:=0;
        hide_all;
        CASE spec_display_mode OF
                phases : BEGIN
                        spec_window.set_data(spektrum,0,30);
                        phases_window.set_data(phase,0,30);
                        spec_window.enable;
                        phases_window.enable;
                END;
                koeff : BEGIN
                        sin_window.set_data(sin_koeff,0,used_spec-1);
                        cos_window.set_data(cos_koeff,0,used_spec-1);
                        sin_window.enable;
                        cos_window.enable;
```



```
                END;
        END;
        mess_window.set_mark(3,data_mark[1],FALSE); { nur wg TRUE }
        mess_window.set_mark(4,data_mark[2],FALSE);
        mess_window.set_reaction(dummy,'Kurve der Messwerte',0);
        display_spec:=TRUE;
END;

FUNCTION display_only_zoom( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
BEGIN
        hide_all;
        zoom_window.set_reaction(dummy,'Lupe (des roten Ausschnitts)',0);
        display_only_zoom:=display_zoom(rel_x, rel_y, nr);
END;



FUNCTION display_zoom( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        d_center, d_delta, tmp : WORD;
        strg : STRING[5];
BEGIN
        spec_window.disable;            { kostet nichts wenn nicht dargestellt }
        phases_window.disable;
        sin_window.disable;
        cos_window.disable;

        d_center:=data_mark[mark_to_set];
        d_delta:=(fg_x2-fg_x1-2*innenrand) DIV (2*zoom_faktor) ;
        IF ((d_delta*2)>=used_data) THEN d_delta:=used_data DIV 10;
        IF (d_center+d_delta >= used_data) THEN BEGIN
                d_center:=used_data-d_delta-1;
                data_mark[mark_to_set]:=d_center;
        END;
        IF (d_center<d_delta) THEN BEGIN
                d_center:=d_delta;
```

```
                data_mark[mark_to_set]:=d_delta;
        END;
        IF (d_center+d_delta >= used_data) THEN BEGIN
                d_center:=used_data div 2;
                data_mark[mark_to_set]:=used_data div 2;
                d_delta:=d_center div 2;
                show_help(183);
        END;
        zoom_window.set_mark(3,-1,FALSE);
        zoom_window.set_mark(4,-1,TRUE);
{*} IF (data_mark[1]>data_mark[2]) THEN BEGIN
                tmp:=data_mark[1];
                data_mark[1]:=data_mark[2];
                data_mark[2]:=tmp;
                mark_to_set:=3-mark_to_set;
        END;
        zoom_window.set_data(work_data,d_center-d_delta,d_center+d_delta);
        zoom_window.enable;
        IF (mark_to_set<>0) THEN BEGIN
                zoom_window.set_mark(2+mark_to_set,data_mark[mark_to_set],TRUE);


                mess_window.set_mark(2+mark_to_set,data_mark[mark_to_set],TRUE);    {
nur wg TRUE }
                mess_window.set_mark(5-mark_to_set,data_mark[3-mark_to_set],FALSE);
        END
        ELSE BEGIN
                mess_window.set_mark(3,data_mark[1],FALSE);
                mess_window.set_mark(4,data_mark[2],FALSE);
        END;
        mess_window.set_mark(1,d_center-d_delta,FALSE);
        mess_window.set_mark(2,d_center+d_delta,FALSE);

        IF (mark_to_set<>0) THEN BEGIN


                d_center:=data_mark[3-mark_to_set];
                IF (d_center<d_delta) THEN BEGIN
                        d_center:=d_delta;
```



```
                        data_mark[3-mark_to_set]:=d_delta;
                END;
                IF (d_center+d_delta >= used_data) THEN BEGIN
                        d_center:=used_data-d_delta;
                        data_mark[3-mark_to_set]:=used_data-d_delta;
                END;
                back_zoom_window.set_data(work_data,d_center-
d_delta,d_center+d_delta);
                back_zoom_window.ignore_first_time;
                back_zoom_window.enable;
                shift_left.enable;
                shift_right.enable;
                swap_marks.enable;
                Str(mark_to_set,strg);
                mess_window.set_reaction(do_zoom,'Auswahl des Lupenbereichs für Marke
'+strg,2);
        END ELSE BEGIN
                mess_window.set_reaction(do_zoom,'Auswahl des allgemeinen
Lupenbereichs',2);
        END;
        display_zoom:=TRUE;
END;

FUNCTION set_first_mark( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
BEGIN
        mark_to_set:=1;
        zoom_window.set_reaction(dummy,'Lupe um Marke 1 ( in weiß )',0);
        set_first_mark:=display_zoom(0,0,0);
END;

FUNCTION set_second_mark( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
BEGIN
        mark_to_set:=2;
        zoom_window.set_reaction(dummy,'Lupe um Marke 2 ( in weiß )',0);
        set_second_mark:=display_zoom(0,0,0);
END;
```

```
FUNCTION do_swap_marks( rel_x, rel_y: LONGINT; nr: WORD) : BOOLEAN;
VAR
        mr : mouse_rec;
BEGIN
        IF mark_to_set=1 THEN
                do_swap_marks:=set_second_mark(0,0,0)
        ELSE
                do_swap_marks:=set_first_mark(0,0,0);
        REPEAT
                get_mouse(mr);
        UNTIL mr.left=FALSE;
END;

PROCEDURE fg_get_mem;
BEGIN
        work_data:=New(data_array_ptr);
        spektrum:=New(spec_array_ptr);
        phase:=New(spec_array_ptr);
        sin_koeff:=New(spec_array_ptr);
        cos_koeff:=New(spec_array_ptr);
        spec_calculated:=FALSE;
END;

PROCEDURE fg_setup;
var
        d               : Integer;
        temp,f1,f2,f3,f4,amp  : Extended;
        mouse : Mouse_Rec;
        text    : string[20];
begin
        IF MaxAvail < (SizeOf(data_array)+4*SizeOf(spec_array)) THEN BEGIN
                show_help(104);
                halt(1);
        END;
        fg_get_mem;
        used_data:=1000;
```



```
        used_spec:=100;
        spec_step:=1;
        data_mark[1]:=used_data div 4;
        data_mark[2]:=used_data * 3 div 4;
        for d:=0 to used_data do begin
                work_data^[d]:=0;
        end;
        mess_max:=max_value;
        mess_min:=-max_value;
        mess_freq:=1000;

        FOR d:=0 TO max_spect DO
                spektrum^[d]:=0;
END;

PROCEDURE fg_init(x,y,xx,yy,dy,frame_color,graph_color,area_color : Word);
BEGIN
        fg_setup;
        fg_x1:=x;                                       {linker Rand}
   fg_x2:=xx;                   {rechter Rand}
   fg_y1:=y;                    {oberer Rand}
   fg_y2:=yy;                   {unterer Rand}
   fg_delta:=dy;                {Abstand zwischen den Fenstern}
   fg_m1:=fg_y1+(fg_y2-fg_y1-fg_delta) div 2;{unterer Rand oberes Fenter}
   fg_m2:=fg_m1+fg_delta;               {oberer Rand unteres Fenster}
        fg_zx:=(fg_x2+fg_x1)div 2;          { x-zentrum/mitte }
        fg_frame_col:=frame_color;
        fg_graph_col:=graph_color;
        fg_area_col :=area_color;
        fg_back_col :=area_color;
        spec_begin:=0;
        mess_window.init(fg_x1,fg_y1,fg_x2-fg_x1,fg_m1-fg_y1,area_color,frame_color,
                        graph_color XOR area_color,80,0,no_hot_key,'Kurve der
Messwerte',dummy,'Kurve',left);
        higru_window.init(fg_x1,fg_y1,fg_x2-fg_x1,fg_m1-fg_y1,area_color,frame_color,
                        col_higru                                       XOR
area_color,0,0,no_hot_key,'Higru',dummy,'Hintergrund',right);
```

```
        spec_window.init(fg_x1,fg_m2,fg_x2-fg_x1,fg_y2-fg_m2,area_color,frame_color,
                         graph_color                                    XOR
area_color,82,1,no_hot_key,'Spektrum',dummy,'Spektrum',left);
        phases_window.init(fg_x1,fg_m2,fg_x2-fg_x1,fg_y2-
fg_m2,area_color,frame_color,
                         col_phase                                      XOR
area_color,99,1,no_hot_key,'Phases',dummy,'Phasen',right);
        sin_window.init(fg_x1,fg_m2,fg_x2-fg_x1,fg_y2-fg_m2,area_color,frame_color,
                         col_sin                                        XOR
area_color,105,0,no_hot_key,'Koeffizienten',dummy,'Sinus',left);
        cos_window.init(fg_x1,fg_m2,fg_x2-fg_x1,fg_y2-fg_m2,area_color,frame_color,
                         col_cos                                        XOR
area_color,99,1,no_hot_key,'cos_koeff',dummy,'Cosinus',right);
        zoom_window.init(fg_x1,fg_m2,fg_x2-fg_x1,fg_y2-fg_m2,area_color,frame_color,
                         graph_color                                    XOR
area_color,81,0,no_hot_key,'Vergrösserung',dummy,'Lupe',left);
        back_zoom_window.init(fg_x1,fg_m2,fg_x2-fg_x1,fg_y2-
fg_m2,area_color,frame_color,
                         col_higru                                      XOR
area_color,99,1,no_hot_key,'back_zoom',dummy,'Umgebung andere Marke',right);
        shift_left.init(fg_zx-8*char_width-2,fg_m1+2,8*char_width,fg_m2-fg_m1-
4,Blue,Yellow,Yellow,83,
                         '<','Verschieben der Lupe nach links','<< | <',do_shift_left);
        shift_right.init(fg_zx+2,fg_m1+2,8*char_width,fg_m2-fg_m1-
4,Blue,Yellow,Yellow,83,
                         '>','Verschieben der Lupe nach rechts',' > | >>',do_shift_right );
        swap_marks.init(fg_x1+10*char_width,fg_m1+2,15*char_width,fg_m2-fg_m1-
4,Blue,Yellow,Yellow,105,
                         'v','Setzen der anderen Marke','andere Marke',do_swap_marks
);


        mess_window.set_data(work_data,0,used_data-1);
        spec_window.set_data(spektrum,0,used_spec-1);
        spec_window.set_analytic(TRUE,change_touch_text);
        spec_window.disable;
        spec_window.ignore_first_time;
        phases_window.set_data(phase,0,used_spec-1);
        phases_window.disable;
        phases_window.ignore_first_time;
        phases_window.set_scale(0,0,clocks);
```



```
        zoom_window.set_data(work_data,0,1);
        zoom_window.set_girwidz_color(area_color XOR LightRed);
        back_zoom_window.set_data(work_data,0,1);
        back_zoom_window.disable;
        back_zoom_window.ignore_first_time;
        back_zoom_window.set_scale(-max_value,max_value,dotted);
        shift_left.disable;
        shift_right.disable;
        swap_marks.disable;
        sin_window.ignore_first_time;
        sin_window.set_analytic(TRUE,change_sin_touch_text);
        sin_window.disable;
        cos_window.ignore_first_time;
        cos_window.disable;
        higru_window.ignore_first_time;
        higru_window.disable;


        scale_spec;
        scale_mess;
        mark_to_set:=0;
        zoom_faktor:=1;
END;



END.
```

E.11. Die Unit F_HELP

```
{ f_help ist die Unit, die die Hilfstexte anzeigt }
unit f_help;

{$F+} {$O+}
interface
        FUNCTION explain_help( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
        procedure show_help(nr : Integer);
        procedure init_help;

implementation
USES gui, gui_def, gui_tool, DOS, CRT, f_mouse, f_text;
var
        in_help_flag : BOOLEAN;

FUNCTION explain_help( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
BEGIN
        show_help(142);
        explain_help:=TRUE;
END;

procedure show_help( nr : Integer);
var
        dat                     : Text;
        strg            : string[80];
   suchstrg     : string[10];
        help_text       : text_rec;
        d                       : Integer;
        normal_help : BOOLEAN;
begin
IF NOT in_help_flag THEN BEGIN
        WHILE nr>0 DO BEGIN
                in_help_flag:=TRUE;   { kein rekursiver Hilfeaufruf }
                show_status('Suche Hilfstext');
                str(nr:2, strg);
                suchstrg := '[' + strg + ']';
```



```
                assign ( dat, 'helpdat.pas');
                reset(dat);
                readln(dat, strg);
                while (strg<>suchstrg) and (not eof( dat ) ) do
                begin
                        Readln(dat,strg);
                end;
                IF (eof(dat)) THEN BEGIN
                  FOR d:=2 to 4 DO help_text.strg[d]:='';
                  help_text.strg[1]:='Interner Fehler';
                  help_text.strg[5]:='Es ist leider kein Hilfstext verfügbar';
                  help_text.strg[6]:='   Interne Nummer:'+suchstrg;
                  help_text.last_string:=6;
                END
                ELSE BEGIN
                        d:=1;
                        while (strg[1]<>'#') and (not eof(dat) and (d<25)) do
                        begin
                                Readln(dat, strg);
                                if (d<=24) then
                                        help_text.strg[d]:=strg;
                                Inc(d);
                        end;
                        help_text.last_string:=d-2;
                END;
                close( dat);
                normal_help:=TRUE;                                      { keinen weiteren
Hilfstext }
                IF strg[2]='>' THEN BEGIN
                        suchstrg[0]:=CHAR(3);
                        suchstrg[1]:=strg[3];
                        suchstrg[2]:=strg[4];
                        suchstrg[3]:=strg[5];
                        IF (strg[5]<'0') OR (strg[5]>'9') THEN suchstrg[0]:=CHAR(2);
                        Val(suchstrg,nr,d);
                        IF d<>0 THEN nr:=0
                        ELSE BEGIN
```

```
                        d:=big_alert(help_text,'  Mehr Hilfe | Genug Hilfe ',TRUE);
                        normal_help:=FALSE;
                    END;
                    IF (d=2) THEN nr:=-1;
                END ELSE nr:=0;
                IF nr=0 THEN
                    d:=big_alert(help_text,'Gelesen',TRUE);
                in_help_flag:=FALSE;
        END;
END;
end;

procedure init_help;
BEGIN
        in_help_flag:=FALSE;
END;

end.
```


E.12. Die Unit F_LERN

```
UNIT f_lern;
{$F+}{$O+}
INTERFACE
        FUNCTION lernen( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;

IMPLEMENTATION
{ Ich danke Herrn R.Girwidz z.Zt. Uni Würzburg/PhysikDidaktik für die Routinen
  die er mir zur Verfügung gestellt hat. Sie betreffen die hardwarenahe Bearbeitung
  (Laden/Palette...) des VGA-Standardbildschirms. }

USES dos,crt,f_mouse,gui_def,gui_tool,gui,Graph,
                        f_datei,f_gr_def,f_global,f_graph,f_help;
CONST
        lern_datei : ARRAY[1..3] OF STRING[14]=('f_lern.dat','f_lern2.dat','f_lern3.dat');
TYPE
                file_name = STRING[15];
{**** Beginn der v. H.R.Girwidz übernommenen & stark modifizierten Routinen ********}
                Puffer22Prt = Array[0..38400] of BYTE;
      Puffer2List = ^Puffer22Prt;
      RGBRecord       = Record   R, G, B : Byte;   END;
      PCXHeaderRecord = Record
        Id          : Byte;
        Version     : Byte;
        Coding      : Byte;
        Bits_Pixel  : Byte;
        MinX,MinY   : Word;
        MaxX,MaxY   : Word;
        ResX,ResY   : Word;
        Palette     : Array[0..15] of RGBRecord;
        Reserved    : Byte;
        Planes      : Byte;
        Planesize   : Word;
        PaletteTyp  : Word;
        LineCount   : Word;
```

```
        Farbliste   : Array[0..16] of Byte;
        Dummy       : Array[89..127] of Byte
      END;

VAR
      Puffer2   : Array[0..3] of Puffer2List;
      screenAdr : Array[0..38400] of Byte Absolute $A000:$0000;
      _file     : File;
      ch        : Char;
      Error     : Word;
      rr        : Byte;
      VGAPlane  : Integer;
      Reg       : Registers;
                  PaletteVGA256 : Array[0..255,0..2] of Byte;
                  PCXHeader : PCXHeaderRecord;

PROCEDURE  PCXHeader_auswerten;
  Var  i : Integer;
   BEGIN
    Seek( _file, 0 );
    BlockRead( _file, PCXHeader, 128 );
    With  PCXHeader DO
     BEGIN
      If (Farbliste[1]=0) AND (Farbliste[2]=0) AND (Farbliste[3]=0) AND
        (Farbliste[4]=0) AND (Farbliste[5]=0) AND (Farbliste[6]=0) AND
        (Farbliste[7]=0) AND (Farbliste[8]=0) AND (Farbliste[9]=0) THEN EXIT;
          END;
END;



{-------------------------------------------------------------------------}
PROCEDURE  VGA_Palette_laden;
BEGIN
                  Seek( _file, FileSize( _file ) - 768 );
                  BlockRead( _file, PaletteVGA256, 768 );
END;
```



```
FUNCTION init_bpl:BOOLEAN;
VAR
        d:WORD;
        oom : BOOLEAN; {out of memory}
BEGIN                                                                   {
SetzeVideoModus }
        Reg.AH := $0;   Reg.AL := $12;   Intr( $10, Reg );
        oom:=FALSE;
        d:=0;
        WHILE (d<4) AND (NOT oom) DO BEGIN
                IF (MaxAvail<SizeOf(Puffer22PRT)) THEN BEGIN
                        {show_help(178);geht nicht wg. Grafik aus!}
                        oom:=TRUE;
                END   ELSE
                        NEW(  Puffer2[d]  );   {  Speicher  wird  wg.  fragmentiertem  Heap
einzeln allokiert!}
                INC(d);
        END;
        DEC(d);
        WHILE oom AND (d>0) DO BEGIN
                Dispose(Puffer2[d-1]);
                DEC(d);
        END;
        init_bpl:=NOT oom;
END;

PROCEDURE load_bpl( Bpl_Bild : String );
BEGIN
        Assign( _file, BPL_Bild );
         {$I-}
         ReSet( _file, 1 );
         Error := IOResult;
         IF Filesize(_file)<150000 THEN Error:=1;
         {$I+}
         IF Error=0 THEN BEGIN
                PCXHeader_auswerten;
```

```
                VGA_Palette_laden;
                Seek( _file, 128 );
                FOR rr:=0 TO 3 DO   BlockRead( _file, Puffer2[rr]^, 38400 );
                CLOSE( _file );
        END;
END;

PROCEDURE show_bpl;
BEGIN                   { Farbliste lesen über Int$10, Fkt$10 }
        Reg.AH := $10;   { Fkt 10 }
        Reg.AL := $2;
        Reg.DX := OFS( PcxHeader.Farbliste );
        Reg.ES := SEG( PCXHeader.Farbliste );
        Intr( $10, Reg );

        Reg.AH := $10;    { Farbregisterblock laden über Int$10, Fkt$10/12 }
        Reg.AL := $12;
        Reg.BX := 0;
        Reg.cx := 255;
        Reg.DX := OFS( PaletteVGA256 );
        Reg.ES := SEG( PaletteVGA256 );
        Intr( $10, Reg );

        VGAPLANE := 1;
        For rr:=0 TO 3 DO
                BEGIN
                        Port[$3c4]:=2;    Port[$3c5]:=VGAPlane;
                        MOVE( Puffer2[ rr ]^, ScreenAdr, 38400 );
                        VGAPLANE := VGAPlane shl 1;
                END;
END;

PROCEDURE disable_bpl;
BEGIN
  Dispose(Puffer2[0]); Dispose(Puffer2[1]); Dispose(Puffer2[2]); Dispose(Puffer2[3]);
END;
```



```
{********** Ende der v. H.R.Girwidz übernommenen & modifizierten Routinen **********}

FUNCTION do_learn(learn_file : file_name): BOOLEAN;
VAR
        d, oldd, last_pic : WORD;
        strg : ARRAY[1..50] OF file_name;
        dat : Text;
        mr : mouse_rec;
        c : CHAR;
BEGIN
IF FileExists(learn_file) THEN BEGIN
        assign ( dat, learn_file);
        reset(dat);
        last_pic:=0;
        WHILE (NOT Eof(dat)) AND (last_pic<50) DO BEGIN
                Inc(last_pic);
                readln(dat, strg[last_pic]);
                IF NOT fileexists(strg[last_pic]) THEN DEC(last_pic);
        END;
        IF init_bpl THEN BEGIN
                d:=1;
                oldd:=0;
                load_bpl(strg[d]);
                WHILE (d<=last_pic) DO BEGIN
                        c:=' ';
                        IF d<>oldd THEN show_bpl;
                        oldd:=d;
                        IF (d<last_pic) THEN
                                load_bpl(strg[d+1]);
                        REPEAT
                        UNTIL KeyPressed;
                        IF KeyPressed THEN c:=ReadKey;
                        IF c=CHAR(27) THEN d:=1000;
                        IF c=CHAR(0) THEN BEGIN
                                c:=ReadKey;
                                IF c=CHAR(77) THEN c:='+';
```

```
                        IF c=CHAR(75) THEN c:='-';
                        IF c=CHAR(59) THEN BEGIN
                                load_bpl('Lernhelp.bpl');
                                show_bpl;
                        END;
{$IFDEF extras}
                        IF c=CHAR(66) THEN BEGIN
                                show_mouse;
                                edit_document;
                                hide_mouse;
                        END;
{$ENDIF}
                END;
                IF (c='-') AND (d>1) THEN DEC(d);
                IF (c='+') THEN INC(d)
                ELSE
                        IF (d<>oldd) AND (d<=last_pic) THEN
                                load_bpl(strg[d]);
        END;
        disable_bpl;
        do_learn:=TRUE;
    END   ELSE do_learn:=FALSE;
END ELSE BEGIN
    do_learn:=TRUE;
    show_help(180);
END;
END;


FUNCTION lernen( rel_x, rel_y : LONGINT; nr : WORD):BOOLEAN;
VAR
    mr : mouse_rec;
    txt : Select_Array;
    d : WORD;
    menptr : Menu_Button_PTR;
    titel : STRING;
```



```
    lernprg_nr : WORD;
    screen_saver : INTEGER;
    memo_flag, success : BOOLEAN;
BEGIN
    IF GetMaxY>=479 THEN BEGIN
        get_mouse(mr);      { SelectBox erscheint an Mausposition, Abfrage bei
create_select auf Bildschirmränder }
        titel:='Wählen Sie Ihr Wunschprogramm zum Lernen';
        txt[1]:='Grundlagen (Maus, Menü, F.-Analyse)';
        txt[2]:='Weitere Analyse der Daten';
        txt[3]:='Messen';
        create_select(menptr, txt, 3, mr.x, mr.y,titel);
        d:=handle_select(menptr);
        dispose(menptr,done);
        memo_flag:=FALSE;
        IF (MaxAvail<(4*SizeOf(Puffer22PRT))) THEN memo_flag:=TRUE;
        IF d>0 THEN BEGIN
            lernprg_nr:=d;
            IF memo_flag THEN
                save_temp;
            hide_all;
            mess_window.disable;
            screen_saver:=save_big_screen(0,0,GetMaxX,GetMaxY);
            hide_mouse;
            IF memo_flag THEN BEGIN
                Dispose(work_data);   {mehr Stoff...}
                Dispose(spektrum);
                Dispose(phase);
                Dispose(sin_koeff);
                Dispose(cos_koeff);
            END;

{$IFNDEF extras}
            RestoreCrtmode;
{$ENDIF}
            success:=do_learn(lern_datei[lernprg_nr]);
    {$IFNDEF extras}
```

```
                SetGraphMode(GetGraphMode);
        {$ENDIF}
                restore_big_screen(screen_saver);
                IF NOT success THEN show_help(178);
                IF memo_flag THEN
                        fg_get_mem; {reserviert wieder Speicher für alles...}
                mess_window.set_data(work_data,0,1);
                mess_window.enable;
                IF memo_flag THEN
                        load_temp;
            END;
        END     {of IF Grafik=VGA}
        ELSE BEGIN
            show_help(179);
        END;
        lernen:=TRUE;
END;

END.
```


E.13. Die Unit F_MAIN

```
program f_main;
{$F+}
uses f_global,Overlay, Graph,gui_def,gui,f_mouse,f_graph,f_help,f_analyse,
        f_datei,f_mess,f_text,f_change,f_calc,gui_tool,f_sound,f_menu;
{$O f_help}
{$O f_lern}
{$O f_datei}
{$O f_change}
{$O f_calc}
{$O gui_tool}
{$O f_mess}
{$O f_menu}
{$O f_analyse}



VAR
        graphdriver,
        graphmode,
        errorcode       : INTEGER;
        hires_flag,
        exit_flag       : BOOLEAN;

        flag : BOOLEAN;
        x : INTEGER;
        s : STRING[20];
BEGIN
        IF TRUE THEN BEGIN
                OvrInit('f_main.ovr');
                x:=OvrResult;
                OvrInitEMS;
                x:=OvrResult;
        END;

        s:=ParamStr(1);
        init_global_params;
```

```
   GraphDriver := Detect;
{$IFNDEF extras}
        IF param_svga THEN
                GraphDriver := InstallUserDriver ('svga', NIL);
{$ENDIF}
        IF (GraphDriver < 0) THEN
                GraphDriver:=Detect
        ELSE
                GraphMode := 4;         { 800*600 }
        hires_flag := TRUE;       { kein automat. Mauszeiger }

   InitGraph(GraphDriver, GraphMode, 'bgi');
   ErrorCode   := GraphResult;
        IF ErrorCode <> grOK THEN
        BEGIN
                Writeln ('Grafikfehler: ', GraphErrorMsg(ErrorCode));
                Halt(1);
        END;
        SetColor(col_main);
        Rectangle(0, 0, GetMaxX, GetMaxY);
        SetFillStyle(CloseDotFill,col_stat_back);
        Bar(0+1,0+1,GetMaxX-1,GetMaxY-1);
        mouse_reset(hires_flag,GetMaxX-16,GetMaxY-8);
        f_text_init;
        init_help;
        init_gui;
        show_status('--- Fourier ---');
        init_higru;
        fg_init(20,3*char_height,GetMaxX-20,GetMaxY-2*char_height,(5*char_height)div
3,col_main,col_spec,col_graph_back);
        calc_init;
        init_menu;
        show_menu;
        init_mess;
        flag:=set_standards(0,0,0);
        init_sound;
```



```
        get_user_name;
        IF param_master THEN
                flag:=switch_menu_on(0,0,0)
        ELSE
                say_hello;
        load_messung; {überprüft auf user_id!}
                { früher...      flag:=main_menu.handle_klick(0,0,0); { Dieses Object hat
not_leave gesetzt}
        main_menu.direct_call;
        main_menu.done; { Gibt alle Speicher wieder frei... }
        CloseGraph;
        WriteLn('Das war Christian Bienmüllers FourierProgramm');
end.
```

E.14. Die Unit F_MENU

```
unit f_menu;
{$F+} {$O+}
Interface
uses f_global,Graph, gui_def, gui,f_mouse,f_graph,f_help,f_analyse,
     f_datei,f_mess,f_text,f_change,f_calc,gui_tool,f_sound,f_lern;

VAR
    main_menu    : Menu_Button;          { DAS Hauptmenu}

    PROCEDURE init_menu;
    PROCEDURE show_menu;
    PROCEDURE say_hello;
    FUNCTION  switch_menu_on(rel_x, rel_y : LONGINT;nr:WORD):BOOLEAN;
    FUNCTION  switch_menu_off(rel_x, rel_y : LONGINT;nr:WORD):BOOLEAN;
    PROCEDURE init_global_params;
    PROCEDURE get_user_name;

IMPLEMENTATION
CONST
    max_ts = 20;   { ts bedeutet to_switch und entspricht einer Liste (ts_list)
                                    die zum Umschalten zw. Profi/AnfängerMenü
angelegt wird }
VAR
    menu_list     : ARRAY[1..5] OF Menu_Button; { Die einzelnen PullDowns}
    sub_menu      : ARRAY[1..2] OF Menu_Button; { Submenus, numerierung egal}
    ts_list       : ARRAY[1..max_ts] OF Button_PTR; { Liste der an/ausschaltbaren
Menüs }
    ts_invers  : Button_Ptr;
    last_ts       : WORD;

FUNCTION not_implement( mx, my: LONGINT; nr : WORD) : BOOLEAN;
VAR
    mr : mouse_rec;
    zahl, text: STRING[40];
BEGIN
```



```
    Str(nr,zahl);
    text:='Funk. '+zahl+' ist nicht impl. ';
    Str(mx,zahl);
    text:=text+zahl+' ';
    Str(my,zahl);
    text:=text+zahl;
    show_status(text);
    REPEAT
        get_mouse(mr);
    UNTIL NOT mr.left;
    not_implement:=True;
END;

FUNCTION quit( mx, my:LONGINT; nr : WORD) : BOOLEAN;
VAR
    flag : BOOLEAN;
    wtd,
    strg : STRING[45];

BEGIN
    IF param_noexit AND (NOT param_master) THEN BEGIN
        strg:='';
        wtd:='Geben Sie das Codewort ein:';
        input(wtd,strg,10);
        IF strg='Ossau *93*' THEN flag:=TRUE
        ELSE BEGIN
            flag:=FALSE;
            show_help(181);
        END;
    END
    ELSE flag:=TRUE;
    IF flag THEN BEGIN
        main_menu.set_leave(FALSE);
        mess_window.force_enabled_flag(FALSE);
    END;
    quit:=TRUE;
```

```
END;

FUNCTION ts(ptr : Button_PTR):Button_PTR;
BEGIN
        IF last_ts<max_ts THEN BEGIN
                INC(last_ts);
                ts_list[last_ts]:=ptr;
        END;
        ts:=ptr;                        { Eine Art von Durchschleifen }
END;

FUNCTION ts_inv(ptr : Button_PTR):Button_PTR;
BEGIN
        ts_invers:=ptr;
        ts_inv:=ptr;                        { Eine Art von Durchschleifen }
END;

FUNCTION switch_menu_on(rel_x, rel_y : LONGINT;nr:WORD):BOOLEAN;
VAR
        d: WORD;
BEGIN
        FOR d:=1 TO last_ts DO BEGIN
                ts_list[d]^.force_enabled_flag(TRUE);
        END;
        ts_invers^.force_enabled_flag(FALSE);
        switch_menu_on:=TRUE;
END;

FUNCTION switch_menu_off(rel_x, rel_y : LONGINT;nr:WORD):BOOLEAN;
VAR
        d: WORD;
BEGIN
        FOR d:=1 TO last_ts DO BEGIN
                ts_list[d]^.force_enabled_flag(FALSE);
        END;
        ts_invers^.force_enabled_flag(TRUE);
```



```
        switch_menu_off:=TRUE;
END;

PROCEDURE show_menu;
VAR
        d : WORD;
BEGIN
        d:=1;
        hide_mouse;
        SetFillStyle( SolidFill , col_stat_back );
        SetColor(col_main);
        Bar ( 1, 1, GetMaxX -2, char_height +5 );
        Rectangle ( 0, 0, GetMaxX -1, char_height +6 );
        show_mouse;
END;

PROCEDURE init_menu;
VAR
        y_0,x_0,dx,dy : WORD;
        ignore : BOOLEAN;
        stat_line,
        info_line : Button_ptr;
TYPE
        OBP = ^Order_Button;

BEGIN
        IF MaxAvail < (30*SizeOf(Order_Button)+5*SizeOf(Menu_Button)) THEN
BEGIN
                show_help(105);
                halt(1);
        END;
        last_ts:=0;

        dy:=char_height+2;
        x_0:=2;
        y_0:=2;
        dx:=7*char_width;
```

```
{------------------ Erste Spalte -------------------}
menu_list[1].init(x_0,y_0,dx,dy,col_stat_back,col_main,col_main,10,
                                    'd','Allgemeine Datenverarbeitung','Datei ');
dx:=11*char_width;
Inc(y_0,dy+4);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,11,
                    'l','Laden älterer Messdaten von
Festplatte','Laden',load_data)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,12,
                    's','Sichern der aktuellen
Messdaten','Speichern',save_data)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,13,
                    'e','Alte Messdaten von Festplatte
löschen','Entfernen',delete_file)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,18,
                    'w','Daten auswählen/Anzahl
einschränken','ausWahl',do_select_work_area)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,14,
                    'd','Druckt die aktuelle Messung aus','Drucken',print)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                    no_hot_key,'','---------',dummy)),FALSE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,15,
                    'm','Führt eine neue Messung
durch','Messung',do_mess)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,17,
```



```
                    'a','Ausgabe der Daten als
Töne/Klang','Abspielen',play_sample)),TRUE);
Inc(y_0,dy);
menu_list[1].add_button(ts(New(                                     OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                    no_hot_key,'','---------',dummy))),FALSE);
Inc(y_0,dy);
menu_list[1].add_button(ts(New(                                     OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,16,
                    'b','Ende des Programms','Beenden',quit))),TRUE);


x_0:=x_0+7*char_width+1;
y_0:=2;
dx:=9*char_width;
{------------------ Zweite Spalte -------------------}
menu_list[2].init(x_0,y_0,dx,dy,col_stat_back,col_main,col_main,20,
                                    'r','Datenverarbeitung','Rechnen');
dx:=17*char_width;
Inc(y_0,dy+4);
menu_list[2].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,21,
                    'k','Berechnet aus der Kurve das Spektrum','Kurve-
>Spectr',calc_spec)),TRUE);
Inc(y_0,dy);
menu_list[2].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,22,
                    's','Berechnet aus dem Spektrum eine Kurve','Spectr-
>Kurve',calc_kurve)),TRUE);
Inc(y_0,dy);
menu_list[2].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                    no_hot_key,'','----------',dummy)),FALSE);
Inc(y_0,dy);
menu_list[2].add_button(New(                                        OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,23,
                    'p','Wählen Sie volle Periode(n) zum
Rechnen','Periodenwahl',set_first_mark)),TRUE);
Inc(y_0,dy);
menu_list[2].add_button(ts(New(                                     OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,24,
```

```
                    'm','Stellen Sie hier Messparameter
ein','Messparameter',get_mess_parameters))),TRUE);
    Inc(y_0,dy);
    menu_list[2].add_button(ts(New( OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,25,
                    'o','Festlegen der Anzahl v.
Oberwellen','Oberwellen',set_spec_used))),TRUE);
    Inc(y_0,dy);
    menu_list[2].add_button(ts(New( OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                    no_hot_key,'','----------',dummy))),FALSE);
    Inc(y_0,dy);
    { zwei Submenü initialisieren, das erste wird in das zweite eingebunden }
    sub_menu[1].init(x_0+dx,y_0+(7*dy) div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,60,
                    't','versch. Eingabe des Spektrums in Tabelle','Tabelle...');
    sub_menu[1].add_button(New( OBP,
Init(x_0+2*dx,y_0+4*dy,dx,dy,col_menu_back,col_menu_back,col_menu,61,
                    'a','Verändern des aktuelle Spektrums','Aktuelle
Tab',tab_eingabe)),TRUE);
    sub_menu[1].add_button(New( OBP,
Init(x_0+2*dx,y_0+5*dy,dx,dy,col_menu_back,col_menu_back,col_menu,62,
                    'n','Eingabe einer neuen Tabelle','Neue
Tabelle',new_tab_eingabe)),TRUE);

    sub_menu[2].init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,65,
                    'v','Untermenü zum Verändern der Daten','Verändern...');
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+dy div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,66,
                    'm','Verändern der Messwerte mit der
Maus','Messwerte',change_kurve)),TRUE);
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+(3*dy)div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,0,
                    no_hot_key,'','----------',dummy)),FALSE);
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+(5*dy)div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,67,
                    's','Verändern des Spektrums mit der
Maus','Spektrum',change_spec)),TRUE);
    sub_menu[2].add_button(@sub_menu[1],TRUE);
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+(9*dy)div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,69,
```



```
                    'o','Schieben des Spektrums um Oktave n.oben','Oktave
',octave_up)),TRUE);
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+(11*dy)div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,69,
                    'k','Schieben des Spektrums um Oktave n.unten','oKtave
',octave_down)),TRUE);
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+(13*dy)div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,69,
                    'i','Spektrum wird integriert','Integrieren',integriere)),TRUE);
    sub_menu[2].add_button(New( OBP, Init(x_0+dx,y_0+(15*dy)div
2,dx,dy,col_menu_back,col_menu_back,col_sub_menu,69,
                    'd','Spektrum wird
differenziert','Differenzieren',differenziere)),TRUE);

    menu_list[2].add_button(ts(@sub_menu[2]),TRUE);

    x_0:=x_0+9*char_width+1;
    y_0:=2;
    dx:=9*char_width;
    {---------- Dritte Spalte ----------}
    menu_list[3].init(x_0,y_0,dx,dy,col_stat_back,col_main,col_main,30,
                    'a','Bietet Variationen in der
Darstellung','Anzeige');
    dx:=19*char_width;
    Inc(y_0,dy+4);
    menu_list[3].add_button(New( OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,31,
                    'p','Zeigt Spektrum als Amplituden &
Phasen','Phasenspektrum',set_spec_phase_mode)),TRUE);
    Inc(y_0,dy);
    menu_list[3].add_button(New( OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,32,
                    'k','Zeigt Spektrum in Koeffizienten','Koeff.-
spektrum',set_spec_koeff_mode)),TRUE);
    Inc(y_0,dy);
    menu_list[3].add_button(New( OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,33,
                    'l','Zeigt Messkurve und
Vergrösserung','Lupe',display_only_zoom)),TRUE);
    Inc(y_0,dy);
```

```
      menu_list[3].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                        no_hot_key,'','----------',dummy)),FALSE);
      Inc(y_0,dy);
      menu_list[3].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,34,
                        'd','Gibt Ihnen Daten zu Messung, Rechnung...','Daten-
Info',show_parameter)),TRUE);
      Inc(y_0,dy);
      menu_list[3].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                        no_hot_key,'','----------',dummy))),FALSE);
      Inc(y_0,dy);
      menu_list[3].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,35,
                        'v','Einstellen der Vergrößerung der
Lupe','Vergrößerung',select_zoom_factor))),TRUE);
      Inc(y_0,dy);
      menu_list[3].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,38,
                        'h','Hintergrund im Messfenster
festlegen','Hintergr.setzen',set_higru))),TRUE);
      Inc(y_0,dy);
      menu_list[3].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,39,
                        'a','Hintergrund an- bzw. ausschalten','An/aus
d.Hintergr.',toggle_higru))),TRUE);
      Inc(y_0,dy);
      menu_list[3].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                        no_hot_key,'','----------',dummy))),FALSE);
      Inc(y_0,dy);
      menu_list[3].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,35,
                        't','zeigt Tabelle aller
Spektrumskoeff.','Tabelle',alles_tabelle))),TRUE);


      x_0:=(GetMaxX div 2)-8*char_width-2;
      y_0:=2;
      dx:=8*char_width;
```



```
      {------------------ Vierte Spalte --------------------}
      menu_list[4].init(x_0,y_0,dx,dy,col_stat_back,col_main,col_main,40,
                                    'e','zusätzliche Befehle (Hilfe &
Optionen)','Extras');
      dx:=15*char_width;
      Inc(y_0,dy+4);
      menu_list[4].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,41,
                        'ü','Wie man Hilfe benutzt','Über Hilfe',explain_help)),TRUE);
      Inc(y_0,dy);
      menu_list[4].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,43,
                        'l','Startet das Lernprogramm','Lernen',lernen)),TRUE);
      Inc(y_0,dy);
      menu_list[4].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                        no_hot_key,'','----------',dummy)),FALSE);
      Inc(y_0,dy);
      menu_list[4].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,42,
                        's','Setzt Werte wie beim
Programmanfang','Standardwerte',set_standards)),TRUE);
      Inc(y_0,dy);
      menu_list[4].add_button(New(                                    OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,0,
                        no_hot_key,'','----------',dummy)),FALSE);
      Inc(y_0,dy);
      menu_list[4].add_button(ts(New(                                 OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,36,
                        'e','Umschalten auf Einfaches
Menü','Einfachmenü',switch_menu_off))),TRUE);
      menu_list[4].add_button(ts_inv(New(                             OBP,
Init(x_0,y_0,dx,dy,col_menu_back,col_menu_back,col_menu,37,
                        'm','Umschalten auf ausführliches Menü','volles
Menü',switch_menu_on))),TRUE);


      {---------------- Abschluß ------------------------}
      stat_line:=New(Button_PTR,Init(GetMaxX div 2,2,GetMaxX div 2+2,
char_height+2,col_stat_back,col_stat_back,col_stat_back,
```

```
                                        92,no_hot_key,'Sie berühren hier die
Statusmeldung'));
        stat_line^.invertable:=FALSE;
        info_line:=New(Button_PTR,Init(2,GetMaxY-char_height,GetMaxX-2,char_height
,col_stat_back,col_stat_back,col_stat_back,
                                        93,no_hot_key,'Informationszeile'));
        info_line^.invertable:=FALSE;
        main_menu.init(0,0,0,0,0,0,0,142,no_hot_key,'*** Fehler ***','');
        main_menu.add_button(@menu_list[1],TRUE);
        main_menu.add_button(@menu_list[2],TRUE);
        main_menu.add_button(@menu_list[3],TRUE);
        main_menu.add_button(@menu_list[4],TRUE);
        main_menu.add_button(@mess_window,TRUE);    { die folgenden 3 wurden schon
in f_graph }
        main_menu.add_button(@zoom_window,TRUE);                { initialisiert }
        main_menu.add_button(@spec_window,TRUE);
        main_menu.add_button(@sin_window,TRUE);
        main_menu.add_button(@shift_left,TRUE);
        main_menu.add_button(@shift_right,TRUE);
        main_menu.add_button(@swap_marks,TRUE);
        main_menu.add_button(stat_line,TRUE);
        main_menu.add_button(info_line,TRUE);
        main_menu.set_leave(TRUE);     { da Hauptmenü normalerweise nicht verlassen
                                                wird,      bleibt      es
dargestellt. SubMenüs werden
                                                dagegen gelöscht }
        ignore:=switch_menu_off(0,0,0);
END;

PROCEDURE get_user_name;
VAR
        d, code :WORD;
        ir : input_rec;
        x : EDEXCODE;
        strg : STRING;
        fehler : BOOLEAN;
BEGIN
        IF NOT param_master THEN BEGIN
```



```
                ir.data[1]:='';
                ir.data[2]:='';
                ir.titel:='Fourierprogramm';
                ir.head_line:='Bitte geben Sie folgendes ein:';
                ir.base_line:='Die Abkürzung identifiziert Ihre Dateien!';
                ir.wtd[1]:='Gruppenname (bis 38 Zeichen):';
                ir.dtyp[1]:=ascii;
                ir.dlen[1]:=38;
                ir.wtd[2]:='Abkürzung davon (7-10 Zeichen):';
                ir.dtyp[2]:=ascii;
                ir.dlen[2]:=10;
                ir.double_flag:=FALSE;
                ir.last_edit:=2;
                REPEAT
                        x:=handle_input(ir);
                        d:=10;
                        IF (Length(ir.data[1])<10) OR (Length(ir.data[2])<7) OR (x<>ok)
THEN BEGIN
                                fehler:=TRUE;
                                d:=alert_select('Ihre Eingaben sind zu|knapp gehalten|Bitte
geben Sie sie neu ein','Gelesen');
                                END ELSE fehler:=FALSE;
                UNTIL NOT fehler;
                user_name:=ir.data[1];
                user_id:=ir.data[2];
                IF user_id='Ossau *93*' THEN BEGIN
                        user_id:='Master';
                        param_master:=TRUE;
                END;
        END ELSE BEGIN
                user_name:='**** Systemverwaltung ****';
                user_id:='Master';
        END;
END;

PROCEDURE say_hello;
VAR
```

```
        flag : BOOLEAN;
        tr : text_rec;
        x : INTEGER;
BEGIN
        tr.strg[1]:='Willkommen beim';
        tr.strg[2]:='Fourieranalyseprogramm';
        tr.strg[3]:='';
        tr.strg[4]:='    (c) `93   Christian Bienmüller    ';
        tr.strg[5]:='Zulassungsarbeit 1993';
        tr.strg[6]:='';
        tr.strg[7]:='Wünschen Sie eine Anleitung';
        tr.strg[8]:=' oder ein Lernprogramm?';
        tr.last_string:=8;
        x:=big_alert(tr,' Lernen | Anleitung | Nichts ',FALSE);
        IF x=1 THEN flag:=lernen(0,0,0);
        IF x=2 THEN show_help(90);
END;

PROCEDURE init_global_params;

        FUNCTION test_in_params(strg : STRING):BOOLEAN;
        VAR   d: WORD;
        BEGIN
                test_in_params:=FALSE;
                FOR d:=1 TO ParamCount DO BEGIN
                        IF strg=ParamStr(d) THEN
                                test_in_params:=TRUE;
                END;
        END; { of function test_in_params }

BEGIN         { of init_global_params }
        param_mess:= NOT (test_in_params('nomess') OR test_in_params('NOMESS'));
        param_sound:=          NOT          (test_in_params('nosound')          OR
test_in_params('NOSOUND'));
        param_master:=(test_in_params('master') OR test_in_params('MASTER'));
        param_svga:=(test_in_params('svga') OR test_in_params('SVGA'));
        param_noexit:=(test_in_params('noexit') OR test_in_params('NOEXIT'));
```



```
END;

end.
```

E.15. Die Unit F_MESS

```
unit f_mess;
{$F+} {$O+}
interface
USES f_global, crt;

{  ***************************************************************
  *          M E I L H A U S   E L E C T R O N I C          *
  *           T H E   M E - 4 0   A/D  B O A R D           *
  * Ab hier bis zum nächsten entsprechenden Kommentar wurde alles *
  * vom ME-40.pas übernommen (wie in der Anleitung gefordert)     *
  * entsprechend ist es (c) Meilhaus, wenn auch gestutzt          *
  ***************************************************************  }


  {*****  The following Declaration part and the include file  ****
   *****  ME-40lib.PAS must be included in every user written  ****
   *****  software for the ME-40 card                          ****}

  CONST
        BA   = $300;          {base address of the ME-40 card}
        ME_40_CLK = 2.4576E+6;  {clock frequency of the on board clock}

  VAR   {global variables in the Data segment of TURBO PASCAL}

    chmin,       {first channel to sample}
    chmax,       {last channel to sample}
    NoCh,        {number of channels to sample=chmax-chmin+1}
    Timer_low    {number of ME_40_CLK cycles Timer is low}
    : BYTE;

    BA_2,        {Boardaddress + 2 = A/D result port}
    BA_3         {Boardaddress + 3 = port for D7 synchro bit}
    : INTEGER;

    Ch_s_rate,   {true sampling rate of each channel}
```



```
    AD_s_rate,   {true sampling rate of the A/D=Ch_s_rate*NoCh}
    SC_fmax,     {max. single channel sampling rate}
    MC_fmax      {max multi channel sampling rate}
    : REAL;

        ch :char;    {global character for keyboard input}

{***************************************************************}
{* das war mal:                                          *}
{*    Turbo Pascal Include library :    ME-40lib.pas     *}
{*    for the ME-40 Analog/Digital converter board.      *}
{*    This program has to be compiled with Turbo 4.0     *}
{*    Meilhaus Electronic, Fischerstr. 2                 *}
{*            8039 Puchheim , West Germany               *}
{*            Tel 0049-89-807081                         *}
{*                                                       *}
{-------------------------------------------------------------}
{*   S U M M A R Y   O F   P R O C E D U R E S :    *}

        PROCEDURE beep;
        FUNCTION  ADSample (channel:INTEGER): INTEGER;
        PROCEDURE SCSample (VAR Storage_arry;
                                        Number_of_samples:INTEGER;
                                        Channel:INTEGER);
        PROCEDURE MCSample (VAR Storage_arry;
                                        Number_of_samples:INTEGER;
                                        First_channel:INTEGER;
                                        Last_channel:INTEGER);
        PROCEDURE Trigger(Channel:INTEGER);
        PROCEDURE Int_off;
        PROCEDURE Int_on;
        PROCEDURE Set_AD_s_rate (VAR AD_s_rate:REAL);
        PROCEDURE System_check;
        FUNCTION do_mess( mx,my : LONGINT; nr : WORD): BOOLEAN;
        PROCEDURE init_mess;
        FUNCTION  get_mess_parameters( mx, my : LONGINT; nr : WORD):
BOOLEAN;
```

```
{************************************************************}

implementation
uses gui, gui_tool, f_help, f_graph, f_gr_def, f_datei, f_change;

PROCEDURE beep;

BEGIN
  Sound(500);
  Delay(150);
  Nosound;
END; {beep}

{*****************************************************************}

  FUNCTION ADSAMPLE (channel:INTEGER): INTEGER;

  {this function samples the designated channel and returns the
   result as and INTEGER between 0 and 4095 }

   BEGIN
     inline($BA/$03/$03/$EC/$24/$80/$75/$FB/$8B/$46/$06/$BA/$00/$03
       /$EE/$BA/$03/$03/$EC/$24/$80/$74/$FB/$EC/$24/$80/$75/$FB/$EC
       /$24/$80/$74/$FB/$EC/$24/$80/$75/$FB/$4A/$ED/$25/$FF/$0F/$89
       /$46/$FE);
   END;

{*****************************************************************}



  PROCEDURE SCSample (VAR Storage_arry; Number_of_samples:INTEGER;
                      Channel:INTEGER);

  {this procedure samples the selected channel at the set sampling
```



```
   rate; The result is stored in the indicated storage array which
   must be of TYPE INTEGER; This array may be an local or global
   array of maximum 30000 samples; The variable Number_of_samples
   indicates the number of samples which will be stored in the
   Storage_array; The sampling rate must be selected with the
   procedure Set_AD_s_rate (VAR AD_s_rate:REAL);                }


    BEGIN
      inline($8B/$46/$06/$BA/$00/$03/$EE/$8B/$4E/$08/$C4/$7E/$0A/$55
        /$8B/$EF/$BF/$00/$00/$BA/$03/$03/$EC/$24/$80/$75/$FB/$EC/$24
        /$80/$74/$FB/$EC/$24/$80/$75/$FB/$EC/$24/$80/$74/$FB/$EC/$24
        /$80/$75/$FB/$4A/$ED/$25/$FF/$0F/$26/$89/$03/$83/$C7/$02/$42
        /$E2/$ED/$5D);
    END;


{*****************************************************************}

  PROCEDURE MCSample (VAR Storage_arry; Number_of_samples:INTEGER;
                      First_channel:INTEGER; Last_channel:INTEGER);

  {This procedure samples Number_of_samples/(Last_channel-first_channel+1)
   samples per channel from the designated channels First_channel to
   Last_channel. e.g. Number_of_samples = 9000, First_channel = 3 and
   Last_channel = 5 then 3000 samples are sampled from channel 3,4 and 5
   and stored in the Storage_array of TYPE INTEGER as follows :
   Storage_array [0] = no care,
   Storage_array [1] = no care,
   Storage_array [2] = 1st sample channel 3,
   Storage_array [3] = 1st sample channel 4,
   Storage_array [4] = 1st sample channel 5,
   Storage_array [5] = 2nd sample channel 3,
   Storage_array [6] = 2nd sample channel 4,
   Storage_array [7] = 2nd sample channel 5,
   Storage_array [8] = 3rd sample channel 3,
```

```
  Storage_array [9] = 3rd sample channel 4,
  Storage_array [10] = 3rd sample channel 5,
  .
  .
  Storage_array [9002] = 3000th sample channel 5,

  NOTE that the first two samples of the storage array must be disregarded !

  The maximum size of the  global Storage_array is 30000
  INTEGER samples ; see TYPE declaration in the Main Program !
  The sampling rate must be selected prior to the entry of this procedure
  with the procedure Set_AD_s_rate (VAR AD_s_rate:REAL); Note that the
  sampling rate per channel is the set rate devided by the number of
  channels to sample.                                    }


   BEGIN
     inline($8B/$46/$08/$8B/$5E/$06/$8A/$FB/$8A/$D8/$FE/$C7/$8B/$4E
     /$0A/$83/$C1/$02/$C4/$7E/$0C/$55/$53/$8B/$EF/$BF/$00/$00/$BA
     /$03/$03/$EC/$24/$80/$75/$FB/$EC/$24/$80/$74/$FB/$EC/$24/$80
     /$75/$FB/$EC/$24/$80/$74/$FB/$EC/$24/$80/$75/$FB/$BA/$00/$03
     /$8A/$C3/$EE/$FE/$C3/$3A/$DF/$7C/$02/$5B/$53/$BA/$02/$03/$ED
     /$26/$89/$03/$83/$C7/$02/$42/$E2/$E0/$5B/$5D);
   END;

{*****************************************************************}

  PROCEDURE Trigger(Channel:INTEGER);

  {This procedure samples the designated channel until the input
  of this channel is between 2048 and 2096 and the signal is
  positive going between two samples. If this condition is not
  true for 2000 consecutive samples the procedure terminates auto-
  matically to avoid a hang up of the program.                  }

Begin
```



```
inline($B9/$00/$04/$BA/$00/$03/$8B/$46/$06/$EE/$BA/$03/$03/$EC
 /$24/$80/$75/$F8/$EC/$24/$80/$74/$FB/$EC/$24/$80/$75/$FB/$49
 /$83/$F9/$00/$74/$3E/$BA/$02/$03/$ED/$8B/$D8/$81/$E3/$FF/$0F
 /$81/$FB/$20/$03/$7C/$02/$EB/$D6/$B9/$00/$04/$BA/$03/$03/$EC
 /$24/$80/$75/$FB/$EC/$24/$80/$74/$FB/$EC/$24/$80/$75/$FB/$BA
 /$02/$03/$ED/$25/$FF/$0F/$49/$83/$F9/$00/$74/$0A/$3D/$10/$08
 /$7F/$05/$BA/$03/$03/$EB/$DA);
End;

{*****************************************************************}

PROCEDURE Enable_INT(Int : BYTE);

{ This procedure enables the IBM Interrupt number Int }

  VAR
    imr, mask   : INTEGER;
  BEGIN
    mask       := not ( 1 shl Int );
    imr        := PORT [$21];       { get Interrupt mask register
                                      from 8259 }
    imr        := imr and mask;     { clear mask bit of Int     }
    PORT [$21] := imr;              { and return to controller  }
  END;

{*****************************************************************}

PROCEDURE Disable_INT (Int : BYTE);

{ This procedure  disables the IBM Interrupt number Int }

  VAR
    imr, mask   : INTEGER;
  BEGIN
    mask       :=  ( 1 shl Int );
    imr        := PORT [$21];       { get Interrupt mask register
```

```
                          from 8259 }
    imr       := imr or mask;     { set mask bit of Int       }
    PORT [$21] := imr;            { and return to controller  }
  END;

{*********************************************************************}

Procedure Int_off;     {Switches all Interrupts off which otherwise
                        would interfere with the highspeed Assembler
                        sampling routines !}
 BEGIN
   Disable_INT (0);    {Disable Timer Interrupt}
   Disable_INT (1);    {Disable Keyboard Interrupt}
   Disable_INT (5);    {Disable Interrupt 5 }
 END;

{*********************************************************************}

Procedure Int_on;      {Switches all above interrups on again}
 BEGIN
   Enable_INT (5);     {Enable Interrupt 5}
   Enable_INT (1);     {Disable Keyboard Interrupt}
   Enable_INT (0);     {Enable Timer Interrupt}
 END;


{*********************************************************************}

PROCEDURE Set_AD_s_rate (VAR AD_s_rate:REAL);

{Programs the on-board timer to create a pulse of Timer_low lenghth
 (in micro seconds) , with a repetition frequency of AD_s_rate in Hz,
 the actual sampling rate is returned via the variable AD_s_rate}

 VAR
   Factor : REAL;
```



```
   Count0,Count1,Count2 : INTEGER;

 BEGIN
   IF AD_s_rate < 0.0166 THEN AD_s_rate := 0.0166;  {min. AD_s_rate= }
   Factor := ME_40_CLK/AD_s_rate;                   {1 per minute !  }
   Count0 := Trunc (Factor/32767) + 1;
   IF Count0 <= 1 THEN Count0 := 2;
   IF Count0 > 3000 THEN Count0 := 3000;
   Count1 := ROUND (Factor/Count0);
   AD_s_rate := ME_40_CLK/Count0;
   AD_s_rate := AD_s_rate/Count1;

   PORT [BA+7] := $34;        {Timer 0 in mode 2}
   PORT [BA+4] := LO (Count0);
   PORT [BA+4] := HI (Count0);

   PORT [BA+7] := $74;        {Timer 1 in mode 2}
   PORT [BA+5] := LO (Count1);
   PORT [BA+5] := HI (Count1);

   Count1 := ROUND(Timer_low * (ME_40_CLK/1E6));

   PORT [BA+7] := $B2;     {Timer 2 in mode 1 creating a}
   PORT [BA+6] := LO (Count1); {negative going pulse 30 ME_40_CLK}
   PORT [BA+6] := HI (Count1); {long,ie. 12.2 us }

 END; {Set_AD_s_rate}


PROCEDURE SCSpeedw (VAR sample;Nos:INTEGER);
  BEGIN
    inline($8B/$4E/$06/$C4/$7E/$08/$55/$8B/$EF/$BF/$00/$00/$B3/$80
      /$BA/$03/$03/$EC/$24/$00/$EB/$01/$90/$EC/$24/$00/$75/$FB/$4A
      /$ED/$26/$89/$03/$83/$C7/$02/$42/$E2/$EA/$5D);
  END;
```

```
PROCEDURE SCSpeedo (VAR sample;Nos:INTEGER);
  BEGIN
    inline($8B/$4E/$06/$C4/$7E/$08/$55/$8B/$EF/$BF/$00/$00/$B3/$80
      /$BA/$03/$03/$EC/$24/$00/$75/$FB/$4A/$ED/$26/$89/$03/$83/$C7
      /$02/$42/$E2/$F0/$5D);
  END;

PROCEDURE MCSpeed (VAR sample;NoS:INTEGER);
  BEGIN
    inline($8B/$4E/$06/$C4/$7E/$08/$55/$8B/$EF/$BF/$00/$00/$B3/$00
      /$BA/$03/$03/$EC/$24/$00/$EB/$01/$90/$EC/$24/$00/$75/$FB/$BA
      /$00/$03/$8A/$C3/$EE/$FE/$C3/$3A/$FF/$7F/$02/$5B/$53/$BA/$02
      /$03/$ED/$26/$89/$03/$83/$C7/$02/$42/$E2/$DA/$5D);
  END;


{*********************************************************************}

Procedure System_check;

{This procedure checks the PC/AT processor speed and determines the
 timing parameters of the Timer clock and the maximum sampling rates.
 THIS PROCEDURE MUST BE CALLED AT THE BEGINNING OF EVERY OF YOUR
 SELF WRITTEN ME-40 APPLICATION PROGRAMS !!!!!!!!!!!!!!!!!!!!!!!!!! }


VAR i,x,loop_speed,repeat_speed,MC_loop_speed,Count1 : INTEGER;
    f,f_max : REAL;
        test_samples : ARRAY [0..4000] OF INTEGER;


{---------------------------------------------------------------------}

BEGIN
  BA_2 := BA + 2;
  BA_3 := BA + 3;
```



```
f := 200;            {repetition frequency of test pulse in Hz}

{set timer to create a pulse 2500 us low with a repetition
 frequency of 200 Hz}

Set_AD_s_rate (f);

Count1 := ROUND(2500 * (ME_40_CLK*1E-6));
PORT [BA+7] := $B2;        {Timer 2 in mode 1 creating a}
PORT [BA+6] := LO (Count1); {negative going pulse 2500 us low}
PORT [BA+6] := HI (Count1);

SCSpeedw (Test_samples,4000);

x := 0;
REPEAT       {find sample where timer bit is low}
  x := x+1;
UNTIL ((HI (test_samples [x]) AND $80) < $80) OR (x>3990);
x := x+1;
REPEAT       {find sample where timer bit is high}
  x := x+1;
UNTIL ((HI (test_samples [x]) AND $80) >= $80) OR (x>3990);
x := x+1;
REPEAT       {find sample where timer bit is low}
  x := x+1;
UNTIL ((HI (test_samples [x]) AND $80) < $80) OR (x>3990);

i := x;    {mark sample i}
REPEAT     {find first sample i where timer bit is again high}
  i := i+1;
UNTIL ((HI (test_samples [i]) AND $80) >= $80) OR (i>3990);
IF (i>3990) THEN no_mess:=TRUE
ELSE BEGIN
        loop_speed := ROUND(2500/(i-x));   {loop_speed in micro seconds}
```

```
        SCSpeedo (Test_samples,4000);

        x := 0;
        REPEAT        {find sample where timer bit is low}
              x := x+1;
        UNTIL ((HI (test_samples [x]) AND $80) < $80);
        x := x+1;
        REPEAT        {find sample where timer bit is high}
              x := x+1;
        UNTIL ((HI (test_samples [x]) AND $80) >= $80);
        x := x+1;
        REPEAT        {find sample where timer bit is low}
              x := x+1;
        UNTIL ((HI (test_samples [x]) AND $80) < $80);

        i := x;    {mark sample i; x is first sample where timer bit low}
        REPEAT     {find first sample i where timer bit is again high}
              i := i+1;
        UNTIL ((HI (test_samples [i]) AND $80) >= $80);

        Repeat_speed := ROUND(loop_speed-ROUND(2500/(i-x)));
                                    {repeat_speed in micro seconds}
        SC_fmax := 1/((loop_speed)*1E-6);
        IF (Port[BA_3] AND $40)=0 THEN f_max := 55E3  {D6 of Port BA+3=0 for}
              ELSE f_max := 90E3;                     {the 12.5 us Version, }
        IF SC_fmax > f_max THEN SC_fmax := f_max;     {and 1 for the 5.5 us }

       {Version             }
        Timer_low := ROUND(2*repeat_speed);    {min. low pulse width of
                                                                    the   timer
in us}
        MCSpeed (Test_samples,4000);
        x := 0;
        REPEAT        {find sample where timer bit is low}
              x := x+1;
        UNTIL ((HI (test_samples [x]) AND $80) < $80);
        x := x+1;
```



```
        REPEAT        {find sample where timer bit is high}
              x := x+1;
        UNTIL ((HI (test_samples [x]) AND $80) >= $80);
        x := x+1;
        REPEAT        {find sample where timer bit is low}
              x := x+1;
        UNTIL ((HI (test_samples [x]) AND $80) < $80);

        i := x;    {mark sample i}
        REPEAT     {find first sample i where timer bit is again high}
              i := i+1;
        UNTIL ((HI (test_samples [i]) AND $80) >= $80);
        MC_loop_speed := ROUND(2500/(i-x));   {loop_speed in micro seconds}
        MC_fmax := 1/((MC_loop_speed)*1E-6);
        IF MC_fmax > f_max*0.9 THEN MC_fmax := f_max*0.9;
       {
        PUTS (20,10,'SC_loop_speed = ');Write (loop_speed);
        PUTS (20,12,'Timer_low  = ');Write (Timer_low);
        PUTS (20,14,'Repeat_speed = ');Write (repeat_speed);
        PUTS (20,18,'MC_loop_speed = ');Write (MC_loop_speed);
        PUTS (20,20,'MC_loop_freq = ');Write (MC_fmax:8:1);

        Delay (3000);
       }
  END;
END;  {system_check}

{*********************************************************************}

{ bis hierher wurde übernommen ! Ab jetzt wieder (c) Chr. Bienmüller }
{*********************************************************************}

FUNCTION do_mess( mx,my : LONGINT; nr : WORD): BOOLEAN;
VAR
      t,d : INTEGER;
      length : WORD;
```

```
        bunny : BOOLEAN;
BEGIN
        IF no_mess THEN show_help(95)
        ELSE BEGIN
                Set_AD_s_rate(mess_freq);
                length:=Round((mess_freq*mess_intervall)/1000);
                IF length >= max_datas-5 THEN length:=max_datas-6;
                mess_max:=0;
                mess_min:=0;
                mess_window.disable;
                hide_all;
                show_status('Messung läuft');

                SCSample(work_data^,length,0);  {*** Die Messung ***}

                show_status('Messung beendet');
                FOR d:=0 TO length-1 DO BEGIN
                        t:=work_data^[d];
                        Dec(t,2048);
                        t:=t*4;
                        IF t>mess_max THEN mess_max:=t;
                        IF t<mess_min THEN mess_min:=t;
                        work_data^[d]:=t;
                END;
                used_data:=length;
                IF data_mark[1]>length THEN data_mark[1]:=length-1;
                IF data_mark[2]>length THEN data_mark[2]:=length-1;
                mess_window.set_data(work_data,0,length-1);
        mess_window.set_scale(Round(mess_min*1.1),Round(mess_max*1.1),points);
                mess_window.enable;
        zoom_window.set_scale(Round(mess_min*1.1),Round(mess_max*1.1),points);

        back_zoom_window.set_scale(Round(mess_min*1.1),Round(mess_max*1.1),dotte
d);
                save_messung;
        END;
        bunny:=display_zoom(0,0,0);
```



```
        do_mess:=do_select_work_area(0,0,0);
END;

FUNCTION get_mess_parameters( mx, my : LONGINT; nr : WORD): BOOLEAN;
VAR
        wtd, wtd2 : STRING;
        dauer, freq : LONGINT;
BEGIN
        wtd:='Messfrequenz in Hz';
        wtd2:='Messdauer in ms';
        dauer:=Round(mess_intervall);
        freq:=Round(mess_freq);
        input2_numbers(wtd,wtd2,freq,dauer);
        IF ((dauer*freq) div 1000)>(max_datas-5) THEN BEGIN
                show_help(97);
        END
        ELSE IF ((dauer*freq) div 1000)<300 THEN BEGIN
                show_help(98);
        END
        ELSE BEGIN
                IF freq>sc_fmax THEN freq:=Round(sc_fmax);
                mess_freq:=freq;
                mess_intervall:=dauer;
        END;
        get_mess_parameters:=TRUE;
END;

PROCEDURE init_mess;
VAR
        s : STRING;
        bunny : BOOLEAN;
BEGIN
        s:=ParamStr(1);
        IF (s='nomess') OR (ParamStr(2)='nomess') THEN BEGIN
                no_mess:=TRUE;
        END
```

```
        ELSE BEGIN
                no_mess:=FALSE;
                System_check;                   {this procedure must be called at the
                                                 beginning of every ME-40 program    }
                IF no_mess=TRUE THEN                { keine Karte da oder defekt ...}
                        show_help(96)
                ELSE BEGIN
                        Set_AD_s_rate (SC_fmax);
                        {bunny:=do_mess(0,0,0);} {lädt automatisch}
                END;
        END;
END;
END.
```


E.16. Die Unit F_MOUSE

```
{ f_mouse ist die Unit, die die Mausverwaltung übernimmt }
{$D-}  {nervt beim Debuggen...}
unit f_mouse;
interface
        uses DOS, graph;
        type
                Mouse_Rec = record
                        x,y             : Integer;
                        left,right      : BOOLEAN
        end;

        procedure mouse_reset( auto_mouse_off :Boolean; x_max,y_max : Word);
   procedure show_mouse;
   procedure hide_mouse;
   procedure get_mouse(var mr : mouse_rec);
   procedure set_mouse(x,y :Integer);
   procedure set_mouse_style(nr : Integer);
        procedure set_mouse_activ( activ : BOOLEAN);
        procedure pacman;

implementation
uses gui, CRT, f_help;
type
        mouse_info = Record
        pattern : array[0..300] of Byte;
      mask      : array[0..300] of Byte;
      action_x ,
                action_y : Integer;
        end;
var
        mouse_style     : Integer;
        emulate_mouse : Boolean;
        mouse_visible : Boolean;
        mouse_activ   : BOOLEAN;
        last_m_visi     : Boolean;
```

```
        force_move      : Boolean;
        dummy                   : mouse_rec;
        moldx,moldy   : Word;
        moldmx,moldmy : Word;
   mbackgr              : array[0..300] of Byte;
   mice                 : array[0..2] of mouse_info;
   emulation_magic : Word;
        nothing_happens : LongInt;
        time_waiting : LongInt;
CONST
        boring = 90;  { sekunden bis pacman kommt }

procedure emulate( mr : mouse_rec);
begin
 if ((( last_m_visi <> (mouse_visible AND mouse_activ) ) or force_move or
          (moldmx<>mr.x) or (moldmy<>mr.y) ) and (emulation_magic=$CB12)) then
 begin
        force_move:=False;
        if ( last_m_visi = True ) then begin
        PutImage(moldx, moldy, mbackgr, NormalPut);
        end;
   moldx:=mr.x - mice[mouse_style].action_x;
   moldy:=mr.y - mice[mouse_style].action_y;
        if ( mouse_visible AND mouse_activ ) then begin
                GetImage(moldx, moldy, moldx + 15, moldy + 15, mbackgr);
                PutImage(moldx, moldy, mice[mouse_style].mask, AndPut);
                PutImage(moldx, moldy, mice[mouse_style].pattern, OrPut);
   end;
        last_m_visi := mouse_visible AND mouse_activ;
        moldmx:=mr.x;
        moldmy:=mr.y;
        nothing_happens:=0;
 end
 ELSE
        INC(nothing_happens);
end;
```



```
procedure mouse_reset( auto_mouse_off : Boolean; x_max,y_max : Word);
var
        regs : Registers;
const
        arrow : array[1..7] of PointType =
        ((      x: 1;   y: 1),
                (  x: 1;y:13),
                (       x: 6;   y: 7),
                (       x:12;   y:13),
                (       x:13;   y:12),
                (       x: 7;   y: 6),
                (       x:13;   y: 1));
        arrow_mask : array[1..10] of PointType =
                ((      x: 0;   y: 0),
                (  x: 0;y:15),
                (  x: 2;   y:15),
                (       x: 7;   y:10),
                (       x:12;   y:15),
                (  x:15;   y:15),
                (       x:15;   y:12),
                (       x:10;   y: 7),
                (  x:15;   y: 2),
      (  x:15;   y: 0));
CONST
        no_mouse : STRING = 'Programm ist ohne Maus nicht lauffähig...';

begin
   emulate_mouse := auto_mouse_off;
        last_m_visi := False;
        mouse_activ:=TRUE;
        mouse_visible := False;
        mouse_style:=0;
        regs.ax := 0;
        Intr($33,regs);
        IF (regs.ax <> $FFFF ) THEN BEGIN
```

```
            SetTextJustify(LeftText,TopText);
            OutTextXY(10,10,no_mouse);
            GotoXY(10,100);
            Halt(0);
        END;
        regs.ax:=7;
        regs.cx:=8;
        regs.dx:=x_max;
        Intr($33,regs);
        regs.ax:=8;
        regs.cx:=8;
        regs.dx:=y_max;
        Intr($33,regs);
        if (emulate_mouse=True) then begin
            SetColor(White);
            emulation_magic:=$CB12;
            GetImage(0,0,15,15,mbackgr);
            SetFillStyle(SolidFill,Black);
            Bar(0,0,15,15);
            SetFillStyle(SolidFill,White);
            FillPoly(7,arrow);
            GetImage(0,0,15,15,mice[0].pattern);
            SetFillStyle(SolidFill,White);
            Bar(0,0,15,15);
            SetFillStyle(SolidFill,Black);
            SetColor(Black);
            FillPoly(10,arrow_mask);
            GetImage(0,0,15,15,mice[0].mask);
            mice[0].action_x:=0;
            mice[0].action_y:=0;

        SetFillStyle(SolidFill,White);
        Bar(0,0,15,15);
            SetColor(Black);
            Ellipse(9,8,0,360,6,6);
            Ellipse(10,8,0,360,6,6);
```



```
            Line(9,0,9,5);
            Line(9,11,9,15);
            GetImage(0,0,15,15,mice[1].mask);
            SetFillStyle(SolidFill,Black);
            Bar(0,0,15,15);
            SetColor(White);
            Ellipse(8,8,0,360,6,6);
            Line(8,0,8,6);
            Line(0,8,6,8);
            Line(8,10,8,15);
            Line(10,8,15,8);
            GetImage(0,0,15,15,mice[1].pattern);
            mice[1].action_x:=8;
        mice[1].action_y:=8;

        SetFillStyle(SolidFill,White);
        Bar(0,0,15,15);
            SetColor(Black);
            Ellipse(7,6,0,360,6,6);
            Ellipse(8,6,0,360,6,6);
            GetImage(0,0,15,15,mice[2].mask);
            SetFillStyle(SolidFill,Black);
            Bar(0,0,15,15);
            SetColor(White);
            Ellipse(6,6,0,360,6,6);
            SetLineStyle(SolidLn,0,3);
            Line(11,11,14,14);
            SetLineStyle(SolidLn,0,1);
            GetImage(0,0,15,15,mice[2].pattern);
            mice[2].action_x:=6;
            mice[2].action_y:=6;

        PutImage(0,0,mbackgr,NormalPut);
    end;
end;
```

```
procedure show_mouse;
var
  regs : Registers;
begin
        case emulate_mouse of
                False: begin
                        regs.ax := 1;
                        Intr($33,regs);
      end;
      True: begin
        mouse_visible := True;
        get_mouse(dummy);
      end;
   end;
end;

procedure hide_mouse;
var
        regs : Registers;
begin
        case emulate_mouse of
                False: begin
                        regs.ax := 2;
                        Intr($33,regs);
      end;
      True: begin
        mouse_visible := False;
        get_mouse(dummy);
      end;
   end;
end;

procedure get_mouse;
var
        regs : Registers;
        hour,min,sek,hundreds : WORD;
```



```
        time : LongInt;
begin
        regs.ax := 3;
        Intr($33,regs);
   mr.x := regs.cx;
   mr.y := regs.dx;
        mr.left := (regs.bx and 1) = 1;
        mr.right:= (regs.bx and 2) = 2;
        if (emulate_mouse = True) then emulate(mr);
        if nothing_happens>100 THEN BEGIN
                GetTime(hour,min,sek,hundreds);
                time:=sek+60*(min+60*(hour));
                IF (time<time_waiting) OR (nothing_happens<150) THEN BEGIN
                        time_waiting:=time; {mitternacht...}
                END;
                nothing_happens:=150;
                IF (time>time_waiting+boring) THEN
                        pacman;
        END ;
end;

procedure set_mouse;
var
        regs : Registers;
begin
        regs.ax := 4;
   regs.cx := x;
   regs.dx := y;
        Intr($33,regs);
        get_mouse(dummy);
end;

procedure set_mouse_style(nr : Integer);
begin
        IF mouse_style <> nr THEN BEGIN
                mouse_style:=nr;
```

```
                force_move:=True;
                get_mouse(dummy);
        END;
END;

procedure set_mouse_activ( activ : BOOLEAN);
BEGIN
        mouse_activ:= activ;
END;

procedure pacman;
TYPE
        player = ARRAY[0..700] OF WORD;
        player_array = ARRAY[0..9] OF player;
VAR
        blacky : player;
        packy, back_packy  : ^player_array;
        d,x,y, time,
        p_color, counter    : WORD;
        handle,f : INTEGER;
        mr       : mouse_rec;
        continue : BOOLEAN;
BEGIN
        Randomize;
        handle:=save_big_screen(0,0,GetMaxX,GetMaxY);
        show_info_line('Bildschirmschoner gestartet. Zum Weiterarbeiten Maus
bewegen...');
        hide_mouse;
        IF MaxAvail < (2*SizeOf(player_array)) THEN BEGIN
                show_help(102);
                halt(1);
        END;
        GetMem(packy,SizeOf(player_array));
        GetMem(back_packy,SizeOf(player_array));
        SetFillStyle(SolidFill,Black);
        Bar(0,0,23,22);
        GetImage(0,0,23,22,blacky);
```



```
        SetFillStyle(SolidFill,GetMaxColor);
        Bar(12,0,23,22);
        SetFillStyle(SolidFill,0);
        Bar(0,0,11,22);
        SetColor(0);
        SetLineStyle(SolidLn,0,ThickWidth);
        SetFillStyle(SolidFill,0);
        FOR d:=9 DOWNTO 0 DO BEGIN
                PieSlice(13,11,d*8,360-d*7,10);
                GetImage(0,0,23,22,back_packy^[d]);
        END;

        PutImage(0,0,blacky,NormalPut);
        continue:=TRUE;
        x:=0; y:=0;
        Randomize;
        counter:=0;
        WHILE continue DO BEGIN
                INC(counter);
                REPEAT
                        y:=Random(GetMaxY-10);
                UNTIL (counter>100) OR (y<GetMaxY-30);
                x:=0;
                p_color:=Random(8);
                SetFillStyle(SolidFill,Black);
                Bar(0,y,23,y+22);
                SetColor(p_color);
                SetFillStyle(SolidFill,p_color+8);
                SetLineStyle(SolidLn,0,ThickWidth);
                FOR d:=9 DOWNTO 0 DO BEGIN
                        PieSlice(13,y+11,d*8,360-d*7,10);
                        GetImage(0,y,23,y+22,packy^[d]);
                END;
                SetLineStyle(SolidLn,0,NormWidth);
                WHILE (x<=GetMaxX-23) AND continue DO BEGIN
```

```
                    d:= (x div 2) mod 18;
                    IF d>9 THEN d:=18-d;
                    f:=INTEGER(x)-12;
                    IF f<0 THEN f:=0;
                    PutImage(x,y,back_packy^[d],AndPut);
                    PutImage(x,y,packy^[d],OrPut);
                    time:=0;
                    FOR d:=0 TO 24 DO
                        IF(GetPixel(x+24,y+d)>0) THEN Inc(time,5);
                    Delay(20+time);
                    nothing_happens:=10;
                    get_mouse(mr);
                    INC(x,2);
                    IF (nothing_happens<10) OR mr.left OR KeyPressed THEN
continue:=FALSE;
                END;
                Sound(30);
                PutImage(x-2,y,blacky,NormalPut);
                Delay(10);
        NoSound;
          END;
          dispose(packy);
          dispose(back_packy);
          hide_info_line;
          restore_big_screen(handle);

          show_mouse;
END;

begin
          emulation_magic:=0;
          nothing_happens:=0;
end.
```


E.17. F_SOUND

```
UNIT f_sound;                   { Soundblaster-Unterstützung }
                          { Es steht (bald) zur Verfügung:
                              - Test auf Anwesenheit (init_sound),
                              - Sample-Ausgabe       (put_sound),
                          }
{ S-}{ I-}{ D-}{ L-}
{$F+} {$O-}  { far calls schon (kostet ja nix) aber
                                  wg.  Interrupt  KEINE  Overlayverwaltung
      (sicher=sicher)}

INTERFACE
uses f_global, f_gr_def;
    PROCEDURE init_sound;
    PROCEDURE get_sound(var buffer; acount : WORD);
    PROCEDURE put_sound( buffer : data_array_ptr; account : WORD; freq :
      EXTENDED);
    PROCEDURE set_sound(frequency : WORD);

TYPE
    sound_array = ARRAY[0..64000] OF BYTE;
VAR
    sound_buffer : ^sound_array;

IMPLEMENTATION
USES f_help,CRT,DOS,gui_tool;
CONST
    oldint = 125;
    sample_rate = 11000;
    timer_rate = 2*sample_rate;
VAR
    orig_int_proc : POINTER; {  hier  wird  Adresse  des  alten  Interrupts
      gespeichert }
    old_exit : POINTER;          {   merkt   sich   exitprocedure,   da
      restore_timer eingefügt wird }
    zaehler : WORD;              { merkt sich TimerInterruptaufrufe }
    timer : LONGINT;         {  zählt  Timerticks  für  uns,  wird  per  irq
      geändert }

    sound_magic : LONGINT;
    anschluss : WORD;    { Basisadresse }

PROCEDURE init_blaster;
VAR
    gefunden : BOOLEAN;
    count1, count2 : WORD;
BEGIN
    anschluss:=$210;
    gefunden := FALSE;
    count1:=10;
    WHILE( anschluss<=$260) AND NOT gefunden DO BEGIN
      PORT[anschluss+$6]:=1;
      {gefunden:=FALSE;          }   { Zeit schinden }
      PORT[anschluss+$6]:=0;
      count2:=500;

      WHILE (count2>0) AND (PORT[anschluss+$E]<128) DO
         DEC(count2);

      IF (count2=0) OR (PORT[anschluss+$A]<>$AA) THEN BEGIN
         DEC(count1);
         IF ( count1=0 ) THEN BEGIN
              count1:=10;              { Reset mißlungen }
              anschluss:=anschluss+$10;{ nächste Adresse }
         END;
      END
```

```
      ELSE
         gefunden:=TRUE; { Reset erfolgreich, das wars... }
    END; {while}
    IF not gefunden THEN anschluss:=0; { disable }
END;

PROCEDURE gebe_byte_aus( bt : BYTE);
VAR
    temp :  BYTE;
    adresse : INTEGER;
BEGIN
    adresse:=anschluss+$0C;
{   REPEAT
      temp:=PORT[adresse];
    UNTIL (temp AND 128) = 0;}
    PORT[adresse]:=bt;
END;

PROCEDURE play_buffer;
VAR
    temp : LONGINT;
    repeatn,
    d,f : WORD;
BEGIN
    temp:=timer;
    IF sound_magic=$CB1203 THEN BEGIN
      d:=0;
      WHILE d<SizeOf(sound_buffer^) DO BEGIN
         REPEAT UNTIL temp<>timer;
         temp:=timer;
         gebe_byte_aus($10);
         REPEAT UNTIL temp<>timer;
         temp:=timer;
         gebe_byte_aus(sound_buffer^[d]);
         INC(d);
      END;
    END;
END;

{ die folgenden 3 Routinen sind mehr abgeändert als dem erstaunlich
     schlechten Sybex
    Soundblaster Buch entnommen. Schlecht, da keine Disk beiliegt, die
     Listings
    gerastert hinterlegt sind (gegen Scanner) und die Informationen umso
     spärlicher..}
    { man fühlt sich an die Zeiten erinnert, als der kopierschutz von
     Programmen aufwendiger
    war als die Programme selber }

PROCEDURE handle_timer; Interrupt;
{neue Timer-Interrupt-Routine}
VAR
    regs : REGISTERS;
BEGIN
(*  DEC(zaehler);

    IF zaehler = 0 THEN BEGIN
      Intr(oldint,regs);
      zaehler:=sample_rate DIV 18;     { setze zähler für aufruf d. alten
      Timerinterrupts neu }
    END
    ELSE *)
      PORT[$20]:=$20;             { Interrupt ist verarbeitet }

    INC(timer);
END;

PROCEDURE restore_timer;
```



```
VAR
    oldv : POINTER;
BEGIN
    ExitProc:=old_exit;
    Inline($FA);

    PORT[$43]:=$36;
    PORT[$40]:=0;
    PORT[$40]:=0;

    GetIntVec(oldint,oldv);
    SetIntVec(8,oldv);

    Inline($FB);
END;

PROCEDURE init_timer;
VAR
    izaehler : WORD;
    oldv : POINTER;
BEGIN
    old_exit:=ExitProc;
    ExitProc:=@restore_timer;
    Inline($FA);              { CLI, Interrupts aus }
    izaehler:=1193180 DIV timer_rate; { für 22 kHz }

    PORT[$43]:=$36;
    PORT[$40]:=Lo(izaehler);
    PORT[$40]:=Hi(izaehler);

    GetIntVec(8,oldv);            { bisherigen Int-vektor v. Timer holen
      }
    SetIntVec(oldint,oldv);       { einen neuen interrupt ernennen... }
    SetIntVec(8,@handle_timer);   { neue Routine einbinden }

    Inline($FB);
END;

{ jetzt wieder meine Routinen }

PROCEDURE init_sound;
BEGIN
    IF (param_sound) THEN BEGIN
      IF MaxAvail<SizeOf(sound_array) THEN show_help(146)
      ELSE BEGIN
         init_blaster;
         IF anschluss>0 THEN BEGIN
             sound_magic:=$CB1203;
         END ELSE show_help(147);
      END;
    END;
END;

PROCEDURE get_sound(var buffer; acount : WORD);
BEGIN
END;

PROCEDURE  put_sound(  buffer  :  data_array_ptr;  account:WORD;  freq  :
     EXTENDED);
VAR
    to_repeat    : LONGINT;
    d, f     : WORD;
    max,min  : data_type;
    delta,faktor, value: EXTENDED;
    ems_handle  : INTEGER;
    s1,s2,s3    : STRING;
BEGIN
```

```
    IF  (sound_magic=$CB1203)  AND  (MaxAvail>SizeOf(sound_buffer))  THEN
      BEGIN
      New(sound_buffer);
      s1:='Einen Moment bitte';
      s2:='Die Daten werden umgerechnet';
      s3:='';
      ems_handle:=show_alert(s1,s2,s3);
      max:=0;
      min:=0;
      FOR d:=0 TO account-1 DO BEGIN
         IF (max<buffer^[d]) THEN max:=buffer^[d];
         IF (min>buffer^[d]) THEN min:=buffer^[d];
      END;
      delta:=max-min;
      faktor:=255/delta;
      FOR d:=0 TO SizeOf(sound_buffer^)-1 DO BEGIN
         delta:=d*freq/sample_rate;
         f:=Round(delta) MOD account;
         delta:=delta-Int(delta);
         value:=((buffer^[f]*(1-delta)+buffer^[f+1]*delta)-min)*faktor;
         sound_buffer^[d]:=Round(value);
      END;
      hide_alert(ems_handle);
      init_timer;
      play_buffer;
      restore_timer;
      Dispose(sound_buffer);
    END ELSE show_help(146);
END;

PROCEDURE set_sound(frequency : WORD);
BEGIN
END;

END.
```


E.18. F_TEXT

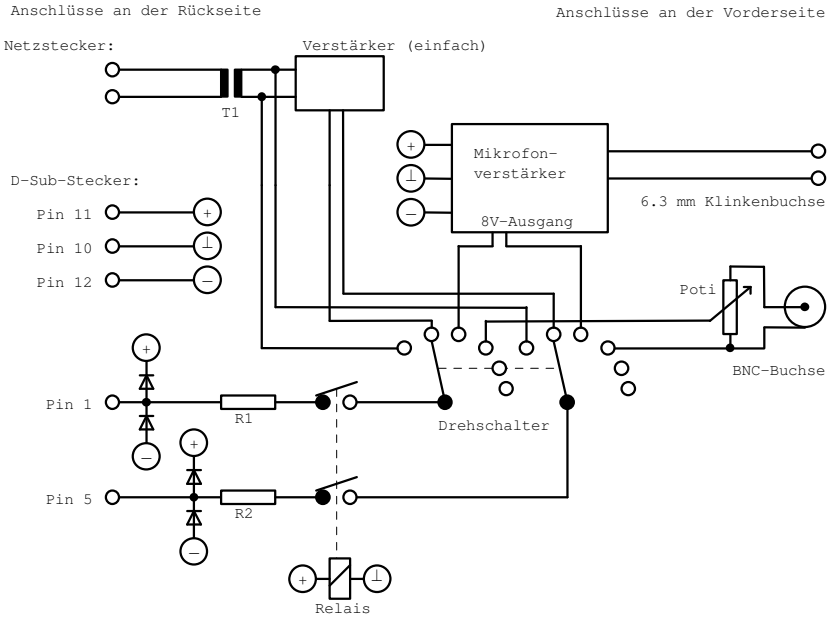
```
unit f_text;

interface
uses graph;
VAR f_text_bold : BOOLEAN; {dient einmaliger Fettschrift}
    EGA_flag : BOOLEAN;
    PROCEDURE f_text_init;
    PROCEDURE f_outtextXY( x, y, horiz, vert :WORD; strg : STRING);
implementation
TYPE
    char_array = ARRAY[0..2047] OF WORD;
VAR
    char_set_ptr : ^char_array;
PROCEDURE f_text_init;
VAR
    f : FILE;
BEGIN
    IF GetMaxY<470 THEN
      EGA_flag:=TRUE
    ELSE BEGIN
      Assign(f,'charset.f16');
      Reset(f,1);
      GetMem(char_set_ptr,4096);
      BlockRead(f,char_set_ptr^,4096);
      Close(f);
      f_text_bold:=FALSE;
      EGA_flag:=FALSE; {ein Glück}
    END;
END;

PROCEDURE f_outtextXY( x, y, horiz, vert :WORD; strg : STRING);
VAR
    d, i, c,dx,f : WORD;
BEGIN
    CASE EGA_flag OF
      FALSE: BEGIN
         IF f_text_bold THEN f:=2
         ELSE                f:=1;
         d:=Length(strg);
         IF horiz=RightText  THEN x:=x-d*8*f;
         IF horiz=CenterText THEN x:=x-d*4*f;
         IF vert =BottomText THEN y:=y-16;
         IF vert =CenterText THEN y:=y-8;
         FOR d:=1 TO WORD(strg[0]) DO BEGIN
              c:=WORD(strg[d]);
              dx:=x+f*8*(d-1);
              IF  (dx<GetMaxX-8)  AND  (y<GetMaxY-15)  THEN  FOR i:=0  TO  7  DO
      BEGIN       SetLineStyle(userbitln,char_set_ptr^[c*8+i],NormWidth);
                  Line(f*i+dx,y,f*i+dx,y+15);
                  IF f=2 THEN Line(f*i+dx+1,y,f*i+dx+1,y+15);
              END;
         END;
         f_text_bold:=FALSE;
      END;
      TRUE: BEGIN
         SetTextJustify(horiz,vert);
         OutTextXY( x, y, strg );
      END;
    END;
    SetLineStyle(SolidLn,0,NormWidth);
END;
END.
```

Anhang F: Schaltplan

Schaltung



Erklärung:

R1, R2 : Widerstände zur Strombegrenzung für die Schutzdioden bzw. Spannungsversorgung, ca. 10kΩ.

Poti: ermöglicht Abschwächen des Eingangs, ca. 10kΩ

Dioden: Schützen den Eingang vor Überspannung, 30V und möglichst schnell.

Relais: Schaltet Eingang erst frei, wenn der Computer Spannung hat (da erst dann die Schutzdioden "wirken"). Spannungswert: 12V.

T1: Netztransformator mit 5V-Sekundär, geringe Leistung, da nur für Meßzwecke.

T2: NF-Übertrager 1:1

Mikrofonverstärker: Eingang für dynamisches 500-Ω-Mikrofon, Ausgang ca. 8V.

Drehschalter: Schaltet zwischen den Eingängen um; sollte mindestens 2*6-polig sein, um Raum für Erweiterungen zu lassen.

Pins 10-12 des D-Sub-Steckers sind direkt die 12V Spannungsversorgung des Computers und werden für das Relais, die Schutzdioden und den Mikrofonverstärker benutzt.

Pin 1 bzw 5 ist der +/- Eingang des Kanals 0 der Meßkarte. Er kann bis +/- 10V messen und ist mindestens bis +/- 20V spannungsfest. Der Eingangswiderstand ist 10 Gigaohm groß.