Proseminar „Mikroprozessoren“

SS 2001

Thema: „Systemsimulation: Beispiel Palm OS Emulator“

Referenten

Marcus Heinzel
Thorsten Juckel

I. Motivation

Motivation dieser Ausarbeitung ist es, dem Leser zum Einen einen Überblick über die Hardwarearchitektur des Palm zu verschaffen und zum Anderen eine Einführung in die Softwareentwicklung auf dem Palm, als Beispiel für eine Entwicklung auf Fremdsystemen, zu geben. Die Softwareentwicklung soll dabei doppelt „motivieren“. Der Leser soll anhand des Palms ein Beispiel für die Entwicklung von Software auf Fremdsystemen, also mit Crosscompilern bekommen. Dies läßt sich gut damit verbinden, dem Leser das Programmieren von einfachen Anwendungen auf dem Palm beizubringen.

Eine kurze Einleitung in die Hardware des Palms soll zeigen, was für ein Prozessor den Kern eines solchen PDAs bildet und welche Komponenten miteinander agieren können um solch ein kleines System so leistungsfähig zu machen, wie es ist. Man wird auch sehen, dass der Palm nur einen Bruchteil der Komponenten nutzt, die der zentrale Chip zur Verfügung stellt und wie er über die Zeit immer leistungsfähiger wurde.

Die Einführung in die Softwareentwicklung gliedert sich in vier Abschnitte. Zum Ersten wird die Architektur des Palm-Betriebssystems dargestellt. Anschließend werden wichtige Eigenschaften des Betriebssytems, deren Verständnis Voraussetzung für das Entwickeln von Software auf dem Palm sind, herausgegriffen und näher erläutert. Darauf aufbauend wird die Entwicklung von Programmen, zunächst allgemein, dann anhand eines Tutorials beschrieben. Zum Schluß wird noch eine Übersicht über Debuggingmöglichkeiten gegeben. Elementare C-Kenntnisse werden bei der Einführung in die Programmierung vorausgesetzt.

II. Hardwarearchitektur

Im Inneren des Palm arbeitet ein Prozessor der Dragonball-Reihe von Motorola. Dieser Chip kann eigentlich viel mehr, als vom Palm genutzt wird. So bietet er zum Beispiel einen Controller für ein LC-Display mit einer Auflösung von 640x480, von dem nur 160x160 vom system genutzt wird. Er bietet außerdem Anschlüsse für diverse Peripherie und hat ein16-Bit PWM Modul für Soundausgabe. Die umfangreichen Optionen für Power Management machen den Dragonball außerdem gut geeignet für batteriebetriebene Geräte.

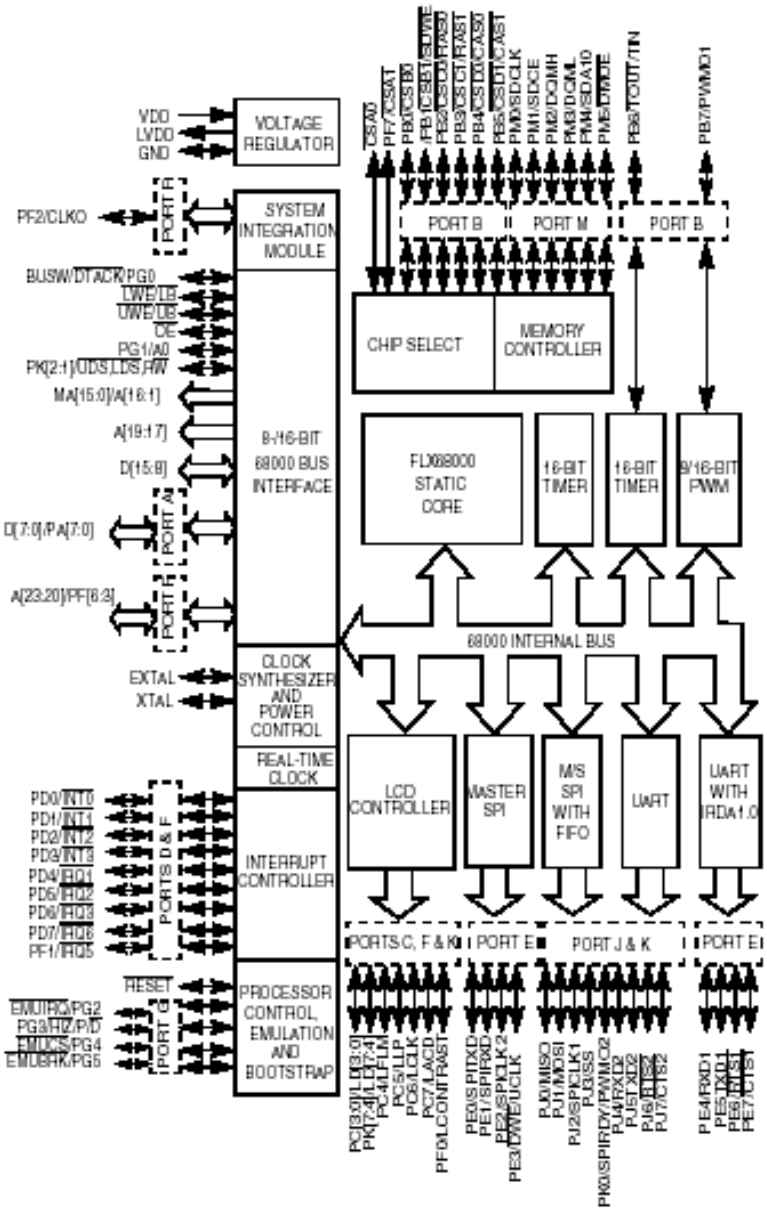


Abbildung 1: MC68VZ328 Block Diagramm

Zentraler Baustein im Dragonball ist ein 68000-Prozessor, der mit 33MHz getaktet 5,4 MIPS leistet. Der interne Adressbus hat 32 Bit, der externe Datenbus besitzt 16 Bit, lässt sich aber auch auf 8 Bit umschalten.

Der Palm verfügt über keinen permanenten Speicher, wie eine Festplatte. Alle Daten werden im RAM abgelegt. Dieser dient einerseits als Datenspeicher, andererseits auch als Hauptspeicher für laufende Programme.

Feature	DragonBall	DragonBall EZ	DragonBall VZ
CPU	68EC000	68EC000	Synthesizable 68000
Chip Selects	16	8	8
LCD Controller	4 gray levels	16 gray levels	16 gray levels
LCD Resolution	Up to 1024 * 512	Up to 640 * 512	Up to 640 * 480
Timer	2 * 16-bit	1 * 16-bit	2 * 16-bit
SPI	Master and Slave	Master	Master and Config. Master/Slave
PWM	16-bit	8-bit with FIFO	16-bit, 8-bit with FIFO
UART	UART 1	UART 1	UART 1 UART 2
RTC	Yes	511 day count	511 day count
PCMCIA 1.0	Yes	No	No
DRAM Controller	No	EDO/Fast Page DRAM	EDO/Fast Page DRAM
GPIO	Up to 78	Up to 54	Up to 76
Boot Strap Mode	No	Yes	Yes
Speed	16 MHz	16/20 MHZ	33 MHz
Voltage	3.3 V ± 10%	3.3 V ± 10%	3.0 V ± 10%
Package	144 TQFP	100 TQFP,144 MAP BGA	144 TQFP,144 MAP BGA

Abbildung 2: Ausstattungsmerkmale der Dragonball-Serie

III. Betriebssystem

Architektur

Über der oben angesprochenen Hardware, sowohl der integrierten Komponenten, als auch zusätzlicher, durch Steckkarten eingebundener Hardware, befindet sich der Hardware Abstraction Layer. Dieser ermöglicht es, dass das Palm OS nicht nur auf einem spezifischen Palm läuft, sondern losgelöst von verschiedenen Hardwarekonfigurationen arbeiten kann. Je nach Hersteller und Modell kann der Palm unterschiedliche Hardwarebausteine haben.

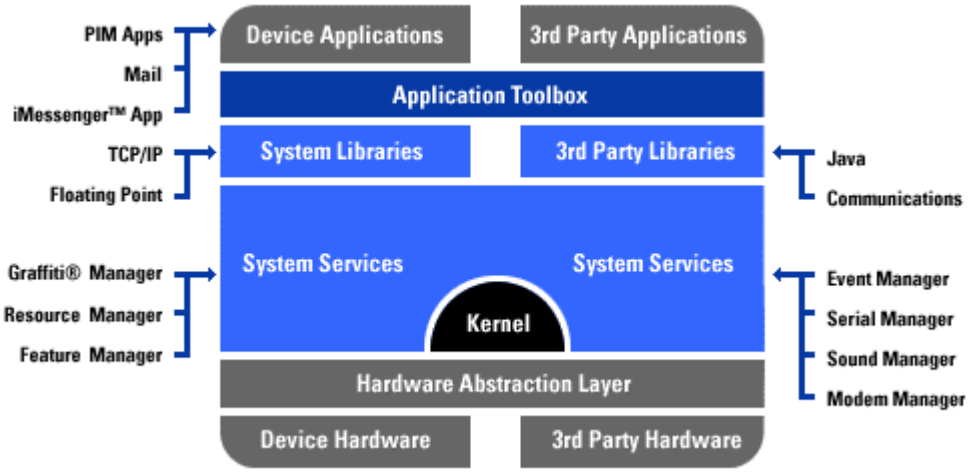


Abbildung 3: Aufbau des Palm OS

Über dem Hardware Abstraction Layer befindet sich der Kernel. Dieser ist das Herz des Betriebsystems und, wie man es vielleicht bei einer so kleinen Maschine nicht erwarten würde, ist multitaskingfähig. Das heißt aber nicht, das man auf dem Palm mehrere Programme gleichzeitig ausführen kann, wie man es auf dem heimischen Computern gewohnt ist. Der Kernel benötigt diese Unterstützung, damit er die komplexe Benutzereingabesteuerung parallel zu den Programmen bearbeiten kann.
Die System Services sind Manager für die einzelnen Hardwarekomponenten des Palm. So steuert der Graffiti Manager zum Beispiel die Eingabe über den Stift, der Sound Manager die Soundausgabe (soweit unterstützt), der Serial Manager überwacht die Schnittstellen des Handhelds. Mehr zu den einzelnen Services später.
Die über den System Services liegenden System Libraries erlauben einfachen Zugriff auf Hardware. Dort befinden sich die Protokolle für die einzelnen Hardwarekomponenten, was dadurch ein komfortables Programmieren ermöglicht. Auf dieser Ebene befinden sich auch Dienste, wie die Java Virtual Machine, sofern sie implementiert ist.
Unter der Application-Toolbox versteht man die vom Betriebssystem zur Verfügung gestellten Benutzerschnittstellen, die zur Interaktion mit dem Palm nötig sind. Dies sind zum Beispiel Schaltflächen, Eingabefelder oder andere Steuerelemente.
Die höchste Stufe der OS-Abstraktionsebenen spricht wohl für sich. Die Device-Applications sind die von Palm mitgelieferten Programme und die 3rd-Party-Applications sind die von Fremdanbietern entwickelten Programme.

Wichtige Systemdienste (System Services)

Scribble Manager
Erwähnenswert ist, daß der Palm im Gegensatz zu anderen Geräten nicht die klassischen Eingabemöglichkeiten wie Tastatur oder Maus besitzt, da dies aus Platzgründen offensichtlich nicht möglich ist. Die einzige Eingabemöglichkeit ist der Scribblestift der eine Interaktion mit dem Gerät durch direktes schreiben auf dem Display ermöglicht. Der Systemdienst „Scribble Manager" ist in der Lage vom Benutzer geschriebene Formen als alphanumerische Zeichen zu interpretieren. Erwähnt wird der Scribble Manager an dieser Stelle nur, damit die Leser, die nicht mit dem Palm vertraut sind, ein Verständnis dafür bekommen, das die Eingabeelemente des Palm-GUI sich ein wenig anders verhalten, als man dies vom PCs gewohnt ist.

Resource Manager
Der Resource Manager ist für die Verwaltung des Dateisystems zuständig. Im Gegensatz zu anderen Betriebssystemen, wie Mac OS, Win9x, usw., die beliebig viele Dateiformate unterstützen, kennt der Palm nur drei spezifische Arten von Dateien:

Palm Resource (PRC): Bei PRCs handelt es sich um ausführbare Programme. Eine PRC-Datei setzt sich aus dem von einem Crosscompiler generierten Binärcode und den Ressourcen für grafische Elemente des GUI zusammen. Alle auf einem Palm befindlichen PRC-Dateien werden durch ein Icon in der Startansicht des Palms symbolisiert und können via Klick ausgeführt werden.

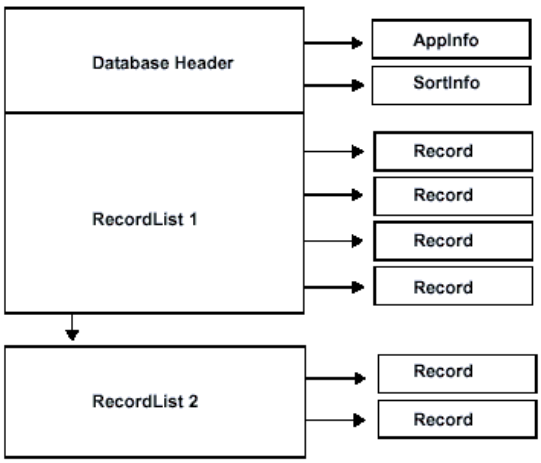
Palm Database (PDC): Bei PDBs handelt es sich um Datenbankdateien. Die einzige Möglichkeit für einen Programmierer Daten aus Programmen (also PRCs) heraus abzulegen, ist es eine PDB mit einem Programm zu verknüpfen. Die Datenbanken sind ähnlich zu handhaben, wie man es von „großen" Datenbanken, die über Query-Sprachen wie z.B. SQL angesprochen werden, kennt. (Das Verknüpfen mit PRCs gestaltet sich jedoch recht umständlich, deswegen wird in dieser Ausarbeitung kein Beispiel für eine PDB gegeben.)

Palm Query Application (PQA): PQAs sind spezielle PDB-Dateien, die HTML-Code enthalten, der PQA-spezifisch angepaßt wurde. PQAs können wie PRCs über ein Icon vom Benutzer aufgerufen werden, um den Webinhalt der PQAs darzustellen.

An dieser Stelle sollte deutlich geworden sein, daß ein Entwickler für das Palm OS bzgl. der Dateiformate nur eingeschränkte Möglichkeiten hat, da man eben keine eigenen Dateiformate entwickeln kann.

Vielleicht eher von technischer Relevanz, aber dennoch erwähnenswert ist, daß sich die drei Dateiformate von der Struktur her gleichen. Allen gleich ist der Header, der Informationen über das Dateiformat enthält. Unterschiede gibt es jedoch bei den Records (Datensätze).

PDAs: Records = Datensätze
PRCs: Records = Ressourcen / Binärcode
PQAs: Records = „HTML"-Seiten



Event Manager
Wie man es auch von anderen GUI-Betriebssystemen her kennt löst das Palm-OS Events als Reaktion auf bestimmte Ereignisse aus. Dies können vom Benutzer generierte Ereignisse sein, wie z.B. das Berühren des Displays mit dem Scribble-Stift oder das betätigen eines GUI-Buttons auf dem Display. Es gibt aber auch diverse vom System erzeugte Ereignisse. Im Grunde sind die Events in drei Arten gegliedert: Es gibt SystemEvents, MenuEvents und ApplicationEvents. Der Vollständigkeit halber seien hier alle aufgezählt:

```
enum events {
nilEvent = 0,           penDownEvent,           penUpEvent,
penMoveEvent,           keyDownEvent,           winEnterEvent,
winExitEvent,           ctlEnterEvent,          ctlExitEvent,
ctlSelectEvent,         ctlRepeatEvent,         lstEnterEvent,
lstSelectEvent,         lstExitEvent,           popSelectEvent,
fldEnterEvent,          fldHeightChangedEvent,  fldChangedEvent,
tblEnterEvent,          tblSelectEvent,         daySelectEvent,
menuEvent,              appStopEvent,           frmLoadEvent,
frmOpenEvent,           frmGotoEvent,           frmUpdateEvent,
frmSaveEvent,           frmCloseEvent,          frmTitleEnterEvent,
frmTitleSelectEvent,    tblExitEvent,           sclEnterEvent,
sclExitEvent,           sclRepeatEvent,         tsmFepModeEvent,
menuCmdBarOpenEvent,    menuOpenEvent,          menuCloseEvent,
frmGadgetEnterEvent,    frmGadgetMiscEvent,     firstINetLibEvent,
firstWebLibEvent,       firstUserEvent
} eventsEnum;
```

Die Events dienen der Steuerung des Programmablaufs und sind somit elementarer Bestandteil aller Palm-Programme. Einstiegspunkt für alle Events ist eine Applikationseventschleife „AppEventLoop“ die von der Main-Routine eines Programms aufgerufen wird und eine Endlosschleife zum Fangen von Events durchläuft, bis das Event zum Beenden der Anwendung empfangen wird.

Was bedeutet es, wenn ein Event „gefangen“ wird? Wenn ein Event ausgelöst wird, dann wird eine Referenz auf eine Datenstruktur vom Typ Event „gefangen“. Die Datenstruktur enthält den Typ des Events, Daten über den Scribble-Stift und eine eventspezifische Struktur, die Daten zum ausgelösten Event enthält:

```
typedef struct {
    eventsEnum eType;
    Boolean penDown;
    UInt8 tapCount;
    Int16 screenX;
    Int16 screenY;
    union{
      ...
    } data;
} EventType;
```

Weitere System Services
Wie bereits erwähnt gibt es noch eine Vielzahl weiterer Systemdienste, deren Verständnis für das Entwickeln einfacher Programme jedoch nicht von Relevanz sind. (z.B. SoundManager, SerialManager (RS-232), ModemManager, MemoryManager, ExchangeManager (PC-Communication), etc.)

IV. Programmierung

Notwendige Tools

Um Programme für den Palm entwickeln zu können, benötigt man den „Ressource Manager“ von Palm, um grafische Benutzeroberflächen zu konstruieren und ein Programmiertool um diese Oberflächen mit Code zu verknüpfen. In unserem Fall haben wir den „Code Warrior“ von Metroworks als Programmiertool verwendet, der sich als GUI-Entwicklungswerkzeug verwenden läßt, ähnlich wie man es von „Forte4Java“ für die Entwicklung von Java Programmen oder „MS Visual Studio“ für die Entwicklung von C / C++ Programmen kennt.
Weiterhin benötigt man den Palm OS-Emulator von Palm, um die geschriebenen Programme auf dem PC simulieren zu können. Um den Palm OS-Emulator betreiben zu können benötigt man ein Betriebssystem-ROM des Palm OS welches man simulieren möchte.

Beschreibung der Tools

Emulator

Um Software auf einem Fremdsystem zu entwickeln braucht man, der Bequemlichkeit wegen, einen Emulator. Emulatoren „gaukeln“ einem Computer ein nicht vorhandenes System vor. Die dabei zugrundelegende Hardware des Fremdsystems wird softwaretechnisch simuliert. Dadurch kann man zwar im Grunde so arbeiten, wie auf dem richtigen System, kann aber hardwarespezifische Vor- oder Nachteilen nicht simulieren.
Der Palm OS Emulator, kurz POSE, bietet alle Eigenschaften, die auch der Palm selbst bringt, kann aber noch viel mehr. Man ist zum Beispiel nicht an die Stifteingabe gebunden, sondern kann auch mit der Computertastatur Texte eingeben, er bietet die Möglichkeit verschiedene Betriebsystemversionen schnell zum Testen umschalten zu können. Dies ist nötig, weil viele ältere Palms, wie zum Beispiel der Palm III, noch mit recht wenig RAM ausgestattet waren, und damit nicht auf aktuellere Betriebsystemversionen geupdated werden können. Programme sollten daher auch auf älteren Versionen getestet werden.

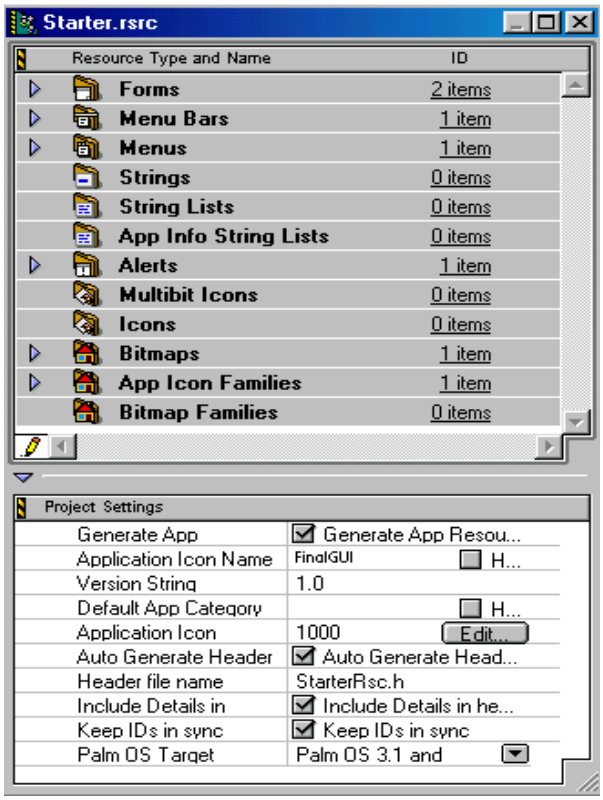
Abbildung 4: Der Palm OS Emulator (POSE)

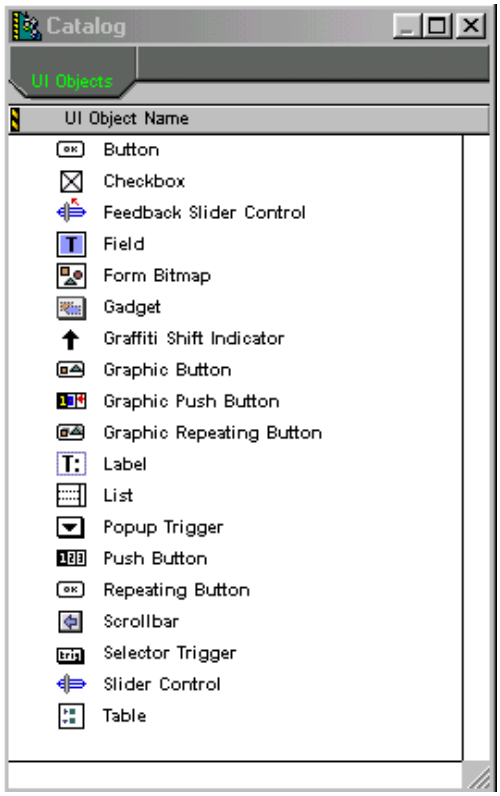
Programme lassen sich im Emulator schnell installieren, und müssen nicht umständlich erst einmal über die serielle Schnittstelle auf dem Palm übertragen werden. Außerdem liefert er umfangreiche Test und Debuggingoptionen, die so auf dem Palm nicht möglich sind.
Ein Besonderes Highlight hierbei sind die sogenannten Gremlins. Hierbei werden schnell hintereinander alle möglichen Eingaben mit dem Stift ausgeführt. Alles erdenkbare wird mit

dem selbstentwickelten Programm ausprobiert, um es so schnell zu testen. Fehler lassen sich so einfach finden. Diese werden auch durch den Emulator gut abgefangen und ausgewertet, ohne dass das ganze System abstürzt und neu gestartet werden muss. Speicherüberläufe und Deadlocks werden vom Emulator erkannt und ausgegeben.
Nachteil des POSE ist die Geschwindigkeit. Nicht, dass er zu langsam läuft, wie man es von Emulatoren erwarten sollte. Durch das Nutzen des PCs ist der Emulator um ein vielfaches schneller als der Palm und so nur bedingt nutzbar. Was hier einwandfrei läuft, ist auf dem echten Gerät eine Zumutung. Ein Testen auf dem Palm bleibt also nie aus.
Ein weiterer Nachteil ist, dass der POSE ohne ein wirkliches Betriebsystem kommt. Dieses muss man sich erst einmal von einem vorhandenen Gerät aus dem Speicher ziehen, oder von Palm besorgen. Die eine Möglichkeit ist Umständlich, die andere ziemlich teuer. Vorteil, wenn man die Versionen von Palm direkt bezieht, ist, dass man immer die neusten Versionen des Palm OS auf dem Computer hat.

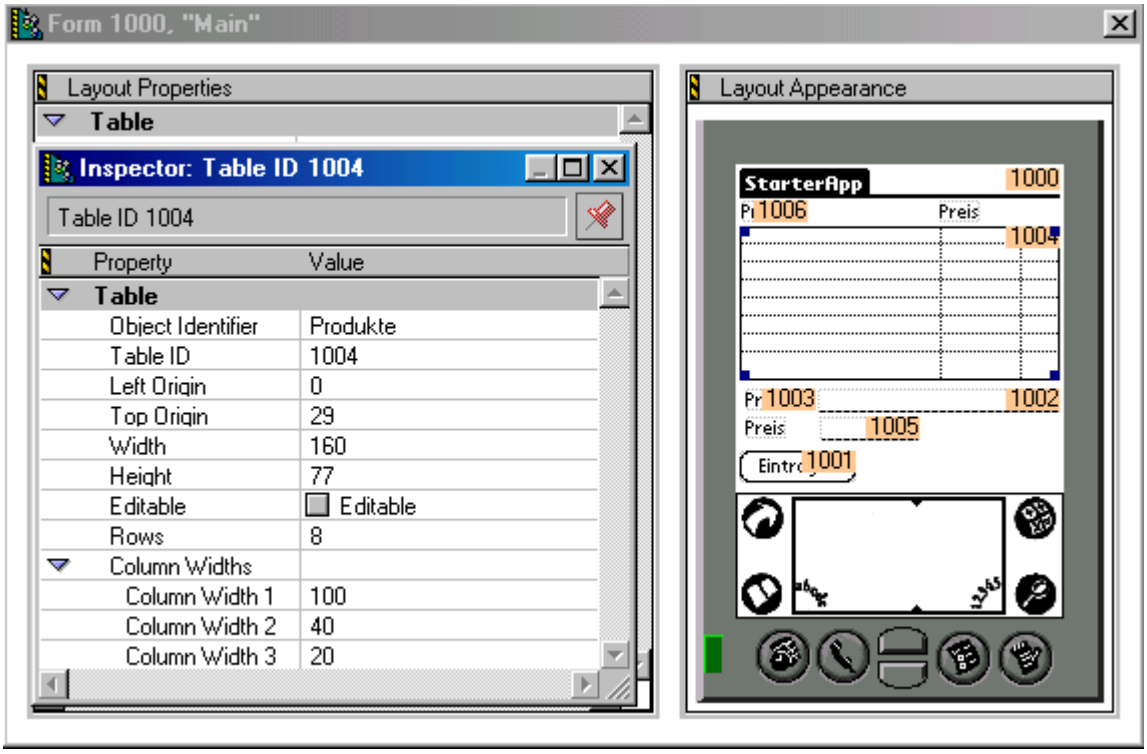
Palm OS Resource Editor
Der Resource Editor dient dem Erstellen von Resourcen für Palm-Anwendungen, also dem Erstellen von grafischen Oberflächen. Der Resource-Editor deckt den Bereich „Application Toolbox“ der Betriebssystemhierarchie ab. Das Resourcen-Konzept des Palm-OS ist allerdings ein klein wenig anders, als das von „größeren“ Betriebssystemen. Es gibt keinerlei dynamische Strukturen. Wenn man zum Beispiel eine Listbox anlegt, dann hat diese nicht beliebig viele Einträge, sondern ist durch eine vom Benutzer vorgegebene Anzahl begrenzt. Eine dynamische Darstellung von Daten läßt sich nur im Zusammenhang mit den Palm-Datenbank-Dateien simulieren. Die Resource-Dateien werden nach dem Compilieren von Programmcode zusammen mit diesem in eine PRC-Datei gelinkt.

In der linken Grafik sind alle möglichen Resourcen-Typen zu sehen und in der rechten sind alle vom Palm unterstützten UI (User Interface) – Objekte zu sehen.





Die UI (User Interface) – Objekt lassen sich auf einer Skizze des Palms zu komplexen Oberflächen anordnen, ähnlich dem Erstellen von Dialogen für andere Betriebssysteme. Die einzelnen Elemente lassen sich detailliert editieren. Exemplarisch durch folgende Grafik dargestellt:



Code Warrior

Um Software für den Palm zu entwickeln gibt es eigentlich eine ganze Reihe von Wegen. Zwei Programme haben sich hierbei aber durchgesetzt. Zum Einen der gcc, als kostenloser Compiler, zum Anderen für Macintosh und PC der CodeWarrior. Die Entwicklungsumgebung unserer Wahl ist hierbei der CodeWarrior von Metroworks.

Der CodeWarrior ist eine graphische IDE mit vielen bereits integrierten Features, direkt auf die Entwicklung von Palm Programmen abgestimmt. Nach der Installation lässt sich in wenigen Schritten das SDK integrieren und Projekte erstellen. Hierbei bietet der CW allen komfort. Projekte werden sofort in eigene Verzeichnisse angelegt, das SDK gibt bereits die anfängliche Programmstruktur vor.

Im CodeWarrior lässt sich komplett POSE integrieren, sodass Programme sofort nach dem kompilieren eigenständig zum Testen auf den Emulator übertragen werden. Außerdem bietet der CodeWarrior umfangreiche Debuggingoptionen, auf die wir nicht im Detail eingehen wollen.

Elementare Programmstruktur

Die elementare Programmstruktur (mal vorausgesetzt, daß wir in C und C++ programmieren) ähnelt der von Programmen für bekannte Betriebssysteme mit grafischer Benutzeroberfläche. Es gibt einen Einstiegspunkt in das Palmprogramm die Funktion **PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)**. Diese Funktion wird vom Betriebssystem aufgerufen, wenn das Programm durch klicken auf das entsprechende Icon in der Palmübersicht für Programme gestartet wird.
Weiterhin gibt es eine Funktion die für das Abhandeln von Events jeglicher Art zuständig ist: **AppEventLoop()** diese Funktion ist eine Endlosschleife, die solange durchlaufen wird, bis ein Event zum Beenden des Programms empfangen wird. Von ihr aus werden Event-Funktionen für das Abhandeln von System-, Menu- und weiteren Applikationsevents aufgerufen.
Die weiteren Applikationsevents sind zum Beispiel die Events die ausgelöst werden, wenn ein Button auf der grafischen Benutzeroberfläche betätigt wird. Für jede grafische Oberfläche, die man mit dem Resource-Editor erstellt hat, kann man eine Callback-Funktion schreiben, die die entsprechenden Events auffängt. Diese Callback-Funktionen werden in der Applikationsevent-funktion registriert. Die Callback-Funktionen funktionieren ähnlich wie ein Thread, nachdem sie dem Betriebssystem angemeldet wurden, nur das sie nicht ständig durchlaufen werden, sondern nur dann, wenn eine Event geschmissen wurde, was von einer Callback-Funktion zu verarbeiten ist.
Über diese Standard-Funktionen hinaus, kann man natürlich beliebige Funktionen für Unterprogrammfunktionalität schreiben. Man kann seine Programme dabei sowohl in C++, als auch in C schreiben. Es ist also objekt-orientiertes Programmieren, wie auch prozedurales Programmieren möglich.

Ressourcen mit Code verknüpfen

Compiliert man ein Palm-Programm, dann werden die Ressourcen mit dem nach dem Compilieren in Binärcode transformierten C-Code zu einer Datei verbunden. Wie läßt man jetzt jedoch den Programmcode mit den Ressource-objekten interagieren? Dies geschieht natürlich über Events. Die Objekte des GUI lösen nach bestimmten Aktionen, die mit diesen durchgeführt werden, bestimmte Events aus. Das Auslösen der Events ist für den Programmierer nicht transparent. Das einzige was der Programmierer sieht, ist eine Event-Struktur, die vom Betriebssystem generiert wird und Informationen über das ausgelöste Event enthält. Der Programmierer kann diese Events mit den bereits Beschriebenen Event-Funktionen auswerten. Die Stuktur, welche Informationen über die Events enthält wurde ja auch bereits beschrieben.

Hierzu ein kleines Beispiel: Angenommen es befindet sich ein Knopf namens „MainEintragenButton“ auf der grafischen Oberfläche, dann kann man das Drücken dieses Knopfes wie folgt auswerten. Zunächst überprüft man den Typ des Events (muß ein ControlSelectEvent sein, dann die ID des Elements, welche man über eine programminterne Namenskonstante referenzieren kann).

```
switch (eventP->eType) {
      case ctlSelectEvent:
      switch (eventP->data.ctlEnter.controlID) {
            case MainEintragenButton:
                  // Hier jetzt der Code der durch Knopfdruck ausgeführt
                  // werden soll
            break;
      }
}
```

V. Tutorial „Einkaufsliste“

1.) Erstellen eines neue Projekts

- Code Warrior öffnen und den Menupunkt *File→New* aufrufen.
- *Palm OS 3.5 Stationary* auswählen, einen Projektnamen und ein Verzeichnis wählen und OK drücken. Palm OS CApp auswählen und OK drücken.
- Der Code Warrior erstellt nun einige Dateien. Unter anderem die bereits beschriebene Elementare Programmstruktur, die sich in der Datei *starter.c* befindet und eine Ressourcedatei *Starter.rsrc*.
- An dieser Stelle sollte man sich ein wenig mit der Entwicklungsumgebung vertraut machen. Anschließend sollte man versuchen mit Hilfe der am Ende dieses Dokuments gemachten Quellenangaben den bereits generierten Programmcode zu verstehen.
- Jetzt das ganze testweise mal compilieren und sich im Emulator anschauen. Das Compilieren führt man mit F7 oder Menupunkt *Project→Make* aus. Im Projektverzeichnis auf der Festplatte befindet sich nun im Unterverzeichnis obj eine Datei *Projektname.prc.* Dies ist die Programmdatei, die auf den Palm übertragen werden kann. Jetzt startet man den Emulator und wählt das ROM mit dem man arbeiten will aus (Notfalls erstmal von einem echten Palm runterladen). Dann mit rechter Maustaste auf den Emulator klicken den Menupunkt *Install Application Database→Other* auswählen und die gerade generierte PRC-Datei suchen. Ab jetzt befindet sich ein neues Icon auf der Palm-Startseite mit dem unser kleines „Programm“ gestartet werden kann.

2.) Ändern von Ressourcen und neues Verlinken

- Im Code Warrior auf die Datei *Starter.rsrc* doppelklicken. Es öffnet sich nun der Resource Manager. An dieser Stelle mit dem Ressource Manager vertraut machen und ein wenig rumspielen.
- Im Fenster *Starter.rsrc* auf den Eintrag *App Icon Families* klicken und dann im Menu *Edit→NewApp Icon Familie Resource auswählen.* Es ist nun ein Eintrag *Application Icon* mit der ID 1000 sichtbar. Die Kategorie Bitmaps markieren und über *Edit→New Bitmap Resource* ein neues Bitmap anlegen. Auf den Eintrag *Application Icon 1000* doppelt klicken und als Bitmap Id die ID des gerade angelegten Bitmaps angeben (Die Farbtiefe „Depth“ muß auch noch auf das benutzte Rom angepaßt werden). Die Änderungen über *File→Save* sichern und in den Code Warrior wechseln und mit F7 Resourcen und Code neu verlinken. Läd man die Anwendung nun neu in den Palm-Emulator, dann hat unsere Anwendung ein neues Icon.
- Wichtig ist, daß man die Änderungen an den Resourcen im Resource-Editor immer sichert, bevor man versucht sie neu zu verlinken.

3.) Eventhandling *(Buttonevent abfangen)*

- Im Fenster *Starter.rscr* den Eintrag *Forms* öffnen und auf *Main* doppelt klicken. Man bekommt das Layout unseres Startdialogs der Anwendung zu sehen.
- Aus dem Fenster Catalog (Wenn nicht sichtbar über *Window→Catalog* anzeigen lassen) einen Button auf den Dialog ziehen. Und folgende Eigenschaften per Doppelklick auf den Button editieren: *Objekt Identifier: „Eintragen“* und *Label: „Eintragen“*. Dann abspeichern.
- In der Funktion *MainFormOpen* folgenden Code an passender Stelle einfügen:

```
case ctlSelectEvent:
  switch (eventP->data.ctlEnter.controlID){
    case MainEintragenButton:
      //Aufrufen des Aboutdialogs
      MenuEraseStatus(0);
      frmP = FrmInitForm (AboutForm);
      FrmDoDialog (frmP);
      FrmDeleteForm (frmP);
      handled = true;
    break;
  }
break;
```

Es wird ein ctlSelectEvent, welches zur ID unsere Buttons paßt, abgefangen. Im Codewarrior kann man anstelle der IDs Bezeichner verwenden, die den IDs in der Datei StarterRsc.h im Unterverzeichnis Rsc des Projektes zugeordnet werden (Daher MainEintragenButton anstelle von 1000). Das ctlSelectEvent wird ausgelöst, wenn der Button Eintragen gedrückt wird. Um dies sichtbar machen, wird der About-Dialog exemplarisch aufgerufen.

- Jetzt das Projekt neu compilieren und das Ergebnis im Emulator begutachten. Wir wissen nun wo und wie wir Events abfangen.

4.) Strukturen referenzieren *(Von Textfeld nach Textfeld kopieren)*

- Vorbemerkung: Wenn man die Änderungen an den Ressourcen im Resource-Editor abspeichert, wird automatisch eine C-Header-Datei namens „StarterRsc.h“ im Unterverzeichnis „Rsc“ des Projektes angelegt. In dieser Datei sind Makros definiert, die die IDs der GUI-Elemente durch sprechende Namen ersetzen, welche dann entsprechend im Programmcode zur Referenzierung der Elemente benutzt werden können.
- In unserem Main-Dialog sollen nun folgende GUI-Elemente angelegt werden:
  - Ein **Field** mit *Objekt Identifier: „Produkt“*
  - Ein **Field** mit *Objekt Identifier: „Speicherfeld“*
  - Ein **Label** mit *Objekt Identifier: „Produkt“* und *Label: „Produkt“*
  - Ein **Label** mit *Objekt Identifier: „Speicherfeld“* und *Label: „Speicherfeld“*
- Das ganze nun sinnvoll anordnen und abspeichern. Den Button der bereits angelegt wurde soll nicht gelöscht werden, da er noch benötigt wird.
- Im CodeWarrior nehmen wir in der Methode in der das ButtonEvent abgefangen wird nun ein paar Änderungen vor. Im Kopf der Prozedur werden folgend Variablen (Zeiger!) angelegt:

```
FieldPtr fieldProdukt;
FieldPtr fieldSpeicherfeld;
FormPtr  frmP;
```

An der Stelle wo das ButtoEvent verarbeitet wird, werden nun folgende Änderungen Vorgenommen (Das aufrufen des Dialoges an der Stelle CASE MainEintragenButton: Wird durch folgenden Code ersetzt):

```
frmP = FrmGetActiveForm();
fieldProdukt  = FrmGetObjectPtr(frmP, FrmGetObjectIndex(
                          frmP,  MainProduktField));
fieldSpeicherfeld = FrmGetObjectPtr(frmP, FrmGetObjectIndex
```

```
                                        (frmP, MainSpeicherfeldField));
    FldInsert(fieldSpeicherfeld, FldGetTextPtr(fieldProdukt),
                    FldGetTextLength(fieldProdukt));
```

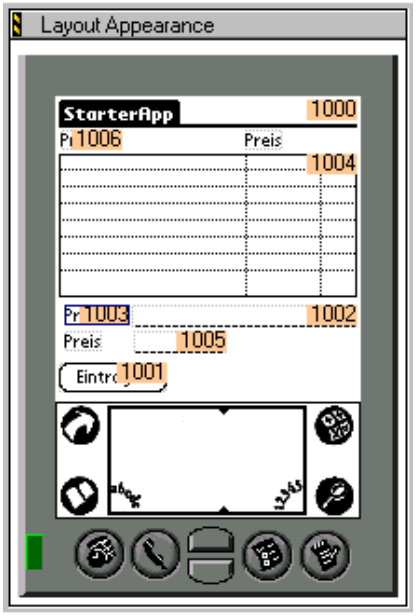
Das Referenzieren eines Objektes geschieht also folgendermaßen: Man legt einen Zeiger auf eine bestimmte Objekt-Datenstruktur an und versucht mit vordefinierten Methoden von

Palm den Zeiger auf ein bestimmtes Objekt zeigen zu lassen. Hat man das gemacht, dann sind die Variablen der Strukturen, die durch den Zeigertyp spezifiziert sind, mit dem refe renzierten Objekt entsprechenden Werten, gefüllt.

- Jetzt das Projekt neu compilieren und sich das Ergebnis im Emulator anschauen.
- Drei elementare Verfahren zum Programmieren sollten nun verstanden sein: Der Umgang mit dem ResourceEditor, das Verknüpfen von Events die von GU-Objekten ausgelöst werden mit Code und das Referenzieren von den Strukturen, die den GUI-Objekten zugrunde gelegt sind.

5.) Komplexe Strukturen *(Erstellen einer Tabelle)*

- Vorbemerkung: Nun folgt der letzt Schritt zur Erstellung der Einkaufsliste, nämlich das Anlegen der Tabelle für die Produkte. Das Label „Speicherfeld“ und das Field „Speicher-feld“ wieder entfernen.
- **Table** auf dem Main Dialog erstellen.
  ObjectIdentifier: „Produkte“
  Rows: 8
  Columns Widths: Auf 3 Spalten über das Menu Edit→NewColumnWidth erweitern mit folgenden Abmessungen: ColumnWidth1 = 100 ColumnWidth2 = 40 und ColumnWidth3 = 20
- Ein **Field** mit *Objekt Identifier: „Preis“ erstellen*
- Ein **Label** mit *Objekt Identifier: „Preis“* und *Label: „Preis“erstellen*
- Der MainDialog sollte nun in etwa wie folgt aussehen:



- Im Kopf der Methode „MainFormHandleEvent“ werden nun folgend Variablen ergänzt:

```
TablePtr myTableRef;
FieldPtr fieldPtr1;
FieldPtr fieldPtr2;
int      t;
```

- Das Event „frmOpenEvent“ der Methode „MainFormHandleEvent“ wird durch folgenden Code, zur Formatierung der Tabelle bei Programmstart, ergänzt:

```
myTableRef = (TableType*) (FrmGetObjectPtr (frmP,
         FrmGetObjectIndex (frmP, MainProdukteTable)));

for (t = 0; t < 10; t++) {
  TblSetItemStyle (myTableRef, t, 0, textTableItem);
  TblSetItemStyle (myTableRef, t, 1, textTableItem);
  TblSetItemStyle (myTableRef, t, 2, checkboxTableItem);
  TblSetRowUsable (myTableRef, t, false);
}

TblSetColumnUsable (myTableRef, 0, true);
TblSetColumnUsable (myTableRef, 1, true);
TblSetColumnUsable (myTableRef, 2, true);

TblDrawTable(myTableRef);
```

- Der Code der durch Betätigen des Button ausgelöst wird, wird durch folgenden Code, zum Eintragen eines Datensatzes in die Tabelle, ersetzt:

```
frmP = FrmGetActiveForm();
myTableRef = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP,
             MainProdukteTable));

TblSetRowUsable(myTableRef, CurrentColumn, true);

FieldPtr1 = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP,
            MainProduktField));

FrmSetFocus(frmP, FrmGetObjectIndex(frmP, MainProdukteTable));
TblGrabFocus(myTableRef, CurrentColumn, 0);
FieldPtr2 = TblGetCurrentField(myTableRef);
FldInsert(fieldPtr2, FldGetTextPtr(fieldPtr1),
          FldGetTextLength(fieldPtr1));

FrmSetFocus(frmP, FrmGetObjectIndex(frmP, MainProduktField));
FldGrabFocus(fieldPtr1);

FieldPtr1 = FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP,
            MainPreisField));
FrmSetFocus(frmP, FrmGetObjectIndex(frmP, MainProdukteTable));
TblGrabFocus(myTableRef, CurrentColumn, 1);
fieldPtr2 = TblGetCurrentField(myTableRef);
FldInsert(fieldPtr2, FldGetTextPtr(fieldPtr1),
          FldGetTextLength(fieldPtr1));
CurrentColumn++;
```

- Im Header der C-Datei folgende globale Variable anlegen (und in der Methode „Pilot-Main“ mit CurrentColumn = 0 initialisieren):

```
int CurrentColumn;
```

- Projekt compilieren, im Emulator bestaunen, auf den echten Palm überspielen und im Kaufhaus anwenden ☺....
- **Schlußbemerkung zum Programmieren:** Wer auch wissen will, wie man mit Datenbanken, etc., umgeht, sollte das wirklich allumfassende Tutorial von Metroworks ausprobieren, das bereits mit der Demoversion des CodeWarriors im Pdf-Format mitgeliefert wird. Es befindet sich in folgendem Verzeichnis: **...Palm\Code Warrior\Documentation\Palm OS Documentation\PDF\Palm OS Tutorial Windows.Pdf**

6.) Einsatz von Gremlins

- Um jetzt unser Programm zu testen können wir die „Gremlins“ des Palm Emulators einsetzen. Exemplarisch stellen wir die Anzahl der Testdurchläufe auf 100 und starten den Gremlinemulator. Wir erkennen, dass unser Programm augenscheinlich einwandfrei läuft. Es werden keine Fehler entdeckt.
- Jetzt erhöhen wir die Testdurchläufe auf 1000 und sind ganz überrascht, dass sich wohl ein kleiner Fehler in das Programm geschlichen hat. Was ist passiert?
- Wir wissen: Dadurch, dass wir nur eine statische Tabelle und keine Datenbank für unsere Einkaufsliste verwendet haben, kommt es zu einem Speicherüberlauf. Dieses hätten wir als Programmierer wohl so erkannt und deshalb beim Testen nicht mehr als die möglichen acht Einträge vorgenommen. Die Gremlins wissen davon aber nichts und testen deshalb alles.
- Falls der Fehler aber nicht so offensichtlich ist, wie in unserem Programm, dann muss man wohl oder übel auf die Debuggingfunktionen des Palm zurückgreifen.

7.) Debugging

- Wie schon erwähnt besitzt unser Programm über keinerlei dynamischer Datenstrukturen (das wäre auch im Rahmen des Projekts zu aufwändig gewesen). Wir haben also nur statische Datenstrukturen verwendet um so das Tutorial so einfach wie möglich zu halten.
- Wenn beim Testen eines Programmes ein Fehler auftritt und man ihn nicht sofort entlarven kann, muss man das Programm debuggen. Der Code Warrior bietet durch die Integration des Palms und des Emulators das Programm direkt auf dem Gerät zu debuggen.
- Dabei biete das Programm alle Möglichkeiten eines guten Debuggers. Man kann Breakpoints setzen, schnell in Funktionen reinspringen, usw.
- Debugging auf dem Palm ist somit nicht anders als bei jedem anderen System. Einziger Unterschied ist, dass man sich auf dem PC mit dem Assembler des 68000-Prozessors „herumschlagen“ muss. Es lohnt sich also nicht näher auf das Debugging einzugehen.

VI. Quellenangaben

Iternetquellen

- Hardware
  - www.Palm.com / www.PalmOS.com / www.3com.com

- Programmieren
  - www.palmos.com/dev/tech/docs/devguide/TableOfContents.htm
  - www.pdaforum.de
  - frankscaslpage.home.att.net

- Datenblätter Dragonball
  - www.motorola.com/SPS/WIRELESS/pda

Hilfreiche Dokumente

- Palm OS Reference.pdf
- Palm OS Companion.pdf
- Constructor for Palm OS.pdf

Die Pdf-Dateien erhält man entweder auf der Seite www.palmos.com/dev/tech/docs/devguide/TableOfContents.htm oder indem man sich die Demoversion des Code Warriors herunterlädt. Evaluiert man die Demoversion des Code Warriors, dann ist außerdem das mitgelieferte Dokument „Palm OS Tutorial Windows.pdf“ sehr lesenswert, da es gut zum Einsteigen in die Programmierung mit dem Code Warrior und dem Verstehen elementarer Konzepte geeignet ist.