Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Prozessinformatik

Betreuer: Herr Detlef Streitferdt

Hauptseminar
Wintersemester 2003/2004
zum Thema

Grundlagen der Palm-Programmierung

Bearbeiter: Daniela Vock
Termin: 15.01.2004

Inhaltsverzeichnis

1. Einleitung

Motivation und Zielstellung der Arbeit

In der heutigen Zeit gewinnt der Palm immer mehr an Bedeutung. Nicht nur im privaten Bereich sondern gerade auch im Business-Bereich kann er dem Anwender nützliche Dienste erweisen. Aufgrund seiner geringen Größe kann man ihn jederzeit überall mit hinnehmen. Ein weiterer Vorteil des Palms ist die Tatsache, dass er mit Batterie beziehungsweise Akku läuft und somit auch mehr Standortflexibilität gewährleistet, da kein unmittelbarer Stromanschluss benötigt wird.
Da in den letzten Jahren auch viele Funktionen des Palms weiterentwickelt wurden, steht er in diesem Punkt einem herkömmlichen PC kaum noch nach.
So verfügt man zum Beispiel auch auf dem Palm über Anwendungen wie Adressverwaltung, Terminplanung, To-Do-Listen, Verwaltung von persönlichen und geschäftlichen Daten, Grafikprogramme, Kalender, Taschenrechner, Textverarbeitung und anderen.
Ein erstes Ziel dieser Arbeit soll deshalb sein, kurz die Unterschiede zwischen einem herkömmlichen PC und einem Palm zu erläutern.
Dank der heutigen Entwicklung besteht mit den aktuellen Palms sogar die Möglichkeit, via Modem oder auch Wireless Lan E-Mails zu verschicken und den Palm von einem beliebigen Standort aus zu synchronisieren.
Durch die Möglichkeiten der Synchronisation kann man auch jederzeit, den aktuellen Stand zum Beispiel von Adressdaten, Terminkalendern oder E-Mails von seinem PC auf den Palm übertragen. Da die Synchronisation bei der alltäglichen Arbeit mit dem Palm eine wichtige Rolle spielt, ist ein zweites Ziel der Arbeit, die Synchronisation genauer zu erläutern. Dabei soll auch auf die Rolle der Conduits näher eingegangen werden.
Wem die herkömmlichen Anwendungen für den Palm nicht ausreichen, für den besteht jederzeit die Möglichkeit, sich neue passende Anwendungen zu programmieren beziehungsweise sich diese programmieren zu lassen. Um ein Grundverständnis für die Programmierung von Anwendungen für den Palm zu erlangen, besteht ein drittes und letztes Ziel dieser Arbeit darin, etwas zu den Themen Emulatoren, Code Warrior, Palm Application Programming Interface (Palm-API) und PRC-Tools zu sagen.

Aufbau der Arbeit

Zunächst soll zu Beginn der Arbeit erst einmal eine Gegenüberstellung von Palm (in der Arbeit auch als Handheld bezeichnet) und herkömmlichen PC (in der Arbeit auch als Desktop bezeichnet) erfolgen.
Da die Synchronisation eine wichtige Voraussetzung für die Palm-Programmierung darstellt, wird im Anschluss an die Gegenüberstellung von PC und Palm etwas ausführlicher auf das Thema Synchronisation eingegangen, sowie auf das Thema Conduits, das in engem Zusammenhang mit dem Thema Synchronisation steht.
Eine weitere wesentliche Voraussetzung für die Palm-Programmierung ist ein funktionierender Emulator, der es ermöglicht, am PC zu testen, ob die neu entwickelten Programme auch wirklich unter Palm-Bedingungen funktionieren. Deshalb soll die Arbeit dem Leser auch einen kurzen Überblick bezüglich der Alternativen bei Entwicklungsumgebungen und der Thematik Emulator vermitteln.
Im Anschluss daran sollen anhand eines einfachen „HelloWorld“- Programms die Grundlagen der Palm-Programmierung und somit auch ein Teil der Palm-API vermittelt werden. Zum Abschluss der Arbeit wird es noch eine kurze Zusammenfassung mit einem Fazit geben.

2. Grundlagen der Palm-Programmierung

2.1. allgemeine Grundlagen

Palm versus PC

Wie bereits im Einleitungskapitel erwähnt, besitzt der Palm viele ähnliche Funktionen, die auch ein PC aufweist. Dennoch gibt es auch einige Eigenschaften, wo sich beide klar voneinander unterscheiden. In Anlehnung an einer Tabelle von *Maxwell* [MAXW 1999, S. 6] soll deshalb jetzt eine tabellarische Gegenüberstellung von Palm- und PC-Eigenschaften erfolgen:

Palm-Eigenschaften	PC-Eigenschaften
geringe Prozessorleistung	ausreichend große Prozessorleistung
sehr wenig Speicher	ausreichend großer Speicher
kleines Display	großer Bildschirm
nur einzelnes Fenster sichtbar	mehrere Fenster möglich
spezieller Einsatzzweck	universell einsetzbar
Benutzer kann sich überall befinden	Benutzer sitzt gewöhnlich am Schreibtisch
einfache Bedienelemente für den Nutzer	komplexe Bedienelemente für den Nutzer
Dateneingabe erfolgt mittels der Schrifterkennungssoftware Graffiti	Dateneingabe erfolgt mittels Tastatur
Batteriebetrieb	Netzbetrieb

Abb. 2.1/1 „Übersicht Palm-Eigenschaften versus PC-Eigenschaften“

Ein wesentlicher Punkt, in dem sich Palm und PC unterscheiden, ist der Speicher.
Prinzipiell gibt es für den PC zwei verschiedene Arten des Speichers – den langsamen nichtflüchtigen Speicher (zum Beispiel Datenträger wie Festplatte oder CD) und den schnellen flüchtigen Speicher – nämlich den RAM. Das hat zur Folge, dass der Computer relativ viel Zeit benötigt, um Daten beispielsweise von Festplatte ins RAM zu bringen.
Beim Palm hingegen gibt es nur einen Speichertyp – den schnellen permanenten Speicher im RAM. Das führt zum Beispiel auch zu Unterschieden bei der Programmierung für Palm-Geräte und bei der Programmierung für Handhelds, die das Betriebssystem Windows CE verwenden. Unter Windows CE nämlich wird der vorhandene Speicher auf klassische Weise in Haupt- und Festplattenspeicher unterteilt, was zur Folge hat, dass Daten und Programme zwischen Haupt- und Festplattenspeicher hin und her kopiert werden müssen. Dies wiederum führt zu einem Anstieg im Ressourcen-Bedarf.
Eine weitere Besonderheit des Palms ist, dass er kein Dateisystem im Sinne des Windows-Dateisystems besitzt, stattdessen ist der Speicher in einer Datenbank organisiert. Sogar die Anwendungen für Palm OS werden auf diese Weise gespeichert. Die Datenbanken für Anwendungen beinhalten verschiedene Datensätze. Zum einen gibt es Datensätze, die ausführbaren Code enthalten und zum anderen existieren Datensätze, die compilierte Ressourcen des Ressourceneditors enthalten. Wenn man etwas innerhalb des Datensatzes ändern möchte, erfolgt ein Betriebssystemaufruf, über den der betreffende Datenbankeintrag gesperrt wird. Dadurch ist es anderen Programmen nicht mehr möglich, auf diesen Datenbankeintrag zuzugreifen. Durch die Sperrung wird dem Programm, das Änderungen vornehmen möchte, ein Zeiger übergeben und es kann nun die Änderungen vornehmen. Nachdem die Änderungen abgeschlossen sind, wird der betreffende Datenbankeintrag durch das Programm wieder freigegeben.

Der Palm hat ein eigenes Betriebssystem. Dieses Betriebssystem heißt Palm OS. [Anmerkung: Wenn im weiteren Verlauf von Handhelds die Rede ist, dann wird davon ausgegangen, dass diese als Betriebssystem Palm OS und nicht Windows CE benutzen.]
Die wichtigsten Palm-Dateiformate sind *.prc, *.scp, *.pdb, *.pnc, *.pqa.
Wie im Einleitungskapitel bereits erwähnt, stehen einem auf dem Palm ähnliche Anwendungen, wie es sie auch für den PC gibt, zur Verfügung. Solche sind zum Beispiel Adress- und Aufgabenverwaltungen, Kalender, Taschenrechner, Spiele und so weiter.
Ein weiteres Merkmal des Palms ist, dass alle Displayanzeigen über Forms erfolgen. Diese sind ähnlich den Fenstern unter Windows. Allerdings kann auf dem Palm im Gegensatz zum PC immer nur ein Form angezeigt werden.

Laut der Dokumentation „Introduction to Conduits“ von Palm [IntroToConduits 2000, S. 4] gibt es sechs Software-Komponenten, die bei der Benutzung eines Handhelds zum Einsatz kommen können. Diese heißen desktop applications, Palm OS applications, HotSync Manager, HotSync Client, Conduits, Sync Manager API und Notifiers.
Die desktop applications laufen auf dem Desktop und arbeiten mit Daten, die zum Handheld gesendet beziehungsweise von diesem empfangen werden.
Die Palm OS applications werden entwickelt, um auf palmunterstützenden Handhelds zu laufen. Sie werden auch handheld applications genannt.
Eine weitere Komponente ist der HotSync-Manager. Er läuft auf dem Desktop und kommuniziert mit dem Handheld. Dafür wird der HotSync-Button an der Dockingstation gedrückt. Danach ruft der HotSync-Manager jedes der installierten und auf dem Desktop konfigurierten Conduits auf. Ein HotSync wird nie durch den Desktop ausgelöst, sondern es ist immer ein Signal vom Handheld notwendig. [Anmerkung: Auf die genaue Funktionalität von HotSync-Manager und HotSyncs wird an späterer Stelle in dieser Arbeit noch einmal eingegangen.]
Eine vierte Softwarekomponente ist der HotSync Client. Sobald der HotSync-Button gedrückt wird, startet der HotSync Client auf dem Handheld. Diese Handheld-Anwendung startet den HotSync-Manager und antwortet den Datenbankanfragen des Sync-Managers.
Conduits sind Programme, die der Interaktion des HotSync-Managers mit den spezifischen Daten des Handhelds dienen. Sie können mit dem Conduit Development Kit (CDK) erstellt werden. Im Allgemeinen wird beim Erstellen einer Handheld-Anwendung auch ein Conduit zum Synchronisieren ihrer Daten mit den Daten des Desktops geschrieben.
Die Aufgabe der Sync Manager API ist es, eine Programmschnittstelle anzubieten, die von Conduits zur Kommunikation mit dem Handheld genutzt wird. Diese Kommunikations-API erlaubt es dem Conduit, unabhängig von der Verbindungsart zwischen Handheld und Desktop zu bleiben.
Die sechste Softwarekomponente bilden die Notifiers. Damit die Desktopprogramme wissen, dass der HotSync-Manager aktiv ist, werden diese Notifiers durch den HotSync-Manager aufgerufen. Ein Notifier ist eine Windows- Dynamic Link Library (DLL), deren Zweck es ist, der Desktopanwendung mitzuteilen, dass das Conduit die Desktopdaten verändert. Durch dieses Verhalten wird gewährleistet, dass die Anwendung und ihr zugehöriges Conduit nicht zur selben Zeit Daten ändern.

In der nachfolgenden Abbildung (s. Abb. 2.1/2) werden die sechs Softwarekomponenten noch einmal grafisch dargestellt:

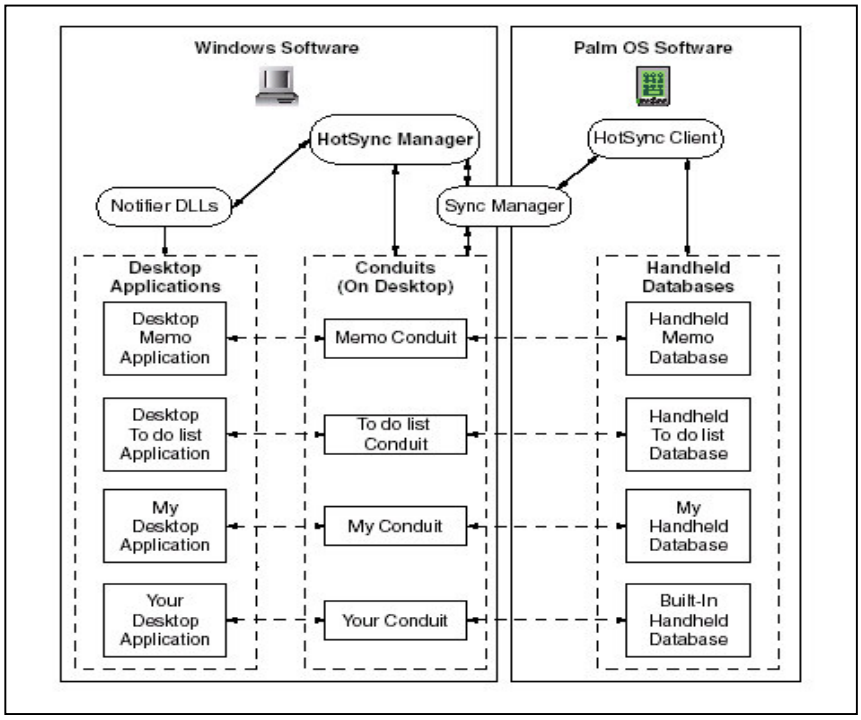

Abb. 2.1/2 „Palm OS platform process flow“ [IntroToConduits 2000, S. 5]

Um Daten vom PC zum Palm zu übertragen (was für die Programmierung relevant ist) und umgekehrt, ist es notwendig, eine Synchronisation zwischen PC und Palm durchzuführen. Deshalb beschäftigt sich der nächste Abschnitt etwas ausführlicher mit der Synchronisation.

Synchronisation und Conduits

Grundvoraussetzung für eine Synchronisation ist zunächst erst einmal, dass eine Verbindung zwischen dem Handheld und dem Desktop erfolgt. Diese Verbindung kann auf verschiedenen Wegen erfolgen. So gibt es zum Beispiel die Möglichkeit einer seriellen Verbindung oder einer Verbindung mittels USB-Kabel. Besitzt der an der Synchronisation beteiligte Desktop eine Infrared Data Association Schnittstelle (IrDA-Schnittstelle), dann besteht auch die Möglichkeit einer Synchronisation über Infrarot. Im Zeitalter des Internets gibt es natürlich auch die Synchronisationsmöglichkeit via Modem. Dazu wird allerdings als spezielles Zubehör ein GSM-Modem benötigt. Des Weiteren hat man noch die Möglichkeit einer Synchronisation über Netzwerk.
Neben dem Herstellen einer Verbindung zwischen Palm und Desktop, ist für die Synchronisation auch wesentlich, dass sich sowohl auf dem Palm als auch auf dem Desktop Synchronisationssoftware befindet. Die Synchronisationssoftware für den Palm ist meist schon auf dem Handheld vorhanden, jedoch die Synchronisationssoftware für den Desktop muss noch auf dem PC installiert werden. Für den Desktop gibt es verschiedene Synchronisationstools. So kann man zum Beispiel unter Windows die Software Palm Desktop (s. Abb. 2.1/3) verwenden oder unter Linux pilot-link, KPilot oder JPilot installieren.

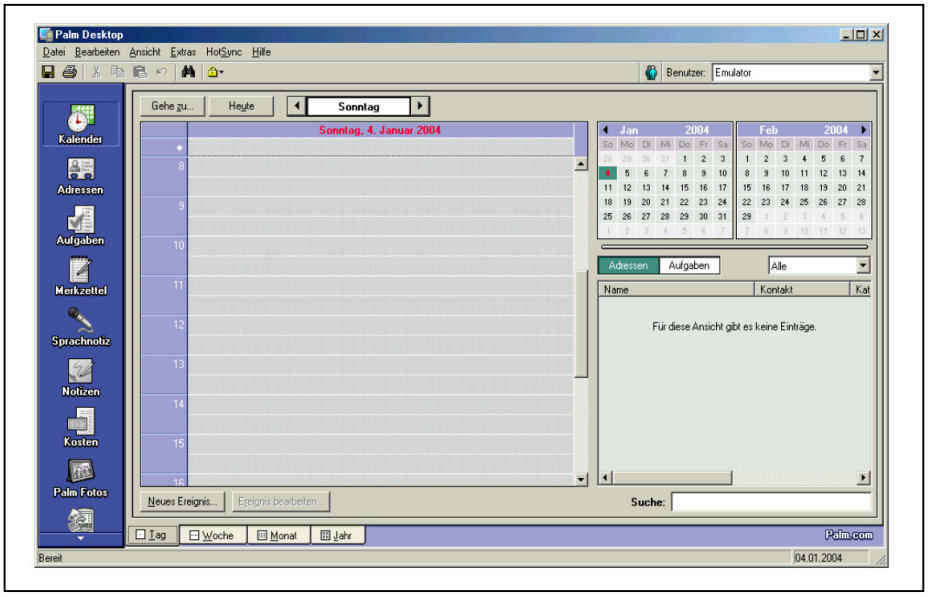

Abb. 2.1/3 „Anwendung Palm Desktop“

Pilot-link ist ein Softwarepaket, das standardmäßig bei vielen Linux-Distributionen dabei ist. Bei pilot-link stellt pilot-xfer ein wichtiges Programm für den Datenaustausch dar. Pilot-xfer besitzt die Funktionalität, Anwendungen auf dem Palm zu installieren, installierte Anwendungen aufzulisten, Backups anzufertigen und bei Bedarf auch Palm-Daten zu restaurieren.
KPilot (s. Abb. 2.1/4) ist ein Synchronisationsprogramm, das für die grafische Oberfläche KDE vorgesehen ist. In dem Programm KPilot ist der KpilotDaemon integriert, der es ermöglicht, dass HotSyncs über die Taste an der Dockingstation gestartet werden können. Die Funktionalitäten des KPilot sind das Durchführen von Palm-Backups/Palm-Restores, das Übertragen von Anwendungen auf den Palm, die Bearbeitung von Adressdatenbank und Merkzetteln sowie die Synchronisation von E-Mails, Adressdatenbank und Merkzetteln.

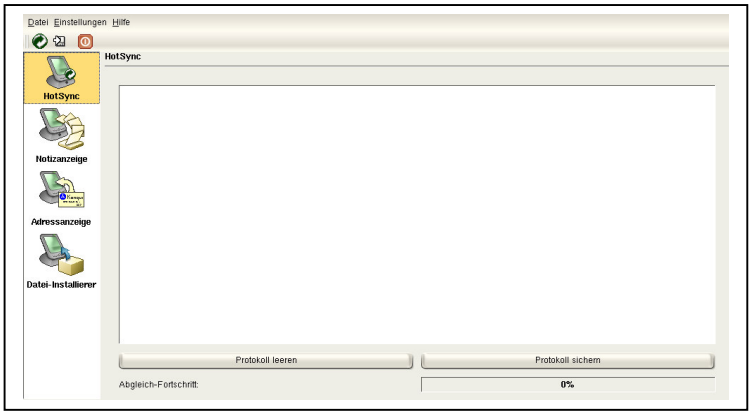

Abb. 2.1/4 „Anwendung KPilot“

JPilot (s. Abb. 2.1/5) weist ähnliche Funktionen auf.

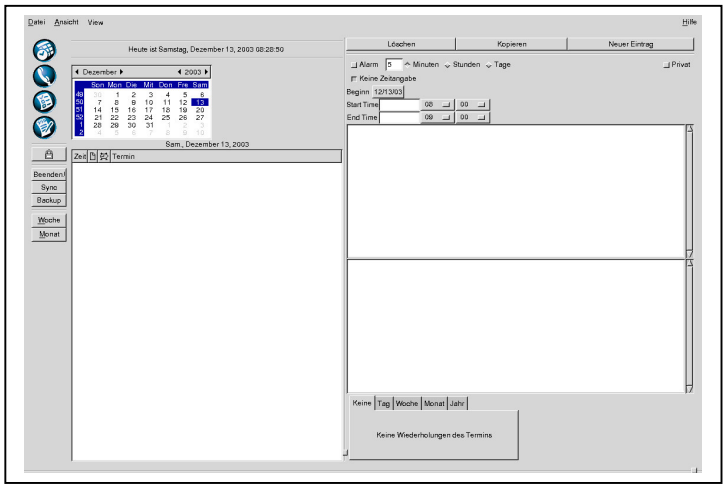

Abb. 2.1/5 „Anwendung JPilot“

In der Synchronisationssoftware für den Desktop ist der HotSync-Manager enthalten, ohne den eine Synchronisation zwischen Desktop und Handheld nicht möglich wäre. In Palm OS – Dokumentationen wird der HotSync-Manager als Desktopanwendung beschrieben, die die Kommunikation mit dem Handheld organisiert. HotSync nutzt dabei eine Kommunikations-API (die auch Sync Manager API genannt wird), um das aktuelle Senden und Empfangen von Bytes von und zum Handheld zu organisieren, wodurch der HotSync-Manager und die Sync Manager API unabhängig von der Verbindungsart sind. [IntroToConduits 2000, S. 3]
Die Aufgabe des HotSync-Managers ist es, HotSyncs auszuführen. Wie solche HotSync-Vorgänge ablaufen, soll im nachfolgenden (in Anlehnung an *Maxwell* [MAXW 1999, S.32]) erläutert werden.
Als erstes muss der HotSync-Knopf betätigt werden, woraufhin der Palm einige Aktionen ausführt. Danach wird der Kommunikationsport geöffnet. Im Anschluss daran werden durch den Palm Initiierungspakete über die Verbindung gesendet. Der Vorgang des Senden erfolgt solange, bis der Palm eine Bestätigung erhält oder der Prozess abgebrochen wurde, da innerhalb von 60 Sekunden keine Bestätigung erfolgt ist. Sobald der Desktop ein Initiierungspaket erhält, antwortet er und startet den HotSync-Vorgang. Bevor Daten geändert werden können, müssen noch einige Dinge erledigt werden. Der Handheld fordert eine Liste von Datenbanken an. Des Weiteren wird jede Datenbank dem dazugehörigen Conduit zugeordnet. Falls für eine bestimmte Datenbank kein Conduit gefunden werden kann, wird ein standardmäßiges backup Conduit genutzt und auf dem Desktop gesichert. Nun wiederum besitzt jedes Conduit die Chance, seine bestimmte Aufgabe auszuführen. Der Desktop nimmt die Rolle des Client an, das heißt, er stellt Anfragen an den Palm und dieser wiederum liefert Antworten.

Da im vorangegangenen Abschnitt öfters von Conduits die Rede war, soll nun im nachfolgenden Abschnitt auch auf die Conduits eingegangen werden.
Maxwell definiert die Conduits so: „A conduit is a piece of software that runs on the desktop and performs any necessary synchronization of data between the Palm and the desktop.“ [MAXW 1999, S.32] Conduits sind also Desktop-Software und für die Datensynchronisation zwischen Palm und Desktop zuständig.
Ihre Aufgaben bestehen darin, Daten für bestimmte Handheld-Anwendungen mit dem Desktop zu synchronisieren, Handheld-Datenbanken zu öffnen und zu schließen, Datensätze

auf dem Handheld und dem Desktop hinzuzufügen, zu löschen und zu verändern sowie erforderliche Formatkonvertierungen durchzuführen.
In der Palm-Dokumentation „Introduction to Conduits“ [IntroToConduits 2000, S. 10] wird die folgende Klassifikation von Conduits beschrieben:
Zunächst werden drei Arten von Conduits unterschieden – das Install Conduit, das Backup Conduit und das Synchronization Conduit.
Das Install Conduit installiert Palm OS Anwendungen und Datenbanken im Speicher des Handhelds oder Dateien auf Erweiterungskarten. Dieses Install Conduit ist standardmäßig schon beim HotSync-Manager vorhanden.
Die Aufgabe des Backup Conduits ist es, die gesamten Handheld-Datenbanken und Anwendungen auf den Desktop zu kopieren, ohne dabei einzelne Datensätze innerhalb der Datenbanken zu synchronisieren. Der HotSync-Manager startet das Backup Conduit nach dem Synchronization Conduit. Es ist für Datenbanken und Anwendungen gedacht, die keine zugehörigen Synchronization Conduits besitzen. Auch das Backup Conduit ist schon standardmäßig beim HotSync-Manager dabei.
Das dritte Conduit – nämlich das Synchronization Conduit – synchronisiert typischerweise die Handheld-Datenbanken mit ihren Desktoppendants. Zu beachten bei Synchronization Conduits ist, dass die Windows-Version des HotSync-Managers nur dann registrierte Synchronization Conduits ausführt, falls es auf dem Handheld eine Anwendung mit einer passenden creator ID gibt. [Anmerkung: Die creator ID identifiziert die Applikation oder Applikationsgruppe, die die Datenbank anlegen. Es ist wichtig, dass die creator ID einmalig ist, deshalb müssen auch die Entwickler von solchen Applikationen diese creator ID auf den Webseiten von Palm Computing registrieren.]
Neben der Klassifikation von Conduits beschreibt die Palm-Dokumentation „Introduction to Conduits“ [IntroToConduits 2000, S.13f] auch die Typen der Conduit -Synchronisation. Die Typen der Conduit-Synchronisation werden wie folgt eingeteilt: Es gibt den Typ mirror-image, one-directional, file linking und transaction based.
Die mirror-image-Synchronisation ist sehr gut für Anwendungen geeignet, die sowohl auf dem Handheld als auch auf dem Desktop laufen. Die Aufgabe der mirror-image-Synchronisation ist, dafür zu sorgen, dass nach der Synchronisation die Daten sowohl auf dem Handheld als auch auf dem Desktop identisch sind. Ein Beispiel für Anwendungen, die sich für diesen Synchronisationstyp eignen, ist das Adressbuch.
Der zweite Synchronisationstyp – nämlich die one-directional-Synchronisation – ist ebenfalls für Anwendungen geeignet, die auf Handheld und Desktop laufen. Allerdings ist bei diesem Synchronisationstyp nur eine Datenänderung auf einem der beiden möglich – entweder auf dem Handheld oder auf dem Desktop. Die Aufgabe besteht also darin, Daten von dem einen Gerät auf das andere zu kopieren. Das Aktiennotierungsprogramm wäre hier als ein Beispiel für Anwendungen zu nennen, die sich für die one-directional-Synchronisation eignen.
Ein weiterer Synchronisationstyp ist die file linking-Synchronisation. Sie ist für Handheldanwendungen gedacht, die Daten aus einer externen Datei einfügen und aktualisieren. Eine solche Handheldanwendung ist das Adressbuch. Zu beachten ist für diesen Synchronisationstyp, dass Daten nur von einer Desktopdatei zum Handheld transferiert werden und dass er nur von Windows unterstützt wird.
Ein letzter Synchronisationstyp ist die transaction based-Synchronisation. Sie ist für Anwendungen vorgesehen, die zusätzliche Aktionen zwischen den Datensatzsynchronisationen auf dem Desktop ausführen. Ein Vertreter für solch eine Anwendung sind E-Mail-Programme.

2.2. Entwicklungsumgebung

Um für Palm-Geräte Programme schreiben zu können, ist es notwendig, eine entsprechende Entwicklungsumgebung zu haben. Da in dieser Arbeit davon ausgegangen wird, dass die Programmierung mit der Programmiersprache C erfolgt, bieten sich zwei mögliche Entwicklungsumgebungen an.
Alle Palmprogrammierer, die unter Linux arbeiten, benötigen als Entwicklungsumgebung die PRC-Tools, das Software Development Kit (SDK) und das Paket PILRC.
Für alle Palmprogrammierer, die es bevorzugen unter Windows zu programmieren, bieten sich zwei Möglichkeiten – zum einen das kommerzielle Produkt Code Warrior und zum anderen besteht die Möglichkeit mittels Cygwin (ein Programm, mit dem Linux emuliert wird) mit den freien PRC-Tools zu arbeiten. Bei der „Cygwin-Variante“ ist die Programmiervorgehensweise ähnlich wie unter Linux.

Zunächst erst einmal soll die kommerzielle Entwicklungsumgebung Code Warrior vorgestellt werden, mit deren Hilfe auch später das „Hello World“ - Programm entstehen wird.

Code Warrior

Prinzipiell ist der Code Warrior eine kommerzielle Entwicklungsumgebung. Allerdings stellt Metrowerks – der Hersteller des Code Warrior – eine Demo-Version zur Verfügung. Der Unterschied zwischen Demo-Version und Voll-Version besteht hauptsächlich darin, dass der Programmierer bei der Demo-Version eine Beschränkung im Quellcodeumfang hinnehmen muss.
Das Produkt Code Warrior enthält einen Editor, einen Projektmanager, einen C/C++ Compiler, einen Linker und eine Schnittstelle für das Zielgerät, welche das Debuggen auf Quellcode- und Assemblerebene ermöglicht.
Für das Programmieren von Anwendungen für den Palm werden die Code Warrior IDE und der Constructor for Palm OS benötigt. Der Constructor for Palm OS ist ein Werkzeug mit dessen Hilfe man die grafischen Oberflächen des zu entwickelnden Programms und deren Eigenschaften zusammenstellen kann.
In der Code Warrior IDE stehen der Projektmanager, der Editor, der C/C++ Compiler, der Linker und die Schnittstelle für das Zielgerät zur Verfügung.

PRC-Tools und PILRC

Da sich die Arbeit schwerpunktmäßig der Palm-Programmierung mittels Code Warrior widmet, soll an dieser Stelle auch nur ein kurzer Überblick über die PRC-Tools erfolgen, da sie die größte Alternative zur Programmierung mit Code Warrior darstellen.

Die PRC-Tools bestehen aus dem GNU C Compiler (GCC), einem assembler, linker und einem symbolic debugger. Diese Tools sind an die Programmierung für Palm angepasst, so dass die für Palm OS eigentümlichen Funktionen unterstützt werden.
Die PRC-Tools alleine reichen jedoch noch nicht aus, um für Palm-Geräte Anwendungen zu programmieren. Es wird nämlich noch das SDK und das Paket PILRC benötigt.
Im SDK sind die Header-Dateien und Bibliotheken enthalten, die man benötigt, um auf die Palm OS- Funktionen zugreifen zu können.

Die Aufgabe des Pakets PILRC besteht darin, die Ressource-Dateien, die die grafischen Oberflächen für die Palm-Programme beinhalten, zu einer prc-Datei zusammenzusetzen, die von dem Palm ausgeführt werden kann.

Emulatoren

Der Emulator ist eine „Software, mit der sich die Hardware des Palm auf dem Desktop-Computer nachstellen lässt“. [IMML u.a. 2000, S.462]
Der Emulator bildet allerdings nur den Rahmen für die Hardware-Emulierung. Damit man mit dem Emulator auch effektiv arbeiten kann, ist es notwendig, sich das ROM-File des eigenen Palms entweder selbst vom eigenen Palm auf den Desktop zu ziehen oder aber es sich von der Firma Palm Computing zu beschaffen. Erst wenn man sich dieses ROM-File in den Emulator geladen hat, kann man mit dem Emulator unter den Bedingungen arbeiten, wie sie auch bei dem eigenen Palm existieren.
Der Emulator bietet vor allem für die Programmierung von Palm-Anwendungen einige Vorteile. Wenn der Programmierer Quellcode für eine von ihm neu entwickelte Palm-Anwendung testen und debuggen möchte, muss er nicht erst eine Verbindung zu seinem Palm herstellen, sondern kann das Testen und Debuggen gleich vorort an seinem Arbeitsrechner mit Hilfe des Emulators durchführen. Auf diese Weise können dem Programmierer unangenehme Abstürze seines Palms erspart bleiben.
Der gängigste Emulator ist der Palm OS Emulator (POSE). Ihn gibt es sowohl für Windows als auch für Linux. Für Linux gibt es außerdem noch den mittlerweile veralteten Emulator xCopilot.
Der POSE macht es möglich, Palm OS-Software auf Macintosh, Unix und Windows sowohl zu testen als auch zu debuggen. Wenn man eine Palm OS-Anwendung mit dem POSE auf dem PC laufen lässt, holt sich der POSE Anweisungen, aktualisiert die Bildschirmanzeige des emulierten Palms, arbeitet mit speziellen Registern und geht mit Unterbrechungen genauso um, wie es auch der Prozessor des Handhelds tun würde. Der Unterschied besteht lediglich darin, dass der POSE diese Anweisungen in Software auf dem PC ausführt.
Des Weiteren besitzt der POSE viele Funktionen. So bildet er zum Beispiel das Display des Handhelds exakt nach, inklusive dem Graffitibereich und den umliegenden Icons. Für das Display hat man auch die Möglichkeit, die Helligkeit der Hintergrundbeleuchtung zu variieren. Außerdem besitzt er ebenfalls wie der Palm die Buttons für die Anwendungen zum Hinauf- und Hinabscrollen und zum Ein- bzw. Ausschalten. Man kann den Emulator sogar mit der Palm Desktop-Software synchronisieren, was auch eine wesentliche Voraussetzung für das Programmieren von Palm-Anwendungen ist. Denn nur, wenn mit dem POSE auch eine korrekte Synchronisation möglich ist, kann man den Emulator zur Entwicklung von Palm-Software mit einbeziehen.

2.3. Erläuterungen am Beispiel eines „Hello World“ - Programms

Zunächst sei erst einmal das Ziel des „Hello World“ – Programms beschrieben. Nach dem Starten der Anwendung „Hello World“ erscheint auf dem Display des Palms ein Button mit dem Schriftzug „Hello World?“. Immer wenn der Anwender diesen Button betätigt, wird auf dem Display an einem zufälligen Ort der Schriftzug „Hello World!“ ausgegeben. Beendet wird die Anwendung, indem man über die Kalender-, Adressen-, Aufgabenlisten- oder Notizentaste am Palm eine andere Anwendung aufruft.

Das Ergebnis sieht dann wie folgt aus:

Abb. 2.3/1 „Emulator, auf dem die „Hello World“ -Anwendung läuft“

Als nächstes soll der Weg zum fertigen „Hello World“ –Programm beschrieben werden. Dabei werden das Vorgehen beim Erstellen eines Programms mittels Code Warrior sowie Quellcodebestandteile mit dazugehörigen Erläuterungen vorgestellt.

Als ersten Schritt gilt es, im Code Warrior ein neues Projekt anzulegen und die vom Code Warrior generierten C-Sourcen, Header-Dateien und Ressourcen-Dateien aus dem Projektmanager zu eliminieren.
Als nächstes erstellt man sich seine gewünschte grafische Oberfläche mit Hilfe des Tools Constructor for Palm OS. Dabei ist zu beachten, dass man einstellt, dass automatisch eine Header-Datei generiert wird.

Für das in dieser Arbeit verwendete Beispiel „Hello World“ sehen die Eigenschaften dann wie folgt aus:

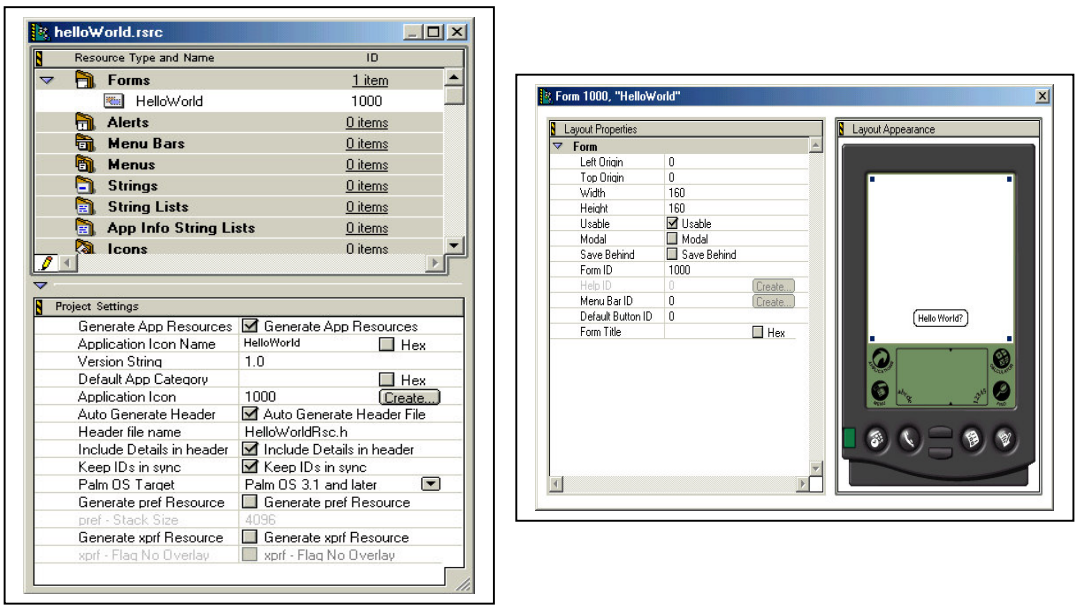

Abb. 2.3/2 „Eigenschaften der grafischen Oberfläche für das „Hello World“ - Beispiel“

Wenn man die neu gebastelte grafische Oberfläche abspeichert, entsteht die Ressourcen-Datei mit der Endung *.rsrc und es wird die Header-Datei für die Ressourcen-Datei automatisch generiert.

Die Header-Datei „HelloWorldRsc.h“beinhaltet folgenden Quellcode (s. Abb. 2.3/3):

```
#define HelloWorldForm                  1000
#define HelloWorldHelloWorldButton      1001
```

Abb. 2.3/3 „Quellcode der Header-Datei HelloWorldRsc.h“

Hierbei werden Variablen für die grafischen Elemente definiert, denen IDs zugewiesen werden. Diese Zuweisung hat den Vorteil, dass man die grafischen Elemente in dem eigentlichen Programm (hier: helloWorld.c) durch den Namen ansprechen kann.

Nachdem die grafische Oberfläche jetzt fertig ist, wechselt man wieder zur Code Warrior IDE und fügt im Projektmanager die Ressourcen-Datei dem Resources-Ordner und die Header-Datei dem Headers-Ordner hinzu.
Im nächsten Schritt legt man ein neues Textfile an und fügt dieses dem Source-Ordner hinzu. Bei dem hier verwendeten „Hello World“ - Beispiel heißt diese Datei helloWorld.c.

Jetzt kann man mit dem Programmieren des Hauptprogramms beginnen. Dabei geht man wie folgt vor:

1. Includes

Als erstes werden die benötigten Header-Dateien eingebunden (s.Abb. 2.3/4). Bei der Datei HelloWorldRsc.h handelt es sich um die von Code Warrior automatisch generierte Header-Datei.

```
#include <PalmOS.h>
#include <Window.h>

/* resource file header */
#include "HelloWorldRsc.h"
```

Abb. 2.3/4 „Einbinden der Header-Dateien“

2. Prototypes

An dieser Stelle wird die form handler function und die application stop function definiert (s. Abb. 2.3/5).

```
/* form handler function */
static Boolean HelloHandleEvent( EventType* event);

/* application stop function */
void AppStop( void )
```

Abb. 2.3/5 „Definition der form handler function und der application stop function“

3. PilotMain

Wie in Abb. 2.3/6 ersichtlich ist, wird in der Funktion ***PilotMain*** zunächst der „Launch Code“ ausgewertet. Das ist notwendig, damit das Programm weiß, ob es beispielsweise im Rahmen eines HotSyncs aufgerufen wurde oder ob das Programm durch einen Benutzer aufgerufen wurde. Beide Aufrufmöglichkeiten erfordern ein unterschiedliches Verhalten. Sobald der Benutzer die Anwendung aufruft, generiert das System den ***sysAppLaunchCmdNormalLaunch***. Damit weiß die Anwendung, dass sie komplett starten soll, was sie dann auch tut. Ihre grafische Oberfläche wird somit auf dem Display angezeigt.
Nachdem der Launch Code ausgewertet wurde, könnte jetzt eine Überprüfung der Palm OS-Version erfolgen, damit mögliche Abstürze wegen Nichtkompatibilität des Programms vermieden werden können. Darauf wurde allerdings in dem hier verwendeten Beispielprogramm verzichtet, damit es ein einfaches und verständliches „Hello World“-Programm bleibt. Aus dem Grund der Einfachheit des Programms gibt es an dieser Stelle auch keine Funktion namens ***AppStart***, mit der man zum Beispiel die Datenbank für das Programm öffnen könnte.
Als nächstes wird im „HelloWorld“- Programm (s. Abb. 2.3/6) mit ***FrmInitForm(HelloWorldForm)*** das HelloWorldForm (so heißt das Form der grafischen Oberfläche des „Hello World“- Programms) als Form initialisiert. Im Anschluss daran wird mit ***FrmSetEventHandler(form, HelloHandleEvent)*** der form handler function das

selbstentwickelte ***HelloHandleEvent*** zugewiesen. Nachdem das geschehen ist, erhält das Form mit ***FrmSetActiveForm(form)*** den Focus und wird mit ***FrmDrawForm(form)*** angezeigt.
Ein weiterer wichtiger Teil der Funktion ***PilotMain*** ist die ***event loop***. Normalerweise empfiehlt es sich, eine eigene Funktion (zum Beispiel mit dem Namen ***AppEventLoop***) für sie zu schreiben. Da es sich aber bei dem hier verwendeten „Hello World"- Beispiel um ein recht übersichtliches Programm handelt, wurde hier in diesem speziellen Fall darauf verzichtet.
Allgemein gilt, dass Programme für den Palm Event-gesteuert sind. Soll also das Programm auf etwas reagieren, wird diese Information durch das Betriebssystem an das Programm in Form von Events übermittelt. Das ist auch der Grund, warum beim Programmieren von Palm-Anwendungen die ***event loop*** verwendet wird. Die ***event loop*** ist eine Do-While-Schleife, in der die einzelnen Events ausgewertet und bearbeitet werden. Sie wird erst dann verlassen, wenn die laufende Anwendung durch das Starten einer anderen Anwendung unterbrochen wird.
Zu Beginn der ***event loop*** wird mit ***EvtGetEvent*** das nächste Event geholt. Im nächsten Schritt wird geprüft, ob es sich bei dem geholten Event um ein ***SysHandleEvent*** (welches ein Systemevent ist) handelt. Solche Events werden beispielsweise ausgelöst, wenn entweder die Kalender-, Adressen-, Aufgabenlisten- oder Notizentaste gedrückt wird. Handelt es sich bei dem geholten Event nicht um ein ***SysHandleEvent***, wird das Event zumindestens in diesem hier verwendeten „HelloWorld"- Programm an den Event Handler des aktuellen Forms weitergegeben. Das erfolgt mittels der Funktion ***FrmDispatchEvent***.
In umfangreicheren Programmen ist es üblich, nach der Prüfung, ob ein ***SysHandleEvent*** vorliegt, eine Prüfung auf Vorliegen eines ***MenuHandleEvents*** vorzunehmen. Liegt eins vor, dann wird dieses bearbeitet, andernfalls erfolgt eine Prüfung auf Vorliegen eines ***ApplicationHandleEvents***. Handelt es sich um solch ein Event, dann wird es bearbeitet. Ist es kein solches Event, dann wird das Event an den Event Handler des aktuellen Forms weitergegeben.
Die ***event loop*** wird so lange ausgeführt, bis sie durch ein ***appStopEvent*** beendet wird. Dieses ***appStopEvent*** kann zum Beispiel durch den Aufruf einer anderen Anwendung ausgelöst werden.
Da die Anwendung nachdem Auslösen des ***appStopEvent*** noch sauber beendet werden muss, wird nach der ***event loop*** noch die Funktion ***AppStop*** aufgerufen.

```
/* main function */
UInt32 PilotMain( UInt16 cmd, MemPtr, UInt16)

{
   FormPtr      form;   /* pointer to form structure */
   EventType  event;  /* event structure */

   /* If this is not a normal launch, don't launch */
   if( cmd != sysAppLaunchCmdNormalLaunch )
      return( 0 );

   /* Initialize form */

   form = FrmInitForm( HelloWorldForm );
   FrmSetEventHandler( form, HelloHandleEvent );
   FrmSetActiveForm( form );
   FrmDrawForm( form );

   /* event loop */
   do
   {
      /* Get the next event */
      EvtGetEvent( &event, evtWaitForever );

      /* Handle system events */
      if( SysHandleEvent( &event ) )
         continue;

      /* Handle form events */
      FrmDispatchEvent( &event );

   /* If it's a stop event, exit */
   } while( event.eType != appStopEvent );


   AppStop();

   return( 0 );
}
```

Abb. 2.3/6 „Funktion PilotMain“

4. AppStop

Die in der Abbildung 2.3/7 dargestellte Funktion ***AppStop*** wird beim Beenden der Anwendung aufgerufen. Ihre Aufgabe ist es, die Anwendung sauber zu beenden. In dem hier vorliegenden Beispiel werden mit ihr alle geöffneten Forms der Anwendung geschlossen.

```
/* application stop function */
void AppStop( void )
{
        FrmCloseAllForms ();
}
```

Abb. 2.3/7 „Funktion AppStop“

5. HelloHandleEvent

Die in Abbildung 2.3/8 gezeigte Funktion ***HelloHandleEvent*** stellt die Form Handler Funktion dar. Sie wurde im Hauptprogramm (***PilotMain***) mittels ***FrmSetEventHandler*** dem Form „HelloWorldForm“ als Form Handler Funktion zugewiesen. Aufgrund dieser Zuweisung ist es der Funktion ***FrmDispatchEvent*** möglich, das ***FrmHandleEvent*** der Funktion ***HelloHandleEvent*** zu übergeben.
Die Aufgabe der Funktion ***HelloHandleEvent*** besteht darin, dass wenn durch das Betätigen des „HelloWorld?“- Buttons ein Event ausgelöst wird, an einer beliebigen Stelle auf dem Palm-Display der Schriftzug „Hello World!“ausgegeben wird.

```
/* form handler function */
static Boolean HelloHandleEvent( EventType* event )
{
        /* Parse the event */
        if( event->eType == ctlSelectEvent )
                WinDrawChars("Hello World!",12, SysRandom(0) %100,
SysRandom(0) %110+20);

        return( false );
}
```

Abb. 2.3/8 „Funktion HelloHandleEvent“

Nachdem das Hauptprogramm fertig ist, kann man es jetzt compilieren und linken, was im Code Warrior über den Menüpunkt Make erfolgt.
Im Anschluss daran bietet es sich an, das compilierte Programm mit Hilfe des Debuggers und des Emulators zu debuggen. Bei Code Warrior und Palm OS ist der Debugger auf dem PC zu finden. Der Debugger kommuniziert mit der Konsolenapplikation des Palms (hier in diesem Fall ist das die Konsolenapplikation des Emulators) und testet das Programm zur Laufzeit. Wenn man nach dem Debuggen zu der Erkenntnis kommt, dass das Programm das tut, was man sich vorgestellt hat, dann kann man die durch Code Warrior generierte *.prc Datei per HotSync auf den Palm überspielen.

3. Zusammenfassung und Fazit

Die drei wesentlichen Ziele dieser Arbeit waren:

(1) kurze Erläuterung der Unterschiede zwischen einem herkömmlichen PC und einem Palm-Gerät
(2) Erläuterung von Synchronisation und Conduits
(3) etwas zu den Themen Emulatoren, Code Warrior, Palm-API und PRC-Tools zu sagen.

Diese Ziele wurden im Wesentlichen erreicht. Aufgrund der Umfangsbegrenzung war es jedoch nötig Schwerpunkte zu setzen. Diese lagen vorrangig auf den Zielen eins und zwei, sowie auf der Thematik Palm-Programmierung mit Code Warrior. Das hat zur Folge, dass die Thematik der PRC-Tools nur kurz angeschnitten werden konnte.

Abschließend kann man sagen, dass die Palm-Programmierung ein sehr interessantes Themenfeld ist und für jeden geeignet ist, der Kenntnisse in C-Programmierung besitzt und eventuell einen Palm sein Eigen nennen kann.
Ein großer Vorteil für die Palm-Programmierung besteht darin, dass man die Möglichkeit hat, ohne kommerzielle Produkte auszukommen – dank PRC-Tools, PILRC und SDK. Jedoch sollte man bedenken, dass wenn man professionell Palm-Programmierung betreiben will, ein Tool wie Code Warrior eine enorme Arbeitserleichterung darstellen kann.

Abkürzungsverzeichnis

API	**A**pplication **P**rogramming **I**nterface
CDK	**C**onduit **D**evelopment **K**it
DLL	**D**ynamic **L**ink **L**ibrary
GCC	**G**NU **C** **C**ompiler
IrDA-Schnittstelle	**I**nfra**r**ed **D**ata **A**ssociation Schnittstelle
Palm-API	Palm **A**pplication **P**rogramming **I**nterface
POSE	**P**alm **OS** **E**mulator
SDK	**S**oftware **D**evelopment **K**it

Abbildungsverzeichnis

Literaturverzeichnis

[IMML u.a. 2000] Immler, C. und Salomon N., Das große Buch Palm, 1. Auflage, Verlag DATA BECKER, Düsseldorf. 2000.

[IntroToConduits 2000] Introduction to Conduit Development http://www.palmos.com/dev/support/docs/conduits/win/

[MAXW 1999] Maxwell, G., Sams Teach Yourself Palm Programming in 24 Hours, 1. Auflage, Verlag Sams Publishing, Indianapolis. 1999.

Anhang

Damit der Quellcode des „Hello World“- Programms noch einmal auf einen Blick verfügbar ist, wird die Datei helloWorld.c an dieser Stelle noch einmal komplett abgebildet:

```
#include <PalmOS.h>
#include <Window.h>

/* resource file header */
#include "HelloWorldRsc.h"

/* form handler function */
static Boolean HelloHandleEvent( EventType* event);

/* application stop function */
void AppStop( void );

/* main function */
UInt32 PilotMain( UInt16 cmd, MemPtr, UInt16)
{
   FormPtr      form;   /* pointer to form structure */
   EventType  event;  /* event structure */

   /* If this is not a normal launch, don'  t launch */
   if( cmd != sysAppLaunchCmdNormalLaunch )
      return( 0 );

   /* Initialize form */
   form = FrmInitForm( HelloWorldForm );
   FrmSetEventHandler( form, HelloHandleEvent );
   FrmSetActiveForm( form );
   FrmDrawForm( form );

    /* event loop */
   do
   {
      /* Get the next event */
      EvtGetEvent( &event, evtWaitForever );

      /* Handle system events */
      if( SysHandleEvent( &event ) )
         continue;

      /* Handle form events */
      FrmDispatchEvent( &event );

   /* If it'  s a stop event, exit */
   } while( event.eType != appStopEvent );
```

```
    AppStop();

    return( 0 );
}


/* form handler function */
static Boolean HelloHandleEvent( EventType* event )
{
        /* Parse the event */
        if( event->eType == ctlSelectEvent )
                WinDrawChars("Hello World!",12, SysRandom(0) %100,
SysRandom(0) %110+20);

        return( false );
}


/* application stop function */
void AppStop( void )
{
        FrmCloseAllForms ();
}
```