PalmPilotKiosk I:

Wireless Local Network for Palmtop Computers

Jan Beutel, Tobias Bösch
Electronics Laboratory, ETH Zürich
Phone: +41-1-632 51 44
FAX: +41-1-632 12 10
e-mail: jbeutel@ife.ee.ethz.ch

6th February 1998

Contents

Tables

Figures

Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Departement Elektrotechnik
Institut für Elektronik

Wintersemester 1997/98

SEMESTERARBEIT

für

Jan Beutel und Tobias Bösch

Betreuer: Rolf Sommerhalder, ETZ H60.1
Stellvertreter: Thomas Sailer, ETZ H60.1

Ausgabe: 21.Oktober 1997
Abgabe: 6. Februar 1998

PalmKiosk I: Drahtloses lokales Netzwerk für Palmtop Computer

Einleitung

Wieso geht man heute noch am Kiosk vorbei und kauft sich Tageszeitungen, die gedruckt, transportiert und entsorgt werden müssen? Portable Computer wie der PalmPilot (Figur 0-1) könnten doch interessierende Nachrichten bei “elektronischen Kiosken” im Vorbeigehen herunterladen und bezahlen!

Portable, digitale Assistenten (PDA) sind batteriebetriebene Computer, die typischerweise kleiner, leichter und billiger sind als Laptops. Herkömmliche PDA wurden bisher als Ersatz fuer Agenda mit Terminkaldender, Alarmfunktionen sowie als Adressdatenbank und Telefonbuch verwendet. Neuere PDA (PalmPilot von U.S. Robotics/3Com, Newton von Apple) können auch benutzt werden, um E-Mail abzurufen und zu beantworten, oder um Zeitungen (“News”) in elektronischer Form zu lesen. Auf Informationsservern stellen Agenten und Filter die Zeitungen nach Interessenprofil des Benutzers individuell aus verschiedensten Quellen zusammen (Newsserver, WWW-Seiten, Presseagenturen, Börsen, usw.).

Drahtlose Datenübertragung zu niedrigen Kosten wird von den modernen Computer- und Kommunikations-“Nomaden” gefordert, um volle Bewegungsfreiheit zu erreichen. In Räumen kann dies mittels Infrarotübertragung (IrDA) mit hoher

Figure 0-1
Erweiterung des PalmPilot PDA mit drahtloser Datenkommunikation.

Bandbreite und dafür kurzer Reichweite geschehen. Für Verbindungen innerhalb von Gebäuden bieten sich Standards wie z.B. Digital European Cordless Telephone (DECT) an. Paketorientierte Datenfunknetze wie Mobitex [8] und verbindungsorientierte Mobiltelefonsysteme wie GSM können grössere Distanzen überbrücken. Mobilsatellitensysteme wie Inmarsat ermöglichen bereits heute weltweite Datenverbindungen mittels portabler Bodenstationen.

Für Experimente auf einem Hochschulcampus bietet sich von von den Kosten, der Reichweite und der Machbarkeit her gesehen das “Industrial, Scientific and Medical” (ISM) Band bei 430 MHz an. Werden homologierte Sende-/Empfängermodule eingesetzt, deren abgestrahlte Sendeleistung gewisse Grenzwerte nicht überschreitet, so dürfen diese Funknetze lizenzfrei betrieben werden. Diese Module sind preiswert (<100 SFr. pro Modul) und lassen Datenverbindungen über einige hundert Meter zu.

In dieser Arbeit soll aus Standardbausteinen eine Erweiterung zum PalmPilot entwickelt werden, welche drahtlose Verbindungen von mehreren PDA mit einem Server, oder von mehreren PDA untereinander, ermöglicht (Figur 0-2). Diese Erweiterung soll weniger als 200 SFr. kosten und mit zwei AAA-Batterien auskommen. Sie soll an der seriellen Schnittstelle des PalmPilot angeschlossen werden, evtentuell auch an einer seriellen Schnittstelle von anderen Rechnern. Stellen Sie die notwendige Systemsoftware für eine fehlerfreie Datenübertragung bereit, die den zur Verfügung stehenden Übertragungskanal optimal benutzt. Der Zugriff auf diesen einzigen Kanal soll für alle Benutzer fair geregelt werden. Weil der Kanal im ISM-Band liegt und von fremden Systemen teilweise gleichzeitig belegt werden kann, müssen die eingesetzten Protokolle fehlertolerant und robust ausgelegt werden.

In Kooperation mit der parallel laufenden Semesterarbeit “PalmKiosk II” sollen die Hardwareerweiterung, die erforderlichen Softwareschnittstellen sowie die

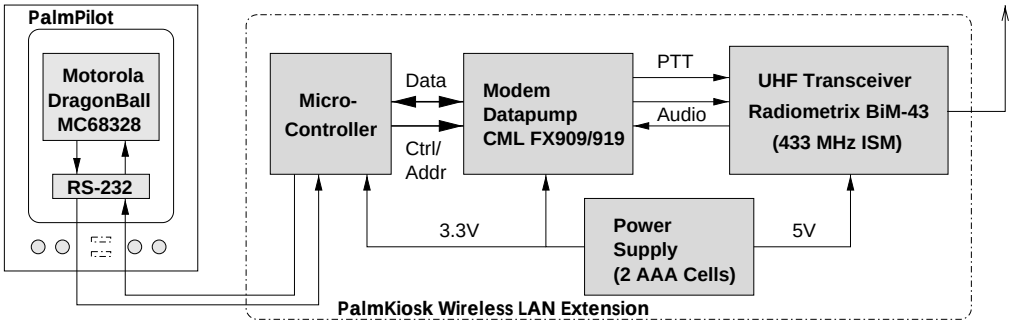


Figure 0-2
Blockdiagramm des drahtlosen Netzadapters.

Datenlink- und Netzwerkprotokolle spezifiziert und implementiert werden.

Aufgabenstellung

1. Erstellen Sie einen Projektplan und legen Sie Meilensteine fest [1]. Erarbeiten Sie in Absprache mit dem Parallelprojekt “PalmKiosk II” und dem Betreuer ein Pflichtenheft. Es ist vorgesehen, dass Standardbausteine wie z.B. das Sende-/Empfangsmodul [14] sowie ein stromsparender Modemchip [16, 15] eingesetzt werden. Diese Bausteine lösen bereits wesentliche Teilprobleme wie Synchronisation, Kapselung in Pakete (Framing) sowie Fehlerdetektion und -korrektur.
2. Führen Sie eine Literaturrecherche zu Themen wie Wireless LAN, Datenlinkprotokolle, oder Kanalzugriff (Medium Access Control) durch. Ausgangspunkte bilden z.B. die Arbeiten [17, 3, 5, 7, 6, 4]. Suchen Sie nach neueren Publikationen.
3. Erstellen Sie nach dem Literaturstudium in Absprache mit dem Parallelprojekt “PalmKiosk II” Varianten des Erweiterungsmoduls für die drahtlose Datenübertragung. Wägen Sie die Vor- und Nachteile der verschiedenen Varianten gegeneinander ab, unter Berücksichtigung von Kriterien wie: Stromverbrauch, Preis, Auswirkungen auf die Systemkomplexität und Einfachheit der Software.
4. Stellen Sie zwei oder drei Prototypen des Erweiterungsmodules her und überprüfen Sie deren Funktionstüchtigkeit durch Messung von Bit Error Raten in Abhängigkeit zum Signal-Rauschabstand.
5. Arbeiten Sie sich in die Softwareentwicklungsumgebung des PalmPilot ein [13, 12, 11, 10]. Implementieren Sie Systemsoftwareroutinen (Driver) zum Ansteuern Ihres Erweiterungsmodules durch die serielle Schnittstelle des PalmPilot und evtl. einer Sun Workstation als Gateway zum Internet.

6. Entwerfen und implementieren Sie ein *einfaches Kanalzugriffsverfahren sowie einfache und robuste Datenlink- und Netzwerkprotokolle.* Demonstrieren Sie deren Funktionstüchtigkeit anhand einer kleinen Anwendung wie z.B. Datentransfer zwischen zwei PalmPilot.
7. Dokumentieren Sie Ihre Arbeit sorgfältig mit einem Vortrag, einer kleinen Demonstration, sowie mit einem Schlussbericht.

Durchführung der Semesterarbeit

Allgemeines

- Der Verlauf des Projektes “Semesterarbeit” soll laufend anhand des Projektplanes und der Meilensteine evaluiert werden. Unvorhergesehene Probleme beim eingeschlagenen Lösungsweg können Änderungen am Projektplan erforderlich machen. Diese sollen dokumentiert werden.
- Sie verfügen über einen PC mit Protel oder eine Sun Workstation mit Cadence für die Schemaerfassung und das Printlayout. Die Softwareentwicklung für den PalmPilot sowie für den eventuell erforderlichen Mikrocontroller kann auf PC oder Sun erfolgen.
- Stellen Sie Ihr Projekt zu Beginn der Semesterarbeit in einem Kurzvortrag (am *7. November 1997*, 5 Minuten) vor und präsentieren Sie die erarbeiteten Resultate am Schluss im Rahmen des Institutskolloquiums Ende Semester.
- Besprechen Sie Ihr Vorgehen regelmässig mit Ihrem Betreuer.

Abgabe

- Geben Sie zwei unterschriebene Exemplare des Berichtes spätestens am *6. Februar 1998* dem betreuenden Assistenten oder seinen Stellvertreter ab. Diese Aufgabenstellung soll vorne im Bericht eingefügt werden (vgl. [1], Kap. 1.7 Bericht).
- Räumen Sie Ihre Rechnerkonten soweit auf, dass nur noch die relevanten Source- und Objectfiles, Konfigurationsfiles, benötigten Directorystrukturen usw. bestehen bleiben. Eine spätere Anschlussarbeit soll auf dem hinterlassenen Stand aufbauen können.

Bibliography

[1] M.Thaler. *Semester- und Diplomarbeiten am Institut für Elektronik*. IFE/ETHZ, 10/95, Oktober 1995.

[2] D. Brown. *Techniques for Privacy and Authentication in Personal Communication Systems*. IEEE Personal Communications, pages 6-10, August 1995.

[3] K.-C. Chen. *Medium Access Control of Wireless LANs for Mobile Computing*. IEEE Network, pages 50-63, Sept./Oct. 1994.

[4] V. N. Padmanabhan. *Design and Evaluation of a Reliable Link-Layer Protocol*. Class Project, Spring 1996.

[5] G. H. Forman and J. Zahorjan. *The Challenges of Mobile Computing*. IEEE Computer, pages 38-47, April 1994.

[6] M.T. Le and S. Seshan and F. Burghardt and J. Rabaey. *Software Architecture of the Infopad System*. Proceedings of Mobidata Workshop on Mobile and Wireless Information Systems, New Brunswick, NJ, Nov. 1994.

[7] Tomasz Imielinski and Henry F. Korth. *Mobile Computing*. Kluwer, 1996.

[8] A. K. Salkintzis and C. Chamzas. *Mobile Packet Data Technology: An Insight into MOBITEX Architecture*. IEEE Personal Communications, pages 10-18, Feb. 1997.

[9] M. Sirbu and J. D. Tygar. *NetBill: An Internet Commerce System Optimized for Network-Delivered Services*. IEEE Personal Communications, pages 34-39, Aug. 1995.

[10] Motorola Inc. *Integrated Portable System Processor – DragonBall MC68328*. 1995. http://www.mot.com/SPS/ADC/pps/prod/3XX/mc68328.html

[11] Metrowerks Inc. *CodeWarrior C-Crosscompiler Development Tools*. http://www.metrowerks.com/.

[12] D. Massena. *Pilot Software Development WWW-Pages*. http://www.massena.com/darrin/pilot/.

[13] Albert. *Pilot-UNIX mailing list archive*. http://www.acm.rpi.edu/~albert/pilot/.

[14] Radiometrix Ltd. *Low Power UHF Data Transceiver Module BiM-433-F*. Sept. 1995.

[15] Consumer Microcircuits Ltd. (CML Semiconductors). *Wireless Modem Data Pump FX909A*. March 1996.

[16] Consumer Microcircuits Ltd. (CML Semiconductors). *4-Level FSK Modem Data Pump FX919A*. March 1996.

[17] H. Balakrishnan and V. N. Padmanabhan and S. Seshan and R. Katz. *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*. SIGCOMM Conference, ACM, Aug. 1996.

Motivation

The initial motivation to embark on this project was that among the vast amount of projects offered for the winter term 97/98 at ETH Zurich it had a clear goal and resembled much of a complete project. We liked that we would have to work in many different fields, offering us to learn about the entire design process to develop and implement a complete product and not only the parts of one. This would offer us to practice those skills that lectures cannot offer.

Thanks

At this point we would like to thank Rolf Sommerhalder, Thomas Sailer as well as the rest of the staff at the ETH Electronics Laboratory for their great attitude and help with this project. It was a pleasure to find such an open and inspiring atmosphere with you all.

In addition to that we would like to thank Mr. Rütimann of Mero, Mr. Sanchez of Maxim Germany, the companies Eurodis and Elbatex for their kind contribution of IC samples for our prototype boards.

Andi Karrer was especially helpful with his hints on LaTeX that were necessary to complete this document.

Zurich, February 6th, 1998

Jan Beutel

Tobias Bösch

1

Overview on Mobile Computing

1.1 General Overview

In our modern, mobile society the unrestricted use of communication and computing devices is becoming a part of daily life. Not only researchers and technicians make use of the advantages of new devices and functionalities, every person encounters numerous applications and equipment in everyday life even though he or she might

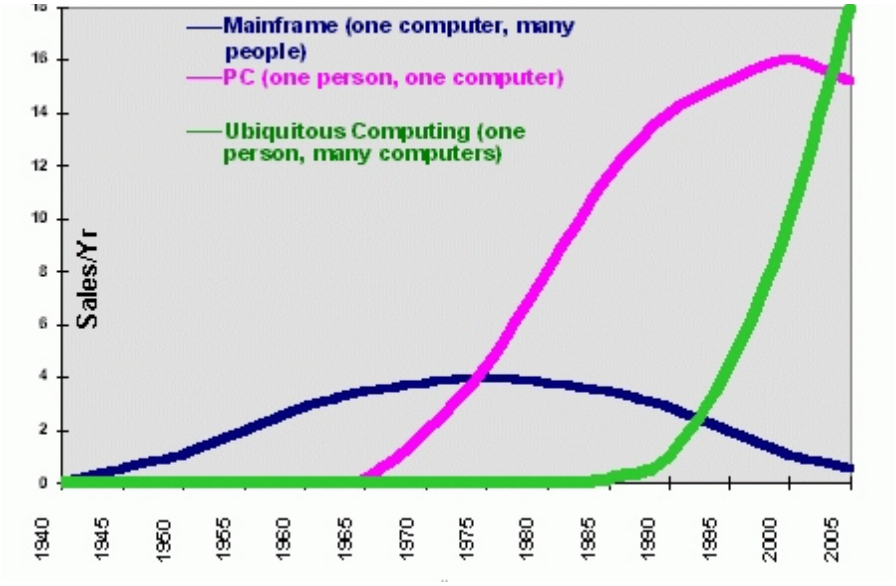


Figure 1-1
The major trends in computing.

not at all be aware of them being an electronics or even computing device.

From this non-awareness of the user of the actual technology we can learn that people using modern devices must not be skilled or even specially trained to use them but should be able to do so intuitively, on their own and quickly. More and more devices and services are being integrated into a whole or are being made remotely accessible from common interfaces.

In the near future tens of millions of people will carry portable or laptop computers with wireless communication connections to fixed networks and to other mobile computers (Figure 1-1). These environment demand on an entire new class of applications that focus on unrestricted mobility, access and portability.

As more and more people are getting used to mobile telephones, pagers, labtop and palmtop computers with network connection and other remote services, it is now time for the evaluation of these systems. One of the factors users of mobile telephones point out to be by far the most important for the use and sale of the equipment is the time available for use with a single battery package, or in other words the energy consumption of the device. Size, ergonomical features, functionality, services included or even the electromagnetic radiation from the antenna are only minor factors. This has been observed since the first introduction of such devices and it is feasible to say, that in the near future users will not notice when they had to service, i.e. recharge their mobile equipment for the last time.

1.2 Known Solutions and Problems

Although there are numerous products on the market the past years have shown, that the strategy and functionality of mobile devices is not yet decided upon. With an ever advancing technology and recent leaps in the development of highly integrated and energy consumption aware products we will see many new innovative solutions on the market in the near future.

Large research groups have established and an enormous effort is beeing shown towards the development of new strategies, technologies and systems. To mention only a few the BAYOU project and PARCTab at Xerox PARC, Daedalus/BARWAN, Glomop and InfoPad at the University of California at Berkeley, the Rover project at the Massachusetts Institute of Technology and the Project Mandarin are given here. Characteristic for all these efforts is, that they are all approaching the topic of mobility and distributed services and systems in a very general way and are still far from giving a solution. There is still a lot of conceptual work to do in this field.

Today mobile telephones are widely spread. The GSM standard allows to transfer data at a rate up to 9600 Byte/sec and many people are using PC card modems and a labtop computer to have their remote office at hand. Others are using smaller devices, socalled Personal Digital Assistants or PDAs in connection with serial or infrared links to host computers in the office, or a GSM. Recent followups of the GSM like Motorolas Iridium satelite network can offer an evergrowing mobility with

a coverage of almost the whole planet, but no increase in bandwidth with only 2400 bits/s for datatransfers or facsimilies. A new revision of GSM is reported to allow 200kB/s data transfers.

At UC Berkeleys CS department a vertical overlay network. the BARWAN, was implemented using satelite communication, wide and local area packet radio networks as well as infrared communications for in room situations. A proxy service that would adapt the requested data to the available client and link bandwidth was connected with all services on this network.

2

The USRobotics/3COM PalmPilot

2.1 Original PalmPilot

The PalmPilot (figure 2-1) from U.S. Robotics is one of the first truly pocket-size personal organizers designed specifically to extend and enhance the capabilities of your desktop computer and computer network. It has an ample microprocessor and display to let the user use applications alike those that one is used to have on a desktop PC or workstation.

Running a highly efficient operating system on a microprocessor dedicated and optimized for hand-held devices, the PalmPilot gives you instant access to powerful applications – no need to wait for the system to boot or for an application to load into memory. And with a PalmPilot, your organizer and your personal computer are always in sync with each other. The PalmPilot's HotSync (Figure 2-2) technology automatically synchronizes information with a Windows, Macintosh PC or workstation at the touch of a button. You can even synchronize your PalmPilot and desktop PC from a remote location, using a dial-up link or a wide area network connection.

This close coupling of a hand-held organizer and a desktop PC allows the two devices to work in tandem, with the PC taking on the heavy processing and storage chores while the PalmPilot does the light and quick tasks when away from the office. The PalmPilot is the most compact, low-cost, easy-to-use yet powerfull product on the market today.

The software on the PalmPilot can exchange information seamlessly with popular personal information management (PIM) applications, including Microsoft's Schedule+, Lotus' Organizer, and Starfish Software's Sidekick. Optional software also lets the user connect to the enterprise network, send and receive e-mail, and synchronize data remotely.

The PalmPilot features an intuitive graphical interface and a highly accurate text input system called Graffiti (Figure 2-3). Using the PalmPilot's stylus, you can enter alphanumeric information with a slightly adapted handwriting that one can learn in about 15 minutes and thus take notes at a rate of 30 words per minute. Or you can use the PalmPilot's on-screen keyboard or the keyboard on your PC to enter data.

In addition, PalmPilot supports standard development tools that make it easy to add custom applications to suit your organization's information infrastructure and unique work environment. Development tools are available for Unix (gcc) PC and

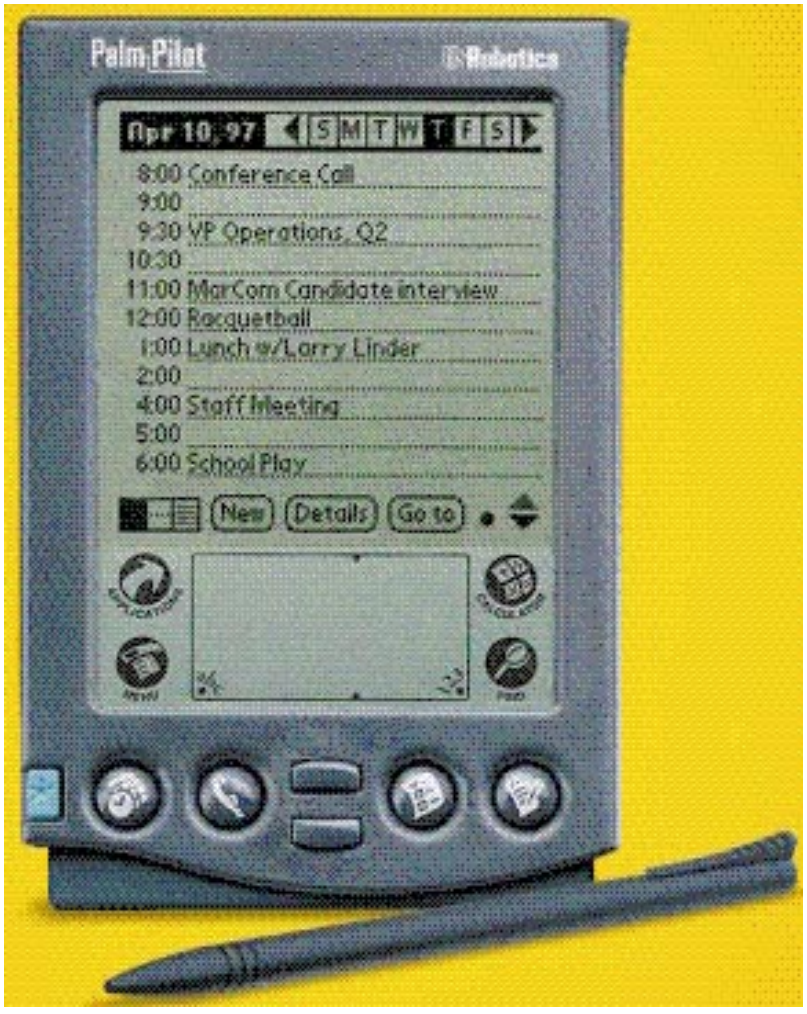


Figure 2-1
The U.S. Robotics/3Com PalmPilot in original size.

Macintosh (Code-Warrior) as well as an on screen emulation with debugging environment. With the device's flexible design, you can expand memory and upgrade functionality easily.

2.2 Applications and Services available Today

There are large numbers of applications available for the PalmPilot today. Among the most common are personal diaries and calendars, address-books and notebook functions. For communication purposes there are various email, usenet news and messaging applications available. A html-browser that uses a “Transsend” proxy on the host system serving the PalmPilot was developed by a group at the UC Berkeley. The application Wingman is available for download and receives rerendered 2-bit graphic images and html code that was scanned by the transsend proxy to fit the capabilities of the application and userinterface. Palmscape 4.1 and Handweb 1.0 are other html browsers currently available for the PalmPilot.

Abundant amounts of shareware software is available in various archives like

Figure 2-2
The HotSync cradle that connects the PalmPilot to your PC or workstation.

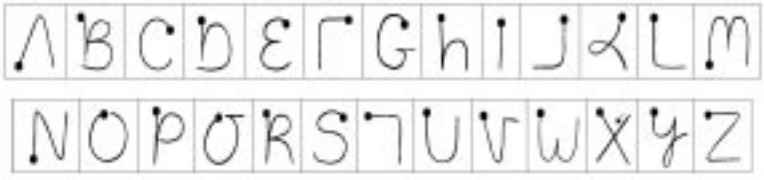


Figure 2-3
The one-stroke-per-letter handwriting recognition system “Graffiti” of the Palmpilot uses this alphabet.

http://www.pilotzone.com or *http://www.pilotgear.com* and numerous private pages and with powerfull software development tools available the commercial softwareproducers are closing in on the market.

An emulation of the PalmPilot is available under the name “Copilot” for INIX, PC and Macintosh computers along with gcc compiler and debugger, specially for the Motorola Dragon Ball processor in the PalmPilot. Metrowerks Codewarrior was developed for PalmPilot together with US Robotics and is available for about $600 for Windows PC and Macintosh computers.

2.3 The PalmPilot in the News

On November 10, 1997 renowned Fortune Magazine [21] wrote:

The Consumerization of Computing Devices

The fastest-selling personal computer device of all time is . . . the PalmPilot. 3Com announced that in November, 1997 it will have sold more than a million of the hand-held devices since April 1996. That's a faster adoption rate than for the first PC, the first Macintosh, or the first laptops. The 5.7-ounce device (which costs $250 to $370) translates handwriting to text; stores your calendar, address book, and other data; and synchronizes with your PC. The Pilot is so hot that Cross sells special PalmPilot stylus-pens, and leathermaker Dooney & Bourke offers tony $50 PalmPilot cases.

The PalmPilot's success is the most visible sign of a new market for specialized microprocessor-powered devices. Such gadgets are less replacements of the PC than complements. Says Charlie Federman: “As computing power moved from the mainframe onto the desktop, so we'll see more computing power in our hands than on our desktops.”

The shift is under way. According to Deutsche Morgan Grenfell, the vast majority – 3.6 billion – of microprocessors sold last year were modest ones known as “embedded processors.” They are the brains in phones, cars, washing machines, and an estimated 14.5 million Tamagotchi virtual pets. That's simplicity for you: Kids who can't read a manual learn to feed and walk a digital dog.

Up next: devices connected wirelessly to networks. AT&T just announced a cell phone that can fish your E-mail off your PC. Chris Shipley, editor of the trend-watching DemoLetter in San Francisco, expects to see home Internet servers that wirelessly control a range of appliances, allowing you to, for instance, regulate your alarm system from your browser at work.

Shipley calls Hewlett-Packard, which has built hand-held devices for years, “a sleeper that will win big.” Sun Microsystems' Diba subsidiary designs Java-based networked gadgets. Microsoft is also a player, with software for

set-top boxes and hand-held devices. Another winner may be Wind River Systems of Alameda, Cal., the top vendor of operating systems and programming tools for embedded processors. Sales for the year ended in January grew 45%, to $64 million. Even the Mars Pathfinder is guided by Wind River software.

– D.K.

2.4 Other PDA's

The Newton Message Pad from Apple is the oldest of the familiy of PDAs with handwriting recognition and sells successfully since about 3 years. It's first versions had problems with the limited computational power and memory. Due to this the handwriting recognition was not very good. This has changed now and the Newton now comes with fast memory and expansion capabilities, software and an adaptable and teachable handwriting recognition system embedded in the operating system.

The Nokia Communicator was the first device on the market that was PDA and mobile telephone integrated into the same device, but due to the high price and the fact that it is more a phone with integrated calendar and notebook function than the other way around.

AT&T Handy and Philips are coming on the market with similar devices. Casio, HP and Psion are known to have calculator type organizers in their catalogs, but are now developing devices with touch screen as well.

The Sharp Phonizer is a brandnew integration of mobile phone and PDA and it looks as if this time it is an aproach more towards a PDA with voice function and not another Nokia Communicator.

Figure 2-4
The Sharp Phonizer, a neat integration of PDA and handy.

Also Sharp and Texas Instruments (TI) set out recently to produce copycat units to the PalmPilot. Sharp's SE-500 and TI's Avigo are currently available for $299 each. Like the PalmPilot, these are pen-based products targeting mobile professionals

who need access to PIM information as well as the ability to synchronize to a PC. The most notable difference from the PalmPilot is the method of data input. The PalmPilot uses Graffiti software to enable handwriting recognition; the SE-500 and the Avigo use onscreen keyboards. The Avigo also uses a text-input system from Tegic Communications called Innovative T9.

The SE-500 and the Avigo share dimensions and weight, but are slightly larger and heavier than the Pilot. Both new units have a larger display with a 240x160 resolution; the Pilot has a 160x160. This shows a clear trend towards higher resolution and possibly color displays in the near future.

3

Transceiver Module Hardware

This chapter should give an overview on how the hardware of the transceiver module was designed and the components were selected. It was the aim to minimize the design in regards to size, chip count, power consumption and cost.

3.1 Overview of the Design Flow

In order to give a brief overview of the design flow of the transceiver module a lineup of all steps taken is given here:

1. Functional design
    - Block diagram
    - Review of functions and programming libraries available
    - Study of PalmPilot
2. Selection of components
    - Functionality of single components
    - Interaction with other components
    - Powerconsumption
    - Availability
    - Single or dual voltage
    - Price/samples
    - Order
3. Schematic of functional elements
    - Check with available libraries

- Development of library elements
- Placement of parts
- Functional interactions
- Logic level conversions (3-5 Volts)
- Analog amplification
- Netlist checks
- Assign footpronts and dimensions for each component

4. Schematic of test circuits
    - Definition of testbusses
    - Selection of testconnections and methods
    - Implementation in schematic
5. Component placement on PCB board
    - Definition of PCB board
    - Definition of layers, vias, pads and other feature sizes
    - Check with available libraries
    - Development of appropriate footprints, pads and library elements
    - Placement according to functional interactions
6. Routing
    - Testroutes with autorouter
    - Testroutes by hand
    - Replacement of parts according to density plot
    - Routing with priorities
    - Routing with different feature sizes
    - Netlist compare
    - Last changes by hand
7. In house made PCB
    - Postscript file
    - Conversion to EPS file
    - Processing of PCB board
    - Drilling
8. Step by step assembly and test

Figure 3-1
Blockdiagram of the identical transceivers on the basestation and mobile unit.

The transceiver systems are made up of identical interfaces that are attached to the serial ports of either the PalmPilot or a networked workstation (Figure 3-1) that serves as a basestation for the mobile units. Except for the casing and powersupply these interfaces are made up of the same components and have the same functionality. It was a major goal to develop these transceivers from standard components for under 200sFr each.

Figure 3-2
The interaction of modem chip and UHF module.

A UHF Transceiver (Figure 3-3) module that handles audio in- and output signals is attached to a modem chip that modulates and demodulates the signal, codes- and decodes the digital information and serves as primary errorcorrection device. The modem chip uses an eight-bit set of four registers for data and control functions as well as a few more control lines and is specially dedicated towards wireless applications. Together with the control lines of the UHF Transceiver these are attached to a microcontroller that works as a more or less “intelligent” serial to parallel interface and buffer. This microcontroller does all the controlfunctions necessary to send and receive bytes via the transmition line modem chip, UHF transceiver and controls the powerdown of the UHF transceiver and shutdown or sleep mode of the other devices (Figure 3-2). A dual powersupply that generates 3.3V and 5V from two AAA batteries, accumulators or external powerjack rounds up the functional blocks of the interface. On the first evaluation board prototype a number of testing jumpers, LEDs and connectors are included.

3.3 Evaluation of the components

The selection of the components was focused to standard components with low power consumption and possibly shutdown or sleep modes to preserve energy on the handheld devices. We tried to minimize package size as well, in order to achieve a compact design in highly integrated surface mount technology.

The availability of small amounts of components as well as the possibility to obtain samples free of charge was taken into account. We encountered numerous difficulties with the distributors around and sometimes had to go to foreign distributors or their european representatives in order to have the components available in such short time.

3.3.1 UHF Transceiver

We selected the BIM-433-F UHF [34] transceiver module (Figure 3-3) from Radiometrix Ltd. since it offered all the necessary functionality and would be available as an European version (433Mhz) as well as an UK version (418Mhz) at a competitive pricing. This band is known as the *ISM-Band* (Industrial, Scientific and Medical Band) that is located at 433,050-434,790 Mhz and is used for household, wireless communication, alarms, controls and indoor microphone applications throughout Europe. In most European countries the frequency of 433,92Mhz$\pm 0,2\%$ is being used at transmitting powers up to 10mW complying to Europes ETS 300-220 standard.

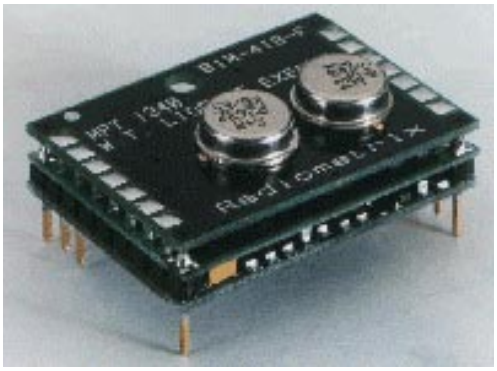


Figure 3-3
The BIM UHF Transceiver module.

There are various chipsets for wireless datacommunication for this band on the market, such as chips form vendors such as RF Micro Devices or Germany based Municom. Many Vendors offer ready made receivers or transmitters that are low power and very small package size but the gap from these single building blocks to full scale wireless modems is very large. These ready made modems sell for approximately $500-1000 and are very bulky and require quite some energy supply.

Radiometrix offers a so called Radio Packet Controller that operates on an 4 Bit data bus with hardware handshake. It is made up of a BIM UHF module and a small microcontroller as well as a few other components. In order to use this module it would have been necessary to use another microcontroller to communicate with the PalmPilots serial interface. This would have been an unnecesary amount of chips. It was close at hand to use the BIM UHF module and attach a custom microcontroller

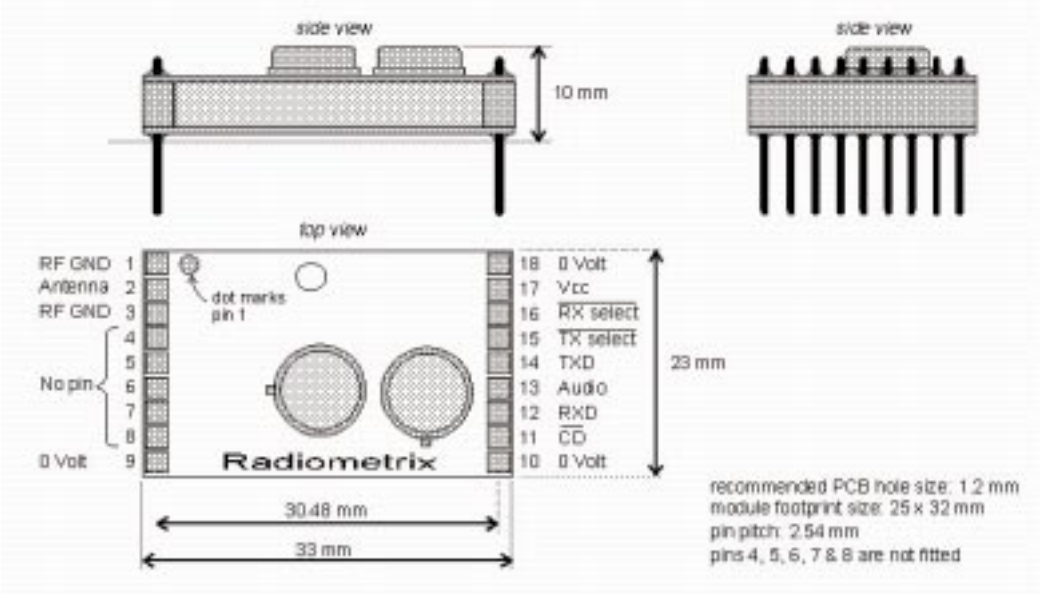


Figure 3-4
The BIM UHF Transceiver module mechanical dimensions and pinout.

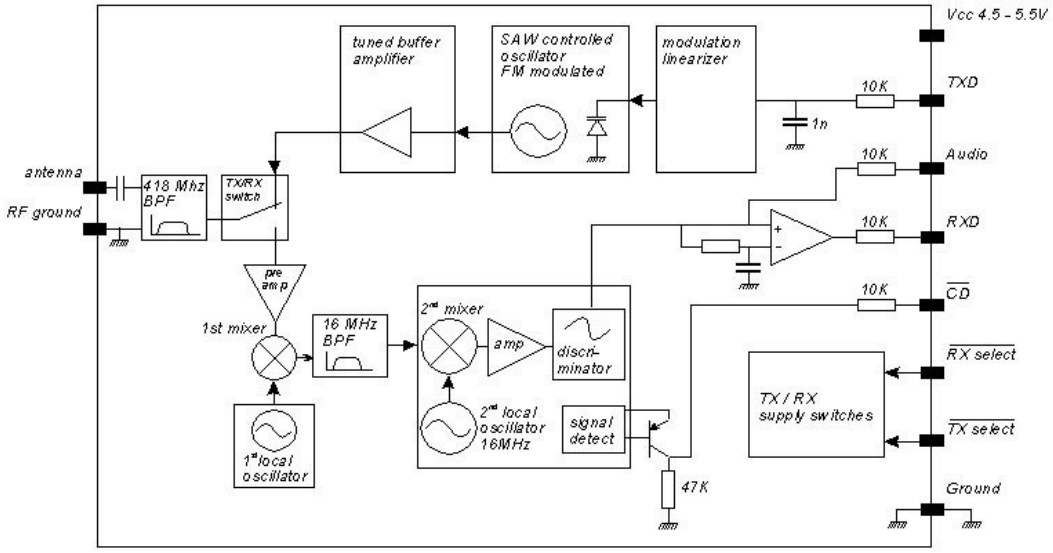


Figure 3-5
The BIM UHF Transceiver module block diagram.

with a serial interface included.

The BIM UHF module sells at $80 for single units and at $48 for amounts of 100+.

<table>
<tr><th>Pin</th><th>Name</th><th>Description</th></tr>
<tr><td>pin 1 & 3</td><td>RF GND</td><td>These pins should be connected to the ground plane against which the integral antenna radiates. Internally connected to pins 9,10,18.</td></tr>
<tr><td>pin 2</td><td>Antenna</td><td>RF input / RF output for connection to an integral antenna. It has a nominal RF impedance of 50W and is capacitively isolated from the internal circuit.</td></tr>
<tr><td>pin 9, 10, 18</td><td>Vss</td><td>0 volt connection for the modulation and supply.</td></tr>
<tr><td>pin 11</td><td>CD</td><td>Carrier Detect - When the receiver is enabled, a low indicates a signal above the detection threshold is being received. The output is high impedance (50kOhm) and should only be used to drive a CMOS logic input.</td></tr>
<tr><td>pin 12</td><td>RXD</td><td>This digital output from the internal data slicer is a squared version of the signal on pin 13 (AF). This signal is used to drive external digital decoders, it is true data (i.e. as fed to the transmitters data input). The 10kOhm output impedance is suitable for driving CMOS logic.
Note: this output contain squared noise when no signal is being received.</td></tr>
<tr><td>pin 13</td><td>RX Audio</td><td>This is the FM demodulator output. It has a standing DC bias of approximately 1.5 Volts and may be used to drive analogue data decoders such as modems or DTMF decoders. Output impedance is 10KOhm. Signal level approx. 0.4V pk to pk. We recommend this signal always be available on a convenient test point for diagnostic purposes.
Note: unlike the RXD output which is always true data, this output is true data on the BiM-418 and inverted on the BiM-433.</td></tr>
<tr><td>pin 14</td><td>TXD</td><td>Should be driven directly by a CMOS logic device running on the same supply voltage as the module. Analogue drive may be used but must not drive this input above Vcc or below 0V. This input should be held at <0.5V when the TX is not selected to prevent current leak (see block diagram).</td></tr>
<tr><td>pin 15</td><td>TX select</td><td>Active low transmit / receive selects with 10kOhm internal.</td></tr>
<tr><td>pin 16</td><td>RX select</td><td>pull-ups. They may be driven by open collector or CMOS logic.</td></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Pin 15 TX</th><th>Pin 16 RX</th><th>Function</th></tr>
<tr><td>1</td><td>1</td><td>power down (<1μA)</td></tr>
<tr><td>1</td><td>0</td><td>receiver enabled</td></tr>
<tr><td>0</td><td>1</td><td>transmitter enabled</td></tr>
<tr><td>0</td><td>0</td><td>self test loop back
Note: loop test is at reduced TX power.</td></tr>
</table>
</td></tr>
<tr><td>pin 17</td><td>Vcc</td><td>positive supply, supply voltages from +4.5V to +5.5V may be used. Reverse polarity will destroy the module. Supply is internally decoupled. Maximum ripple content 50mV pk to pk.</td></tr>
</table>

Table 3-1: Pin assignment and functions on the BIM UHF module.

Part	Description
FX429A	FFSK Modem
FX469	1200/2400/4800 Baud FFSK Modem
FX529	FFSK Modem
FX579	Half Duplex GMSK Modem
FX589	GMSK Modem
FX809	FFSK Modem
FX909A	GMSK Packet Data Modem (mobitex)
FX919A	4-Level FSK Packet Data Modem (old)
FX919B	4-Level FSK Packet Data Modem (new)
FX929B	4-Level FSK Packet Data Modem (RD-LAP)
FX949	Formatted CDPD Modem

Table 3-2: Radio modem chips supplied by CML Microcicuits Ltd.

It features an integrated low power UHF FM transmitter and matching superhet receiver together with data recovery and TX/RX change over circuits and a 1mS power up for power saving (figure 3-5). It's only 23x33mm in size (figure 3-4) and uses a 5 volt power supply with a typical supply current of 12mA in transmit and receive mode. Table 3-1 describes the pins of the module and their function.

3.3.2 Modem Chip

The modem chip was supposed to be used for the packetizing, sending a destination id and error correction of the data on the wireless link. There are various products on the market today that encompass these features in a single package. CML offers a wide variety of radio data modems for different protocols and modulations schown in table 3-2.

The following items were taken into consideration to find the appropriate modem chip:

- Low power consumption
- Possibility to modify protocol
- Error correction and packeting
- Possibility for interrupt and external wakeup
- Low power consumption
- Adaptable clocking and sleep modes
- Dedicated to wireless communication
- Small SMD package
- Price

- Availability

The FX929B 4-Level FSK Modem Data Pump encompasses the following features:

- 4-Level FSK Modulation
- Half-Duplex, 4.8kb/s to 19.2kb/s Operation
- Full Data Packet Framing
- RD-LAP Compatible
- Flexible Operating Modes
- Host Processor Interface
- Low Power 3.3 to 5.5 Volt Operation
- Powersave Option
- 24-Pin Small-Form Package Operation

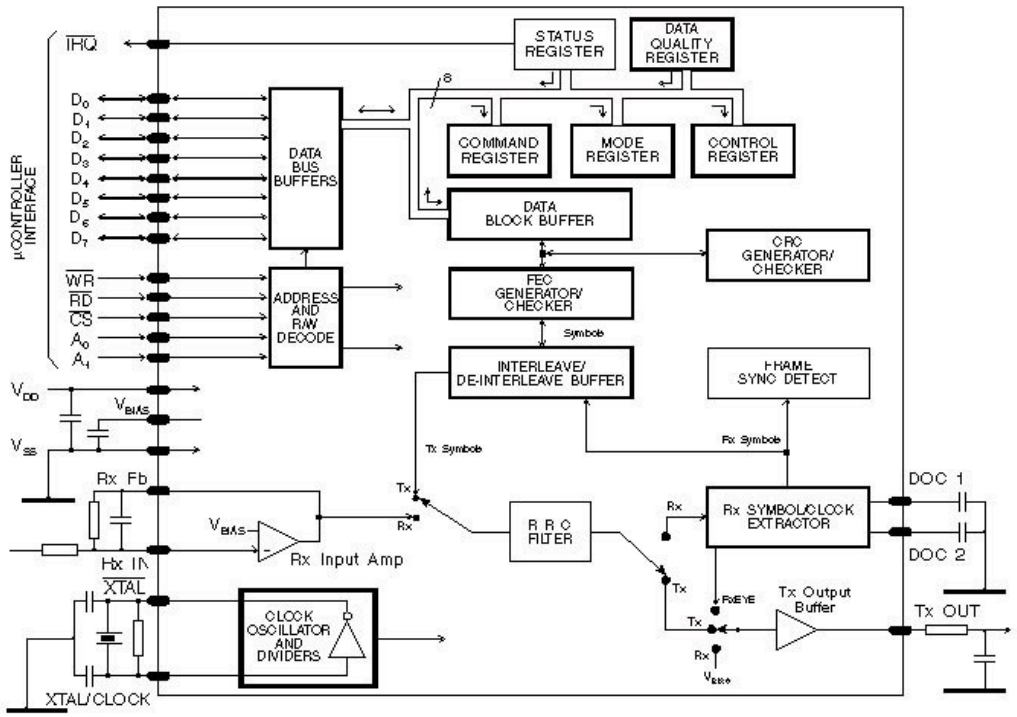


Figure 3-6
The FX929B Modem Data Pump block diagram.

The FX929B is a CMOS integrated circuit that contains all of the baseband signal processing and Medium Access Control (MAC) protocol functions required for a high performance 4-level FSK Wireless Packet Data Modem (figure 3-6).

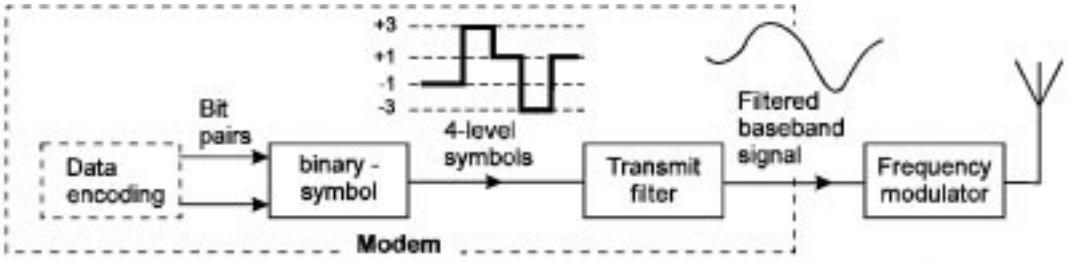


Figure 3-7
The generation of the RRC Filtered 4-Level Tx Baseband Signal on the FX929B.

It interfaces with the modem host processor (figure 3-8) and the radio modulation/demodulation circuits to deliver reliable two-way transfer of the application data over the wireless link. The FX929B assembles application data received from the processor, adds forward error correction (FEC) and error detection (CRC) information and interleaves the result for burst-error protection.

After adding symbol and frame sync codewords, it converts the packet into filtered 4-level analog signals for modulating the radio transmitter (figure 3-6). In receive mode, the FX929B performs the reverse function using the analog signals from the receiver discriminator. After error correction and removal of the packet overhead, the recovered application data is supplied to the processor. Any residual uncorrected errors in the data will be flagged. A readout of the SNR value during receipt of a packet is also provided.

The FX929B uses data block sizes and FEC/CRC algorithms compatible with the RD-LAP over-air standard (figure 4-1). The format used is suitable for other private applications which require the high-speed transfer of data over narrow-band wireless links. The device is programmable to operate at most standard bit-rates from a wide choice of Xtal/clock frequencies.

3.3.3 Microcontroller

Among the vast variety of available microcontrollers the following specifications had to be met:

- Low power consumption
- 3 Volt operation
- 25-30 I/O lines directly accessible
- Serial I/O available
- Possibility for interrupts and external wakeup
- Low power consumption
- Adaptable clocking and sleep modes

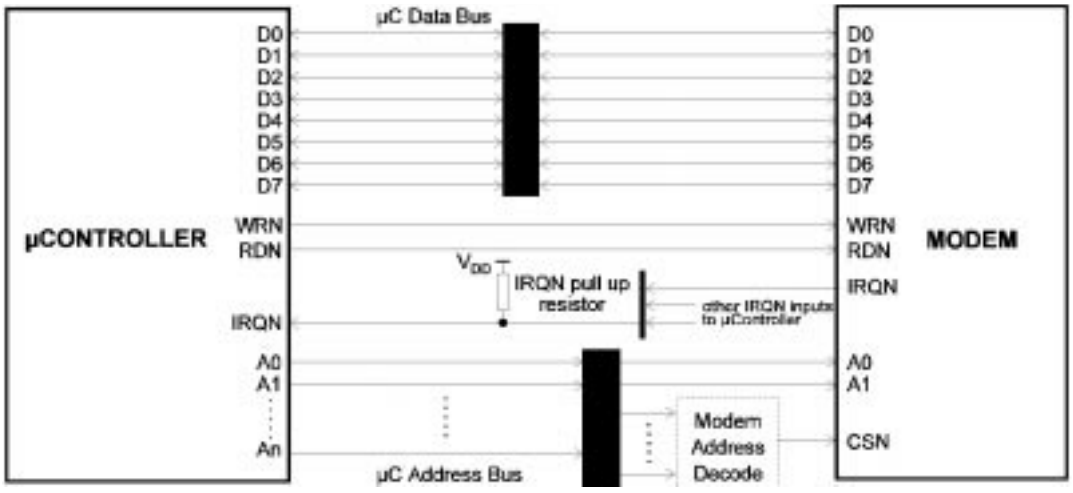


Figure 3-8
The interconnection of modem chip and microcontroller.

- On-chip RAM and ROM
- On-chip EPROM or EEPROM for development purposes
- Small SMD package
- Easy in-circuit programming
- Availability
- Price

Motorolas 68HC11 and 68HC12 [17], Intel, National Semiconductor, Thompson, Hitachi, Scenix SXC 28 [39] and the Microchip PIC [29, 29] series were evaluated. It was soon clear that the amount of microcontrollers with up to 30 directly accessible I/O lines was very limited and that the constaints given by the power consumption and low voltage operation would soon limit the available candidates.

Microchip is a leading supplier of 8-bit microcontrollers, with one of the broadest product offerings. Products range from 8-pin 12 bit instruction word to 68-pin 16-bit instruction word devices. Fast operation, low power and low cost combine to make the Microchip PICmicro family one of the most popular product lines in the world.

The Scenix SCX 28 is a 8-bit in-system programmable microcontroller with 2048x12-bits EE/Flash memory and an operating frequency up to 50 Mhz. It has only 20 I/O pins but features a sophisticated development environment with socalled *Virtual Peripherals*^{TM}, a set of ready made software modules. Due to the amount of I/O pins needed and the comparably high energy consumtion we decided not to use this chip but to concentrate on the PIC family instead. Moreover a performance of 50 MIPS seemed a little too much for our relatively small application.

The Microchip PICmicro 16C6xx familiy (table 3-9) features 22 or 33 I/O pins as well as USART, SCI or I^2C serial I/O, up to 368 Bytes of on-chip RAM, timers as well as an in circuit programming interface.

PIC16C6X Features	61	62	62A	R62	63	R63	64	64A	R64	65	65A	R65	66	67
Program Memory (EPROM) x 14	1K	2K	2K	—	4K	—	2K	2K	—	4K	4K	—	8K	8K
(ROM) x 14	—	—	—	2K	—	4K	—	—	2K	—	—	4K	—	—
Data Memory (Bytes) x 8	36	128	128	128	192	192	128	128	128	192	192	192	368	368
I/O Pins	13	22	22	22	22	22	33	33	33	33	33	33	22	33
Parallel Slave Port	—	—	—	—	—	—	Yes	Yes	Yes	Yes	Yes	Yes	—	Yes
Capture/Compare/PWM Module(s)	—	1	1	1	2	2	1	1	1	2	2	2	2	2
Timer Modules	1	3	3	3	3	3	3	3	3	3	3	3	3	3
Serial Communication	—	SPI/ I^2C	SPI/ I^2C	SPI/ I^2C	SPI/I^2C, USART	SPI/I^2C, USART	SPI/ I^2C	SPI/ I^2C	SPI/ I^2C	SPI/I^2C, USART	SPI/I^2C, USART	SPI/I^2C, USART	SPI/I^2C, USART	SPI/I^2C, USART
In-Circuit Serial Programming	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Brown-out Reset	—	—	Yes	Yes	Yes	Yes	—	Yes	Yes	—	Yes	Yes	Yes	Yes
Interrupt Sources	3	7	7	7	10	10	8	8	8	11	11	11	10	11
Sink/Source Current (mA)	25/20	25/25	25/25	25/25	25/25	25/25	25/25	25/25	25/25	25/25	25/25	25/25	25/25	25/25

Figure 3-9
The PIC 16C6x family features.

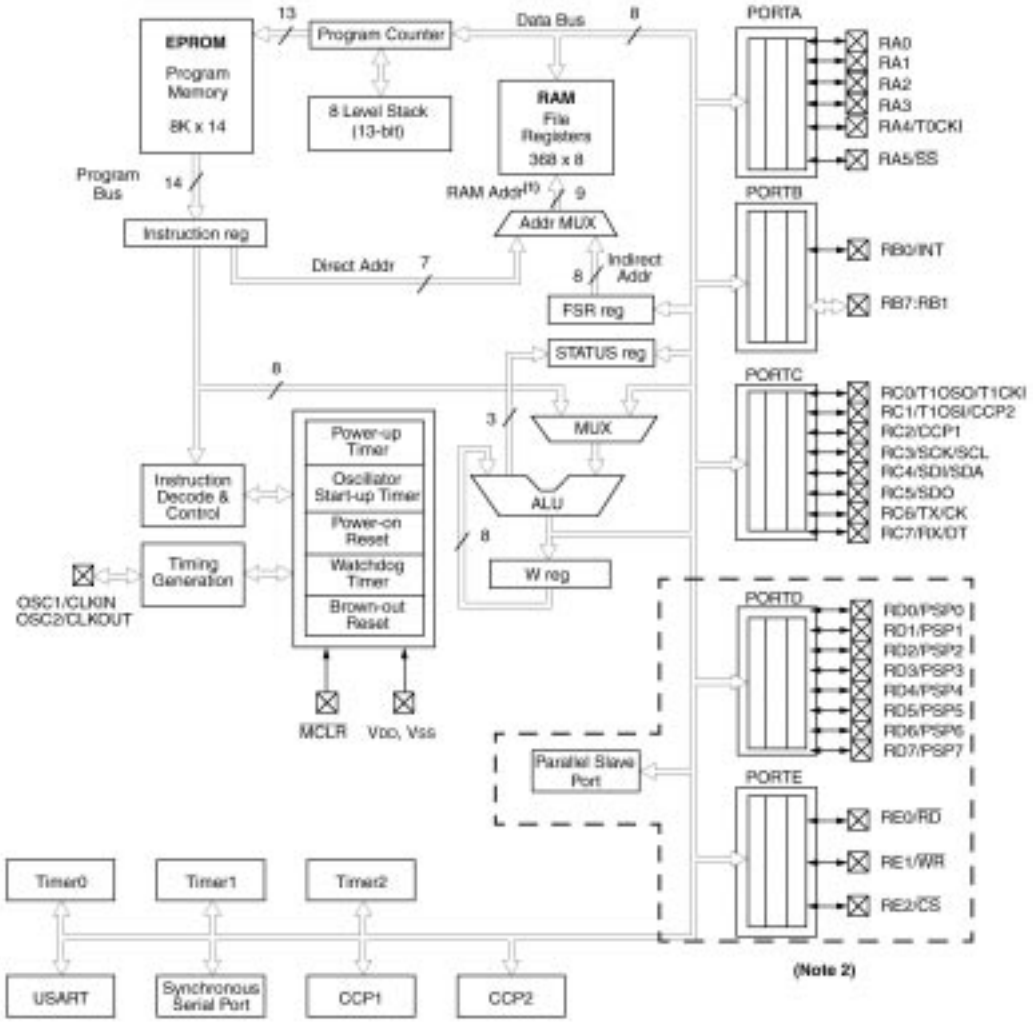
The PIC16C65/65A/R65 devices have 192 bytes of RAM, while the PIC16C67 has 368 bytes. All four devices have 33 I/O pins. In addition, several peripheral features are available, including: three timer/counters, two Capture/Compare/PWM modules and two serial ports. The Synchronous Serial Port can be configured as either a 3-wire Serial Peripheral Interface (SPI) or the two-wire Inter-Integrated Circuit (I^2C) bus. The Universal Synchronous Asynchronous Receiver Transmitter (USART) is also known as a Serial Communications Interface or SCI. An 8-bit Parallel Slave Port is also provided (figure 3-10).

The PIC16C6X device family has special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption.

There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for High Speed crystals. The SLEEP (power-down) mode offers a power saving mode. The user can wake the chip from SLEEP through several external and internal interrupts, and resets. A highly reliable Watchdog Timer with it's own on-chip RC oscillator provides protection against software lock-up.

A UV erasable CERDIP packaged version is ideal for code development, while the cost-effective One-Time-Programmable (OTP) version is suitable for production in any volume.

The MPLAB development environment is available for free download from Microchips server and includes a full programming, testing and debugging environ-



Note 1: Higher order bits are from the STATUS register.
2: PORTD, PORTE and the Parallel Slave Port are not available on the PIC16C66.

Figure 3-10
The PIC 16C67 block diagram.

ment.

Since there were problems obtaining the designated PIC 16C67 we decided to use the PIC 16C77 wich is almost the same chip. The only difference is, that there is an additional AD converter on the chip as well.

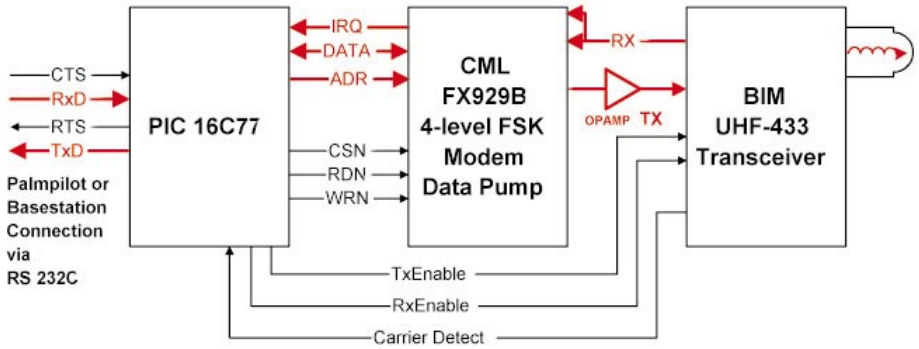


Figure 3-11
The amplifier in the Tx data path.

3.3.4 Operational Amplifier

In order to drive the designated BIM UHF module with the full transmitting power an audio input signal of 4 Volt peak to peak would be necessary. Since the FX929B modem chip can only supply a signal of approximately 1 Volt peak to peak it is necessary to include an amplification in the transmitting signal path (figure 3-11.

This amplification was implemented (figure 3-12) with a *MAX4331 Rail-to-Rail Operational Amplifier* that features a very low shutdown current and has ample amplification power for this application.

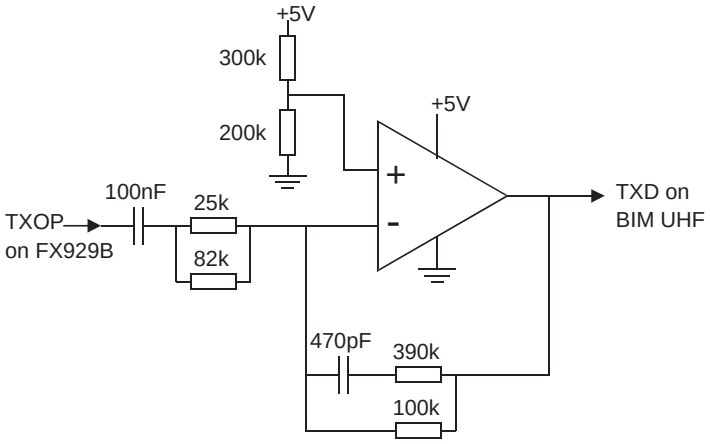


Figure 3-12
The implementation of the TX data amplifier.

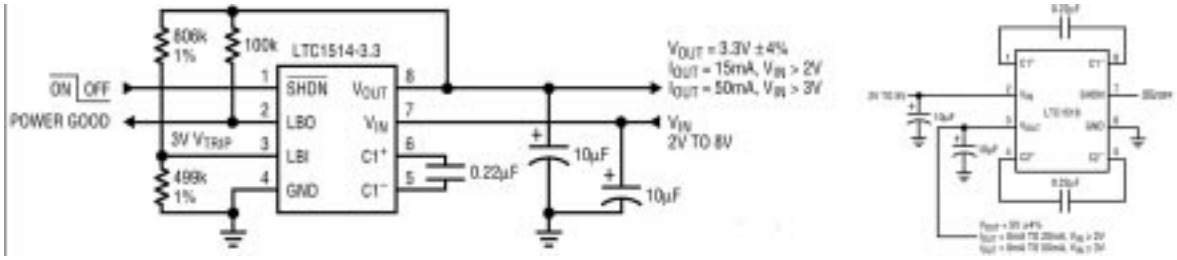


Figure 3-13
The LTC 1514-33 DC/DC converter on the left and the LTC 1516 DC/DC converter on the right.

3.3.5 DC/DC Converter

The UHF transceiver needs a 5 Volt supply. All other components can be driven with 5 Volts too, but the calculation of the powerdissipation (see chapter 3.4) shows that a split powersupply with 3.3 Volts and 5 Volts derived from two AAA battery cells consume less energy.

Thus a dual powersupply was evaluated and the following points were used as guidelines:

- 3.3 and 5 Volts output
- Input range from 2 to 5 Volts for 2 cell AAA-size accumulator use
- High efficiency
- Low sleep current
- Few external components
- No external inductors
- External shutdown

The two Linear Technology DC/DC converters selected (figure 3-13) would allow an input voltage range from 2-5 Volts thus allowing the whole device to be driven by two AAA accumulators of 1.2 Volts or batteries of 1.5 Volts each or a variable DC power supply on the basestation. There are no external inductors needed for these DC/DC convertors and a number of pincompatible devices exist.

3.3.6 RS232 Transceiver

In order to operate the PIC microcontroller on the RS232 Interface it is necessary to transform the signal level from 0-3 Volts to the ±3-25 Volt level specified for RS 232C by EIA. It is important to note thet these transceivers invert the signals. A directly accessed signal would thus be inverted from the RS 232. There are various

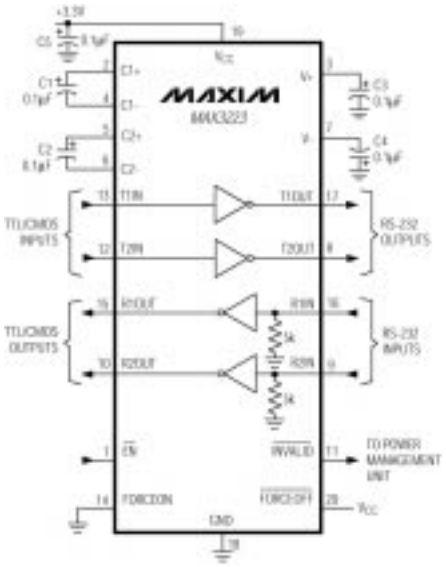


Figure 3-14
The MAX 3223 RS 232 serial transceiver.

devices for this purpose on the market. We concentrated on the minimum device that would allow to use 4 lines of the RS 232 interface, 2 in each direction and that would be dedicated to use in portable equipment. The following items were outlined to be necessary:

- 3 Volt operation
- True RS 232 conversion for 4 lines I/O
- Wakeup on communication request via RS232 lines
- Low power consumption
- Low sleep current
- Few external components
- No external Inductors
- Small SMD packet

The MAX 3223 +3V to +5.5V RS 232 transceiver is a new component that derives from the well established MAX 232 transceiver but features autoshutdown and 1μA sleep current. This solution was preferred to a simpler one with only pull-up resistors and capacitors on the CTS, RxD TxD and RTS signals, even though it is necessary to have more components in the complete design.

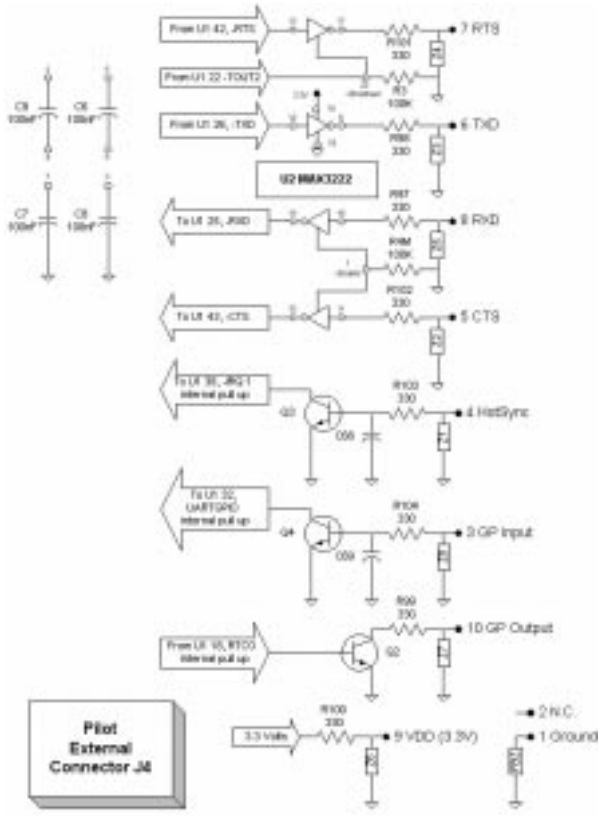


Figure 3-15
The serial interface of the former PalmPilot that features GP Input and GP Output on pins 3,4 and 10.

3.3.7 Other Devices

For the three control lines (Tx select, Rx select and carrier detect) of the BIM UHF module attached to the microcontroller a level conversion from 3.3 Volt to 5 Volt would be necessary. For the first evaluation prototype this feature was included on the print, but the zener diodes were not fitted because this board operates on 5 Volt for the reprogrammable PIC microcontroller.

Since the USRobotics PalmPilot was reengineered shortly it is no longer easily possible to use the extra lines in the serial interface (General Purpose IO) for external output; for example to power down the 3.3V powersupply and thus the microcontroller. In the first versions of the PalmPilot these lines were attached to the microprocessors normal IO ports directly, now only the General Purpose Input and the Hotsync (Pins 3 and 4 on the PalmPilots connector as seen from the bottom) are still connected.

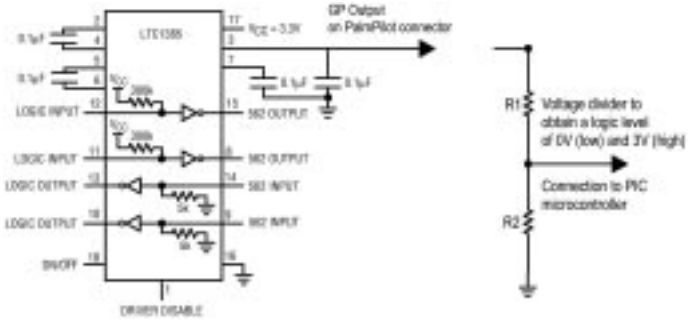


Figure 3-16
The LTC1385 serial transceiver that is now used in the PalmPilot that connects the GP Output to pins 3.

What was formerly the General Purpose Output (GP Output) is now connected to the LTC1385 Transceiver (Pin 3 V^+ output) on the PalmPilots serial interface and serves as Data Terminal Ready Signal on the serial transmition line. This line is high (6 Volt) with the serial transceiver enabled and floating at about 3,3 Volt when disabled since it is attched to the LTC1385. Therefore it was no longer possible to use this line directly on the mobile unit to control the 3,3V power on the transceiver board. We implemented a simple voltage devider with two resistors that made it possible to switch the 3,3V power on and off from the PalmPilot.

This makes it possible to shut the whole transceiver down except for the shutdown current of the two DC/DC convertors and the leakage current of the two resistors on the GP output line.

3.4 Power Consumption and Frequency Modes

It is important to evaluate the power consumption of the whole interface and to decide wether to run the device in a single voltage or dual voltage mode since it is a portable and battery operated device. Evaluations of the market for mobile telephones have shown that the users decision to buy a device is influenced most by the available time of operation, i.e. the battery capacity and energyconsumption.

The energyconsumption was calculated for the most important parts on the transceiver, namely the BIM UHF module, modem chip and PIC microcontroller. On the four right-hand coloumns of table 3-3 the consumption in 5 Volt single voltage operation is given on the left and the consumption of 3.3 and 5 Volt dual voltage operation is given on the right. Since the BIM UHF module is restricted to 5 Volt operation the powerdissipation of this component cannot be altered. A significant reduction by 67% can be achieved by operating modem chip and microcontroller in 3.3 Volts for these two devices only. Since the 5 Volt supply can be shut down seper-

Table 3-3: Evaluation of the powerdissipation of components and frequency modes.

ately without influencing the operation of the other devices a significant reduction in energyconsumption can be achieved by turning the BIM transceiver and its powersupply off, whenever there is no data to transmit on the mobile device.

The bottom section in figure 3-3 shows the amount of power lost due to conversion and efficiency of different DC/DC converters in the dark gray fields. Its results are that the amount of energy consumed for conversion and a second DC/DC convertor is less than the energy consumed in single voltage 5 Volt operation.

A projected powerconsumption of about 80mW in full operation and less than 0.1mA in sleep mode with both the 3.3 Volt and 5 Volt supply switched off would yield an operation time sufficient for most users and applications with a set of two AAA accumulators of about 170mAh each. One strategy to save energy would be to have the mobile unit check with the basestation only every once in a while and not to be in receiving mode all the time. There are numerous others to be still evaluated together with the PalmPilot.

3.5 Schematic and PCB Design

The Transceiver Prototype Board was designed using Protels EDA Client 3.5 with Advanced Schematic, Advanced PCB and Advanced Route. The first contact with the software tools seemed very easy and straightforward but with the project advancing, numerous problems came up. The complexity of the design process made it clear that the right order of the steps taken is of the greatest importance. It was very troublesome to find out about how to do and when to do (or not to do) on the fly, i.e. whenever the specific step would follow. In this learning process a lot of steps had to be redone and it is of vital importance to save the project at different levels as often as possible and to keep strictly to naming conventions, i.e. not to give the same name to different items or labels.

3.5.1 Schematic Libraries

The definition of a new schematic library element was necessary because not all parts used were available in the standard libraries. Here it was necessary to define the amount and type of interconnections and logical layout of the part. A pinout of the library element should be assigned according to functional aspects and not according to the physical pinout. The names given to pins here resemble the function on the component itself.

It is easy to reorder, change, add or delete parts and pins in the schematic library during the design process. It is possible to update the placed elements in the schematic editor with a single command later on.

3.5.2 Schematic Editor

3.5.2.1 General Information

In the schematic editor sheetsize and and dependencies of the different sheets contained in the project should be defined first.

The components are placed on the sheet in functional blocks, it is important not to squezze them together to much but to leave ample free space inbetween. Different sheets can be linked together by assigning ports and net connections.

If connections cannot be placed without crossing other connections they can be made with ports that assign a certain net to a pin but do not have a visible connection. The ground and power nets were distributed in this way.

The footprints can then be assigned directly in the schematic editor.

3.5.2.2 Transceiver Prototype Schematics

For test purposes a large amount of testconnectors was implemented in the schematic, both for logic analyzer with high density connectors and normal testpins.

The PIC 16C77 was included with both the SMD TQFP 44 pin package and the CERDIP 40 pin package on a socket. Only one of them would be used at a time but

Figure 3-17
The transceiver schematic.

The final schematic is included in figure 3-17. Most significant is the lack of all testconnectors and nets and the lack of the second PIC CERDIP package as well as

Figure 3-18
The powersupply with both battery and external supply possibilities.

The power supply is shown in figure 3-18. It can be driven either with two AAA size accumulators or batteries or a 2-5 Volt external supply. For the mobile unit the external power supply connector should be omitted, for the basestation the batteryclips.

For shutdown of the 3.3 Volt DC powersupply with the GP output a voltage divider according to figure 3-16 should be included. The current on the voltage divider cannot be zero with this 'trick' but it is very easy to implement. Maybe it would be worthwile to use a different method or circuitry in order to be able to switch on the 3.3 Volt supply on a mobile transceiver unit. The only reason why 3Com/US Robotics changed the configuration on the PalmPilots connector must be to prohibit others from developing usefull devices to go along with the PalmPilot.

When wires are collected into busses a netlabel should be given to all connections

where the signal is attached. These labels must have different names for every net in the whole circuit and are not characteristic to one bus only. If these netlabels are ommited or mixed the whole circuit design is worthless.

3.5.3 PCB Libraries

In order to make a PCB board all devices used in the schematic must be assigned to a footprint resembling the physical dimensions of the component. For those not contained in the Protel libraries custom footprints had to be developed according to the drawings given in the datasheets.

It is important to assign a geometrical origin to each component, because otherwise they may be placed out of the physical area of the PCB board. There it is not possible to select or move components.

A second important fact is to not assign the same pad and number twice or with the same name. only one footprint can be assigned to each component, so components in parallel like the two types of testconnectors and the two PIC devices on the prototype board must be seperate parts.

It is important to check the orientation of the components.

3.5.4 PCB Editor

With the PCB Editor the PCB board is created. Dimensions of the board, all hole and via sizes, maximum and minimum clearances and widths should be known in advance and be selected according to the specific process that is used to make the PCB board. There is a wizard included in the program that helps a lot with this setup, a change lateron is possible as well as to adapt several other variables to the process used, but in order to change them later it is exteremely important to know your way through the vast amount of configuration possibilities and parameters. Not all parameters are found where one would think at first contact with the software, and there are more parameters that one would think available too. Consulting an exoert helps. A lot of help can be found on Protels Webpage *http://www.protel.com* or the Swiss distributor IDK-Elektronik Software *http://www.idk.ch*.

The schematic is included in form of a netist. All components and all nets are described therein. A visual crosscheck should be performed with the netlist, since it is not visible in the schematic editor, which pins and wires are directly connected and which are not.

The autoplacement option is not very practical for first time users, so a placement of components should be done by hand. It is important to place and orient components so that the connections would be the most direct. In this prototype board it was very difficult to find appropriate positions for the two PIC microcontroller sockets, since the pinout on the two components did not resemble each other. The amount of testcircuitry and connecters used up so much space that it was hardly possible to stay on a single euro-sized board only.

3.5.5 Router

The first trial runs were done by hand, but since it is impossible to protect already routed circuits from being torn up by the Protel router this was nothing more than mere exercise.

The autorouter runs as a seperate program. Priorities can be given to certain nets, but apart from some parameters to be adjusted prior to routing nothing can be influenced on this system. This is an advantage as well as a disadvantage, since a lack of flexibility in the configuration possibilities of the routing software may voids all prior efforts to place components 'intelligently'. Several trial runs should be performed and components replaced according to the density plots given until the design can be completely routed. It is also helpfull to slowly adjust the width of the tracks starting from a very small value to the desired width.

On boards with two layers components can be placed on both sides as well. It is desirable to start with all components on one side and to slowly move some to the other side during the different routing runs. The whole job is very dependand on ones experience and luck too.

3.6 Assembly and Test

The PCB board was made at the in-house lab and holes were drilled according to the diameters given.

All vias were soldered with copper wire through the holes and a visual check was performed along with a check against short circuits on the most important nets. The processing of the PCB seemed to be fairly good, but the soldering of the vias was a very timeconsuming and elaborate work.

The powersupply was fitted first. The test of this part of the device was very straightforward. Only one power connector had to be fitted under the PCB, because it was oriented the wrong way on the layout. The external power for the CERDIP PIC is now fitte dunder the board, the external power for the whole circuit on top.

The socket for the PIC microcontroller as well as some other sockets and connectors where the pins were used as vias had to be fitted along with the crystal oscillator and some capacitors and resistors for the trial of the PIC. Before the microcontroller would be engaged for the first time it was necessary to check all pins for the correct voltage, and it appeared that there were numerous spots, were a pad was soldered to a track, because there was no layer of solder mask available.

The first trial of the PIC was successfull so the next step was a serial connection to a SUN workstattion. The necessary circuits and the MAX3223 transceiver were fitted as well and the first communication was implemented.

In order to test the communication of two BIM UHF modules it was necessary to set up a second board. This took some time, because there were numerous badly soldered connections on vias and some pads had again soldered to the neighboring

tracks. When the second board was up and running we used one BIM UHF module as receiver with a loudspeaker attached and the other as transmitter with a walkmann transmitting an analog audio signal from one to the other. The PIC microcontroller wouls then allow to switch the transmitter and receiver on or off respectedly as well as the onboard 5 Volt power supply.

Next was to fit the modem chip with its oscillator and capacitors and to try to send a full packet. This was implemented and different software versions were tested (see chapter 4.6).

It showed that it would be necessary to amplify the transmitting signal to reach a maximum output on the BIM UHF transmitter. To adapt our circuitry to amplify the signal without disturbing it an external setup of the amplifier was done first (see figure 3-12).

The prototype board was not yet fitted completely, due to the fact that problems occurred with the receiving and transmitting signals.

4

Data Transmission

It was one of our goals to provide a platform for testing different medium layer access schemes. Therefore our module has to provide some sort of flexibility. Flexibility in two ways:

- it should be easily possible to implement new access schemes
- the hardware and especially the modem chip should offer enough general frame and block formatting commands to experiment with

The second can be satisfied by selecting the 'right' modem chip, the first, however, is not easy to achieve using a OTP microcontroller. Let's first have a look at the modem chip we choose and its formatting capabilities.

4.1 FX929B block formatting capabilities

The FX929B's main block types as seen in figure 4-1 are in general used to compose frames conforming to the formatting rules used in RD-LAP systems. A RD-LAP frame consists of a frame preamble (comprising a 24-symbol frame synchronization pattern and station id block) followed by a header block, one or more intermediate blocks and a last block. Channel status (S) symbols are included at regular intervals. The first frame of any transmission is preceded by a symbol synchronization pattern.

Proprietary systems like ours do not have to use the RD-LAP format, but are free to build up their own alternative frame formats, suited best for the particular needs and given S/N ratio. The block structures provided by the FX929B may be used or not.

The FX929B offers the following main block types:

- Station ID

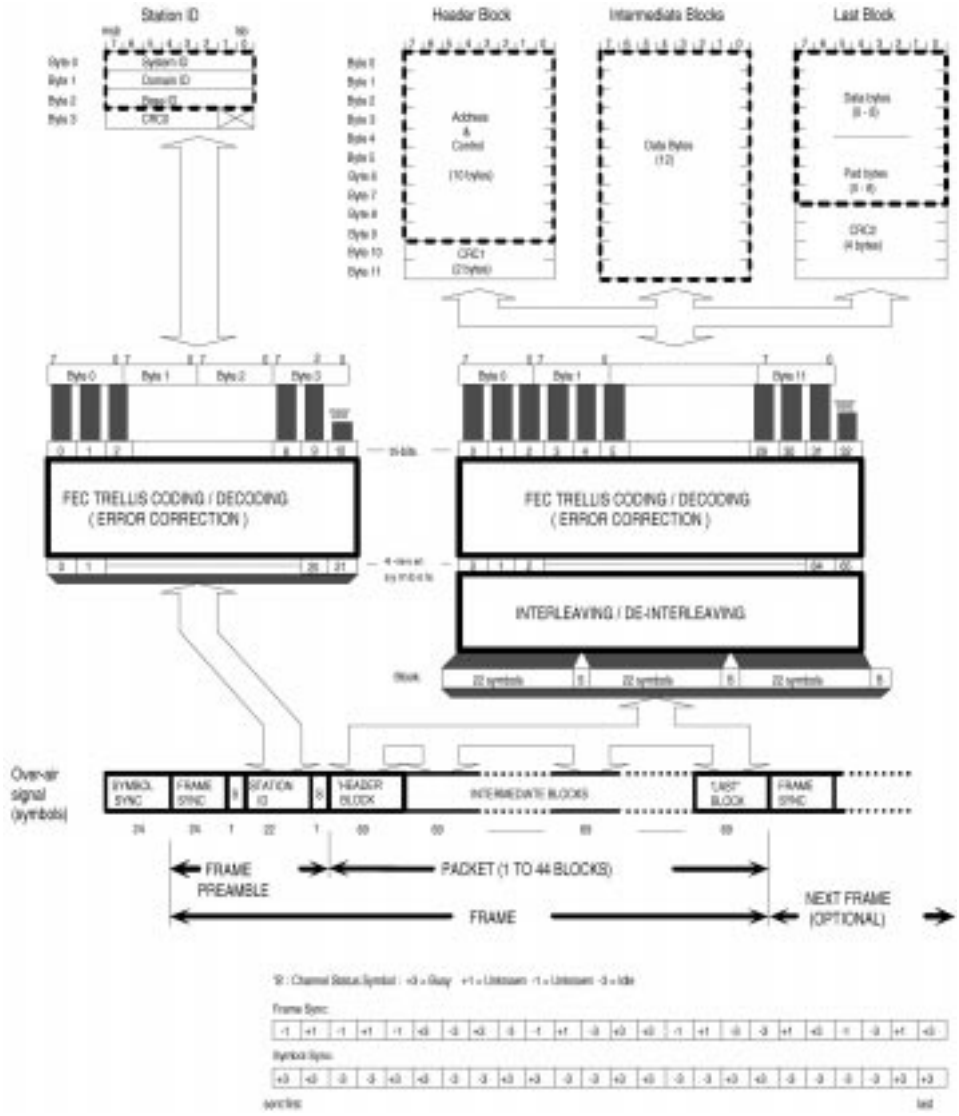


Figure 4-1
The over air data frame format of the FX929B modem chip .

- Header Block
- Intermediate Block
- Last Block

It performs all of the block formatting and de-formatting, the binary data transferred between the modem and the microcontroller being that enclosed by the fat dashed rectangles in figure 4-1.

The Header Block is self-contained in that it includes its own checksum (CRC1), and would normally carry information such as the address of the calling and called parties, the number of following blocks in the frame (if any) and miscellaneous information.

The Intermediate Block(s) contain only data, the checksum at the end of the Last Block (CRC2) also checks the data in any preceding Intermediate Blocks. Checksumerrors are flagged in the status register.

The Transmit Station ID (TSID), Transmit Header Block (THB), Transmit Intermediate Block (TIB) and Transmit Last Block (TLB) tasks are used to transmit these main blocks. Read Station ID (RSID), Read Header Block (RHB), Read Intermediate or Last Block (RILB) are the corresponding block receiving tasks.

Among these the FX929B offers four tasks for transmitting four and 24 symbols at a time: Transmit 24 Symbols (T24S), Transmit 4 Symbols (T4S) and Read 24 Symbols (R24S), Read 4 Symbols (R4S) respectively.

The Search for Frame Preamble (SFP) task can be used to search the incoming symbols for a valid frame synchronization pattern. This task also reads the Station ID following the frame synchronization pattern.

4.2 The PIC's function

What is the PIC microcontroller's job in the game? In a first version the microcontroller was thought only to receive one byte of data from the serial interface and transmit it to the selected modem register. This solution has the big advantage of being flexible. Changes to the protocol can be implemented by just modifying and recompiling the C source code for the SUN or the Pilot. Unfortunately this solution didn't meet the timing requirements of the modem chip. This became obvious very late, when we had already programmed a lot of C code on the SUN. The problems were not the transmission time over the serial link or the processing time in the PIC, as one might expect. In fact the SUN needed to long to send and afterwards receive one byte via `ioctl-calls`.

In the second version we put more intelligence into the PIC. It now receives whole blocks via the serial link and writes them to the modem registers. It too does check the answers from the modem and reports them to the SUN or the Pilot, if necessary. In this version it is possible to send frames without getting in conflict with the timing specification of the modemchip. Receiving however is not yet possible. This version moves these parts of the program to the PIC, which do not need changing. The others among the retransmitbuffers rest on the SUN or the Pilot, i.e. if a block needs to be retransmitted it has to be loaded to the PIC again. The decision which

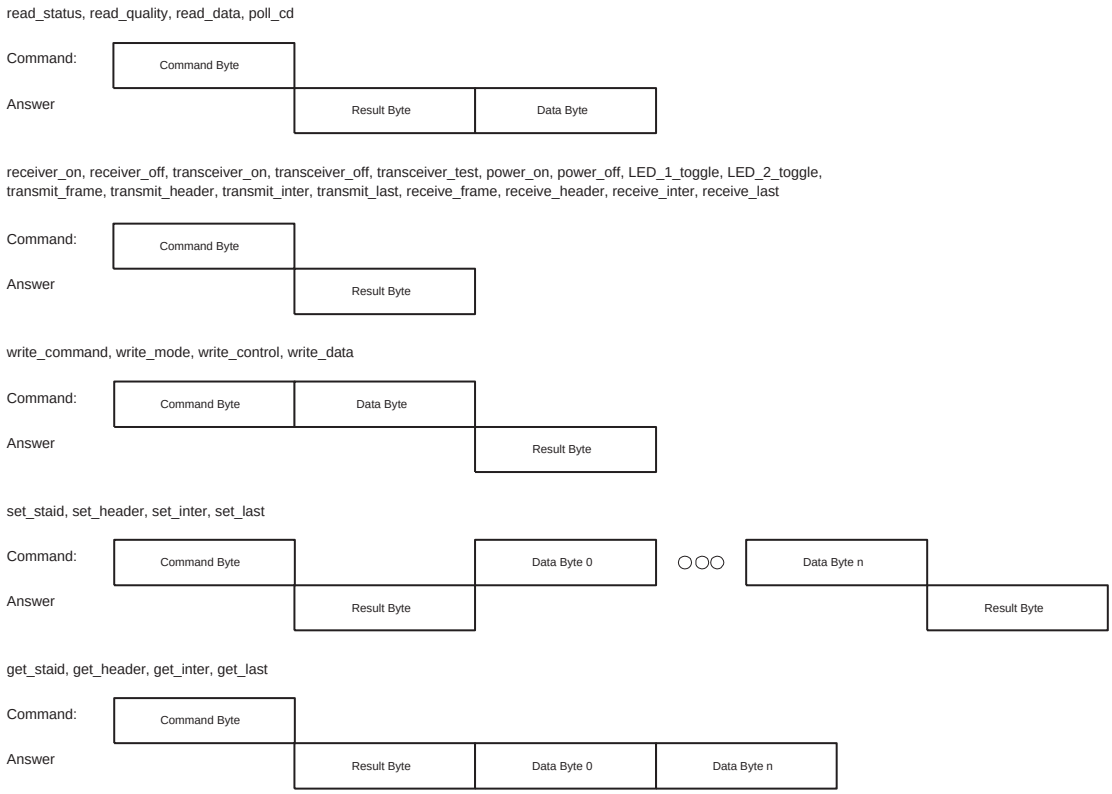


Figure 4-2
Host to PIC to Host communication

blocks need to be sent or retransmitted is made in the SUN or the Pilot and not in the PIC.

4.3 The PIC's command interface

The PIC understands commands for receiving and transmitting blocks, directly accessing the modem registers and controlling the rest of the board (see Table 4-1).

Commands are one byte long and are followed by none, one or more bytes (up to twelve) of data. The PIC answers with one byte followed by none, one or more data bytes and if more then one data byte had to be received from the Pilot, it appends another result byte. Errors are indicated by the result byte. If an error occurs the PIC does not try to recover, but just waits for a new command without completing the one, that failed. Refer to figure 4-2 and figure 4-3

Module Command	Description
read_status read_quality read_data	Read the appropriate modem register. Result is one byte indicating success or error and one byte containing the registers content.
poll_cd	Poll the carrier detect line. Result is one byte indicating success or error and one byte containing no_carrier_detect or carrier_detect.
receiver_on receiver_off transceiver_on transceiver_off transceiver_test	Force the BiM module in the appropriate state. Result is one byte indicating success or error.
power_on power_off	Turn on or off the 5V power supply. Result is one byte indicating success or error.
LED_1_toggle LED_2_toggle	LED it blink. Result is one byte indicating success or error.
write_command write_mode write_control write_data	Write the data byte following the command to the appropriate modem register. Result is one byte indicating success or error.
set_staid set_header set_inter set_last	Write the data bytes following the command to the internal buffer. Result is one byte indicating success or error after receiving the command and another byte indicating success or error after having received all the data bytes. Always check the first result byte before sending the data block. The data block is 3 bytes long forset_staid, 10 for set_header, 12 for set_inter and 8 for set_last.
get_staid get_header get_inter get_last	Read the appropriate internal buffer. Result is one byte indicating success or error followed by up to 12 data bytes.
transmit_frame	Sends a frame structured like shown in figure 4-1.
transmit_header transmit_inte transmit_last	These are thougt to send this block types. They have not been implemented yet.
receive_frame receive_header receive_inter receive_last	These have not been implemented yet.

Table 4-1: Module Commands as they are defined in palm_com.h

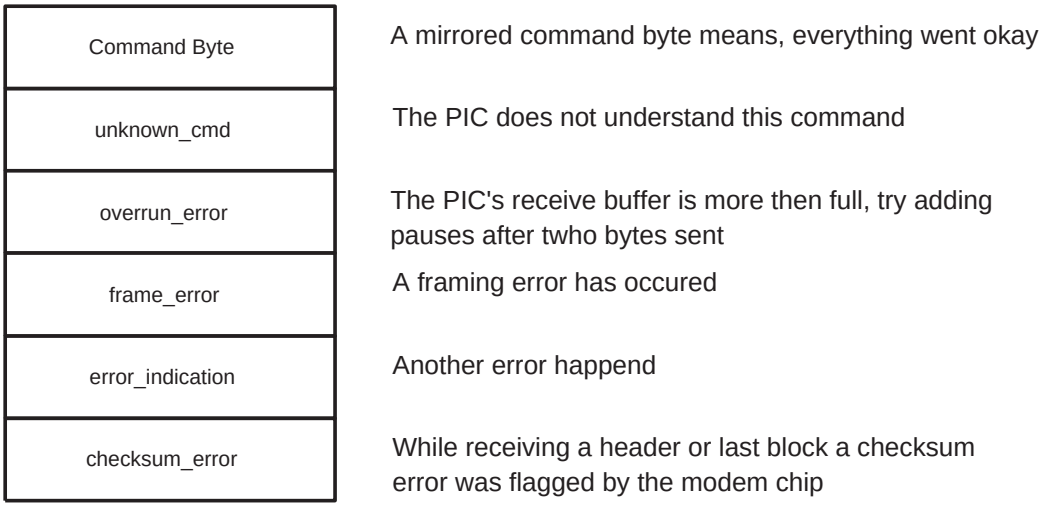


Figure 4-3
Result codes as they are defined in palm_err.h

4.4 Software implementation

The first version was mainly written in C running, but not working (see above), on a SUN SparcStation. Only small parts had to be written in assembler. The second version is hardly written in C, but in assembler. We used the GNU C compiler, which is available for SUN workstations and PalmPilot handhelds. Pilot programs were simulated using xcopilot. The PIC assembler is integrated into MPLAB, a MS Windows based development environment consisting of an editor, an assembler, a debugger, a simulator and optionally an emulator and a programmer. We were able to simulate some basic features in the simulator, but most things could only be tested by trying them out in the real environment, which can be very time consuming.

4.5 Interface to upper layer protocols

We defined an interface for communicating with PalmKiosk II [14] protocols and applications, but didn't have the time to implement it. These functions only exist as small prototypes.

4.6 Software issues

We were most troubled by the following things:

- using the GPIO pins on the Pilot
- getting the serial link to work
- talking to the modem chip fast enough

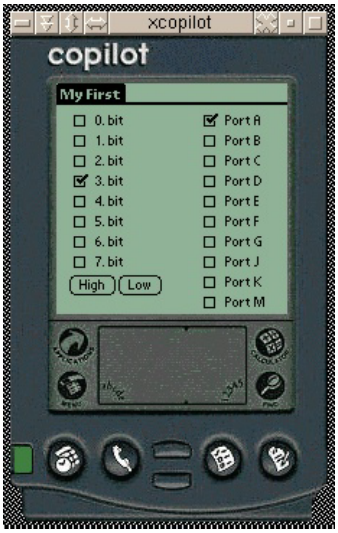


Figure 4-4
The test of the PalmPilots output registers with myfirst.c.

The first thing we wanted to do was turning off the 3.3V power converter via the Pilot's general purpose output pin. Therefor we had to figure out, how to change its state. We checked the Motorola Dragon Ball manual [13] , and were pretty sure it must be connected to pin 18 (RTCOUT/PG7). In order to change this pins state, we had to set and clear some bits in some Dragon Ball registers. In a first approach we did this with external assembler routines. This worked fine for changing the bits, but the connector pin's voltage did not change at all, instead the backlight turn on and off. We decided to write a program for systematically checking the various output registers (see `myfirst.c`, appendix B.1.2). If you plan to try it out make sure you have got a recent backup at hand. This new program no longer uses external assembler routines, but `#defines` using the volatile keyword. This is a very convenient way to read and write registers, and doesn't even need memory space (figure 4-4). After some probing, we were pretty sure that none of the Dragon Balls I/O pins was connected to our GPIO pin. Rolf Sommerhalder decided it was time for some surgery. He opened his Pilot and had a look inside. It turned out that USRobotics had changed the RS232C transceiver. The GPIO pin was no longer connected to the Dragon Ball's pin 18, but to the RS232C transceivers mark voltage output. Looking through the PalmOS' include files revealed the same thing: the name of the connector pin has changed from GPIO to DTR in recent Pilot designs (see chapter 3.3.7 and figure 4-6).

Next thing to do was getting the PIC to talk to our SUN. This is no big thing to do, as Thomas Sailer assured us. All we had to do was to select the baudrate by setting the correct prescale ratio in the Baudrate Generation Register (SPBRG) in the PIC, and enabling the transmitter by clearing the TXEN bit in the transmit

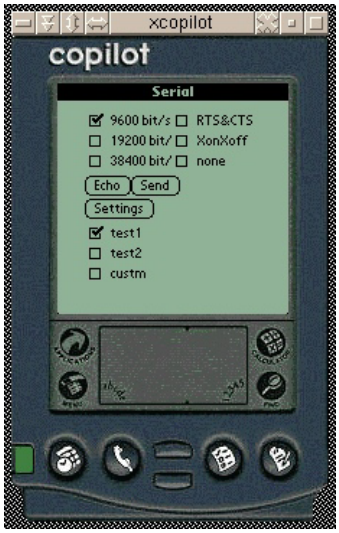


Figure 4-5
The test and configuration of the PalmPilots serial output with serial.c.

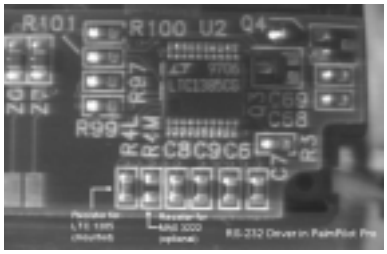


Figure 4-6
A look into the opened PalmPilot with the new GP output connected to an LTC1385.

status and controll register (TXSTA). The manual said a prescale ratio of 22 would be a good choice to get 9'600 bits/s, assuming you use a 3.579 MHz crystal to clock the microcontroller. So we did, and when our PIC was trying to send some useful information to the SUN, it received nothing but garbage. We had a look at the signals on the serial interface and it turned out that the rectangles representing a single bit were much to long for the selected baudrate. After playing around a while we figured out that we had to use a prescale ratio of 16 instead of 22 for the baudrate generation register in order to achieve 9'600 bits/s. We were quite surprised, when we realized that the smaller prescaler values used to select higher baudrates didn't have to be adapted in the same way. After a closer examination of the 'problem' and the prescaler values we recognized that in fact 16 to the base sixteen is the same as 22 to the base ten. The PIC assembler was just not able to guess, that we wanted to have 22 to the base ten and not the base sixteen as it wanted to have it. What we learned: 22 doesn't have to be twenty-two.

On the Pilot we wrote a program to check out the different serial port settings. Getting the Pilot to talk to minicom on the sun was a lot easier, than doing the same with the PIC (see figure 4-5 and appendix B.1.1).

In our first version program we tried to write directly to the modem registers from the SUN. It should take about 540 μs to write one byte and get another byte back (at 38'400 bits/s, including PIC processing time). Our SUN functions (see appendix B.1.11) needed a lot more time. We weren't able to figure out where the time was lost. Perhaps our program got to many signals from the operating system? We tried to ignore all signals, but that didn't help. Sending more then one byte at a time was not possible, without ignoring all the result bytes. But without result bytes we didn't know what the PIC was doing. We decided to move more code to the PIC to enable it to receive and buffer up to twelve bytes. That has the advantage that the *write system call* has to be performed only once for twelve bytes of data, reducing the time for transferring the bytes to the PIC from 120 ms to 10 ms. Implementing the sending procedures directly in the PIC (see appendix B.1.3 and B.1.5) , we were able to send the symbolsync pattern, then the framesync pattern and at the end complete frames like they are shown in figure 4-1.

Receiving frames seemed to be more complex, then sending them. First we weren't able to recognize any of the sent symbols. After debugging the hardware a 'little' bit the symbols looked like an eye pattern and the modemchip was able to distinguish between symbol-, frame-syncing and other symbols. We have not succeeded in decoding any other symbols, yet. What we learned: always place crystals at the PCB boards border (away from anything else, but the chip they are used for), make connections as short as possible and use blocking capacitors wherever possible.

5

Outlook

Where to go on from here?

The work presented here shows, that it is possible to implement a relatively simple wireless network interface for the PalmPilot. Standard and ready made components were used to result in a cost efficient solution. The main components for a single transceiver module sum up to 192,22 sFr as seen in figure 5-1 and this sum can be lowered by about 40% when more modules are being produced.

It would be desirable to make a second PCB board with smaller track widths and smaller vias and possibly an additional solder mask layer over the copper in order to prevent the tracks from attaching to the soldered vias. Also the opamp circuits are now off board and should be fitted directly.

We found it difficult to have both PIC footprints on the same board, because the lengths of different nets would be very long and not transmitt the signal properly anymore, i.e. clock for the PIC.

There are two main directions to follow on the side of the software. First further work should be directed into developing a fair medium layer access control protocol.

Device	**Cost**
BIM UHF 433	117.21 sFr
FX 929B	52.10 sFr
PIC 16C77	14.00 sFr
MAX 3223	4.75 sFr
LTC 1514	2.36 sFr
LTC 1516	1.80 sFr
Total	**193.22 sFr**

Table 5-1: Cost for a single transceiver module.

This includes a systematic investigation of the modem chips capabilities. Maybe a EV9000 Evaluation Kit could be helpful here.

Second all the power saving functions should be tied together to one program to really save as much energy as possible. This includes using the FX929B's `psave-mode` and the PIC's sleep function. The powerconsumption should be measured for the various operating modes. The wakeup from the PalmPilot should be investigated again.

Elaborate testing of the whole system in respect to transmittion power, quality and bandwidth should be performed. For this purpose it would be necessary to attach a transceiver module to a PalmPilot and to make it 3 Volt powered with an OTP version of the PIC microcontroller.

A

Appendix Hardware

Figure A-1
Transceiver Module Prototype Schematic.

Figure A-2
Transceiver Module Prototype PCB.

Figure A-3
Transceiver Module Prototype PCB Top Layer.

Figure A-4
Transceiver Module Prototype PCB Bottom Layer.

Figure A-5
Transceiver Module Schematic.

Figure A-6
Transceiver Module Powersupply Schematic.

Figure A-7
Transceiver Module Prototype Testpins.

B

Appendix Software

B.1 Software

B.1.1 Pilot_serial.c

A program to test the Pilot's serial interface. Shows how to open the serial port on the pilot. Have a look at the function to generate events at the end.

```
#include "Common.h"
#include "System/SysAll.h"
#include "UI/UIAll.h"

#include "pilot_serial.h"
#include "System/SerialMgr.h"

static int StartApplication(void);
static void EventLoop(void);
static void StopApplication(void);
static Boolean serialEvtHandler(EventPtr event);
static void GenEvent(enum events eType, Word controlID);

DWord PilotMain (Word cmd, Ptr cmdPBP, Word launchFlags)
{
   int error;

   if (cmd == sysAppLaunchCmdNormalLaunch) {
     error = StartApplication();
     if (error) {
         return error;
      }
     EventLoop();
     StopApplication();
  }
   return 0;
}

static int StartApplication (void)
{
   FrmGotoForm(formID_serial);
}

static void EventLoop(void)
{
   short err;
   int formID;
   FormPtr form;
   EventType event;

   do {
      EvtGetEvent(&event, evtWaitForever);
      if (! SysHandleEvent(&event)) {
        if (! MenuHandleEvent((void *)0, &event, &err)) {
          if (event.eType == frmLoadEvent) {
            formID = event.data.frmLoad.formID;
            form = FrmInitForm(formID);
            FrmSetActiveForm(form);
            switch (formID) {
            case formID_serial:
              FrmSetEventHandler(form, (FormEventHandlerPtr) serialEvtHandler);
              break;
            }
```

```
          }
        }
      }
      FrmDispatchEvent(&event);
   } while(event.eType != appStopEvent);
}

static Boolean serialEvtHandler(EventPtr event)
{
  static UInt ser_refNum;
  static ULong baud = 9600;
  static Char msg1[6] = "test1";
  static Char msg2[6] = "test2";
  static Char msg3[6] = "custm";
  static Char *msgP = msg1;
  static SerSettingsType ser_settings;
  ULong numBytes;
  Char buffer[20];
  FormPtr form;
  Boolean handled = 0;
  Err error;

  switch (event->eType) {
  case frmOpenEvent:
    form = FrmGetActiveForm();
    FrmDrawForm(form);

/*****************************************************

    error = SysLibFind("Serial Library", &ser_refNum);
    error = SerOpen(ser_refNum, 0, 9600);
    error = SerGetSettings(ser_refNum, &ser_settings);
    ser_settings.flags |= (serSettingsFlagRTSAutoM | serSettingsFlagCTSAutoM);
    error = SerSetSettings(ser_refNum, &ser_settings);

*****************************************************/

    handled = 1;
    break;
  case menuEvent:
    switch (event->data.menu.itemID) {
    case menuitemID_about:
      FrmAlert(alertID_about);
      break;
    case menuitemID_quit:
      GenEvent(appStopEvent, NULL_ID);
      break;
    }
    handled = 1;
    break;
  case ctlSelectEvent:
    switch (event->data.ctlSelect.controlID) {
    case checkID_9600:
      ser_settings.baudRate = 9600;
      GenEvent(ctlSelectEvent, buttonID_settings);
      break;
    case checkID_19200:
      ser_settings.baudRate = 19200;
      GenEvent(ctlSelectEvent, buttonID_settings);
      break;
    case checkID_38400:
      ser_settings.baudRate = 38400;
```

```
      GenEvent(ctlSelectEvent, buttonID_settings);
      break;
    case checkID_pat1:
      msgP = msg1;
      break;
    case checkID_pat2:
      msgP = msg2;
      break;
    case checkID_cust:
      msgP = msg3;
      break;
    case checkID_rtscts:
      ser_settings.flags &= ~serSettingsFlagXonXoffM;
      ser_settings.flags |= (serSettingsFlagRTSAutoM | serSettingsFlagCTSAutoM);
      GenEvent(ctlSelectEvent, buttonID_settings);
      break;
    case checkID_xonxoff:
      ser_settings.flags &= ~(serSettingsFlagRTSAutoM | serSettingsFlagCTSAutoM);
      ser_settings.flags |= serSettingsFlagXonXoffM;
      GenEvent(ctlSelectEvent, buttonID_settings);
      break;
    case checkID_none:
      ser_settings.flags &= ~(serSettingsFlagRTSAutoM | serSettingsFlagCTSAutoM);
      ser_settings.flags &= ~serSettingsFlagXonXoffM;
      GenEvent(ctlSelectEvent, buttonID_settings);
      break;
    case buttonID_send:
      SerSend(ser_refNum, msgP, 6, &error);
      break;
    case buttonID_echo:
      numBytes = 20;
      error = SerReceiveWait(ser_refNum, numBytes, -1);
      if (error == serErrLineErr) {
        error = SerClearErr(ser_refNum);
      }
      else {
        SerReceive(ser_refNum, &buffer, numBytes, 0, &error);
        SerSend(ser_refNum, &buffer, numBytes, &error);
      }
      break;
    case buttonID_settings:
      error = SerSetSettings(ser_refNum, &ser_settings);
      break;
    }
    handled = 1;
    break;
  }
  return handled;
}

static void StopApplication (void)
{
  Err error;
  UInt ser_refNum;

  error = SysLibFind("Serial Library", &ser_refNum);
  error = SerClose(ser_refNum);
  FrmCloseAllForms();
}

/*********************************************************
```

```
static void GenEvent (enum events eType, Word controlID)
{
  EventType newEvent;

  MemSet(&newEvent, sizeof(EventType), 0);
  newEvent.eType = eType;
  if (controlID != NULL_ID) {
    newEvent.data.ctlSelect.controlID = controlID;
  }
  EvtAddEventToQueue(&newEvent);
}

*********************************************************/
```

B.1.2 Myfirst.c

A program to toggle bits in the PalmPilot's output registers. Shows how to write directly to the dragon ball registers.

```
#include "Common.h"
#include "System/SysAll.h"
#include "UI/UIAll.h"

#include "myfirst.h"

#define hwrTD1PortMDockIn 0x80

static void DriveHigh(unsigned char mask, char port);
static void DriveLow(unsigned char mask, char port);
static int StartApplication(void);
static void EventLoop(void);
static void StopApplication(void);
static Boolean myfirstEvtHandler(EventPtr event);

DWord PilotMain (Word cmd, Ptr cmdPBP, Word launchFlags)
{
   int error;

   if (cmd == sysAppLaunchCmdNormalLaunch) {
     error = StartApplication();
     if (error) {
         return error;
      }
     EventLoop();
     StopApplication();
  }
   return 0;
}

static int StartApplication (void)
{
   FrmGotoForm(formID_myfirst);
}

static void EventLoop(void)
{
   short err;
   int formID;
```

```
   FormPtr form;
   EventType event;

   do {
      EvtGetEvent(&event, evtWaitForever);
      if (! SysHandleEvent(&event)) {
        if (! MenuHandleEvent((void *)0, &event, &err)) {
          if (event.eType == frmLoadEvent) {
            formID = event.data.frmLoad.formID;
            form = FrmInitForm(formID);
            FrmSetActiveForm(form);
            switch (formID) {
            case formID_myfirst:
              FrmSetEventHandler(form, (FormEventHandlerPtr) myfirstEvtHandler);
              break;
            }
          }
        }
      }
      FrmDispatchEvent(&event);
   } while(event.eType != appStopEvent);
}

static Boolean myfirstEvtHandler(EventPtr event)
{
  volatile unsigned char *PMDATA = (unsigned char *)0xfffff449;
  volatile unsigned char *PMDIR = (unsigned char *)0xfffff448;
  static unsigned char mask;
  static char port;
  static char high[] = "high";
  static char low[] = "low";
  FormPtr form;
  Boolean handled = 0;
  EventType newEvent;

  switch (event->eType) {
  case frmOpenEvent:
    form = FrmGetActiveForm();
    FrmDrawForm(form);
/*     *PMDIR &= ~hwrTD1PortMDockIn; */
    handled = 1;
    break;
  case ctlSelectEvent:
    switch (event->data.ctlSelect.controlID) {
    case buttonID_high:
        DriveHigh(mask, port);
        break;
    case buttonID_low:
        DriveLow(mask, port);
        break;
    case checkID_bit0:
      mask = 0x01;
      break;
    case checkID_bit1:
      mask = 0x02;
      break;
    case checkID_bit2:
      mask = 0x04;
      break;
    case checkID_bit3:
      mask = 0x08;
      break;
```

```
    case checkID_bit4:
      mask = 0x10;
      break;
    case checkID_bit5:
      mask = 0x20;
      break;
    case checkID_bit6:
      mask = 0x40;
      break;
    case checkID_bit7:
      mask = 0x80;
      break;
    case checkID_portA:
      port = 'A';
      break;
    case checkID_portB:
      port = 'B';
      break;
    case checkID_portC:
      port = 'C';
      break;
    case checkID_portD:
      port = 'D';
      break;
    case checkID_portE:
      port = 'E';
      break;
    case checkID_portF:
      port = 'F';
      break;
    case checkID_portG:
      port = 'G';
      break;
    case checkID_portJ:
      port = 'J';
      break;
    case checkID_portK:
      port = 'K';
      break;
    case checkID_portM:
      port = 'M';
      break;
    }
    handled = 1;
    break;
  case menuEvent:
    switch (event->data.menu.itemID) {
    case menuitemID_about:
      FrmAlert(alertID_about);
      break;
    case menuitemID_quit:
      MemSet(&newEvent, sizeof(EventType), 0);
      newEvent.eType = appStopEvent;
      EvtAddEventToQueue(&newEvent);
      break;
    case menuitemID_poll:
      if (*PMDATA & hwrTD1PortMDockIn) {
/*      GPIO high */
        WinEraseChars(low, StrLen(low), (SWord) 60, (SWord) 20);
        WinDrawChars(high, StrLen(high), (SWord) 60, (SWord) 20);
      }
      else {
```

```
/*      GPIO low */
        WinEraseChars(high, StrLen(high), (SWord) 60, (SWord) 20);
        WinDrawChars(low, StrLen(low), (SWord) 60, (SWord) 20);
      }
      break;
    }
    handled = 1;
    break;
  }
  return handled;
}

static void StopApplication (void)
{
   FrmCloseAllForms();
}

static void DriveHigh(unsigned char mask, char port)
{
  #include "dragonball_registers.h"

  switch (port) {
  case 'A':
    *PASEL |= mask;
    *PADIR |= mask;
    *PADATA |= mask;
    break;
  case 'B':
    *PBSEL |= mask;
    *PBDIR |= mask;
    *PBDATA |= mask;
    break;
  case 'C':
    *PCSEL |= mask;
    *PCDIR |= mask;
    *PCDATA |= mask;
    break;
  case 'D':
    *PDDIR |= mask;
    *PDDATA |= mask;
    *PDPUEN |= mask;
    break;
  case 'E':
    *PESEL |= mask;
    *PEDIR |= mask;
    *PEDATA |= mask;
    *PEPUEN |= mask;
    break;
  case 'F':
    *PFSEL |= mask;
    *PFDIR |= mask;
    *PFDATA |= mask;
    *PFPUEN |= mask;
    break;
  case 'G':
    *PGSEL |= mask;
    *PGDIR |= mask;
    *PGDATA |= mask;
    *PGPUEN |= mask;
    break;
  case 'J':
    *PJSEL |= mask;
```

```
    *PJDIR |= mask;
    *PJDATA |= mask;
    break;
  case 'K':
    *PKSEL |= mask;
    *PKDIR |= mask;
    *PKDATA |= mask;
    *PKPUEN |= mask;
    break;
  case 'M':
    *PMSEL |= mask;
    *PMDIR |= mask;
    *PMDATA |= mask;
    *PMPUEN |= mask;
    break;
  }
}


static void DriveLow(unsigned char mask, char port)
{
  #include "dragonball_registers.h"

  switch (port) {
  case 'A':
    *PASEL &= ~mask;
    *PADIR &= ~mask;
    *PADATA &= ~mask;
    break;
  case 'B':
    *PBSEL &= ~mask;
    *PBDIR &= ~mask;
    *PBDATA &= ~mask;
    break;
  case 'C':
    *PCSEL &= ~mask;
    *PCDIR &= ~mask;
    *PCDATA &= ~mask;
    break;
  case 'D':
    *PDDIR &= ~mask;
    *PDDATA &= ~mask;
    *PDPUEN &= ~mask;
    break;
  case 'E':
    *PESEL &= ~mask;
    *PEDIR &= ~mask;
    *PEDATA &= ~mask;
    *PEPUEN &= ~mask;
    break;
  case 'F':
    *PFSEL &= ~mask;
    *PFDIR &= ~mask;
    *PFDATA &= ~mask;
    *PFPUEN &= ~mask;
    break;
  case 'G':
   /*  *PGSEL &= ~mask; */
/*     *PGDIR &= ~mask; */
    *PGDATA &= ~mask;
 /*    *PGPUEN &= ~mask; */
    break;
```

```
  case 'J':
    *PJSEL &= ~mask;
    *PJDIR &= ~mask;
    *PJDATA &= ~mask;
    break;
  case 'K':
    *PKSEL &= ~mask;
    *PKDIR &= ~mask;
    *PKDATA &= ~mask;
    *PKPUEN &= ~mask;
    break;
  case 'M':
    *PMSEL &= ~mask;
    *PMDIR &= ~mask;
    *PMDATA &= ~mask;
    *PMPUEN &= ~mask;
    break;
  }
}
```

B.1.3 Pilot.asm

Excerpts from the assembler main program and some debugging functions.

```
;************************************
; PALM.ASM
;
; PamlKiosk Wireless Network Module
;
; Running on a PIC16C77
;
;************************************

               LIST P=16C77, R=DEC     ; do not remove R=DEC

               INCLUDE "P16C77.INC"
               INCLUDE <PALM.H>
               INCLUDE <PALM_COM.H>
               INCLUDE <PALM_ERR.H>
               INCLUDE <MODULE.H>
               INCLUDE <PALM_DOS.ASM>
               INCLUDE <PALM_TX.ASM>
               INCLUDE <PALM_RX.ASM>

;*****  most of these were used within the ISR in an older version      *****
ScratchPadRam   EQU     0x20
STATUS_TEMP     EQU     ScratchPadRam+0
PCLATH_TEMP     EQU     ScratchPadRam+1
FSR_TEMP        EQU     ScratchPadRam+2
TempPortB       EQU     ScratchPadRam+3
LastPortB       EQU     ScratchPadRam+4

CountDown       EQU     ScratchPadRam+5 ; used in 'Delay' and send_symb_sync

staid_base      EQU     ScratchPadRam+6
s0              EQU     38              ; these are for the simulator
s1              EQU     39              ; they can be traced in a watch
s2              EQU     40              ; window
```

```
header_base     EQU     ScratchPadRam+9
h0              EQU     41              ; these are for the simulator
h1              EQU     42              ; they can be traced in a watch
h2              EQU     43              ; window
h3              EQU     44
h4              EQU     45
h5              EQU     46
h6              EQU     47
h7              EQU     48
h8              EQU     49
h9              EQU     50

inter_1_base    EQU     ScratchPadRam+19
i0              EQU     51              ; these are for the simulator
i1              EQU     52              ; they can be traced in a watch
i2              EQU     53              ; window
i3              EQU     54
i4              EQU     55
i5              EQU     56
i6              EQU     57
i7              EQU     58
i8              EQU     59
i9              EQU     60
i10             EQU     61
i11             EQU     62

inter_2_base    EQU     ScratchPadRam+31        ; not used yet

last_base       EQU     ScratchPadRam+43
l0              EQU     75              ; these are for the simulator
l1              EQU     76              ; they can be traced in a watch
l2              EQU     77              ; window
l3              EQU     78
l4              EQU     79
l5              EQU     80
l6              EQU     81
l7              EQU     82

CountUp         EQU     ScratchPadRam+51

;*****  the sixteen locations following this address are visible in all banks        *****
ScratchMapped   EQU     0x70
NotUsed         EQU     ScratchMapped+0
SaveWReg        EQU     ScratchMapped+1
RX0             EQU     ScratchMapped+2 ; receive buffers
RX1             EQU     ScratchMapped+3
TX0             EQU     ScratchMapped+4 ; transmit buffers
TX1             EQU     ScratchMapped+5
D0              EQU     ScratchMapped+6 ; data buffer
send_inter      EQU     ScratchMapped+7 ; address of next intermediate
                                        ; block to be sent
load_inter      EQU     ScratchMapped+8 ; address of next buffer to be
                                        ; filled with data from the serial link


                ORG     0
                movlw   HIGH Start
                movwf   PCLATH
                GOTO    Start

                ORG     4
                movlw   HIGH Interrupt
```

```
                movwf   PCLATH
                GOTO    Interrupt

                ORG     50
;*************************************************************
;*      ISR has nothing to do
;*************************************************************
Interrupt       RETFIE


;*************************************************************
;*      main program
;*************************************************************
Start
                movlw   HIGH Init_Ports ; set i/o ports to defaults
                movwf   PCLATH
                CALL    Init_Ports

                clrf    STATUS

                movlw   HIGH do_init_serial ; init the serial communication
                movwf   PCLATH
                call    do_init_serial

                movlw   #inter_1_base   ; move inter_1_buffer's address to
                movwf   send_inter      ; send_inter
                movwf   load_inter      ; and load_inter (we use only one buffer)


error_cont
main_loop

                movlw   HIGH main_loop
                movwf   PCLATH
                goto    main_loop

;*************************************************************
;*      init all I/O ports
;*************************************************************
;*****  init bank 0 first       *****
Init_Ports      CLRF    STATUS

;*****  set ports to defaults   *****
                CLRF    INTCON          ; disable all interrupts
                MOVLW   PORTA_INIT
                MOVWF   PORTA
                MOVLW   PORTB_INIT
                MOVWF   PORTB
                MOVWF   LastPortB
                CLRF    PORTC
                CLRF    PORTD
                MOVLW   PORTE_INIT
                MOVWF   PORTE

;*****  init bank 1 next        *****
                BSF     STATUS, RP0

;*****  set port directions     *****
                CLRF    TRISA
                MOVLW   PORTB_INOUT
                MOVWF   TRISB
                MOVLW   PORTC_INOUT
```

```
                MOVWF   TRISC
                CLRF    TRISD
                CLRF    TRISE
                MOVLW   ADCON1_INIT
                MOVWF   ADCON1
                RETURN

;*************************************************************
;*      set baudrate and enable receiver & transmiter
;*************************************************************
do_init_serial  BSF     STATUS, RP0     ; bank 1
                MOVLW   _38400h
                MOVWF   SPBRG
                BSF     TXSTA, BRGH
                BCF     TXSTA, SYNC
                BCF     STATUS, RP0     ; bank 0
                BSF     RCSTA, SPEN
                BSF     STATUS, RP0     ; bank 1
                BSF     TXSTA, TXEN
                BCF     STATUS, RP0     ; bank 0
                BSF     RCSTA, CREN

                RETURN
;*************************************************************
;*      these routines are for debugging purposes
;*************************************************************
;*****  delay n times 100ms -- n in w_reg       *****
Delay           CLRF    STATUS          ; bank 0
                MOVWF   CountDown
_reset_timer1   MOVLW   0x47
                MOVWF   TMR1L
                MOVLW   0x51
                MOVWF   TMR1H
                BCF     PIR1, TMR1IF
                MOVLW   TIMER1_2
                MOVWF   T1CON
_wait_timer1    movlw   HIGH _wait_timer1
                movwf   PCLATH
                BTFSS   PIR1, TMR1IF
                GOTO    _wait_timer1
                movlw   HIGH _reset_timer1
                movwf   PCLATH
                DECFSZ  CountDown, F
                GOTO    _reset_timer1

                RETURN

;*****  blink the LEDs once     *****
LED_blink       MOVWF   SaveWReg
                SWAPF   STATUS, W
                CLRF    STATUS          ; bank 0
                MOVWF   STATUS_TEMP
                BCF     PORTE, LED_1
                BSF     PORTE, LED_2
                MOVLW   0x01
                movlw   HIGH Delay
                movwf   PCLATH
                CALL    Delay
                CLRF    STATUS          ; bank 0
                BCF     PORTE, LED_2
                BSF     PORTE, LED_1
                MOVLW   0x01
```

```
                movlw   HIGH Delay
                movwf   PCLATH
                CALL    Delay
                SWAPF   STATUS_TEMP, W
                MOVWF   STATUS
                SWAPF   SaveWReg, F
                SWAPF   SaveWReg, W

                RETURN

error_symb      bsf     PORTE, LED_1
                bcf     PORTE, LED_2
                movlw   HIGH error_symb
                movwf   PCLATH
                goto    error_symb

error_transmit_init
                bsf     PORTE, LED_2
                bcf     PORTE, LED_1
                movlw   HIGH error_transmit_init
                movwf   PCLATH
                goto    error_transmit_init

error_receive_init
                bsf     PORTE, LED_2
                bcf     PORTE, LED_1
                movlw   HIGH error_receive_init
                movwf   PCLATH
                goto    error_receive_init

error_staid     bsf     PORTE, LED_2
                bsf     PORTE, LED_1
                movlw   HIGH error_staid
                movwf   PCLATH
                goto    error_staid

error_header    bsf     PORTE, LED_1
                bsf     PORTE, LED_2
_err_head       bcf     PORTE, LED_1
                bcf     PORTE, LED_2
                movlw   HIGH _err_head
                movwf   PCLATH
                goto    _err_head

error_frame     movlw   HIGH error_frame
                movfw   PCLATH
                GOTO    error_frame

error_inter     bsf     PORTE, LED_2
                bsf     PORTE, LED_1
                movlw   HIGH error_inter
                movwf   PCLATH
                goto    error_inter

error_last      bsf     PORTE, LED_2
                bsf     PORTE, LED_1
                movlw   HIGH error_last
                movwf   PCLATH
                goto    error_last

;*****************************************************************
;*      default data
```

```
;*************************************************************
                ORG     0x1E00

pattern_symb    addwf   PCL, F
                nop
                retlw   0x5F
                retlw   0xF5
                retlw   0xF5
                retlw   0xF5
                retlw   0xF5
                retlw   0xF5

pattern_frame   addwf   PCL, F
                nop
                retlw   0x1B
                retlw   0x5B
                retlw   0xF2
                retlw   0x49
                retlw   0x37
                retlw   0x22

                END
```

B.1.4 Palm_dos.asm

Excerpts from palm_dos.asm: functions for talking to the modem and controling the board.

```
                ORG     0x0800

;*************************************************************
;*      decode command
;*************************************************************
;*****  receive one byte first  *****
do_decode_command
                movlw HIGH do_receive_one
                movwf   PCLATH
                call    do_receive_one

;*****  range check     *****
                movf    D0, W
                sublw   last_command
                movlw   HIGH do_unknown_cmd
                movwf   PCLATH
                btfss   STATUS, C
                goto    do_unknown_cmd
                btfsc   STATUS, Z
                goto    do_unknown_cmd

;*****  we got a valid command  *****
                movlw   HIGH 0x0800
                movwf   PCLATH
                movf    D0, W
                addwf   PCL, F
                goto    do_read_status
                goto    do_read_quality
                goto    do_read_data
                goto    do_poll_cd
                goto    do_receiver_on
```

```
                goto    do_receiver_off
                goto    do_transceiver_on
                goto    do_transceiver_off
                goto    do_power_on
                goto    do_power_off
                goto    do_LED_1_toggle
                goto    do_LED_2_toggle
                goto    do_write_command
                goto    do_write_mode
                goto    do_write_control
                goto    do_write_data
                goto    do_set_staid
                goto    do_set_header
                goto    do_set_inter
                goto    do_set_last
                goto    do_get_staid
                goto    do_get_header
                goto    do_get_inter
                goto    do_get_last
                goto    do_transmit_frame
                goto    do_transmit_header
                goto    do_transmit_inter
                goto    do_transmit_last
                goto    do_receive_frame
                goto    do_receive_header
                goto    do_receive_inter
                goto    do_receive_last
                goto    do_transceiver_test

;*************************************************************
;*      receive one byte, check for errors
;*************************************************************
;*****  receive one byte        *****
do_receive_one
                movlw   HIGH do_receive_one
                movwf   PCLATH
                BTFSS   PIR1, RCIF
                GOTO    do_receive_one

; DOLATER WATCHDOG

                movlw   HIGH do_frame_error
                movwf   PCLATH
                BTFSC   RCSTA, FERR
                GOTO    do_frame_error
                movlw   HIGH do_overrun
                movwf   PCLATH
                MOVLW   HIGH do_overrun
                MOVWF   PCLATH
                BTFSC   RCSTA, OERR
                GOTO    do_overrun

                MOVF    RCREG, W
                MOVWF   D0

                RETLW   0x00

;*************************************************************
;*      transmit one byte, if transmit buffer is empty
;*************************************************************
;*****  transfer buffer empty ? *****
do_transmit_one
```

```
                movwf   D0
_transmit_one_txif
                movlw   HIGH _transmit_one_txif
                movwf   PCLATH
                BTFSS   PIR1, TXIF              ; transmit register empty?
                GOTO    _transmit_one_txif      ; no, wait!

; DOLATER WATCHDOG

                movf    D0, W
                movwf   TXREG

                RETLW   0x00

;*************************************************************
;*      transmit one to two bytes, if transmit buffer is empty
;*************************************************************
;*****  transmit two bytes      *****
do_transmit_2
                movlw   HIGH do_transmit_2
                movwf   PCLATH
                BTFSS   PIR1, TXIF      ; transmit register empty?
                GOTO    do_transmit_2   ; no, wait!

; DOLATER WATCHDOG

                MOVF    TX0, W
                MOVWF   TXREG

;*****  transmit one byte       *****
do_transmit_1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                BTFSS   PIR1, TXIF      ; transmit register empty?
                GOTO    do_transmit_1   ; no, wait!

; DOLATER WATCHDOG

                MOVF    TX1, W
                MOVWF   TXREG

                retlw   0x00

;***** same as transmit one byte        *****
do_transmit_error
                movlw   HIGH do_transmit_error
                movwf   PCLATH
                BTFSS   PIR1, TXIF      ; transmit register empty?
                GOTO    do_transmit_error       ; no, wait!

; DOLATER WATCHDOG

                MOVF    TX1, W
                MOVWF   TXREG

                movlw   HIGH error_cont
                movwf   PCLATH
                goto    error_cont

;*************************************************************
;*      handle various errors
;*************************************************************
```

```
;*****  unknown command error   *****
do_unknown_cmd  MOVLW   unknown_cmd     ; send (U)nknown command error
                MOVWF   TX1
                movlw   HIGH do_transmit_error
                movwf   PCLATH
                GOTO    do_transmit_error

;*****  data overrun error      *****
do_overrun      MOVLW   overrun_error   ; send (O)verrun error
                MOVWF   TX1
                movlw   HIGH _reset_cren
                movwf   PCLATH
                GOTO    _reset_cren

;*****  framing error   *****
do_frame_error  MOVLW   frame_error     ; send (F)rame error
                MOVWF   TX1
_reset_cren     BCF     RCSTA, CREN
                BSF     RCSTA, CREN
                movlw   HIGH do_transmit_error
                movwf   PCLATH
                GOTO    do_transmit_error

;*****  indicate error by sending the error flag        *****
do_error        MOVLW   error_indication        ; send (E)rror
                MOVWF   TX1
                movlw   HIGH do_transmit_error
                movwf   PCLATH
                GOTO    do_transmit_error       ; transmit both bytes

;************************************************************
;*      turn on 5V dc-dc convertor
;************************************************************
do_power_on
                BCF     PORTA, SHDN_5V
                MOVLW   power_on
                MOVWF   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                GOTO    do_transmit_1

;************************************************************
;*      turn off 5V dc-dc convertor
;************************************************************
do_power_off
                BSF     PORTA, SHDN_5V
                MOVLW   power_off
                MOVWF   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                GOTO    do_transmit_1

;************************************************************
;*      turn on transceiver & OPAMP
;************************************************************
do_transceiver_on
                BCF     PORTB, TXselectN
                bsf     PORTB, RXselectN
                bcf     PORTE, SHDN_AMP
                MOVLW   transceiver_on
                MOVWF   TX1
                movlw   HIGH do_transmit_1
```

```
                movwf   PCLATH
                GOTO    do_transmit_1

;*************************************************************
;*      turn off transceiver & OPAMP
;*************************************************************
do_transceiver_off
                BSF     PORTB, TXselectN
                bsf     PORTB, RXselectN
                bsf     PORTE, SHDN_AMP
                MOVLW   transceiver_off
                MOVWF   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                GOTO    do_transmit_1

;*************************************************************
;*      turn on receiver
;*************************************************************
do_receiver_on
                BCF     PORTB, RXselectN
                bsf     PORTB, TXselectN
                MOVLW   receiver_on
                MOVWF   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                GOTO    do_transmit_1

;*************************************************************
;*      turn off receiver
;*************************************************************
do_receiver_off
                BSF     PORTB, RXselectN
                bsf     PORTB, TXselectN
                MOVLW   receiver_off
                MOVWF   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                GOTO    do_transmit_1

;*************************************************************
;*      transceiver test
;*************************************************************
do_transceiver_test
                bcf     PORTB, RXselectN
                bcf     PORTB, TXselectN
                movlw   transceiver_test
                movwf   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                goto    do_transmit_1

;*************************************************************
;*      toggle LED 1
;*************************************************************
do_LED_1_toggle
                MOVF    PORTE, W
                XORLW   LED_1_mask
                MOVWF   PORTE
                MOVLW   LED_1_toggle
                MOVWF   TX1
                movlw   HIGH do_transmit_1
```

```
                movwf   PCLATH
                GOTO    do_transmit_1

;*************************************************************
;*      toggle LED 2
;*************************************************************
do_LED_2_toggle
                MOVF    PORTE, W
                XORLW   LED_2_mask
                MOVWF   PORTE
                MOVLW   LED_2_toggle
                MOVWF   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                GOTO    do_transmit_1

;*************************************************************
;*      poll the carrier detect line
;*************************************************************
do_poll_cd
                movlw   no_carrier_detect
                btfss   PORTB, CDN
                movlw   carrier_detect
                movwf   TX1
                movlw   poll_cd
                movwf   TX0
                movlw   HIGH do_transmit_2
                movwf   PCLATH
                goto    do_transmit_2

;*************************************************************
;*      read modem status register
;*************************************************************
do_read_status
                MOVLW   PORTD_IN
                BSF     STATUS, RP0
                MOVWF   TRISD
                BCF     STATUS, RP0
                MOVF    PORTA, W
                IORLW   status_or
                ANDLW   status_and
                MOVWF   PORTA

                MOVF    PORTD, W
                MOVWF   TX1
                MOVF    PORTA, W
                IORLW   idle_or
                MOVWF   PORTA

                movlw   read_status
                movwf   TX0
                movlw   HIGH do_transmit_2
                movwf   PCLATH
                goto    do_transmit_2


;*************************************************************
;*      write modem command register
;*************************************************************
do_write_command
                movlw   HIGH do_receive_one
                movwf   PCLATH
```

```
                call    do_receive_one
                MOVF    PORTA, W
                IORLW   command_or
                ANDLW   command_and
                MOVWF   PORTA

                MOVF    D0, W
                MOVWF   PORTD
                MOVLW   PORTD_OUT
                BSF     STATUS, RP0
                MOVWF   TRISD
                BCF     STATUS, RP0

                MOVF    PORTA, W
                IORLW   idle_or
                MOVWF   PORTA

                movlw   write_command
                movwf   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                goto    do_transmit_1


;*************************************************************
;*      set the station id
;*************************************************************
do_set_staid
                movlw   set_staid
                movwf   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                call    do_transmit_1

                movlw   0x03
                movwf   CountDown
                movlw   #staid_base
                movwf   FSR
_staid_set_count
                movlw   HIGH do_receive_one
                movwf   PCLATH
                call    do_receive_one
                movwf   INDF
                incf    FSR, F
                movlw   HIGH _staid_set_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _staid_set_count

                movlw   set_staid
                movwf   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                goto    do_transmit_1


;*************************************************************
;*      set iblock
;*************************************************************
do_set_inter
                movlw   set_inter
                movwf   TX1
```

```
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                call    do_transmit_1

                movlw   0x0C
                movwf   CountDown
                movf    load_inter, W
                movwf   FSR
_inter_set_count
                movlw   HIGH do_receive_one
                movwf   PCLATH
                call    do_receive_one
                movwf   INDF
                incf    FSR, F
                movlw   HIGH _inter_set_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _inter_set_count

                movlw   set_inter
                movwf   TX1
                movlw   HIGH do_transmit_1
                movwf   PCLATH
                goto    do_transmit_1

;*************************************************************
;*      get iblock
;*************************************************************
do_get_inter
                movlw   HIGH do_transmit_one
                movwf   PCLATH
                movlw   get_inter
                call    do_transmit_one

                movlw   0x0C
                movwf   CountDown
                movf    load_inter, W
                movwf   FSR
_inter_get_count
                movlw   HIGH do_transmit_one
                movwf   PCLATH
                movf    INDF, W
                call    do_transmit_one
                incf    FSR, F
                movlw   HIGH _inter_get_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _inter_get_count

                RETLW   0x00


;*************************************************************
;*      get staid
;*************************************************************
do_get_staid
                movlw   HIGH do_transmit_one
                movwf   PCLATH
                movlw   get_staid
                call    do_transmit_one

                movlw   0x03
```

```
                movwf   CountDown
                movlw   #staid_base
                movwf   FSR
_staid_get_count
                movlw   HIGH do_transmit_one
                movwf   PCLATH
                movf    INDF, W
                call    do_transmit_one
                incf    FSR, F
                movlw   HIGH _staid_get_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _staid_get_count

                RETLW   0x00

;*************************************************************
;*      transmit frame
;*************************************************************
do_transmit_frame
                movlw   HIGH do_transmit_init
                movwf   PCLATH
                call    do_transmit_init

                movlw   HIGH do_send_frame_one
                movwf   PCLATH
                call    do_send_frame_one
                movwf   D0
                movlw   HIGH do_error
                movwf   PCLATH
                btfsc   D0, 0
                goto    do_error
                movlw   HIGH do_transmit_one
                movwf   PCLATH
                movlw   transmit_frame
                call    do_transmit_one

                RETLW   0x00

;*************************************************************
;*      receive frame
;*************************************************************
do_receive_frame
                RETLW   0x00
```

B.1.5 Palm_tx.asm

Excerpts from palm_tx.asm: functions for sending blocks.

```
                ORG     0x1000

;*************************************************************
;*      init the modem for transmiting
;*************************************************************
do_transmit_init
                clrf    STATUS
                movlw   ctrl_reg_transmit
                movwf   D0
                movlw   HIGH do_write_control_private
```

```
                movwf   PCLATH
                call    do_write_control_private

                movlw   mode_reg_transmit
                movwf   D0
                movlw   HIGH do_write_mode_private
                movwf   PCLATH
                call    do_write_mode_private

                movlw   RESET
                movwf   D0
                movlw   HIGH do_write_command_private
                movwf   PCLATH
                call    do_write_command_private

                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private
                movlw   HIGH error_transmit_init
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_transmit_init     ; error BFREE not set
                movlw   (mode_reg_transmit | IRQNEN)
                movwf   D0
                movlw   HIGH do_write_mode_private
                movwf   PCLATH
                call    do_write_mode_private

                RETURN
;*************************************************************
;*      send frame, containing one iblock
;*************************************************************
do_send_frame_one
                movlw   HIGH do_send_symb_sync
                movwf   PCLATH
                call    do_send_symb_sync

wait_symb_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private

                movlw   HIGH wait_symb_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_symb_irq   ; IRQ not set

                movlw   HIGH error_symb
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_symb      ; BFREE not set

                movlw   HIGH error_symb
                movwf   PCLATH
                movf    D0, W
                andlw   IBEMPTY
```

```
                btfss   STATUS, Z
                goto    error_symb      ; IBEMPTY not clear

                movlw   HIGH do_send_frame_sync
                movwf   PCLATH
                call    do_send_frame_sync

wait_frame_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private

                movlw   HIGH wait_frame_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_frame_irq  ; IRQ not set

                movlw   HIGH error_frame
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_frame     ; BFREE not set

                movlw   HIGH error_frame
                movwf   PCLATH
                movf    D0, W
                andlw   IBEMPTY
                btfss   STATUS, Z
                goto    error_frame     ; IBEMPTY not clear

                movlw   HIGH do_send_staid
                movwf   PCLATH
                call    do_send_staid

wait_staid_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private

                movlw   HIGH wait_staid_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_staid_irq          ; IRQ not set

                movlw   HIGH error_staid
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_staid     ; BFREE not set

                movlw   HIGH error_staid
                movwf   PCLATH
                movf    D0, W
                andlw   IBEMPTY
                btfss   STATUS, Z
                goto    error_staid     ; IBEMPTY not clear
```

```
                movlw   HIGH do_send_header
                movwf   PCLATH
                call    do_send_header

wait_header_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private

                movlw   HIGH wait_header_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_header_irq         ; IRQ not set

                movlw   HIGH error_header
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_header    ; BFREE not set

                movlw   HIGH error_header
                movwf   PCLATH
                movf    D0, W
                andlw   IBEMPTY
                btfss   STATUS, Z
                goto    error_header    ; IBEMPTY not clear

                movlw   HIGH do_send_inter
                movwf   PCLATH
                call    do_send_inter

wait_inter_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private

                movlw   HIGH wait_inter_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_inter_irq  ; IRQ not set

                movlw   HIGH error_inter
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_inter     ; BFREE not set

                movlw   HIGH error_inter
                movwf   PCLATH
                movf    D0, W
                andlw   IBEMPTY
                btfss   STATUS, Z
                goto    error_inter     ; IBEMPTY not clear

                movlw   HIGH do_send_last
```

```
                movwf   PCLATH
                call    do_send_last

wait_last_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private

                movlw   HIGH wait_last_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_last_irq   ; IRQ not set

                movlw   HIGH error_last
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_last      ; BFREE not set

                movlw   HIGH error_last
                movwf   PCLATH
                movf    D0, W
                andlw   IBEMPTY
                btfss   STATUS, Z
                NOP
                RETLW   0x00


;*************************************************************
;*      send symbol syncing pattern
;*************************************************************
do_send_symb_sync
                movlw   0x06
                movwf   CountDown

_symb_count
                movlw   HIGH pattern_symb
                movwf   PCLATH
                movf    CountDown, W
                call    pattern_symb
                movwf   D0
                movlw   HIGH do_write_data_private
                movwf   PCLATH
                call    do_write_data_private
                movlw   HIGH _symb_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _symb_count
                movlw   T24S
                movwf   D0
                movlw   HIGH do_write_command_private
                movwf   PCLATH
                call    do_write_command_private

                RETLW   0x00

;*************************************************************
;*      send last block
;*************************************************************
```

```
do_send_last
                movlw   0x08
                movwf   CountDown
                movlw   #last_base
                movwf   FSR
_last_count
                movf    INDF, W
                movwf   D0
                incf    FSR, F
                movlw   HIGH do_write_data_private
                movwf   PCLATH
                call    do_write_data_private
                movlw   HIGH _last_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _last_count
                movlw   TLB
                movwf   D0
                movlw   HIGH do_write_command_private
                movwf   PCLATH
                call    do_write_command_private

                RETLW   0x00

do_transmit_staid       RETURN
do_transmit_header      RETURN
do_transmit_inter       RETURN
do_transmit_last        RETURN
do_receive_staid        RETURN
do_receive_header       RETURN
do_receive_inter        RETURN
do_receive_last RETURN
```

B.1.6 Palm_rx.asm

Excerpts from palm_rx.asm: functions for reading blocks.

```
                ORG     1800

;*************************************************************
;*      init the modem for receiving
;*************************************************************
do_receive_init
                clrf    STATUS
                movlw   (ctrl_reg_receive)
                movwf   D0
                movlw   HIGH do_write_control_private
                movwf   PCLATH
                call    do_write_control_private

                movlw   mode_reg_receive
                movwf   D0
                movlw   HIGH do_write_mode_private
                movwf   PCLATH
                call    do_write_mode_private

                movlw   RESET
                movwf   D0
                movlw   HIGH do_write_command_private
```

```
                movwf   PCLATH
                call    do_write_command_private

                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private
                movlw   HIGH error_receive_init
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_receive_init      ; error BFREE not set

                RETURN

;*************************************************************
;*      search for frame pattern and read station id
;*************************************************************
do_read_staid
                movlw   (mode_reg_receive | IRQNEN)
                movwf   D0
                movlw   HIGH do_write_mode_private
                movwf   PCLATH
                call    do_write_mode_private

                movlw   HIGH do_write_command_private
                movwf   PCLATH
                movlw   (SFP | AQLEV | AQSC)
                movwf   D0
                call    do_write_command_private

wait_sfp_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private
                movlw   HIGH wait_sfp_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_sfp_irq    ; IRQ not set

                movlw   HIGH error_staid
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_staid     ; BFREE not set

                movlw   HIGH error_symb
                movwf   PCLATH
                movf    D0, W
                andlw   DIBOVF
                btfss   STATUS, Z
                goto    error_symb      ; DIBOVF not clear

                movlw   HIGH error_symb
                movwf   PCLATH
                movf    D0, W
                andlw   CRCERR
                btfss   STATUS, Z
                goto    error_symb      ; CRCERR not clear
```

```
                movlw   0x03
                movwf   CountDown
                movlw   #staid_base
                movwf   FSR
_staid_read_count
                movlw   HIGH do_read_data_private
                movwf   PCLATH
                call    do_read_data_private
                movf    D0, W
                movwf   INDF
                incf    FSR, F
                movlw   HIGH _staid_read_count
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _staid_read_count

                RETLW   0x00

;*************************************************************
;*      search frame sync; do not check crc; decode staid
;*************************************************************
do_search_frame
                movlw   (mode_reg_receive | IRQNEN)
                movwf   D0
                movlw   HIGH do_write_mode_private
                movwf   PCLATH
                call    do_write_mode_private

                movlw   HIGH do_write_command_private
                movwf   PCLATH

;***********************************************

                movlw   SFS

;***********************************************

                movwf   D0
                call    do_write_command_private

wait_sfs_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private
                movlw   HIGH wait_sfs_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_sfs_irq    ; IRQ not set

                movlw   HIGH error_staid
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_staid     ; BFREE not set

                movlw   HIGH do_write_command_private
                movwf   PCLATH
```

```
;************************************************

                movlw   RSID

;************************************************

                movwf   D0
                call    do_write_command_private
wait_rsid_irq
                movlw   HIGH do_read_status_private
                movwf   PCLATH
                call    do_read_status_private
                movlw   HIGH wait_rsid_irq
                movwf   PCLATH
                movf    D0, W
                andlw   IRQ
                btfsc   STATUS, Z
                goto    wait_rsid_irq   ; IRQ not set

                movlw   HIGH error_staid
                movwf   PCLATH
                movf    D0, W
                andlw   BFREE
                btfsc   STATUS, Z
                goto    error_staid     ; BFREE not set

                movlw   0x03
                movwf   CountDown
                movlw   #staid_base
                movwf   FSR

_staid_read_count1
                movlw   HIGH do_read_data_private
                movwf   PCLATH
                call    do_read_data_private
                movf    D0, W
                movwf   INDF
                incf    FSR, F
                movlw   HIGH _staid_read_count1
                movwf   PCLATH
                decfsz  CountDown, F
                goto    _staid_read_count1

                RETLW   0x00
```

B.1.7 Palm.h

Definitions and init values used in the assembler main program.

```
;************************************
; PALM.H
; Running on a PIC16C77
;************************************

;****************************************
; port data directions settings 0=OUT 1=IN
;****************************************

;*****  --PORTA--       *****
```

```
; RA7&RA6 not used, CSN=1, SHDN_5V=1, A1=0, A0=0, RDN=0, WRN=0

#DEFINE PORTA_INIT      B'00110000'
#DEFINE PORTA_INOUT     B'00000000'
#DEFINE SHDN_5V 4

; mask address, csn, rdn, wrn without affecting shdn_5V
#DEFINE command_or      0x06
#DEFINE command_and     0xD6
#DEFINE control_or      0x0A
#DEFINE control_and     0xDA
#DEFINE mode_or         0x0E
#DEFINE mode_and        0xDE
#DEFINE write_or        0x02
#DEFINE write_and       0xD2
#DEFINE status_or       0x05
#DEFINE status_and      0xD5
#DEFINE quality_or      0x09
#DEFINE quality_and     0xD9
#DEFINE read_or         0x01
#DEFINE read_and        0xD1
#DEFINE idle_or         0x23

;*****  --PORTB--       *****
; RB7=0, RB6=0, RTS=0, IRQN=0, RXD=0, TXselectN=1, RXselectN=1, CDN=0

#DEFINE PORTB_INIT      B'00000110'
#DEFINE PORTB_INOUT     B'11111001'
#DEFINE RXselectN       1
#DEFINE TXselectN       2
#DEFINE RTS     5
#DEFINE IRQN    4
#DEFINE CDN     0

; interrupt on change pins
#DEFINE RB5     5       ; RTS
#DEFINE RB4     4       ; IRQN

;*****  --PORTC--       *****
; RX=0, TX=0, CTS=0, EN_N=0, INVALIDN=0, RI=0, DSR=0, CD=0

#DEFINE PORTC_INIT      B'00000000'
#DEFINE PORTC_INOUT     B'10001000'
#DEFINE CTS     5
#DEFINE RI      2
#DEFINE DSR     1
#DEFINE CD      0

;*****  --PORTD--       *****
; D0-D7

#DEFINE PORTD_INIT      B'00000000'
#DEFINE PORTD_IN        B'11111111'
#DEFINE PORTD_OUT       B'00000000'

;*****  --PORTE--       *****
; RE2 - RE0: Amplifier off, LED_2 off, LED_1 off

#DEFINE PORTE_INIT      B'00000000'
#DEFINE PORTE_INOUT     B'00000000'

#DEFINE LED_1   0
```

```
#DEFINE LED_2   1
#DEFINE LED_1_mask      0x01
#DEFINE LED_2_mask      0x02
#DEFINE SHDN_AMP        2

;ADCON1 register init values - all multiplexed AD pins off
#DEFINE ADCON1_INIT     B'00000111'

; USART baudrate settings
#DEFINE _115200h        1       ; does not work
#DEFINE _57600h         3       ; does not work
#DEFINE _38400h         5
#DEFINE _19200h         11      ; does not work
#DEFINE _9600h          22

; Internal Clock, enable, prescaler
#DEFINE TIMER1_8        B'00110001'
#DEFINE TIMER1_4        B'00100001'
#DEFINE TIMER1_2        B'00010001'
#DEFINE TIMER1_1        B'00000001'
```

B.1.8 Module.h

Definitions and init values used to talk to the modemchip.

```
;/*****************************************
; * module.h
; *****************************************/

;/* register inital values in transmit mode */
#define ctrl_reg_transmit 0x40
#define mode_reg_transmit 0x20

;/* register inital values in receive mode */
#define ctrl_reg_receive 0x66
#define mode_reg_receive 0x00

;/* special byte streams */
#define symb_pat "\xF5\xF5\xF5\xF5\xF5\x5F"     ; not used in the
#define frame_pat "\x22\x37\x49\xF2\x5B\x1B"    ; asm version

;/* some prototypes */
#define task_proto 0x20

;/* status register bits */
#define IRQ 0x80
#define BFREE 0x40
#define IBEMPTY 0x20
#define DIBOVF 0x10
#define CRCERR 0x08
#define SRDY 0x04
#define SVAL 0x03

;/* mode register bits */
#define IRQNEN 0x80
#define INVSYM 0x40
#define TXRXN 0x20
#define RXEYE 0x10
#define PSAVE 0x08
```

```
#define SSIEN 0x04
#define SSYM 0x03

;/* control register bits */
#define CKDIV 0xC0
#define FSTOL 0x30
#define LEVRES 0x0C
#define PLLBW 0x03

;/* command register bits */
#define AQSC 0x80
#define AQLEV 0x40
#define CRC 0x20
#define TXIMP 0x10
#define reserved 0x08
#define TASK 0x07

;/* receive mode tasks */
#define NULLTASK 0x20
#define SFP 0x21
#define RHB 0x22
#define RILB 0x23
#define SFS 0x24
#define R4S 0x25
#define RSID 0x26
#define RESET 0x27

;/* transmit mode tasks */
;/* NULL & RESET already defined */
#define T24S 0x21
#define THB 0x22
#define TIB 0x23
#define TLB 0x24
#define T4S 0x25
#define TSID 0x26
```

B.1.9 palm_com.h

Commands used to talk to the module.

```
;**********************************
; PalmKiosk Modem Commands
;**********************************

#define read_status     0x00
#define read_quality    0x01
#define read_data       0x02
#define poll_cd         0x03
#define receiver_on     0x04
#define receiver_off    0x05
#define transceiver_on  0x06
#define transceiver_off 0x07
#define power_on        0x08
#define power_off       0x09
#define LED_1_toggle    0x0A
#define LED_2_toggle    0x0B
#define write_command   0x0C
#define write_mode      0x0D
#define write_control   0x0E
```

```
#define write_data      0x0F
#define set_staid       0x10
#define set_header      0x11
#define set_inter       0x12
#define set_last        0x13
#define get_staid       0x14
#define get_header      0x15
#define get_inter       0x16
#define get_last        0x17
#define transmit_frame  0x18
#define transmit_header 0x19
#define transmit_inter  0x1A
#define transmit_last   0x1B
#define receive_frame   0x1C
#define receive_header  0x1D
#define receive_inter   0x1E
#define receive_last    0x1F
#define transceiver_test        0x20
#define last_command    0x21

#define dummy_data      0xDD
#define carrier_detect  0x43
#define no_carrier_detect 0x4E
```

B.1.10 palm_err.h

Definitions to report various error conditions.

```
/************************************
* PALM_ERR.H
************************************/

/* module return codes */
#define error_indication        0x45
#define frame_error     0x46
#define overrun_error   0x4F
#define unknown_cmd     0x55
#define checksum_error  0x43

/* mod_cmd, mod_read, mod_write return codes */
#define EISBLOCKED -2
#define EINITSERIAL -3
#define EFAILED -1
```

B.1.11 mod_com.c

The SUN's side communication interface for our first program version. Shows how to open the serial port and set the modem control lines.

```
#include <termios.h>
#include <curses.h>
#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
```

```
#include <errno.h>
#include <sys/time.h>


#include "palm_com.h"
#include "palm_err.h"

static int serial_fd;
static int init_done = 0;

int mod_cmd(unsigned char c)
{
  int i = 0;
  unsigned char buf;
  int high = TIOCM_RTS;

  if (!init_done) {
    return(EINITSERIAL);
  }

do_it_again:
  if (ioctl(serial_fd, TIOCMBIC, &high)){
    return(EINITSERIAL);
  }
  if (write(serial_fd, &c, 1) == 0) {
    return(EINITSERIAL);
  }
  if (ioctl(serial_fd, TIOCMBIS, &high)){
    return(EINITSERIAL);
  }
  if (read(serial_fd, &buf, 1) != 1) {
    return(EISBLOCKED);
  }
  if (buf == c) {
    return(0);
  }
  else if((buf == frame_error) || (buf == overrun_error)){
    if (i < 5) {
      i++;
      if (tcflush(serial_fd, TCIOFLUSH)){
        return(EINITSERIAL);
      }
      goto do_it_again;
    }
    return(EINITSERIAL);
  }
  return(EFAILED);
}

int mod_read(unsigned char c, unsigned char *r)
{
  int i = 0;
  unsigned char buf;
  int high = TIOCM_RTS;

  if (!init_done) {
    return(EINITSERIAL);
  }

do_it_again:
  if (ioctl(serial_fd, TIOCMBIC, &high)){
    return(EINITSERIAL);
```

```
  }
  if (write(serial_fd, &c, 1) == 0) {
    return(EINITSERIAL);
  }
  if (ioctl(serial_fd, TIOCMBIS, &high)){
    return(EINITSERIAL);
  }
  if (read(serial_fd, &buf, 1) != 1) {
    return(EISBLOCKED);
  }
  if (buf == c) {
    if (read(serial_fd, r, 1) != 1) {
      return(EISBLOCKED);
    }
    return(0);
  }
  else if((buf == frame_error) || (buf == overrun_error)){
    if (i < 5) {
      i++;
      if (tcflush(serial_fd, TCIOFLUSH)){
        return(EINITSERIAL);
      }
      goto do_it_again;
    }
    return(EINITSERIAL);
  }
  return(EFAILED);
}

int mod_write(unsigned char c, unsigned char v)
{
  int i = 0;
  unsigned char buf;
  int high = TIOCM_RTS;

  if (!init_done) {
    return(EINITSERIAL);
  }

do_it_again:
  if (ioctl(serial_fd, TIOCMBIC, &high)){
    return(EINITSERIAL);
  }
  if (write(serial_fd, &c, 1) == 0) {
    return(EINITSERIAL);
  }
  if (write(serial_fd, &v, 1) == 0) {
    return(EINITSERIAL);
  }
  if (ioctl(serial_fd, TIOCMBIS, &high)){
    return(EINITSERIAL);
  }
  if (read(serial_fd, &buf, 1) != 1) {
    return(EISBLOCKED);
  }
  if (buf == c) {
    return(0);
  }
  else if((buf == frame_error) || (buf == overrun_error)){
    if (i < 5) {
      i++;
      if (tcflush(serial_fd, TCIOFLUSH)){
```

```
        return(EINITSERIAL);
      }
      goto do_it_again;
    }
    return(EINITSERIAL);
  }
  return(EFAILED);
}

int open_port(char *devname)
{
  serial_fd = open(devname, O_RDWR | O_NOCTTY);
  if (serial_fd == -1)
    {
      printf("open_port: unable to open %s - %s\n", devname, strerror(errno));
      return(-1);
    }
  return(0);
}

int set_serial_options (void)
{
  struct termios options;

  if (tcgetattr(serial_fd, &options)){
    printf("set_serial_options: unable to get options - %s\n", strerror(errno));
    return(-1);
  }
  options.c_cc[VTIME] = 1;
  options.c_cc[VMIN] = 0;
  cfsetispeed(&options, B38400);
  cfsetospeed(&options, B38400);
  options.c_cflag |= CLOCAL;
  options.c_cflag |= CREAD;
  options.c_cflag &= ~PARENB;
  options.c_cflag &= ~CSTOPB;
  options.c_cflag |= CS8;
  options.c_lflag &= ~ICANON;
  options.c_lflag &= ~ECHO;
  options.c_oflag &= ~OPOST;
  if (tcsetattr(serial_fd, TCSAFLUSH, &options)) {
    printf("set_serial_options: unable to set options - %s\n", strerror(errno));
    return(-1);
  }
  return(0);
}

int serial_init(void)
{
  if (open_port("/dev/ttya")) {
    return(-1);
  }
  printf("port /dev/ttya opened - fd = %d\n", serial_fd);
  if (set_serial_options()) {
    printf("serial_init: unable to set serial options - %s\n", strerror(errno));
    return(-1);
  }
  else {
    printf("serial_init: serial options set\n");
  }
  if (tcflush(serial_fd, TCIOFLUSH)){
    printf("serial_init: unable to flush queue - %s\n", strerror(errno));
```

```
    return(-1);
  }
  init_done = -1;
  return(0);
}
```

Bibliography

[1] Ocilawn Corp. *Spread spectrum Technology*, 1997. http://www.ocilawn.com/website9.htm

[2] Ocilawn Corp. *Radio Propagation*, 1997. http://www.ocilawn.com/websiteB.htm

[3] My T. Le, Frederick L. Burghardt, Srinivasan Seshan, Jan Rabaey. *InfoNet: the Networking Infrastructure of InfoPad*, Proceedings of Compcon, 1995.

[4] Frederick L. Burghardt. *Architecture and Implementation of the InfoPad Network Prototype*, University of California at Berkeley, 1994.

[5] K. Keeton, B. Mah, S. Seshan, R. Katz, D. Ferrari. *Providing Connection-Oriented Network Services to Mobile Hosts*, Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing, 1993.

[6] E. Amir, H Balakrishnan, S. Seshan, R. Katz. *Efficient TCP over Networks with Wireless Links*, University of California at Berkeley, 1994.

[7] M. Le, S. Seshan, F. Burghardt, J. Rabaey. *Software Architecture of the InfoPad System*, Proceedings of the Mobidata Workshop on Mobile and Wireless Information Systems, 1994.

[8] Frederick L. Burghardt. *Protocols for a Multimedia System: Investigation into Transport Mechanisms for the InfoPad Network Prototype*, University of California at Berkeley, 1994.

[9] Heinz Jäkel. *Skript Elektronik I*, Institut für Elektronik, ETHZ, 1995.

[10] Heinz Jäkel. *Skript Elektronik II*, Institut für Elektronik, ETHZ, 1996.

[11] Hubert Käslin. *VLSI I, Lecture notes on very large scale integration circuits*, Microelectronic Design Center, ETHZ, 1997.

[12] W. Fichtner. *Skript Elektronische Systeme*, Institut für Integrierte Systeme, ETHZ, 1996.

[13] Motorola Inc. *Integrated Portable System Processor – DragonBall MC68328*. 1995. http://www.mot.com/SPS/ADC/pps/prod/3XX/mc68328.html

[14] Leo Breuss. *PalmKiosk II, Drahtloses lokales Netzwerk für PalmtopComputer*, Institut für Elektronik, ETHZ, 1998.

[15] Thomas Schult. *Neue Beweglichkeit, Reisebüro, Kontaktsuche*. C'T Magazin, Heise Verlag, pages 138-159, 15/1997.

[16] H. Alkhatib et al. *Wireless Data Networks: Reaching th eExtra Mile*. IEEE Computer Magazine, pages 59-62, Jan. 1998.

[17] Motorola Corp. *Advanced Microcontroller Division 1996 Device Selection Guide*, 4th quarter, 1996.

[18] T. Oetiker. *The not so short introduction to LaTeX 2ε*, 1997.

[19] 3Com Corporation. *PalmPilot Hardware Development Kit*, 1997.

[20] 3Com Corporation. *PalmPilot Connected Organizer White Paper*, 1997. http://palmpilot.3com.com/products/whitepp.pdf.

[21] D. Karnigan. *Fortune Magazine, November 10, 1997*, 1997. http://pathfinder.com/fortune/1997/971110/ten3.html.

[22] Richard S. Shim. *Computer Shopper, December 1997*, 1997. http://www.zdnet.com/cshopper/content/9712/cshp0173.html.

[23] CML Semiconductors Ltd. *4-Level FSK Modem Data Pump FX929B*, 1997. http://www.cmlmicro.co.uk.

[24] T.Imielinski and H. Korth. *Mobile Computing*. Kluwer Academic Publishers, 1996.

[25] Linear Technology Corp. *LTC 1514-3.3 DC/DC Converters*, 1997.

[26] Linear Technology Corp. *LTC 1516 DC/DC Converters*, 1997.

[27] Maxim Integrated Product. *MAX3223 RS-232 Transceiver with Autoshutdown*, 1997. http://www.maxim.com.

[28] Microchip Technology Inc. *Application Note 555,Software Implementation of Asynchronous Serial IO*, 1997. http://www.microchip.com.

[29] Microchip Technology Inc. *8-Bit Microcontrollers*, 1997.

[30] Microchip Technology Inc. *PIC16C7X Data Sheet*, 1997.

[31] Radiometrix Ltd. *Low Power UHF Data Transceiver Module*, 1997.

[32] Radiometrix Ltd. *Radio Packet Controller*, 1997. http://www.radiometrix.com/products/rpc/rpcsheep.html.

[33] Radiometrix Ltd.Data Sheet. *BIM Evaluation Kit*, 1997.

[34] Radiometrix Ltd.Data Sheet. *Use of Radiometrix BIM Transceiver*, 1997.

[35] Radiometrix Ltd.Data Sheet. *Use of Radiometrix Radio Modules*, 1997.

[36] Radiometrix Ltd.Data Sheet. *FCC Power Limits*, 1997.

[37] Radiometrix Ltd.Data Sheet. *Type Approvals*, 1997.

[38] Radiometrix Ltd. *EMC Requirements*, 1997.

[39] Scenix Semiconductor Inc. *SCX18/SCX28 8-Bit In-System Programmable Microcontroller*, 1997. http://www.scenix.com.

[40] S. Paramananthan, Radiometrix Ltd. *Application Note 100, Error Performance of BIM Transceiver with RPC*, 1997. http://www.radiometrix.co.uk/apps/apnt100.htm.

[41] S. Paramananthan, Radiometrix Ltd. *Application Note 101, Error Performance of BIM Transceiver with RS232 Interface*, 1997. http://www.radiometrix.co.uk/apps/apnt101.htm.