KOMRAD Software Architecture

The Compact Open Mobile Reference Architecture And Design

Software Architecture Document – v0.1(Draft)

Rajiv Poddar (rajiv@tantratele.com)

Tantra Tele

1. Introduction........8
2. Requirements Overview........9
3. Architecture Overview........10
3.1. Hardware Platform........10
3.1.1. ARM Host Processor........11
3.1.2. DSP Baseband Processor........11
3.1.3. RF Subsystem........12
3.2. Microkernel........12
3.2.1. Evaluation of Open Source Microkernels........12
3.2.2. L4 Iguana........13
3.3. MMI Framework........13
3.3.1. Windows........14
3.3.2. Menus........14
3.3.3. Panels........14
3.3.4. Forms........14
3.4. Device Driver Framework........14
3.4.1. Interrupts........14
3.4.2. Session Based Interaction........15
3.4.3. Lock Free Data Structures........15
3.4.4. Direct Memory Access........15
3.4.5. Notifications........15
3.5. Applications........16
3.5.1. Text Based Browser........16
3.5.2. Ccal........16
3.5.3. Games........16
3.6. Networking 17
3.6.1. lwIP........17
3.6.2. GSM/CDMA Protocol Stack........17
3.7. Device Drivers........17
3.7.1. Watchdog Timer........17
3.7.2. Serial Port Driver........18
3.7.3. SIM Driver........18
3.7.4. Keypad........18
3.7.5. LCD Display........18
3.7.6. Sound Driver........19
3.7.7. USB 2.0........19
3.7.8. Baseband........19
3.8. IDE and SDK........19
3.8.1. Eclipse IDE........19
3.8.2. SDK........20
3.9. Interpreted Languages........20
3.9.1. Python........20
4. Modules........21
4.1. MMI Framework........21
4.2. User Interface........21
4.3. GSM Stack........21
4.4. CDMA 2000 Stack........21
4.5. Call Manager........21
4.6. Drivers Porting........21
4.7. DTMF 21
5. Flow Charts........22
5.1. Call Initiation........22
5.2. Call Termination........22
5.3. Redial........22
5.4. Speaker Phone On........22
5.5. Speaker Phone Off........22
5.6. Speed Dial........22
5.7. Phone Book Entry........22

5.8. Phone Book Entry Modification....22
5.9. Phone Book Entry Delete....22
5.10. PC Connection Establishment....22
5.11. PC Connection Termination....22
5.12. PPP Call Setup....22
5.13. PPP Call Termination....22
6. Licensing Considerations....23

Glossary

Term	Full Form
ADC	Analog to Digtal Convertor
ALSA	Advanced Linux Sound Architecture
BSD	Berkley System Distribution
CDMA	Code Division Multiple Access
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DMA	Direct Memory Access
DPC	Deferred Processing Component.
DSP	Digital Signal Processor
DTMF	Dual Tone Multi Frequency
ETSI	European Telecommunications Standards Institute
FTP	File Transfer Protocol
GPL	GNU Public License
GNU	GNU is Not Unix
GSM	Global System for Mobile Communication
IA32	Intel x86 architecture
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IMSI	International Mobile Subscriber Identity
I/O	Input/Output
ISR	Interrupt Service Routine
KB	Kilo Bytes
KOMRAD	Compact Open Mobile Reference Architecture And Design
LCD	Liquid Crystal Display
LCH	Low Cost Handset
MB	Mega Bytes
MMI	Man Machine Interface
NITCA	National ICT Australia
OS	Operating System
OSS	Open Source Software
OzPLB	Australian Public License B
PC	Personal Computer
PIM	Personal Information Manager
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer

Term	Full Form
RF	Radio Frequency
ROM	Read Only Memory
SDK	Software Development Kit
SIM	Subscriber Identity Module
SoC	System On Chip
TFTP	Trivial File Transfer Protocol
TLDP	The Linux Documentation Project
UART	Universal Asynchronous Transmitter Receiver
USB	Universal Serial Bus

References

[1] http://ertos.nicta.com.au/research/l4/
[2] http://nicta.com.au/director/commercialisation/open_source_licence/ozplb_licence%20.cfm
[3] http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/
[4] http://www.tldp.org/HOWTO/Serial-HOWTO.html
[5] http://en.wikipedia.org/wiki/Microkernel
[6] Kevin Elphinstone and Stefan Gotz, Initial Evaluation of a User-Level Device Driver Framework.
[7] http://savannah.nongnu.org/projects/lwip/
[8] http://www.gnu.org/software/gnuradio/index.html
[9] http://www.vovida.org/protocols/gsml/
[10] ETSI GSM 11.11 standard, http://www.ttfn.net/techno/smartcards/gsm11-11.pdf
[11] http://www.linuxnet.com/sourcedrivers.html
[12] http://www.itu.int/rec/T-REC-Q/recommendation.asp?lang=en&parent=T-REC-Q.23
[13] http://www.itu.int/rec/T-REC-Q/recommendation.asp?lang=en&parent=T-REC-Q.24
[14] http://lcdsmartie.sourceforge.net/
[15] http://www.alsa-project.org/
[16] http://www.usb.org/home
[17] http://www.linux-usb.org/
[18] http://www.eclipse.org/
[19] http://www.eclipse.org/cdt/index_old.html
[20] http://en.wikipedia.org/wiki/Python_programming_language

Revision History

Version	*Date*	*Author*	*Remark*
0.1 (Draft)	04/09/2006	Rajiv Poddar	First Draft Version

1. Introduction

This document describes the architecture of a fully integrated software solution for Low Cost Mobile Phones using Open Source Software. The solution consists of an Operating System (OS), Wireless Protocol Stack, Man Machine Interface (MMI) Framework, Device Driver Framework, Integrated Development Environment (IDE) and Software Development Kit (SDK). Various processor architectures including IA32, ARM, Power PC and MIPS are supported. All the software components are derived from open sources which are available publicly.

This document is intended for clients, developers, testers and end users. Section 2 provides an overview of the requirements, Section 3 provides an overview of the Architecture, Section 3 covers the details of the modules, Section 4 provides flow charts, and Section 5 explores some licensing considerations. Prior knowledge of embedded systems, mobile phone architecture and wireless telephony systems is assumed.

2. Requirements Overview

Low Cost Handsets (LCH) usually have very low memory and CPU power. They are also characterized by bare minimum feature set with larger stress on the basic functionality (ie. voice communications). The software for LCH therefore should be able to perform in a highly restricted environment and still provide powerful functionality. A basic set of requirements are as follows.

- **Small Footprint:** The complete software package should be small in footprint so as it can be used in the low memory devices. Generally the memory requirements can be as low as 32 KB to as high as 16 MB's. The build should be configurable so that with the addition and removal of packages the footprint can be adjusted to suit the device.
- **Telephony:** The software should support voice and data communication services. It should be standards compliant and provide all the necessary features of a mobile handset to the end user.
- **Real Time:** The kernel should support real time features. Real time execution is necessary since the telecommunication systems are highly time critical. The kernel must support pre-emptive execution, priority based scheduling and low interrupt latency.
- **Low Power:** The kernel should have the ability to hibernate when there is no user activity. It should be capable of going into hibernation mode very quickly and recover fast from that mode when any user activity is detected.
- **Linux Compatibility:** The kernel should support source code compatibility with Linux kernel. This enables any open source GNU/Linux application for which the source code is available to ported easily to the platform. Essential libraries which are needed by applications should be available on this platform as well.
- **Open Source:** All the software components should be open source or derived from open source implementations. All changes made should also be published back to the original project as patches. License restrictions should be followed with respect to the software packages used.
- **Cost:** The cost of the software should be as low as possible. The usage of open source software shall be maximized and only the time and effort required to make the customizations should be factored into the final cost.
- **Modem:** The software should also have the ability to provide wireless modem functionalities to the user when connected to a PC through an USB (or an equivalent) interface.
- **Manageability:** The software should support firmware and application upgrades from an PC interface or an networked interface.

3. Architecture Overview

The Figure 3.1: Architecture provides a simplified view of the software components.

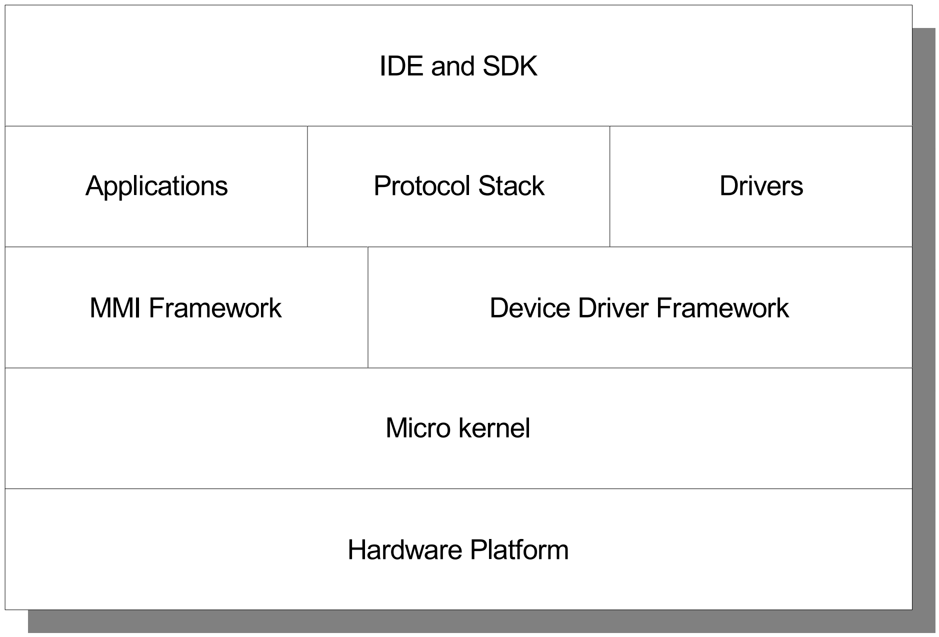

Figure 3.1: Architecture

A microkernel forms the core of the system. The kernel has a very small footprint which can be used on a restricted hardware platform. All device drivers are separate tasks which are managed by the kernel. The MMI Framework is ncurses based and can be used to create menus, panels, forms and windows. The DD Framework provides support for low level devices such as watchdog timers, boot ROM and serial ports. A suite of applications is also provided which includes a text based Web Browser, Games and Personal Information Manager (PIM). The Protocol Stack provides the telephony related functions and consists of all the upper layers of the stack as per the standards. Device drivers for keypad, display, sound and USB are also provided along with the system. An Eclipse based IDE is also provided in the form of plugins. The SDK consists of utility libraries which are ported the kernel API.

3.1. Hardware Platform

The illustration of the hardware platform is shown in Figure 3.1.1: Hardware Architecture. It consists of a ARM based host processor which is connected to a Baseband and RF subsystem via the system bus. External memory and other peripherals are connected to the host processor via the system bus or other interfaces. The hardware platform depicted is a generic embedded system platform. The software solution resides on the host processor and controls the baseband and peripherals through specialized device drivers.

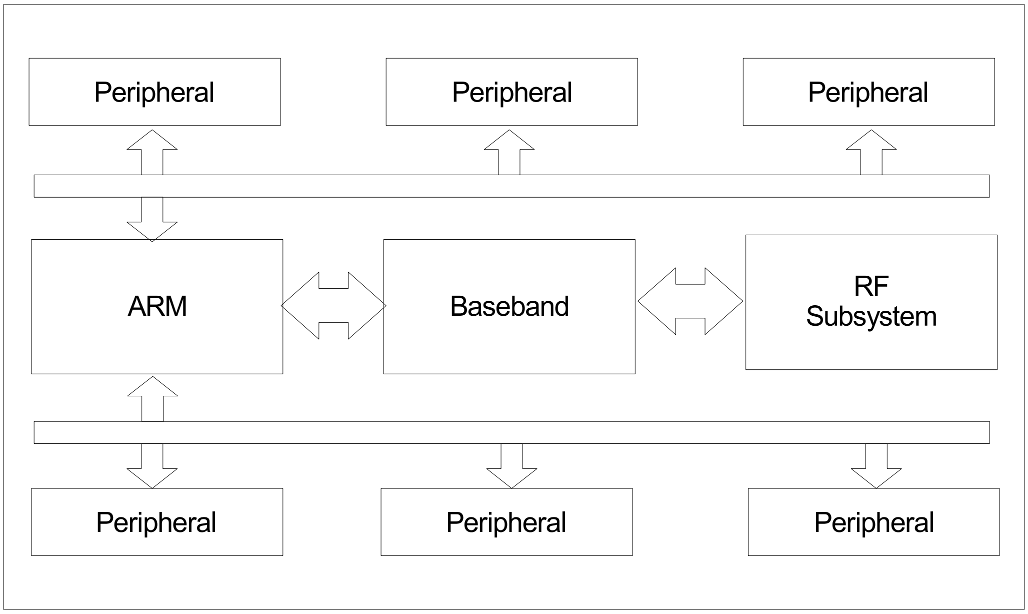

Figure 3.1.1: Hardware Architecture

3.1.1. ARM Host Processor

The ARM family of processor is one of the popular CPU's used in embedded devices. The are characterized by their low power consumption at the same CPU cycles compared to other CPU architectures. This makes the ARM processor an ideal choice for handheld terminals which are powered by rechargeable batteries.

The ARM is an 32 bit RISC processor with a simpler design than the contemporary IA 32 or Power PC CPU's. It features load/store of instructions, orthogonal instruction set, 16 x 32 bit registers and single cycle execution. Further each instruction can be optionally prefixed by an 4 bit condition code to conditionally execute the instruction. A 32 bit barrel shifter is also provided which can be used without any performance penalty. Register move operands can be also used in an instruction to shift the contents of the register. All these makes the assembly program denser leading to lowered foot prints for the binary.

The ARM7 family of CPU's use a three stage pipeline for fetch, decode and execute. The higher revision such as ARM9 and ARM11 have more pipeline stages such as faster adder and more extensive branch prediction logic.

The ARM core has been licensed by various manufacturer's such as Amtel, Broadcomm, CirrusLogic, Intel, Freescale, Samsung, STMicroelectronics, Texas Instruments, etc.

3.1.2. DSP Baseband Processor

The DSP baseband processor is used for the physical layer implementation of the wireless protocol stack and vocoder's. The DSP is a specialized microprocessor which is optimized for Digital Signal Processing functions. They separate program and data memories and support single instruction multiple data (SIMD) instructions. They usually take in data from an Analog to Digital converter (ADC) and produce outputs to an Digital to Analog Converter (DAC).

The baseband usually encodes/decodes the traffic in several stages which include interleaver, convolutional encoder, channel coder and a digital modulator. The exact implementation are specified in the set of standards for wireless telecommunication system being implemented. These stages require processors which can provide optimized mathematical computation and therefore are implemented using a DSP.

Optionally DSP's can also be used for vocoder's. The vocoder's are again specified by the set of standards and are used for producing a digitized version of the voice stream at a particular data rate which can be further used for digital transmission.

Several vendors provide specialized DSP processors for vocoder's and baseband s. TI, Phillips, Samsung etc. are popular amongst wireless communication devices.

3.1.3. RF Subsystem

The RF subsystem is used to upconvert and down convert the signals generated by the baseband processor to the actual transmission frequency. The usually consist of an ADC, DAC, a power amplifier and an upconverter. There are several methods of upconversion and downconversion in use and each one of them has their own specific implementation.

Optionally several System-On-Chip's (SoC's) can also be used for the RF subsystem which combine all the stages into a single chip. SoC's for GSM and CDMA are provided by several manufacturer's.

3.2. Microkernel

Microkernels are minimal operating systems kernels which only provide scheduling, address-space management and inter process communication (IPC). All the other functionalities are provided by user-space programs called servers. The device drivers are also treated as normal user-space programs and interact with the kernel to provide I/O services. Networking and PCMCIA services are also provided as user-space servers. This approach is a fundamental departure from the monolithic kernel based systems which provide all the above mentioned services within the kernel (eg. Linux).

Since the kernel only provides the essential services, the footprint is usually in kilo bytes. This makes them ideal for embedded systems which have low memory. Since all the device drivers are implemented as user-space programs, they are treated as any other process. All these leads to a highly modular and configurable system which is suited for embedded platforms.

3.2.1. Evaluation of Open Source Microkernels

There are several Open Source Microkernel's available, the most popular among them being the L4 family and Minix. There are many proprietary microkernel based operating systems which include Symbian, QNX and Singularity. Many experimental projects for microkernel's are currently in progress, prominent among them being Coyotos. A comprehensive list of microkernel's can be found in Wikipedia [5].

The aim of the evaluation process was to choose the most appropriate microkernel for the mobile handset. The criteria of evaluation in order of importance were the following.

1. Stability
2. Performance
3. Real Time Support
4. Documentation
5. ARM Port
6. Community Support
7. Build Process

The first step of the evaluation process was to exclude all closed source, proprietary and experimental microkernel's. Symbian, QNX and Coyotos were ruled out during this step and the options left were Minix and the L4 Family.

The second step was information gathering where research papers and references was obtained from the Internet. The L4 family proved to be the most researched and referred microkernel. Minix was ruled out at this stage since the build process was cumbersome and needed a prior installation of Minix.

In the third step the each variant of the L4 family of microkernel's were downloaded form their repository and tried out. The specific variants tried out were Iguana, Fiasco and Pistachio. Amongst these the Iguana turned out to be most suitable for embedded systems use. Pistachio and Fiasco turned out to be more suitable for desktop and server systems.

The L4 Iguana therefore was chosen to be the microkernel for further development. The major disadvantages was the lack of wide community support for it. Its a project primarily driven by National ICT Australia with corporate sponsorship. The project is licensed under OzPLB license [2] which is a BSD Style license. The sources are hosted in a bazaar archive and available publicly. The mailing list is not very active.

3.2.2. L4 Iguana

Iguana is an implementation of the L4 version 4 API implemented by L4::Pistachio microkernel and provides an environment for the OS. It supports paging and separation of address spaces. For devices with no Memory Management Unit (MMU) the memory protection can be removed. Most embedded systems fall into this category. The foot print of Iguana is very low (around 100 kilobytes) and a built in kernel debugger is also supported.

The L4 Iguana implements a new version of API called the L4-embedded and the corresponding implementation is called NITCA::Pistachio-embedded. This API version is optimized for embedded systems usage and differs in several ways from the L4::Pistachio which was designed for desktop and server use. The original kernel has been modified and long (string copy) IPC, recursive mappings and timeouts have been removed. This has resulted in lower kernel complexity and memory footprint. The API work is ongoing and the latest version is known as N1.

NITCA also provides an development environment called Kenge which is a set of packages which can be used for creating L4 applications and other L4 based microkernel based systems.

Further documentation can be found in the NITCA website [1].

3.3. MMI Framework

The MMI framework refers to the user interface provided on the mobile handsets. It generally consists of menus, windows and panels. The user uses navigation keys to choose a particular option which can either launch an application or lead to a sub menu.

The customized ncurses library port is used in this solution to provide a simple lightweight MMI Framework. The library is adapted to embedded system usage and provides an API's for creating menus, sub menus, panels, windows and forms. The framework is completely text based and does not support any graphics.

The display is treated as a VT 100 terminal with a specific numbers of rows and columns. These can be configured during the build. Special capabilities of VT 100 based terminals are also supported. UTF-8 character integration is also provided.

A detailed HOWTO for ncurses can be found at TLDP website [3].

3.3.1. Windows

Ncurses has native support for windows through library API calls. During initialization of ncurses it creates a default window named `stdscr`. This default window can be used to print messages using the `printw` API call. New windows can be created by the `newwin`. This call returns back a structure of type `WINDOWS` which can then be used in any other API calls such as `wprintw` and `wrefresh`. Finally the window can be deleted by calling the function `delwin`.

Several windows can be created and displayed simultaneously.

3.3.2. Menus

The Menu's library is provided as an adjunct to the ncurses library. All programs which use menus have to be linked with the menu as well as ncurses library. Menu items can be created with `new_item`, new menus can be created with `new_menu`, the menu can be displayed with `post_menu`, can be scrolled with `menu_driver`, removed with `menu_unpost` and the associated memory deleted with `free_menu` and `free_item`.

3.3.3. Panels

Panels are also provided as an adjunct library to ncurses and programs using panels has to be linked to both panels and the ncurses library. Panels can be created with the `new_panel` function and deleted with `del_panel` function. New windows can be attached to panels and stacked according to desired visibility. The panels themselves can be modified using the `update_panels`, `show_panel`, `hide_panel` etc. function calls.

3.3.4. Forms

Forms are also provided as an adjunct library to ncurses. The programs using forms have to be linked with forms and ncurses library simultaneously. Forms and similar to menus and to create a form, fields have to be created first with `new_field`. Forms can be created with `new_form` and fields can be attached to the forms. The function `form_post` can be used to display the form and a corresponding `form_driver` is provided to process the user actions. The forms can be unposted using `form_unpost` and the memory associated can be freed with `free_form` and `free_field`.

3.4. Device Driver Framework

The Device Driver Framework provides a mechanism to process hardware interrupts from the peripheral devices and redirecting them to the correct device driver processes for handling. All device drivers on L4 are treated as normal user space processes. The provides a service to register Interrupt Service Routine's (ISR) and Deferred Processing Component's (DPC). The ISR is responsible for reacting quickly and efficiently to the device events and deferring them for post processing. The ISR usually arranges for a DPC to continue the processing required to handle the device event. DPC's are usually activated via some kernel synchronization primitive which makes the activity runnable and adds it to the scheduler's run queue.

More information on the L4 Iguana Device Driver Framework can be found in [6].

3.4.1. Interrupts

The interrupts in L4 microkernel are represented as IPC's from virtual interrupt threads which uniquely identify the interrupt source. The real ISR within the kernel masks the interrupt, transforms the interrupt event into an IPC message from the interrupt thread which is delivered to the application’s ISR. The blocked ISR within the application receives the message, unblocks, and performs the normal ISR functionality. Upon completion, the driver ISR sends a reply message to the interrupt thread resulting in the interrupt source being unmasked. The ISR can then block waiting for the next interrupt IPC.

3.4.2. Session Based Interaction

Shared memory is used to to pass data from user space drivers to the kernel. Since copying data across protected boundaries is expensive, shared memory is passed by reference. Establishing shared memory is also expensive in terms of managing the hardware and performing book keeping in the software. Hence a concept of sessions is used for interaction with drivers. A session is logical concept within which a sequence of interactions between the client and drivers is performed. To enable pass-by-reference data delivery, one or more dataspaces can be associated with a session for its duration. Dataspaces can contain a shared memory region used to allocate buffers, a client's entire address space or a small page sized object.

3.4.3. Lock Free Data Structures

Lock Free queues, implemented as linked lists or circular buffers allow work to be enqueued for a driver or a client without any explicit interaction with the driver on every operation. This results in a batch mechanism where several lock free operations follow each other and finally the recipient driver is notified via an explicit interaction.

3.4.4. Direct Memory Access

The translation of dataspace pages to physical frames is required by drivers of DMA-capable devices. This translation is only known by a dataspace manager. To enable translation, the dataspace manager provides a shared memory region between it and the device driver: the translation cache. The translation cache is established between the manager and driver when a dataspace is added to a session between the driver and client. Multiple dataspaces from the same manager can share the same translation cache. The translation cache contains entries that translate pages within dataspaces into frames . The cache is consulted directlyby the driver to translate buffer addresses it has within dataspaces to physical addresses for DMA. After the translation cache is set up, the driver only needs to interact with the object implementor in the case of a cache miss.

In addition to translating a buffer address to a physical address for DMA, the driver needs a guarantee for the duration of DMA that the translation remains valid, i.e. the page (and associated translation) must remain pinned in memory. This can be achieved in two ways. The first method is to use time-based pinning where entries in the translation cache have expiry times. The second method is to share state between the driver and dataspace implementor to indicate the page is in use and should not be paged out.

3.4.5. Notifications

An efficient activation mechanism is provided for ISRs to hand-off work to DPCs, and for both clients and drivers to deliver work and potentially block as the sender and while activating the recipient via queues. In L4 IPC blocking involves waiting for a message, activating involves sending a message. To avoid notifications when unnecessary, the recipient of notifications indicates its thread state via shared memory. If marked inactive, a notification is sent; if not it is assumed that the recipient is (or will be) active and the notification is suppressed. A general mechanism called preemption control, which can make threads aware of preemption. In the rare case that a preemption is detected, the recipient rolls back to a safe active state from where it tries to block again.

3.5. Applications

A suite of applications are provided along with the solution. It includes a text based browser, a simple PIM, and few games. They are all derived from Open Source Software solutions and are relatively simple. They have small foot print and are text based.

3.5.1. Text Based Browser

ELinks is an advanced and well-established feature-rich text mode web (HTTP/FTP/..) browser. ELinks can render both frames and tables, is highly customizable and can be extended via the general purpose extension languages, Lua or Guile. It is very portable and runs on a variety of platforms. ELinks is written in near-ANSI C and is very portable. The following features are supported.

- Lots of protocols (local files, finger, http, https, ftp, smb, ipv4, ipv6)
- Authentication (HTTP authentication, Proxy authentication)
- Persistent cookies
- Cute menus and dialogs
- Tabbed browsing
- Support for browser scripting (Perl, Lua, Guile)
- Tables and frames rendering
- Colors
- Background (non-blocking) downloads

The Elinks application will be ported to the L4 Iguana platform along with other required libraries.

3.5.2. Ccal

Ccal is a curses based calendar, journal, diary and to-do list program. It uses python 2.3 and curses library. It supports the following features.

- Web-based storage of appointments
- Todo list - for items not associated with a particular date
- Calendar - highlights days with appointments and is used to navigate through the entries
- UpNext mode - view upcoming appointments at a glance and scroll through them
- Non-interactive command-line based output of day's appointments and/or todo items
- ical import/export
- Cut and paste of entries between todo and diary and from date to date
- Postscript output of upcoming appointments (for printing)
- Editor based entry for extra information on entries
- Colour coding of entries to indicate priority or whatever else you want
- Email Import

The essential features of ccal will be ported to the L4 Iguana OS.

3.5.3. Games

The following based games will be included in the software package.

- **Tic-Tac-Toe:** a simple game of tic tac toe against the system or an opponent.
- **nInvaders:** clone of windows invaders.
- **Conquest:** ncurses based, multi-player, space warfare game
- **Pente:** ncurses based, multi-player, five-in-a-row game.
- **CBoard:** ncurses based front end for GNU Chess game.

3.6. Networking

L4 Iguana does not support any networking services. Therefore the protocol stacks for TCP/IP, PPP, CDMA or GSM will be added to this solution.

3.6.1. lwIP

The Light Weight IP is a small independent implementation of the TCP/IP protocol suite that has been developed by Adam Dunkels at the Computer and Networks Architectures (CNA) lab at the Swedish Institute of Computer Science (SICS).

The focus of the lwIP TCP/IP implementation is to reduce resource usage while still having a full scale TCP. This making lwIP suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.

The lwIP features are as follows:

- IP (Internet Protocol) including packet forwarding over multiple network interfaces
- ICMP (Internet Control Message Protocol) for network maintenance and debugging
- UDP (User Datagram Protocol) including experimental UDP-lite extensions
- TCP (Transmission Control Protocol) with congestion control, RTT estimation and fast recovery/fast retransmit
- Specialized raw API for enhanced performance
- Optional Berkeley-alike socket API
- DHCP (Dynamic Host Configuration Protocol)
- PPP (Point-to-Point Protocol)
- ARP (Address Resolution Protocol) for Ethernet

More information about lwIP can be found on the project homepage [7].

3.6.2. GSM/CDMA Protocol Stack

There are no open source implementations of the GSM or CDMA protocol stack available. Implementations of the baseband components are although available in GNURadio project [8]. The signaling protocols of GPRS stack can be found at Vovidia [9] as part of a base station implementation.

The complete protocol stack of GSM and CDMA will be developed as part of this project. Initially the GNURadio hardware will be used to develop a BTS Simulator. Thereafter hardware reference boards will be developed which are suitable for GSM and CDMA based devices. The protocol stack will be tested on those boards. Eventually all types of commercially available chipsets will be supported.

3.7. Device Drivers

The Device Drivers are used to control the peripherals connected to the system. They include the mobile handset specific peripherals such as SIM Driver, the Keypad Driver, the Display driver, the Sound Driver, the USB 2.0 Driver and the Baseband Driver and utility peripherals such as watchdog timers and serial ports. Generally the device drivers are supplied by the manufacturer of the peripheral. These are ported to the microkernel OS so that they can be used in the system.

On the L4 microkernel, all the device drivers also appear as processes running above the OS and are indistinguishable from other processes.

3.7.1. Watchdog Timer

The watchdog timer is a hardware timer which resets the system when it detects that the main program neglects to service the watchdog. The watchdog timer is serviced by writing a service pulse to the it and the software should do it at a configurable number of times each minute. If the timer detects that that the Operating System has not serviced the timer for a configurable amount of time, it can initiate a system reset. This mechanism is used to bring the system from an unstable state to normal operation. If a particular task occupies the CPU time indefinitely due to a fault then the watchdog timer resets the system and brings it back into operation.

Watchdog timers are usually built into the CPU or the micro controller. The drivers for the watchdog timers should provide system calls to service the watchdog timer. One specific task called the idle task usually takes care or servicing the watchdog timer.

3.7.2. Serial Port Driver

Serial ports are simple low speed I/O ports which are compliant to the RS-232 standard. They are widely used in embedded systems for debugging in the event the system does not boot up properly or does not have a display. Specialized chipsets called UART are available which can be integrated into the system and provide a serial port connection. It has to be connected to another host system through a serial cable. A terminal emulation software (eg. minicom, hyperterm) can be used thereafter to send commands and receive outputs.

The serial driver is responsible for initialization of the serial port and managing data transfers according to the standard. It provides a wrapper which hides the internal details of the serial protocol and publishes the standardized I/O system calls for it. These I/O calls then can be used by any other module in the system to output data or take inputs.

A detailed HOWTO on serial ports can be found at TLDP [4].

3.7.3. SIM Driver

The Subscriber Identity Module (SIM) card is a smart card used for storing the subscriber information in the GSM and GPRS mobile phones. It stores a variety of information including the number (IMSI), authentication keys, Location Area Identity (LAI) etc. The IETF Standard GSM 11.11 [10] specifies what can be stored in the SIM card. The SIM card follows the PC/SC (PC Smart Card) interface which is based upon the interface and programming model of ISO 7816.

MUSCLE [11] is lists the open source projects for various Linux based PC/SC drivers. The drivers depend upon the choice of the smart card reader hardware. Several vendors provide such hardware. The drivers for all mobile smart card readers will be ported to the L4 Iguana so as to enable wide support for various hardware.

3.7.4. Keypad

The mobile phone keypad operation are governed by the ITU-T Q.23 [12] and Q.24 [13] Dual Tone Multi Frequency standards. It specifies the procedures for DTMF dialing and reception. There are currently no open source implementations of this algorithm available yet.

A project for implementing this standard will be kicked off as part of KOMARD.

3.7.5. LCD Display

Passive Matrix based LCD Display drivers are popular for LCH's. These displays use Supertwist Nematic (STN) or Double-ayer STN (DSTN) technology. They are characterized by slow response times and poor contrast but are widely used for their low cost and simplicity.

There are various LCD Display drivers are available for Linux. One such project is LCD Smartie [14]. There are several others. The exact driver which will be ported would depend upon the results of an evaluation and is currently TBD.

3.7.6. Sound Driver

The Advanced Linux Sound Architecture (ALSA) [15] provides audio and MIDI functionality to the Linux operating system. ALSA has the following significant features:

- Efficient support for all types of audio interfaces, from consumer soundcards to professional multichannel audio interfaces.
- Fully modularized sound drivers.
- SMP and thread-safe design.
- User space library (`alsa-lib`) to simplify application programming and provide higher level functionality.
- Support for the older OSS API, providing binary compatibility for most OSS programs.

The alsa-lib and the driver components will be ported to the L4 Iguana and made available as part of the SDK.

3.7.7. USB 2.0

The USB 2.0 standard is specified by the USB Implementors Forum (USB-IF) [16]. For mobile phones and other portable devices smaller USB plugs and receptacles, called Mini-A and Mini-B, are also available. They are specified by the On-The-Go Supplement to the USB 2.0 Specification.

Several open source implementations of USB drivers are available. The Linux USB project provides information for various such drivers. The Linux kernel driver for USB will be ported to the L4 Iguana platform.

3.7.8. Baseband

Various vendors provide baseband chipsets for GSM and CDMA family of wireless protocols. They also supply the firmware for the baseband along with the chipset. These implement a proprietary API between the host and the baseband and are tied to the chipset provider. Depending upon the chipset chosen, the particular API will be implementation will be wrapped in a generic API based upon the standards. A sub project for this will be executed as part of KOMRAD.

3.8. IDE and SDK

An IDE and SDK is provided for development and debugging. The IDE is Eclipse based and plugins for the OS tool chains are provided. The SDK includes the libraries which are needed for development of applications over the OS. All the essential libraries are provided.

3.8.1. Eclipse IDE

Eclipse [18] is an open source community whose projects are focused on providing a vendor-neutral open development platform and application frameworks for building software. The Eclipse Foundation is a not-for-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services.

The CDT (C/C++ Development Tools) [19] plugin for eclipse provides a fully functional C and C++ Integrated Development Environment (IDE) for the Eclipse platform. It has been developed for Linux but there are ports available for Windows and other unices.

3.8.2. SDK

The Software Development Kit will provide the essential tool chains and libraries for general purpose application development over L4 Iguana. The tool chain would be a ARM based GCC cross complier which can be used to compile applications for the ARM processor.

The list of libraries ported to the L4 Iguana OS is TBD.

3.9. Interpreted Languages

Interpreted languages are very popular in the open source world and widely used. Several projects depend upon these languages for the actual implementation. Ports of these languages would be made available for the L4 Iguana OS as well.

3.9.1. Python

Python is an high level open source interpreted programming language which is widely used for shell scripts as well as application programming. It is fully dynamically typed and uses automatic memory management. Its most notable feature is that it emphasizes the importance of programmer effort over that of computers and rejects more arcane language features. The readability and expressiveness is stressed upon more than the speed.

The interpreter is implemented in C and is called as CPython. A port of Cpython will be made available on the L4 Iguana OS.

A detailed introduction to the Python programming language can be found on Wikipedia [20].

4. Modules

Several modules will be developed as part of this project in order to provide a complete end to end platform for LCH's. These modules are the one's which are not available as open source software yet. They design and architecture of these modules are discussed in details below. Each one is a separate open source sub project.

4.1. MMI Framework

TBD

4.2. User Interface

TBD

4.3. GSM Stack

TBD

4.4. CDMA 2000 Stack

TBD

4.5. Call Manager

TBD

4.6. Drivers Porting

TBD

4.7. DTMF

TBD

5. Flow Charts

5.1. Call Initiation

TBD

5.2. Call Termination

TBD

5.3. Redial

TBD

5.4. Speaker Phone On

TBD

5.5. Speaker Phone Off

TBD

5.6. Speed Dial

TBD

5.7. Phone Book Entry

TBD

5.8. Phone Book Entry Modification

TBD

5.9. Phone Book Entry Delete

TBD

5.10. PC Connection Establishment

TBD

5.11. PC Connection Termination

TBD

5.12. PPP Call Setup

TBD

5.13. PPP Call Termination

TBD

6. Licensing Considerations

All the software used in this solution are derived from Open Sources. They follow various licensing methods and the licenses need to adhered to for commercial applications. The GNU Public License (GPL) which restricts mixing of any proprietary code with it (copyleft clause). Further GPL'ed code cannot be mixed with BSD style licenses since GNU forbids addition of more clauses. The Library GPL (LGPL) licenses are used for the libraries and it does not have the copyleft clause.

In the software solution, the kernel is licensed under BSD Style and the ncurses library is provided under the MIT license. All the device drivers are either proprietary in nature or GPL'ed. The applications provided are mostly GPL.

Open Items

Issue	*Resolution*
Do we assume that the baseband software is provided by the chipset vendor?	
Is Protocol Stack Certification needed?	