Porting Android to New Hardware

Android Builders Summit – April 14th 2011

Karim Yaghmour
karim.yaghmour@opersys.com
@karimyaghmour
About ...

- Author of:

- Introduced Linux Trace Toolkit in 1999
- Originated Adeos and relayfs (kernel/relay.c)
1. Components to port
2. Cross-development toolchain
3. Porting the bootloader
4. Porting the Linux kernel
5. Developing device drivers
6. Getting the AOSP
7. Implementing Android hardware libs
8. Customizing the user-space
9. Building the AOSP
10. Components to write to flash
11. Useful Embedded Linux tricks
## 1. Components to port

<table>
<thead>
<tr>
<th></th>
<th>CPU Architecture</th>
<th>CPU Model</th>
<th>Target Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU toolchain</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>bootloader</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Kernel</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bionic</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSS packages</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dalvik</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware libs</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
2. Cross-development toolchain

- Mainly ARM
- Prebuilt toolchains:
  - Codersourcery
  - Linaro
- Auto-generating a toolchain:
  - crosstool-ng
  - Buildroot
  - PTXdist
  - OpenEmbedded
3. Porting the bootloader

- Check aosp/bootable/bootloader/legacy
  - README
  - fastboot_protocol.txt
- CPU support:
  - include/[cpu]/*
  - arch_[cpu]/*
- Board support:
  - Have your pick ...
4. Porting the Linux kernel

- Requirements
- Kernel Architecture
- Androidisms
- Which kernel to start from
- An intro to kernel source layout
- Using a JTAG debugger
4.1. Requirements

- Kernel is loaded in RAM and run by bootloader
- Board schematics
- Physical memory map
- Chip timings
- Receiving proper boot parameters from bootloader
4.2. Kernel Architecture
4.3. Androidisms

- Wakelocks
- lowmem handler
- Binder
- RAM console
- Logger
- ...

> OPERSYS TM
4.4. Which kernel to start from

- Google:
  - http://android.git.kernel.org/
- Vanilla:
  - http://www.kernel.org
- Either way ... you're screwed:
  - Android kernel is a fork
  - No resolution in sight
  - **Cannot** use vanilla kernel as-is ... wakelocks
  - Learn how to use “git rebase”
### 4.5. An intro to kernel source layout

<table>
<thead>
<tr>
<th>Directory</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch</td>
<td>112MB</td>
<td>architecture-dependent functionality</td>
</tr>
<tr>
<td>block</td>
<td>600KB</td>
<td>block layer</td>
</tr>
<tr>
<td>Documentation</td>
<td>17MB</td>
<td>main kernel documentation</td>
</tr>
<tr>
<td>drivers</td>
<td>231MB</td>
<td>all drivers</td>
</tr>
<tr>
<td>fs</td>
<td>31MB</td>
<td>virtual filesystem and all fs types</td>
</tr>
<tr>
<td>include</td>
<td>20MB</td>
<td>complete kernel headers</td>
</tr>
<tr>
<td>init</td>
<td>150KB</td>
<td>kernel startup code</td>
</tr>
<tr>
<td>ipc</td>
<td>224KB</td>
<td>System V IPC</td>
</tr>
<tr>
<td>kernel</td>
<td>4.7MB</td>
<td>core kernel code</td>
</tr>
<tr>
<td>mm</td>
<td>2.2MB</td>
<td>memory management</td>
</tr>
<tr>
<td>net</td>
<td>20MB</td>
<td>networking core and protocols</td>
</tr>
<tr>
<td>scripts</td>
<td>1.1MB</td>
<td>scripts used to build kernel</td>
</tr>
<tr>
<td>tools</td>
<td>2.1MB</td>
<td>misc. kernel-related tools</td>
</tr>
</tbody>
</table>
- **arch/**
  - 2.4M alpha
  - 29M arm
  - 1.4M avr32
  - 5.3M blackfin
  - 4.9M cris
  - 1.4M frv
  - 856K h8300
  - 4.6M ia64
  - 8.0K Kconfig
  - 1.4M m32r
  - 5.7M m68k
  - 1.1M m68knommu
  - 1.2M microblaze
  - 11M mips
  - 1.7M mn10300
  - 2.4M parisc
  - 13M powerpc
  - 2.4M s390
  - 636K score
  - 5.4M sh
  - 4.7M sparc
  - 1.9M tile
  - 1.9M um
  - 8.5M x86
  - 1.4M xtensa
• arch/arm:
  136K boot
  208K common
  676K configs
  1.1M include
  252K lib
  96K mach-aaec2000
  1.2M mach-at91
  808K mach-bcmring
  ...
  748K mm
  308K nwfpe
  12K oprofile
  60K plat-iop
  788K plat-mxc
  76K plat-nomadik
  ...
• drivers/

accessibility  cpufreq  hwmon  mca  parisc  sbus  uio
acpi           cpuidle  i2c   md    parport  scsi  usb
amba           crypto   ide   media  pci    serial  uwb
ata            dca      idle  memstick pcmcia sfi  vhost
atm            dio      ieee802154 message platform sh  video
auxdisplay     dma      infiniband mfd  pnp    sn    virtio
base           edac     input  misc   power  spi   vlynq
block          eisa     isdn   mmc   pps    ssb   wl
bluetooth      firewire Kconfig mtd  ps3    staging watchdog
cdrom          firmware leds  net   rapidio tc    xen
clocksource    gpu      macintosh of   rtc    telephony zorro
connector      hid      Makefile  oprofile s390  tty

• include/

acpi           config  drm  keys  math-emu  mtd  pcmcia  rxrpc  sound  video
asm-generic   crypto   Kbuild  linux  media  net  rdma  scsi  trace  xen
• Looking for something:
  • Try `grep`
  • Have a look at the Linux Cross-Referencing project:
    – URL: http://lxr.linux.no/
    – Code: http://lxr.sourceforge.net/
  • Advanced kernel searching/understanding:
    – CScope: http://cscope.sourceforge.net/
    – KScope front-end: http://kscope.sourceforge.net/
  • ETAGS (emacs)
4.6. Using a JTAG debugger

- Allows debugging of:
  - Bootloader
  - Early kernel code
  - Device drivers
- Need to find one that supports Linux kernel:
  - Abatron
  - Lauterbach
  - GreenHills Software
  - ...
5. Developing device drivers

- Everything in Unix is a file, including devices
- Get a copy of Linux Device Drivers, 3rd ed.
- BTW, emulator kernel doesn't allow modules >:(
- Use standard Linux model API
- Try avoiding wakelocks in drivers
- Use modules for development
- Build drivers in when you ship
- Remember: kernel is GPL, drivers are ... ???
- Try using user-space “drivers” for proprietary parts
- Android actually promotes use of user-space hardware libs
5.1. User space vs. kernel space

- Separate address space:
  - No explicit references to objects from other space
- Memory protection amongst processes:
  - No process can directly access or alter other processes' memory areas.
- Memory protection between processes and kernel:
  - No process can access anything inside the kernel
  - Processes that attempt die (segfault)
- Crossing between user space and kernel space is through specific events
5.2. Connecting user-space and drivers

**Major / Minor Numbers**

- **Application**
  - User-Space
- **Virtual Filesystem**
  - Major:251
  - Kernel-Space
- **Device Instance**
  - Minor:5
- **Your Driver**
  - Minor:5
- **Filesystem**
  - / dev / foo
  - Major:251
  - Minor:5
5.3. Types of drivers

- Char
- Block
- Net
- Subsystem:
  - USB
  - MTD
  - Framebuffer
  - Input
5.4. Kernel primitives

- Timing
- Interrupt handling and deferral
- Memory management
- /sys, hotplug, etc.
- Locking mechanisms
- Hardware access
- ...
6. Getting the AOSP

- Code-drop every ~6 months
- Location:
  - http://android.git.kernel.org/
- Get “repo”:
  - $ curl http://android.git.kernel.org/repo > ~/bin/repo
  - $ chmod a+x ~/bin/repo
- Fetch the AOSP:
  - Make sure you fetch a tagged release
  - Gingerbread:
    - $ repo init -u git://android.git.kernel.org/platform/manifest.git -b gingerbread
    - $ repo sync
# 6.1. AOSP content

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bionic</td>
<td>C library replacement</td>
</tr>
<tr>
<td>bootable</td>
<td>Reference bootloader</td>
</tr>
<tr>
<td>build</td>
<td>Build system</td>
</tr>
<tr>
<td>cts</td>
<td>Compatibility Test Suite</td>
</tr>
<tr>
<td>dalvik</td>
<td>Dalvik VM</td>
</tr>
<tr>
<td>development</td>
<td>Development tools</td>
</tr>
<tr>
<td>device</td>
<td>Device-specific files and components</td>
</tr>
<tr>
<td>external</td>
<td>Copy of external projects used by AOSP</td>
</tr>
<tr>
<td>frameworks</td>
<td>System services, android.*, Android-related cmds, etc.</td>
</tr>
<tr>
<td>hardware</td>
<td>Hardware support libs</td>
</tr>
<tr>
<td>libcore</td>
<td>Apache Harmony</td>
</tr>
<tr>
<td>ndk</td>
<td>The NDK</td>
</tr>
<tr>
<td>packages</td>
<td>Stock Android apps, providers, etc.</td>
</tr>
<tr>
<td>prebuilt</td>
<td>Prebuilt binaries</td>
</tr>
<tr>
<td>sdk</td>
<td>The SDK</td>
</tr>
<tr>
<td>system</td>
<td>pieces of the world that are the core of the embedded linux platform at</td>
</tr>
<tr>
<td></td>
<td>the heart of Android.</td>
</tr>
</tbody>
</table>

The SDK and ndk are the hardware support libraries designed for the Android platform, providing the necessary components for the effective functioning of the Android operating system in embedded systems. The frameworks, on the other hand, are integral to the Android runtime, offering a wide range of services and functionalities to facilitate the execution of applications and services, from system services to Android-related commands. The SDK serves as the foundation for developers, offering a suite of tools and a comprehensive set of libraries and APIs for building applications that run on Android. The ndk, or Native Development Kit, is the native development framework for Android, providing a platform for developing native code that can be integrated with the Android runtime. Together, these components form the backbone of the Android platform, ensuring its flexibility, performance, and expandability.
6.2. Useful pointers

- See the build system doc at source.android.com
- Check out device/ in AOSP
- Check out Cyanogenmod
- Check out xda-developers
7. Implementing Android hardware libs

<table>
<thead>
<tr>
<th>Component</th>
<th>Implementation Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluetooth</td>
<td>BlueZ through D-BUS IPC (to avoid GPL contamination it seems)</td>
</tr>
<tr>
<td>GPS</td>
<td>Manufacturer-provided libgps.so</td>
</tr>
<tr>
<td>Wifi</td>
<td>wpa_supplicant</td>
</tr>
<tr>
<td>Display</td>
<td>Std framebuffer driver (/dev/fb0)</td>
</tr>
<tr>
<td>Keymaps and Keyboards</td>
<td>Std input event (/dev/event0)</td>
</tr>
<tr>
<td>Lights</td>
<td>Manufacturer-provided liblights.so</td>
</tr>
<tr>
<td></td>
<td>Backlight</td>
</tr>
<tr>
<td></td>
<td>Keyboard</td>
</tr>
<tr>
<td></td>
<td>Buttons</td>
</tr>
<tr>
<td></td>
<td>Battery</td>
</tr>
<tr>
<td></td>
<td>Notifications</td>
</tr>
<tr>
<td></td>
<td>Attention</td>
</tr>
<tr>
<td>Audio</td>
<td>Manufacturer-provided libaudio.so (could use ALSA underneath ... at least as illustrated in their porting guide)</td>
</tr>
<tr>
<td>Camera</td>
<td>Manufacturer-provided libcamera.so (could use V4L2 kernel driver underneath ... as illustrated in porting guide)</td>
</tr>
<tr>
<td>Power Management</td>
<td>“Wakelocks” kernel patch</td>
</tr>
<tr>
<td>Sensors</td>
<td>Manufacturer-provided libsensors.so</td>
</tr>
<tr>
<td></td>
<td>Accelerometer</td>
</tr>
<tr>
<td></td>
<td>Magnetic Field</td>
</tr>
<tr>
<td></td>
<td>Orientation</td>
</tr>
<tr>
<td></td>
<td>Gyroscope</td>
</tr>
<tr>
<td></td>
<td>Light</td>
</tr>
<tr>
<td></td>
<td>Pressure</td>
</tr>
<tr>
<td></td>
<td>Temperature</td>
</tr>
<tr>
<td></td>
<td>Proximity</td>
</tr>
<tr>
<td>Radio Layer Interface</td>
<td>Manufacturer-provided libril-&lt;companyname&gt;-&lt;RIL version&gt;.so</td>
</tr>
</tbody>
</table>
8. Customizing the user-space

- Boot screen
- Status bar
- Network
- Preloaded apps
- Browser bookmarks
- Email provider customization
- Themes
- Adding new applications
- Adding new services / new hardware type
- Init
8.1. Boot screen

- Create 320x480 image
- Install imagemagick
  
  $ sudo apt-get install imagemagick
- Convert image to .r format
  
  $ sudo apt-get install imagemagick
- Convert image to 565 format
  
  $ rgb2565 < screen.rgb > screen.565
- Write image to flash
  
  $ fastboot flash splash1 screen.565
8.2. Status bar

• Location:
  • frameworks/base/packages/SystemUI/src/com/android/systemui/statusbar

• Look for:
  • mService.setIcon(...)

• Disable icons with:
  • mService.setIconVisibility("[ICON_NAME]", false);
8.3. Network

- **Locations:**
  - **Global static:**
    - frameworks/base/core/res/res/xml/apns.xml
  - **Device static:**
    - PRODUCT_COPY_FILES := vendor/acme/etc/apns-conf-us.xml:system/etc/apns-conf.xml
  - **Dynamic:**
    - system/etc/apns-conf.xml

- **Format:**

  ```xml
  <apn carrier="T-Mobile US"
  mcc="310"
  mnc="260"
  apn="wap.voicestream.com"
  user="none"
  server="*"
  password="none"
  proxy="216.155.165.50"
  port="8080"
  mmsc="http://216.155.174.84/servlets/mms"
  />
  ```
8.4. Preloaded apps

- See build/target/products

```bash
PRODUCT_PACKAGES := \ 
  bouncycastle \ 
  com.android.location.provider \ 
  com.android.location.provider.xml \ 
  core \ 
  core-junit \ 
  create_test_dmtrace \ 
  dalvikvm \ 
  dexdeps \ 
  ...
```
8.5. Browser bookmarks

- See packages/apps/Browser/res/values/strings.xml

<!-- Bookmarks -->
<string-array name="bookmarks">
  <item>Google</item>
  <item>http://www.google.com/</item>
  <item>Yahoo!</item>
  <item>http://www.yahoo.com/</item>
  <item>MSN</item>
  <item>http://www.msn.com/</item>
  <item>MySpace</item>
  <item>http://www.myspace.com/</item>
  ...
</string-array>
8.6. Email provider customization

- See packages/apps/Email/res/xml/providers.xml

```xml
<!-- Gmail variants -->
<provider id="gmail" label="Gmail" domain="gmail.com">
  <incoming uri="imap+ssl://imap.gmail.com" username="$email"/>
  <outgoing uri="smtp+ssl://smtp.gmail.com" username="$email"/>
</provider>

<provider id="googlemail" label="Google Mail" domain="googlemail.com">
  <incoming uri="imap+ssl://imap.googlemail.com" username="$email"/>
  <outgoing uri="smtp+ssl://smtp.googlemail.com" username="$email"/>
</provider>

<!-- Common US providers -->

<provider id="aim" label="AIM" domain="aim.com">
  <incoming uri="imap://imap.aim.com" label="IMAP" username="$email"/>
  <outgoing uri="smtp://smtp.aim.com:587" username="$email"/>
</provider>

<provider id="aol" label="AOL" domain="aol.com">
  <incoming uri="imap://imap.aol.com" label="IMAP" username="$email"/>
  <outgoing uri="smtp://smtp.aol.com:587" username="$email"/>
</provider>

...
8.7. Themes

• See
  framework/base/core/res/res/values/styles.xml
8.8. Adding new applications

- Add application in packages/apps
- Can use Eclipse to create initial version
- Copy Eclipse project to packages/apps
- Add an appropriate Android.mk file to project
- Add project to PRODUCT_PACKAGES
8.9. Adding new services / new hardware type

- Add your code to:
  frameworks/base/services/java/com/android/server/
- Have the SystemServer.java init+reg. your service
- Define hardware API for apps
- Expose through:
  - frameworks/base/core/java/android/os/[server].aidl
- Call on native “driver” code through JNI
- Implement or connect to appropriate driver
- Create an app that calls on service
- May need to create new SDK ...
8.10. Init

- Android init semantics are different from:
  - System V init
  - Busybox init
- See “Android Init Language” doc in porting guide
- See init.rc examples:
  - Emulator's init.rc
  - device/[manufacturer]/[device]/init.rc
- Global “properties” that can be set and read
- Can be used to tweak low-memory conditions
9. Building the AOSP

- Requires 64-bit Ubuntu 10.04
- Packages required:
  
  ```
  $ sudo apt-get install ia32-libs bison flex gperf \\
  > g++ libia32 libc6-dev-i386 libz-dev libstdc+++6 \\
  > libstdc++6 libstdc++6-32 ia32-libstdc++6 \\
  > ia32-libstdc++ ia32-libstdc++5 ia32-libs \\
  > libncurses-dev lib32ncurses-dev \\
  > ia32-libncurses-dev ia32-libncurses lib32ncurses \\
  > lib32ncurses5-dev
  ```
- Patch build/core/droiddoc.mk
  - https://groups.google.com/group/android-building/browse_thread/thread/833a8159a0e5c56c
Fix a few symbolic links:

$ sudo ln -s /usr/lib32/libstdc++.so.6 /usr/lib32/libstdc++.so
$ sudo ln -s /usr/lib32/libz.so.1 /usr/lib32/libz.so

Set up build environment:

$ . build/envsetup.sh
$ lunch

Launch build and go watch tonight's hockey game:

$ make -j2

... though you should check your screen at breaks ...

Just launch emulator when it's done:

$ emulator &
Some nice tricks:

- See build/envsetup.sh for commands
- Use “lunch” from AOSP root to set env vars
  - You'll need that if you come back later and want to relaunch emulator from AOSP root.
10. Components to write to flash

- See `out/target/product/[product]/*_.img`
- Typically:
  - Bootloader
  - `boot` (kernel and ramdisk)
  - `system` (/system)
  - `userdata` (/data)
11. Useful Embedded Linux tricks

- crosstool-ng
- Busybox
- uClibc
11.1. crosstool-ng

- Cross-development toolchain generator
- Successor to crosstool
- Available at:
  http://ymorin.is-a-geek.org/projects/crosstool
- Downloads, patches, builds, installs, etc.
- Comprises **23** steps
- Menuconfig-based
- Supports uClibc, glibc and eglibc
- Supports ARM, Blackfin, MIPS, PowerPC, SH, …
- Fairly well maintained
11.2. Busybox

- Replicate Linux CLI experience

, [], acpid, add-shell, addgroup, adduser, adjtimex, arp, arping, ash, awk, base64, basename, beep, blkid, blockdev, bootchartd, brctl, bunzip2, bzcat, bzip2, cal, cat, catv, chat, chattr, chgrp, chmod, chown, chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm, cp, cpio, crond, crontab, cryptpw, ctyhyack, cut, date, dc, dd, deallocvt, delgroup, deluser, depmod, devmem, df, dhcrelay, diff, dirname, dmesg, dnsd, dnsdomainname, dos2unix, du, dumpkm, dumpleases, echo, ed, egrep, eject, env, envdir, envuig, ether-wake, expand, expr, fakeidentd, false, fbset, fbSplash, fdflush, fdformat, fdisk, fgconsole, fgrep, find, findfs, flock, fold, free, freeramdisk, fsck, fsck.minix, fsync, ftpd, ftpget, ftpput, fuser, getopt, getty, grep, gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid, hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifenslave, ifplugd, ifup, inetd, init, insmod, install, ionice, iostat, ip, ipaddr, ipcalc, ipcrm, ipcs, iplink, iproute, iptunnel, kbd_mode, kill, killall, killall5, klogd, last, length, less, linux32, linux64, linuxrc, ln, loadfont, loadmap, logger, login, logname, logread, losetup, lpd, lq, lpr, ls, lsattr, lsmod, lspci, lsusb, lzcat, Izma, Izop, Izopcat, makedevs, makemime, man, md5sum, mdev, mesg, microcom, mkdir, mkdosfs, mke2fs, mkfifo, mkfs.ext2, mkfs.minix, mkfs.vfat, mknt, mkpassw, mkswap, mktemp, modinfo, modprobe, more, mount, mountpoint, mpstat, mt, mv, nameif, nbd-client, nc, netstat, nice, nmeter, nohup, nslookup, ntpd, od, openvt, passwd, patch, pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill, pmap, popmaiddir, poweroff, powertop, printenv, printf, ps, pscan, pwd, raidautorun, rdate, rdev, readahead, readlink, readprofile, realpart, reboot, reformime, remove-shell, renice, reset, resize, rev, rm, rmdir, rmmdev, route, rpm, rpm2cpio, rtcwake, run-parts, runlevel, runsv, runsvdir, rx, script, scriptreplay, sed, sendmail, seq, setarch, setconsole, setfont, setkeycodes, setlogcons, setsid, setuidgid, sh, sha1sum, sha256sum, sha512sum, showkey, slattach, sleep, smemcap, soflimit, sort, split, start-stop-daemon, stat, strings, sty, su, sulogin, sum, sv, svlogd, swapoff, swapon, switch_root, sync, sysct, syslogd, tac, tail, tar, tcpv, tee, telnet, telnetd, test, tftp, ftp, ftp, time, timeout, top, touch, tr, traceroute, traceroute6, true, tty, tty, tsize, tunc1, udhcpc, udhcpcd, udpsv, umount, uname, unexpand, uniq, unix2dos, unlzma, unlzop, unx, unzip, updown, usleep, uudecode, uuencode, vconfig, vi, vlock, volname, wall, watch, watchdog, wc, wget, which, who, whoami, xargs, xz, xzcat, yes, zcat, zcip
Some features of interest:

- color-coded file lists
- tab completion
- "home", "end"
- grep, sed, wc, more, less
- vi
- ifconfig
- httpd
- sendmail
- tftp
- top
- ...


- Download BusyBox (1.18.3)
- Move to the directory for the rest of the setup:
  
  ```
  $ cd busybox-1.18.3
  ```
- Configuration of BusyBox's options:
  
  ```
  $ make menuconfig
  ```
- Options that must be set:
  - “Build Options” -> “Do you want to build BusyBox with a Cross Compiler?”
  - Cross-compiler prefix: `arm-unknown-linux-gnueabi-`
  - “Installation Options” -> “Don't use /usr”
  - Installation prefix: `${PRJROOT}/rootfs`
- Build:
  
  ```
  $ make
  ```
- Install:
  
  ```
  $ make install
  ```
Cheat sheet:

Commands to get the new Busybox onto the rootfs:

$ adb shell mount -o remount,rw rootfs /
$ adb shell mkdir /bin
$ adb push busybox /bin/
$ adb shell /bin/busybox --install /bin
$ adb shell

To do after going into the shell:

# /bin/ash
# export PATH=/bin:$PATH
11.3. uClibc

- Originates from uClinux effort
- Support both CPUs that have and those that lack an MMU and/or an FPU.
- Allows both static and dynamic linking
- Most applications that build with glibc will build and work the same with uClibc.
- Available from: http://uclibc.org/
Thank you ...

karim.yaghmour@opersys.com

www.opersys.com