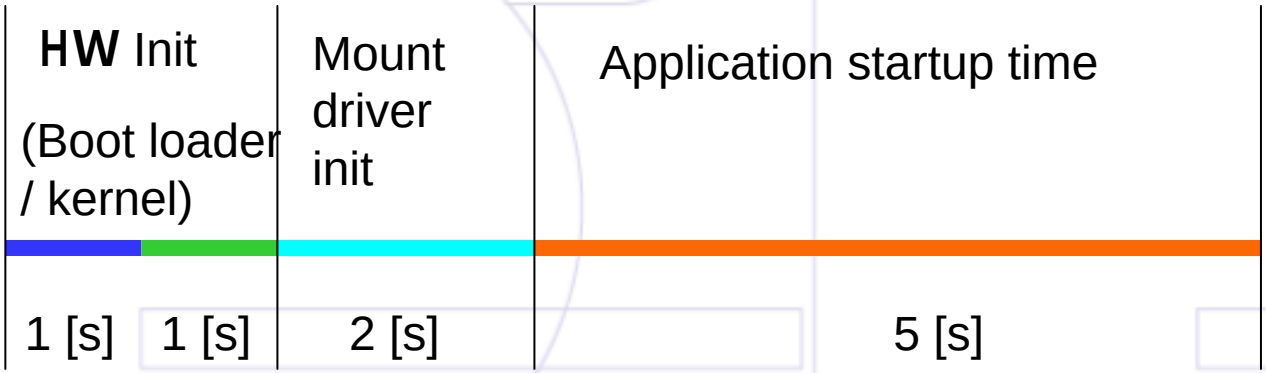
Improving Linux Startup Time Using Software Resume

Hiroki Kaminaga
kaminaga (at) sm.sony.co.jp

$ who am i

- Work for Sony Corporation
  - Corporate Advanced Technology Development
  - Provide Linux to product development teams
    - TV
    - Mobile (battery powered device)
    - Video Recorder
    - Etc.

The Problem of System Startup Time



- System startup done before application is ready
  - Transfer kernel image and userspace pages to RAM
  - Linking dynamic shared library
  - C++ global constructor execution
  - Application startup time IPC
- Application startup time dominates system startup time

Existing Methods of Reducing Startup Time

- Prelinking
  - Pre-calculate dynamic relocation
  - Reduces dynamic link processing time
  - Only fixes one factor, and increases binary size
- XIP (eXecute In Place)
  - Executes directly from NOR flash or ROM
  - Reduces the copy of kernel and userspace pages
  - Slow execution, limited by system design

The Need for Snapshot Boot

- Existing methods only address individual parts of the problem, and do not provide a complete solution
- The proposed snapshot boot method tries to provide a comprehensive solution to the startup time problem
- Configuration rarely changes in embedded systems.

What is Snapshot Boot

- Snapshot boot uses the existing system resume to reduce application startup time by loading a fixed, pre-made system image to RAM.
- Snapshot boot uses boot loader/kernel cooperation to minimize the system initialization needed prior to resume
- Snapshot boot provides a comprehensive solution

Why Utilize Resume

Normal Start up time

- Σ copy text/data to RAM
- Σ I/O init
- Σ run time state init
  - IPC sync/wait/sched
  - Dynamic link
  - Global Constructor execs
  - Other runtime init.

Ideal Resume time

- Σ copy text/data to RAM
- Σ I/O init
- Σ transfer state to RAM
  - Heap
  - Stack

- Typically (run time state init) >> (transfer state to RAM)

Preparations

- Before the details of Snapshot Boot…
- Introduction to software suspend/resume (swsusp)
  - Already in 2.6 kernel
  - Documentation/power/swsusp.txt

Suspend Methods in Linux

- Standby
- Suspend to RAM
- Suspend to Disk
  - writes runtime state to system image on non-volatile storage.
  - resumes pre-suspended state from system image

Suspend to Disk Demo

Target Environment

- Target board
  - OMAP Starter Kit (OSK5912)
- Boot loader
  - U-boot 1.1.4
- OS
  - Linux 2.6.11
- Application
  - mplayer

Hardware Features:
ARM9 core operating at 192 Mhz.
DSP core operating at 192 Mhz.
TLV320AIC23 Stereo Codec
32 Mbyte DDR SDRAM
32 Mbyte Flash
RS-232 Serial Port
10 MBPS Ethernet port
…

Target Adaptation

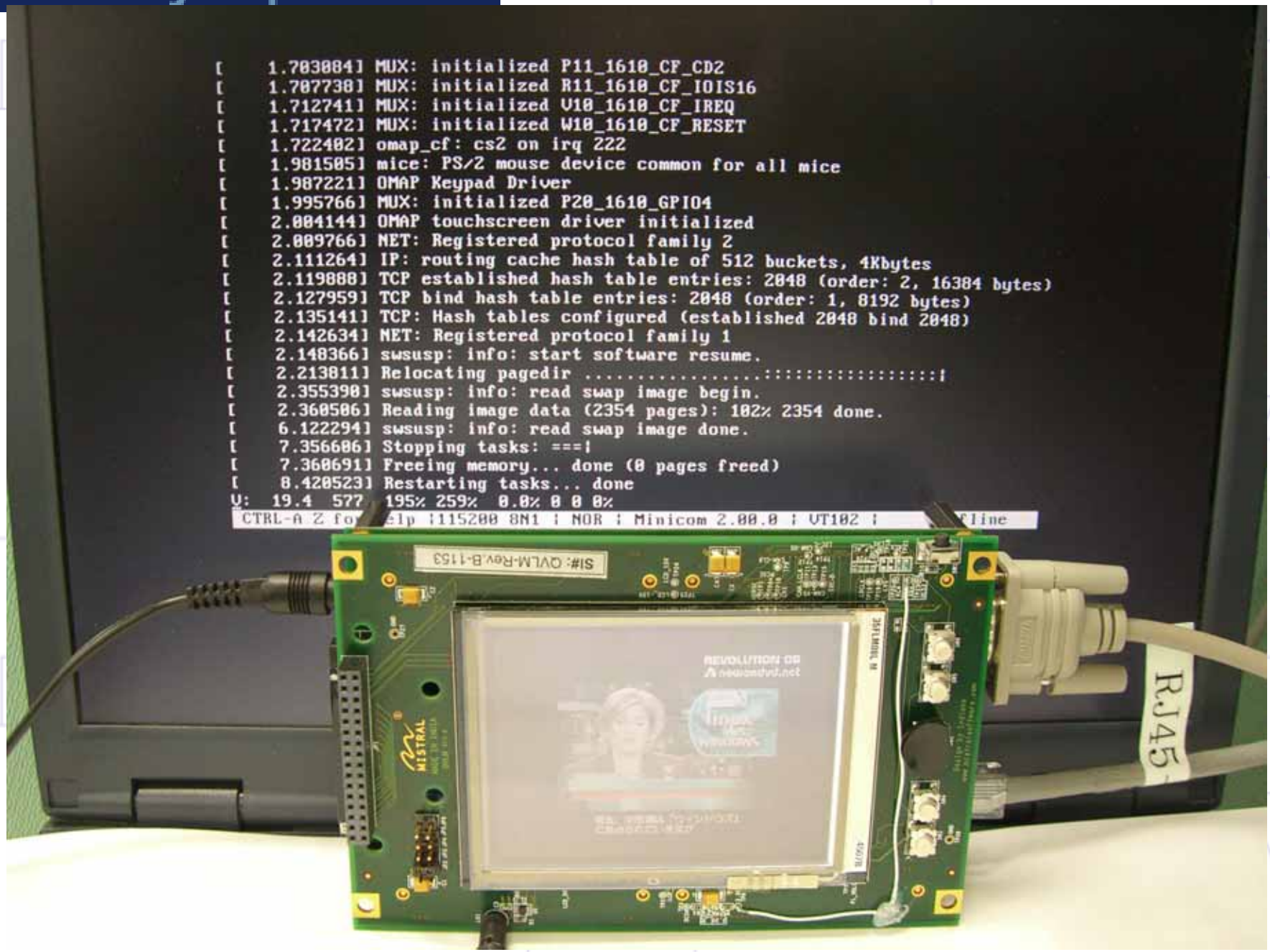
- Flash ROM is used as non-volatile storage
- Porting to ARM was needed

Details of Suspend to Disk

- `freeze_processes()`
- `free_some_memory()`
- `device_suspend()`
- `device_power_down()`
- `save_processor_state()`
- Make snapshot image (in memory)
- `device_power_up()`
- `device_resume()`
- `write_suspend_image()`
- Powerdown machine

Details of Resume From Disk

- `software_resume()` called at `late_initcall`
- Read system image from storage to RAM
- `freeze_processes()`
- `device_suspend()`
- `device_power_down()`
- Restore system image
- `restore_processor_state()`
- `device_power_up()`
- `device_resume()`
- `thaw_processes()`


[1.703084] MUX: initialized P11_1610_CF_CD2
[1.707738] MUX: initialized R11_1610_CF_IOIS16
[1.712741] MUX: initialized V10_1610_CF_IREQ
[1.717472] MUX: initialized W10_1610_CF_RESET
[1.722402] omap_cf: cs2 on irq 222
[1.981505] mice: PS/2 mouse device common for all mice
[1.987221] OMAP Keypad Driver
[1.995766] MUX: initialized P20_1610_GPIO4
[2.004144] OMAP touchscreen driver initialized
[2.009766] NET: Registered protocol family 2
[2.111264] IP: routing cache hash table of 512 buckets, 4Kbytes
[2.119888] TCP established hash table entries: 2048 (order: 2, 16384 bytes)
[2.127959] TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
[2.135141] TCP: Hash tables configured (established 2048 bind 2048)
[2.142634] NET: Registered protocol family 1
[2.148366] swsusp: info: start software resume.
[2.213811] Relocating pagedir .....:::::::::::::::::|
[2.355398] swsusp: info: read swap image begin.
[2.360506] Reading image data (2354 pages): 102% 2354 done.
[6.122294] swsusp: info: read swap image done.
[7.356686] Stopping tasks: ===|
[7.360691] Freeing memory... done (0 pages freed)
[8.420523] Restarting tasks... done
V: 19.4 577 195% 259% 0.0% 0 0 0%
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 | Offline
SI#: QVLM-Rev.B-1153
MISTRAL
REVOLUTION OS
RJ45

Why Resume From Disk Takes So Long

- Resume processing is done after almost all normal kernel startup is done
  - Devices used for resume startup are then shutdown
  - Devices are then re-initialized by the resume code
- System image copy is done twice
  - storage to working buffer
  - working buffer to final address
- Process freeze takes time

Snapshot Boot: Boot Loader Detail

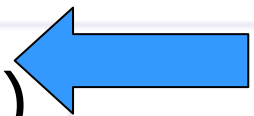
- Boot loader and kernel cooperate
- Procedure on boot loader side:
  - Wakeup board
  - Copy system image to RAM
  - Minimal device setup for resume
  - Jump to kernel resume point

Usual procedure

Added procedures

Snapshot boot: Kernel Detail

- `software_resume()` called at `late_initcall`
- Read system image from storage to RAM
- `freeze_processes()`
- `device_suspend()`
- `device_power_down()`
- Restore system image
- `restore_processor_state()`
- `device_power_up()`
- `device_resume()`
- `thaw_processes()`

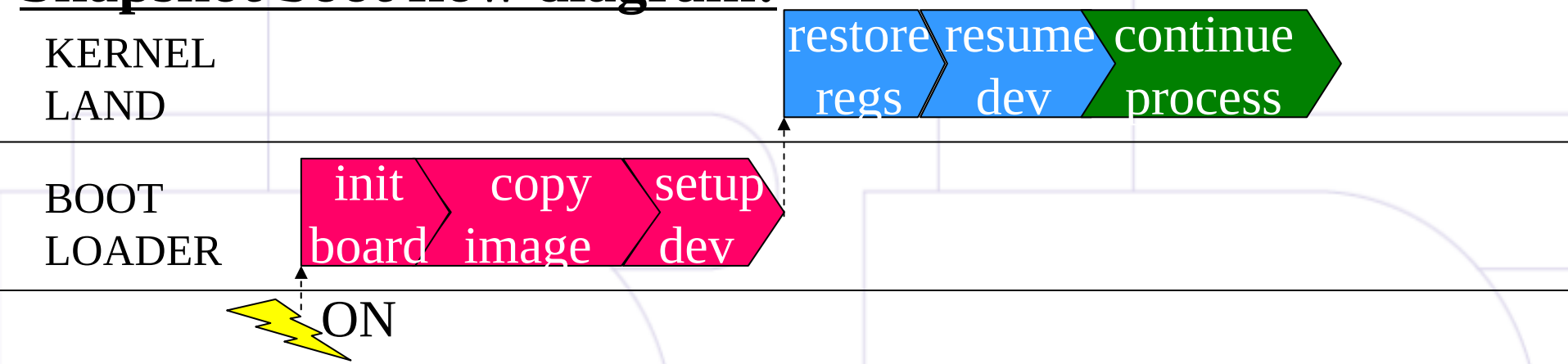


Jump to here

Snapshot Boot: Flow Diagrams

Resume from disk flow diagram:

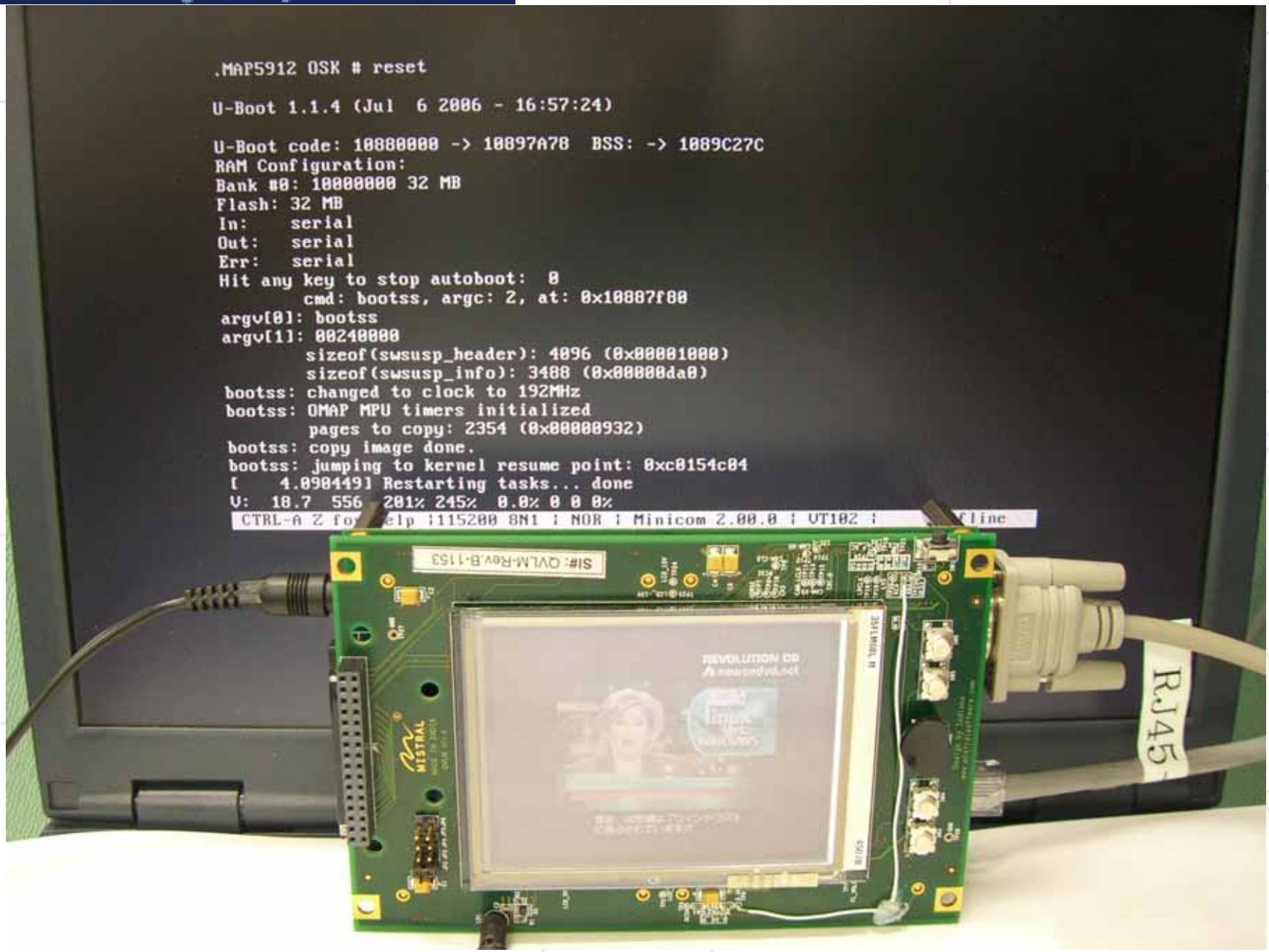
Snapshot boot flow diagram:



Snapshot Boot: Demo

Snapshot Boot: Implementation

- Boot loader support implemented as new command in u-boot
  - bootss <image addr>
- Boot loader startup sequence:
  - Setup clock speed, timer, MMIO regs
  - Copy system image from flash to RAM
  - Setup MMU
  - Jump to kernel resume point
- Minor kernel modification
  - Added resume entry point, enabled interrupts


.MAP5912 OSK # reset
U-Boot 1.1.4 (Jul 6 2006 - 16:57:24)
U-Boot code: 10880000 -> 10897A78 BSS: -> 1089C27C
RAM Configuration:
Bank #0: 10000000 32 MB
Flash: 32 MB
In: serial
Out: serial
Err: serial
Hit any key to stop autoboot: 0
cmd: bootss, argc: 2, at: 0x10887f80
argv[0]: bootss
argv[1]: 00240000
sizeof(swsusp_header): 4096 (0x00001000)
sizeof(swsusp_info): 3488 (0x00000da0)
bootss: changed to clock to 192MHz
bootss: OMAP MPU timers initialized
pages to copy: 2354 (0x00000932)
bootss: copy image done.
bootss: jumping to kernel resume point: 0xc0154c04
[4.090449] Restarting tasks... done
V: 18.7 556 201% 245% 0.0% 0 0 0%
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |
MISTRAL
REVOLUTION OS
RJ45

Some Issues Faced

- Many drivers don't properly implement resume, and just rely on the initialization done at startup.
  - Current snapshot boot implements device initialization at boot loader as a workaround.
  - Similar issue in kexec too…
- System image format changed in recent kernel

Conclusion

- Successfully implemented snapshot boot feature
- Reduced system startup time by 50%
- Implemented Suspend to Disk for ARM

More Information

- Suspend to Disk and snapshot boot for ARM wiki page is available at CE Linux Forum website

http://tree.celinuxforum.org/CelfPubWiki/SuspendToDiskForARM

- Target board wiki page

http://tree.celinuxforum.org/CelfPubWiki/OSK

- Come to CELF Project BOF meeting Fri, 7pm, at Les Suites