

Case Studies in Linux Ports to Embedded Platforms

Claudio Matsuoka
Gustavo Boiko
Thiago Galesi

Mandriva Embedded Systems Labs
Curitiba – Brazil

WSL2006 - VII Free Software Workshop

Outline

1 ARM-based handheld device

- Project description
- Development and runtime environment
- The bootloader

2 PowerPC-based PBX system

- Project description
- Development and runtime environment
- The problems

3 Conclusions & Questions

Outline

1 ARM-based handheld device

- Project description
- Development and runtime environment
- The bootloader

2 PowerPC-based PBX system

- Project description
- Development and runtime environment
- The problems

3 Conclusions & Questions

Device overview



Features

- Rugged handheld device
- 400MHz XScale PXA263
- 64MB RAM, 64MB flash
- SD and CF card reader
- Bluetooth
- ISP1161 USB controller
- S1D13806 display controller
- WM97 touch panel
- WindowsCE 4.20

Project details

Project facts

- Port commissioned by an Integrator
- Good accessibility to the Integrator
- Integrator is cool about licensing
- Some device driver source code supplied
- Two-phase project: **demo** and **final**

BUT...

- No direct contact with the manufacturer
- No architecture documentation
- Reverse engineering needed

Project details

Project facts

- Port commissioned by an Integrator
- Good accessibility to the Integrator
- Integrator is cool about licensing
- Some device driver source code supplied
- Two-phase project: **demo** and **final**

BUT...

- No direct contact with the manufacturer
- No architecture documentation
- Reverse engineering needed

Project goals

Demo phase goals

- Bootable Linux kernel
- Qt-based application
- Wireless networking

Which means support to:

- S1D13806 display controller
- WM97 touch panel
- SD card (for storage)
- CompactFlash (for the Wi-Fi card)
- Other: Bluetooth, keypad

Project goals

Demo phase goals

- Bootable Linux kernel
- Qt-based application
- Wireless networking

Which means support to:

- S1D13806 display controller
- WM97 touch panel
- SD card (for storage)
- CompactFlash (for the Wi-Fi card)
- Other: Bluetooth, keypad

Software used

Cross-development tools

- GNU toolchain with GCC 3.4.3
- Embedded Visual C++ (!) under Wine
- uClibc buildroot

Runtime environment

- Kernels 2.6.12 and 2.6.14.2
- uClibc
- Busybox for almost everything
- Qt/Embedded, Qtopia and Opie

Software used

Cross-development tools

- GNU toolchain with GCC 3.4.3
- Embedded Visual C++ (!) under Wine
- uClibc buildroot

Runtime environment

- Kernels 2.6.12 and 2.6.14.2
- uClibc
- Busybox for almost everything
- Qt/Embedded, Qtopia and Opie

Booting Linux

The problem

Architecture docs insufficient to prepare a bootloader!

Additionally:

- No JTAG
- No serial (or ethernet)

Solution

Boot Linux from WindowsCE (but how?)

This also allows us to:

- Keep WinCE on device for reference
- Easily try new kernels without firmware updates
- Some subsystems are already initialized

Booting Linux

The problem

Architecture docs insufficient to prepare a bootloader!

Additionally:

- No JTAG
- No serial (or ethernet)

Solution

Boot Linux from WindowsCE (but how?)

This also allows us to:

- Keep WinCE on device for reference
- Easily try new kernels without firmware updates
- Some subsystems are already initialized

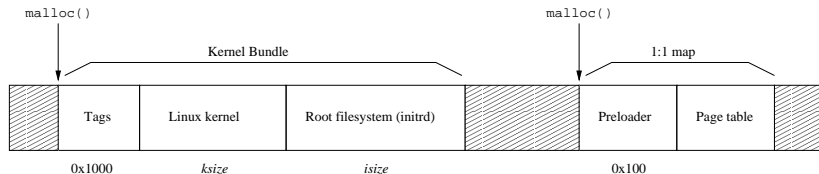
HaRET

HaRET to the rescue

- **H**andheld **R**everse **E**ngineering **T**ool
- It's a native WindowsCE application
- Built with eVC++ (port to gcc would be nice!)
- Performs virtual/physical memory translations
- Allows peeks & pokes, memory dumps, etc.
- **B**oots **L**inux from inside **W**indows**C**E (like linload)

The bootloader

HaRET

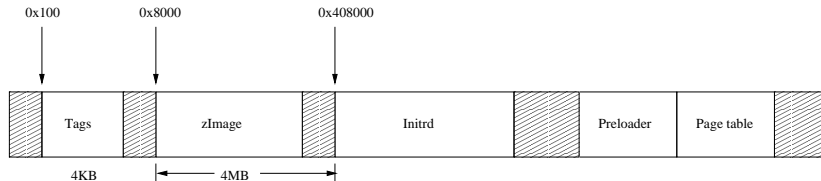


How HaRET works

- 1 Loads the kernel bundle
- 2 Installs preloader in physically contiguous RAM
- 3 Get ring0 privileges (!)
- 4 Stop WinCE tasks
- 5 Transfer control to preloader

The bootloader

HaRET



How HaRET works

- 1 Use private page table with 1:1 mapping
- 2 Take a deep breath, cross fingers, etc.
- 3 **Stop MMU** (with code running)
- 4 Copy linux to the proper location
- 5 Execute the Linux decompressor

Outline

1 ARM-based handheld device

- Project description
- Development and runtime environment
- The bootloader

2 PowerPC-based PBX system

- Project description
- Development and runtime environment
- The problems

3 Conclusions & Questions

System overview

Hardware

- Private branch exchange system
- 48MHz PowerQUICC
- 32MB RAM, 8MB NOR flash (raw)
- UART, PHY
- Watchdog

Software

- Application runs on bare metal
- Deals with time-critical ISDN signaling

System overview

Hardware

- Private branch exchange system
- 48MHz PowerQUICC
- 32MB RAM, 8MB NOR flash (raw)
- UART, PHY
- Watchdog

Software

- Application runs on bare metal
- Deals with time-critical ISDN signaling

Project details

Project facts

- Port commissioned by the manufacturer
- Requires real-time capabilities
- Involves porting of proprietary code
- Two steps: **Linux port** and **application port**

BUT...

- Poor architecture documentation
- Manufacturer is restrictive about licensing
- No contact with engineering team
- Nobody knows how the application works

Project details

Project facts

- Port commissioned by the manufacturer
- Requires real-time capabilities
- Involves porting of proprietary code
- Two steps: **Linux port** and **application port**

BUT...

- Poor architecture documentation
- Manufacturer is restrictive about licensing
- No contact with engineering team
- Nobody knows how the application works

Software used

Cross-development tools

- GNU toolchain with GCC 3.4.4
- uClibc buildroot

Runtime environment

- U-Boot
- Kernel 2.6.9
- RTAI/Fusion
- uClibc
- Busybox
- Manufacturer's application

Software used

Cross-development tools

- GNU toolchain with GCC 3.4.4
- uClibc buildroot

Runtime environment

- U-Boot
- Kernel 2.6.9
- RTAI/Fusion
- uClibc
- Busybox
- Manufacturer's application

Port problems

What went wrong

- Existing support for board was nonfunctional
- CPU has strange cache issues
- Hardware doesn't seem very stable
- Tight (and bad) schedule, high pressure
- C++ application not written with portability in mind
- Real-time system hard to debug
- Application uses writable mmap (doesn't work with JFFS2)
- Limited RAM and flash size

Outline

1 ARM-based handheld device

- Project description
- Development and runtime environment
- The bootloader

2 PowerPC-based PBX system

- Project description
- Development and runtime environment
- The problems

3 Conclusions & Questions

Summary

Conclusions

- Official ports != documentation
- Test solutions, don't assume that they work because someone said so!
- Be prepared for reverse engineering
- HaRET: invaluable tool to port linux to ARM PDAs
- It should be possible to port it to SH, MIPS, etc
- Port Linux to ARM PDAs is **easy**. Have everything working is **hard**.
- Git is your friend!
- If you have MTD, use JFFS2
- If you have MTD, don't use writable mmap