



Getting started with embedded linux

- from PCB to linux platform

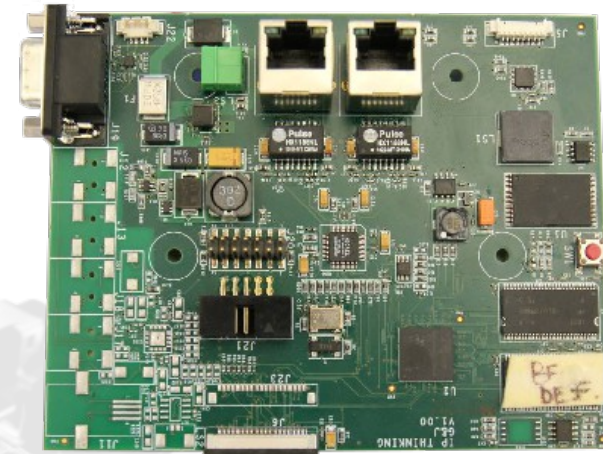
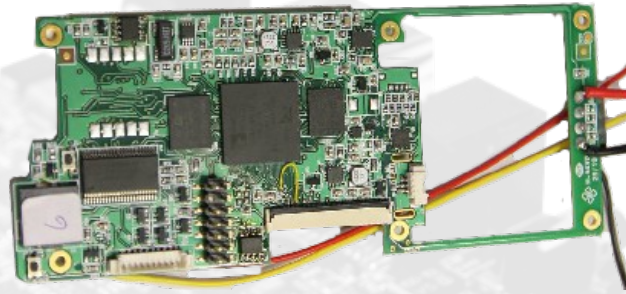
Agenda:

- Who/What is IP Thinking A/S
- Embedded linux overview
- Toolchain
- Bootloader
- Kernel
- User applications
- Debugging
- Complete system images

by
Mark Urup
IP Thinking A/S



About IP Thinking A/S



What we do

Hardware

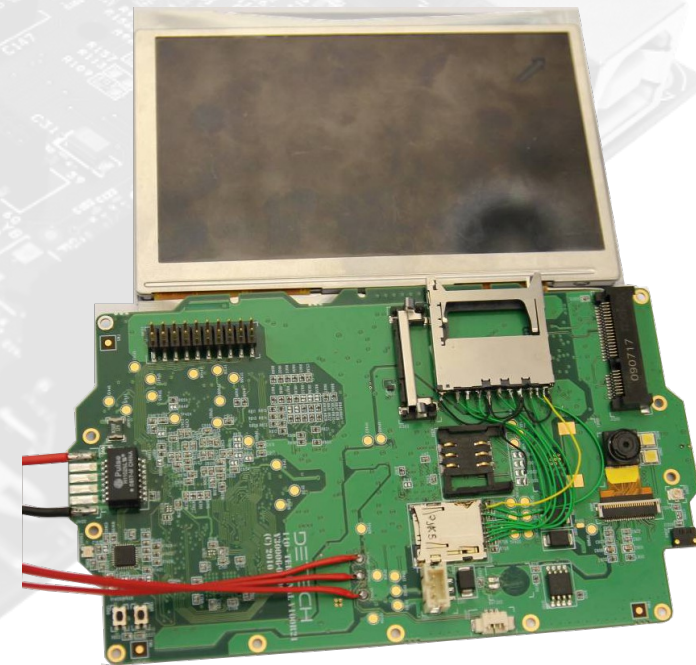
- Design platforms for embedded linux
 - Analog devices Blackfin 5xx DSP's
 - Freescale iMX processors

Software

- Bootloader customization & board bring-up
- (uC)linux customization & drivers
- Userspace applications

Cases

- CAREinfo (BF537)
- Heta brændeovne (BF522)
- Mikroting (iMX.51)
- Gluns & Jensen (BF537/BF536)





Embedded linux overview

Typical embedded system

- Small size
- No kernel modules (all built in)
- Limited (if any) system logging
- Busybox (all system utilities in one app)
- Limited (if any) user management
- non-MMU processors
- Limited amount of RAM (8-128MB)
- Simple MMI

MMU: A memory management unit (MMU), is a computer hardware component responsible for handling accesses to memory requested by the CPU

MMI: Man-Machine Interface



Embedded linux overview - uClinux

uClinux distribution

- Configured via menuconfig
 - Target selection
 - Kernel features
 - Userspace apps & libraries

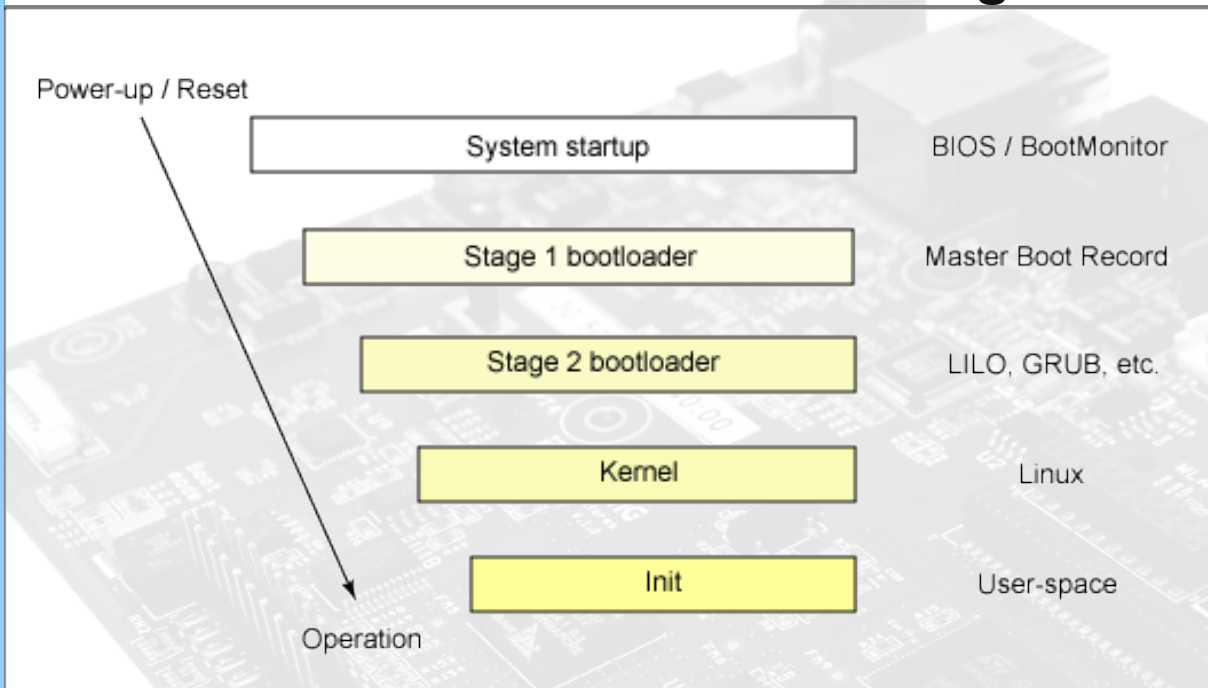
System images starting at ~1MB in size
Boot time as low as 2 seconds

Kernel version 2.4.x or 2.6.x

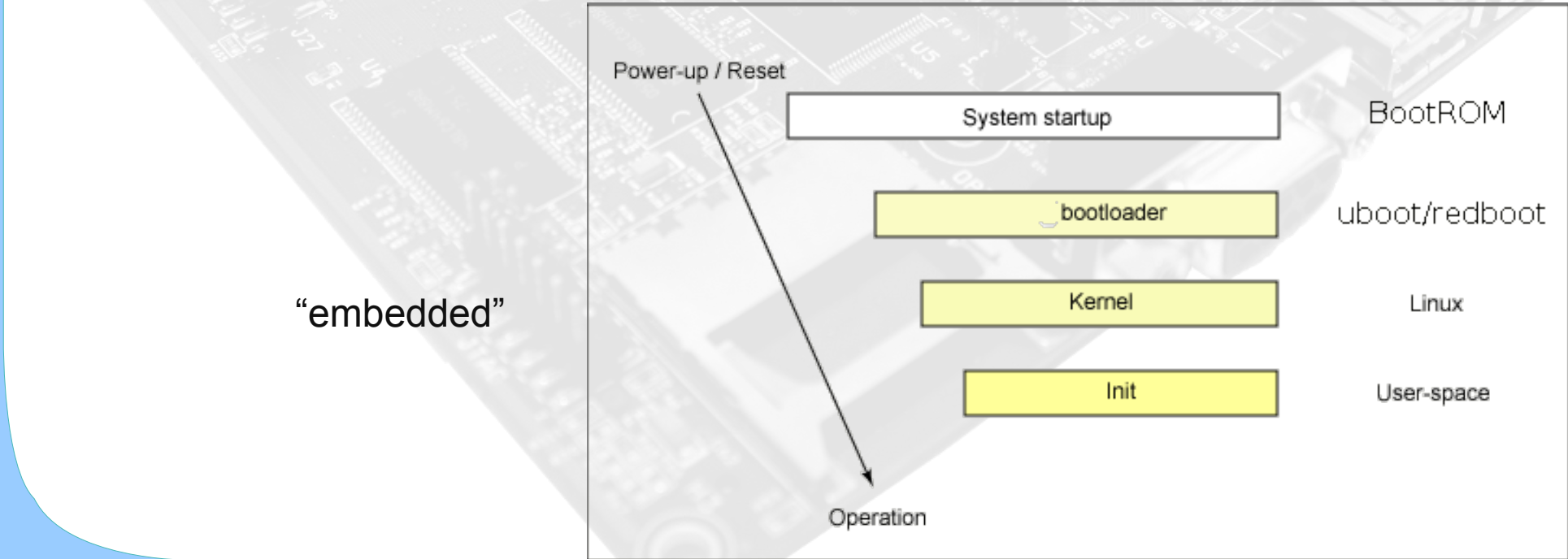
Pronounced "you-see-linux", the name uClinux comes from combining the greek letter "mu" and the english capital "C". "Mu" stands for "micro", and the "C" is for "controller".



Embedded linux overview - booting



“normal pc”



“embedded”



Toolchain

GNU toolchain

- The GNU Binutils, which is a collection of binary tools, the main ones being as (the GNU assembler) and ld (the GNU Linker).
- The GNU Compiler Collection (gcc), which includes front ends for C, C++ and Fortran.
- The GNU Debugger (gdb)
- The generation of uClinux's flat format - elf2flt
- Tools to support bare metal application development and booting
- Libraries, including libdsp, newlib, libgloss and uClibc.

A toolchain is the set of programming tools that are used to create a product (typically another computer program or system of programs). The tools may be used in a chain, so that the output of each tool becomes the input for the next, but the term is used widely to refer to any set of linked development tools.



Bootloader

Das u-boot

- USB support
- SPI flash support
- NAND flash support
- SD/MMC card support
- Network support
- Video support
- UART support

Portability

- Runs on many platforms
- Good documentation

Das U-Boot (Universal Bootloader) is a boot loader for a number of different computer architectures, including PPC, ARM, AVR32, MIPS, x86, 68k, Nios, and MicroBlaze. Its name comes from the abbreviated form of Das Unterseeboot (German for "the submarine").



Bootloader – u-boot - porting to new hardware

What it does

It is the job of U-Boot to initialize all the low-level hardware details. So you're going to need to know the hardware settings, and have your Hardware Reference Manual [HRM] at hand.

Steps

- Create new board files & integrate these into u-boot
- Define CPU, memory and other peripherals
- Define boot-device (make sure it's supported)

Booting for the first time

Boot via UART or load it to memory by JTAG, and run it from there, or have it pre-loaded into your boot device.



Bootloader – u-boot - porting to new hardware

... and when it doesn't boot the first time

- Verify your hardware – use an oscilloscope
- Verify your memory settings
- Verify your CPU settings
- Verify your bootmode
- Hook up your JTAG
- Turn on early debugging output



Kernel

What it does

It provides true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multistack networking.

And a lot more...

How it does this

The linux kernel can be extended by modules. These are often hardware drivers, that provide a arbitration layer from the hardware.

The uClinux kernel, as any other linux distribution is configured by “make menuconfig”.



Kernel - porting

Porting the kernel to your board

- Create target platform (system type)
- Select modules to use
- Define module resources

Defining resources

Normally a module will probe (try to find the hardware itself). Sometimes this can't be done however, so when porting uClinux you also define resources for your modules.



Kernel – creating new modules

Kernel API

The best place to look is `<u>Clinux-dist>linux-2.6.x/include/linux/`
And also `<u>Clinux-dist>linux-2.6.x/Documentation/`

Reuse code

Maybe (most likely) you're doing something similar to something somebody else has already done.

Be inspired; you don't need to invent stuff that already exists.

Make it reusable

Don't write code that uses fixed hardware resources, or makes platform dependent calls if you can avoid it.

Resources

Remember to free all and any resources you've used.

Also remember that your driver might not load properly, and even if it fails, it should clean up after itself (nobody else does).



Userspace applications & libraries

Selecting userspace applications

Since you're building a complete system image, you'll need to select which applications you want in your image.

In uClinux this is also done via a console based menu system (menuconfig).

Important applications

- Busybox
- Init
- Shell (for testing)

Adding custom applications

You can add your own applications to the menu system, so they're easy to include.



Complete system image

The image

The complete system image is called “ulmage”.
It is a combination of the kernel and userspace applications.

Booting the image

When booting ulmages, u-boot will load it into memory, and start executing it from there.
All control is handed over to linux, and the memory occupied by u-boot will be reclaimed and used by linux.
It is also possible to use the two images separately, if desired.

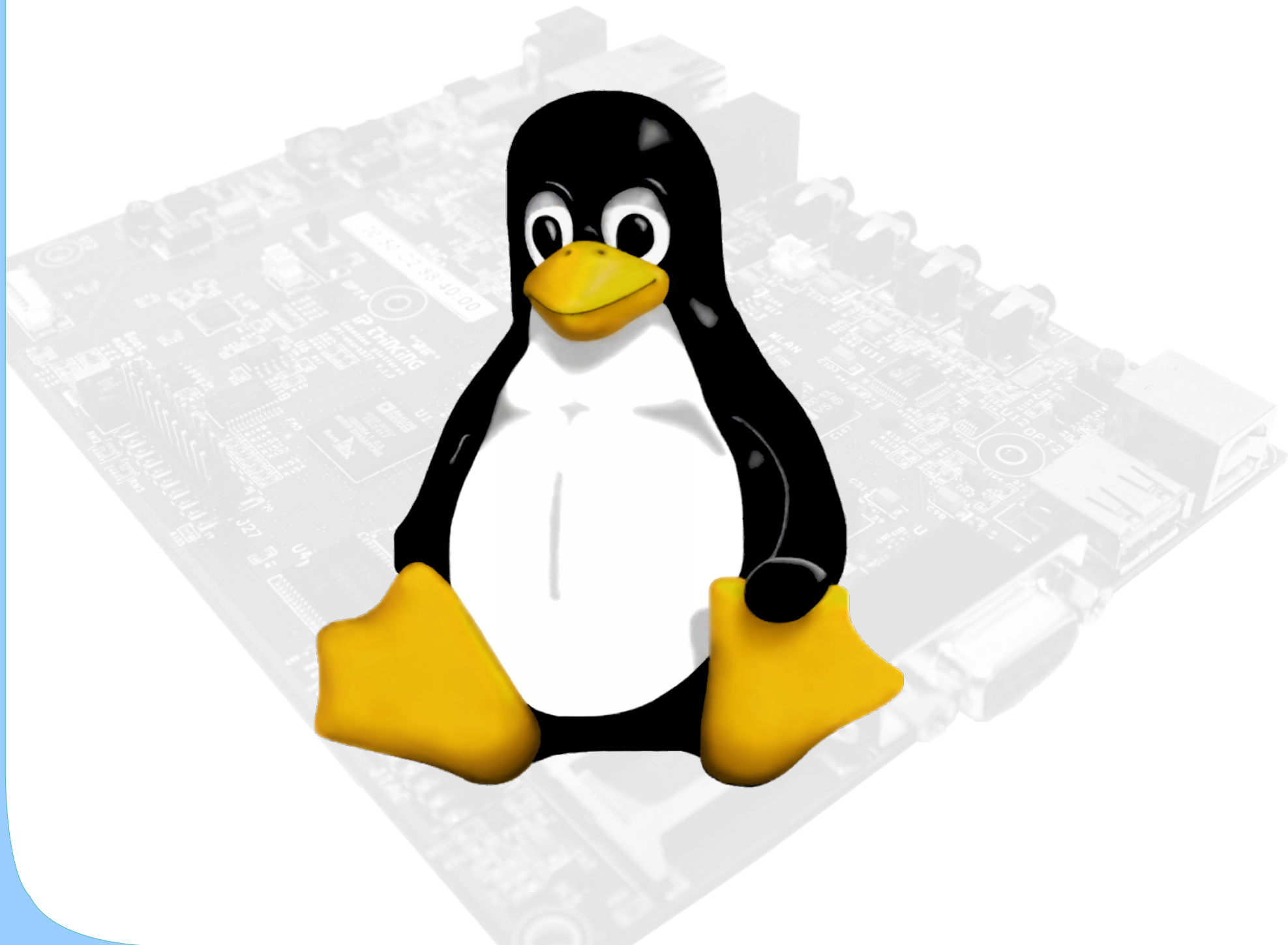


Debugging

Debug options

- JTAG (kernel & userspace)
- gdb (userspace)
- printf (userspace)
- printk (kernel)

Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus making it behave as expected.





Links

Das uboot: <http://sourceforge.net/projects/u-boot/>

uClinux main page: <http://uclinux.org/>

Analog devices uClinux distribution: <http://blackfin.uclinux.org/>

Freescale Semiconductors: <http://freescale.com>

IP Thinking: <http://ipthinking.dk>

Books:

Linux Device Drivers, Third Edition (free): <http://lwn.net/Kernel/LDD3/>