LinuxBIOS presentation

Christer Weinigel, Weinigel Ingenjörsbyrå AB
www.weinigel.se

Doing Linux programming close to the hardware

Presentation overview

Three parts:

1) Introduction to LinuxBIOS and concepts

2) Porting LinuxBIOS to the Nano Computer

3) Questions

What is LinuxBIOS

Open Source firmware for PC systems replacing a normal PC BIOS

Started by Ronald Minnich in 1999 at the Los Alamos National Laboratory (LANL)

Purpose:
Make it easier to manage large computing clusters

Well suited for embedded systems: smaller, faster, less complex and cheaper than a PC BIOS

What does a PC BIOS do?

Initializes the hardware

Loads the operating system

Provides a simple I/O interface to the operating system and applications:

- INT 10h - Video
- INT 13h - Disk
- INT 15h - System information

Today operating systems have their own drivers.
BIOS is only used to load the OS.

Hardware initialization

Configure the CPU and the chipset

Determine memory size

Configure onboard devices:
(Super I/O, keyboard, floppy etc.)

Scan PCI bus and allocate resources for PCI devices.
Also calls any expansion BIOS on a PCI card.

Probe for IDE hard disks

Loading Linux

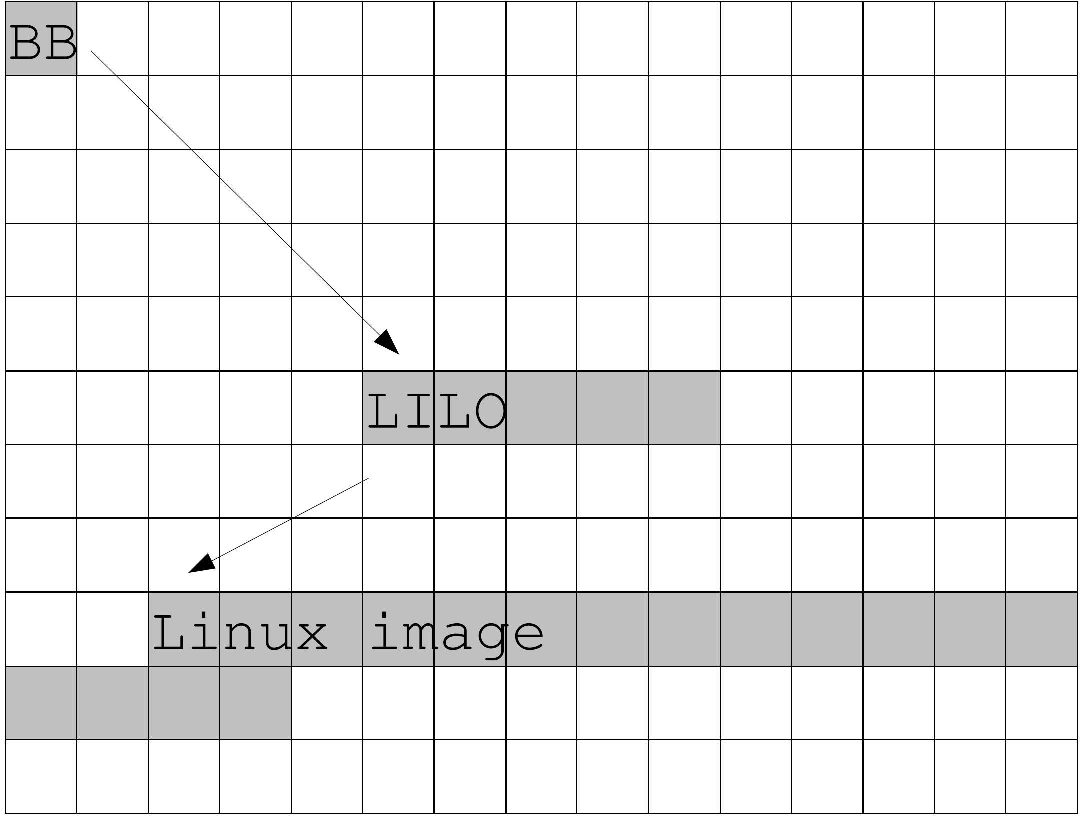
BIOS loads the boot block.

Boot block loads rest of LILO using INT 13h calls

LILO loads kernel image

Jumps to kernel 16 bit entry point

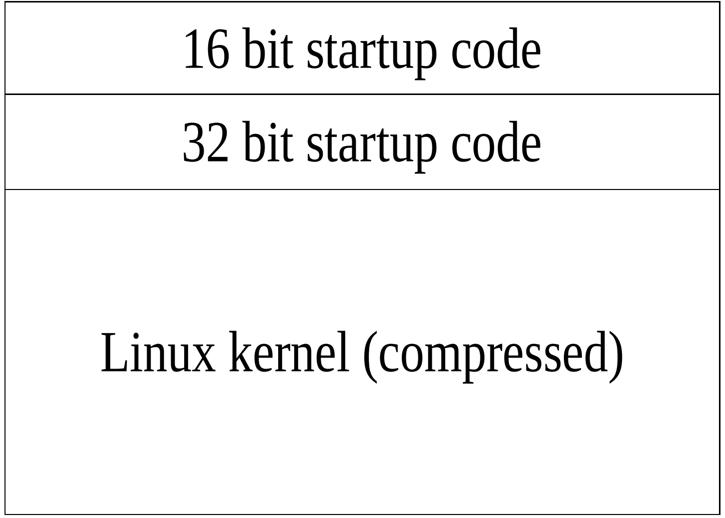
Disk layout



The Linux image

The 16 bit startup code does BIOS INT 10h and INT15h calls to determine the video mode and the amount of memory.

Linux image structure



16 bit startup code

32 bit startup code

Linux kernel (compressed)

Switches to 32 bit mode, uncompresses the Linux kernel and jumps to the kernel entry point.

Linux kernel initializes hardware:

Scan PCI bus, fix up after buggy BIOS

Probe for IDE hard disks

Disadvantages of a PC BIOS

Much duplicated functionality:

- PCI scan done twice
- Slow IDE probe done twice
- Drivers in BIOS, drivers in OS

No remote management

- Screen and keyboard needed to do configuration

Large and complex due to backward compatibility

Often has bugs

The LinuxBIOS way

Only do minimal hardware initialization to get the CPU and chipset started and enter 32 bit mode as soon as possible.

Copy the Linux kernel from flash to RAM

Jump to the Linux kernel entry point

Let Linux do the rest of the hardware initialization

Use a full Linux system to load the OS

Milestones

Ronald Minnich started LinuxBIOS September 1999

Got Linux to boot another Linux

Started on hardware support. Finally SiS, a chipset maker, got involved and helped with the hardware support for their chipsets.

First booted Linux from LinuxBIOS May 2000

LinuxBIOS today

LinuxBIOS has evolved since the beginning and has become more complex.

The hardware initialization does a PCI scan and can probe for IDE hard drives.

Fills in the **LinuxBIOS table** with information that the operating system will need. Memory size etc. Gives better separation from the OS.

Loads the OS into memory and jump to its entry point.

Operating systems

In return for the complexity, LinuxBIOS can do more. It is now possible to boot other operating systems such as:

- Windows CE
- Plan 9
- memtest86 - a memory tester
- Etherboot - a BOOTP/TFP client

As long as the OS can get its information from the LinuxBIOS table it will work. And LinuxBIOS still fits in a 32kByte ROM

Comparison

LinuxBIOS	PC BIOS
– Small <32kByte	– Large, 256 kByte
– Fast	– Slow
– Customizable	– Rigid
– Mostly written in C, clean and portable	– Much assembly language, complex
– No license fees	– Per unit license fees
– Currently lags the hardware development	– Better manufacturer support
– Nonstandard	– De facto standard

Supported mainboards and chipsets

Download the latest source from:
http://freebios.sourceforge.net/

To get the list of supported mainboards:

```
ls freebios/src/mainboard/*/*
```

To get a list of supported chipsets:

```
ls freebios/src/*bridge*/*/*
```

List of mainboards and chipsets

Mainboards:

advantech	pcm-5823	advantech	pcm-9574
asus	a7m	asus	cua
bcm	e100	cocom	voyager2
compaq	ds10	dell	350
digitallogic	smartcore-p5		
elitegroup	k7sem	generic	serverworks
gigabit	ga-6bxc	gigabit	ga-6oxe
ibm	t23		
intel	l440bx	intel	l440gx
irobot	proto1	lanner	em-370
leadtek	winfast6300	lippert	roadrunner2
matsonic	ms7308e	nano	nano
pcchips	m754lmr	pcchips	m758lmr+
pcchips	m810lmr		
rcn	dc1100s	rlx	800i
sis	540	sis	550
sis	635	sis	735
supermicro	p4dc6	supermicro	p4dc6p
supermicro	p4dpr		
supertek	st3wt		
technoland	sbc710		
tyan	guiness	tyan	s1834
tyan	s1846		
via	vt5292	via	vt5426

Chipsets:

acer	m1631	alpha	tsunami
amd	amd76x	intel	430tx
intel	440bx	intel	440gx
intel	82815ep	intel	82830
intel	82860	intel	E7500
micron	21PAD	nsc	gx1
via	vt694	via	vt8601
NSC	scx200	sis	540
sis	550	sis	630
sis	635	sis	730
sis	735	TI	pci1225
acer	m1535	acer	m1543
amd	amd766	intel	82801
intel	82801ca	intel	82806
intel	82870	intel	piix4e
nsc	cs5530	nsc	scx200
via	vt8231	via	vt82c686

Concepts summary

LinuxBIOS has taken on some features from normal PC BIOSes and has become more complex but also more mature.

Is still much smaller, more flexible and more maintainable than a normal BIOS.

Allows faster and more stable booting at a price that can't be beat.

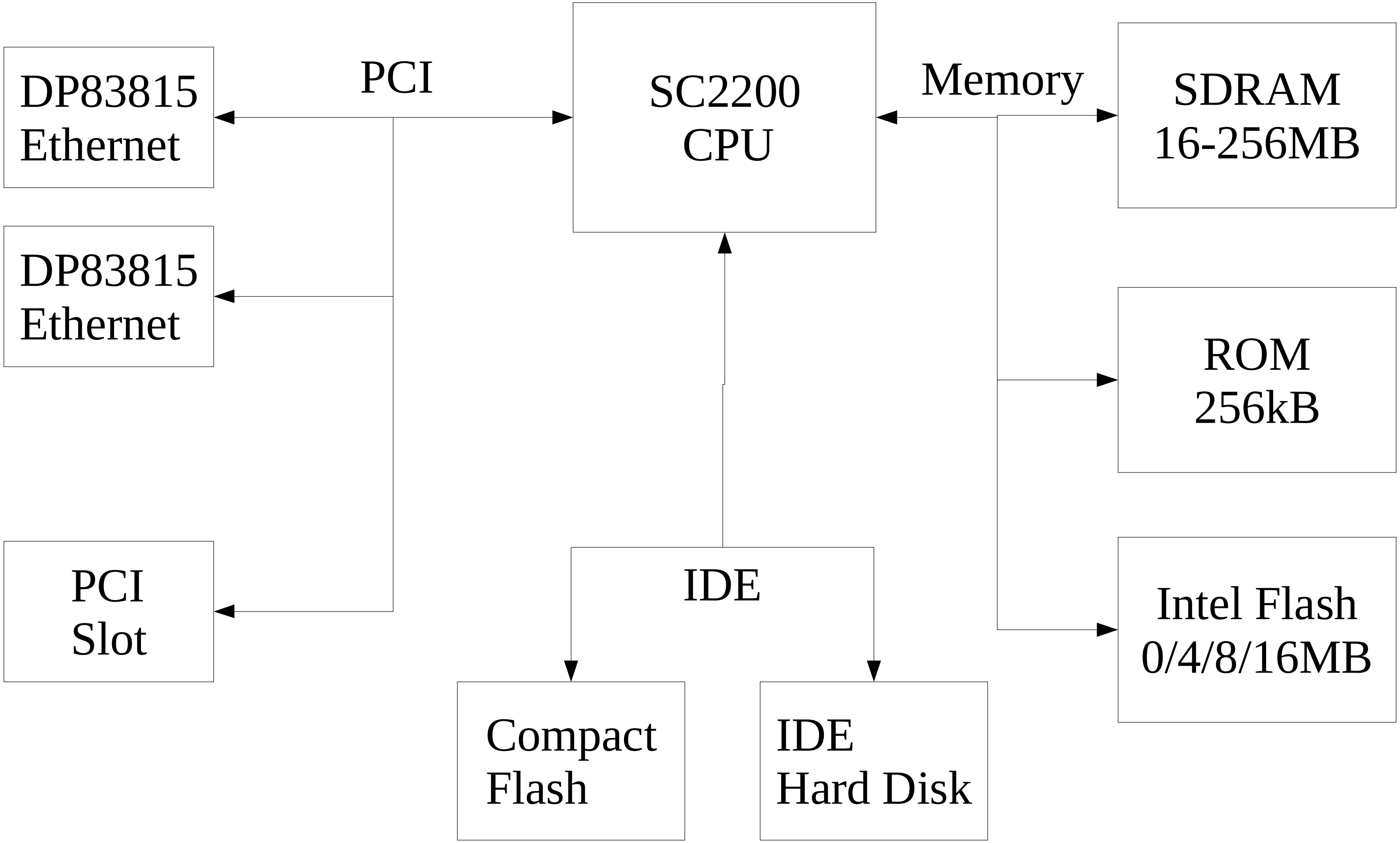
The Nano Computer

The Nano Computer is an embedded computer from Nano Computer Systems AB:

http://www.nano-system.com/

The system is based on a reference design for the National Semiconductor SC2200 processor. This single chip contains most of the parts needed to build a 266MHz Pentium class PC.

Nano Computer block diagram



DP83815 Ethernet

DP83815 Ethernet

PCI Slot

PCI

SC2200 CPU

Memory

SDRAM 16-256MB

ROM 256kB

Intel Flash 0/4/8/16MB

IDE

Compact Flash

IDE Hard Disk

Getting started

Find out if your components are supported:
    CPU, northbridge, southbridge, SuperI/O, etc.

If yes, write a new board description file, add some code to to mainboard fixups and a table describing the IRQ routing on the board.

If not, implement support for your CPU and chipset. This can be a large job, but often new chipsets are slight variations of older chipsets so it's usually not too hard.

Source structure

freebios/src/arch/*CPU*

freebios/src/northbridge/*VENDOR*/*CHIP*

freebios/src/southbridge/*VENDOR*/*CHIP*

freebios/src/northsouthbridge/*VENDOR*/*CHIP*

freebios/src/superio/*VENDOR*/*CHIP*

freebios/src/mainboard/*VENDOR*/*CHIP*

A twisty little maze of includes

Generic 586 init - 16 and 32 bit assembly

Early chipset/board initialization - 32 bit assembly

Chipset specific memory setup - 32 bit assembly

Generic setup code - 32 bit assembly

Generic setup code - C

Generic code to do a PCI scan - C

Northbridge, southbridge and mainboard fixups - C

Generic C code to load and start the OS - C

A mainboard Config file

From `src/mainboard/nano/nano/Config:`

```
arch i386
cpu p5
mainboardinit cpu/i386/entry16.inc
mainboardinit cpu/i386/reset16.inc
mainboardinit \
        southbridge/nsc/scx200/scx200_setup.inc
mainboardinit pc80/serial.inc
mainboardinit arch/i386/lib/console.inc

northbridge nsc/gx1
southbridge nsc/scx200

option SCx200_PMR=0x02860891
```

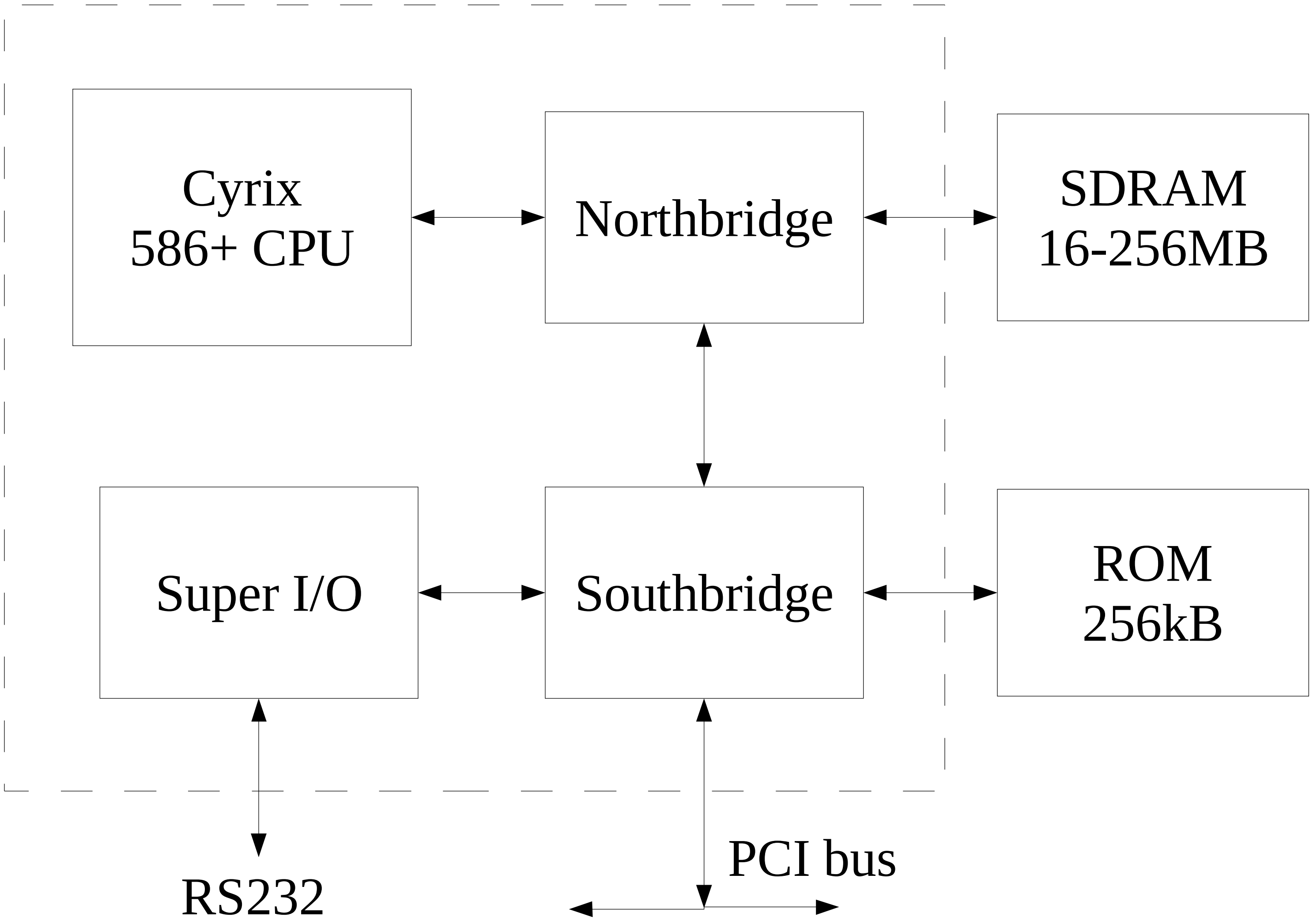
Trying it out

Compile Linuxbios for your mainboard.

Find a tool to program the ROM chip of your motherboard.

Connect a null modem cable to the serial port and finally try it out.

Inside the SC2200



Adding a new northbridge

Add assembly code to do minimal initialization of the CPU and chipset:

From `freebios/src/northbridge/nsc/gx1/Config:`

```
mainboardinit northbridge/nsc/gx1/cpu_setup.inc
mainboardinit northbridge/nsc/gx1/gx_setup.inc
mainboardinit \
     northbridge/nsc/gx1/northbridge_setup.inc
```

Rather hard since there is no way of getting good debug messages out of the system. An ICE is a very good idea, but even one GPIO pin connected to a LED can be of great help.

Getting debug messages

The next step was to enable a serial port to get better debug messages. This requires much on the SC2200 since the SuperI/O is a behind the south bridge which must be enabled first.

From `src/mainboard/nano/nano/Config:`

```
mainboardinit \
        southbridge/nsc/scx200/scx200_setup.inc
```

SDRAM initialization

Time to tackle a hard one, getting SDRAM initializing and autosizing to work.
The memory controller is a part of the northbridge.

From `freebios/src/northbridge/nsc/gx1/Config:`

```
raminit northbridge/nsc/gx1/raminit.inc
```

With the serial console working it was a bit easier.
It was possible to dump register and memory contents to the serial port.

The rest of the port

From `freebios/src/southbridge/nsc/scx200/Config`:

```
object southbridge.o
```

The file `southbridge.c` file contains some code to enable the IRQ mapping and enable the IDE and USB controllers.

Other than this only minor additions and helper functions. No changes needed to the generic code.

How hard is it?

It took me about 2 weeks to get LinuxBIOS running on the Nano Computer.

- 1 week to do CPU and northbridge initialization
- 1 week to implement memory sizing
- a few days for the rest of the north- and southbridge code

Modularization

Originally the SC2200 support was a combined north- and southbridge definition. People were asking for GX1 and CS5530 support. Same CPU core and northbridge, but different southbridge.

Took me about a week to do a port to such a board. Most time was spent on designing a structure so that the two ports could share code. The new structure shares the CPU initialization and SDRAM sizing code and reused parts of the southbridge code.

Didn't take long before there were three more ports for different GX1/CS5530 boards.

Porting summary

Easy to do a port to a new mainboard with a supported chipset and good documentation.

Harder to add support for a new chipset, especially if the chipset has bugs or incomplete documentation.

Often possible to reuse code already in the tree.

All this gets easier with good tools such as an ICE, ROM emulator and a logic analyzer.

Future directions

What is happening to LinuxBIOS for embedded systems?

Adding more features from a normal BIOS:

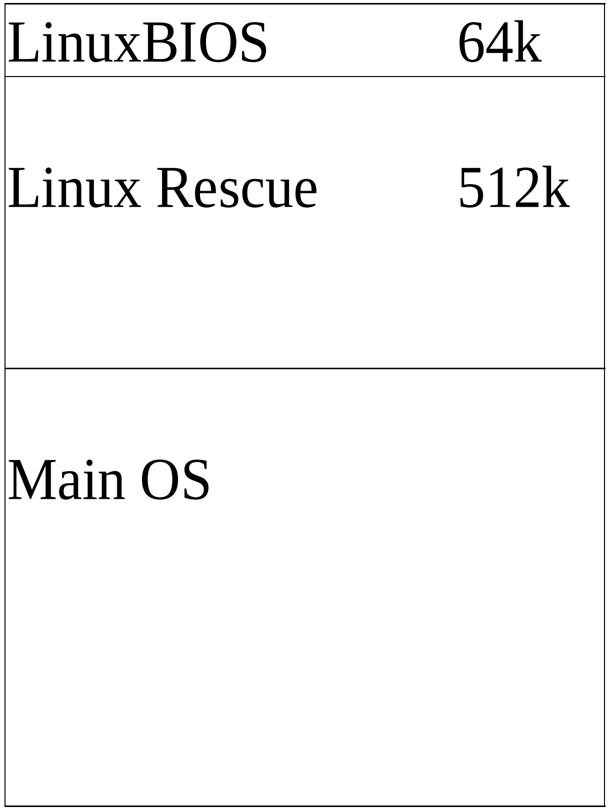
- BIOS INT emulation
- PCI expansion rom emulation (for VGA cards)
- Booting from IDE and network (using Etherboot or GRUB)
- Adding a serial monitor (OpenBOOT/Tiara)

Dual booting

Possible to do tricks such as dual booting from flash

If the main OS is ok, LinuxBIOS boots directly into it. If the checksum is bad or the booting fails it boots into the rescue system.

The rescue system can connect to a remote system, do a crash dump, install a new OS or whatever.



LinuxBIOS 64k

Linux Rescue 512k

Main OS

Future of the NatSemi port

Things I'd like to add to the NatSemi port:

Support for VSA/SMM mode, either do a custom SMM BIOS, or allow the NatSemi VSA BIOS to work with LinuxBIOS.

Support for video and audio

Support for Save to RAM

Other projects

OpenBoot - forth
Tiara - OpenBoot with boot monitor
PPCboot - PowerPC and ARM (etrax)
Lots of loaders for ARM
EtherBoot - BOOTP/TFTP client
oskit - Netboot

Questions

Read More

Christer Weinigel, Weinigel Ingenjörsbyrå AB
http://www.weinigel.se/

The LinuxBIOS mailing lists and web page

The Nano Computer
http://www.nano-system.com/

National Semiconductor - Internet appliances
http://www.national.com/