Running BSD-licensed Software on BSD-licensed Hardware

Marius Strobl
marius@FreeBSD.org

EuroBSDcon 2012
Warsaw University of Technology
Warsaw, Poland
October 20 – 21, 2012

Embedded systems development typical requirements

Microcontroller (µC) based reference design providing:

- Analog/Digital Converters (ADCs)
- General Purpose Input/Output (GPIO) interface
- IEEE 802.3 [1] compliant Ethernet Media Access Controller (MAC)
- Real-Time Clock (RTC)
- Universal (Synchronous/)Asynchronous Receiver Transmitter (U(S)ART) for EIA RS-232-C [2] or RS-485 [3]
- Internal or external volatile (RAM) and non-volatile (NVRAM) random-access memory, flash read-only memory (ROM)
- IEEE 1149.1 [4] compliant Joint Test Action Group (JTAG) interface or Serial Peripheral Interface (SPI) for In-System Programming (ISP) and optionally debugging
- Source code of real-time kernel plus drivers for above devices
- Communication protocol stacks, mainly for Transmission Control Protocol [5]/Internet Protocol [6] (TCP/IP)

Ethernut board family of reference design boards

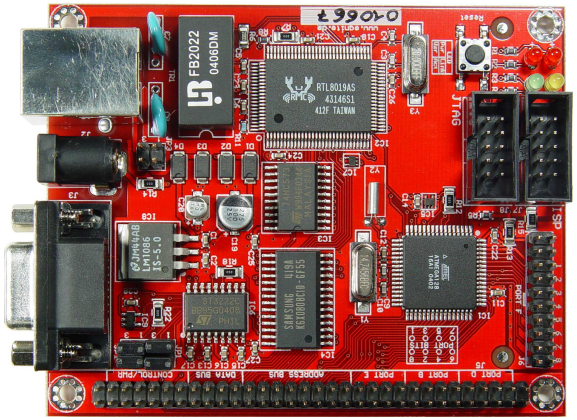

Figure : Ethernut 1.3G [7]

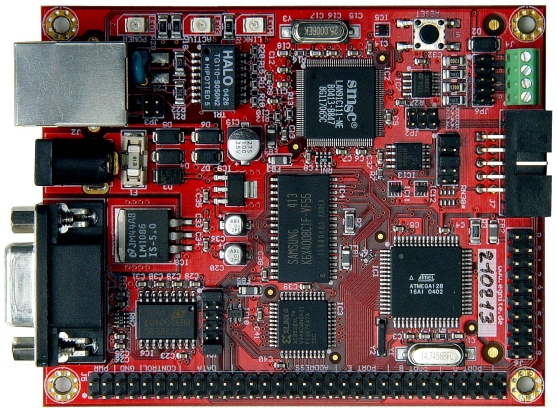

Figure : Ethernut 2.1B [8]

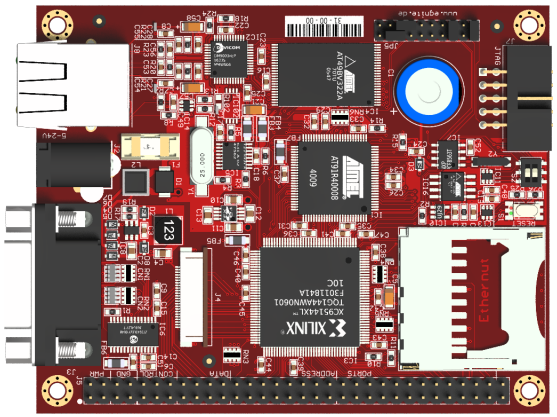

Figure : Ethernut 3.1D [9]

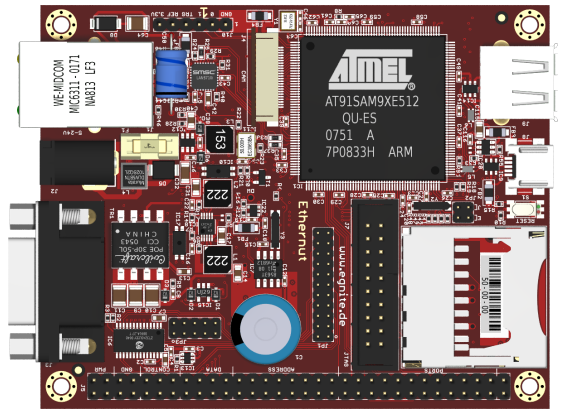

Figure : Ethernut 5.0F [10]

Ethernut board family features

Model	Microcontroller	RAM [Bytes]	Flash [Bytes]	MAC [Mbps]
Ethernut 1	AVR® 8-bit ATmega128	4k int. 32k ext.	128k	10
Ethernut 2	AVR® 8-bit ATmega128	4k int. 512k ext.	128k	10/100
Ethernut 3	ARM7-TDMI 32-bit AT91R40008	256k	4M	10/100
Ethernut 5	ARM9 32-bit AT91SAM9XE512	32k int. 128M ext.	512k int. 4M ext. 1G ext.	10/100

Table : Microcontrollers and memory of the Ethernut board models

Ethernut board family features c'tinued

Model	ADC	GPIO [lines]	NVRAM [Bytes]	I^2C, RTC, MMC/SD slot	UART/ USART
Ethernut 1	8 chan. 10-bit	22	4k		0/2
Ethernut 2	8 chan. 10-bit	28	4k		0/2
Ethernut 3		17+	32k (3.0) 4M (3.1)	available	0/2
Ethernut 5	4 chan. 10-bit	15	(4M)	available	1/4

Table : Specific features of the Ethernut board models

Ethernut power consumption at 12 V

Ethernut 1, running Ethernet-bridging application, link up: $\approx$ 90 mA

Ethernut 5, booted to loader prompt, no MMC/SD card: $\approx$ 160 mA

Ethernut 5, FreeBSD multi-user from SD card, link up: $\approx$ 180 mA

Ethernut boards layouts & schematics BSD-style license [11]

```
Copyright (C) 2001-2004 by egnite Software GmbH.
All rights reserved.
Redistribution and use with or without modification are
permitted provided that the following conditions are met:
1. Redistributions must reproduce the above copyright
   notice and this list of conditions.
2. Neither the name of the copyright holders nor the
   names of contributors may be used to endorse or
   promote products derived from this hardware design
   without specific prior written permission.
3. This hardware design is provided by egnite Software
   GmbH and contributors as is without any warranties.
Ethernut is a registered trademark of egnite Software GmbH.
```

Ethernut boards layouts & schematics EAGLE [12] files

Ethernut 1.3 Revision G:
`http://www.ethernut.de/arc/enut130g.zip`

Ethernut 2.1 Revision B:
`http://www.ethernut.de/arc/enut21b.zip`

Ethernut 2 CPLD (external RAM and MAC interfacing):
`http://www.ethernut.de/arc/enutcpld.zip`

Ethernut 3.1 Revision D:
`http://www.ethernut.de/arc/enut31d.zip`

Ethernut 5.0 Revision F:
`http://www.ethernut.de/arc/enut50f.zip`

Medianut 2.0 Revision D (MP3 decoder):
`http://www.ethernut.de/en/hardware/medianut2/medianut20d.zip`

Ethernut extensions template:
`http://www.ethernut.de/arc/ntmp10a.zip`

Nut/OS features

Simple, modular Real-Time Operation System (RTOS) kernel, main features [13]:

- Cooperative multithreading
- Deterministic interrupt response times
- Priority based event handling
- Periodic and one-shot timers
- Dynamic memory management
- Base protocols Ethernet, ARP, IP, ICMP, UDP, TCP and PPP
- User protocols DHCP, DNS, SNTP, SMTP, FTP, SYSLOG, HTTP and others
- Socket API
- Host, net and default routing
- BSD-style licensed

"Userland" is largely IEEE 1003.1 [14] POSIX® compliant.

Nut/OS features c'tinued

Ready-to-use drivers for a large number of devices, including:

- ADCs, Ethernet MACs, RTCs, USARTs/UARTs
- SPI, I2C and CAN busses
- MultiMedia (MMC) and Secure Digital (SD) cards
- Hardware and software audio codecs
- Serial flash memory
- Infrared remote controls
- Watchdogs and reset controllers
- Character displays
- GPIO and interrupt control

Actively supported microcontrollers:

- Atmel® AVR® ATmega103, ATmega128, ATmega2561, AT90CAN128
- Atmel® ARM7 AT91SAM7S, AT91SAM7SE, AT91SAM7X
- Atmel® ARM9 AT91SAM9260, AT91SAM9XE

Nut/OS resources

Sources (tarballs, SVN, etc., stable version as of Oct 2012: 4.10.3):
`http://sourceforge.net/projects/ethernut/`

NutWiki (examples, FAQ, etc.):
`http://www.ethernut.de/nutwiki/Main_Page`

API Reference:
`http://www.ethernut.de/api/index.html`

Nut/OS build requirements for Ethernut 1/2

Name	Description
avr-binutils-2.20.1_1	GNU binutils for Atmel AVR 8-bit RISC cross-development
avr-gcc-4.5.1_1	FSF GCC 4.x for Atmel AVR 8-bit RISC cross-development
avr-libc-1.8.0,1	A C and math library for the Atmel AVR controller family
avrdude-5.11	Program for programming the on-chip memory of Atmel AVR CPU
gmake-3.82_1	GNU version of 'make' utility
lua-5.1.5_4	Small, compilable scripting language providing easy access
pkgconf-0.8.9	Utility to help to configure compiler and linker flags

Table : Software required for compiling Nut/OS for Ethernut 1/2 boards and for flashing these

Installing Nut/OS

```
> cd $HOME
> tar -xzf /path/to/ethernut-4.10.3.tar.gz
> cd /ethernut-4.10.3
> ./configure --prefix=$HOME/ethernut-4.10.3 \
    --disable-nutconf-gui --disable-qnutconf \
    --disable-nutdisc --disable-qnutdisc
> gmake clean all install
```

Afterwards, add `$HOME/ethernut-4.10.3/bin` (according to `--prefix=FOO` above) to `$PATH`.

Configuring Nut/OS for Ethernut 1 (in-tree method)

In `$HOME/ethernut-4.10.3/nut`:

```
> ./nutsetup
```

Configuring Nut/OS for Ethernut 1 (in-tree method)

In `$HOME/ethernut-4.10.3/nut`:

```
> ./nutsetup
```

For Ethernut 1, flashing with `avrdude` using the `usbasp` protocol and compiling with `avr-gcc` select:

1) `Atmel ATmega128`
1) `Ethernut 1 (10 Mbit Realtek RTL8019AS)`
5) `14.7456 MHz`
3) `usbasp`
2) `avrdude`

Configuring Nut/OS for Ethernut 1 (in-tree method)

In `$HOME/ethernut-4.10.3/nut`:

```
> ./nutsetup
```

For Ethernut 1, flashing with `avrdude` using the `usbasp` protocol and compiling with `avr-gcc` select:

1) `Atmel ATmega128`
1) `Ethernut 1 (10 Mbit Realtek RTL8019AS)`
5) `14.7456 MHz`
3) `usbasp`
2) `avrdude`

Results in `Makedefs`, `Makerules`, `UserConf.mk`, `app/Makedefs` and `app/Makerules` being created (check into local repository if used).

Compiling Nut/OS for Ethernut 1 (in-tree method)

Hack for `UserConf.mk` allowing Nut/OS 4.10.3 to build with `avr-gcc` 4.5.1 (officially supported: 4.3.2) and unb0rking build system:

```
HWDEF=[...] -D__AVR_LIBC_DEPRECATED_ENABLE__ -D__PROG_TYPES_COMPAT__
UCPFLAGS=-Wno-deprecated-declarations
top_blddir = $(top_srcdir)
```

Compiling Nut/OS for Ethernut 1 (in-tree method)

Hack for `UserConf.mk` allowing Nut/OS 4.10.3 to build with `avr-gcc` 4.5.1 (officially supported: 4.3.2) and unb0rking build system:

```
HWDEF=[...] -D__AVR_LIBC_DEPRECATED_ENABLE__ -D__PROG_TYPES_COMPAT__
UCPFLAGS=-Wno-deprecated-declarations
top_blddir = $(top_srcdir)
```

In `$HOME/ethernut-4.10.3/nut/lib`:

```
> gmake clean all install
```

Compiling Nut/OS for Ethernut 1 (in-tree method)

Hack for `UserConf.mk` allowing Nut/OS 4.10.3 to build with `avr-gcc` 4.5.1 (officially supported: 4.3.2) and unb0rking build system:

```
HWDEF=[...] -D__AVR_LIBC_DEPRECATED_ENABLE__ -D__PROG_TYPES_COMPAT__
UCPFLAGS=-Wno-deprecated-declarations
top_blddir = $(top_srcdir)
```

In `$HOME/ethernut-4.10.3/nut/lib`:

```
> gmake clean all install
```

Rerun this step if you change Nut/OS source and manually relink your applications afterwards!

Building & flashing Nut/OS applications (in-tree method)

In `$HOME/ethernut-4.10.3/nut/app`:

```
> cd <application>
> gmake clean all
```

Building & flashing Nut/OS applications (in-tree method)

In `$HOME/ethernut-4.10.3/nut/app`:

```
> cd <application>
> gmake clean all
```

```
> gmake burn
```

Requires necessary permissions on the device node used for the ISP adapter (via `/etc/defaults/devfs.rules` if you can make it work, or just set the `setuid` bit on `avrdude` ...).

Hello World! [15]

```
#include <dev/board.h>
#include <io.h>
#include <stdio.h>

int
main(void)
{
        unsigned long baud = 115200;

        NutRegisterDevice(&DEV_DEBUG, 0, 0);

        freopen(DEV_DEBUG_NAME, "w", stdout);
        _ioctl(_fileno(stdout), UART_SETSPEED, &baud);

        printf("Hello World!");

        for (;;)
                ;

        return (0);
}
```

Hello World! Makefile

```
PROJ = hello_world

include ../Makedefs

SRCS= $(PROJ).c
OBJS= $(SRCS:.c=.o)
LIBS= $(LIBDIR)/nutinit.o -lnutpro -lnutos -lnutarch -lnutdev -lnutarch
LIBS+= -lnutfs -lnutnet -lnutcrt
TARG= $(PROJ).hex
PARM= $(PROJ).eep

all: $(OBJS) $(TARG) $(ITARG) $(DTARG)

include ../Makerules

clean:
        -rm -f $(OBJS)
        -rm -f $(TARG) $(ITARG) $(DTARG)
        -rm -f $(PROJ).eep
        -rm -f $(PROJ).obj
        -rm -f $(PROJ).map
        -rm -f $(SRCS:.c=.lst)
        -rm -f $(SRCS:.c=.bak)
        -rm -f $(SRCS:.c=.i)
```

Nut/OS build requirements for Ethernut 3/5

Name	Description
arm-elf-binutils-2.21 *	GNU binutils port for cross-target development
arm-elf-gcc-4.5.2 *	GNU gcc for cross-target development
gmake-3.82_1	GNU version of 'make' utility
lua-5.1.5_4	Small, compilable scripting language providing easy access
pkgconf-0.8.9	Utility to help to configure compiler and linker flags

*Build cross-{binutils,gcc} using TGTARCH=arm TGTABI=elf

Table : Software required for compiling Nut/OS for Ethernut 3/5 boards

Installing Nut/OS

```
> cd $HOME
> tar -xzf /path/to/ethernut-4.10.3.tar.gz
> cd /ethernut-4.10.3
> ./configure --prefix=$HOME/ethernut-4.10.3 \
    --disable-nutconf-gui --disable-qnutconf \
    --disable-nutdisc --disable-qnutdisc
> gmake clean all install
```

Afterwards, add `$HOME/ethernut-4.10.3/bin` (according to `--prefix=FOO` above) to `$PATH`.

Configuring & compiling Nut/OS for Ethernut 5 (build-tree method)

In `$HOME/ethernut-4.10.3/`:

```
> nutconfigure -bnutbld-50f -cnut/conf/ethernut50f.conf \
    -lnutbld-50f/lib -marm-gcc create-buildtree
```

Results in the `$HOME/ethernut-4.10.3/nutbld-50f` being created (check into local repository if used).

Configuring & compiling Nut/OS for Ethernut 5 (build-tree method)

In `$HOME/ethernut-4.10.3/`:

```
> nutconfigure -bnutbld-50f -cnut/conf/ethernut50f.conf \
    -lnutbld-50f/lib -marm-gcc create-buildtree
```

Results in the `$HOME/ethernut-4.10.3/nutbld-50f` being created (check into local repository if used).

Hack to build with `arm-elf-*` FreeBSD ports as of Oct 2012: in `nutbld-50f/NutConf.mk`, replace `TRGT = arm-none-eabi-` with `TRGT = arm-elf-`.

Configuring & compiling Nut/OS for Ethernut 5 (build-tree method)

In `$HOME/ethernut-4.10.3/`:

```
> nutconfigure -bnutbld-50f -cnut/conf/ethernut50f.conf \
    -lnutbld-50f/lib -marm-gcc create-buildtree
```

Results in the `$HOME/ethernut-4.10.3/nutbld-50f` being created (check into local repository if used).

Hack to build with `arm-elf-*` FreeBSD ports as of Oct 2012: in `nutbld-50f/NutConf.mk`, replace `TRGT = arm-none-eabi-` with `TRGT = arm-elf-`.

```
> cd nutbld-50f
> gmake clean all install
```

Setting up Nut/OS applications tree for Ethernut 5 (build-tree method)

In `$HOME/ethernut-4.10.3/`:

```
> nutconfigure -anutapp-50f -bnutbld-50f \
    -cnut/conf/ethernut50f.conf -lnutbld-50f/lib \
    -marm-gcc create-apptree
```

Setting up Nut/OS applications tree for Ethernut 5 (build-tree method)

In `$HOME/ethernut-4.10.3/`:

```
> nutconfigure -anutapp-50f -bnutbld-50f \
    -cnut/conf/ethernut50f.conf -lnutbld-50f/lib \
    -marm-gcc create-apptree
```

Hack to build with `arm-elf-*` FreeBSD ports as of Oct 2012:
in `nutapp-50f/NutConf.mk`, replace `TRGT = arm-none-eabi-` with `TRGT = arm-elf-`.

Setting up Nut/OS applications tree for Ethernut 5 (build-tree method)

In `$HOME/ethernut-4.10.3/`:

```
> nutconfigure -anutapp-50f -bnutbld-50f \
    -cnut/conf/ethernut50f.conf -lnutbld-50f/lib \
    -marm-gcc create-apptree
```

Hack to build with `arm-elf-*` FreeBSD ports as of Oct 2012:
in `nutapp-50f/NutConf.mk`, replace `TRGT = arm-none-eabi-` with `TRGT = arm-elf-`.

Build all example applications:

```
> cd nutapp-50f
> gmake clean all install
```

Results in `$HOME/ethernut-4.10.3/nutapp-50f/app/*/*.bin`.

Flashing/Netbooting Ethernut 5

Setting Ethernut 5 IP address, TFTP server (`next-server`) and nfsroot (`root-path`, for FreeBSD) via ISC DHCPD in `dhcpd.conf`:

```
host enut5 {
  hardware ethernet 00:06:98:50:00:83;
  option host-name "enut5.lan.zeist.de";
  fixed-address 192.168.1.5;
  always-reply-rfc1048 on;
  next-server 192.168.1.2;
  option root-path "192.168.1.2:/usr/data/nfsroot/arm";
}
```

Flashing/Netbooting Ethernut 5

Setting Ethernut 5 IP address, TFTP server (`next-server`) and nfsroot (`root-path`, for FreeBSD) via ISC DHCPD in `dhcpd.conf`:

```
host enut5 {
  hardware ethernet 00:06:98:50:00:83;
  option host-name "enut5.lan.zeist.de";
  fixed-address 192.168.1.5;
  always-reply-rfc1048 on;
  next-server 192.168.1.2;
  option root-path "192.168.1.2:/usr/data/nfsroot/arm";
}
```

Setting up `tftpd(8)` via `inetd(8)` in `/etc/inetd.conf` (assuming `/usr/data/nfsroot/arm/boot` as boot directory):

```
tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s
    /usr/data/nfsroot/arm/boot
```

Afterwards, set `inetd_enable="YES"` in `/etc/rc.conf` and run `/etc/rc.d/inetd start`.

Flashing/Netbooting Ethernut 5 c'tinued

Connect to serial port of Ethernut 5 using 115200 bps and 8N1:

```
SAMBOOT 1.4 :  Type=3 Loading 0x42000 to 0x27000000  done
[...]
U-Boot 2011.12 (Mar 07 2012 - 13:43:55)

CPU: AT91SAM9XE
Crystal frequency:   18.432 MHz
CPU clock        :  180.634 MHz
Master clock     :   90.317 MHz
[...]
Hit any key to stop autoboot:  0
U-Boot>
```

Flashing/Netbooting Ethernut 5 c'tinued

Connect to serial port of Ethernut 5 using 115200 bps and 8N1:

```
SAMBOOT 1.4 :  Type=3 Loading 0x42000 to 0x27000000  done
[...]
U-Boot 2011.12 (Mar 07 2012 - 13:43:55)

CPU: AT91SAM9XE
Crystal frequency:   18.432 MHz
CPU clock        :  180.634 MHz
Master clock     :   90.317 MHz
[...]
Hit any key to stop autoboot:  0
U-Boot>
```

Alternatively, set Ethernut 5 IP address and TFTP server statically:

```
U-Boot> setenv ipaddr 192.168.1.5
U-Boot> setenv tftpserverip 192.168.1.2
U-Boot> saveenv
```

Flashing/Netbooting Nut/OS binaries on Ethernut 5

Put application, f.e. `hello_world.bin`, into TFTP boot directory.

Flashing/Netbooting Nut/OS binaries on Ethernut 5

Put application, f.e. `hello_world.bin`, into TFTP boot directory.

Run Nut/OS application once via TFTP:

```
U-Boot> setenv nut_image_name hello_world.bin
U-Boot> run tftpbootnut
```

Flashing/Netbooting Nut/OS binaries on Ethernut 5

Put application, f.e. `hello_world.bin`, into TFTP boot directory.

Run Nut/OS application once via TFTP:

```
U-Boot> setenv nut_image_name hello_world.bin
U-Boot> run tftpbootnut
```

Run Nut/OS application automatically via TFTP on boot:

```
U-Boot> setenv nut_image_name hello_world.bin
U-Boot> setenv bootcmd run tftpbootnut
U-Boot> saveenv
```

Flashing/Netbooting Nut/OS binaries on Ethernut 5

Put application, f.e. `hello_world.bin`, into TFTP boot directory.

Run Nut/OS application once via TFTP:

```
U-Boot> setenv nut_image_name hello_world.bin
U-Boot> run tftpbootnut
```

Run Nut/OS application automatically via TFTP on boot:

```
U-Boot> setenv nut_image_name hello_world.bin
U-Boot> setenv bootcmd run tftpbootnut
U-Boot> saveenv
```

Install Nut/OS application (528 kbytes max.) into flash via TFTP and run automatically on boot:

```
U-Boot> setenv nut_image_name hello_world.bin
U-Boot> run tftpinstallnut
U-Boot> setenv bootcmd run flashbootnut
U-Boot> saveenv
```

FreeBSD on Ethernut 5

Supported in head as of r235348 and in stable/9 as of r237386 (thus beginning with 9.1-RELEASE).

FreeBSD on Ethernut 5

Supported in head as of r235348 and in stable/9 as of r237386 (thus beginning with 9.1-RELEASE).

Necessary changes mainly consisted in:

- adding support for Atmel® SAM9XE family (ARM926EJ-S core),

FreeBSD on Ethernut 5

Supported in head as of r235348 and in stable/9 as of r237386 (thus beginning with 9.1-RELEASE).

Necessary changes mainly consisted in:

- adding support for Atmel® SAM9XE family (ARM926EJ-S core),
- writing pcf8563_rtc(4) driver for NXP (Philips) PCF8563 RTC,

FreeBSD on Ethernut 5

Supported in head as of r235348 and in stable/9 as of r237386 (thus beginning with 9.1-RELEASE).

Necessary changes mainly consisted in:

- adding support for Atmel® SAM9XE family (ARM926EJ-S core),
- writing pcf8563_rtc(4) driver for NXP (Philips) PCF8563 RTC,
- fixing at45d(4) (DataFlash), at91_mci(4) (MMC/SD host bridge), at91_spi(4), at91_twi(4) (I^2C controller), at91_usart(4) and mmc(4) (thanks to Ian Lapore) drivers as well as adding workarounds for silicon bugs of specific microcontrollers.

FreeBSD on Ethernut 5

Supported in head as of r235348 and in stable/9 as of r237386 (thus beginning with 9.1-RELEASE).

Necessary changes mainly consisted in:

- adding support for Atmel® SAM9XE family (ARM926EJ-S core),
- writing pcf8563_rtc(4) driver for NXP (Philips) PCF8563 RTC,
- fixing at45d(4) (DataFlash), at91_mci(4) (MMC/SD host bridge), at91_spi(4), at91_twi(4) (I^2C controller), at91_usart(4) and mmc(4) (thanks to Ian Lapore) drivers as well as adding workarounds for silicon bugs of specific microcontrollers.

⇒ Should make FreeBSD the first operating system besides Nut/OS to support Ethernut 5 out of tree.

Cross-compiling FreeBSD userland for Ethernut 5

```
> mkdir 9.1
> cd 9.1
> svn co -q svn://svn.freebsd.org/base/releng/9.1 src
> mkdir /path/to/tmp/9.1
> setenv MAKEOBJDIRPREFIX /path/to/tmp/9.1
> cd src
> make -j4 SRCCONF=/dev/null __MAKE_CONF=/dev/null \
    -DMALLOC_PRODUCTION TARGET=arm buildworld
```

FreeBSD kernel configuration for Ethernut 5

`sys/arm/conf/ETHERNUT5` defaults to netbooting:

```
options         NFS_ROOT
[...]
options         BOOTP
options         BOOTP_COMPAT
options         BOOTP_NFSROOT
options         BOOTP_NFSV3
options         BOOTP_WIRED_TO=ate0
```

FreeBSD kernel configuration for Ethernut 5

`sys/arm/conf/ETHERNUT5` defaults to netbooting:

```
options         NFS_ROOT
[...]
options         BOOTP
options         BOOTP_COMPAT
options         BOOTP_NFSROOT
options         BOOTP_NFSV3
options         BOOTP_WIRED_TO=ate0
```

For mounting root from an MMC/SD card, replace above with:

```
options         ROOTDEVNAME=\"ufs:/dev/mmcsd0a\"
```

FreeBSD kernel configuration for Ethernut 5

`sys/arm/conf/ETHERNUT5` defaults to netbooting:

```
options         NFS_ROOT
[...]
options         BOOTP
options         BOOTP_COMPAT
options         BOOTP_NFSROOT
options         BOOTP_NFSV3
options         BOOTP_WIRED_TO=ate0
```

For mounting root from an MMC/SD card, replace above with:

```
options         ROOTDEVNAME=\"ufs:/dev/mmcsd0a\"
```

Cross-compiling the kernel in `src`:

```
> make -j4 SRCCONF=/dev/null __MAKE_CONF=/dev/null \
    TARGET=arm KERNCONF=ETHERNUT5 buildkernel
```

Installing FreeBSD (nfs)root for Ethernut 5

In `src` (assuming `/usr/data/nfsroot/arm` as nfsroot directory):

```
> make SRCCONF=/dev/null __MAKE_CONF=/dev/null \
    TARGET=arm KERNCONF=ETHERNUT5 KERNEL_KO=kernel.bin \
    DESTDIR=/usr/data/nfsroot/arm \
    installkernel installworld distribution
```

Installing FreeBSD (nfs)root for Ethernut 5

In `src` (assuming `/usr/data/nfsroot/arm` as nfsroot directory):

```
> make SRCCONF=/dev/null __MAKE_CONF=/dev/null \
    TARGET=arm KERNCONF=ETHERNUT5 KERNEL_KO=kernel.bin \
    DESTDIR=/usr/data/nfsroot/arm \
    installkernel installworld distribution
```

Create dummy `/etc/fstab` and probably disable `sendmail(8)` but enable `sshd(1)`:

```
> touch /usr/data/nfsroot/arm/etc/fstab
> printf 'sendmail_enable=\"NONE\"\nsshd_enable="YES"\n' >
    /usr/data/nfsroot/arm/etc/rc.conf
```

Installing FreeBSD (nfs)root for Ethernut 5

In `src` (assuming `/usr/data/nfsroot/arm` as nfsroot directory):

```
> make SRCCONF=/dev/null __MAKE_CONF=/dev/null \
    TARGET=arm KERNCONF=ETHERNUT5 KERNEL_KO=kernel.bin \
    DESTDIR=/usr/data/nfsroot/arm \
    installkernel installworld distribution
```

Create dummy `/etc/fstab` and probably disable `sendmail(8)` but enable `sshd(1)`:

```
> touch /usr/data/nfsroot/arm/etc/fstab
> printf 'sendmail_enable=\"NONE\"\nsshd_enable="YES"\n' >
    /usr/data/nfsroot/arm/etc/rc.conf
```

A FreeBSD 9.1-RC2 nfsroot tarball is available at:
`http://people.freebsd.org/~marius/ethernut/FreeBSD-9.1-RC2-arm-ETHERNUT5-nfsroot.tar.bz2`

Installing FreeBSD (nfs)root for Ethernut 5

In `src` (assuming `/usr/data/nfsroot/arm` as nfsroot directory):

```
> make SRCCONF=/dev/null __MAKE_CONF=/dev/null \
    TARGET=arm KERNCONF=ETHERNUT5 KERNEL_KO=kernel.bin \
    DESTDIR=/usr/data/nfsroot/arm \
    installkernel installworld distribution
```

Create dummy `/etc/fstab` and probably disable `sendmail(8)` but enable `sshd(1)`:

```
> touch /usr/data/nfsroot/arm/etc/fstab
> printf 'sendmail_enable=\"NONE\"\nsshd_enable="YES"\n' >
    /usr/data/nfsroot/arm/etc/rc.conf
```

A FreeBSD 9.1-RC2 nfsroot tarball is available at:
`http://people.freebsd.org/~marius/ethernut/FreeBSD-9.1-RC2-arm-ETHERNUT5-nfsroot.tar.bz2`

$\Rightarrow$ Also usable for root on an MMC/SD card.

Preparing for netbooting FreeBSD on Ethernut 5 (once)

Make sure the nfsroot is allowed in `/etc/exports` on the host à la:

```
/ -alldirs -maproot=0 -network 192.168.1 -mask 255.255.255.0
```

and that the NFS server is running (set `mountd_enable`, `nfs_server_enable`, `rpc_lockd_enable`, `rpc_statd_enable` to `YES` in `/etc/rc.conf`, then run `/etc/rc.d/nfsd start`).

Preparing for netbooting FreeBSD on Ethernut 5 (once)

Make sure the nfsroot is allowed in `/etc/exports` on the host à la:

```
/ -alldirs -maproot=0 -network 192.168.1 -mask 255.255.255.0
```

and that the NFS server is running (set `mountd_enable`, `nfs_server_enable`, `rpc_lockd_enable`, `rpc_statd_enable` to `YES` in `/etc/rc.conf`, then run `/etc/rc.d/nfsd start`).

On the Ethernut 5:

```
U-Boot> setenv fbsd_image_name kernel/kernel.bin
U-Boot> setenv powerup4fbsd 'pwrman on; sleep 2'
U-Boot> setenv select_fbsd
    'image_name=${fbsd_image_name} ;
    start=${linux_flash_start} ;
    end=${linux_flash_end} ;
    setenv filesize ${linux_maxsize}'
U-Boot> setenv tftpbootfbsd 'run powerup4fbsd
    select_fbsd load_tftp_image && go 0x20000000'
U-Boot> saveenv
```

Netbooting FreeBSD on Ethernut 5

```
U-Boot> run tftpbootfbsd
```

Netbooting FreeBSD on Ethernut 5

```
U-Boot> run tftpbootfbsd
```

Optionally for automated netbooting on boot:

```
U-Boot> setenv bootcmd run tftpbootfbsd
U-Boot> saveenv
```

Preparing an MMC/SD card for FreeBSD root & Ethernut 5

A new MMC/SD card typically has FAT stuff on it:

```
> gpart show
=>       63  15759297  mmcsd0  MBR  (7.5G)
         63       8129          - free -  (4M)
       8192  15751168       1  fat32  (7.5G)
> gpart delete -i 1 mmcsd0
> gpart destroy mmcsd0
```

Preparing an MMC/SD card for FreeBSD root & Ethernut 5

A new MMC/SD card typically has FAT stuff on it:

```
> gpart show
=>       63  15759297  mmcsd0  MBR  (7.5G)
         63      8129          - free -  (4M)
       8192  15751168       1  fat32  (7.5G)
> gpart delete -i 1 mmcsd0
> gpart destroy mmcsd0
```

Creating BSD/UFS partitioning and file system:

```
> gpart create -s BSD -n 20 mmcsd0
> gpart add -t freebsd-ufs mmcsd0
> gpart show
=>        0  15759360  da0  BSD  (7.5G)
          0  15759360    1  freebsd-ufs  (7.5G)
> newfs -U -O2 /dev/mmcsd0a
```

$\Rightarrow$ Mount `/dev/mmcsd0a` and install root à la netbooting preparation (either use appropriate `DESTDIR` or copy over, etc.).

Preparing for flashing and running a FreeBSD kernel on Ethernut 5 (once)

```
U-Boot> setenv fbsd_image_name kernel/kernel.bin
U-Boot> setenv powerup4fbsd 'pwrman on; sleep 2'
U-Boot> setenv select_fbsd
    'image_name=${fbsd_image_name} ;
    start=${linux_flash_start} ;
    end=${linux_flash_end} ;
    setenv filesize ${linux_maxsize}'
U-Boot> setenv tftpinstallfbsd
    'run select_fbsd load_tftp_image
    save_protected_flash_image'
U-Boot> setenv flashbootfbsd
    'run powerup4fbsd select_fbsd load_flash_image &&
    go 0x20000000'
U-Boot> saveenv
```

Flashing a FreeBSD kernel onto Ethernut 5

Installing a FreeBSD kernel (2640 kbytes max.) into flash via TFTP:

```
U-Boot> run tftpinstallfbsd
```

Flashing a FreeBSD kernel onto Ethernut 5

Installing a FreeBSD kernel (2640 kbytes max.) into flash via TFTP:

```
U-Boot> run tftpinstallfbsd
```

Or, if FreeBSD is already booted on an Ethernut 5:

```
# dd if=/path/to/kernel.bin of=/dev/map/kernel bs=528
```

Note that both methods take some time!

Flashing a FreeBSD kernel onto Ethernut 5

Installing a FreeBSD kernel (2640 kbytes max.) into flash via TFTP:

```
U-Boot> run tftpinstallfbsd
```

Or, if FreeBSD is already booted on an Ethernut 5:

```
# dd if=/path/to/kernel.bin of=/dev/map/kernel bs=528
```

Note that both methods take some time!

Make sure to use a kernel with `ROOTDEVNAME` set for this, i.e. `${MAKEOBJDIRPREFIX}/arm.arm/‘pwd‘/sys/ETHERNUT5/kernel.bin` after compiling such a kernel or use the one supplied at: `http://people.freebsd.org/~marius/ethernut/FreeBSD-9.1-RC2-arm-ETHERNUT5-kernel.bin-mmc.bz2`

Booting FreeBSD from flash and MMC/SD card on Ethernut 5

```
U-Boot> run flashbootfbsd
```

Booting FreeBSD from flash and MMC/SD card on Ethernut 5

```
U-Boot> run flashbootfbsd
```

Optionally for automated booting:

```
U-Boot> setenv bootcmd run flashbootfbsd
U-Boot> saveenv
```

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),
- get `ubldr` (U-Boot loader) working,

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),
- get `ubldr` (U-Boot loader) working,

For FreeBSD/arm:

- improve ate(4) (speed, reliability),

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),
- get `ubldr` (U-Boot loader) working,

For FreeBSD/arm:

- improve ate(4) (speed, reliability),
- improve at91_spi(4)/spi(4) (speed),

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),
- get `ubldr` (U-Boot loader) working,

For FreeBSD/arm:

- improve ate(4) (speed, reliability),
- improve at91_spi(4)/spi(4) (speed),
- improve MD bus_dma(9) implementation (cache flushing),

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),
- get `ubldr` (U-Boot loader) working,

For FreeBSD/arm:

- improve ate(4) (speed, reliability),
- improve at91_spi(4)/spi(4) (speed),
- improve MD bus_dma(9) implementation (cache flushing),
- unb0rk usb(4) (bus_dma(9) usage),

TODOs

For FreeBSD on Ethernut 5:

- write driver for the power management controller (allows to control the power of the MAC clock, MMC/SD card, RS-232 interface, USB connectors),
- get `ubldr` (U-Boot loader) working,

For FreeBSD/arm:

- improve ate(4) (speed, reliability),
- improve at91_spi(4)/spi(4) (speed),
- improve MD bus_dma(9) implementation (cache flushing),
- unb0rk usb(4) (bus_dma(9) usage),
- add Atmel® AT91 support for nand(4)

Thank you for your attention!

ask questions

[1] *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Std. 802.3, 2008. [Online]. Available: http://standards.ieee.org/about/get/802/802.3.html

[2] *Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*, EIA Std. RS-232-C, 1969.

[3] *Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems*, EIA Std. RS-485, 1983.

[4] *Test Access Port and Boundary-Scan Architecture*, Institute of Electrical and Electronics Engineers Computer Society Std. 1149.1.1, 1990.

[5] "Transmission control protocol," RFC 793, Information Sciences Institute, University of Southern California, Sep. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt

[6] "Internet protocol," RFC 791, Information Sciences Institute, University of Southern California, Sep. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc791.txt

[7] (2006, Nov.) Ethernut version 1.3 Revision G board. Figure. egnite Software GmbH. [Online]. Available: http://www.ethernut.de/img/ethernut13g-large.jpg

[8] (2006, Nov.) Ethernut version 2.1 Revision B board. Figure. egnite Software GmbH. [Online]. Available: http://www.ethernut.de/img/ethernut21b_large.jpg

[9] (2009, Oct.) Ethernut version 3.1 Revision D board. Figure. egnite Software GmbH. [Online]. Available: http://www.ethernut.de/img/ethernut31d-t1030.png

[10] (2011, May) Ethernut version 5.0 Revision F board. Figure. egnite Software GmbH. [Online]. Available: http://www.ethernut.de/img/ethernut50f_top_large.png

[11] (2004, Jul.) Eagle 4.11 schematics and layout file for Ethernut 1.3 Revision G. egnite Software GmbH. [Online]. Available: http://www.ethernut.de/arc/enut130g.zip

[12] CadSoft. Website. CadSoft Computer GmbH. [Online]. Available: http://www.cadsoft.de

[13] Nut/OS Features. Website. egnite GmbH. [Online]. Available: http://www.ethernut.de/en/firmware/nutos.html

[14] *Information technology – Portable Operating System Interface (POSIX®) – Base Specifications, Issue 7*, Institute of Electrical and Electronics Engineers Computer Society Std. 1003.1, 2008.

[15] Nut/OS example Hello World! egnite GmbH. [Online]. Available: http://www.ethernut.de/nutwiki/Hello_World!