Headless systems

msaitoh@netbsd.org

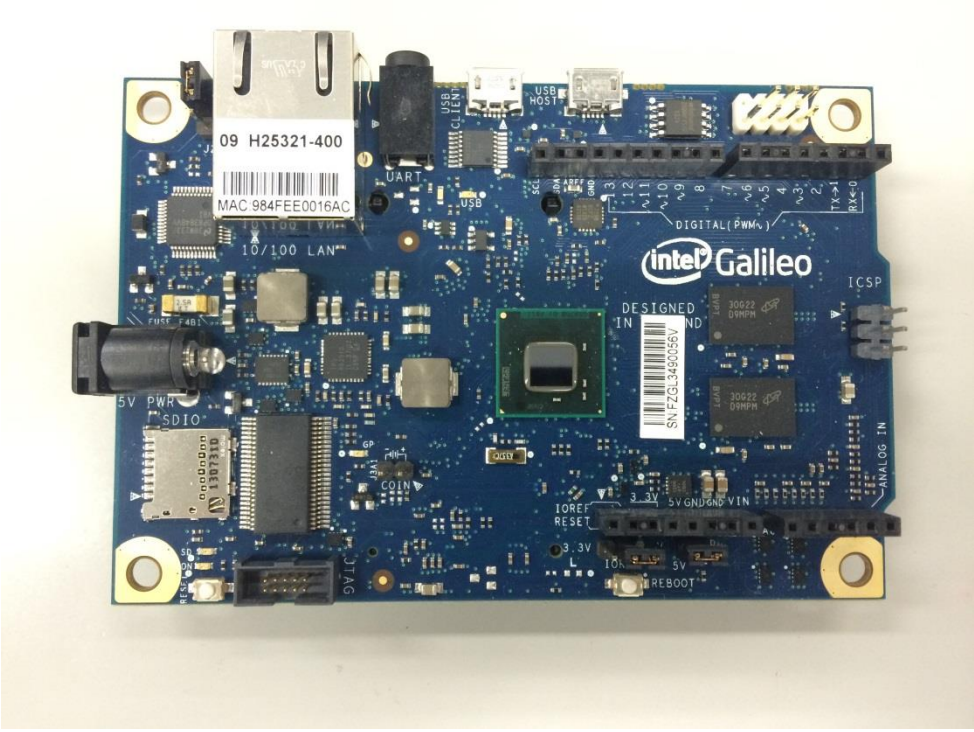
What comes next?

- Sensor network
- VERY_SMALL_DEVICES

Such devices have no video console

Intel Quark processor

- Small and low-power processor
- Intel Galileo
  - Arduino compatible
- Intel Edison
  - SD card size



UEFI only?

- In last December, I knew Intel Galileo's UEFI doesn't support BIOS compatible function.
- And then, I knew MinnowBoard was UEFI only.
- The relase date of Galileo was in January, so I bought MinnowBoard as a test machine of UEFI boot.

products

Product	Company	CPU	Configuration	BIOS or UEFI	video	Serial Console
MinnowBoard	Intel	Atom E600	+EG20T	UEFI only	E600 internal	puc(4)
X9SBAA-F	Supermicro	Atom S1200	1 chip SoC	UEFI+BIOS	BMC	puc(4)
Galileo	Intel	Quark X1000	1 chip SoC	UEFI only	No	puc(4)
86duino	DMP Electronics	Vortex 86EX	1 chip SoC	BIOS	No	isa(4)

Not headless

headless

86duino is not so difficult.

86duino

UEFI boot

- jak@ is working to support UEFI boot.
- I got a test binary from him to boot NetBSD on some devices.
- UEFI bootloader for IA32!
  - Not for x86_64!

UEFI shell

- Variety of commands
- Some of useful commands
  - help
  - ver
  - devices
  - devtree
  - pci
  - drivers
  - smbiosview -a
  - dmpstore
  - memmap

Examples of UEFI shell output

- MinnowBoard
  - http://pastebin.com/Le2AAPby
- Intel Galileo
  - http://pastebin.com/FjpKRtEQ
  - http://pastebin.com/jXyadkXq

One of my SD card

```
five# ls -l /mnt
total 13210
drwxr-xr-x  1 root  wheel        2048 Jul 10   2013 EFI
-rwxr-xr-x  1 root  wheel          75 Jul 10   2013 ID.txt
-rwxr-xr-x  1 root  wheel  13217853 Jan 18  01:03 netbsd
-rwxr-xr-x  1 root  wheel          13 Jan  3   2001 origstartup.nsh
-rwxr-xr-x  1 root  wheel          13 Jul 10   2013 startup.nsh
-rwxr-xr-x  1 root  wheel      148736 Jan 16 22:47 uefiboot_i386-20140115.efi
-rwxr-xr-x  1 root  wheel      148608 Jan 13 06:40 uefiboot_i386.efi
```

Loader’s output on MinnowBoard

UEFI Interactive Shell v2.0. UEFI v2.31 (Firmware Version 1.00, 0x4E278000). Revision 1.02

Mapping table

FS0: Alias(s):HD18b:;BLK1:

PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)/HD(1,GPT,A16FE367-D6CB-400D-AFD5-7BF515F6F4CE,0x800,0x9000)

BLK0: Alias(s):

PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)

BLK2: Alias(s):

PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)/HD(2,GPT,4C14C54B-ACE4-49D9-B190-5464D768712B,0x9800,0x6DF800)

BLK3: Alias(s):

PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)/HD(3,GPT,5AF355CB-1D17-4634-9E74-DBCF60EA36F6,0x6E9000,0x5C800)

Press ESC in 3 seconds to skip startup.nsh or any other key to continue.

2.0 Shell> fs0:

2.0 FS0:¥> uefiboot_i386-20140115.efi

Loader’s output on MinnowBoard (2)

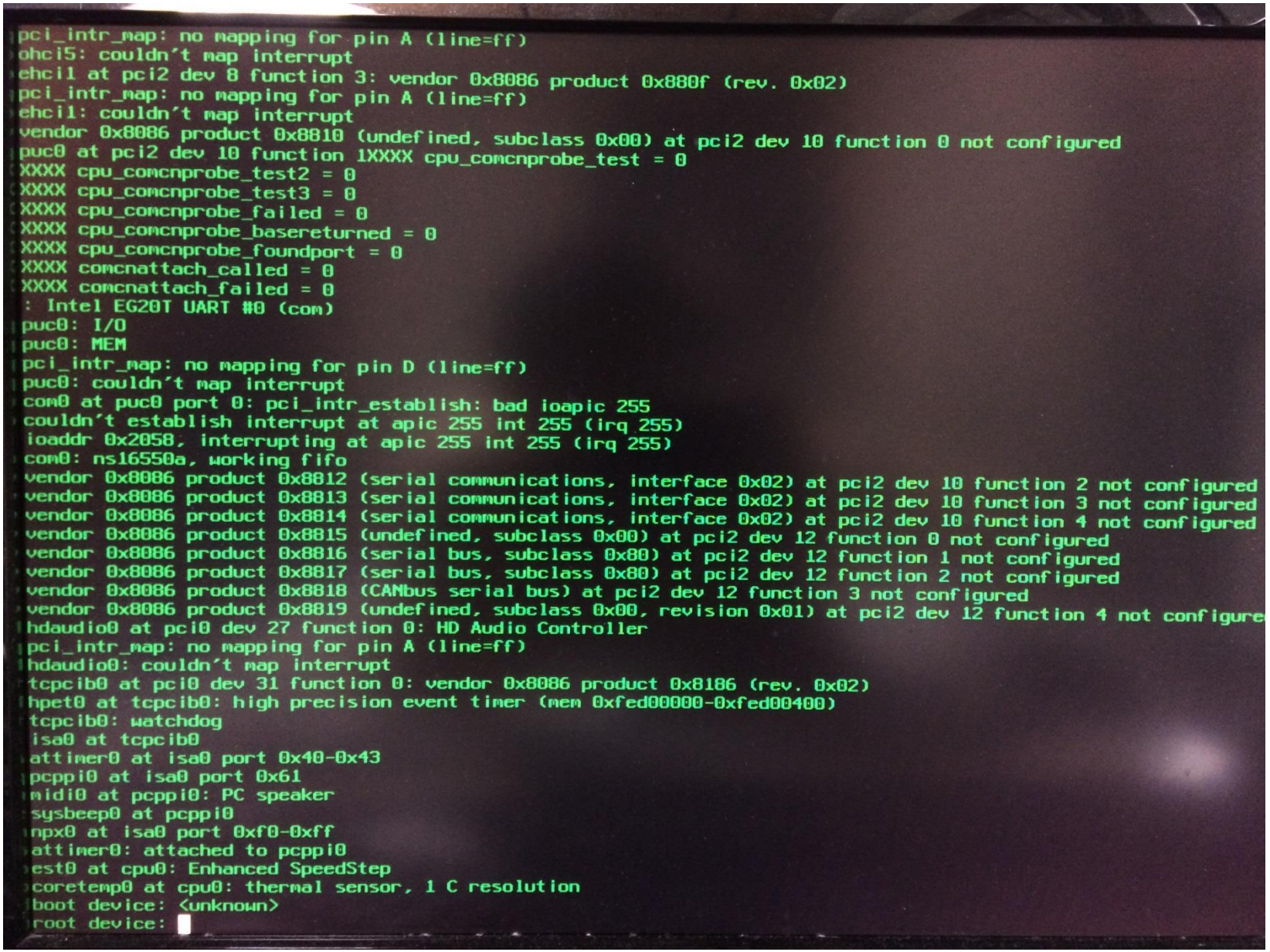
```
2.0 FS0:¥> uefiboot_i386-20140115.efi
alloc 0x3df54f90 54
alloc 0x3df54f10 8
main entry
main stack canary 0x3dee7047
main _start 0x3dec6240
main _stack_top 0x3dee70a0
alloc 0x3df4b010 10464

>> NetBSD/i386 UEFI Boot, Revision 0.0 (from NetBSD 6.99.28)
>> Memory: 636/1034440 k
get_gop GOP OpenProtocol status 80000003
2 4
Handles[0] 0x3e4f5d90
Handles[1] 0x3e615890
get_gop GOP OpenProtocol status 0
dealloc 0x3df55510 8
mi 0 0 640 480 1 ( ff0000 ff00 ff ff000000 ) 640
(snip)
mi 50 0 1920 1080 1 ( ff0000 ff00 ff ff000000 ) 1920
Mode 18 MaxMode 51
FrameBufferBase 3f800000 FrameBufferSize 7b0000
mi 0 1024 768 1 ( ff0000 ff00 ff ff000000 ) 1024
gop_info 32 4096
(snip)
```

```
argc 1
arg 0: uefiboot_i386-20140115.efi

acpi table 0x3f3d8000
00000000 52 53 44 20 50 54 52 20 59 49 4e 54 45 4c 20 00
00000010 00 70 3d 3f 52 53 44 20 50 54 52 20 e3 49 4e 54
00000020 45 4c 20 02 74 70 3d 3f 24 00 00 00 e8 70 3d 3f
00000030 00 00 00 00 08 00 00 00 00 10 00 10 04 00 40 00
acpi20 table 0x3f3d8014
00000000 52 53 44 20 50 54 52 20 e3 49 4e 54 45 4c 20 02
00000010 74 70 3d 3f 24 00 00 00 e8 70 3d 3f 00 00 00 00
00000020 08 00 00 00 00 10 00 10 04 00 40 00 02 00 10 00
00000030 00 02 20 00 80 00 10 00 00 00 00 30 30 01 00 00
> boot
alloc 0x3df48f10 132
devopen
devopen after parsebootfile() ufs hd 0 0 netbsd
00000000 02 01 0c 00 d0 41 03 0a 00 00 00 00 01 01 06 00
00000010 00 17 01 01 06 00 00 00 01 01 06 00 00 04 7f ff
00000020 04 00
tdpath: PciRoot(0x0)/Pci(0x17,0x0)/Pci(0x0,0x0)/Pci(0x4,0x0)
dealloc 0x3df4ed10 0
alloc 0x3df47010 2588
efidisk_open 0x3df47010
efidisk_open BlockSize 200
efidisk_open about to read_partitions()
(snip)
```

Video output on MinnowBoard



Problem for debugging

- Serial console didn’t work, so it was difficult to debug.
- MinnowBoard has video (HDMI), but the output was too fast to see ☹
- My video capture card didn’t work with MinnowBoard’s HDMI output ☹
- I recorded movie using with iPhone5s (120fps) and checked it.

ACPI root pointer

NetBSD 6.99.28 (SMALL) #43: Sat Jan 18 01:….

msaitoh@xxx.example.com:/disk/sources...

total memory = 1010 MB

avail memory = 980 MB

timecounter: Timecounters tick every 10.000 msec

Generic PC

mainbus0 (root)

RTC BIOS diagnostic error 0x37<config_unit,memory_size,invalid_time>

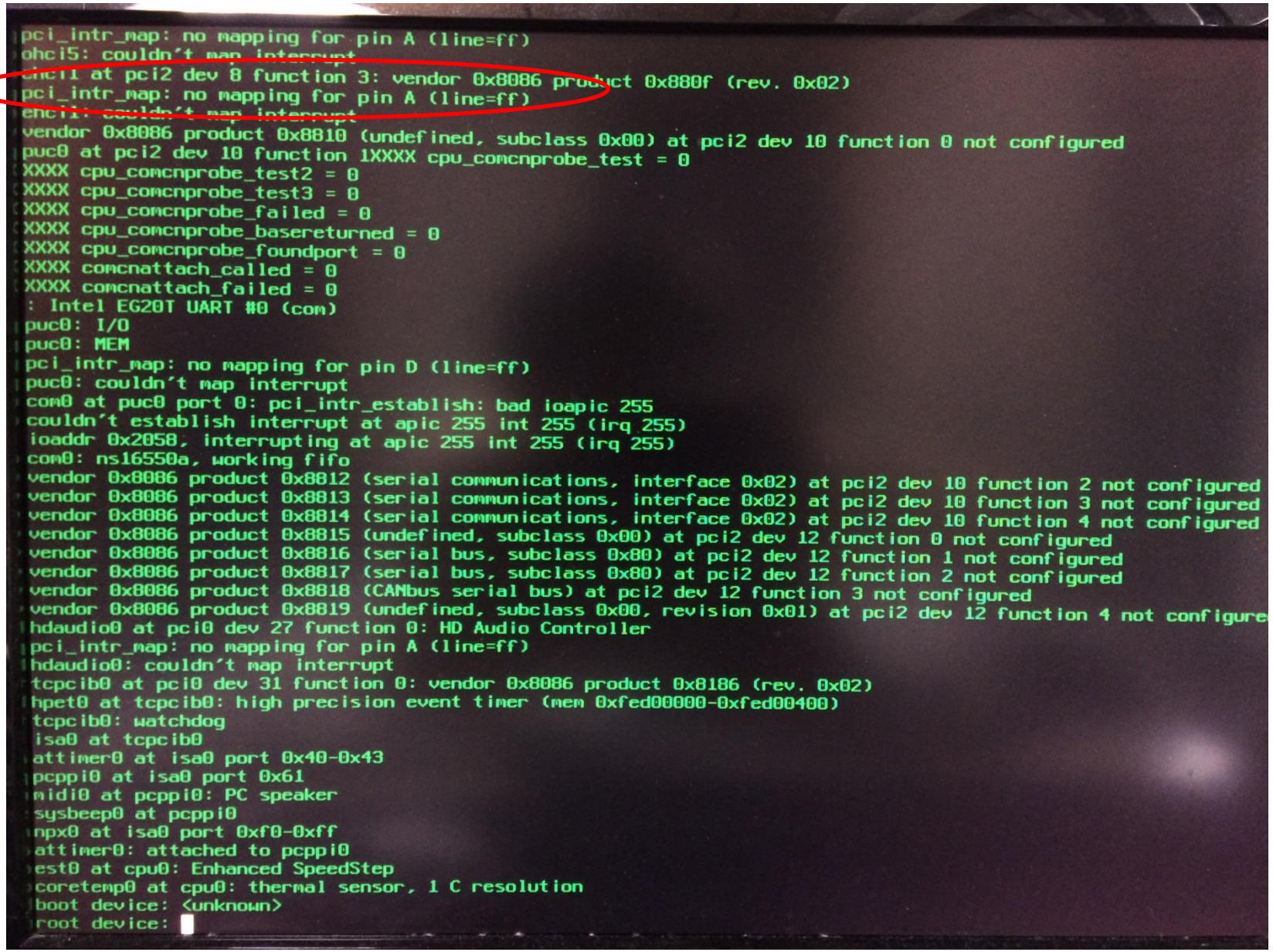
ACPI BIOS Error (bug): A valid RSDP was not found (20131218/tbxfroot-223)

acpi_probe: failed to initialize tables

ACPI Error: Could not remove SCI handler (20131218/evmisc-331)

Remember. Serial console couldn't be used, so I checked this output with my eyes and wrote by hand.

Video output on MinnowBoard



puccn problem

- What is the problem?
  - com device via puc(4) can be used, but the code is very dirty ☹
  - (perhaps) a lot of people include me think that “I don’t want to touch that code because I might break sys/dev/ic/com.c”

Why puccn is important

- Intel PCHs have no ISA serial port.
  - Serial console can’t be used on a lot of new machines ☹
- Intel Galileo’s serial ports are puc(4) com. And more, it has no video console!!!

One of solution for puccn problem

- PR#48016
  - http://gnats.netbsd.org/cgi-bin/query-pr-single.pl?number=48016
  - This PR was resolved 1 month ago.

```
>Responsible:    msaitoh
>State:          closed
>Class:          sw-bug
>Submitter-Id:   net
>Arrival-Date:   Fri Jul 05 00:25:00 +0000 2013
>Closed-Date:    Thu Feb 06 15:53:21 +0000 2014
```

One of bad point in puccn

sys/arch/x86/pci/pci_machdep.c

```
#if NCOM > 0
int
cpu_comcnprobe(struct consdev *cn, struct pci_attach_args *pa)
{
	pci_mode_detect();
	pa->pa_iot = x86_bus_space_io;
	pa->pa_memt = x86_bus_space_mem;
	pa->pa_pc = 0;
#if 0
	pa->pa_tag = pci_make_tag(0, 0, pci_bus_maxdevs(NULL, 0) - 1, 0);
#else
	pa->pa_tag = pci_make_tag(0, 2, pci_bus_maxdevs(NULL, 0) - 1, 0);
#endif
	return 0;
}
#endif
```

- Only bus number 0's bus is scanned ☹
- On E600+EG20T, pci bus 0 is used in E600,
- The bus number of puc com in EG20T is 2, so it doesn't match :-(

Variation of com device

- I/O mapped(A)
  - inb, outb
  - PCI_MAPREG_TYPE_IO
    - Almost all PCI com devices use this type.
- memory mapped
  - Sometimes used in SoC's com.
  - PCI_MAPREG_TYPE_MEM
  - Byte register is
    - 1byte stride(B)
    - 4byte stride(C)
      - Galileo is (C)
      - Small number of SoC use this type.

My patch

- I sent a patch to tech-kern two month ago.
- I got a feedback from tsutsui@
- I modified my patch from his advice, but not tested yet because I have been busy for more than one month.
- I’ll test my patch with some boards and will send new email in this month.

Future work

- Support on-chip Ethernet device in Quark
- Support GPIO in Quark
- Buy Intel Edison
  - Though I don’t know when the board will be released.
- (someone) make a nice application using with GPIO or something else.