Использование Linux во встроенных системах

Шибанов Владимир Александрович,
сотрудник компании D-Link,
доцент каф. САПР ВС, к.т.н.

D-Link®
Building Networks for People

Вместо введения. Где обитает Linux?

“Места обитания” Linux

- Персональные компьютеры (desktop, notebook),
- Серверы (LAMP),
- Суперкомпьютеры,
- **Встроенные системы**.

Понятие встроенной системы

Встроенная система – это специализированная компьютерная система, предназначенная для выполнения одной или нескольких отдельных функций, часто – с ограничениями на обработку в режиме реального времени.

Обычно такая система интегрируется непосредственно в устройство которым она управляет.

В отличие от специализированной системы, универсальное устройство, такое, например, как персональный компьютер, может выполнять множество различных задач в зависимости от установленных программ.

Примеры встроенных систем

Электронная бытовая техника

маршрутизаторы для домашних сетей, DVD-плейеры, телевизоры, цифровые камеры, GPS-навигаторы, видеокамеры с функцией записи, мобильные телефоны, микроволновые печи...

Промышленная техника

устройства для механизированного управления, сигнализации, системы наблюдения, автомобили, железнодорожная и авиатехника, спутники...

Примеры встроенных систем

Особенности встроенных систем

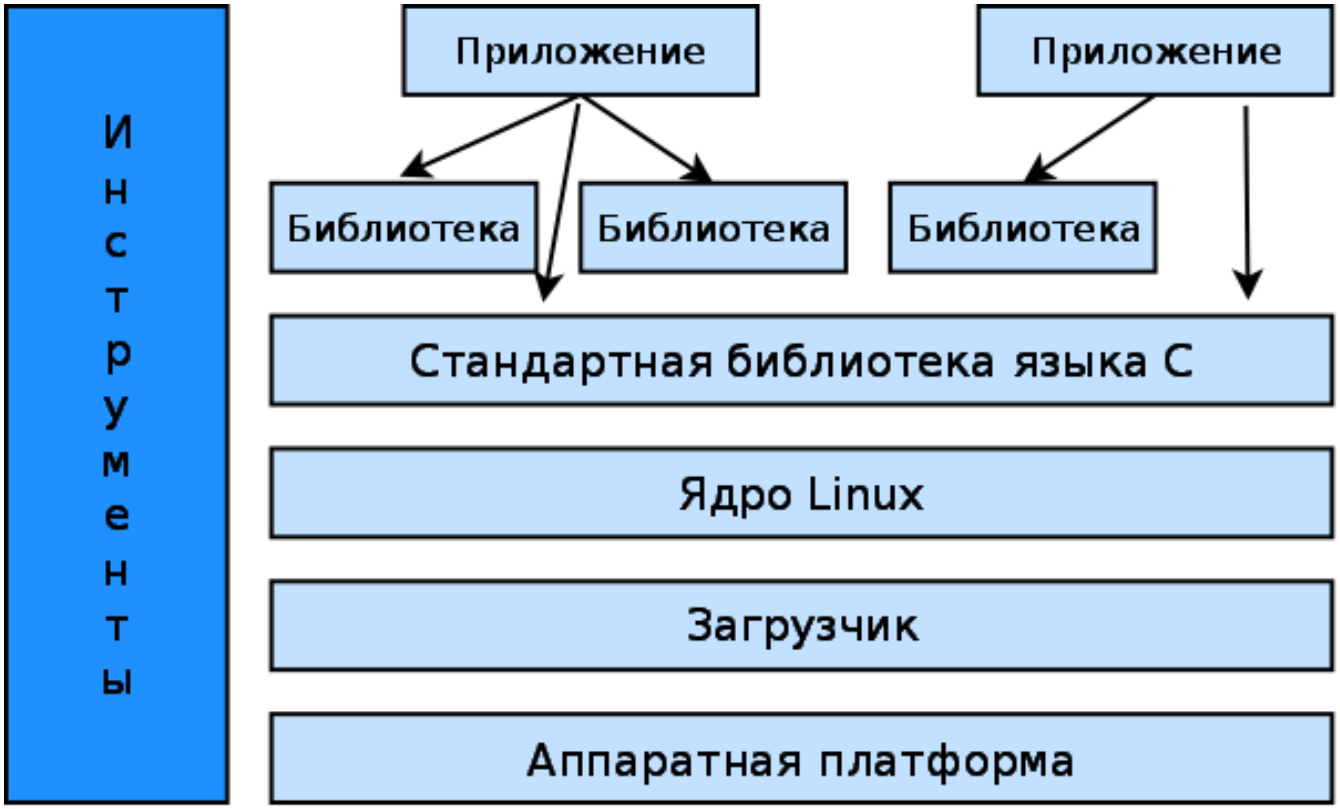
Аппаратные платформы встроенных систем отличаются от аппаратных платформ настольных систем:

- Другая архитектура процессора: чаще всего используется архитектура **ARM**, **MIPS** или **PowerPC**. **x86** также используется.
- Запоминающие устройства на **flash** типа NOR или NAND, чаще всего ограниченного объема (от нескольких KB до сотен MB)
- Ограниченный объем оперативной памяти (**RAM**) (от нескольких MB до десятков MB)
- Множество коммуникационных шин, не характерных для настольных систем: **I2C**, **SPI**, **SSP**, **CAN** и др.

Минимальные требования к аппаратной платформе для работы Linux

- Процессор, поддерживаемый компилятором **gcc** и **ядром Linux**: 32-битный процессор. Процессоры без MMU также поддерживаются в проекте **uClinux**,
- Несколько мегабайтов оперативной памяти: минимум - 4 MB, для нормальной работы - 8 MB,
- Несколько мегабайтов на постоянном запоминающем устройстве: минимум - 2 MB, для нормальной работы - 4 MB.

Обобщенная программная архитектура



Направления работы проектировщика встроенных систем

Разработка Пакета поддержки аппаратной платформы (Board Support Package)

BSP включает загрузчик и ядро с необходимыми драйверами устройств для целевой платформы

Системная интеграция

Интеграция компонентов: загрузчика, ядра, сторонних библиотек и приложений и приложений собственной разработки в единую рабочую систему

Разработка приложений

Обычные приложения Linux, но использующие специальные библиотеки

Корневая файловая система

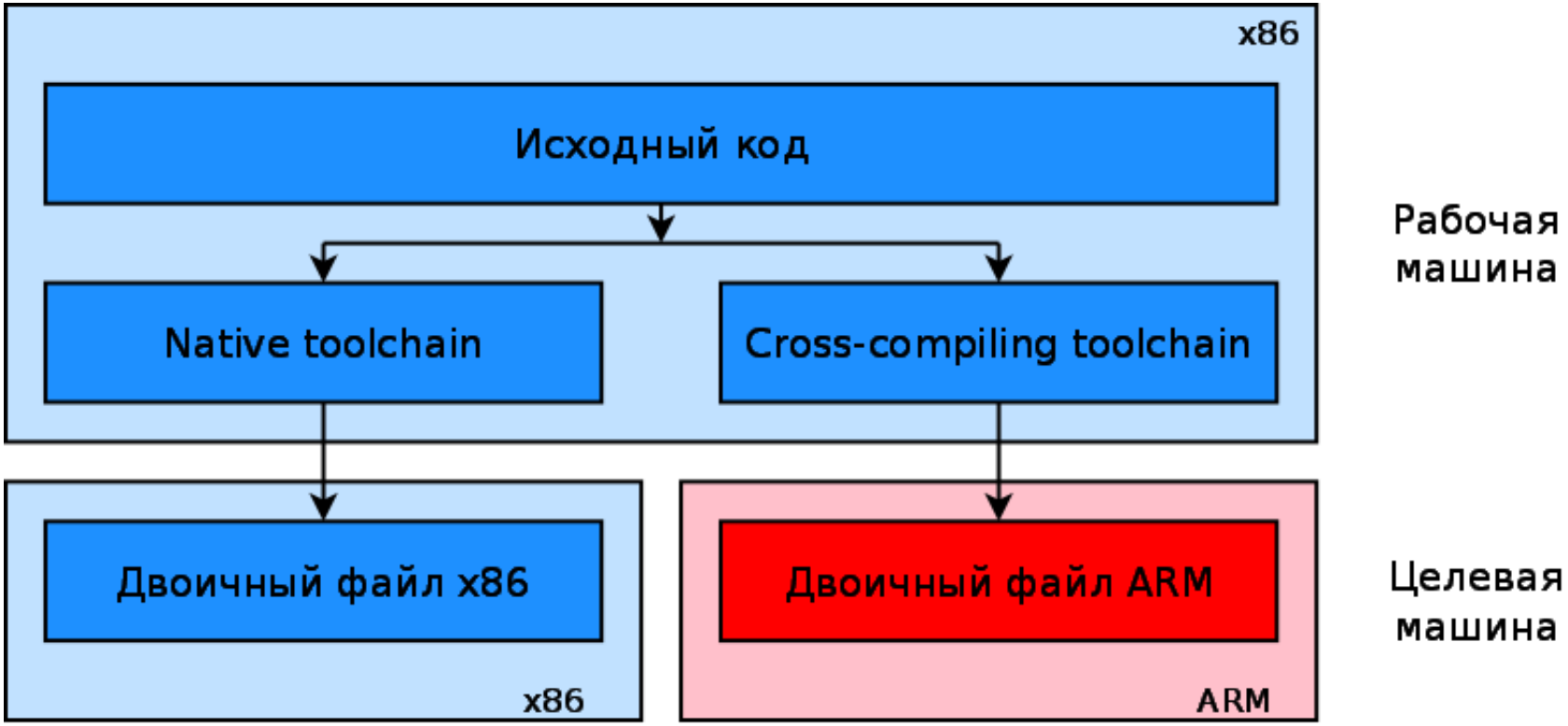
- В Linux различные файловые системы монтируются и создают глобальную иерархию файлов и каталогов.
- Основная файловая система, называемая **корневой** монтируется в **/**.
- Во встроенных системах эта корневая файловая система содержит все библиотеки, приложения и данные системы.
- Поэтому создание корневой файловой системы - это одна из основных задач интеграции компонентов встроенной Linux-системы в устройство.
- Ядро операционной системы обычно хранится отдельно.

Особенности разработки встроенных систем

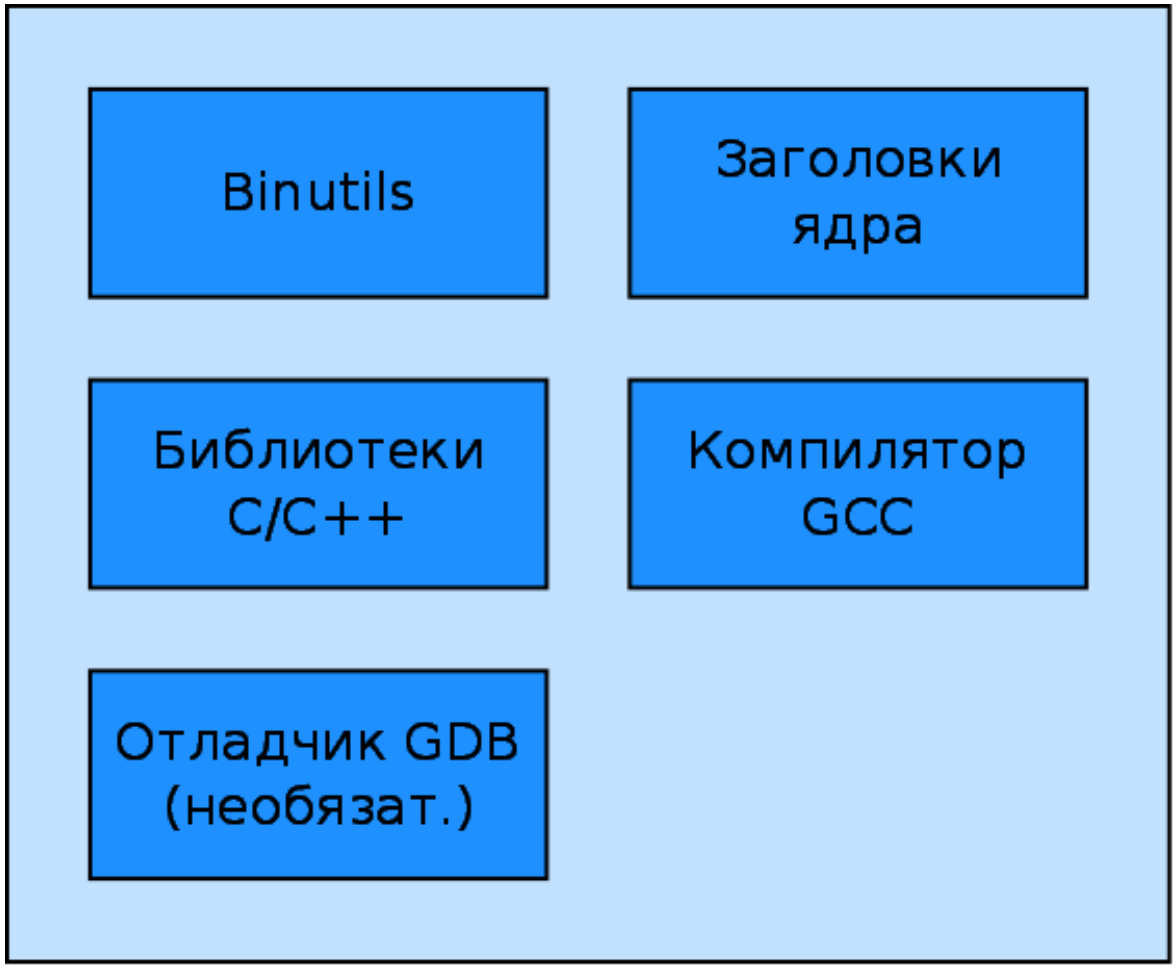
Встроенная Linux-система - это **обычная** Linux-система обычно **с меньшим набором компонентов**. Методы разработки приложений для встроенной Linux точно такие же, как и для настольных систем. Все существующие библиотеки - собственной или сторонней разработки могут быть интегрированы во встроенную систему. Конечно, с учетом ограничений встроенных систем по производительности и объему оперативной и постоянной памяти.

- Основным языком программирования для системных приложений как правило является **язык С**. Библиотека языка С уже присутствует в системе и не нужно ничего к ней добавлять.
- **Язык С++** может быть использован для больших приложений. При этом в систему нужно добавить библиотеку С++. Некоторые библиотеки, например **Qt** разработаны на **С++** и в любом случае требуют наличия **библиотеки С++**.
- **Скриптовые языки** также могут быть полезны для быстрой разработки приложений, web-приложений и скриптов. Они требуют наличия во встроенной системе интерпретатора и обычно обладают большим потреблением памяти и меньшей производительностью. Языки: **Shell**, **Python**, **Perl**, **Ruby**, **TCL**, **PHP** и др.

Native toolchain и Cross-Compiling toolchain



Компоненты toolchain



Загрузчики

Загрузчик – это программа, которая выполняет:

- базовую инициализацию аппаратуры,
- загрузку двоичного файла приложения, обычно – ядра операционной системы, с flash-устройства, из сети, или с другого устройства,
- декомпрессию двоичного файла приложения,
- запуск приложения.

Кроме реализации этих функций загрузчики часто включают командную оболочку, позволяющую выполнять различные действия, например, загрузить данные с устройств или из сети, проверить память, выполнить диагностику аппаратных устройств.

Процесс загрузки во встроенных архитектурах может сильно отличаться в зависимости от используемой платформы.

Существует множество open source загрузчиков, наиболее распространенными из которых являются:

U-Boot – стандарт де-факто. Предназначен для платформы ARM, но так же используется на PowerPC, MIPS, x86 и др.

Barebox – разрабатывался как усовершенствование U-Boot, имеет лучший дизайн, активно развивается, но имеет худшую аппаратную поддержку, чем U-Boot.

Компиляция ядра Linux

Скачивание исходных кодов ядра

Исходные файлы ядра Linux можно свободно скачать с сайта http://www.kernel.org. Исходные файлы ядра Linux версии 3.6.4: несжатые файлы: 446 МБ (42100 файлов), bzip2 (сжатый tar-архив): 78,5 МБ. Однако большую часть ядра составляют необязательные элементы (драйверы устройств, сетевые протоколы). Скомпилированное ядро для встроенной системы может занимать порядка 500 КБ.

Конфигурация ядра

При конфигурации определяют, какие функции необходимо включить в ядро. Набор опций зависит от используемого оборудования и от функций, которые вы хотите включить в ядро. Конфигурация хранится в файле .config в корневом каталоге исходных файлов ядра. Наиболее полезные команды для создания этого конфигурационного файла:

```
make [xconfig|gconfig|menuconfig|oldconfig]
```

Компиляция ядра Linux

Компиляция ядра

Компиляция ядра запускается командой

make

Генерируются

vmlinux, несжатый образ ядра в формате ELF, который полезен для целей отладки, но не может использоваться для загрузки

arch/<arch>/boot/*Image, окончательный, как правило сжатый образ ядра, с которого может выполняться загрузка.

bzImage для x86, zImage для ARM, и т.п. Все модули ядра заданные параметрами конфигурации в виде файлов .ko

Установка ядра и отдельных модулей

sudo make install

Выполняет инсталляцию ядра на рабочем компьютере

sudo make modules_install

Выполняет установку модулей в системные каталоги

Компиляция ядра Linux

Кросс-компиляция ядра

Кросс-компиляция ядра – это компиляция ядра Linux для другой архитектуры процессора.

В названиях модулей кросс-компилятора присутствуют имена целевой системы, архитектуры и, в некоторых случаях, библиотеки. Примеры:

- mips-linux-gcc
- m68k-linux-uclibc-gcc
- arm-linux-gnueabi-gcc

Целевую платформу и компилятор нужно указывать во всех рассмотренных командах:

```
make ARCH=arm CROSS_COMPILE=arm-linux- xconfig
make ARCH=arm CROSS_COMPILE=arm-linux-
make ARCH=arm CROSS_COMPILE=arm-linux- modules_install
```

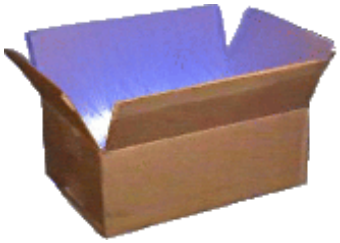
BusyBox

BusyBox представляет собой единственный исполняемый файл, при запуске которого загружается полноценный командный интерфейс, включающий множество традиционных команд UNIX. **BusyBox** даже включает web-сервер!

При этом он занимает всего **500 KB** (при статической компиляции с **uClibc**) или **1 MB** (при статической компиляции с **glibc**)!

Можно выбрать, какие компоненты нужно включить в **BusyBox**.

Исходники программы можно скачать с сайта **www.busybox.net** .



Команды, включенные в BusyBox

[, [[, acpid, add-shell, addgroup, adduser, adjtimex, ar, arp, arping, awk, base64, basename, bbconfig, beep, blkid, blockdev, bootchartd, brctl, bunzip2, bzcat, bzip2, cal, cat, catv, chat, chattr, chgrp, chmod, chown, chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm, conspy, cp, cpio, crond, crontab, cryptpw, cttyhack, cut, date, dc, dd, deallocvt, delgroup, deluser, depmod, devfsd, devmem, df, dhcprelay, diff, dirname, dmesg, dnsd, dnsdomainname, dos2unix, dpkg, dpkg-deb, du, dumpkmap, dumpleases, echo, ed, egrep, eject, env, envdir, envuidgid, ether-wake, expand, expr, fakeidentd, false, fbset, fbsplash, fdflush, fdformat, fdisk, fgconsole, fgrep, find, findfs, flash_eraseall, flash_lock, flash_unlock, flashcp, flock, fold, free, freeramdisk, fsck, fsck.minix, fsync, ftpd, ftpget, ftpput, fuser, getopt, getty, grep, gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid, hostname, httpd, hush, hwclock, id, ifconfig, ifdown, ifenslave, ifplugd, ifup, inetd, init, inotifyd, insmod, install, ionice, iostat, ip, ipaddr, ipcalc, ipcrm, ipcs, iplink, iproute, iprule, iptunnel, kbd_mode, kill, killall, killall5, klogd, last, length, less, linux32, linux64, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, losetup, lpd, lpq, lpr, ls, lsattr, lsmod, lspci, lsusb, lzcat, lzma, lzop, lzopcat, makedevs, makemime, man, md5sum, mdev, mesg, microcom, mkdir, mkdosfs, mke2fs, mkfifo, mkfs.ext2, mkfs.minix, mkfs.reiser, mkfs.vfat, mknod, mkpasswd, mkswap, mktemp, modinfo, modprobe, more, mount, mountpoint, mpstat, msh, mt, mv, nameif, nanddump, nandwrite, nbd-client, nc, netstat, nice, nmeter, nohup, nslookup, ntpd, od, openvt, passwd, patch, pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill, pmap, popmaildir, poweroff, powertop, printenv, printf, ps, pscan, pwd, raidautorun, rdate, rdev, readahead, readlink, readprofile, realpath, reboot, reformime, remove-shell, renice, reset, resize, rev, rfkill, rm, rmdir, rmmod, route, rpm, rpm2cpio, rtcwake, run-parts, runlevel, runsv, runsvdir, rx, script, scriptreplay, sed, sendmail, seq, setarch, setconsole, setfont, setkeycodes, setlogcons, setsid, setuidgid, sh, sha1sum, sha256sum, sha512sum, showkey, slattach, sleep, smemcap, softlimit, sort, split, start-stop-daemon, stat, strings, stty, su, sulogin, sum, sv, svlogd, swapoff, swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar, taskset, tcpsvd, tee, telnet, telnetd, test, tftp, tftpd, time, timeout, top, touch, tr, traceroute, traceroute6, true, tty, ttysize, tunctl, tune2fs, ubiattach, ubidetach, udhcpc, udhcpd, udpsvd, umount, uname, uncompress, unexpand, uniq, unix2dos, unlzma, unlzop, unxz, unzip, uptime, usleep, uudecode, uuencode, vconfig, vi, vlock, volname, wall, watch, watchdog, wc, wget, which, who, whoami, xargs, xz, xzcat, yes, zcat, zcip

Для версии 1.18.

Пример встроенной системы: ADSL-роутер D-Link DSL-2500U

Рассмотрим работу со встроенной Linux-системой на примере роутера **D-Link DSL-2500U**.

Данное устройство соответствует стандарту ADSL2/2+ и поддерживающий скорость входящего потока данных до 24 Мбит/с. Поддерживается протокол Bridged Ethernet по ATM, IPoA, PPPoA, PPPoE. Роутер обеспечивает защиту межсетевым экраном (NAT/SPI/DoS) и систему обнаружения вторжений.

Общие сведения о роутере D-Link DSL-2500U:

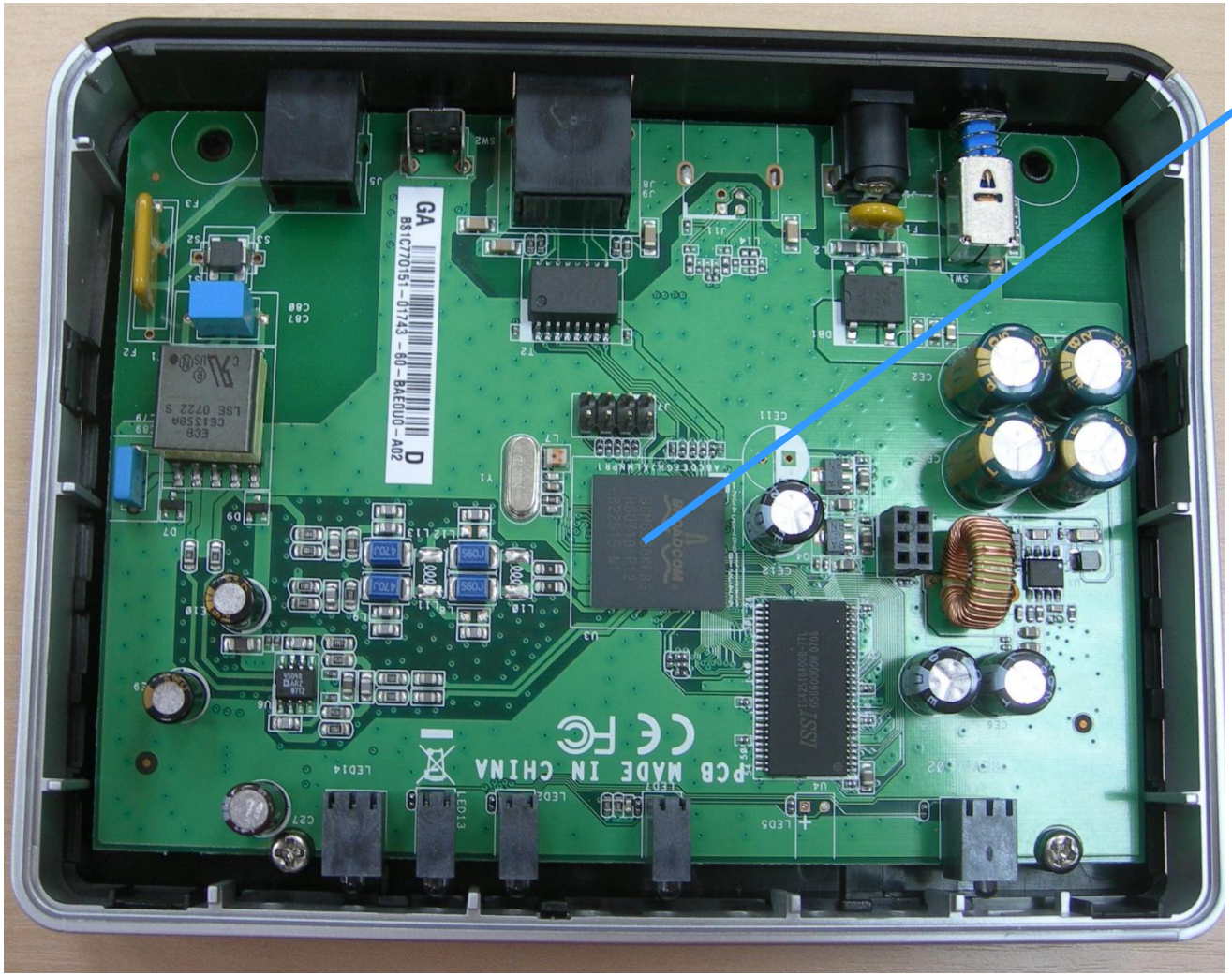
Устройство имеет интерфейсы:

- 1 порт ADSL с разъемом RJ-11
- 1 порт LAN 10/100BASE-TX Ethernet с разъемом RJ-45

Настройка роутера осуществляется через Web-интерфейс. Так же поддерживается протокол SNMP.

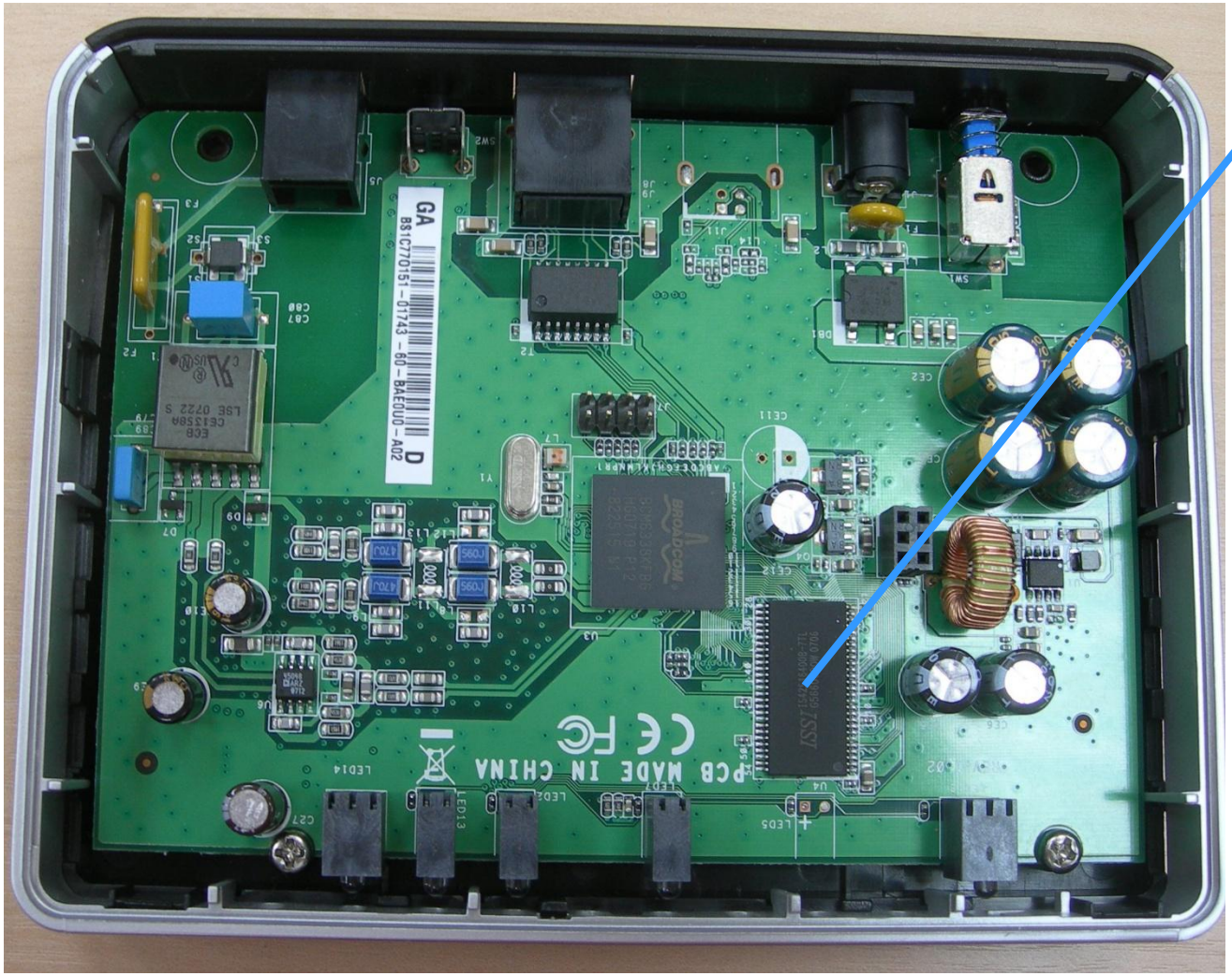
Обновление программного обеспечения может осуществляться через Web-интерфейс или по протоколу TFTP.

“Внутренности” роутера

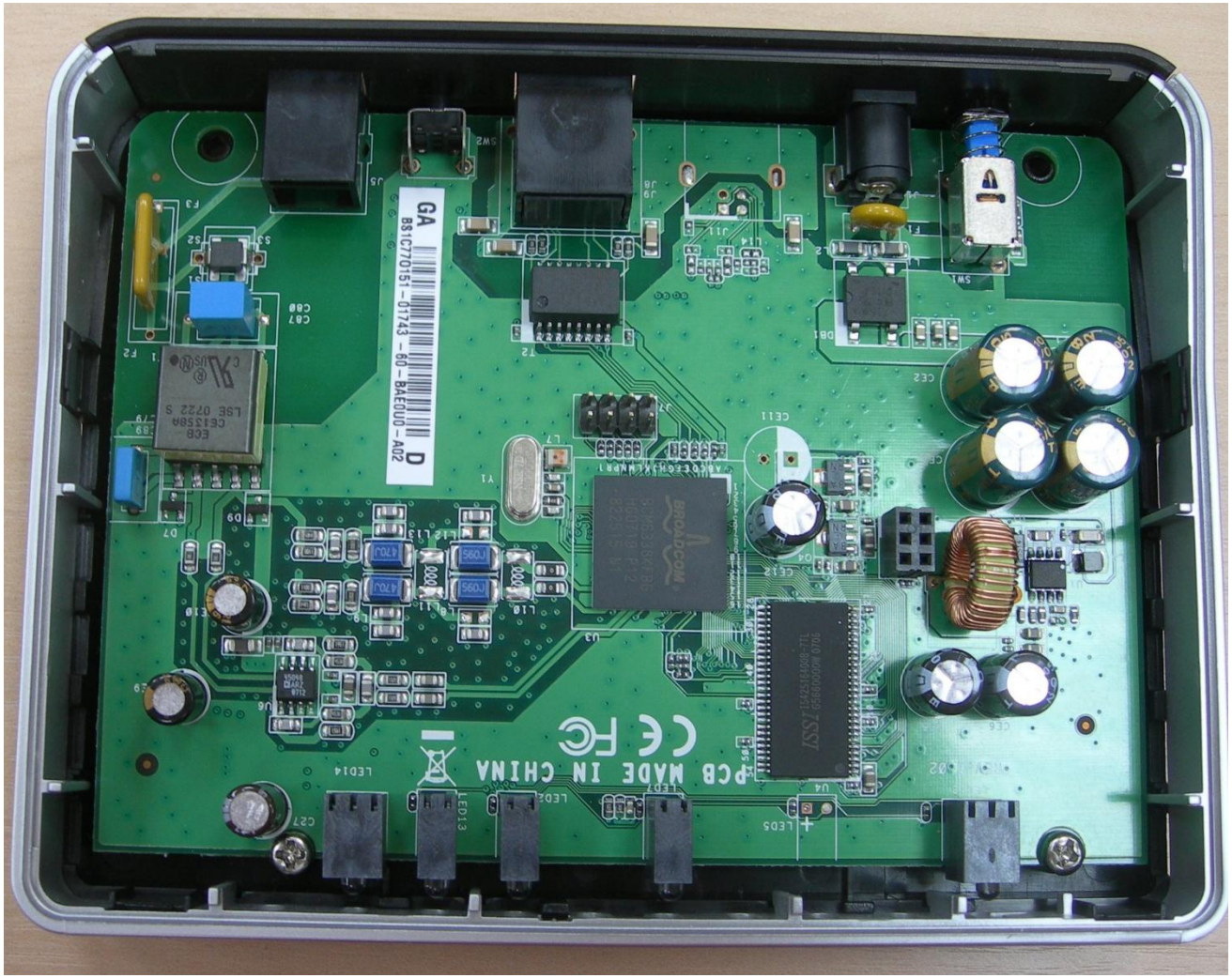


Микропроцессор Broadcom BCM6338 – высокопроизво-дительный микропроцессор MIPS32 с поддержкой ADSL2+ и интерфейсами 10/100 Ethernet и USB 1.1

“Внутренности” роутера

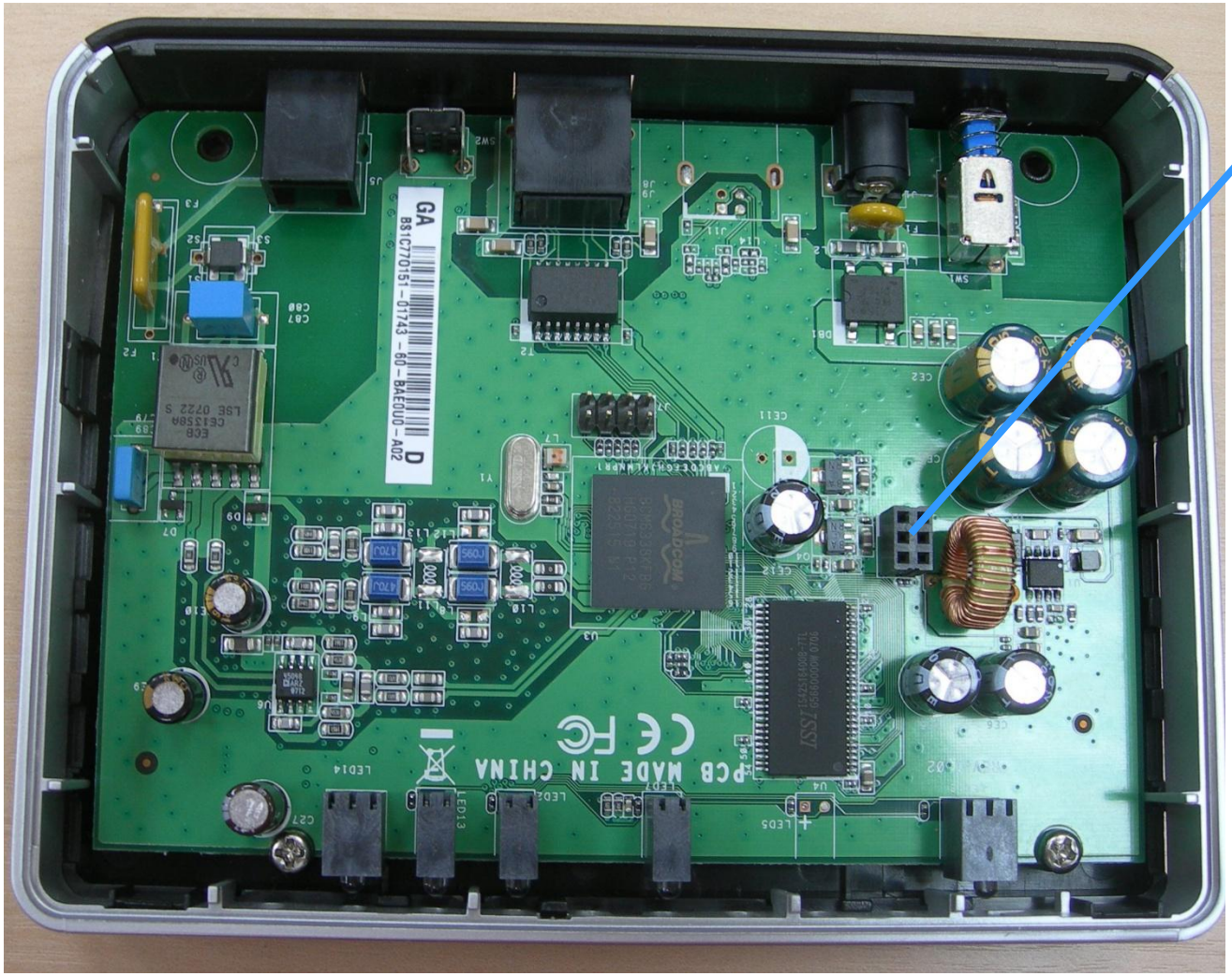


“Внутренности” роутера

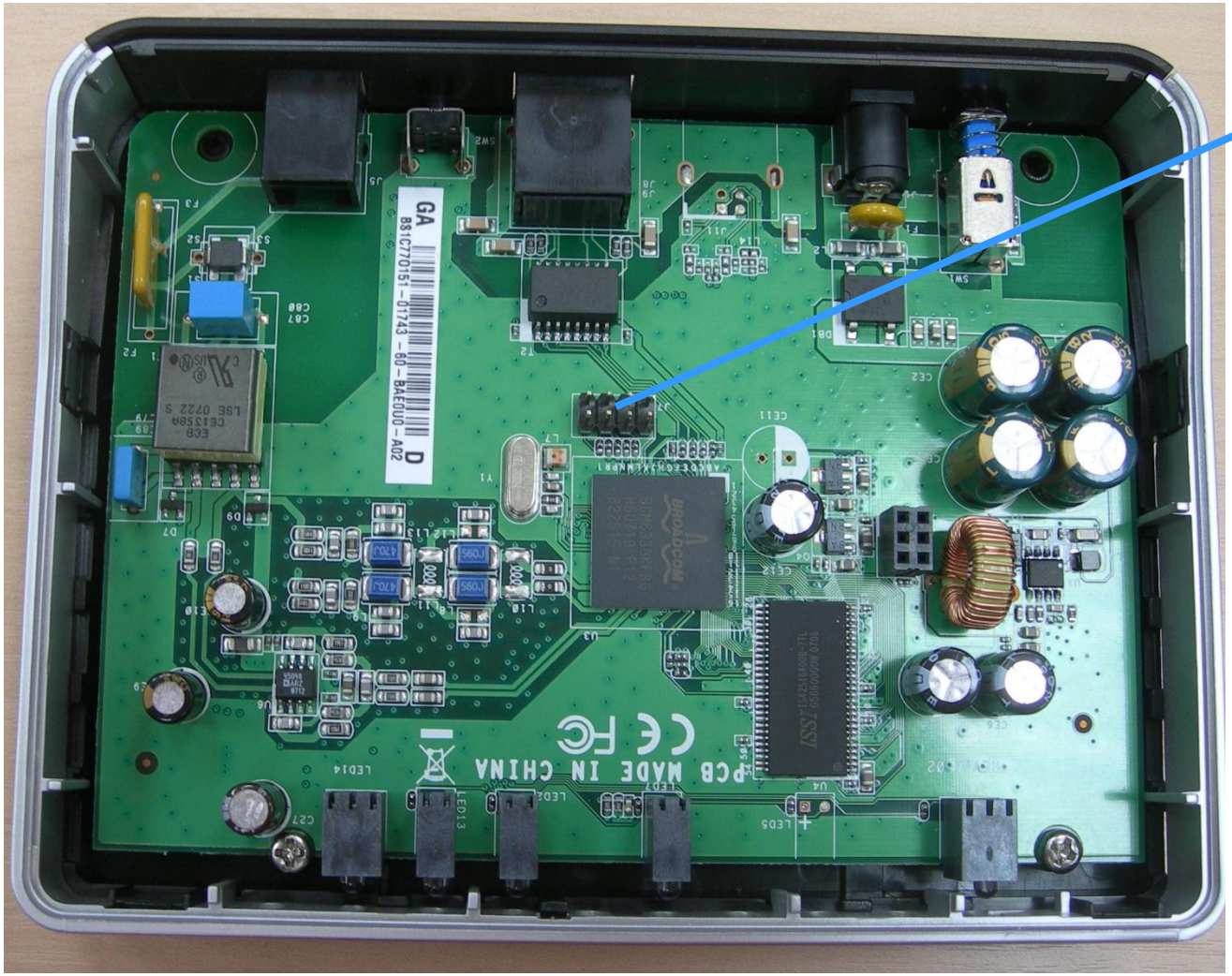


Микросхема флэш-памяти NAND 2 Мбайт (не показана, находится с обратной стороны платы)

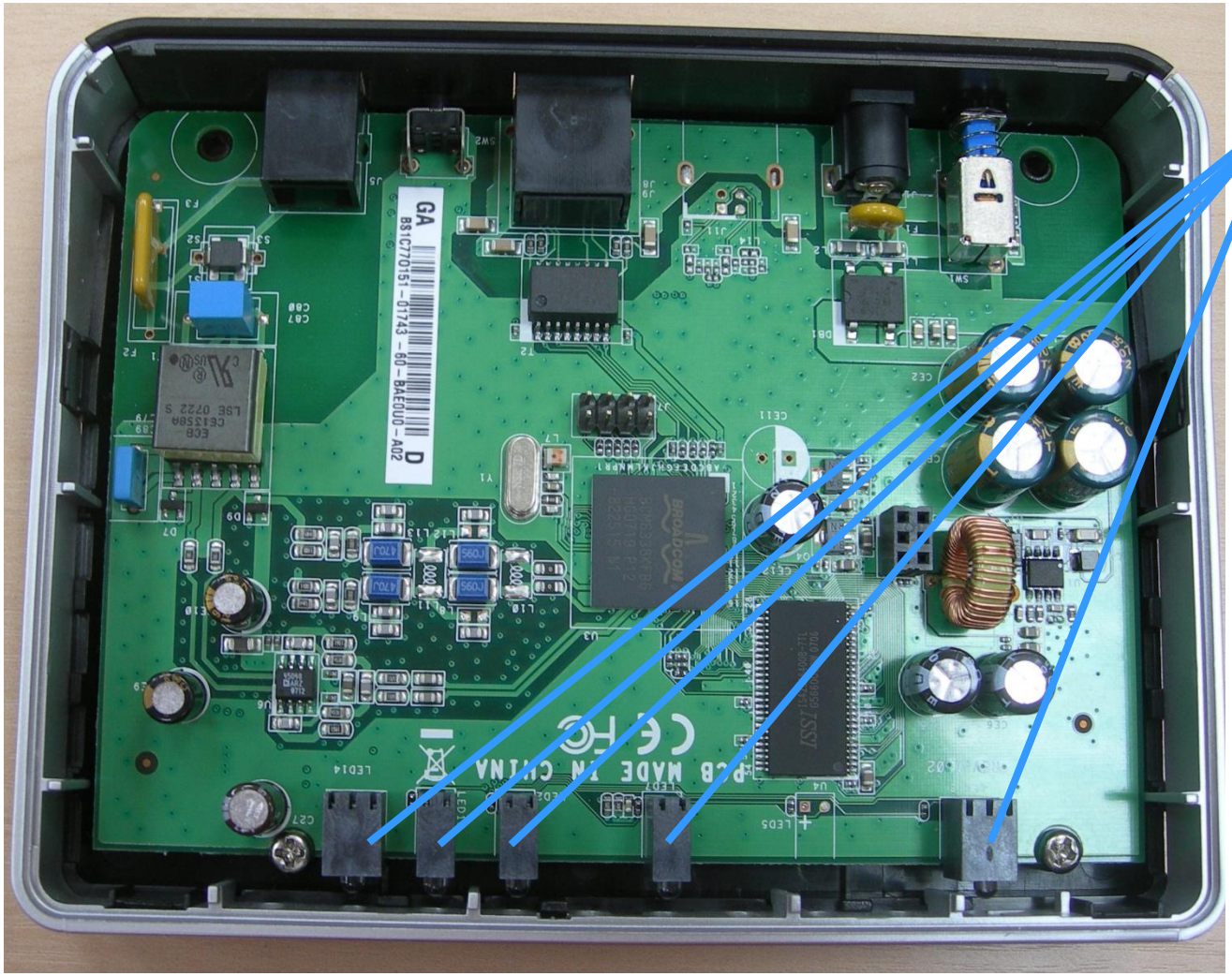
“Внутренности” роутера



“Внутренности” роутера

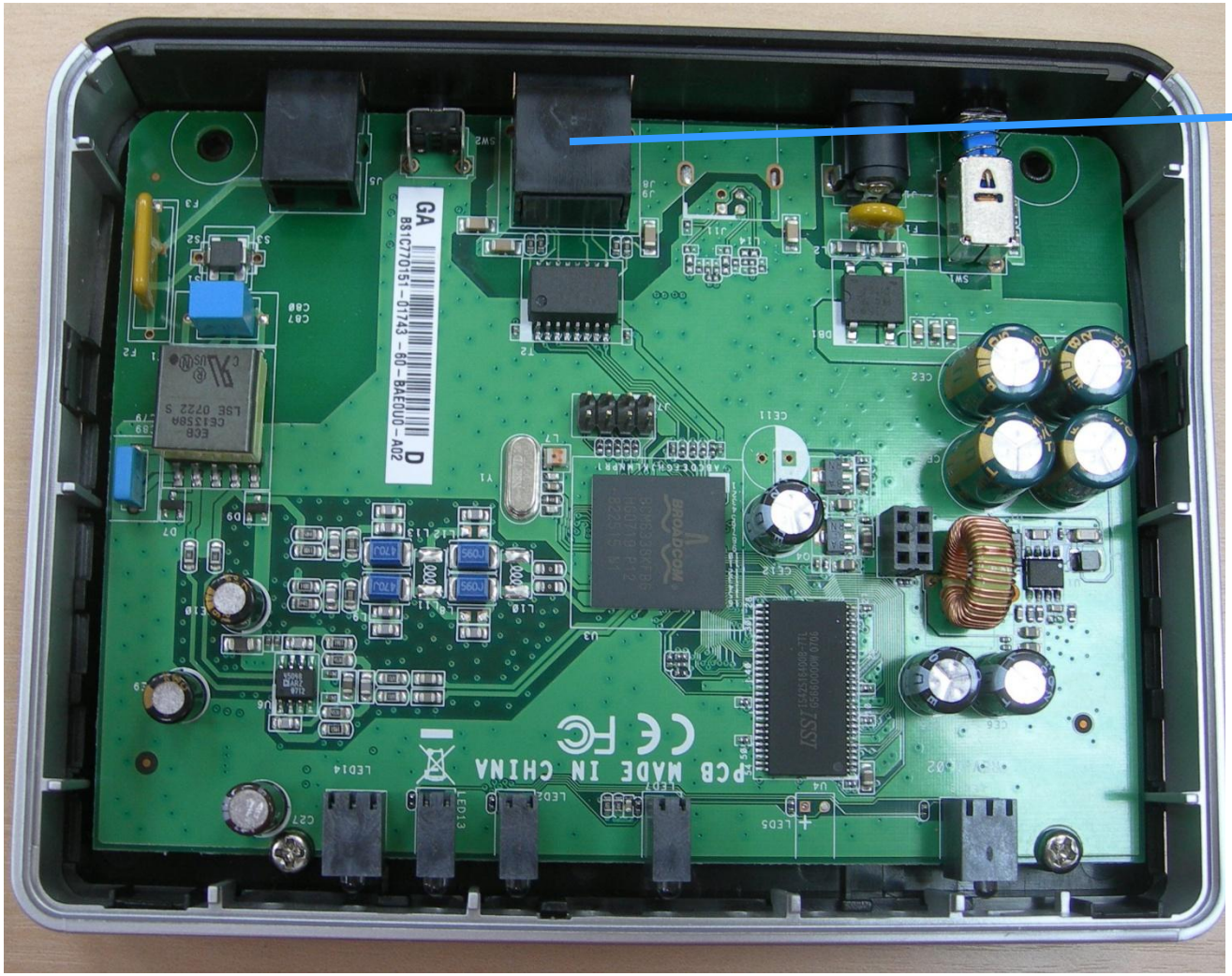


“Внутренности” роутера

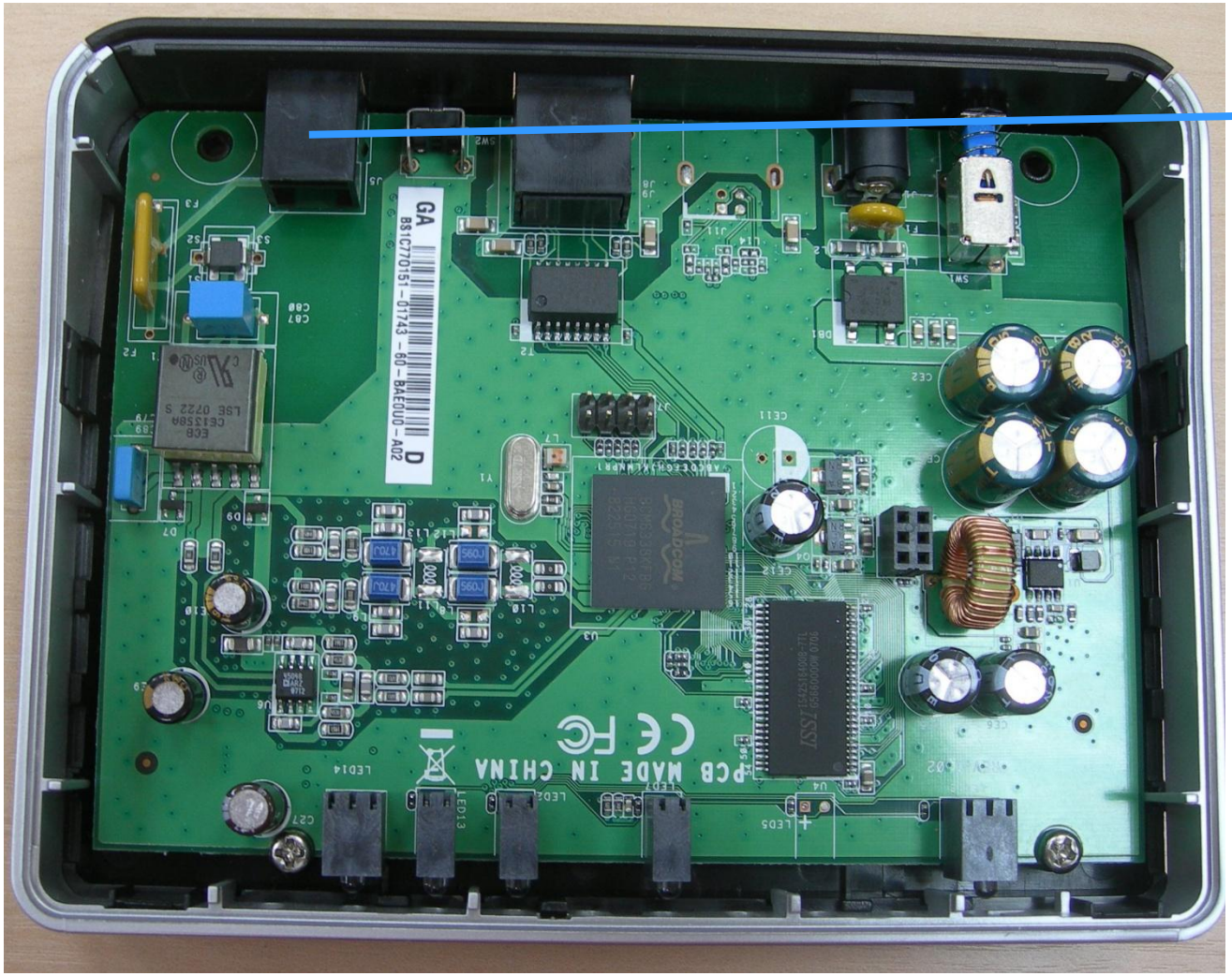


Светодиоды

“Внутренности” роутера



“Внутренности” роутера



Демонстрация работы с роутером через консоль

Сейчас мы подключимся к консольному порту роутера через USB-кабель и поработаем с ним…

Для работы используем программу **minicom**, которую запустим командой:

minicom –c on -b 115200 -D /dev/ttyUSB0

После подключения работаем с командным интерфейсом роутера (**BusyBox**) и с загрузчиком (**CFE**)…

Исходники прошивки для роутера (open source)

Исходный код прошивки роутера можно бесплатно скачать:

ftp://ftp.dlink.ru/pub/ADSL/GPL_source_code/RU_DSL-2500U/RU_DSL-2500U_3-06-04-3C_GPL.tar.gz

Архив включает 4 файла:

- **consumer_install**. Скрипт для установки пакета (Предназначен для запуска на Red Hat Linux),
- **README**. Файл Readme,
- **RU_DSL-2500U_3-06-04-3C_consumer.tar.gz**. Файлы исходного кода и бинарные файлы для сборки прошивки,
- **RU_DSL-2500U_3-06-04-3C_uclibc_crosstools_3.4.2_0.9.27.tar.gz**. Toolchain для кросс-компиляции.

Исходники прошивки для роутера (open source)

Файл **RU_DSL-2500U_3-06-04-3C_consumer.tar.gz** содержит несколько каталогов из которых наиболее важными являются :

- **bcmdrivers** – проприетарные драйверы Broadcom (не Open Source),
- **kernel** – как видно из названия, содержит код ядра Linux (версии 2.6.8.1) для MIPS с дополнениями Broadcom,
- **Userspace/broadcom** – проприетарные программы Broadcom, функционирующие в режиме пользователя. Представлены в виде двоичных файлов,
- **Userspace/opensource** – программы Open Source, функционирующие в режиме пользователя. Представлены в виде файлов исходных кодов.

Исходники прошивки для роутера (open source)

Компиляция прошивки выполняется просто:

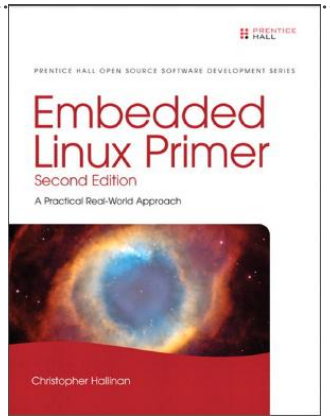
make PROFILE=RU_DSL-2500U

Но есть нюанс…

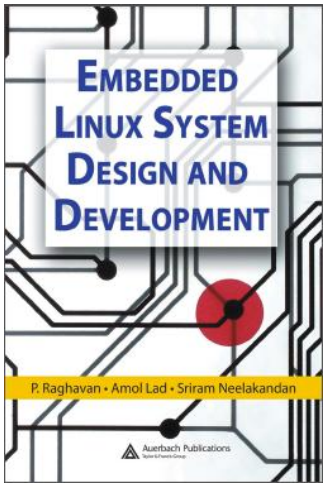
Файлы от 2007 года и для успешной сборки нужно поставить устаревшую версию 3 компилятора **gcc**.

Литература

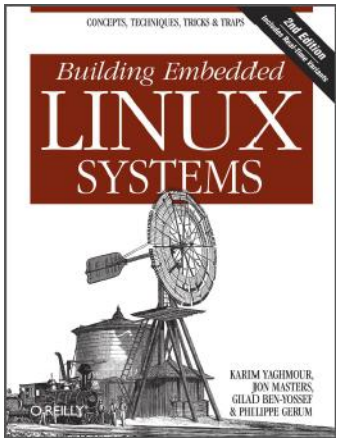
На русском языке литературы нет…



1. Christopher Hallinan. Embedded Linux Primer, Second Edition. A Practical, Real-World Approach. Prentice Hall, 2011. 616 p. ISBN 978-0-13-701783-6



2. P. Raghavan. Amol Lad. Sriram Neelakandan. Embedded Linux system design and development. Auerbach Publications. 2006. 400 p. ISBN 978-0-8493-4058-1



3. Yadhmour K., Masters J., Ben-Yossef G., Gerum P. Building Embedded Linux Systems – USA, OReily, 2008. p. 441. ISBN 978-0-596-52968-0.

4. **www.free-electrons.com**

В РГРТУ читаются курсы от компании D-Link по программированию встроенных систем на основе Linux.

Новый набор планируется в феврале 2013 г.

Контактный E-Mail: vshibanov@dlink.ru

Спасибо за внимание!
Вопросы…