Восстановление роутеров AP99 через JTAG

Проверенно на	Конфиг. файл[1]	U-boot[2]	EJTAG на плате
D-Link DIR-615 rev Ex[3]			Есть
TP-Link TL-WR741ND			Fix Me!
TP-Link TL-WR841NDv7			Fix Me!
TP-Link TL-MR3420/TL-MR3220			Модификация

Производитель	U-boot	art	fullflash
D-Link	–	–	fullflash [http://dioptimizer.narod.ru/files/dir-615e4/dir-615-dd-wrt.bin] (DIR-615revE3/E4), md5 [http://dioptimizer.narod.ru/files/dir-615e4/dir-615-dd-wrt.md5]
TP-Link	u-boot [http://dioptimizer.narod.ru/files/mr3220/backup_uboot.bin] (AP99_4M),md5 [http://dioptimizer.narod.ru/files/mr3220/backup_uboot.md5]	art [http://dioptimizer.narod.ru/files/mr3220/backup_art.bin] (AR9285)[4], md5 [http://dioptimizer.narod.ru/files/mr3220/backup_art.md5]	fullflash [http://dioptimizer.narod.ru/files/mr3220/fullflash.bin] (MR3220), md5 [http://dioptimizer.narod.ru/files/mr3220/fullflash.md5]

Если Вам удалось восстановить роутер через JTAG по этой инструкции, пожалуйста, отредактируйте верхнюю таблицу, чтобы можно было подтвердить описанный метод и на других устройствах. Ваш вклад важен в развитие проекта.

Примечание: Список роутеров на которых должен работать основной конфигурационный файл для OpenOCD, на самом деле, может быть намного больше (практически все процессоры AR724x). Однако для каждой платформы[5] необходим соответствующий, собственный загрузчик, измененный таким образом, чтобы можно было его загружать из SDRAM памяти - без потери функциональности.

Аппаратная часть

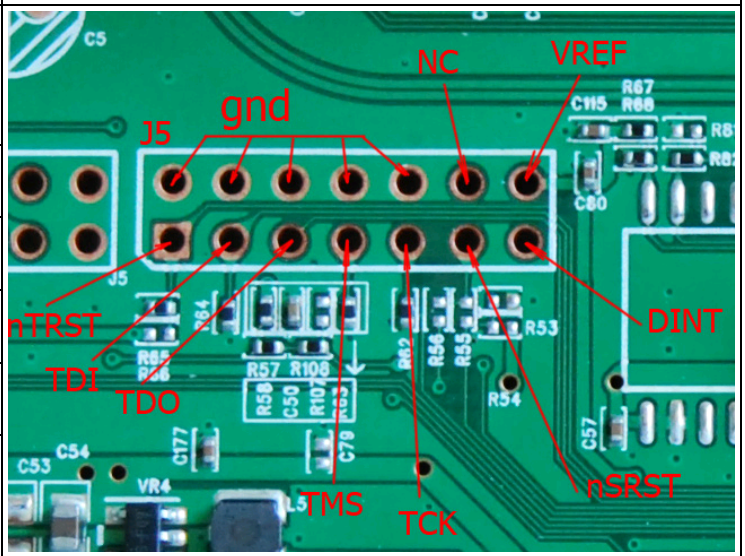
JTAG на процессоре

→ port.jtag Номера пинов EJTAG на процессоре AR724**0** идентичны процессору AR724**1** учитывая такую же архитектуру, скорее всего, идентичны и процессору AR724**2**( ). Единственное отличие на устройствах с процессорами AR724**0** - где 75 пин используется как CS# , а **80** пин как GPIO пин или если в устройстве предусмотрен EJTAG разъем, то **80** пин используется как **nTRST** пин.

В большинстве устройств с процессорами AR724**1**/AR724**2** - разъем EJTAG (как отдельный разъем) не предусмотрен производителем. Это связано с тем, что некоторые JTAG пины используются системой как GPIO пины и настроены в режиме output-mode, плюс ко всему, **80** пин на процессоре используется как **CS#** для

микросхемы флеш-памяти.

Распиновка EJTAG v3.1 [http://www.linux-mips.org/wiki/JTAG] интерфейса на AR724x, выглядит следующим образом:

JTAG Common Name	AR724x Pin	Разъем EJTAG на DIR-615 Ex
nTRST *(только AR7240)*[6]	80	
TDI	81	
TDO	82	
TMS	84	
TCK	85	
~~RST~~ *(не используем)*	93	

Если отсутствует маркировка отсчета ножек процессора на плате, то можно воспользоваться такой ориентировкой на чипе с корпусом LQFP128:

```
          128                 97
            |                  |
       Lock-+------------------+-96
            |                  |
            |        /\        |
DDR Chip -  |       /--\       | - SPI (or NAND) Flash Chip
            |      /    \      |
            |     AR724x       |
            |                  |
         32-+------------------+-65
            |                  |
           33                  64
                     |
                  Ethernet
```

Подключение к JTAG

Если у Вас действительно поврежден или стерт загрузчик на устройствах с процессорами AR7241/AR7242 - Вам **не нужно** использовать **nTRST** пин на JTAG интерфейсе устройства.

Примечание: Отладка и тестирование (для написания этой инструкции) производилось с исправным загрузчиком, поэтому необходимо было использовать линию **nTRST**. Если загрузчик поврежден – то на пине **CS#** микросхемы флеш-памяти будет 3.3V – что в свою очередь, равносильно Лог. "1" для **nTRST** линии. Скорее всего, это дополнительная причина размещения **nTRST** на **CS#** пине микросхемы флеш-памяти, в процессорах AR7241/AR7242.

RST пин - AR7240/ AR7241/AR7242 полностью сбрасывает процессор, т.е. в том числе записанные регистры инициализации процессора, это состояние равносильно отключению питания от устройства – поэтому, этот пин мы тоже **не будем** использовать.

→ port.jtag.cables Подключение, как правило, производится через специальный

JTAG адаптер. Это могут быть простые и дешевые USB-JTAG адаптеры, так и старые адаптеры на основе параллельного порта (LPT) для компьютера - как правило, такие адаптеры самодельные.

В примере, представленном в этом разделе, подключение производилось через старый и медленный JTAG адаптер для LPT порта – Wiggler(без буфера) [http://ciclamab.altervista.org/hard_corpo_jtag.htm]. Конструкция этого адаптера чрезвычайно проста.

Программная часть

Программная часть содержит список программ, которые понадобятся в процессе отладки и восстановления через интерфейс JTAG. Также, эта часть содержит список используемых команд OpenOCD и конфигурационный файл для процессоров AR724x.

Программы для работы с JTAG

- **OpenOCD**[7] - Как собрать OpenOCD для Windows в среде Linux OS [http://dangerousprototypes.com/docs/Compile_OpenOCD_for_Windows], Как собрать OpenOCD для Windows в среде Windows OS [http://kazus.ru/forums/attachment.php?attachmentid=7775&d=1273832853](на русском).
  Также можно воспользоваться последней стабильной [http://www.freddiechopin.info/en/download/category/4-openocd] версией OpenOCD, собранной для Windows (программа включает в себя практически все JTAG адаптеры).
- **PuTTY** – `telnet` консоль. Эта консоль также используется для подключения к роутеру через последовательный порт.
- **ar724x.cfg** - конфиг. файл для программы OpenOCD и Вашего устройства (конфиг. файл, необходимо скопировать в папку программы OpenOCD как target/ar724x.cfg).
- **8Muboot_RAM_version.bin [http://dioptimizer.narod.ru/files/8Muboot_RAM_version.bin]**,md5 [http://dioptimizer.narod.ru/files/8Muboot_RAM_version.md5] - загрузчик, который можно запустить в SDRAM памяти через JTAG (спасибо участнику форума OpenWRT с ником **tthrx**)
- **backup.bin** - условный загрузчик или другой фрагмент данных флеш-памяти, который необходимо восстановить.

  Примечание: Учтите, что в загрузчике хранится информация о MAC адресе[8] и PIN коде[9] устройства. Следует также знать, что используемый **art** раздел с EEPROM информацией для чипа беспроводной связи, должен соответствовать чипу беспроводной связи восстанавливаемого устройства.
  Например: **art** раздел роутера MR3420(WiFi Chip: AR9287) не подойдет к роутеру MR3220(WiFi Chip: AR9285) и наоборот.

Используемые команды OpenOCD

```
reset
```

*В примере представленном в этом разделе, команда используется как - определение идентификатора и состояние устройства, не более. Обычно, при выполнении этой команды задействуется пин **nSRST**, но в нашем случае **RST** пин - не одно и то же.*

```
halt
```

Переводит процессор в режим отладки (прием команд).

```
ar724x.cpu invoke-event <событие>
```

Форсированное применение события (в этом примере используется событие `reset-halt-post`). После выполнения этой команды, будет выполнен скрипт для этого события.
По своей сути, эта команда борьба с ошибкой в программе OpenOCD - где нет возможности использовать весь скрипт для события `reset-init`.

```
reset init
```

Инициализация основного скрипта, заключенного в фигурные скобки, который находится в конфиг. файле (отправка команд в процессор).

```
dump_image <имя файла> <стартовый адрес в области памяти или флешки> <размер>
```

*Эта команда сохраняет дамп **из** памяти/флешки устройства в файл. Команда может быть выполнена до инициализации процессора и памяти устройства. Для чтения флеш-памяти, используйте адрес `0x9f000000`*

```
load_image <имя файла> <адрес в области только памяти> <формат файла>
```

*Эта команда загружает файл **в** память устройства. Команда должна выполнятся **после** инициализации процессора и памяти устройства.*

```
resume <адрес в области памяти или флешки>
```

*Эта команда запускает загрузчик, аналог **go** в uboot'е*

ar724x.cfg

```
# Atheros AR724x MIPS 24Kc SoC.
# tested on AP99 refererence board
#
# this settings are taken from source of u-boot for this board
# (for PLL) file:       u-boot/board/ar7240/common/lowlevel_init.S
# (for DDR) file:       u-boot/cpu/mips/ar7240/meminit.c
#      with file:       u-boot/include/configs/ap99.h
# to execute first part of initialization script
# use this command:     ar724x.cpu invoke-event reset-halt-post

adapter_nsrst_delay 100
jtag_ntrst_delay 100

reset_config trst_only separate         ;# or use only "reset_config none"

set CHIPNAME ar724x

jtag newtap $CHIPNAME cpu -irlen 5 -ircapture 0x1 -irmask 0x1f -expected-id 1

set TARGETNAME $CHIPNAME.cpu
```

```
target create $TARGETNAME mips_m4k -endian big -chain-position $TARGETNAME


$TARGETNAME configure -event reset-halt-post {
        #reset Watchdog Timer (when timer ends - resets SoC - then again halted)
        mww 0xb806000c 0x400000         ;# rst watchdog timer (delay ~250ms for JTAG adapter
        mww 0xb8060008 3                ;# rst watchdog timer control (set control bit in thi

        sleep 250                       ;# wait resetting SoC (delay for JTAG adapter with fa
        poll                            ;# echo target state cpu (must be: running)
        halt

        #setup PLL to lowest(default) common denominator 400/400/200 setting
        mww 0xb8050000 0x00090828       ;# clr pll mask (rst:02090828)
        mww 0xb8050000 0x00050828       ;# CPU:400 DDR:400 AHB:200
        mww 0xb8050000 0x00040828       ;# clr pll bypass

        #next command will reset for PLL changes to take effect
        mww 0xb8050008 2                ;# set reset_switch
        mww 0xb8050008 3                ;# set clock_switch (resets SoC)
}

$TARGETNAME configure -event reset-init {
        #complete pll initialization
        mww 0xb8050008 0                ;# set reset_switch bit & clock_switch bit

        # Setup DDR config and flash mapping
        mww 0xb8000000 0xc7bc8cd0       ;# DDR cfg cdl val (rst:77be8cd0)
        mww 0xb8000004 0x9dd0e6a8       ;# DDR cfg2 cdl val (rst:99d10628)

        mww 0xb8000010 8                ;# force precharge all banks
        mww 0xb8000008 0x133            ;# DDR mode value init
        mww 0xb8000010 1                ;# force EMRS update cycle
        mww 0xb800000c 0                ;# clr ext. mode register

        mww 0xb8000010 2                ;# force auto refresh all banks
        mww 0xb8000010 8                ;# force precharge all banks
        mww 0xb8000008 0x33             ;# set DDR mode value CAS=3
        mww 0xb8000010 1                ;# force EMRS update cycle
        mww 0xb8000014 0x4f10           ;# DDR refresh value
        mww 0xb8000018 0xff             ;# DDR Read Data This Cycle value (16bit: 0xffff)
        mww 0xb800001c 2                ;# delay added to the DQS0 line (normal = 7)
        mww 0xb8000020 2                ;# delay added to the DQS1 line (normal = 7)
        mww 0xb8000024 0
        mww 0xb8000028 0
}

# setup working area somewhere in RAM
$TARGETNAME configure -work-area-phys 0xa0600000 -work-area-size 0x20000

# serial SPI capable flash
# flash bank <driver> <base> <size> <chip_width> <bus_width>
```

Пример восстановления

Восстановление условного загрузчика **u-boot** и **art** раздела, на роутере MR3220(4M) в операционной системе Windows. При желании, таким же образом, дополнительно, можно прошить **firmware** образ или целиком всю флеш-память - **fullflash**.

- Распаковываем собранную версию "openocd-0.5.0.zip" в удобную для нас папку.
- Перемещаем или копируем из папки `bin` (папка расположена в корневой директории OpenOCD), все содержимое в корневую директорию OpenOCD.
- Создаем файл `ar724x.bat` (можно использовать любое другое имя) в той же корневой директории OpenOCD со следующим содержимым:

```
openocd-0.5.0.exe -f interface/parport.cfg -f target/ar724x.cfg
pause
```

*Если Вы используете другой JTAG адаптер, то название **parport.cfg** должно быть*

изменено на соответствующее название конфигурационного файла для Вашего JTAG адаптера.

- Подключаем JTAG к компьютеру и **отключенному** роутеру.
- Подключаем UART к компьютеру и **отключенному** роутеру, также следует открыть **PuTTY** (программа должна быть настроена на параллельный порт Вашего компьютера с соответствующими настройками для Вашего роутера).

Работа с программой OpenOCD

- Запускаем **ar724x.bat**, практически сразу можно включить роутер, цель обнаружить идентификатор 0x00000001 (стандартный идентификатор процессоров Atheros):

```
D:\Free\OpenOCD\0.5.0>openocd-0.5.0.exe -f interface/parport.cfg -f target/ar724x.cfg
Open On-Chip Debugger 0.5.0 (2012-04-06-14:30)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
Warn : Adapter driver 'parport' did not declare which transports it allows; assuming legacy JTAG-o
Info : only one transport option; autoselect 'jtag'
parport port = 0x378
6000 kHz
adapter_nsrst_delay: 100
jtag_ntrst_delay: 100
none separate
131072
Info : clock speed 500 kHz
Info : JTAG tap: ar724x.cpu tap/device found: 0x00000001 (mfg: 0x000, part: 0x0000, ver: 0x0)
Info : accepting 'telnet' connection from 4444
```

Если не получилось определить идентификатор сразу, попробуйте перейти к следующему пункту и набрать в консоли "`reset`". Если все равно программа не определяет идентификатор процессора, то нужно проверить подключение кабеля JTAG на возможные ошибки, также причина неполадок может заключаться в длине используемого кабеля.

- Если все прошло успешно, необходимо запустить консоль `telnet` или еще одно окно **PuTTY** используя адрес `127.0.0.1:4444`, после подключения в консоли должна отобразится строка ввода:

```
Open On-Chip Debugger
>
```

  - Дальше необходимо ввести команды:

```
> reset
JTAG tap: ar724x.cpu tap/device found: 0x00000001 (mfg: 0x000, part: 0x0000, ver: 0x0)
>
```

*Эта команда не должна влиять на состояние процессора, т.к. **RST** мы не используем. Однако команда очередной раз определяет идентификатор и состояние процессора.*

```
> halt
target state: halted
target halted in MIPS32 mode due to debug-request, pc: 0xbfc03860
>
```

Эта команда переводит процессор из состояния "running", в состояние "halted" - в этом состоянии, процессор принимает команды от оператора.

```
> ar724x.cpu invoke-event reset-halt-post
background polling: on
TAP: ar724x.cpu (enabled)
target state: running
target state: halted
target halted in MIPS32 mode due to debug-request, pc: 0xbfc03860
Error writing unexpected address 0x00000000
Error writing unexpected address 0x00000000
in procedure 'mww'

in procedure 'mww'
>
```

*Инициализация скрипта в конфигурационном файле **ar724x.cfg**. В данном случае, скрипт послал группу команд заключенных в фигурные скобки для первого события. В процессе отправки команд, процессор был программно перезагружен и обратно переведен в режим "halted" и еще раз перезагружен.*

Примечание: Ошибка вида "in procedure 'mww'" - баг программы OpenOCD в момент перезагрузки процессора.
Однако не исключено, что эта ошибка, может и не возникнуть, если использовать более скоростные JTAG адаптеры или другую версию OpenOCD. В любом случае, ошибка не должна повлиять на следующие действия.

```
> halt
target state: halted
target halted in MIPS32 mode due to debug-request, pc: 0xbfc03860
>
```

Еще раз переводим процессор из состояния "running", в состояние "halted" - в этом состоянии, процессор принимает команды от оператора.

```
> reset init
JTAG tap: ar724x.cpu tap/device found: 0x00000001 (mfg: 0x000, part: 0x0000, ver: 0x0)
target state: halted
target halted in MIPS32 mode due to debug-request, pc: 0xbffd0ac0
>
```

*Инициализация основного скрипта в конфигурационном файле **ar724x.cfg**. В*

данном случае, скрипт послал группу команд заключенных в фигурные скобки для второго события.

```
> load_image backup_uboot.bin 0x81000000 bin
131072 bytes written at address 0x81000000
downloaded 131072 bytes in 12.250000s (10.449 KiB/s)
> load_image backup_art.bin 0x81020000 bin
65536 bytes written at address 0x81020000
downloaded 65536 bytes in 5.440000s (11.765 KiB/s)
> load_image 8Muboot_RAM_version.bin 0x80000000 bin
262144 bytes written at address 0x80000000
downloaded 262144 bytes in 21.639999s (11.830 KiB/s)
>
```

*Заранее загружаем куда-нибудь в память, загрузчик **u-boot** и **art** раздел – позже, эти данные необходимо будет скопировать на флеш-память.*
*Основное, что мы делаем - мы загружаем **8Muboot_RAM_version.bin** загрузчик в область памяти 0x80000000 – загрузчик был скомпилирован с привязкой к этому адресу.*

Примечание: Возможно обойтись только загрузчиком **8Muboot_RAM_version.bin** (в области памяти 0x80000000) - используя этот загрузчик, можно прошить флеш-память с помощью tftp метода.
Следует знать особенности tftp, в этом загрузчике:

- Для доступа к загрузчику, в консоли необходимо успеть ввести **tt**
- IP адрес компьютера, при этом, должен быть - 192.168.1.23 (или воспользуйтесь командой `setenv`, чтобы изменить значение `serverip`).

- Следующая команда, запустит загрузчик из области SDRAM памяти, одновременно с этим процессом, должна быть запущена еще одна консоль **PuTTY**, настроенная на последовательный порт:

```
resume 0x80000000
```

*Загрузчик попытается загрузить **firmware** из флеш-памяти, чтобы не допустить провала (и как следствие, перезагрузки роутера) - когда в окне появится надпись "Autoboot in .." нужно быстро написать в консоли **tt**.*

Работа в загрузчике U-boot

Заключительный этап восстановления происходит через загрузчик U-Boot.
Необходимо стереть рабочую область флеш-памяти, а потом скопировать ранее записанные данные на флеш-память устройства.
Это можно сделать следующим образом:

```
AR7241# erase 0x9f000000 +0x20000
Erase Flash from 0x9f000000 to 0x9f01ffff in Bank # 1
First 0x0 last 0x1 sector size 0x10000
Erased 2 sectors
AR7241#
```

*Стираем область во флеш-памяти для загрузчика **u-boot**.*

```
AR7241# cp 0x81000000 0x9f000000 0x20000
Copy to Flash... write addr: 9f000000
done
AR7241#
```

*Копируем из области SDRAM памяти, ранее записанный **u-boot** загрузчик, во флеш-память.*

```
AR7241# erase 0x9f3F0000 +0x10000
Erase Flash from 0x9f3f0000 to 0x9f3fffff in Bank # 1
First 0x3f last 0x3f sector size 0x10000
Erased 1 sectors
AR7241#
```

*Стираем область во флеш-памяти для **art** раздела (флеш-память **4M**).*

```
AR7241# cp 0x81020000 0x9f3F0000 0x10000
Copy to Flash... write addr: 9f3f0000
done
AR7241#
```

*Копируем из области SDRAM памяти, ранее записанный **art** раздел, во флеш-память.*

Дополнительная информация

Есть возможность модифицировать исходный код U-Boot (SDRAM) загрузчика - для автоматической прошивки **необходимого** раздела (в зависимости от имени или размера загруженного файла через tftp), при нажатии на определенную кнопку.

Примечание: По умолчанию, представленный в этом разделе загрузчик **8Muboot_RAM_version.bin** (от участника форума OpenWRT - **tthrx**), в момент загрузки из памяти, при определении зажатой кнопки QSS – автоматически пытается загрузить через tftp - **firmware**, а в последствии прошить флеш-память.

Также на основе этого загрузчика, теоретически возможно создать – отладочный загрузчик U-Boot (SDRAM) для мульти-платформ на основе процессоров AR724x.

Загрузчик для SDRAM и исходные коды можно найти в этом разделе форума [https://forum.openwrt.org/viewtopic.php?id=33205].

Расширенные логи работы с программой и используемый материал можно найти на форуме [https://forum.openwrt.org/viewtopic.php?id=34993].

[1)] Конфигурационный файл инициализации процессора и DDR памяти представленный в этом разделе для программы OpenOCD

[2)] Загрузчик U-Boot (SDRAM) представленный в этом разделе благодаря **tthrx** - участнику форума OpenWRT

[3)] Ex - здесь подразумеваются все роутеры DIR-615 с версией E1- E4

[4)] Раздел содержит EEPROM данные для калибровки Wi-FI чипа, эти данные уникальны для каждого устройства.

[5)] Устройство с конкретной архитектурой, используемой NOR/NAND/DDR памятью, кол-вом индикаторов, кнопок, привязанность их к номерам GPIO, прочее. Например, здесь рассматривается платформа – AP99

[6)] На процессорах AR7241/AR7242 - пин nTRST используется только в случае, если загрузчик на флеш-памяти **не** поврежден.

[7)] При необходимости можно заменить на **OCD Commander**, но следует помнить, что там другой формат отправки регистров в процессор, поэтому конфиг. файл необходимо переделать под этот формат.

[8] TP-Link: U-boot MAC offset 0x01fc00 (value in HEX format)
[9] TP-Link: U-boot PIN offset 0x01fe00 (value in Dec format)

Back to top

ru/toh/tp-link/tl-mr3420/debrick.using.jtag.txt · Last modified: 2012/05/16 09:42 by dioptimizer