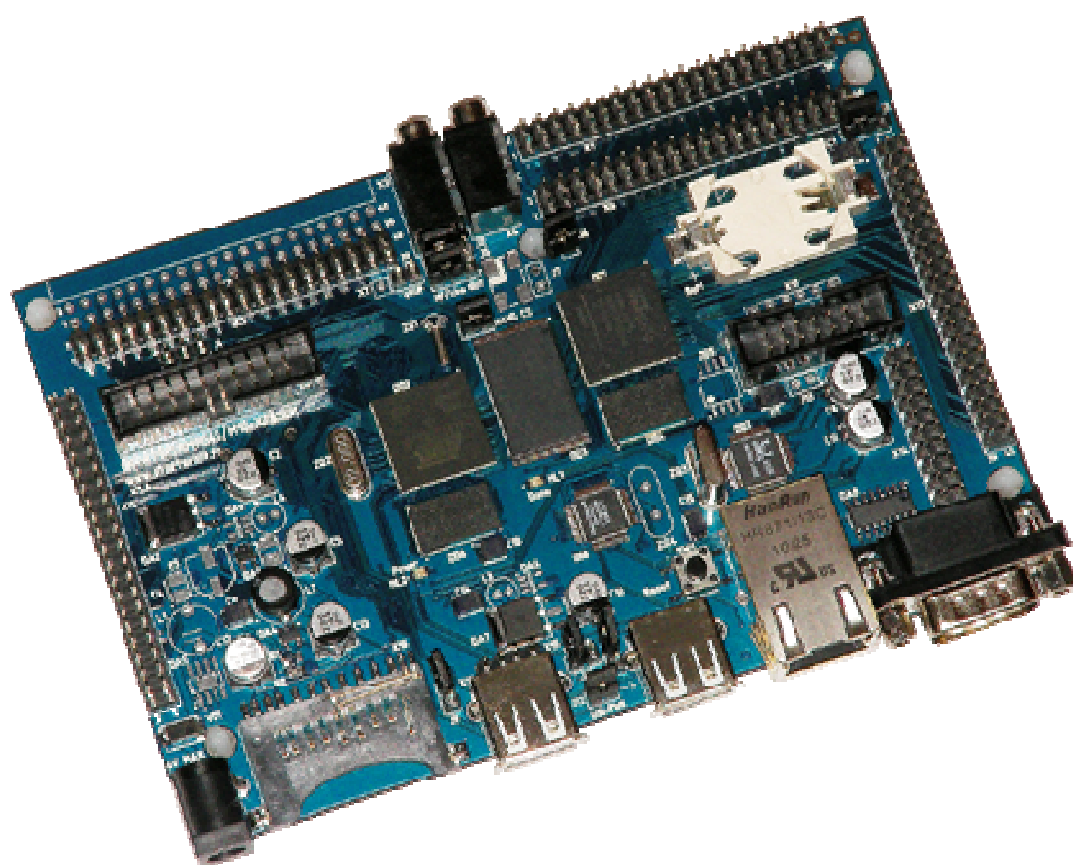


Отладочная плата SK-AT91SAM9G45-XC6SLX

Инструкция пользователя



SK-AT91SAM9G45-XC6SLX:

Atmel AT91SAM9G45 (ARM9 400МГц)
DDR2 128Мбайт (64М подключено к ARM, 64М подключено к FPGA)
Xilinx FPGA Spartan-6 XC6SLX16
NAND Flash 256Мбайт (возможна загрузка)
100/10М Ethernet
AC97 аудио CODEC
uSD держатель (возможна загрузка)
SD/MMC держатель
USB Host/Device
USB Host
RS232
RTC (часы реального времени)
JTAG разъемы
Разъемы расширения
Система питания

Возможность прямого подключения:

SK-MI0430FT-Plug или аналог – плата расширения LCD TFT 4,3” панелей
SK-ATM0700D4-Plug или аналог – плата расширения LCD TFT 7” панелей
SK-HDMI-Plug – плата расширения HDMI выхода
SK-SIMCOM-Plug – плата расширения GSM/GPS/3G модулей
SK-VideoADC-Plug – плата расширения видеозахвата

Комплект поставки: отладочная плата SK-AT91SAM9G45-XC6SLX, RS232 кабель, ссылка на программное обеспечение, демонстрационные проекты и виртуальную машину VMware

Общие характеристики

- Напряжение питания 5В (максимум 6В)
- Рекомендуемый выходной ток БП - 2А
- Габариты 138x92мм

1. Назначение джамперов

1-ый вывод перемычек и переключающих перемычек помечен квадратной контактной площадкой.

- J1 позволяет подключить внешнюю кнопку управления питанием (сигнал WKUP)
- J2, J5 позволяют коммутировать на аудио разъем X3 выход «на наушники» или «линейный выход», положение 1-2 – «выход на наушники»
- J3 управляет напряжением питания Bank0 FPGA (сигналы разъема X13), по умолчанию, в положении 1-2, в этом случае IO Bank0 будут оперировать уровнем 3,3В.
- J4 может использоваться при подключении SK-VideoADC-Plug для выбора синхросигнала вертикальной развертки (FIELD или VS), актуально для чересстрочных источников видеосигнала
- J6 позволяет управлять сигналом FPGA – PROGRAM
- J7 определяет источник загрузки FPGA, замкнут – SPI DataFlash, разомкнут - ARM
- J8 позволяет «исключить» NAND Flash из системы, актуально при программировании через SAM-BA
- J9 позволяет подавать питающее напряжение на USB host (X19) минуя управляющие транзисторные каскады
- J10 USB-ID – позволяет конфигурировать один из USB портов как device/host, актуально для OTG режима работы порта
- J11 позволяет подавать питающее напряжение на USB host (X20) минуя управляющие транзисторные каскады
- J12 управляет подачей питающего напряжения к X20, **ВНИМАНИЕ!!!** В режиме работы порта как Device (например, при программировании NAND flash через SAM-BA) должен быть разомкнут
- J13 позволяет отключить CS сигнал конфигурационной SPI DataFlash

По умолчанию замкнуты перемычки: J8, J12, J3 положение 1-2, J4 положение 1-2, J2 и J5 положение 1-2

2. Начало работы

Подключите RS232 кабель, идущий в комплекте, к COM порту PC (или USB-COM преобразователю), настройте терминальную программу на используемый COM порт с параметрами 115200 Бит, без управления потоком.

Подключите сетевой (Ethernet) кабель, настройте IP адрес сетевой карты PC в диапазоне 192.168.0.XXX (кроме 192.168.0.136 – это адрес самой платы).

При необходимости, подключите SK-MI0430FT-Plug или аналог.

Подключите питание (питающее напряжение – центральная жила разъема), в терминальной программе появятся следующие сообщения:

```
-- AT91bootstrap Project 3.0 -
-- AT91bootstrap Project 3.0 --
-- AT91SAM9M10-EK
-- Compiled: Dec 17 2010 17:24:43 --
-I- Setting: MCK = 133MHz
-I- I cache enabled.
-I- Init DDRAM
-I- Init NAND Flash
-I- Nandflash ID is 0x9510DAEC
-I- Nandflash driver initialized
-I- Size of the whole device in bytes : 0x10000000
-I- Size in bytes of one single block of a device : 0x20000
-I- Number of blocks in the entire device : 0x800
-I- Size of the data area of a page in bytes : 0x800
-I- Number of pages in the entire device : 0x40
-I- Bus width : 8
-I- Copy "" (262144 bytes) from NAND 0x00020000 to 0x73f00000
-I- Jump to 0x73f00000
```

U-Boot 2010.09 (Nov 19 2010 - 02:07:08)

```
DRAM: 64 MiB
## Unknown FLASH on Bank 1 - Size = 0x00000000 = 0 MB
Flash: 0 Bytes
NAND: 256 MiB
In: serial
Out: serial
Err: serial
Net: macb0
Hit any key to stop autoboot: 0
```

```
NAND read: device 0 offset 0x80000, size 0x790000
7929856 bytes read: OK
## Booting kernel from Legacy Image at 70400000 ...
Image Name: Linux Kernel Image
Image Type: ARM Linux Kernel Image (gzip compressed)
Data Size: 2227641 Bytes = 2.1 MiB
Load Address: 70008000
Entry Point: 70008000
Verifying Checksum ... OK
Uncompressing Kernel Image ... OK
```

Starting kernel ...

```
Linux version 2.6.36-rc6 (user@debian) (gcc version 4.2.0 20070413 (prerelease)
(CodeSourcery Sourcery G++ Lite 2007q1-10)) #329 Fri Nov 19 08:23:11 EST 2010
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: SK-AT91SAM9G45/M10-XC6SLX
Ignoring unrecognised tag 0x54410009
Memory policy: ECC disabled, Data cache writeback
Clocks: CPU 400 MHz, master 133 MHz, main 12.000 MHz
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
Kernel command line: console=ttyS0,115200 ubi.mtd=1 root=ubi0:nandfs rw rootfstype=ubifs
PID hash table entries: 256 (order: -2, 1024 bytes)
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 64MB = 64MB total
Memory: 60448k/60448k available, 5088k reserved, 0K highmem
Virtual kernel memory layout:
vector : 0xffff0000 - 0xffff1000 ( 4 kB)
fixmap : 0xffff0000 - 0xffffe000 ( 896 kB)
DMA : 0xffa00000 - 0xffe00000 ( 4 MB)
vmalloc : 0xc4800000 - 0xfe000000 ( 934 MB)
lowmem : 0xc0000000 - 0xc4000000 ( 64 MB)
modules : 0xbf000000 - 0xc0000000 ( 16 MB)
 .init : 0xc0008000 - 0xc0029000 ( 132 kB)
 .text : 0xc0029000 - 0xc040a000 (3972 kB)
 .data : 0xc0426000 - 0xc044c360 ( 153 kB)
Hierarchical RCU implementation.
Verbose stalled-CPU detection is disabled.
NR_IRQS:192
AT91: 160 gpio irqs in 5 banks
Console: colour dummy device 80x30
console [ttyS0] enabled
Calibrating delay loop... 199.47 BogoMIPS (lpj=997376)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
SCSI subsystem initialized
```

```

usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
i2c-gpio i2c-gpio.0: using pins 52 (SDA) and 53 (SCL)
Advanced Linux Sound Architecture Driver Version 1.0.23.
cfg80211: Calling CRDA to update world regulatory domain
Switching to clocksource pit
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
msgmni has been set to 118
io scheduler noop registered (default)
atmel_lcdfb atmel_lcdfb.0: backlight control is not available
atmel_lcdfb atmel_lcdfb.0: 255KiB frame buffer at 73900000 (mapped at ffa00000)
Console: switching to colour frame buffer device 60x34
atmel_lcdfb atmel_lcdfb.0: fb0: Atmel LCDC at 0x00500000 (mapped at c4814000), irq 23
atmel_usart.0: ttyS0 at MMIO 0xfeff0000 (irq = 1) is a ATMEL_SERIAL
atmel_usart.1: ttyS1 at MMIO 0xffff8c000 (irq = 7) is a ATMEL_SERIAL
atmel_usart.2: ttyS2 at MMIO 0xffff90000 (irq = 8) is a ATMEL_SERIAL
brd: module loaded
loop: module loaded
NAND device: Manufacturer ID: 0xec, Chip ID: 0xda (Samsung NAND 256MiB 3,3V 8-bit)
Scanning device for bad blocks
Bad eraseblock 1800 at 0x00000e100000
Creating 2 MTD partitions on "atmel_nand":
0x0000000000000-0x0000010000000 : "Boot partition"
0x0000010000000-0x0000100000000 : "FS partition"
UBI: attaching mtd1 to ubi0
UBI: physical eraseblock size: 131072 bytes (128 KiB)
UBI: logical eraseblock size: 129024 bytes
UBI: smallest flash I/O unit: 2048
UBI: sub-page size: 512
UBI: VID header offset: 512 (aligned 512)
UBI: data offset: 2048
UBI: max. sequence number: 379
UBI: attached mtd1 to ubi0
UBI: MTD device name: "FS partition"
UBI: MTD device size: 240 MiB
UBI: number of good PEBs: 1919
UBI: number of bad PEBs: 1
UBI: max. allowed volumes: 128
UBI: wear-leveling threshold: 4096
UBI: number of internal volumes: 1
UBI: number of user volumes: 1
UBI: available PEBs: 270
UBI: total number of reserved PEBs: 1649
UBI: number of PEBs reserved for bad PEB handling: 19
UBI: max/mean erase counter: 1/0
UBI: image sequence number: 2126406133
atmel_spi atmel_spi.0: Atmel SPI Controller at 0xffffa4000 (irq 14)
UBI: background thread "ubi_bgt0d" started, PID 881
atmel_spi atmel_spi.1: Atmel SPI Controller at 0xffffa8000 (irq 15)
MACB_mii_bus: probed
eth0: no PHY found
usbcore: registered new interface driver rt2500usb
usbcore: registered new interface driver rt73usb
usbcore: registered new interface driver rt2800usb
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
at91_ohci at91_ohci: AT91 OHCI
at91_ohci at91_ohci: new USB bus registered, assigned bus number 1
at91_ohci at91_ohci: irq 22, io mem 0x00700000
usb usb1: New USB device found, idVendor=1d6b, idProduct=0001
usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
usb usb1: Product: AT91 OHCI
usb usb1: Manufacturer: Linux 2.6.36-rc6 ohci_hcd
usb usb1: SerialNumber: at91
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
usbcore: registered new interface driver usbserial
USB Serial support registered for generic
usbcore: registered new interface driver usbserial_generic
usbserial: USB Serial Driver core
USB Serial support registered for FTDI USB Serial Device
usbcore: registered new interface driver ftdi_sio

```

```

ftdi_sio: v1.6.0:USB FTDI Serial Converters Driver
USB Serial support registered for moto-modem
usbcore: registered new interface driver moto-modem
USB Serial support registered for GSM modem (1-port)
usbcore: registered new interface driver option
option: v0.7.2:USB Driver for GSM modems
atmel_usba_udc atmel_usba_udc: MMIO registers at 0xffff78000 mapped at c481e000
atmel_usba_udc atmel_usba_udc: FIFO at 0x00600000 mapped at c4900000
mice: PS/2 mouse device common for all mice
setting trigger mode 2 for irq 149 failed (gpio_irq_type+0x0/0x20)
ads7846 spi3.0: trying pin change workaround on irq 149
ads7846 spi3.0: touchscreen, irq 149
input: ADS7843 Touchscreen as /devices/platform/spi_gpio.3/spi3.0/input/input0
rtc-at91sam9 at91_rtt.0: rtc core: registered at91_rtt as rtc0
rtc-at91sam9 at91_rtt.0: rtc0: SET TIME!
i2c /dev entries driver
Linux video capture interface: v2.00
AT91: atmel_isi_init_module
AT91: atmel_isi_probe
atmel_isi atmel_isi: Reading configuration
atmel_isi atmel_isi: video buffer: 155648 bytes at ffa80000 (phys 73180000)
atmel_isi atmel_isi: video buffer: 155648 bytes at ffc80000 (phys 73a40000)
atmel_isi atmel_isi: video buffer: 155648 bytes at ffd00000 (phys 73a80000)
atmel_isi atmel_isi: video buffer: 155648 bytes at ffd80000 (phys 73ac0000)
atmel_isi atmel_isi: capture buffer: 1331200 bytes at ffb00000 (phys 0x73000000)kfifo_alloc
atmel_isi atmel_isi: Atmel ISI V4L2 device at 0xffffb4000
at_hdmac at_hdmac: Atmel AHB DMA Controller ( cpy slave ), 8 channels
usbcore: registered new interface driver hiddev
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
atmel_ac97c atmel_ac97c.0: Atmel AC97 controller at 0xc48a0000, irq = 24
ALSA device list:
  #0: Atmel AC97 controller
TCP cubic registered
NET: Registered protocol family 17
lib80211: common routines for IEEE802.11 drivers
rtc-at91sam9 at91_rtt.0: hctosys: unable to read the hardware clock
atmel_mci atmel_mci.0: Using dma0chan0 for DMA transfers
atmel_mci atmel_mci.0: Atmel MCI controller at 0xffff80000 irq 11, 1 slots
atmel_mci atmel_mci.1: Using dma0chan1 for DMA transfers
atmel_mci atmel_mci.1: Atmel MCI controller at 0xffffd0000 irq 29, 1 slots
UBIFS: mounted UBI device 0, volume 0, name "nandfs"
UBIFS: file system size: 208244736 bytes (203364 KiB, 198 MiB, 1614 LEBs)
UBIFS: journal size: 10450944 bytes (10206 KiB, 9 MiB, 81 LEBs)
UBIFS: media format: w4/r0 (latest is w4/r0)
UBIFS: default compressor: lzo
UBIFS: reserved for root: 4952683 bytes (4836 KiB)
VFS: Mounted root (ubifs filesystem) on device 0:12.
Freeing init memory: 132K
Initializing random number generator... done.
ifconfig: SIOCSIFHWADDR: No such device
ifconfig: SIOCSIFADDR: No such device
route: SIOCADDRT: Network is unreachable
Starting wi-fi network ...
Error for wireless request "Set Mode" (8B06) :
  SET failed on device wlan0 ; No such device.
Error for wireless request "Set ESSID" (8B1A) :
  SET failed on device wlan0 ; No such device.
ifconfig: SIOCSIFADDR: No such device
rm: can't remove '/var/lock/*': No such file or directory
Begin FPGA loading ...
pio mapped
pins preset done
prog pulse done, wait for init
input file size 464303
buffer allocated
file read and closed
start main loop
loaded 453 kB
main loop finished
FPGA started
start sequence completed
buffer freed
USB Host power management utility.
For SK-AT91SAM9G45(M10)-XC6SLX development board.
USB Host-0 turn ON, Host-1 turn ON
ARM-FPGA interconnection tests is beginning:
!!!Make sure downloaded testing FPGA configuration!!!
  ARM address bus test ... PASSED.
  Testing FPGA Block RAM ... PASSED.
  MicroBlaze selftest DDR2, waiting ready ... PASSED
  MicroBlaze selftest cycles - 1
All ARM-FPGA interconnection tests is PASSED.

```

Welcome to SK-AT91SAM9G45/M10-XC6SLX development board!

buildroot login:

Означает, что система успешно загрузилась и готова к работе. Для входа в консоль введите имя пользователя root, пароль не требуется (других пользователей в системе нет), после чего имеете полный консольный доступ к системе. Так же можно подключиться с помощью Telnet, FTP, HTTP, сетевой адрес платы 192.168.0.136. При подключении-отключении USB, uSD, SD/MMC карт памяти, они будут автоматически монтироваться-размонтироваться в системе.

Если был подключен SK-MI0430FT-Plug, на экране появится графическое изображение и сообщение о старте системы, при первой загрузке, необходимо откалибровать сенсорный экран с системной библиотекой TSLIB, для этого запустите ts_calibrate и следуйте инструкциям, после чего можете запустить ts_test для демонстрации.

Если были сконфигурированы J2, J5 и подключены наушники, в конце загрузки системы будет слышен музыкальный проигрыш.

2.1. Подключение модулей расширения

SK-MI0430FT-Plug – разъем X11

В штатной поставке ядро сконфигурировано на использование данного модуля расширения, в качестве контроллера TP включен ADS7843 (или аналог). Для работы TP непосредственно со встроенным в AT91SAM9G45 контроллером TP, необходимо:

а) на модуле расширения разорвать переключки J5-J8, переключки J1-J4 перевести в положение 2-3

б) в свойствах ядра (скрипт make_menuconfig) зайти в меню Device Drivers -> Input device support -> Touchscreens выключить модуль «ADS7846/TSC2046/AD7873 and AD(S)7843 based touchscreens», включить модуль «Atmel Touchscreen Interface»

в) пересобрать и обновить ядро (или просто загрузить ядро по TFTP)

SK-ATM0700D4-Plug – разъем X11

а) в свойствах ядра (скрипт make_menuconfig) зайти в меню Device Drivers -> Graphics support -> Support for frame buffer devices -> Starterkit.ru TFT plug selection выбрать «SK-ATM0700D4-Plug»

б) если предполагается использовать встроенный в ARM контроллер TP, в свойствах ядра Device Drivers -> Input device support -> Touchscreens выключить модуль «ADS7846/TSC2046/AD7873 and AD(S)7843 based touchscreens», включить модуль «Atmel Touchscreen Interface». На самом модуле расширения разомкнуть переключки J2,J3,J6,J7, переключки J4,J5,J8,J9 перевести в положение 2-3.

Если предполагается использовать внешний контроллер TP (на модуле расширения), в свойствах ядра Device Drivers -> Input device support -> Touchscreens включить модуль «ADS7846/TSC2046/AD7873 and AD(S)7843 based touchscreens», выключить модуль «Atmel Touchscreen Interface». На самом модуле расширения замкнуть переключки J2,J3,J6,J7, переключки J4,J5,J8,J9 перевести в положение 1-2

в) пересобрать и обновить ядро (или просто загрузить ядро по TFTP)

SK-HDMI-Plug – разъем X11

а) в свойствах ядра Device Drivers -> Input device support -> Touchscreens обязательно выключить модуль «Atmel Touchscreen Interface»

б) в свойствах ядра Device Drivers -> Graphics support -> Support for frame buffer devices -> Starterkit.ru TFT plug selection выбрать необходимые параметры экрана

в) на самом модуле расширения замкнуть J1, обращаю внимание, светодиод «DETECT» работает в «инверсном» режиме, т.е. когда монитор подключен – он гаснет

г) пересобрать и обновить ядро (или просто загрузить ядро по TFTP)

SK-VideoADC-Plug – разъем X2

Важно – на данный момент драйвер ISI не устанавливается в системе при разрешениях FB более 640x480, видимо есть неточности с выделением памяти, позже будем разбираться.

В штатной поставке драйвер ISI включен в ядро, для теста достаточно запустить скрипт `isi_test`, в результате работы которого появится графический файл `image.ppm`.

SK-SIMCOM-Plug – разъем X2

В папке корневой ФС `/bin/SK-SIMCOM-Plug` находятся скрипты для операций с различными модемами данного модуля расширения.

3. Состав ОС Linux

Ядро 2.6.36, включая драйвера:

- Ethernet
- NAND flash
- AC97
- USB-host
- USB-gadget
- I2C
- Image Sensor Interface (драйвер видеозахвата)
- SPI
- UART
- RTC
- WatchDog
- Frame Buffer
- TP ADS7843
- ...

4. Способы загрузки и содержимое корневой файловой системы

AT91SAM9G45/M10 подразумевает различные возможные источники загрузки, на плате их предусмотрено два - NAND flash и uSD карта

5.1. NAND flash

NAND flash разбита на две части:

- 1) 16M – для хранения загрузчиков, ядра системы и системы загрузки «safe mode»
- 2) 240M – раздел UBI файловой системы, используется в качестве корневой файловой системы

5.2. uSD

Загрузчик (нестираемый, располагаемый в самом процессоре), в случае отсутствия приложения на NAND flash (или разомкнут джампер NAND CS), пытается загрузить приложение с MCI0 порта процессора (uSD держатель). Карта должна иметь либо один раздел, либо обязательно первый раздел, отформатированный под FAT файловую систему и в корневой папке должны присутствовать:

- BOOT.BIN – загрузчик, верхний регистр имени важен (при копировании в Linux системах). Копирует из корневой папки карты image_sd.bin и передает ему управление
- Image_sd.bin – образ Linux системы, на диске присутствует его вариант со встроенной (initramfs) корневой файловой системой

Дополнительно, в корневую папку uSD карты можно скопировать с поставляемого диска файлы:

- system_prepare – скрипт подготовки системы к исходному состоянию. Корневая система safe режима, сканирует внешние накопители на предмет наличия system_prepare файла и запускает его. В текущем варианте, скрипт очищает первый раздел NAND flash, копирует необходимые образы, форматирует и распаковывает корневую ФС во второй раздел NAND flash.
- bootstrap – загрузчик для запуска с NAND flash
- uboot – загрузчик u-boot
- zlinux – ядро linux
- zlinux_safe – ядро linux включая корневую ФС
- rootfs.tgz – архив корневой ФС

При наличии всех вышеперечисленных файлов на uSD карте, система загрузится и подготовит, отформатирует и скопирует необходимые файлы на NAND flash, обращаю ВНИМАНИЕ – джампер NAND CS должен быть разомкнут во время включения-сброса платы и замкнут в первые секунды старта ядра (для того чтобы ядро правильно инсталлировало драйвер MTD устройства)!

При отсутствии system_prepare файла, система просто загрузится.

Корневая файловая система (ФС), в поставляемом варианте платы, хранится в NAND flash и монтируется во время загрузки, поэтому, следует внимательней относиться к изменениям в скриптах загрузки системы.

Корневая ФС содержит набор базовых приложений (большинство из которых являются реализацией мультифункционального приложения BusyBox), содержит:

- HTTPD – сервер HTTP
- FTPD – сервер FTP
- Telnetd – сервер Telnet
- TFTP – утилита приема-передачи файлов по TFTP протоколу
- Z-modem утилиты (для обмена файлами через COM порт)
- Microcom – терминальная программа
- TS-lib – набор утилит для операций с сенсорной панелью
- Memtester – тест памяти
- Mplayer – медиа-проигрыватель
- MC – файловый менеджер
- ...

На случай аварии корневой файловой системы, предусмотрен режим «Safe boot», для его активации необходимо прервать загрузку в U-boot (нажав на любую клавишу) и выполнить команду «run safe_boot». Загрузится образ системы, в котором корневая ФС расположена в памяти и можно будет приступить к ремонту основной корневой ФС,

например, запустить скрипт «install_rootfs», в результате работы которого будет заново отформатирован второй раздел NAND flash, скопирован с TFTP сервера и распакован архив корневой ФС.

5.2. Управление питанием USB портов

Если в программном управлении USB портов нет необходимости, можно замкнуть соответствующие джампера, для постоянной подачи питания.

Утилита «usbh_power» управляет подачей питающих напряжений, например «usbh_power 1 1» - подключит питающее напряжение к обоим USB портам.

5.3. Утилиты взаимодействия с FPGA

Загрузку конфигурации в FPGA может осуществлять ARM процессор, т.е. для конфигурирования FPGA не потребуются специальные кабели и специальные внешние микросхемы хранения конфигурации. В составе корневой ФС присутствует утилита для загрузки FPGA - «fpga_loader», аргументом запуска которой является стандартный BIT файл. По завершении работы «fpga_loader» должен загореться светодиод HL1 «Done», что символизирует успешную загрузку конфигурации. Часть лога системы, демонстрирующий работу утилиты:

```
Begin FPGA loading ...
pio mapped
pins preset done
prog pulse done, wait for init
input file size 464303
buffer allocated
file read and closed
start main loop
loaded 453 kB
main loop finished
FPGA started
start sequence completed
buffer freed
```

Для принудительного обновления конфигурации FPGA достаточно запустить скрипт «fpga_load» и предварительно обновить (если необходимо) файл конфигурации в корневой ФС самой платы /lib/firmware/fpga.bit .

По умолчанию, в FPGA загружается демонстрационный-тестовый проект на основе синтезируемого процессора MicroBlaze, который тестирует DDR2 память подключенную к FPGA и передает результаты теста ARM процессору. Так же FPGA проект содержит блочную память и регистр-защелку (с возможностью чтения) адресной шины. В составе корневой ФС есть утилита для тестирования взаимодействия с FPGA – «fpga_test», алгоритм работы:

- тестирует адресную шину (используя регистр-защелку)
- тестирует шину данных – записывает в блочную память FPGA последовательность случайных чисел и контрольную сумму, в последствии считывает и сравнивает суммы
- считывает и показывает результаты тестов MicroBlaze

Часть сообщений системы демонстрирующий работу утилиты:

```
ARM-FPGA interconnection tests is beginning:
!!!Make sure downloaded testing FPGA configuration!!!
ARM address bus test ... PASSED.
Testing FPGA Block RAM ... PASSED.
MicroBlaze selftest DDR2, waiting ready ... PASSED
MicroBlaze selftest cycles - 1
```

Так же предусмотрена возможность конфигурирования FPGA с внешней микросхемы SPI DataFlash, для этого должны быть замкнуты J7 и J13, после кратковременного замыкания-размыкания J6 (или включения питания) FPGA загрузит конфигурацию.

5. Виртуальная машина VMware

Для сборки ядра и корневой ФС используется виртуальная машина VMware с установленной ОС Debian, в состав которой входят все исходные тексты, компилятор и утилиты для сборки (toolchain), скрипты. Так же в виртуальной машине установлены и настроены сервисы для удобства взаимодействия с «материнской» ОС и отладочной платой: SSH, FTP, TFTP.

Разархивируйте файл “SK-AT91SAM9G45-XC6SLX_linux_build_machine.exe”, установите VMware-player или VMware, откройте проект виртуальной машины.

Для работы необходимо настроить сетевые интерфейсы (появляющиеся после установки VMware), присвоив им описываемые ниже IP адреса:

Eth0 (Bridget) с адресом 192.168.0.2, предназначен для взаимодействие с платой, для загрузки образов по TFTP ... Т.е. для нормальной работы, потребуется присвоить IP адрес PC сетевой карты (к которой подключается отладочная плата) 192.168.0.1

Eth1 (Host-only) с адресом 192.168.2.2, предназначен для взаимодействия с PC (т.к. Bridget интерфейс отключается при физически выключенном кабеле), в частности, для возможности копирования файлов из виртуальной системы по FTP. В свойствах сетевых устройств, этому виртуальному адаптеру нужно присвоить IP 192.168.2.1

После правильной настройки (и с подключенной платой) должны успешно проходить PING с PC по адресам 192.168.2.2, 192.168.0.2, 192.168.0.136.

После того, как сетевые интерфейсы настроены, можно запускать виртуальную машину, после загрузки ее не обязательно выключать, достаточно будет нажать кнопку паузы и во время следующего сеанса работы не придется ждать загрузки виртуальной ОС, но при этом, в некоторых случаях, нужно следить за системными временем, особенно при копировании новых файлов (имеющих более позднюю дату создания относительно системы) для сборки.

По умолчанию, в системе присутствует два пользователя:

- root, пароль 123456
- user, пароль 123456 (настоятельно рекомендую работать под этим пользователем, или создать нового, но не вести всю работу под root)

После входа переключаемся на консоль (Ctrl+Alt+F(1-6)) (потребуется в опциях VMware освободить сочетание клавиш Ctrl+Alt - по умолчанию это выход из окна виртуальной машины), запускаем MidnightComander (mc).

Основная рабочая папка /home/user/src, ее содержимое:

- buildroot-2010.08 - пакет сборки корневой файловой системы
- buildroot-2010.08_safe - пакет сборки корневой файловой системы для «Safe mode»
- linux-2.6.36 - ядро, скрипты сборки внутри
- u-boot-2010.XX - вторичный загрузчик (загружается SAM-BA в NAND flash по адресу 0x20000)

В корневом каталоге ядра присутствует два скрипта:

make_kernel – собирает ядро и копирует файл в папку TFTP сервера

make_menuconfig – запускает конфигурационное меню ядра

Так же в корневом каталоге ядра присутствуют образец конфигурационного файла для «Safe mode» загрузки – «config_safe».

В корневом каталоге buildroot-2010.08 присутствует скрипт:

build_system – собирает файловую систему и запускает скрипт сборки ядра

Например, необходимо обновить ядро Linux, для этого:

- запускаем скрипт linux-2.6.XX/make_kernel
- включаем/перезагружаем плату с подключенным Ethernet и RS232 кабелями
- прерываем в u-boot процесс загрузки нажатием любой клавиши
- выполняем “run system_update”

6. Общий принцип работы системы

После подачи питания (перезагрузки), процессор запускает первичный загрузчик (находится во внутренней не перепрограммируемой ROM) и по определенному алгоритму определяет наличие исполняемого кода во внешних носителях. Если приложение не найдено, процессор остается в режиме, который подразумевает взаимодействие с ним утилиты SAM-BA, которая позволяет программировать внешние носители, подключенные к процессору.

Поскольку внешняя DDR2 (или любая другая память не инициализирована), первое запускаемое приложение должно быть загрузчиком т.к. его максимальный размер не может превышать размера внутренней памяти процессора. Это приложение (загрузчик) в первую очередь должен проинициализировать внешнюю память (например, правильно настроить параметры DDR2), скопировать исполняемое приложение из внешней Flash памяти во внешнюю DDR2 память и передать ему управление.

В нашем контексте, первым приложением является так называемый bootstar загрузчик (предоставляемый фирмой Atmel и адаптированный под конкретную плату), который инициализирует DDR2, копирует из NAND приложение и запускает его.

Вторым приложением так же является загрузчик (u-boot) но с уже более обширными возможностями, например, он умеет копировать файлы по TFTP, поддерживает целый набор команд и режимов. В переменных окружения u-boot есть команда запуска, в которой указано, по какому адресу NAND flash следует прочитать образ ядра, куда этот образ памяти записать и по какому адресу запустить. Следующие сообщения консоли иллюстрируют этот процесс:

```
U-Boot 2010.09 (Nov 16 2010 - 23:20:50)

DRAM: 64 MiB
## Unknown FLASH on Bank 1 - Size = 0x00000000 = 0 MB
Flash: 0 Bytes
NAND: 256 MiB
*** Warning - bad CRC or NAND, using default environment

In: serial
Out: serial
Err: serial
Net: macb0
Hit any key to stop autoboot: 0

NAND read: device 0 offset 0x80000, size 0x790000
7929856 bytes read: OK
```

Переменные окружения u-boot подразумевают возможность хранить команды и некоторые аргументы (например, IP адрес сервера) в специально отведенном блоке flash памяти,сообщение вида “ Warning – bad CRC, using default environment” не говорит о том, что NAND flash неисправна, оно означает, что u-boot не находит валидные переменные окружения и использует переменные окружения по умолчанию.

Перед запуском Linux, ядро сначала проверяет контрольную сумму собственного архива, распаковывает себя и запускает, иллюстрация:

```
## Booting kernel from Legacy Image at 70400000 ...
Image Name:      Linux Kernel Image
Image Type:      ARM Linux Kernel Image (gzip compressed)
Data Size:       2006613 Bytes = 1.9 MiB
Load Address:    70008000
Entry Point:     70008000
Verifying Checksum ... OK
Uncompressing Kernel Image ... OK

Starting kernel ...
```

Далее идет инициализация всей системы, драйверов, файловых систем, после чего управление передается скрипту начального запуска, который, в свою очередь, запускает утилиту конфигурирования FPGA, утилиту тестирования взаимодействия с FPGA.

7. Программирование внешних носителей с помощью SAM-BA

В большинстве случаев, обновить-восстановить систему можно из загрузчика u-boot, но если он или bootstrap поврежден, тогда восстановить систему можно только с помощью утилиты (предоставляемой фирмой Atmel) SAM-BA. К сожалению, с ростом версии этой утилиты, стабильность ее работы не улучшается, скорее наоборот ...

В добавок, на текущий момент AT91SAM9G45 содержит ошибку USB соединения во внутреннем загрузчике (см. errata), к счастью, ошибка не фатальна и при многократном повторе попытки соединения, связь установить можно.

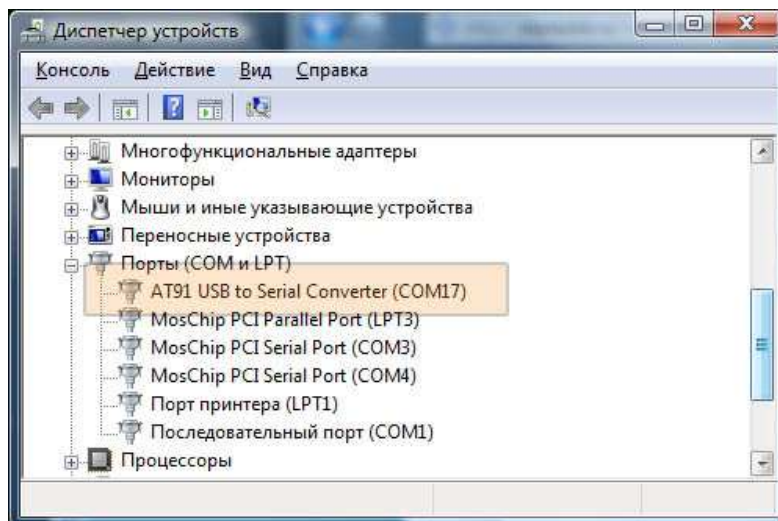
В общем виде, возможно подключение к плате 3-мя интерфейсами USB-device, через COM порт, JTAG (в текущей версии его поддержка отсутствует, присутствовала в более ранних версиях SAM-BA). Самым «надежным» себя зарекомендовал USB интерфейс, обращаю внимание, после установки драйвера (при первом подключении платы) не следует менять порт USB хаба вашего PC для взаимодействия с платой.

Для того, чтобы процессор был способен взаимодействовать с SAM-BA, необходимо, чтобы его первоначальный загрузчик не смог найти исполняемые коды во внешних носителях (в случае с JTAG интерфейсом этого условия не требуется), для этого достаточно разомкнуть переключки J2 и выключить-включить питание платы, после этого, J2 следует замкнуть.

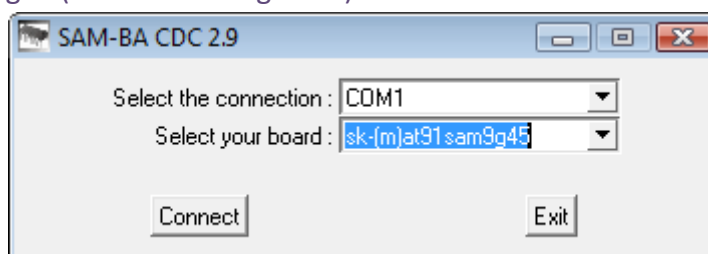
8.1 Распаковываем sam-ba_2.9_cdc_xp_vista.rar, инсталляции не требует, скрипты плат уже в архиве.

8.2 Подключаем USB-A кабель к разъему X13, J7 обязательно разомкнут!!! При включении питания, система должна найти новое USB устройство (если ранее эта процедура не выполнялась), присваиваем ему драйвер /drv/atm6124_cdc.inf .

В результате манипуляций, при подключении платы, в диспетчере устройств должен появляться "AT91 USB to Serial Converter COMXX":



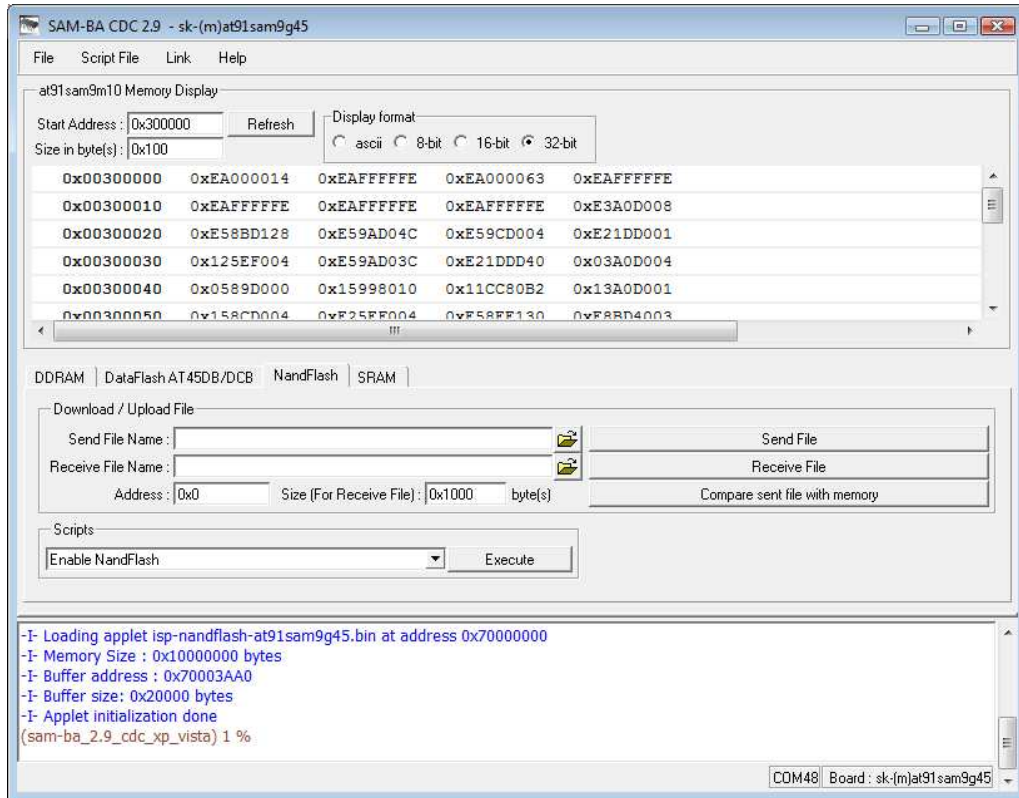
8.3 Запускаем SAM-BA, выбираем COM порт AT91 USB to Serial Converter, выбираем плату sk-(m)at91sam9g45 (или at91sam9g45-ek):



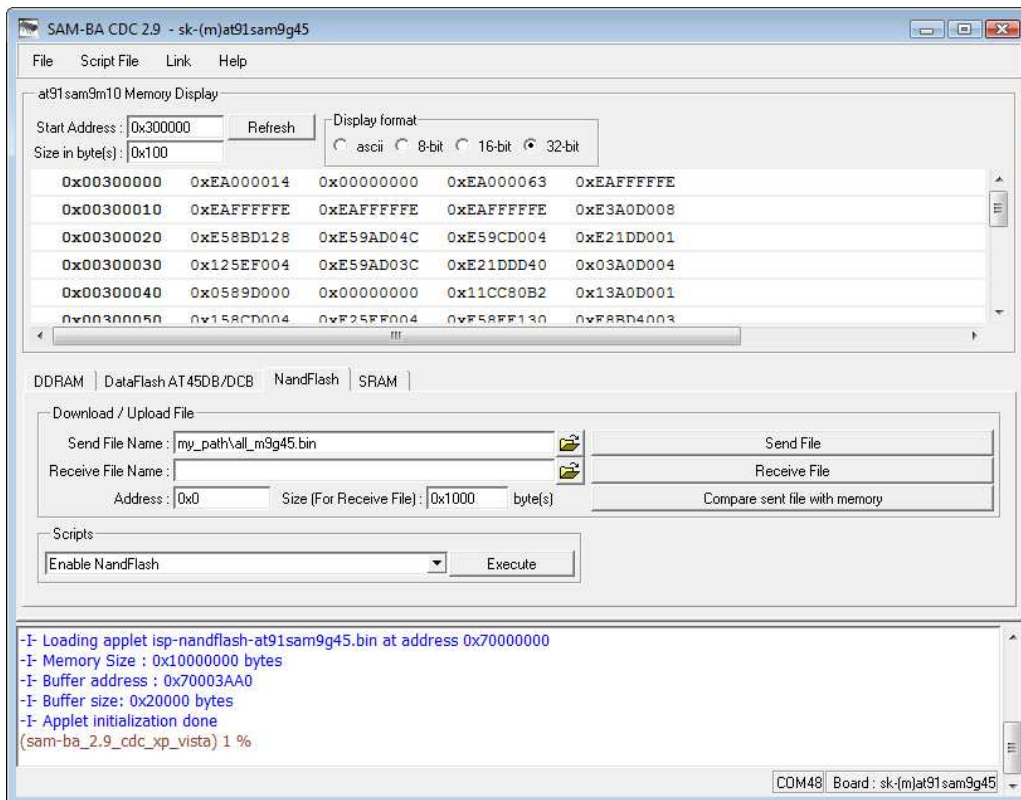
В виду описываемой выше ошибки внутреннего загрузчика AT91SAM9G45, соединение может устанавливаться далеко не с первого раза, необходимо повторять (возможно потребуется это сделать не один десяток раз) процедуру:

- 1) удалить через диспетчер задач процесс SAM-BA
- 2) нажать кнопку сброса на плате
- 3) запустить SAM-BA

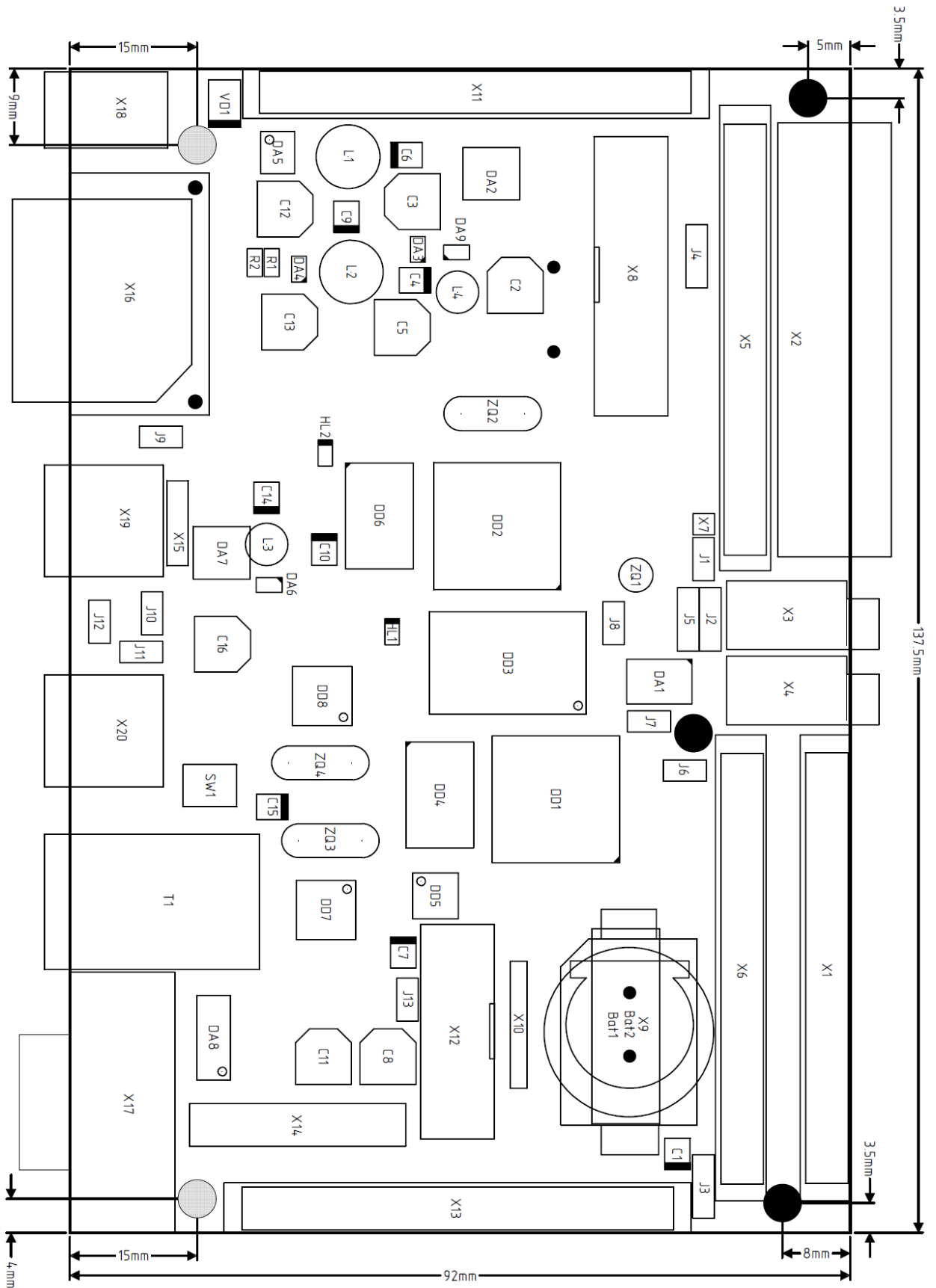
7.4 Выполняем скрипт Enable NandFlash:



8.5 Открываем файл all_9g45.bin и записываем (Send File) его по адресу 0:



8.6 По окончании записи (примерно через 2-3 минуты) перезагружаемся, если основная корневая ФС была удалена, останавливаем процесс загрузки u-boot, выполняем "run safe_boot" для дальнейших операций с корневой ФС.



Габаритный чертеж