2273-5

Second Workshop on Open Source and the Internet for Building Global Scientific Communities with Emphasis on Environmental Monitoring and Distributed Instrumentation

28 November - 16 December, 2011

Booting the ARM systems

U. Raich

CERN, Geneva
Switzerland

Booting the ARM Systems

Ulrich (Uli) Raich
CERN BE department
Beam Instrumentation Group

Reminder from last lecture

To start the system we need:

- A boot loader
- A Linux kernel
- A Linux root file system

You find all ARM software in:

/opt/ICTP/micros/armputer-vmax

Boot possibilities

- Boot from SD cards (looks for boot.bin file)
- Boot from Nand flash (looks for valid reset vectors)
- Boot from dataflash (looks for valid reset vectors)

We will use the dataflash option (I do not have 15 SD cards and there is no Nand flash on the boards.

The boot loader

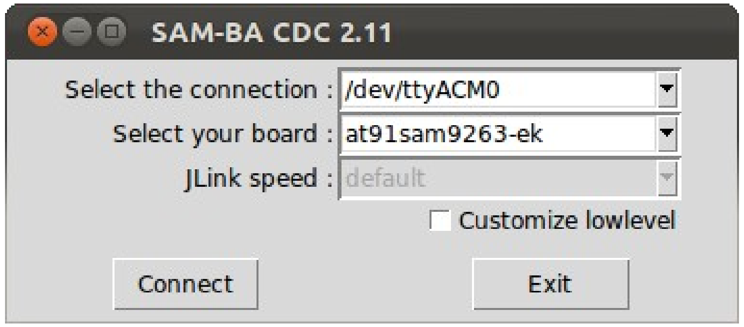
- Atmel delivers the boot loader in Open Source
- Atmel delivers
  sam-ba, the at91sam boot assistant.
- Sam-ba uses serial over USB and driver must be installed
- lsmod | grep usbserial
  if usbserial is loaded, remove everbody using it as well as usbserial itself using
  rmmod usbserial

Check, who is on USB

- Take off the jumper near the console port
- Power the ARM with USB cable connected
- Check USB devices with
  lsusb
- Find the vendor and product ID of the Arm (Atmel) board
  0xvvvv:0xpppp
- modprobe cdc_acm
- lsmod
- ls /dev/ttyACM*

SAM-BA

- If everything os ok:



SAM-BA CDC 2.11

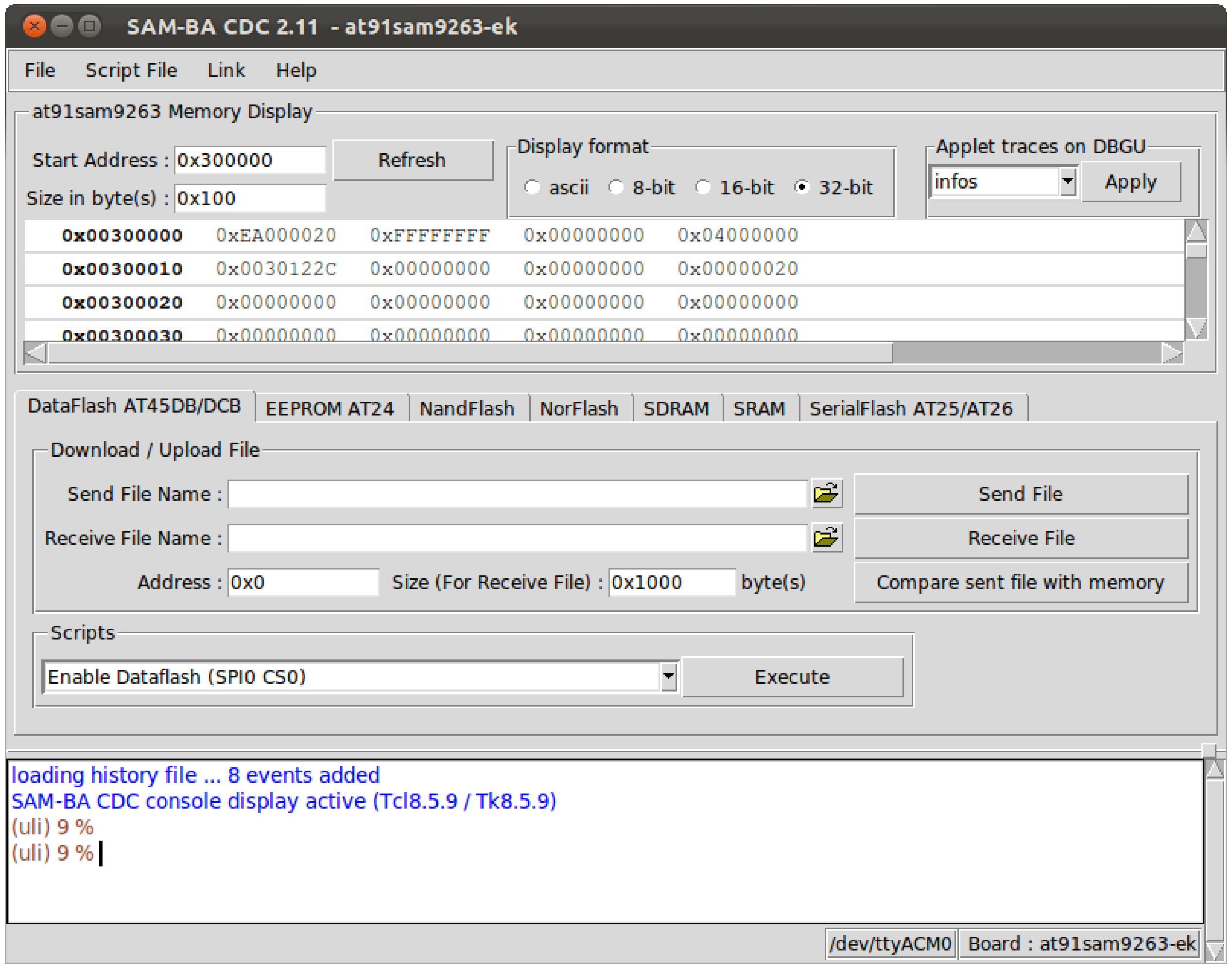
Select the connection : /dev/ttyACM0

Select your board : at91sam9263-ek

JLink speed : default

☐ Customize lowlevel

The SAM-BA window



SAM-BA CDC 2.11 - at91sam9263-ek

File Script File Link Help

at91sam9263 Memory Display

Start Address : 0x300000 Refresh

Size in byte(s) : 0x100

Display format: ascii 8-bit 16-bit [✓ 32-bit]

Applet traces on DBGU: infos Apply

```
0x00300000   0xEA000020   0xFFFFFFFF   0x00000000   0x04000000
0x00300010   0x0030122C   0x00000000   0x00000000   0x00000020
0x00300020   0x00000000   0x00000000   0x00000000   0x00000000
0x00300030   0x00000000   0x00000000   0x00000000   0x00000000
```

DataFlash AT45DB/DCB | EEPROM AT24 | NandFlash | NorFlash | SDRAM | SRAM | SerialFlash AT25/AT26

Download / Upload File

Send File Name : Send File

Receive File Name : Receive File

Address : 0x0 Size (For Receive File) : 0x1000 byte(s) Compare sent file with memory

Scripts

Enable Dataflash (SPI0 CS0) Execute

```
loading history file ... 8 events added
SAM-BA CDC console display active (Tcl8.5.9 / Tk8.5.9)
(uli) 9 %
(uli) 9 %
```

/dev/ttyACM0 Board : at91sam9263-ek

Getting started

- Use Atmel’s *getting started* procedure
- Enable the SDRAM clicking on the TAB
- Define the environment variable ARMDIR to point to /opt/ICTP/micros/arm/armputer-vmax and put it into your .bash_profile file
- Define the file $ARMDIR/blinkingLEDs/at91sam9263_getting_started.bin
- Send File (to the ARM)
- Execute the program on the ARM:
  go 0x2000000

Load boot loaders

- Put back the jumper
- Enable Dataflash (SPI0 CS0)
- Send Boot File

  This file goes onto address zero of the dataflash and contains reset vectors recognized by the at91sam9263
- Once this file is programmed it will automatically be executed at power up

The primary boot loader is in $ARMDIR/Bootstrapv1.13/board/at91sam9263ek/dataflash /dataflash_at91sam9263ek.bin

This program is delivered by Atmel

Secondary Bootloader

- “Das u-boot” is supplied by Denx, a german company
- It contains drivers for flash memories and Ethernet and allows to download the kernel
- We burn $ARMDIR/u-boot-1.3.4-u-boot.bin into the dataflash at address 0x8400
- Send File and Execute

U-boot

- We have a command interpreter now
- This gives access to the at91sam9263 hardware
- Test the network with ping
- Configure u-boot through environment variable
- Configure bootcmd and bootargs
- Configure kernel filename

Boot the machine

- Save the environment (saveenv)
- dhcp downloads the kernel
- bootm starts it

The kernel must have the network driver and nfs compiled in

The NFS server must give access to the root file system

Possible Problems

- Badly configured dhcpd.conf
- Network on the PC badly configured
- /etc/exports file not ok
- dhcp and/or have not been restarted after modification of configuration files

First steps with arm linux

- Log in with

  root/openICTP
- Have a look if you see things different from the PC system

Compile a program for the ARM

- Buildroot: a collection of Makefiles to build an arm system

  Builds

  - A cross compiler toolchain
  - Builds all the libraries needed
  - Builds the root file system
  - Builds the Linux kernel
  - Builds the boot loders

Build helloworld for ARM

- Must use the cross compiler tools

  arm-linux-gcc to cross compile the program
- Use *file* to see the file type
- Cannot execute on the PC but must be copied to a directory seen by the ARM
- For testing you can use qemu

Scratchbox

- A sandbox to compile and run arm program
- Allows to log into an arm environment
- Now gcc becomes arm-linux-gcc
- Running works through qemu