



CUBIEBOARD
<http://cubieboard.org>

Using ARM Streamline base on Cubieboard

ARM-DS-5

Website: <http://cubieboard.org/>
Support: support@cubietech.com



文档版本	作者	初审	复审
V-0.1-20150122	Payne		



Table of Contents

1. Abstract	4
2. Introduction of ARM Streamline	4
2.1. What is ARM Streamline ?	4
2.2. Why use Streamline ?	4
3. Preparation before development.....	5
4. Download the source code and tools.....	5
5. Add kernel options.....	6
5.1. General Setup	6
5.2. Kernel Features	7
5.3. CPU Power Management	8
5.4. Kernel hacking.....	9
6. Compile Gator driver.....	10
7. Register arm account.....	11
8. Installing DS-5 on your computer	14
9. Open Streamline debugging tools in the CT.....	16
9.1. Load drive and shell.....	16
9.2. Uses the ADB to interactive data.....	18
9.3. The use of interactive network data.....	18
10. Use DS-5.....	19
10.1. Create project of Streamline Data	20
10.2. The working effect of DS-5 graph.....	22
10.3. Streamline simple analysis.....	24

1. Abstract

The document which described how to debug the Android or Linux system with ARM-Streamline on cubieboard, the ways of debugging including network debugging and ADB debugging(Android only).

2. Introduction of ARM Streamline

2.1. What is ARM Streamline ?

ARM Streamline performance analyzer is a part of the ARM DS-5 tool chain, it has made the software developers make full use of the available resources to create a high performance and high energy efficiency products which based on the ARM-processor 's system. It has a visual graphical user interface that can display information from the CPU and the GPU performance counters to the source code hotspots and display the actual power consumption, in this way, the developers relieve the performance bottleneck easily, improve the code parallelism, extend the battery life and enhance the user experience. Streamline based on system tracking point, hardware and software performance counters, and sample analysis and user comments. It offers the powerful functional system analysis environment used for the software optimization. More details and introductions please visit the homepage: <http://ds.arm.com/>

2.2. Why use Streamline ?



Improve the rate of code

- Find the position of CPU which consumed the time is more
- Improve multi-core platform code parallelism
- Adjust the code for achieve the most optimal use of the cache and vector.



Reduce energy consumption

- Using the ARM energy detector to monitor actual power, current and voltage
- Find out the chance of improved power management solutions
- The optimization of calculation task is to achieve the best energy efficiency



Use the system resources effectively

- Analysis and optimization of Mali GPU utilization and CPU code
- Monitor the CPU and Mali GPU cache usage and system memory
- Check the distribution of the load across multiple cores.



Customize system accordingly

- Make your data connect to Streamline analysis view
- Expand open source drivers for watch variables and components
- Test the code like printf which sends message to streamline

3. Preparation before development

- 1) Ubuntu12.04 operating system of the computer
- 2) A piece of Cubietruck development board
- 3) USB-MiniUSB data cable, for PC and development board data interaction
(The Linux system does not have ADB tools, data interaction through network)
- 4) Download the source code of cubieboard android or linux
- 5) Download the DS-5 source code pack

4. Download the source code and tools

Linux source code:

```
$ mkdir linux-sdk-card
```

```
$ cd linux-sdk-card
```

1) kernel-source:

```
$ git clone https://github.com/cubieboard/linux-sdk-kernel-source.git
```

```
$ mv linux-sdk-kernel-source linux-sunxi
```

2) tools:

```
$ git clone https://github.com/cubieboard/linux-sdk-card-tools.git
```

```
$ mv linux-sdk-card-tools tools
```

3) products:

```
$ git clone https://github.com/cubieboard/linux-sdk-card-products.git
```

```
$ mv linux-sdk-card-products products
```

4) rootfs&u-boot:

```
$ git clone https://github.com/cubieboard/linux-sdk-binaries.git
```

```
$ mv linux-sdk-binaries binaries
```

Get file from:

```
http://dl.cubieboard.org/model/commom/linux-sdk-binaries
```

android4.2 source code:

```
git clone https://bitbucket.org/cubietech/a20-android4.2_lichee.git
```

```
git clone https://bitbucket.org/cubietech/a20-android4.2\_android.git
```

DS-5 tool for source code package:

```
http://pan.baidu.com/s/1pJG66bL
```

How to compile and build the cubieboard firmware, please refer to the following tutorial:

android: <http://pan.baidu.com/s/1dDF5cVR>

linux: <http://pan.baidu.com/s/1o6LYsDs>

5. Add kernel options

Support ARM Streamline need to recompile the kernel, the public version of SDK ,
Android kernel directory:

```
lichee/linux-3.4
```

The kernel directory of linux is linux-sunxi .

5.1. General Setup

Into “General setup” and put on “Profiling support”, as below:

```
(0) Default panic timeout
[ ] Configure standard kernel features (expert users) --->
[ ] Embedded system
    Kernel Performance Events And Counters --->
[*] Disable heap randomization
    Choose SLAB allocator (SLAB) --->
[*] Profiling support
<*> Profile system profiling
[ ] Kprobes
[ ] Optimize very unlikely/likely branches
    COV-based kernel profiling --->
```

Enter “General setup”-> Kernel Performance Events And Counters Selecte “Kernel performance events and counters”, as below:

```
Kernel Performance Events And Counters
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

[*] Kernel performance events and counters
[ ] Kernel performance counters (old config option)
[ ] Debug: use vmalloc to back perf mmap() buffers
```

5.2. Kernel Features

Enter “Kernel Features” Selecte “High Resolution Timer Support”, as below:

```
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

[*] Tickless System (Dynamic Ticks)
[*] High Resolution Timer Support
[*] Symmetric Multi-Processing
[*] Allow booting SMP kernel on uniprocessor systems (EXPERIMENTAL)
[*] Support cpu topology definition
[*] Multi-core scheduler support
[*] SMT scheduler support
[*] Architected timer support
[*] Timer counter delay
Memory split (3G/1G user/kernel split) --->
```

Enter “Kernel Features” Selecte “Enable hardware performance counter support for perf events ”, as below:

```
(2) Maximum number of CPUs (2-32)
-*- Support for hot-pluggable CPUs (EXPERIMENTAL)
[*] Use local timer interrupts
    Preemption Model (Preemptible Kernel (Low-Latency Desktop)) --->
[ ] Compile the kernel in Thumb-2 mode (EXPERIMENTAL)
[*] Use the ARM EABI to compile the kernel
[*] Allow old ABI binaries to run with this kernel (EXPERIMENTAL)
[*] High Memory Support
[*] Allocate 2nd-level pagetables from highmem
[*] Enable hardware performance counter support for perf events
    Memory model (Flat Memory) --->
[ ] Allow for memory compaction
[ ] Enable KSM for page merging
(4096) Low address space to protect from user allocation
[ ] Enable cleancache driver to cache clean pages if tmem is present
[ ] Use kernel mem{cpy,set}() for {copy_to,clear}_user() (EXPERIMENTAL)
```

Enter “Kernel Features” Selecte Use local timer interrupts, as below:

```
[ ] Architecture timer support
[*] Timer counter delay
    Memory split (3G/1G user/kernel split) --->
(2) Maximum number of CPUs (2-32)
-*- Support for hot-pluggable CPUs (EXPERIMENTAL)
[*] Use local timer interrupts
    Preemption Model (Preemptible Kernel (Low-Latency Desktop)) --->
[ ] Compile the kernel in Thumb-2 mode (EXPERIMENTAL)
[*] Use the ARM EABI to compile the kernel
[*] Allow old ABI binaries to run with this kernel (EXPERIMENTAL)
[*] High Memory Support
[*] Allocate 2nd-level pagetables from highmem
```

5.3.CPU Power Management

Enter CPU Power Management -> CPU Frequency scaling
Select “CPU Frequency scaling”, as below:


```

CPU Frequency scaling
Arrow keys navigate the menu. <Enter> selects submenus ---. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

[*] CPU Frequency scaling
<*> CPU frequency translation statistics
[*] CPU frequency translation statistics details
Default CPUFreq governor (performance) --->
-* 'performance' governor
<*> 'powersave' governor
<*> 'userspace' governor for userspace frequency scaling
<*> 'ondemand' cpufreq policy governor
< > 'interactive' cpufreq policy governor

```

5.4. Kernel hacking

Enter “Kernel Features” select Tracers as below:

```

< > CPU notifier error injection module
[ ] Fault-injection framework
[*] Debug page memory allocations
[*] Deprecated power event trace API, to be removed
[*] Tracers --->
[ ] Enable dynamic printk() support
[ ] Enable debugging of DMA-API usage
[ ] Perform an atomic64_t self-test at boot
[ ] Sample kernel code --->

```

The final confirm "CONFIG_GENERIC_TRACER" and "CONFIG_TRACING" were selected, as below:

```

SEARCH RESULTS
Symbol: GENERIC_TRACER [=y]
Type : boolean
Selects: TRACING [=y]
Selected by: FUNCTION_TRACER [=y] && TRACING_SUPPORT [=y] && FTRACE [=y] && HAVE_FUNCTION_TRACER [=y] || I

```

```
Symbol: TRACING [=y]
Type : boolean
Selects: DEBUG_FS [=y] && RING_BUFFER [=y] && STACKTRACE [=y] && TRACEPOINTS [=y] && NOP_TRACER [=y] && BI
Selected by: GENERIC_TRACER [=y] || ENABLE_DEFAULT_TRACERS [=n] && TRACING_SUPPORT [=y] && FTRACE [=y] &&
```

When the above options were selected, compile the kernel, making firmware, we will use the firmware later.

6. Compile Gator driver

Unzip the source code package from the first step and get the gator-driver driver source code. Then starting to compile the gator.ko drive.

Enter gator-driver directory

\$ cd gator-driver

Executive compiler directive, please replace the path corresponding to the kernel, ensure that the PC development environment has been installed on the cross compiler tool.

\$ make -C /work/android4.2_tablet_A20/lichee/linux-3.4 M=`pwd` ARCH=arm

CROSS_COMPILE=arm-linux-gnueabi- modules

It will generate gator.ko in the current directory after compiled successfully,as below:

```
parker@parker:/work/jtag/gator-driver$ make -C android4.2_tablet_A20/lichee/linux-3.4 M=`pwd` ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules
make: *** android4.2_tablet_A20/lichee/linux-3.4: No such file or directory. Stop.
parker@parker:/work/jtag/gator-driver$ make -C /work/android4.2_tablet_A20/lichee/linux-3.4 M=`pwd` ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- modules
make: Entering directory `/work/android4.2_tablet_A20/lichee/linux-3.4'
  CC [M] /work/jtag/gator-driver/gator_main.o
  CC [M] /work/jtag/gator-driver/gator_events_block.o
  CC [M] /work/jtag/gator-driver/gator_events_irq.o
  CC [M] /work/jtag/gator-driver/gator_events_meminfo.o
  CC [M] /work/jtag/gator-driver/gator_events_mmapped.o
  CC [M] /work/jtag/gator-driver/gator_events_net.o
  CC [M] /work/jtag/gator-driver/gator_events_perf_pmu.o
  CC [M] /work/jtag/gator-driver/gator_events_sched.o
  CC [M] /work/jtag/gator-driver/gator_events_armv6.o
  CC [M] /work/jtag/gator-driver/gator_events_armv7.o
  CC [M] /work/jtag/gator-driver/gator_events_l2c-310.o
  CC [M] /work/jtag/gator-driver/gator_events_scorpion.o
  LD [M] /work/jtag/gator-driver/gator.o
Building modules, stage 2.
MODPOST 1 modules
  CC      /work/jtag/gator-driver/gator.mod.o
  LD [M] /work/jtag/gator-driver/gator.ko
make: Leaving directory `/work/android4.2_tablet_A20/lichee/linux-3.4'
```

Note: The part of yellow need to fill according to their actual situation, the kernel path to choose what you used the kernel in the second step. The cross compiler tool as same as kernel.

7. Register arm account

Teaching you sign up for an arm account here, you can obtain the right of DS-5 30 day trial and skip this step if already have a arm account .

Access to the arm registration page directly

<https://login.arm.com/register.php>

Registered in 4 steps, just to fill in with * :

ARM registration **1** Name | 2 Details | 3 Activation | 4 Complete

Welcome to ARM

Use this form to register for a customer account with ARM. You do not need to re-register if you already have an account for the applications below. You may login in the upper right area now.

- silver.arm.com (for downloads, support cases)
- DesignStart for Downloads for Physical IP and Processor Design kits

ARM Connected Community access

Reading blogs, discussions, the partner directory, whitepapers, news, videos, technical documents and discussions doesn't require an account. To contribute, and take part in the discussions, please [register to the community](#)

Enter your name and email address * Required



Email Address: *


We recommend using your business email address to ensure you can access all of your relevant services.

First Name: *

Last Name: *

Word Verification:

 Audio  Try a new code



Type the characters you see in the picture to the left: *

Next



Continue to fill in the information:

ARM registration **1** Name | **2** Details | **3** Activation | **4** Complete

Enter your details * Required

Email Address:

First Name: *

Last Name: *

Preferred Name:

Preferred Language:

Company Name: *

Job Title:

Address:

City:

Country: *

State/Country:

Zip/Postal Code:

500 Characters left

Telephone:

Mobile:

Fax:

Choose your password * Required

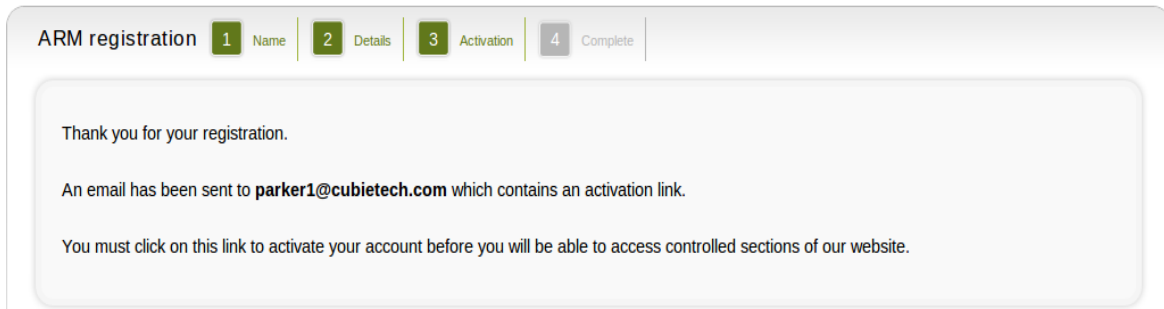
Choose a password: *

Confirm password: *

Passwords must have:

- A minimum length of 8 characters
- At least one uppercase and lowercase letter
- At least one number or special character

From time to time, your details may be used to send information regarding ARM's products and services that we believe would be of interest to you. If you would like to receive this information please tick this box



Registration information has sent to your mailbox, you can complete the registration with click on the link in the mailbox.

8. Installing DS-5 on your computer

Enter the source code directory of DS500-BN-00019-r5p0-20rel1

Add executable permissions of script

\$ chmod +x install.sh

Execute the script

\$./install.sh

Select B to enter the next step, as shown below:

```
=====
Welcome to the Installer for ARM DS-5
=====
--- Host target check...[x86_64]
This installation has a post install step that requires root privileges
The post install stage performs the following functions:
- Set default toolkit selection
- Installation of USB drivers for RealView ICE and DSTREAM hardware units

(A) Abort the installation and try again as a user with root privileges (Recommended)
(B) Continue with the installation and skip stages that require root privileges
    (execute "run_post_install_for_ARM_DS-5.sh" afterwards as root to gain full functionality)

Please answer with one of: 'A/a' or 'B/b'
Enter option: [default: A] B
```

Always press the Enter key to reading information , enter "yes", as shown below:

```
R. Clang is licensed to you under the University of Illinois/NSCA Open Source License.

S. The Python interpreter and standard libraries, version 2.7.4 found in your installation at <install_dir>\sw\python
2.7\
and licensed to you under the Python Software Foundation Licence for Python 2.7.4. This package is also subject to ot
her third party licenses.

T. libstdc++ is licensed to you under the GNU General Public License version 3 plus runtime exception.

U. Portions of the software and firmware of ARM's Target Connection Products contain:
(i) an ARM Embedded Linux operating system together with patches developed by ARM to the Linux kernel, both of which
are licensed under the GNU General Public License version 2;
(ii) the GNU C library (glibc), licensed to you under the GNU Lesser General Public License versions 2.0 and 2.1;
(iii) Busybox licensed to you under the GNU General Public License version 2;
(iv) Python 2.5.2 is licensed to you under the Python Software Foundation License version 2; and
(v) other embedded software or data in the hardware unit, other than files in embedded directory /real-ice and subdir
ectories, is licensed to you under the GNU General Public License version 2.

V. Jython is licensed to you under the Python Software Foundation License version 2 and portions of the code are also
subject to other terms and legal notices, including but not limited to the Apache Software License version 2.0, GNU
General Public License version 3, GNU Lesser General Public License version 3 and BSD.

W. The CoreSight Access Library is licensed to you under the Apache Licence version 2.0.

X. The Streamline Annotation Client is licensed to you under the terms of the BSD licence.

To the extent that ARM is obliged to do so, ARM hereby offers to supply the files which are subject to GNU licences (
identified above), in source code form, subject to the terms of the applicable GNU licence, upon request. This offer
is valid for three (3) years from the date of your acceptance of this Licence.

ARM Development Studio 5 v5.20
/end

Please answer with one of: 'yes' or 'no/quit'
Do you agree to the above terms and conditions? yes
```

Always enter "yes", select the DS-5 installation directory, as shown below:

```
Please answer with one of: 'yes/y' or 'no/n'
Run installation platform requirement checks? [default: yes] yes

--- Running installation platform requirement checks

Running dependency check [succeeded]

Where would you like to install to? [default: /home/parker/DS-5]
```

Waiting for the installation, enter the last "yes" after the installation is complete, as shown in the following figure:

```
Please answer with one of: 'yes/y' or 'no/n'
Install desktop menu item additions? [default: yes] yes

--- Installing menu entries

--- Skipping post install setup scripts
    You can run these later by executing ./run_post_install_for_ARM_DS-5.sh with root privileges from inside the installation.

-----
Installation completed successfully
-----

To start using ARM DS-5 either:
- Add /home/parker/DS-5/bin to your PATH
- Create a suite sub-shell using /home/parker/DS-5/bin/suite_exec <shell>
- Launch GUI tools via their desktop menu entries


The Release notes for the product can be found here: file:///home/parker/DS-5/sw/ARM_DS-5/readme.html

=====
parker@parker: /work/jtaq/DS-5/DS500-BN-00019-r5p0-20rel1$
```

Add the DS-5 environment variable, the bin folde under the DS-5 installation directory r, we need to get him to the environment variable, as shown below:

\$ vim ~/.bashrc

```
JAVA_HOME=/work/tools/jdk1.6.0_45
export ANDROID_SDK_PATH=/work/tools/adt-bundle-linux-x86_64-20140321/sdk
export JRE_HOME=/work/tools/jdk1.6.0_45/jre
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$ANDROID_SDK_PATH/platform-tools:/home/parker/DS-5/bin:$PATH
```



\$ source ~/.bashrc

9. Open Streamline debugging tools in the CT

9.1. Load drive and shell

The compiled firmware of android4.2 cubietrck in the first step, burning to the board, computer and board connected with the USB line.

The gator.ko of compiled previously and source code in the gator /data directory, pushed to the board.

```
$ adb push gator.ko /data
```

```
$ adb push gator /data
```

Enter the board file system

```
$ adb shell
```

Load gator.ko driver

```
$ insmod gator.ko
```

Running gator

```
$ chmod 777 gator
```

```
$ ./gator &
```

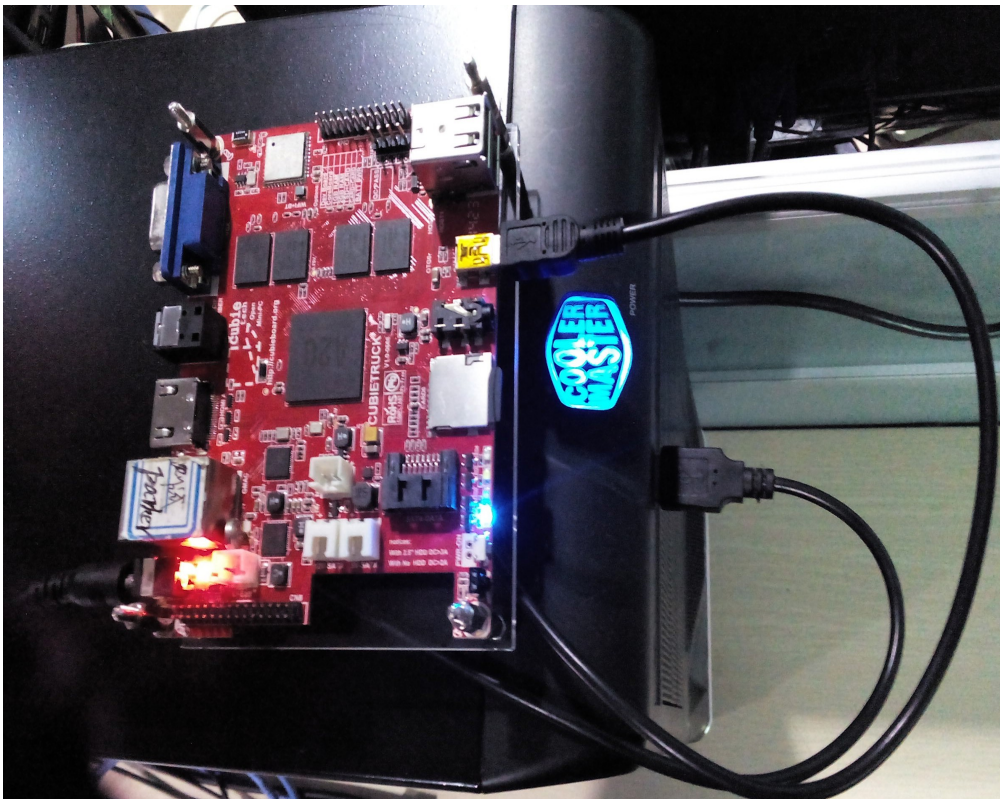
The board internal environment are good, as shown below:

```
root@android:/data # insmod gator.ko
root@android:/data # lsmod
gator 57201 0 - Live 0x00000000 (0)
cdc_ether 3163 0 - Live 0x00000000
rtl8150 9023 0 - Live 0x00000000
mcs7830 5644 0 - Live 0x00000000
qf9700 5884 0 - Live 0x00000000
asix 13586 0 - Live 0x00000000
usbnet 13741 4 cdc_ether,mcs7830,qf9700,asix, Live 0x00000000
sunxi_csi0 30818 0 - Live 0x00000000
gc2035 13734 0 - Live 0x00000000
gc0308 11800 0 - Live 0x00000000
camera 36086 1 sunxi_csi0, Live 0x00000000
videobuf_dma_contig 4157 1 sunxi_csi0, Live 0x00000000
videobuf_core 16284 2 sunxi_csi0,videobuf_dma_contig, Live 0x00000000
sun7i_ir 5797 0 - Live 0x00000000
security_system 1067129 0 - Live 0x00000000
sw_device 11512 0 - Live 0x00000000
mali 151201 31 - Live 0x00000000 (0)
hdmi 25437 0 - Live 0x00000000 (0)
lcd 5155 0 - Live 0x00000000
disp 288683 13 mali,hdmi,lcd, Live 0x00000000
nand 142727 8 - Live 0x00000000 (0)
root@android:/data # chmod +x gator
Bad mode
10|root@android:/data # chmod 777 gator
root@android:/data # ./gator &
[1] 3646
```

9.2. Uses the ADB to interactive data

Android uses mini-USB as the CT and PC data transmission media, ADB development and debugging options have been turned on by default, we just need connecte the hardware, ensure that the PC can identify to CT.

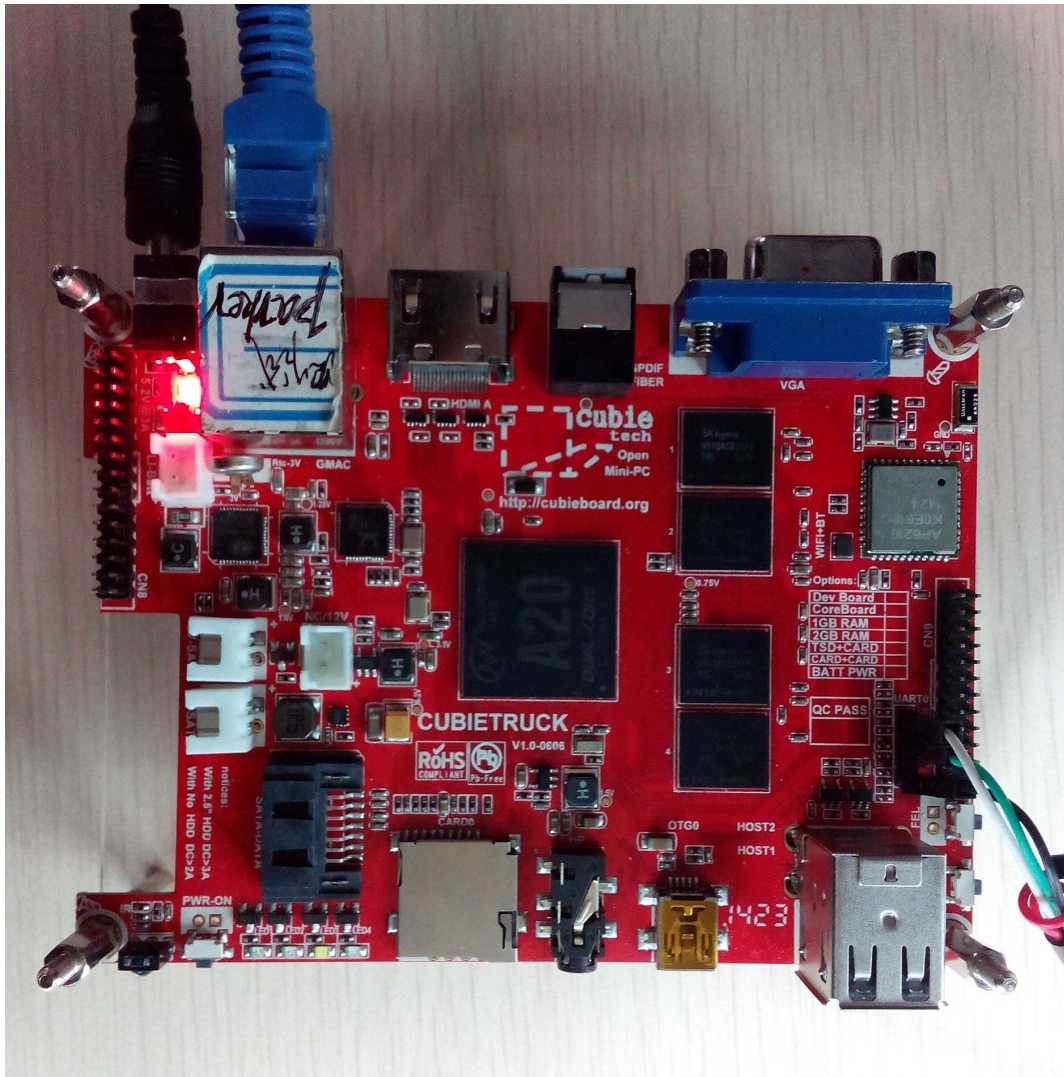
The method of connection is very simple, only needs the power on and connected to the mini-usb:



9.3. The use of interactive network data

Using mini-USB is only one way, you can be connected to the PC and CT by the network and serial debugging tools if there is no mini-USB and ADB tools, but gator.ko and gator.d will use the USB stick or TF card copy to the board of internal.

If you are using WiFi, you do not need to take the network cable, just make sure to get the IP address:



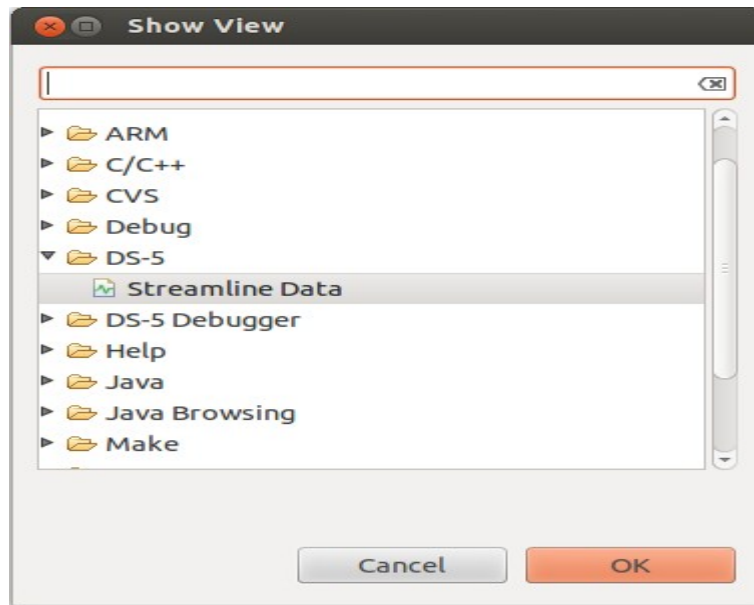
10. Use DS-5

Terminal input

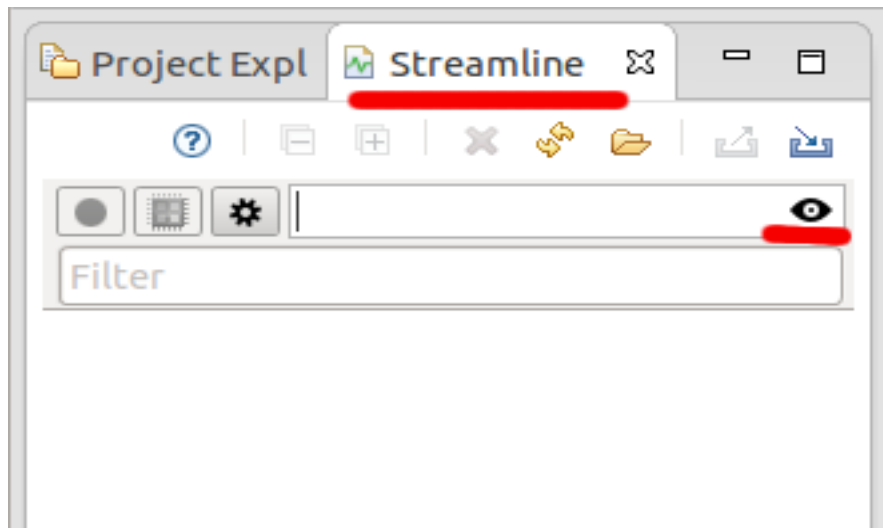
\$ eclipse

10.1. Create project of Streamline Data

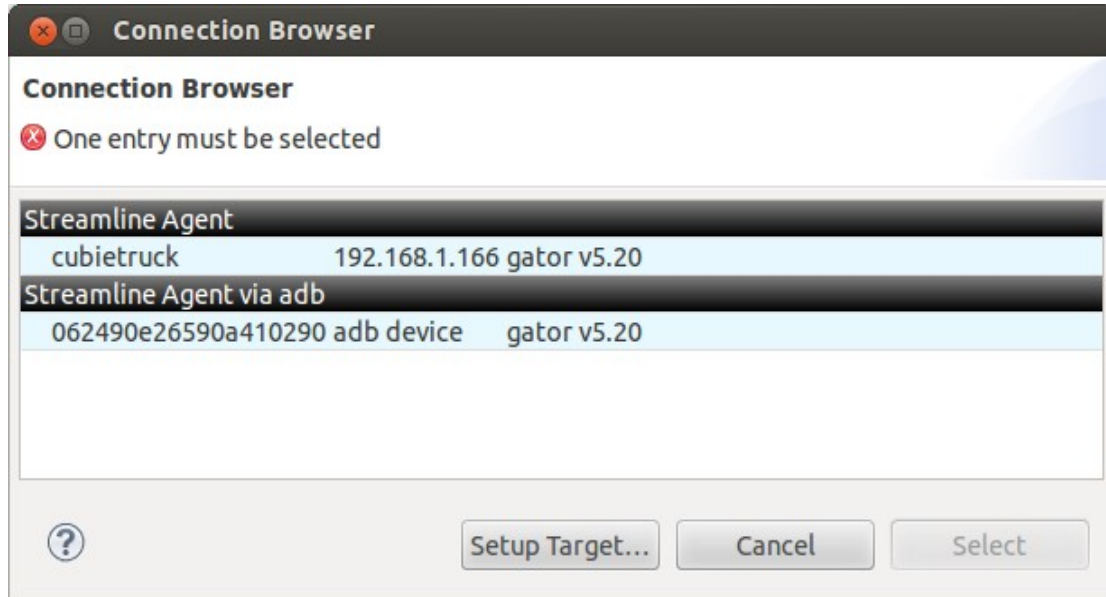
Click on the menu bar of Window > Show View > Other..., select Streamline Data, click "OK", as shown below:



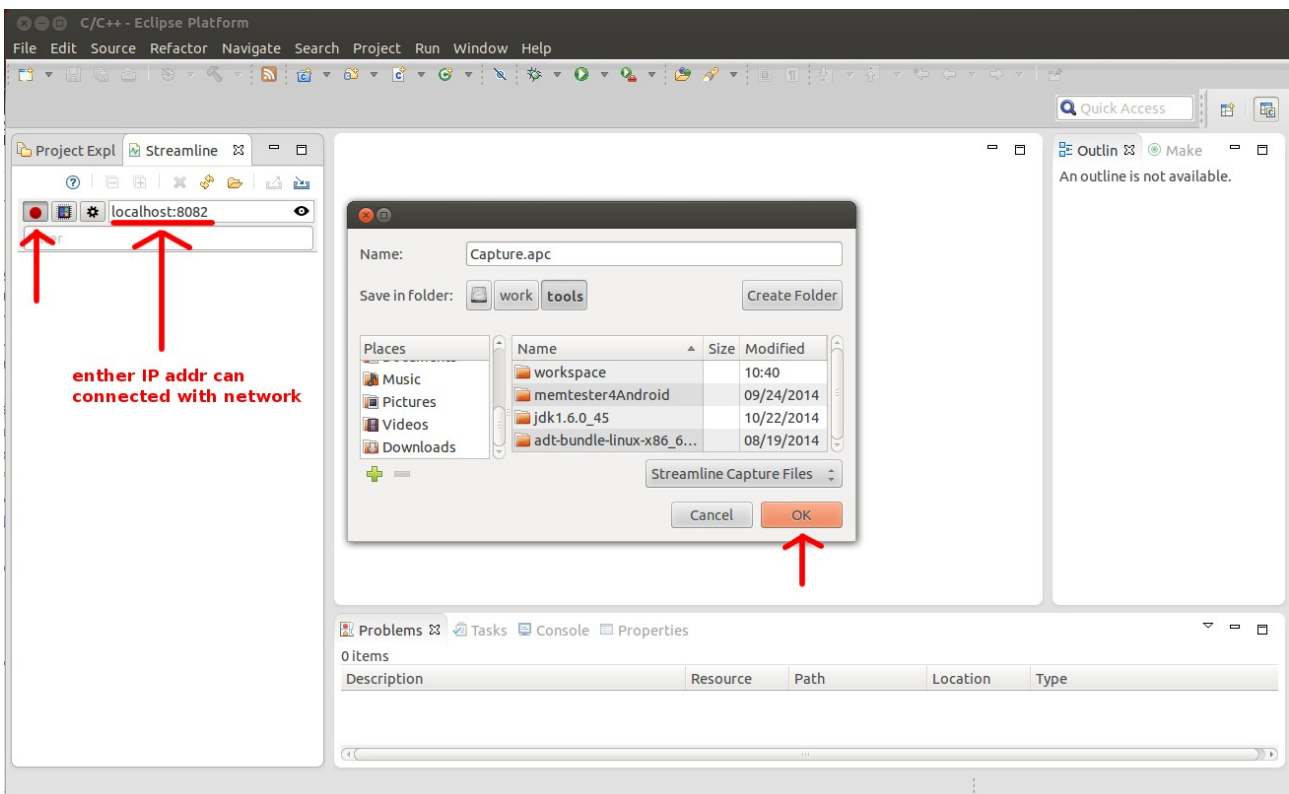
We have been set up the board environment in the sixth step, ensure that the USB line has been connected to the computer and board, select the Streamline project, and click on the eye like icons, as shown below:



Then, select the "Streamline Agent via ADB", as shown below:



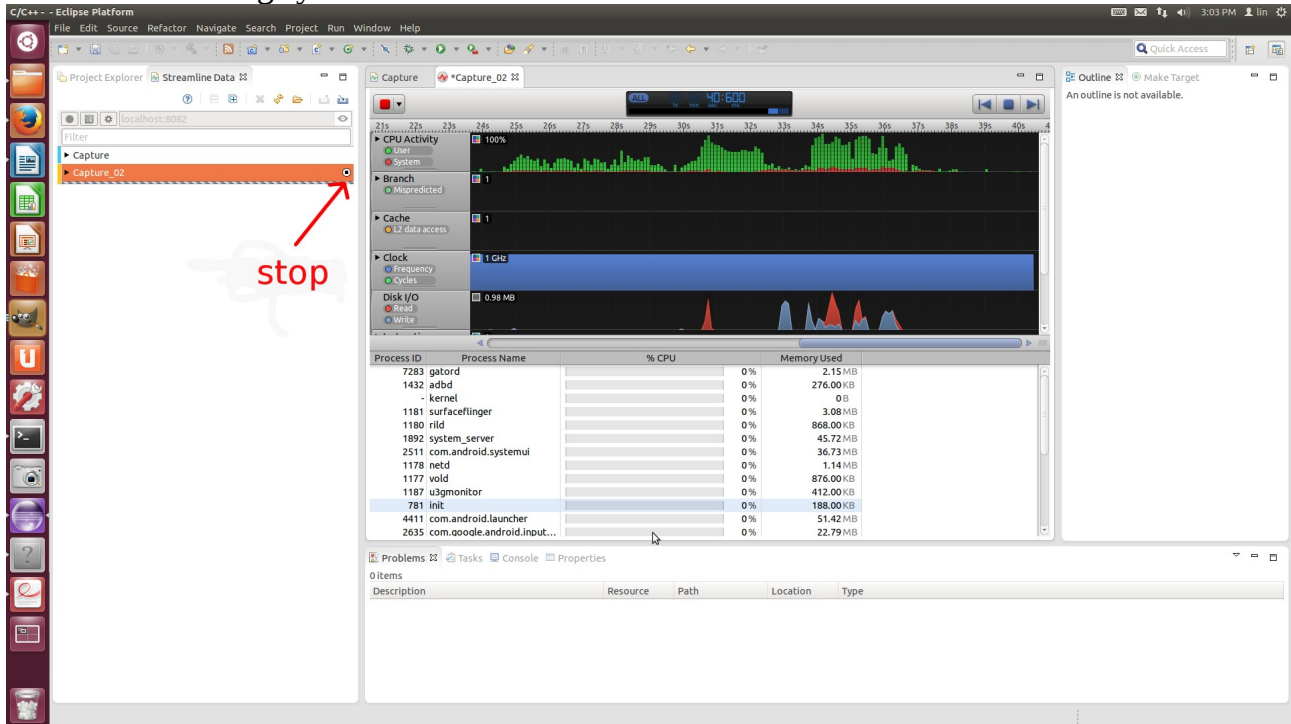
Finally, click on the red icon, select the save path, you can start debugging, as shown below:

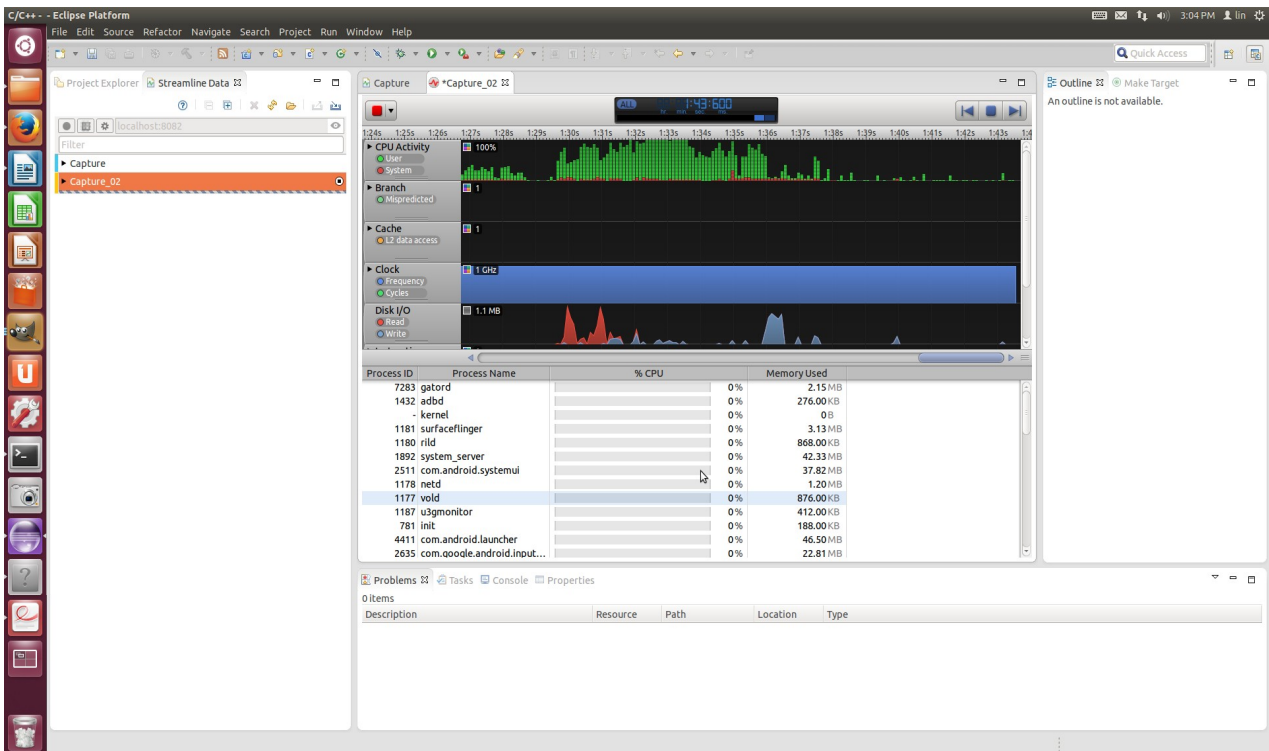


Note: the board with Linux system can not be connected ADB, as long as we replace "localhost:8082" with the board of IP address (e.g. input: 192.168.1.174) can be connected by the way of network.

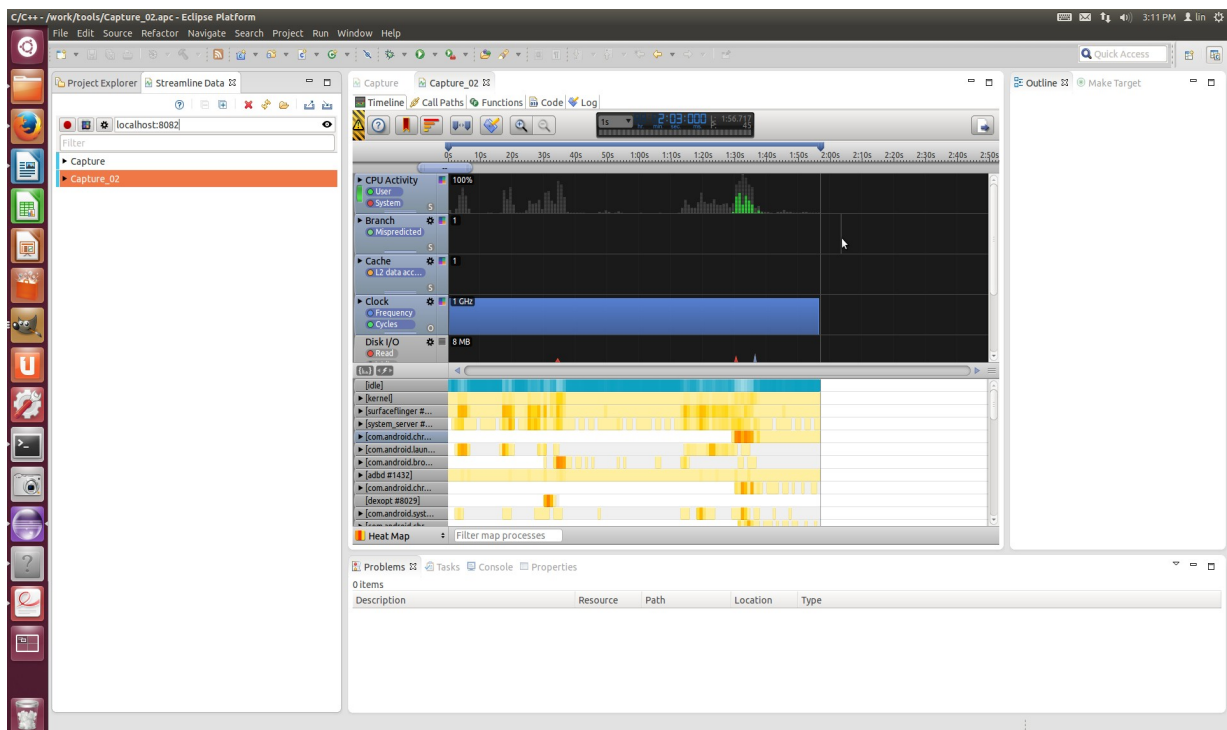
10.2. The working effect of DS-5 graph

DS-5 is monitoring system:



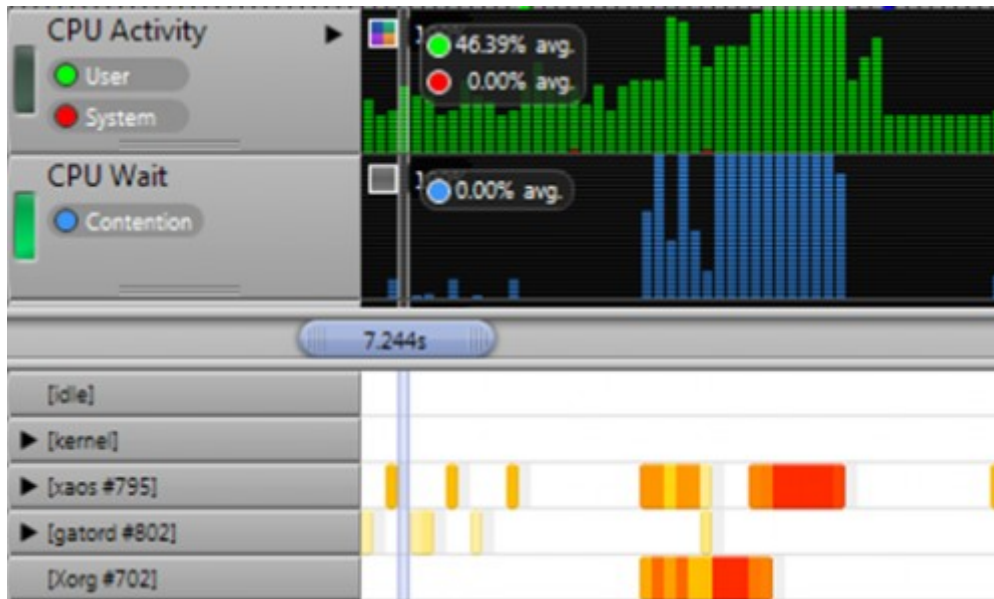


The DS-5 stops working, the derived analytical results:



10.3. Streamline simple analysis

The Streamline process can be configured to display the different information, click on activity diagram in rectangular beside GPU and CPU, we can know time that each thread in GPU and CPU latency. For example, you can see the Xorg who spent a lot of time waiting for the release of xaos processor in the example below, xaos also waiting to the end of other threads activity.



Please refer to the official ARM documentation you need detailed instructions:
<http://ds.arm.com/developer-resources/ds-5-documentation/>