



**CUBIEBOARD**  
<http://cubieboard.org>

**DS-5 debug kernel of linux and android combine DSTREAM**

ARM DS-5

Website: <http://cubieboard.org/>  
Support: [support@cubietech.com](mailto:support@cubietech.com)



Version	Author	Modification	Check
V-0.1-20150203	Payne	Init version	



## Table of Contents

1.Equipment.....	3
2.Hardware Wiring.....	3
3.Add a new chip to the DS-5 Debug device list.....	4
3.1.Explanation.....	4
3.2.Steps .....	5
3.2.1.Create DS-5 Configuration Database.....	5
4.Using DSTREAM simulator debugging Android kernel .....	8
4.1.Add the kernel options .....	8
4.2.Compile the kernel .....	10
4.3.Import the kernel source to DS - 5 .....	10
4.4.Debug configuration.....	12
4.5.Debug interface specification.....	14



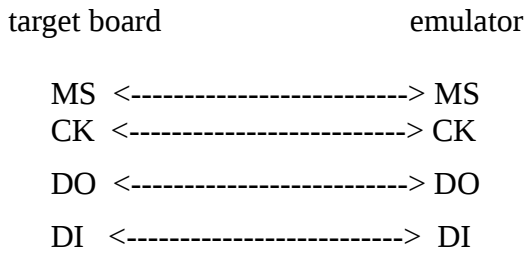
## 1. Equipment

- 1) PC x 1
- 2) DS-5 Software Development Tools x 1
- 3) DSTREAM Emulator x 1
- 4) Kernel development board(Take cubietruck for example) x 1
- 5) Other Cables

note: **The following operate in the linux operating system which is similar to the windows system, you need to pay attention to the paths.**

## 2. Hardware Wiring

Hardwired is very simple, PC machine and simulation connect with the usb cable.  
The target board and emulator just connect to the following lines except VCC and GND.



More details about DSTREAM hardware information, please visit ARM's official website:  
<http://infocenter.arm.com/help/index.jsp>

About pin figure of cubieboard hardware, please visit the official website :  
<http://cubieboard.org>

Attaching the picture, cubietruck and DSTREAM physical connection diagram:



### 3. Add a new chip to the DS-5 Debug device list

#### 3.1. Explanation

DS-5 supports all ARM processors, but most of the processors require their own database which support the target processor. All ARM target process was imported to the database were supported by DS-5. The database can set the features of the target device flexibly. Such as trace and memory-mapped registers, can reduce the additional connection steps.

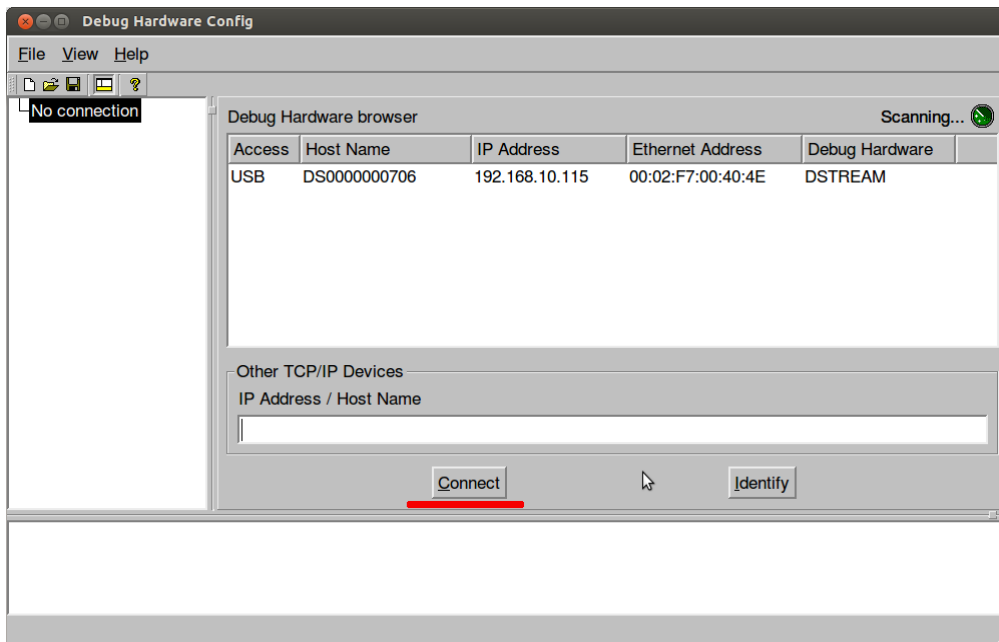
We assume installed directory on DS - 5 : /usr/local/DS-5 , If you haven't installed DS - 5, please go to the arm's official website to download: <http://ds.arm.com/downloads/>

#### 3.2. Steps

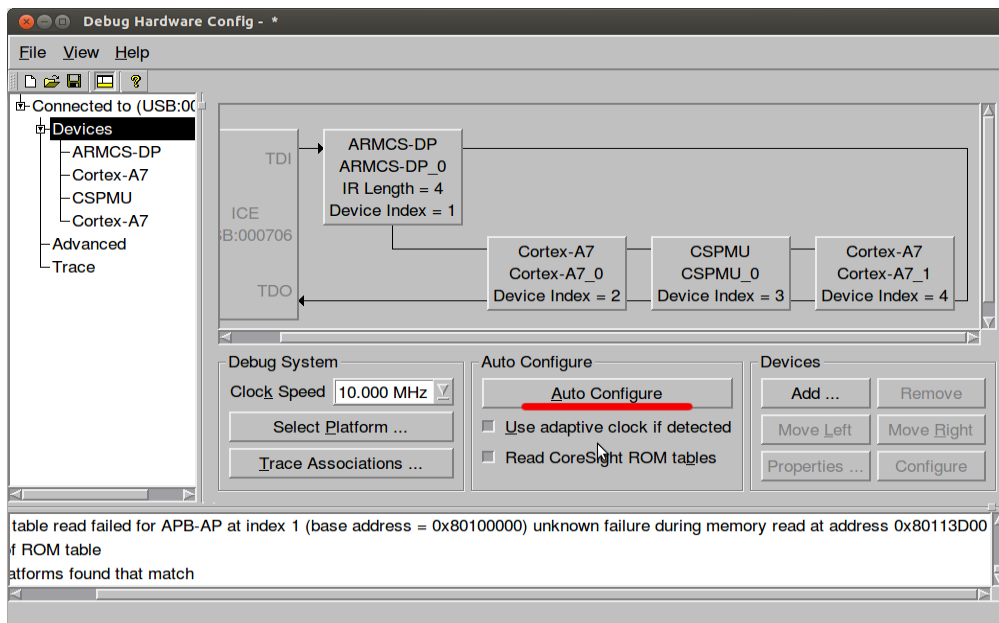
##### 3.2.1. Create DS-5 Configuration Database

1) We need to connect the hardware according to the step 2, input in the terminal

## \$ dbghwconfig



Selected device in the debug hardware browser, click "Connect", jump to the next step



Click on "Auto Configure" and will be appear a chip information (due to the hardware characteristics, please connect the cubietruck debug serial port as well, otherwise the emulator hardly read the chip information), save the configuration information then exit.

note: Here to do special configuration for different chip, be sure to use the configuration files of A20 chip which we offer, the above steps is how to generate configuration files, and only for reference.

A20 configuration file download address:

[http://dl.cubieboard.org/developers/debug-tools/ARM-DS-5/source/AW\\_A7MP2NOETM.rvc](http://dl.cubieboard.org/developers/debug-tools/ARM-DS-5/source/AW_A7MP2NOETM.rvc)

2) Generated configuration files combine dbghwconfig, run the configure database import tool, and use the appropriate parameters.

usage : **cdbimporter [-c config\_db] [-t destination\_db] rvc\_file**

config\_db: The full path of the master DS-5 configuration database

destination\_db: The full path of the new configuration database

rvc\_file: The full path which hardware configuration tool to generate the RVC

Example:

```
$ cdbimporter -c /usr/local/DS-5/sw/debugger/configdb -t /home/parker/cubie_configdb  
A20_chip.rvc
```

Select a core to modify (enter the index and hit return) or press enter to continue. [] : PRESS  
ENTER

Enter Platform Manufacturer

[default:'Imported'] > COMPANYYX

Enter Platform Name

[default:'target'] > PLATFORMX

The files of the Import tool generated which stored in the specified target database, generated in this tutorial:

**/home/parker/cubie\_configdb**

```
Reading /work/tools/DS-5-Workspace/configdb_Aw/Boards/AW/A7MP2NOETM/AW_A7MP2NOETM.rvc

Found 2 ARM cores
Import Summary -
ID  Name      Definition  Associated TCF files
--  -
1   Cortex-A7_0 Cortex-A7   <none>
3   Cortex-A7_1 Cortex-A7   <none>

Select a core to modify (enter its ID and hit return) or press enter to continue. []

Enter Platform Manufacturer
[default:'AW'] >

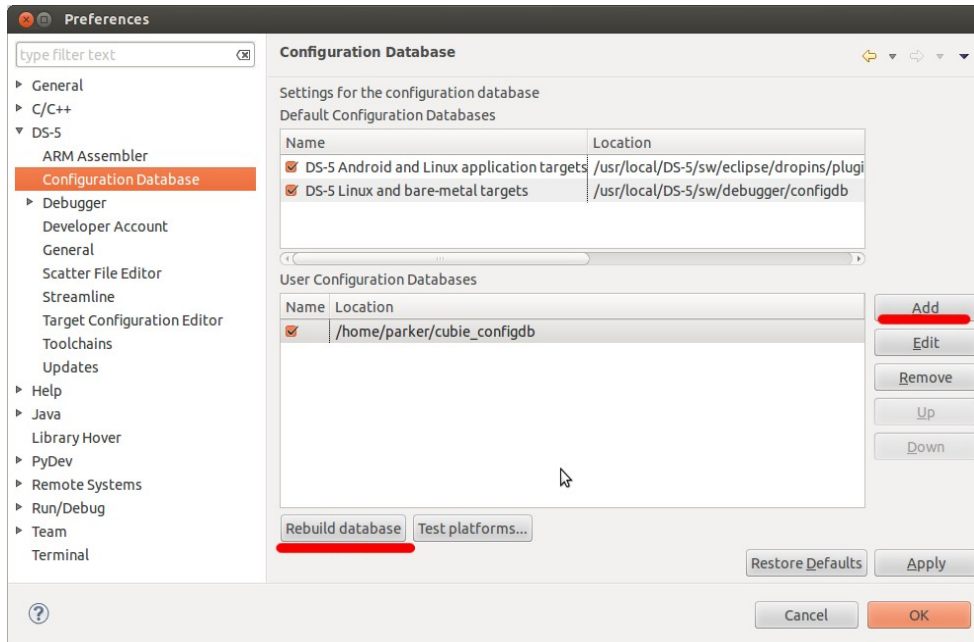
Enter Platform Name
[default:'A7MP2NOETM'] >

Building configuration XML...
Creating database entry...
Import successfully completed

The new platform will not be visible in the DS-5 Debugger until the destination database
has been added to the "User Configuration Databases" list and the database has been rebuilt.
A rebuild is done either when DS-5 is (re)started, a user configuration database is added or
by forcing a database rebuild.
To force a rebuild or add a database, select the "Window -> Preferences" menu item,
then expand the DS-5 group. To rebuild, select "Configuration Database", then press
the "Rebuild database" button.
To add a database to the "User Configuration Databases" list, click the "Add" button
and supply a suitable "Name" (E.g. Imported) and "Location" for the database.

parker@parker: /work/tools/DS-5-Workspace$
```

3) Start DS - 5, open the "Preferences" option under "Window" menu, a DS - 5 options, select "Configuration Database". Click "Add" button to add new configuration database.



As the same interface with the above , click "Rebuild the Database", to ensure that the new target Database load,then click "OK" to exit.

## 4. Using DSTREAM simulator debugging Android kernel

### 4.1. Add the kernel options

Description: We take the android kernel for example, the operation method of the Linux kernel are similar, need to pay attention to the path and the method of compilation.

Select a cubieboard kernel source code, execute “make ARCH = arm menuconfig ”.



Choose “kernel hacking”>“Kernel debugging” option ,symbol is “DEBUG\_KERNEL”.

```
+-----+
[ ] Show timing information on printks
(4) Default message log level (1-7)
[*] Enable __deprecated logic
[*] Enable __must_check logic
(1024) warn for stack frames larger than (needs gcc 4.4)
[ ] Magic SysRq key
[*] Strip assembler-generated symbols during link
[ ] Generate readable assembler code
[ ] Enable unused/obsolete exported symbols
- *- Debug Filesystem
[ ] Run 'make headers_check' when building vmlinux
[ ] Enable full section mismatch analysis
- *- Kernel debugging
[ ] Debug shared IRQ handlers
[ ] Detect Hard and Soft Lockups
[ ] Panic on oops
^(-)
+-----+
```

Choose “kernel hacking”>“compile the kernel with debug info” option, symbol is“DEBUG\_INFO”.

```
+-----+
^(-)
[ ] Mutex debugging: basic checks
[ ] Lock debugging: detect incorrect freeing of live locks
[ ] Lock debugging: prove locking correctness
[ ] RCU debugging: sparse-based checks for pointer usage
[ ] Lock usage statistics
[ ] Sleep inside atomic section checking
[ ] Locking API boot-time self-tests
[ ] Stack utilization instrumentation
[ ] kobject debugging
[*] Verbose BUG() reporting (adds 70K)
- *- Compile the kernel with debug info
[ ] Reduce debugging information (NEW)
[ ] Debug VM
[ ] Debug filesystem writers count
[*] Debug memory initialisation
[ ] Debug linked list manipulation
^(-)
+-----+
```

After Configuratte,press two times, prompt exit select < Yes > save and exit.

```
+-----+
Do you wish to save your new configuration ? <ESC><ESC>
to continue.
< Yes > < No >
+-----+
```

## 4.2. Compile the kernel

After modify the kernel configuration, we need to recompile the kernel and generate vmlinux, and use to this file later.

Compile methods don't elaborate on here, please read:

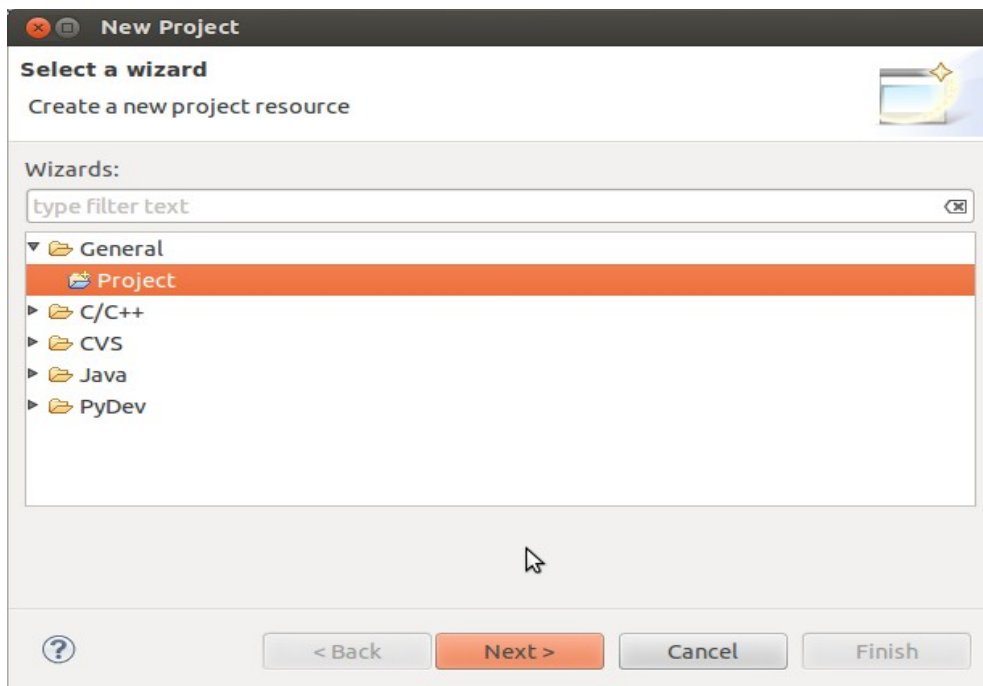
android: <http://pan.baidu.com/s/1dDF5cVR>

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

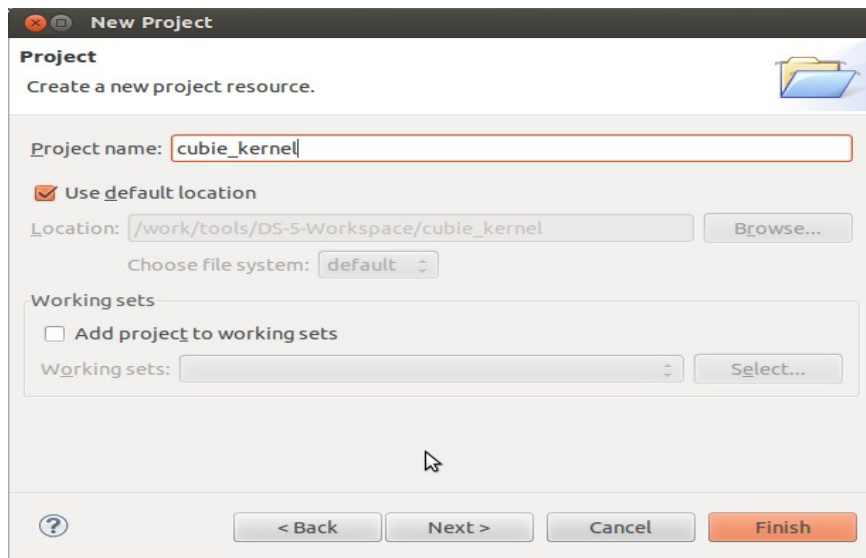
## 4.3. Import the kernel source to DS - 5

Create a new project in DS - 5, named "MYD - SAMA5D3X\_kernel" ,import the kernel source code.

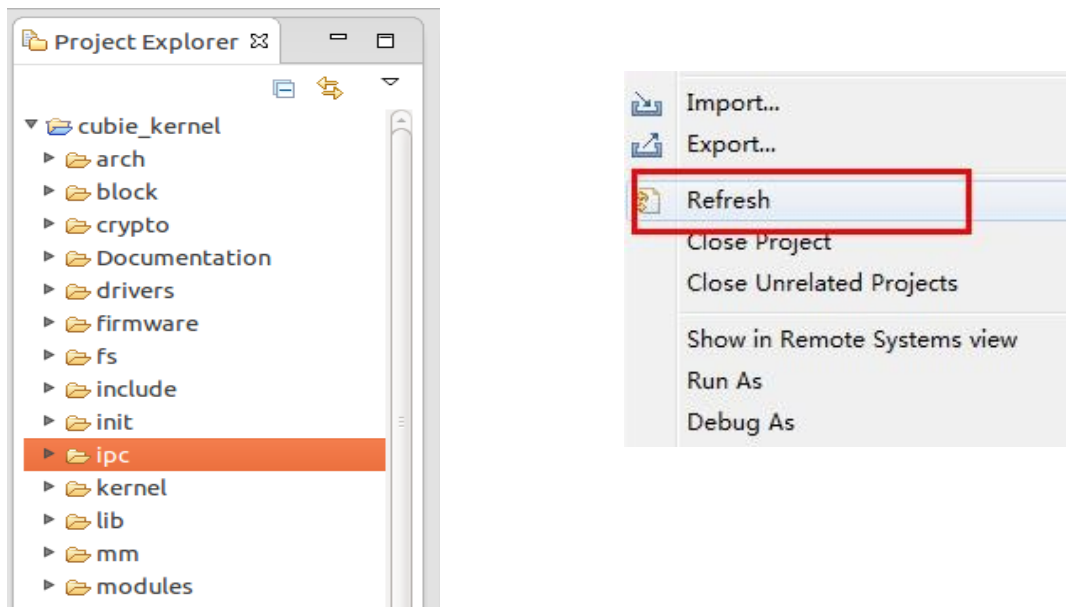
Open DS-5,choose "File" > "New" > "Project..." .



Input the project name in the “project name”, be named "cubie\_kernel", then click "finish" to complete the project creation.

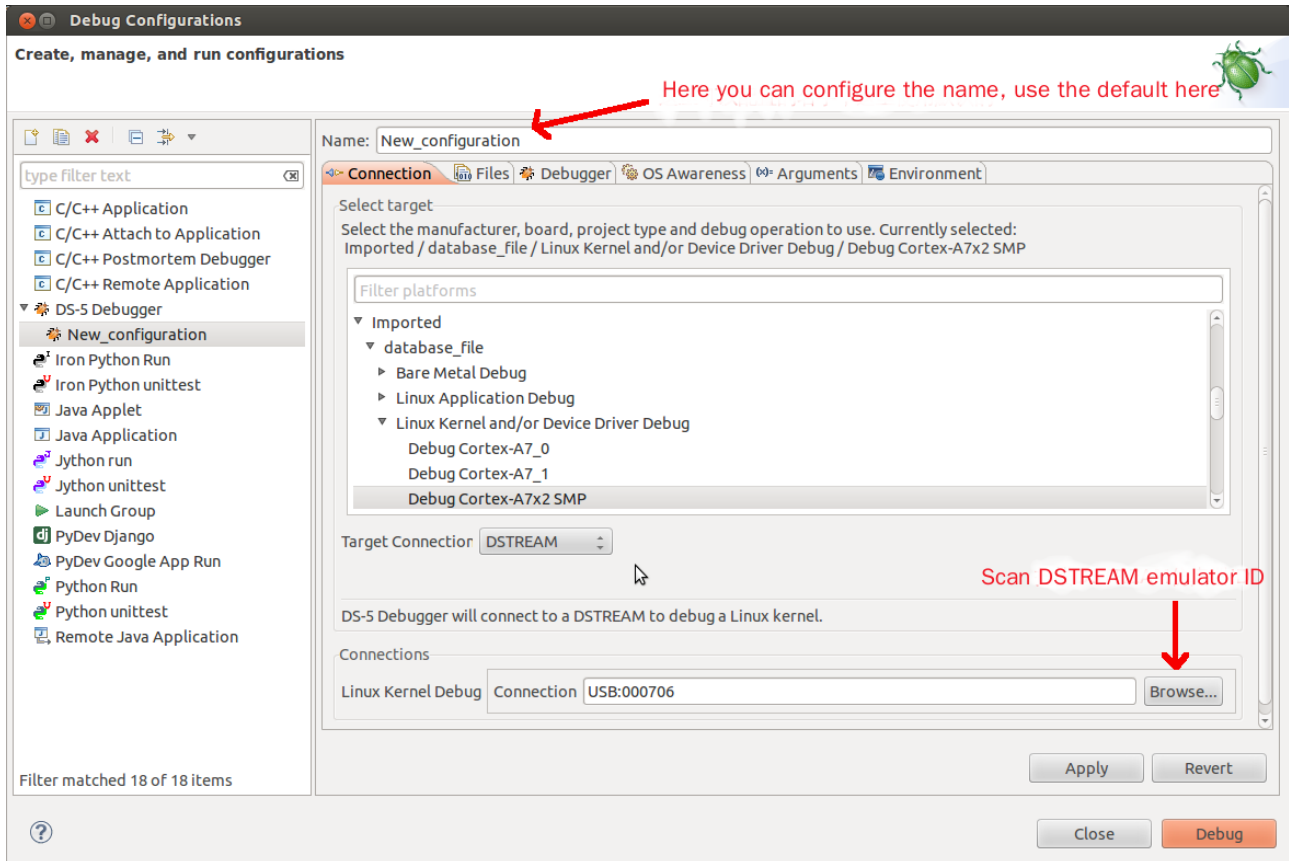


Copy all the contents of compiled to the project directory(example:cp -a /lichee/linux-3.4/\* /work/tools/DS-5-Workspace/cubie\_kernel),then right-click project name, select "Refresh" to Refresh. The add file of DS - 5 will be displayed.



## 4.4. Debug configuration

Open the menu bar "Run" > "the Debug Configurations..." , "DS - 5 Debugger" selected "New\_configuration". Choose "Select target" > > "Imported" > "database\_file" "Linux Kernel and/or Device Driver Debug" > "Debug Cortex-A7x2 SMP"(red font is the database directory name for you create, please find according to actual situation ).Target "connections" click "browse..." Select search to DSTREAM simulator.

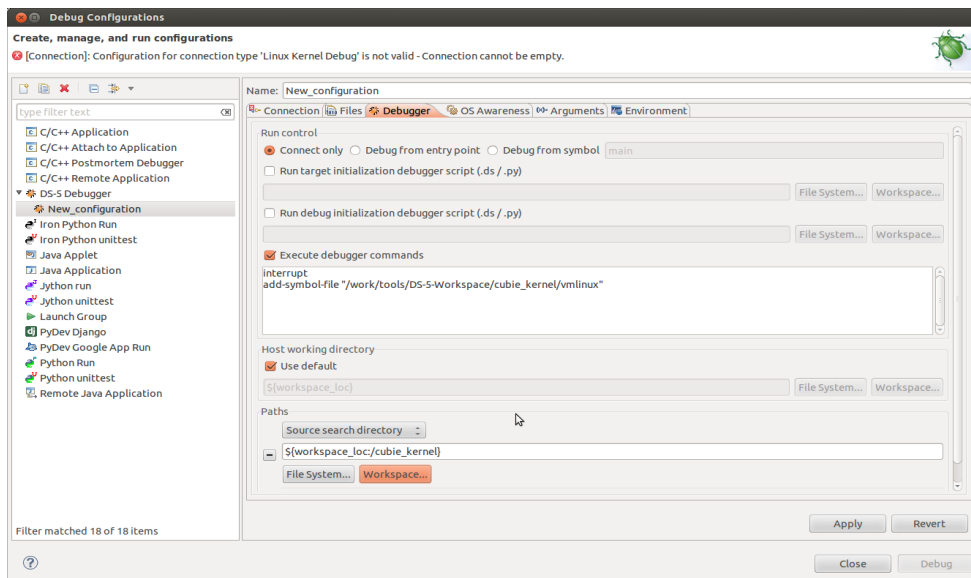


Configure "Debugger" option below ,operation control "Run control" select "connect only" .  
Click on the "Execute the debugger commands" and input in the input box:

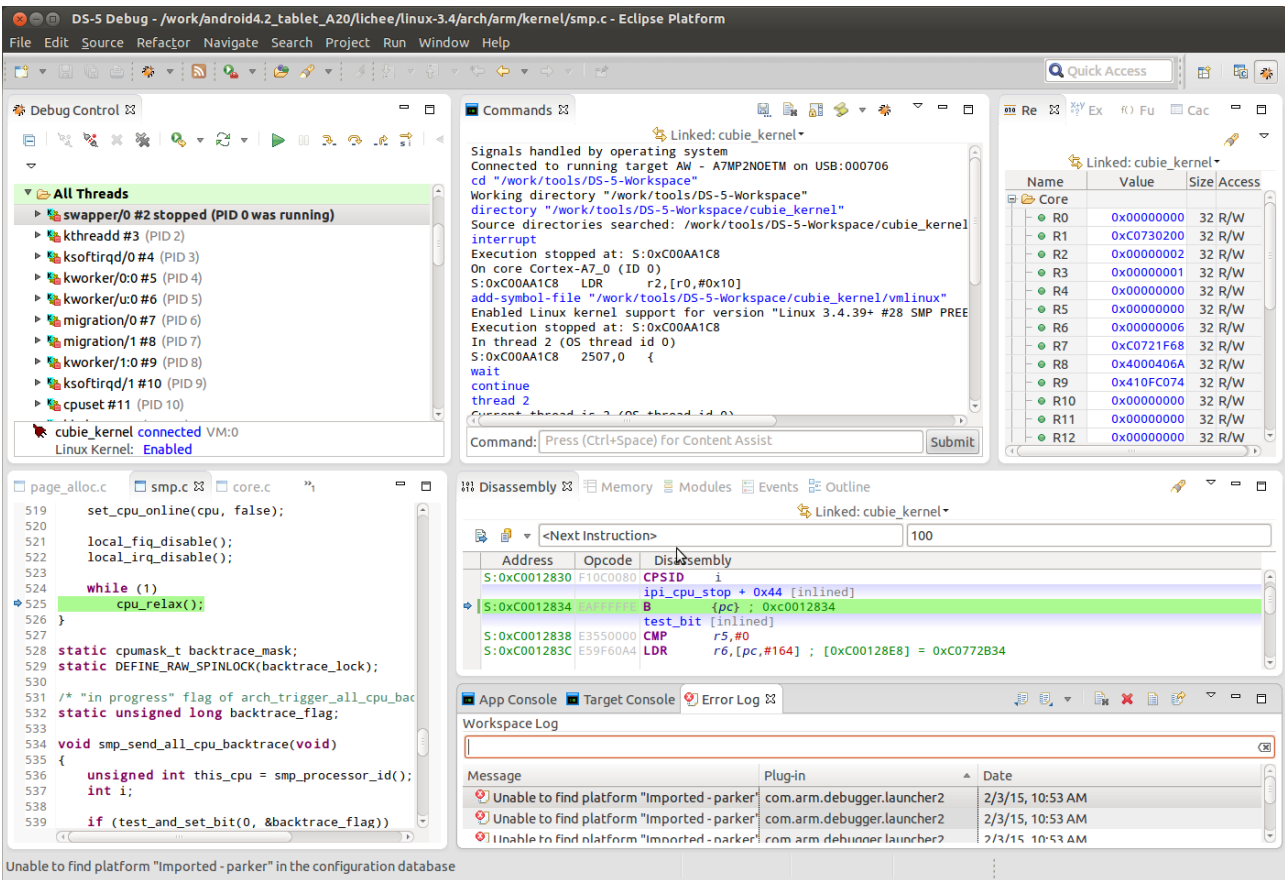
```
interrupt
```

```
add-symbol-file "/work/tools/DS-5-Workspace/cubie_kernel/vmlinux"
```

Click on the "Workspace..." button below "Paths" ,choose "cubie\_kernel" project as DS - 5 source search path.Open the cubietruck power (or reset), let u-boot guide the kernel, and then click the "Debug" button at the DS - 5 to start debugging.

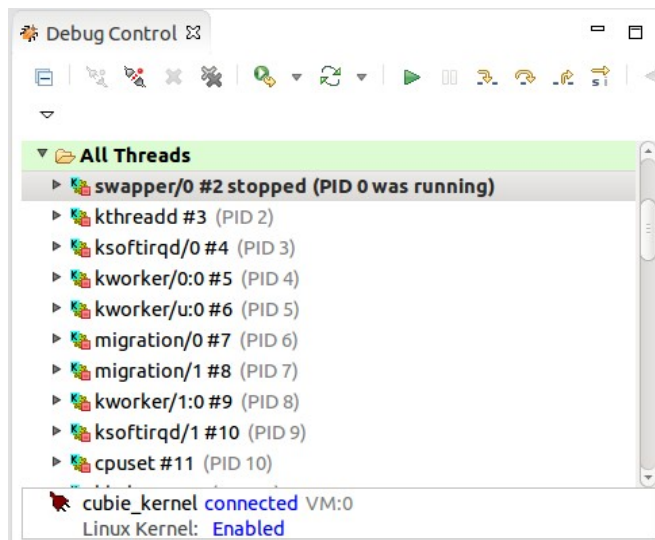


Finally, we will see an interface in the following . Represents the target plate and the simulator has been successfully connected, and you can start debugging.











## 4.5. Debug interface specification

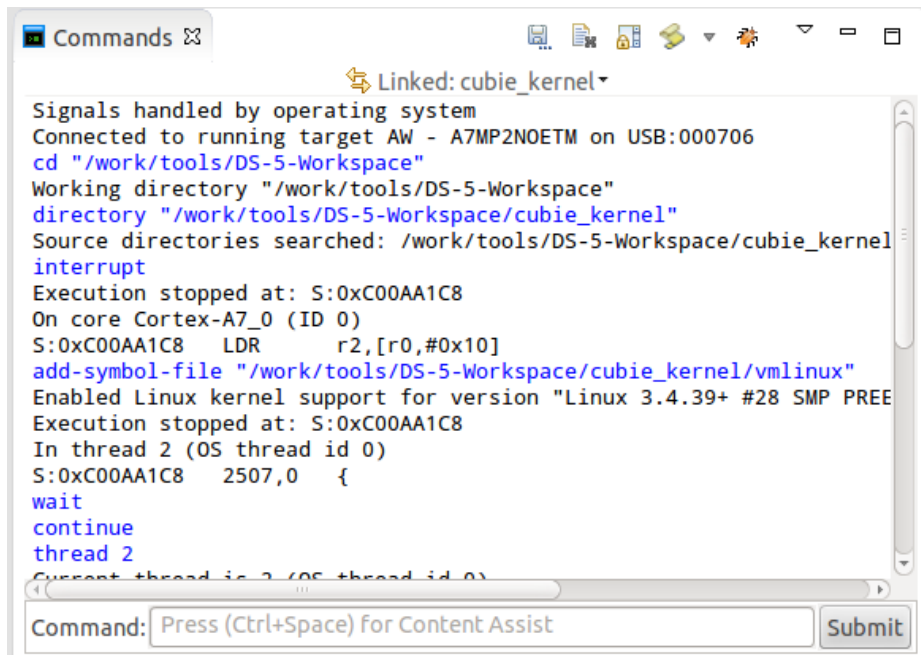
The DS - 5 started to connect the development board, the view is as follows, all show the current name of debugging, and it can debug control.



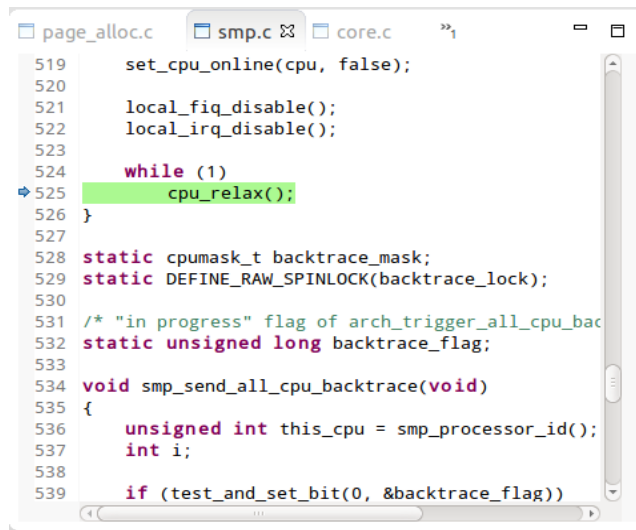
In the figure above, the function of each control button as follows:

-  Connect plate
-  Disconnect
-  Delete the connection
-  From the main function or entry point debugging
-  Continue to run at full speed
-  Stop running
-  Single step debugging
-  Choose according to the C program on the basis of single step debugging or assembler debugging

Command bar which can input command after “commands” and make development board running, such as input "step" would doing single step debugging . The mouse is located in the input box, press "Alt + /" can obtain the command prompt.



Assembler bar, display program corresponding the assembler, address and the operands, etc.

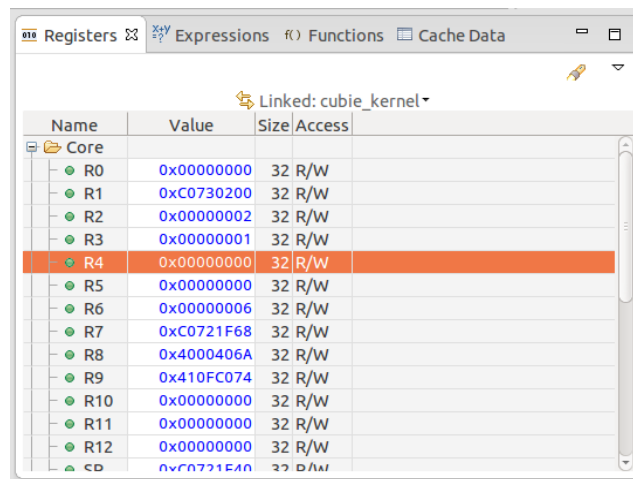


```

519     set_cpu_online(cpu, false);
520
521     local_fiq_disable();
522     local_irq_disable();
523
524     while (1)
525     cpu_relax();
526 }
527
528 static cpumask_t backtrace_mask;
529 static DEFINE_RAW_SPINLOCK(backtrace_lock);
530
531 /* "in progress" flag of arch_trigger_all_cpu_bac
532 static unsigned long backtrace_flag;
533
534 void smp_send_all_cpu_backtrace(void)
535 {
536     unsigned int this_cpu = smp_processor_id();
537     int i;
538
539     if (test_and_set_bit(0, &backtrace_flag))

```

Register bar, shows all registers, the inside of the kernel, can be modify register when debugging



Name	Value	Size	Access
Core			
R0	0x00000000	32	R/W
R1	0xC0730200	32	R/W
R2	0x00000002	32	R/W
R3	0x00000001	32	R/W
R4	0x00000000	32	R/W
R5	0x00000000	32	R/W
R6	0x00000006	32	R/W
R7	0xC0721F68	32	R/W
R8	0x4000406A	32	R/W
R9	0x410FC074	32	R/W
R10	0x00000000	32	R/W
R11	0x00000000	32	R/W
R12	0x00000000	32	R/W
SP	0xC0721E40	32	R/W

About more detailed content, please refer to the document of the arm's official website:

<http://infocenter.arm.com/help/index.jsp>