DS-5 Debug Linux and Android Kernel by DSTREAM

1. Recompile Kernel

The MYD-SAMA5D3X development board has not enabled the Linux/Android kernel debug information by default. So please follow below steps to enable it.

1) Please copy the Linux kernel package “linux-at91.tar.bz2” under “05-Android_Source\Linux-at91” in MYD-SAMA5D3X product DVD to Linux Host;
Note: the Linux kernel package might not be in the same folder, so please be adjusted to the actual DVD.

2) Please switch to the folder which “linux-at91.tar.bz2” is located and input below command to decompress the kernel package:

```
# tar xvjf linux-at91.tar.bz2
```

3) Please get into the decompressed folder using below command:

```
# cd linux-at91
```

4) Please configure the environment variables using below command:

```
# export ARCH=arm
```

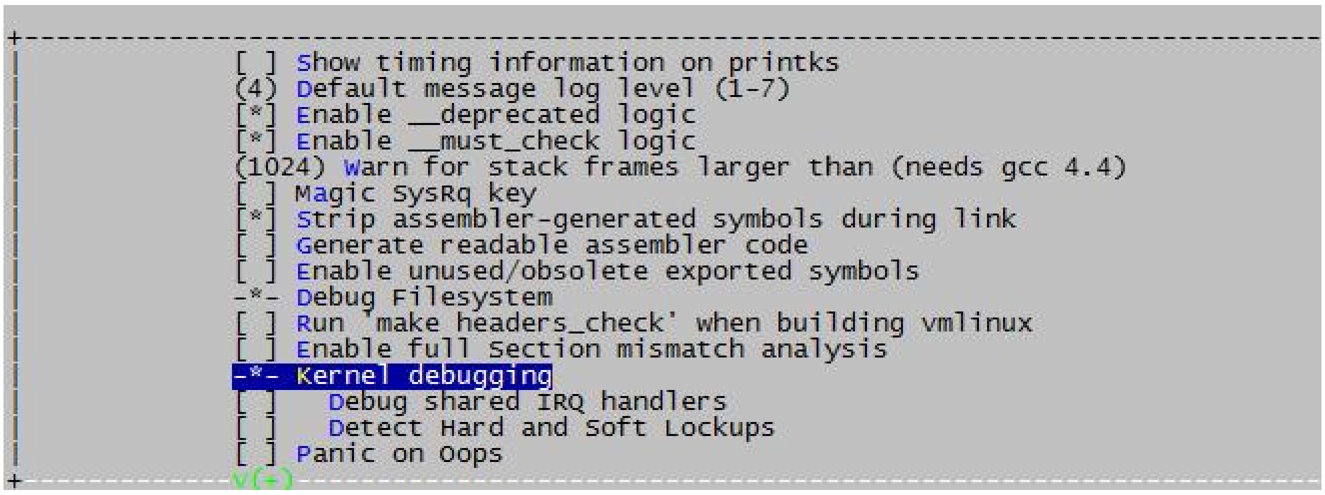
5) Compile the configuration file of the development board:

```
# make sama5d3_android_defconfig
```

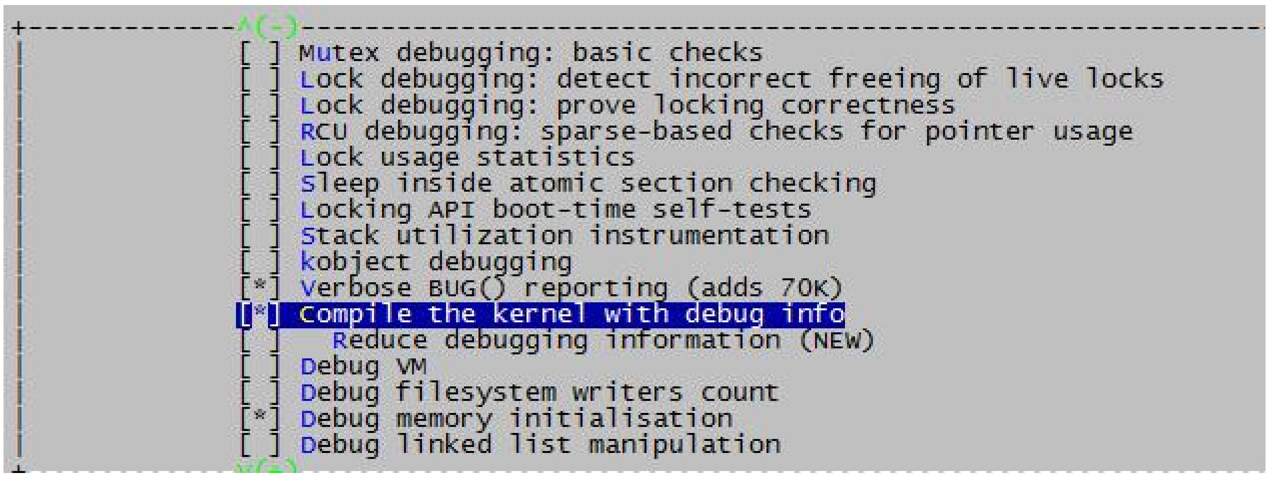
6) Configure the kernel by invoking graphics menu using below command:

```
# make menuconfig
```

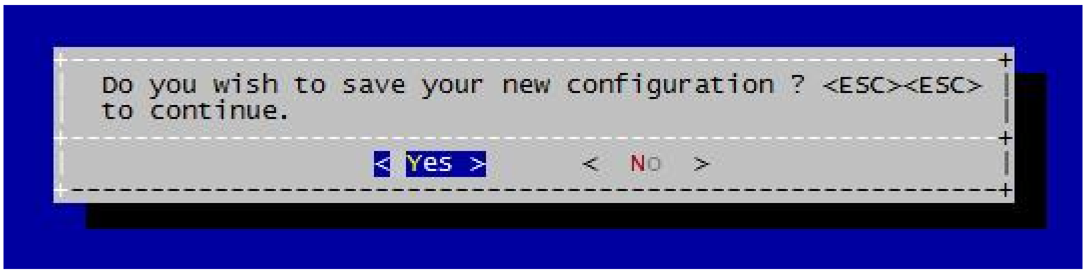
7) Select “kernel hacking”>“Kernel debugging”, its symbol is “DEBUG_KERNEL” and it is enabled by default.


```
+--------------------------------------------------------------------------------
|              [ ] 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
+--------------v(+)--------------------------------------------------------------
```

8) Select “kernel hacking”>“compile the kernel with debug info”, its 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
+--------------v(+)--------------------------------------------------------------
```

9) After configuration, press the button twice, it will remind you selecting <Yes> to save and exit.


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

10) Configure cross-compiling tool using below command:

```
# export
CROSS_COMPILE=/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.04-20130415_linux/bin/arm-linux-gnueabihf-
```

Note: Please refer to MYD-SAMA5D3X development board user manual to build the cross-compiling tool. The folder might have difference, please be adjusted to actual conditions.

11) Compile kernel using below command:

```
# make
```

12) Create Linux image file using below command:

```
# make uImage
```

13) There will be two files created which we shall use for debug kernel.

-- /arch/arm/boot/uImage

14) It is the image file specially used for U-boot. It has added a tag with 0x40 length before the zImage. We need then download it to the target board.

-- /vmlinux

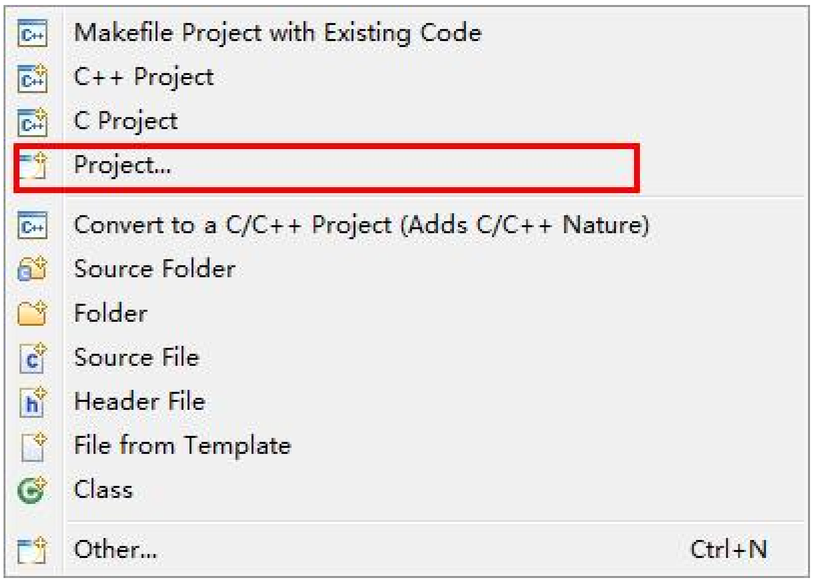
15) The compiled uncompressed file is the original kernel file. The following DS-5 debugging will use this file.

16) Copy the uImage file above created to the Image folder and download it to the development board following the user manual.

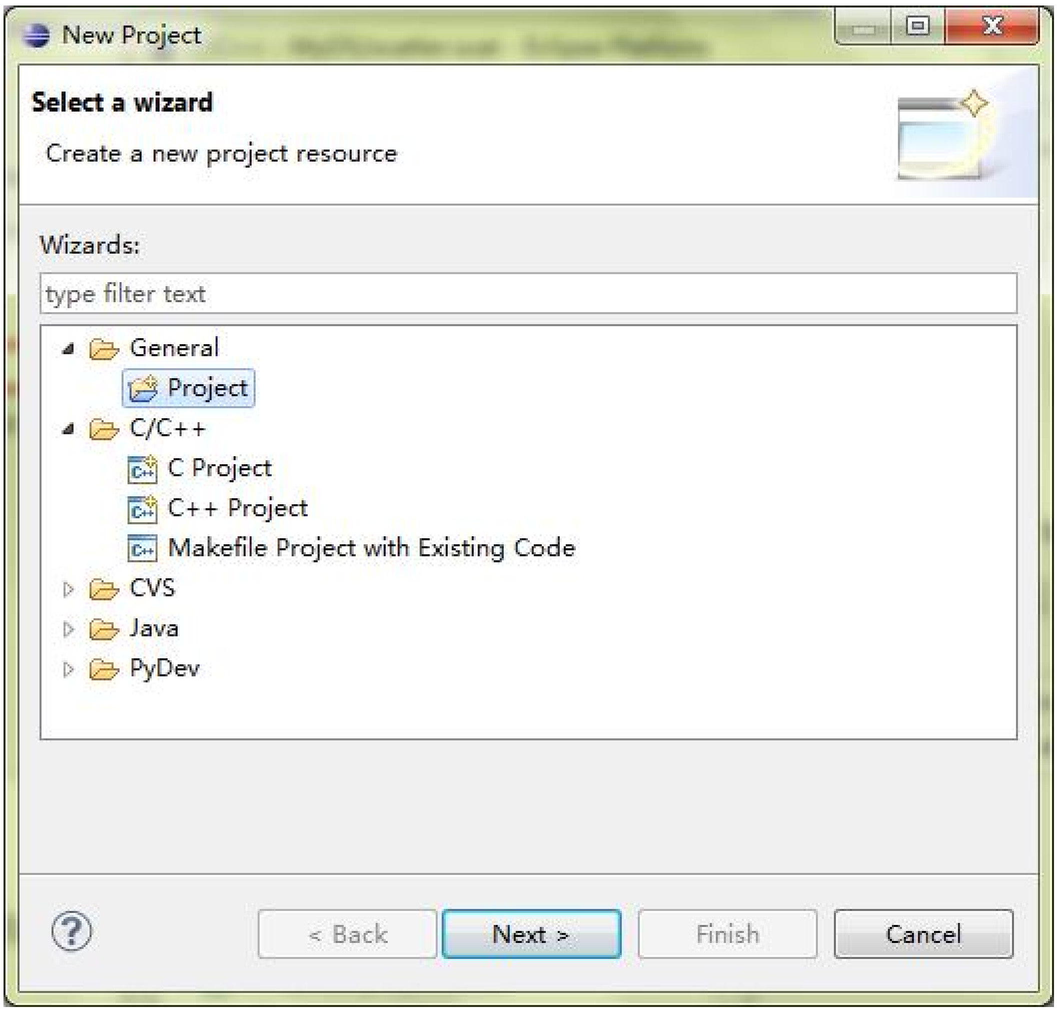
2. Import Linux/Android Source Code

Now we will build a new project named “MYD-SAMA5D3X_kernel” in DS-5 and import kernel source code.

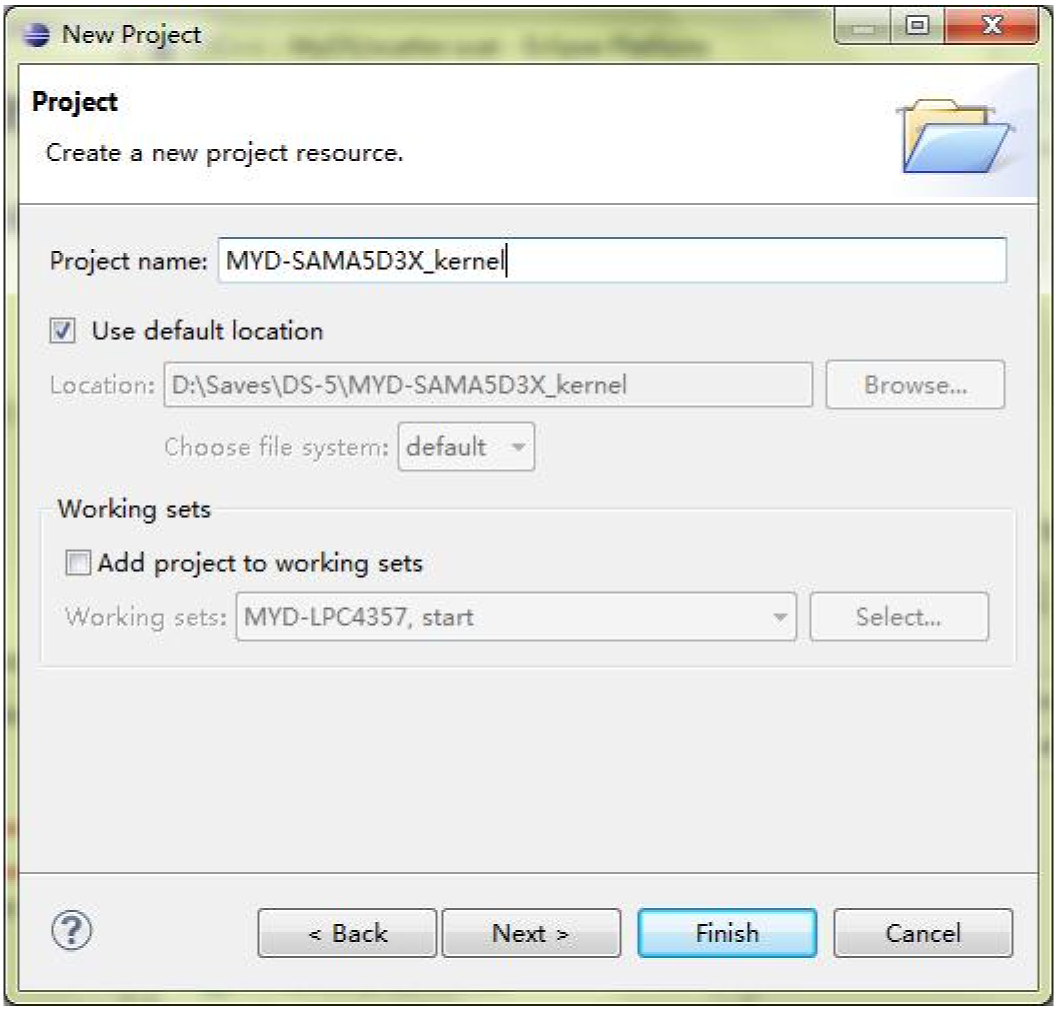
17) Open DS-5 and select “File”>“Project...”



18) Select “General”>“Project”



19) Input project name, here we use “MYD-SAMA5D3X_kernel”, click “finish” to complete the creation.


New Project

Project

Create a new project resource.

Project name: MYD-SAMA5D3X_kernel

☑ Use default location

Location: D:\Saves\DS-5\MYD-SAMA5D3X_kernel

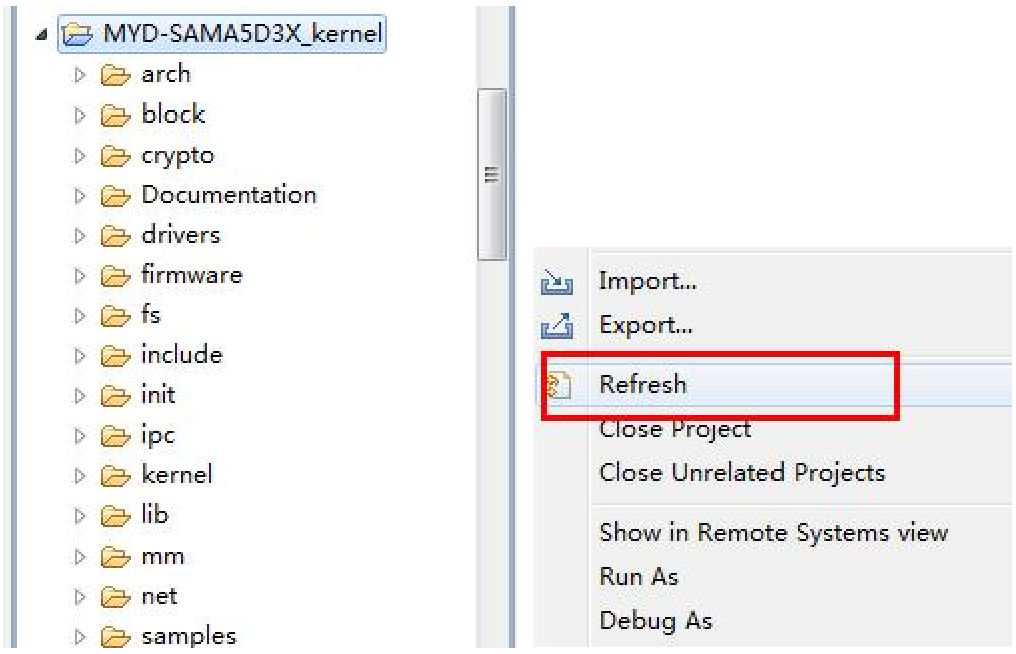
Choose file system: default

Working sets

☐ Add project to working sets

Working sets: MYD-LPC4357, start

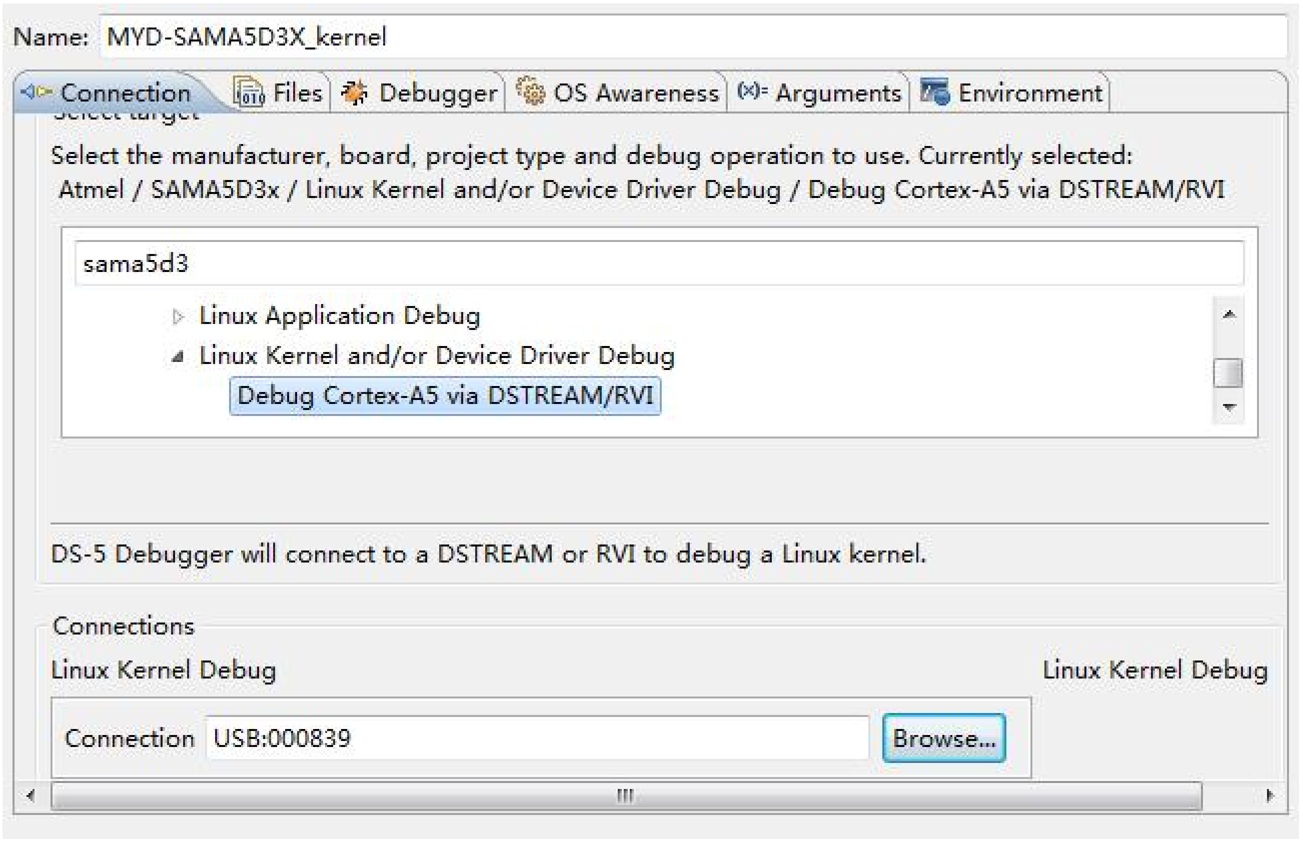
20) Open the MYD-SAMA5D3X product DVD and decompress the “05-Android_Source\Linux-at91\ linux-at91.tar.bz2” in linux-at91 folder to the “MYD-SAMA5D3X_kernel” project in DS-5. Right-click the project and select “Refresh”, then the DS-5 will show the added file.



21) Copy the created kernel image file “vmlinux” to the project folder “MYD-SAMA5D3X_kernel” and refresh.

3. Debug and Configuration

22) Please connect the ARM JTAG 20 of the DSTREAM to the JTAG (J23) interface of the MYD-SAMA5D34 development board. And also well connect the DSTREAM and development board power.



Name: MYD-SAMA5D3X_kernel

Connection | Files | Debugger | OS Awareness | Arguments | Environment

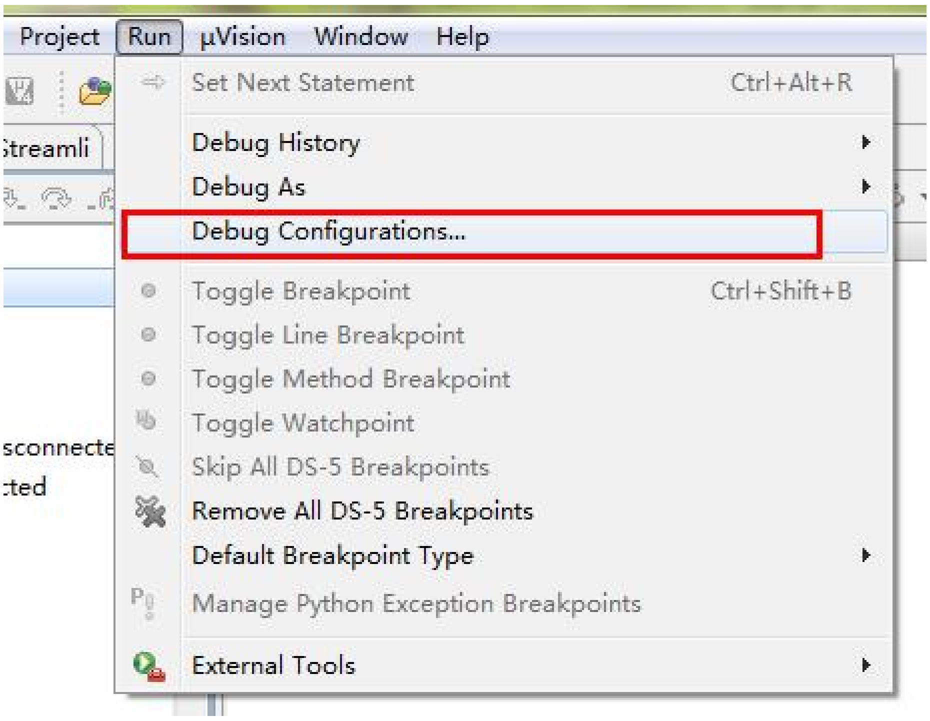
Select the manufacturer, board, project type and debug operation to use. Currently selected:
Atmel / SAMA5D3x / Linux Kernel and/or Device Driver Debug / Debug Cortex-A5 via DSTREAM/RVI

sama5d3
Linux Application Debug
Linux Kernel and/or Device Driver Debug
Debug Cortex-A5 via DSTREAM/RVI

DS-5 Debugger will connect to a DSTREAM or RVI to debug a Linux kernel.

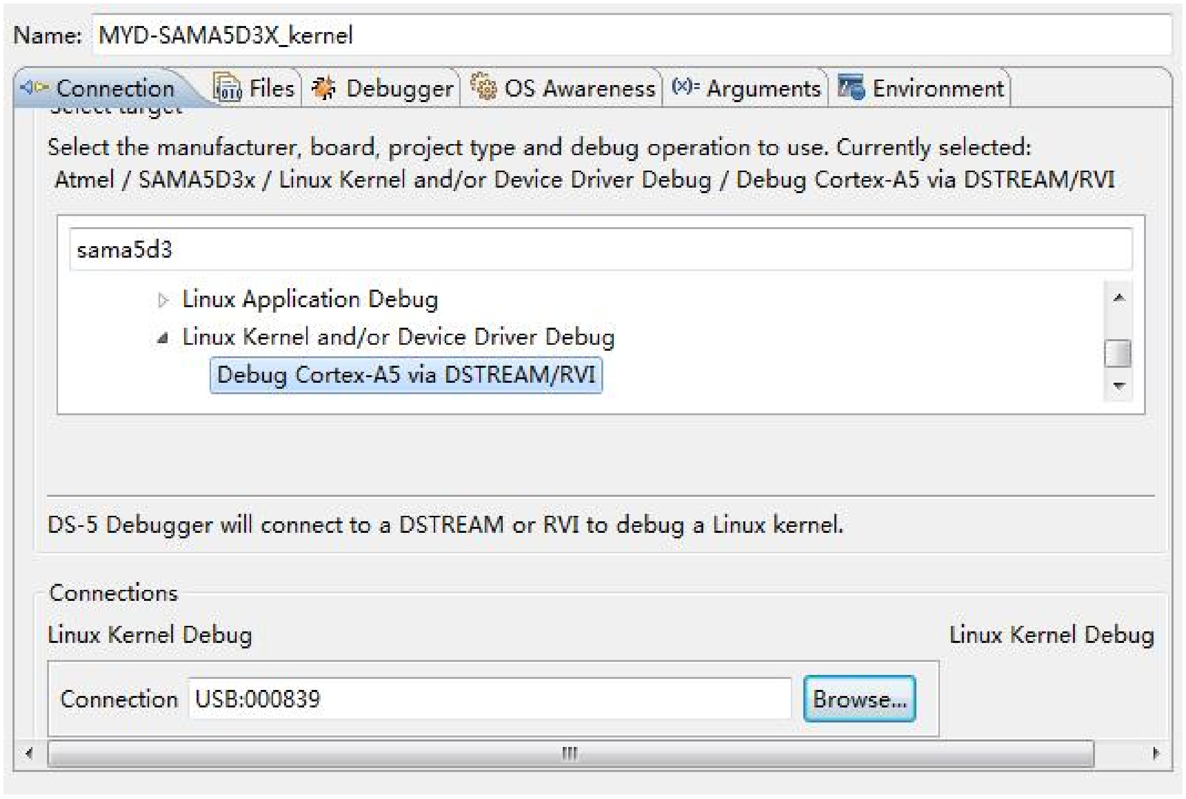
Connections
Linux Kernel Debug　Linux Kernel Debug
Connection USB:000839 Browse...

23) Select “run”> “debug configuration…” in DS-5 and make your configurations.



24) In the configuration,

Input “MYD-SAMA5D3X_kernel” in “name” as connection name;
Select “Atmel”>“SAMA5D3x”>“Linux Kernel and/or Device Driver Debug” under “target”;
Link to “connections” and click “browse…” to select searched DSTREAM emulator.



Name: MYD-SAMA5D3X_kernel

Select the manufacturer, board, project type and debug operation to use. Currently selected:
Atmel / SAMA5D3x / Linux Kernel and/or Device Driver Debug / Debug Cortex-A5 via DSTREAM/RVI

sama5d3

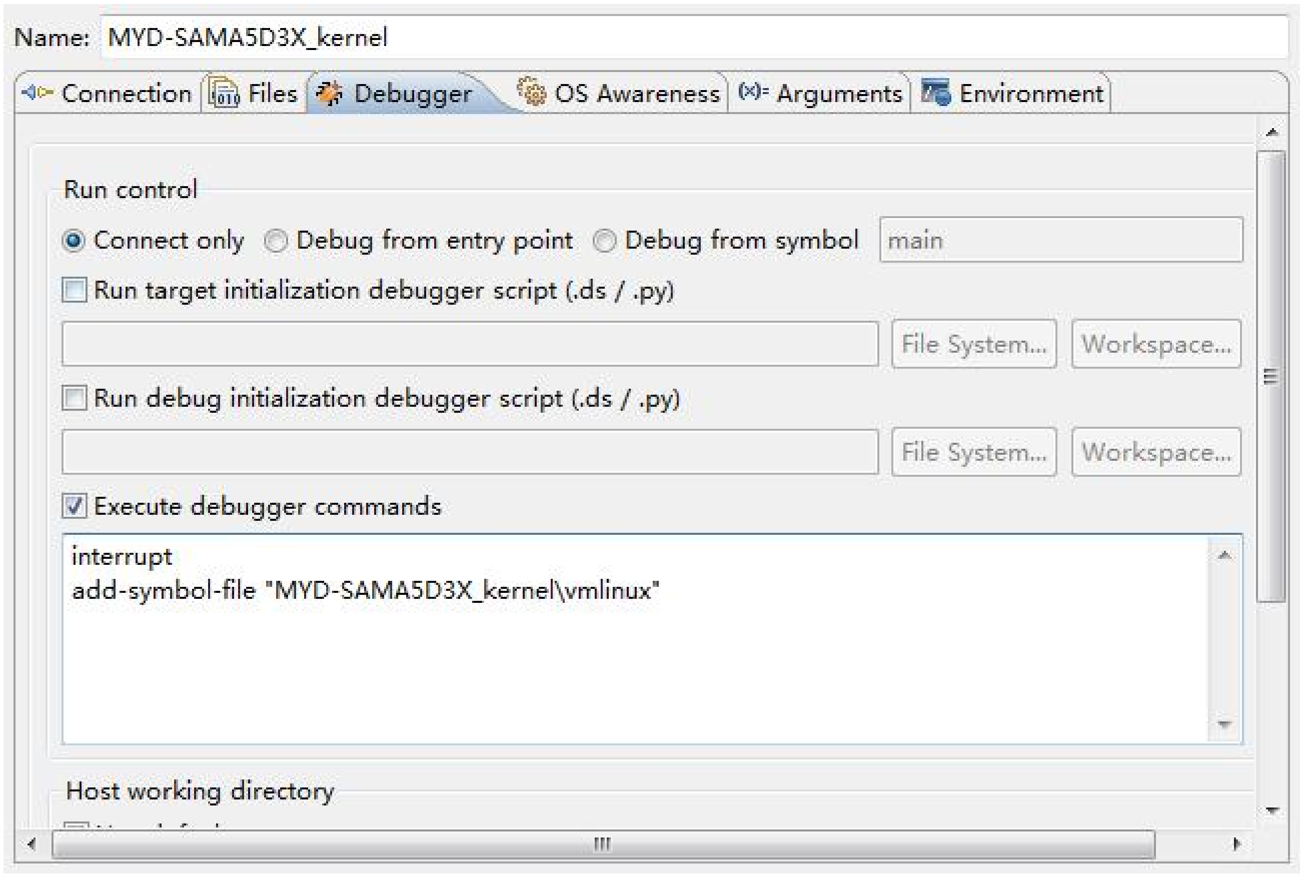
DS-5 Debugger will connect to a DSTREAM or RVI to debug a Linux kernel.

Connections
Linux Kernel Debug
Connection USB:000839

25) Please leave blank for Files option. And configure the Debugger as below:
Select “connect only” for “run control”.
Click “execute debugger commands” and input below command:

```
interrupt

add-symbol-file "MYD-SAMA5D3X_kernel\vmlinux"
```



Name: MYD-SAMA5D3X_kernel

Run control

[✓ Connect only] Debug from entry point Debug from symbol main

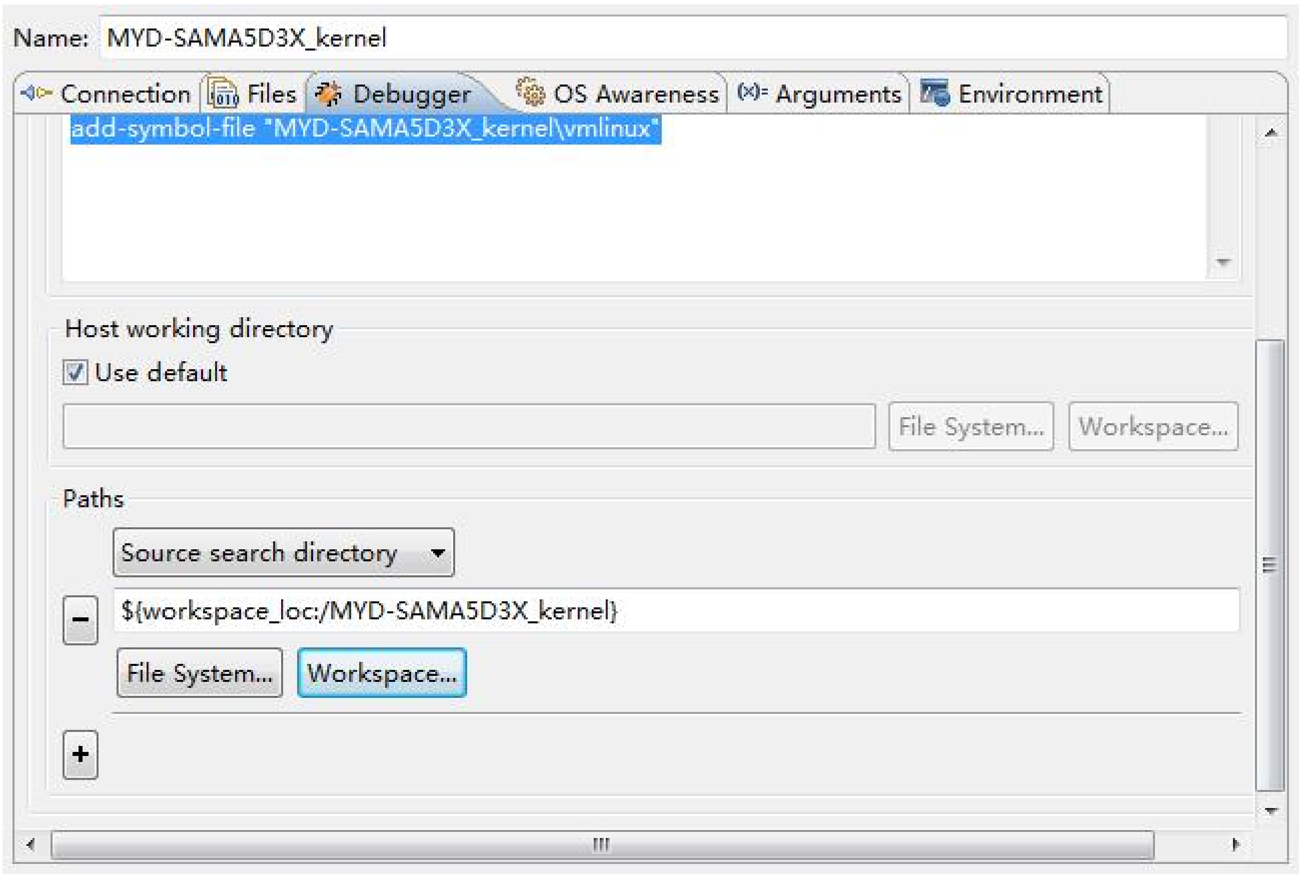
☐ Run target initialization debugger script (.ds / .py)

☐ Run debug initialization debugger script (.ds / .py)

☑ Execute debugger commands

```
interrupt
add-symbol-file "MYD-SAMA5D3X_kernel\vmlinux"
```

26) Drop down the “debugger” option, click “workspace” button under “paths” and select “MYD-SAMA5D3X_kernel” project as the search path of the DS-5 source code.


Name: MYD-SAMA5D3X_kernel

```
add-symbol-file "MYD-SAMA5D3X_kernel\vmlinux"
```

Host working directory

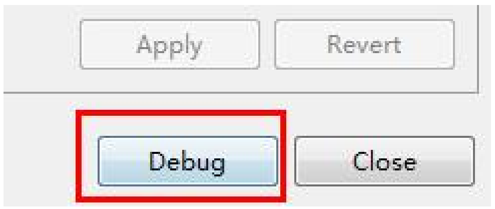
☑ Use default

Paths

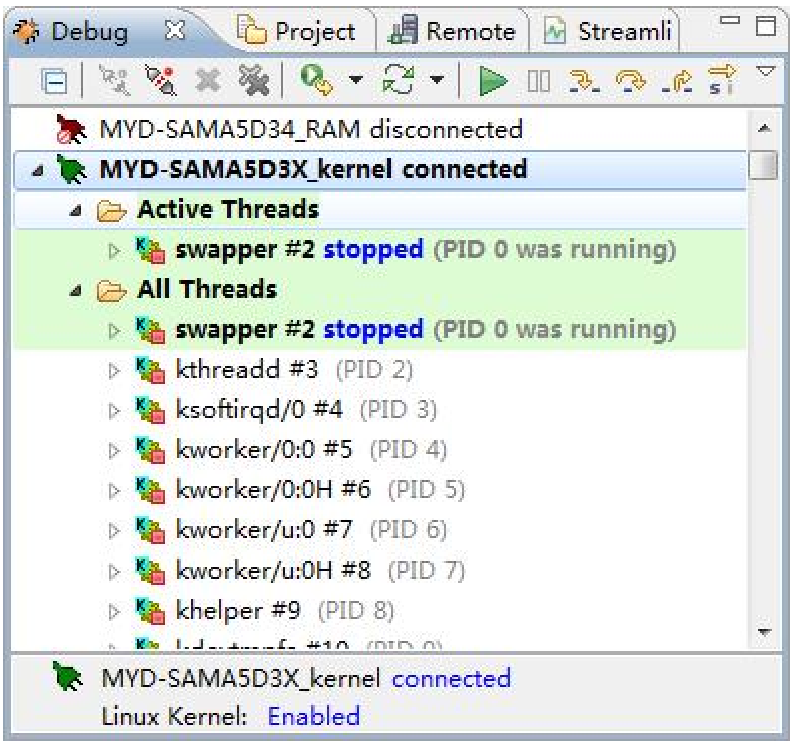
Source search directory

${workspace_loc:/MYD-SAMA5D3X_kernel}

27) Power on the development board (or reset) and use the u-boot to lead kernel, then click “debug” button in DS-5 to start debugging.



28) So the DS-5 will connect with the development board. Please refer to below images which have shown all debugging and can be controlled.



MYD-SAMA5D34_RAM disconnected
MYD-SAMA5D3X_kernel connected
Active Threads
swapper #2 stopped (PID 0 was running)
All Threads
swapper #2 stopped (PID 0 was running)
kthreadd #3 (PID 2)
ksoftirqd/0 #4 (PID 3)
kworker/0:0 #5 (PID 4)
kworker/0:0H #6 (PID 5)
kworker/u:0 #7 (PID 6)
kworker/u:0H #8 (PID 7)
khelper #9 (PID 8)

MYD-SAMA5D3X_kernel connected
Linux Kernel: Enabled

The purposes of the control buttons showed in above image are as below:

Connect target board

Disconnect

Delete connection

Debug from the main function or entry point

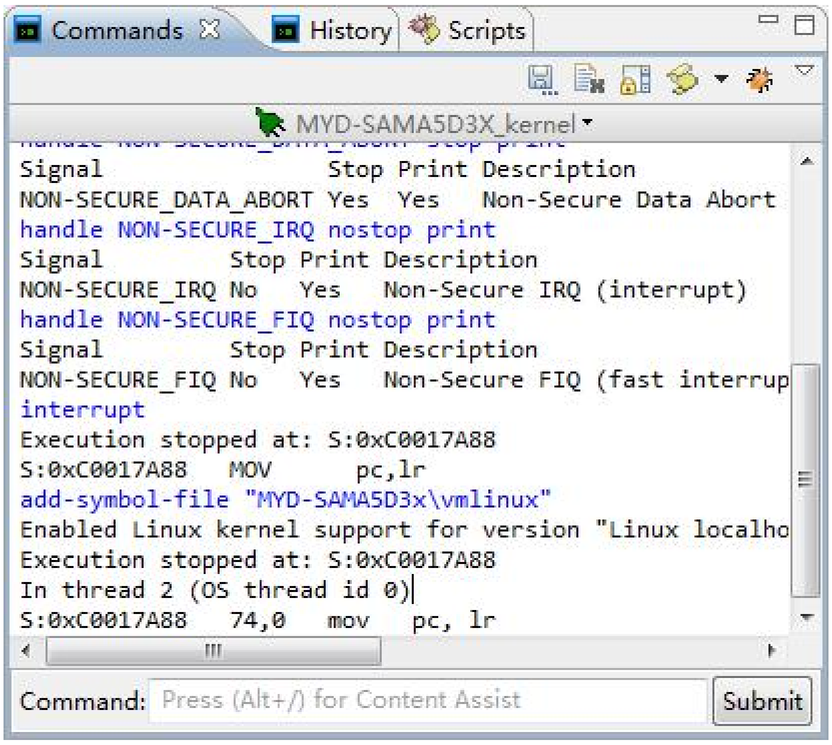
Continue to run at full speed

Stop running

Single-step debugging

Select C program for single-step debugging or use assembly program for debugging

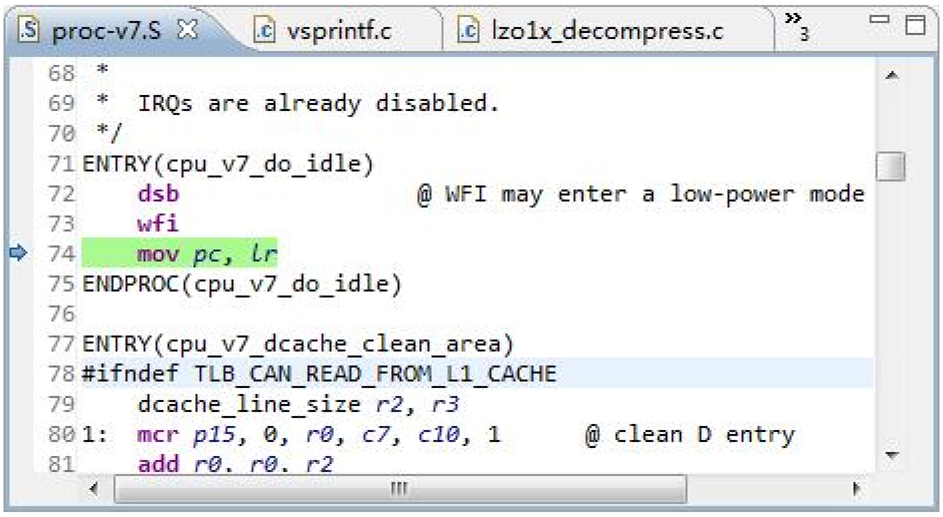
29) Commands column
User can input commands after “commands” and let the development execute. For example, if you input “step”, it will start single-step debugging. The mouse is in input box and press “Alt+/” can get command prompt.



```
Signal                  Stop Print Description
NON-SECURE_DATA_ABORT Yes  Yes   Non-Secure Data Abort
handle NON-SECURE_IRQ nostop print
Signal         Stop Print Description
NON-SECURE_IRQ No   Yes   Non-Secure IRQ (interrupt)
handle NON-SECURE_FIQ nostop print
Signal         Stop Print Description
NON-SECURE_FIQ No   Yes   Non-Secure FIQ (fast interrup
interrupt
Execution stopped at: S:0xC0017A88
S:0xC0017A88   MOV      pc,lr
add-symbol-file "MYD-SAMA5D3x\vmlinux"
Enabled Linux kernel support for version "Linux localho
Execution stopped at: S:0xC0017A88
In thread 2 (OS thread id 0)
S:0xC0017A88   74,0   mov   pc, lr
```

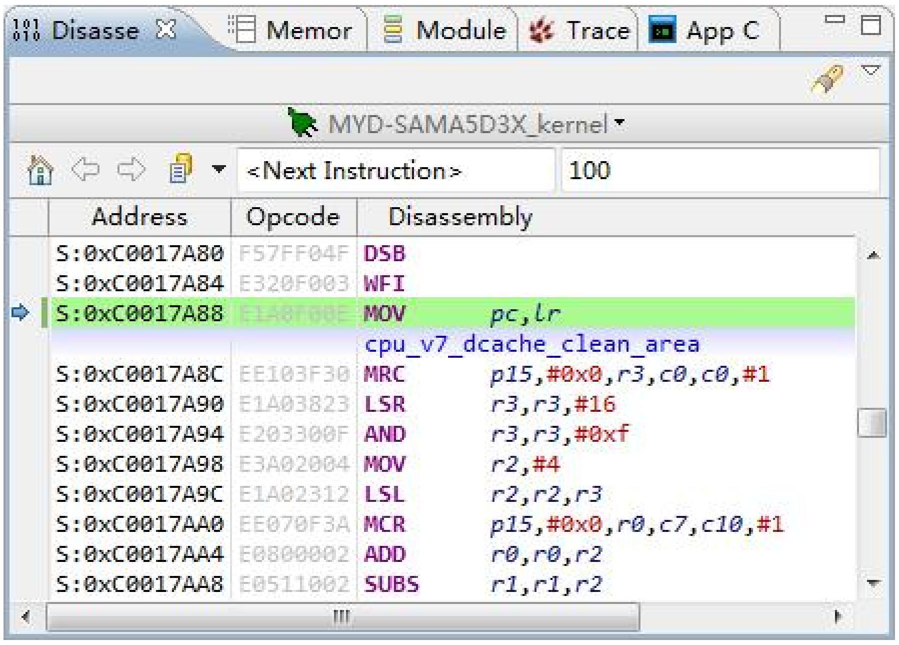
30) Assembly program column

It will show the assembly program, address, operand, etc.



```
 *
 *  IRQs are already disabled.
 */
ENTRY(cpu_v7_do_idle)
    dsb                     @ WFI may enter a low-power mode
    wfi
    mov pc, lr
ENDPROC(cpu_v7_do_idle)

ENTRY(cpu_v7_dcache_clean_area)
#ifndef TLB_CAN_READ_FROM_L1_CACHE
    dcache_line_size r2, r3
1:  mcr p15, 0, r0, c7, c10, 1      @ clean D entry
    add r0. r0. r2
```



Disasse | Memor | Module | Trace | App C

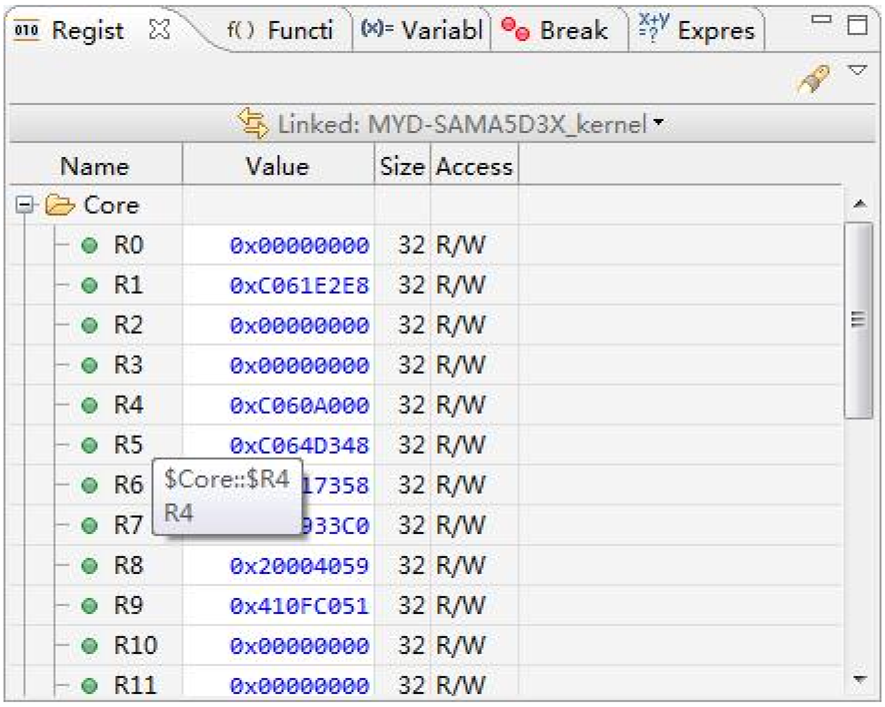
MYD-SAMA5D3X_kernel

<Next Instruction> 100

```
Address          Opcode    Disassembly
S:0xC0017A80     F57FF04F  DSB
S:0xC0017A84     E320F003  WFI
S:0xC0017A88     E1A0F00E  MOV      pc,lr
                           cpu_v7_dcache_clean_area
S:0xC0017A8C     EE103F30  MRC      p15,#0x0,r3,c0,c0,#1
S:0xC0017A90     E1A03823  LSR      r3,r3,#16
S:0xC0017A94     E203300F  AND      r3,r3,#0xf
S:0xC0017A98     E3A02004  MOV      r2,#4
S:0xC0017A9C     E1A02312  LSL      r2,r2,r3
S:0xC0017AA0     EE070F3A  MCR      p15,#0x0,r0,c7,c10,#1
S:0xC0017AA4     E0800002  ADD      r0,r0,r2
S:0xC0017AA8     E0511002  SUBS     r1,r1,r2
```

31）Register column

It shows all registers in the kernel, user can modify the register during debugging.



Regist | Functi | Variabl | Break | Expres

Linked: MYD-SAMA5D3X_kernel

Name	Value	Size	Access
Core			
R0	0x00000000	32	R/W
R1	0xC061E2E8	32	R/W
R2	0x00000000	32	R/W
R3	0x00000000	32	R/W
R4	0xC060A000	32	R/W
R5	0xC064D348	32	R/W
R6	17358	32	R/W
R7	933C0	32	R/W
R8	0x20004059	32	R/W
R9	0x410FC051	32	R/W
R10	0x00000000	32	R/W
R11	0x00000000	32	R/W

$Core::$R4
R4