

---

# Using **VS CODE + QEMU** debugging **RT-THREAD**

---

**RT-THREAD** Document Center

Copyright ©2019 Shanghai Ruisaide Electronic Technology Co., Ltd.



[WWW.RT-THREAD.ORG](http://WWW.RT-THREAD.ORG)

Friday 28th September, 2018

[Table of contents](#)

Table of contents	i
1 Purpose and structure of this paper . . . . .	1
1.1 Purpose and Background of this Paper . . . . .	1
1.2 Structure of this paper . . . . .	1
2. Preparation. . . . .	1
3. Run and debug RT-Thread. . . . .	1
3.1 Step 1 Install the debug plugin. . . . .	1
3.2 Step 2: Open the VS Code project. . . . .	2
3.3 Step 3 Compile RT-Thread. . . . .	3
3.4 Step 4: Modify the qemu-dbg.bat file. . . . .	6
3.5 Step 5: Debug the project. . . . .	6
4. Notes. . . . .	9
5 References. . . . .	9
6 Frequently Asked Questions. . . . .	9

This application note describes how to use VS Code to debug RT-Thread on Windows.

qemu-vexpress-a9 BSP project.

## 1 Purpose and structure of this paper

### 1.1 Purpose and Background of this Paper

VS Code (full name: Visual Studio Code) is a lightweight and powerful code editor available for Windows, OS X, and Linux. It includes built-in support for JavaScript, TypeScript, and Node.js, and boasts a rich ecosystem of plugins that can be installed to support other languages, such as C++, C#, Python, and PHP.

This article mainly introduces how to use VS Code to debug the qemu-vexpress-a9 BSP project on the Windows platform.

### 1.2 Structure of this paper

This article mainly introduces the preparation for VS Code debugging and how to debug the project.

## 2. Preparation

- [Download RT-Thread](#) Source code, note: One-click **VS Code** debugging only supports **RT-Thread v3.1.0** and above  
Previous version.
- [Download RT-Thread Env](#) Tools, it is recommended to download version 1.0.0 or above.
- [Download VS Code](#)

## 3 Running and Debugging RT-Thread

### 3.1 Step 1 Install the debugging plug-in

Download and install the debugging extension that supports C/C++ in VS Code Extensions:

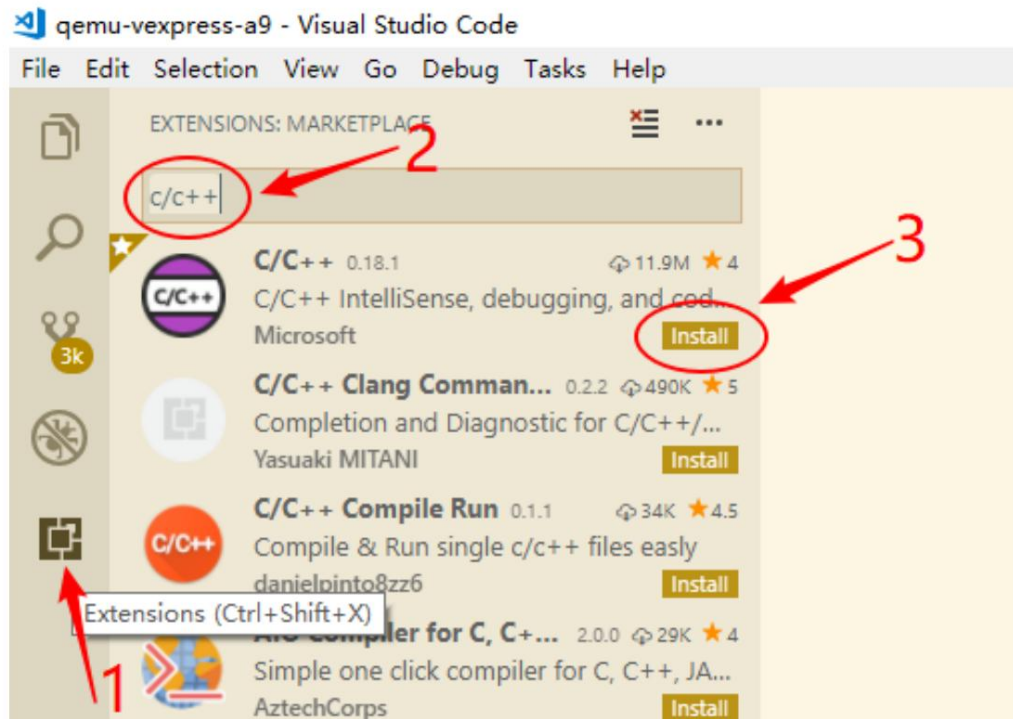


Figure 1: Install C/C++ plugins

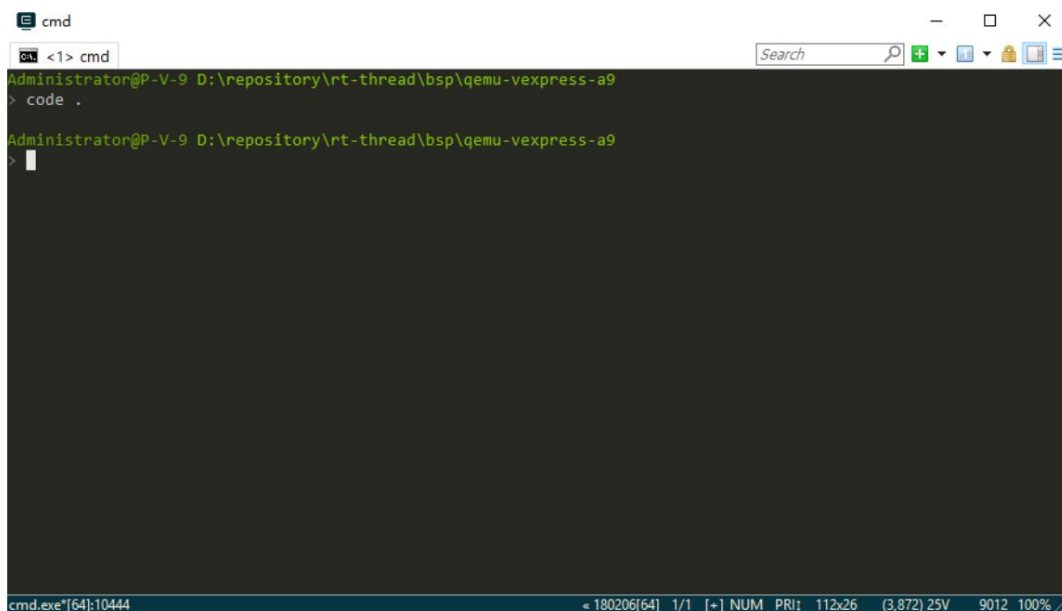
After installation, confirm that the plugin is in the following status. If not, click Reload:



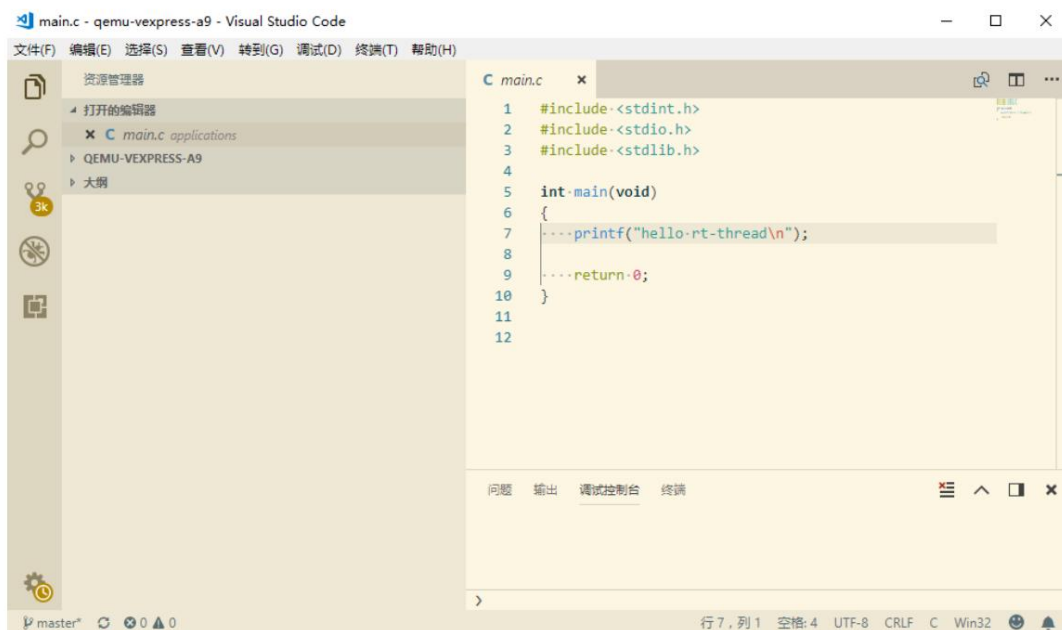
Figure 2: C/C++ Plugin Status

### 3.2 Step 2 Open the VS Code project

Enter the qemu-vexpress-a9 BSP root directory in the Env console and enter the command `code .` (Note: `code` There is a dot after it) to open VS Code, which means to open the current directory with VS Code.

Figure 3: Open *Env* console

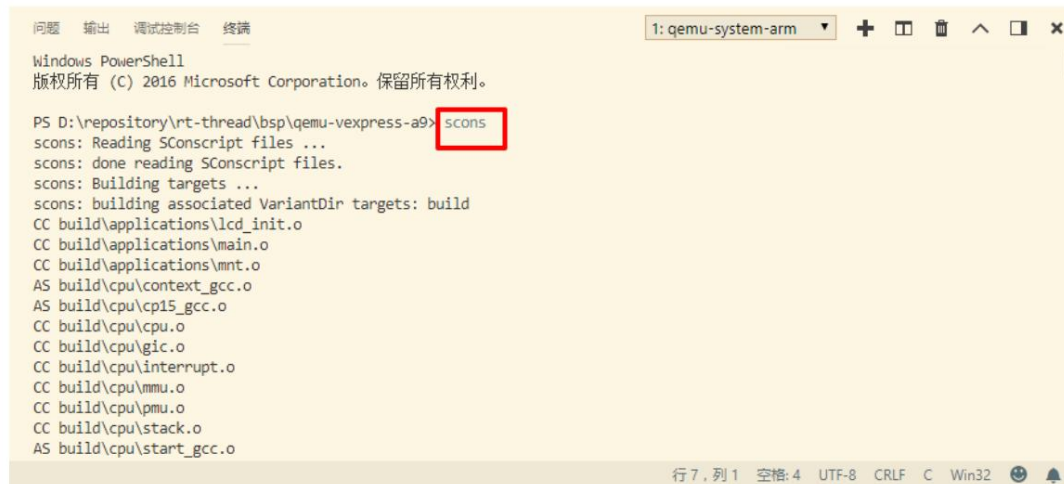
When VS Code opens, it automatically opens the qemu-vexpress-a9 BSP folder, as shown in the following figure.

Figure 4: Open *VS Code*

### 3.3 Step 3 Compile RT-Thread

Click VS Code "View -> Terminal" to open the VS Code internal terminal and enter the command `scons` in the terminal .

Compile the project and the terminal will print out the compilation information.



```
Windows PowerShell
版权所有 (C) 2016 Microsoft Corporation。保留所有权利。

PS D:\repository\rt-thread\bsp\qemu-vexpress-a9> scons
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
scons: building associated VariantDir targets: build
CC build\applications\lcd_init.o
CC build\applications\main.o
CC build\applications\mnt.o
AS build\cpu\context_gcc.o
AS build\cpu\cp15_gcc.o
CC build\cpu\cpu.o
CC build\cpu\gic.o
CC build\cpu\interrupt.o
CC build\cpu\mmu.o
CC build\cpu\pmu.o
CC build\cpu\stack.o
AS build\cpu\start_gcc.o
```

Figure 5: Compile project

After the compilation is complete, enter the `.\qemu.bat` command to run the project. The terminal will output the RT-Thread startup logo. Information, QEMU is also running.

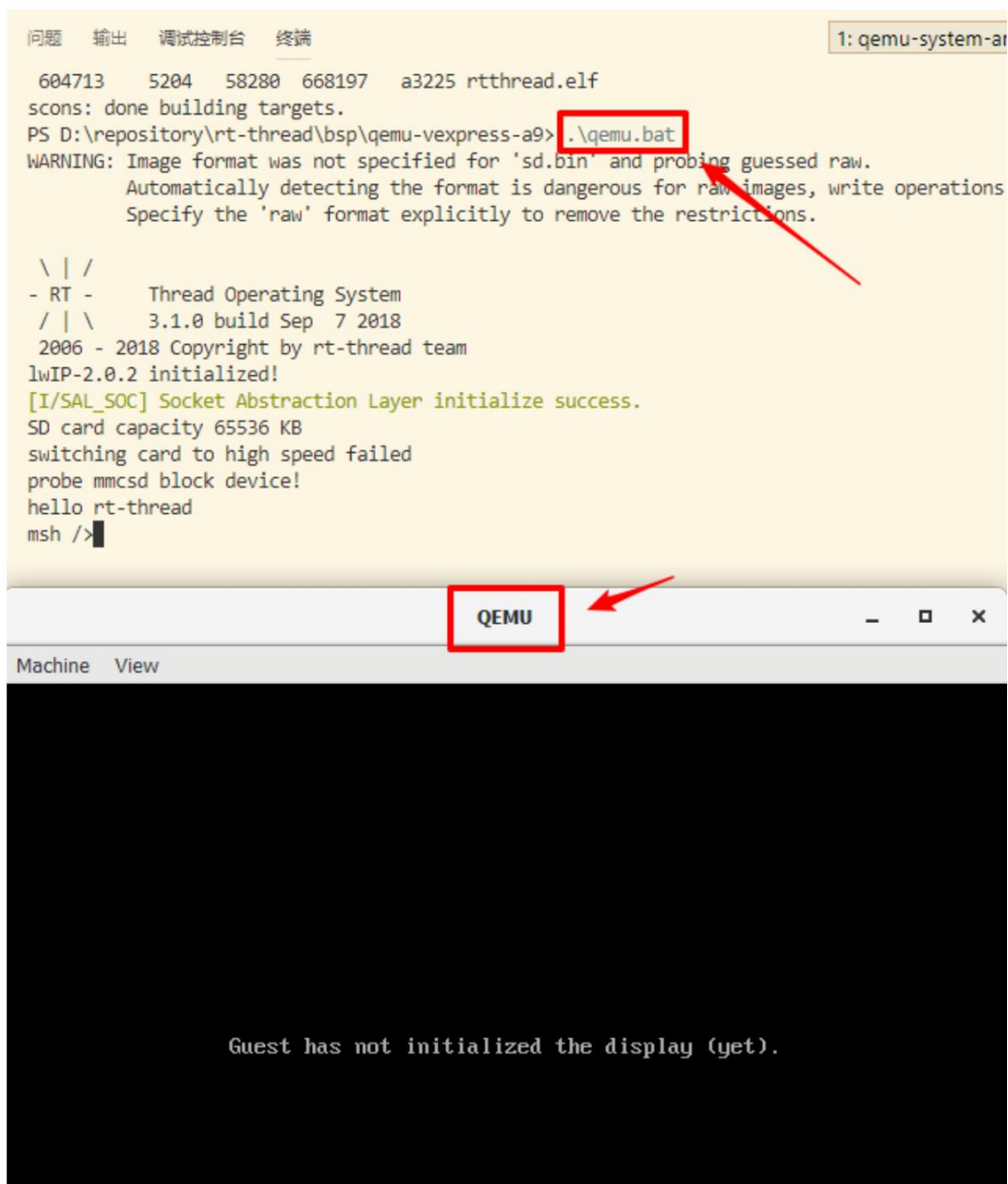


Figure 6: Run the project

#### Notes:

1. Before debugging the BSP project, you need to compile the project to generate the rtthread.elf file.

2. You can use the `scons -target=vsc -s` command to update the C/C++ header files needed by VS Code.

Search path information. It does not need to be updated every time. It is only updated when RT-Thread is reconfigured or updated using menuconfig.

Only needed when the `rtconfig.h` header file is modified.

### 3.4 Step 4 Modify the qemu-dbg.bat file

Before starting debugging, you need to edit the `qemu-dbg.bat` file in the `qemu-vexpress-a9` directory and add start before `qemu-system-arm`:

```
@echo off if
exist sd.bin goto run qemu-img create
-f raw sd.bin 64M

:run

start qemu-system-arm -M vexpress-a9 -kernel rtthread.elf -serial stdio -
sd sd.bin -S -s
```

### 3.5 Step 5 Debugging the Project

As shown in the figure below, in VS Code, click the debug menu (the little bug icon), select Windows as the debugging platform, and then press F5 to start QEMU debugging mode with the breakpoint stopped at the main function. The VS Code debugging options are shown in the figure below:

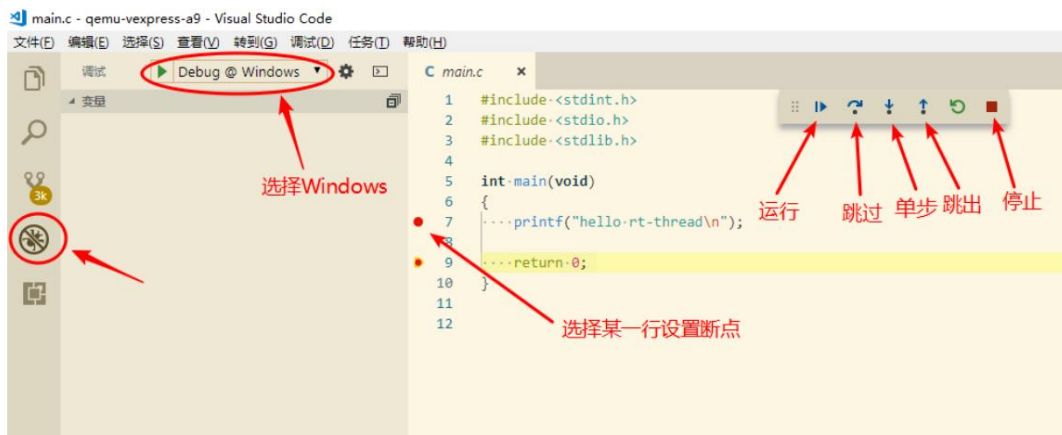
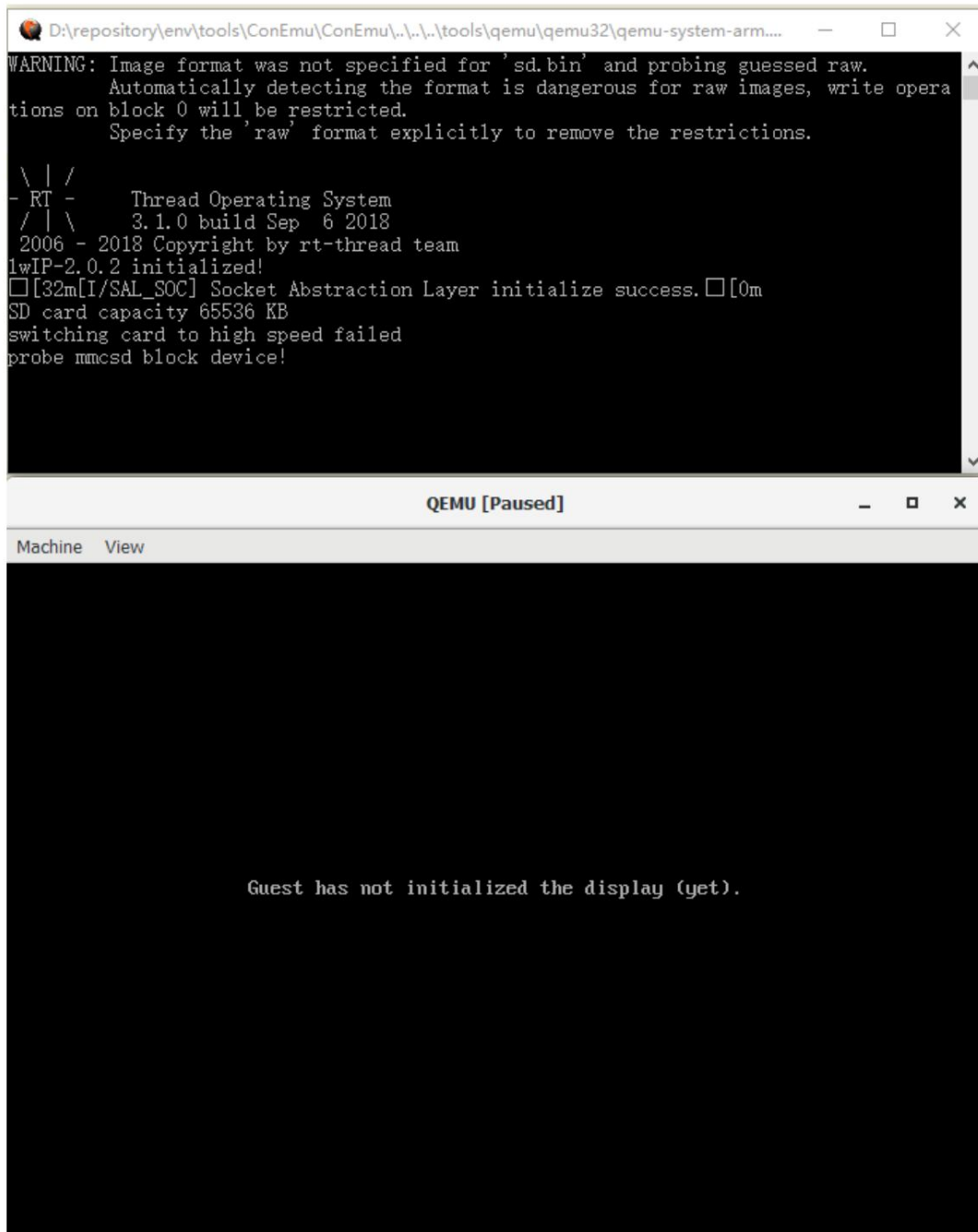


Figure 7: Env Debugging surface

QEMU is also running, as shown in the figure below.



Figure 8: *qemu* Debugging surface

To use GDB commands in VS Code, you need to add `-exec` at the beginning. For example, `-exec info`

The `registers` command can view the contents of the registers:

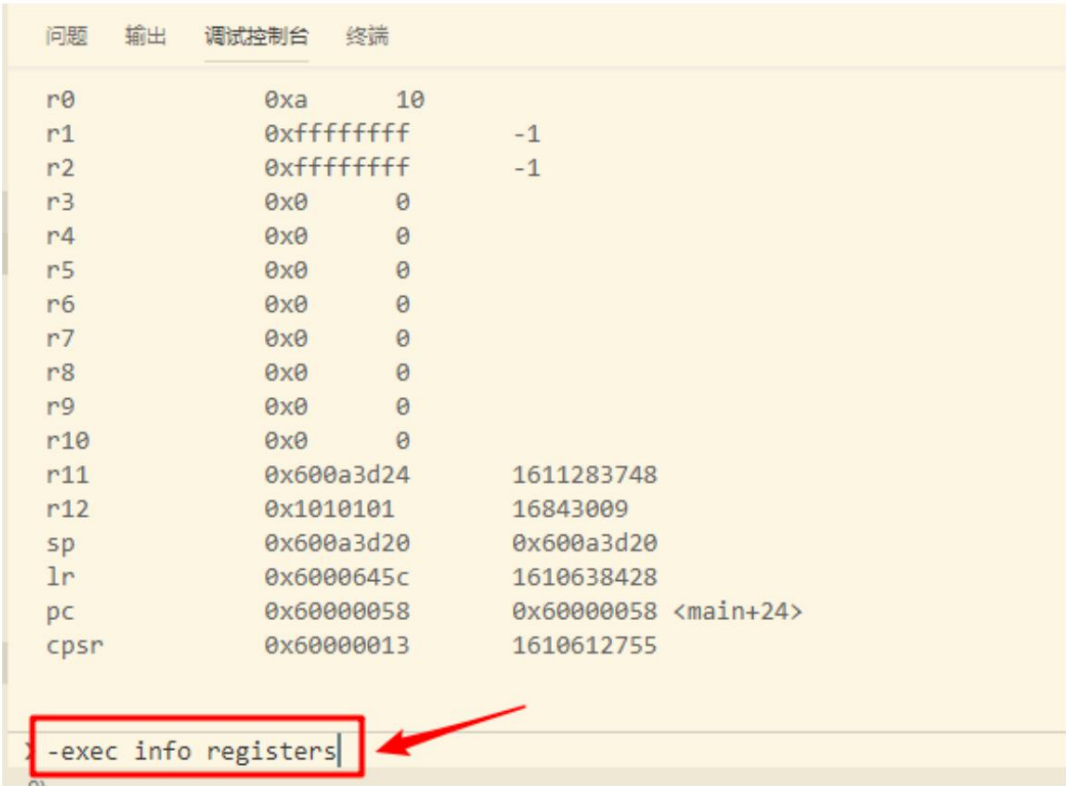


Figure 9: gdb View Memory

Some other main commands are introduced as follows:

View the memory address content: `x/<n/f/u> <addr>`. The description of each parameter is as follows:

- n is a positive integer, indicating the number of memory cells to be displayed, that is, the number of cells to be displayed from the current address.
- The content of a memory unit. The size of a memory unit is defined by the u behind it.
- f indicates the display format, see below. If the address refers to a character string, the format can be s. Other formats

The formula is shown in the following table:

parameter	describe
x	Display variables in hexadecimal format
d	Display variables in decimal format
u	Displays an unsigned integer in hexadecimal format
o	Display variables in octal format
t	Display variables in binary format
a	Display variables in hexadecimal format
c	Display variables in character format
f	Display variables in floating-point format

- `u` indicates the number of bytes to request from the current address. If not specified, GDB defaults to 4 bytes. The `u` parameter can be replaced by the following characters: `b` for single-byte, `h` for double-byte, `w` for four-byte, and `g` for eight-byte. When the byte length is specified, GDB starts at the specified memory address, reads and writes the specified bytes, and retrieves them as a single value.

- `addr` represents a memory address.

Note: Strictly distinguish the relationship between `n` and `u`. `n` represents the number of units, and `u` represents the size of each unit.

Example: `x/3uh 0x54320` means reading content from memory address `0x54320`, `h` means double bytes as a unit, `3` means output three units, and `u` means display in hexadecimal.

View the contents of the current program stack: `x/10x $sp->` print the first 10 elements of the stack  
 View the information of the current program stack: `info frame`—list general info about the frame  
 View the parameters of the current program stack: `info args`—lists arguments to the function  
 View the local variables of the current program stack: `info locals`—list variables stored in the frame  
 View the value of the current register: `info registers` (excluding floating-point registers)  
`info all-registers` (including floating-point registers)  
 View the exception handlers in the current stack frame: `info catch` (exception handlers)

**Tips:** When entering a command, you can enter only the first letter of each command. For example, you can enter only `i r` for `info registers`.

## 4. Notes

- If you add additional folders to the VS Code directory, debugging will not start.
- Each time you start debugging, you need to use the `Env` tool to open VS Code in the BSP root directory using the `code` command.

Normal debugging project.

## 5References

- [Env tool user manual](#)

## 6 Frequently Asked Questions

- For questions related to the `Env` tool, please refer to the Common Resource Links section of the `Env` Tool User Manual.
- It says that 'qemu-system-arm' cannot be found.

Solution: This error will occur when you directly open VS Code to debug the project. For each debugging, please use the **Env** tool to open **VS Code** in the **BSP** root directory using the `code .` command.

- The VS Code debugging options do not have the `Debug@windows` option or other debugging issues occur.

Solution: Please update the RT-Thread source code to v3.1.0 or above.

- VS Code displays an error message: Unable to start debugging. Unexpected GDB output from command "-interpreter-exec console".....

Solution: Please modify the qemu-dbg.bat file, especially when the source code is updated.

Please follow the document steps and check whether you have completed each step.