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

RT-THREAD Documentation Center

WWW.RT-THREAD.ORG

Friday 28th September, 2018

Table of contents

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 that supports Windows, OS X and Linux. It has built-in support for JavaScript, TypeScript and Node.js, and has a rich plugin ecosystem that can support other languages such as C++, C#, Python, PHP, etc. by installing plugins.

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 plugin that supports C/C++ in VS Code Extensions:

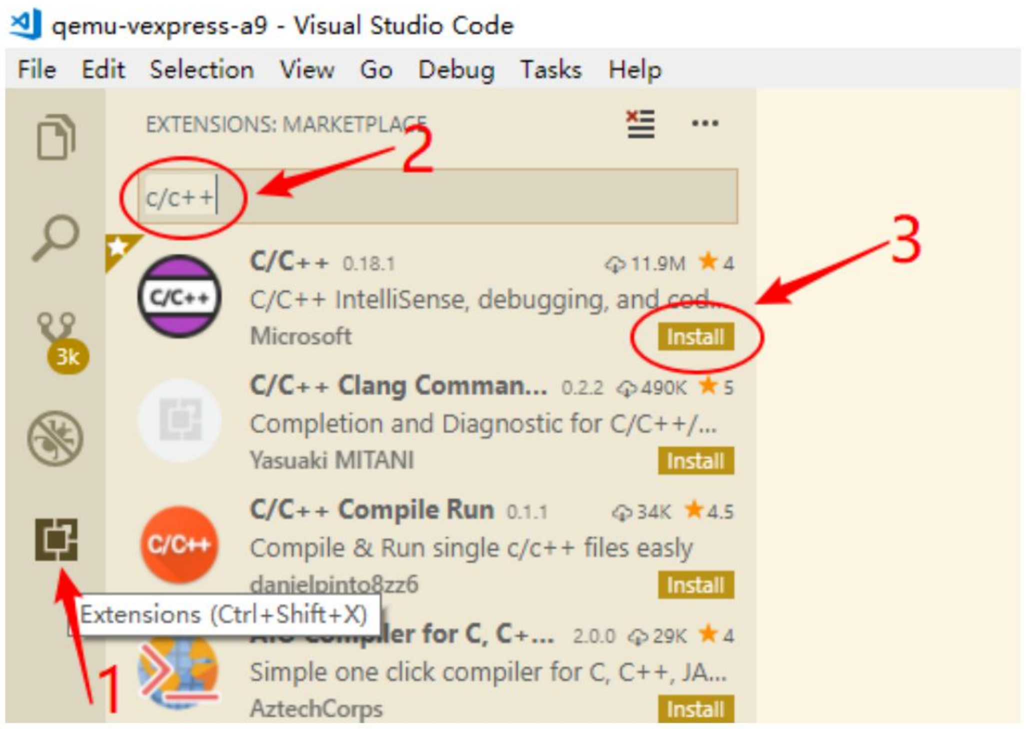

qemu-vexpress-a9 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXTENSIONS: MARKETPLACE
c/c++
C/C++ 0.18.1 11.9M 4
C/C++ IntelliSense, debugging, and cod...
Microsoft
Install
C/C++ Clang Comman... 0.2.2 490K 5
Completion and Diagnostic for C/C++/...
Yasuaki MITANI
Install
C/C++ Compile Run 0.1.1 34K 4.5
Compile & Run single c/c++ files easly
danielpinto8zz6
Install
Extensions (Ctrl+Shift+X)
...piler for C, C+... 2.0.0 29K 4
Simple one click compiler for C, C++, JA...
AztechCorps
Install
1
2
3

figure 1: Install C/C++ Plugins

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

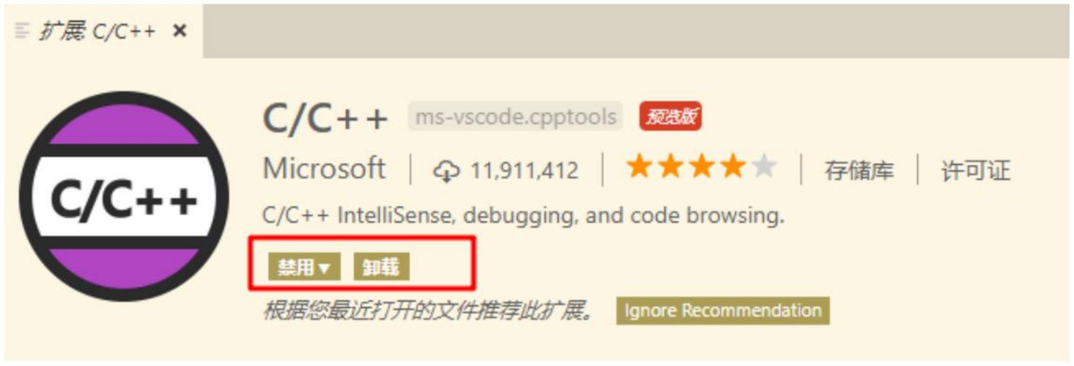

扩展 C/C++
C/C++ ms-vscode.cpptools 预览版
Microsoft | 11,911,412 | 存储库 | 许可证
C/C++ IntelliSense, debugging, and code browsing.
禁用▾ 卸载
根据您最近打开的文件推荐此扩展。 Ignore Recommendation

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.

Machine Translated by Google

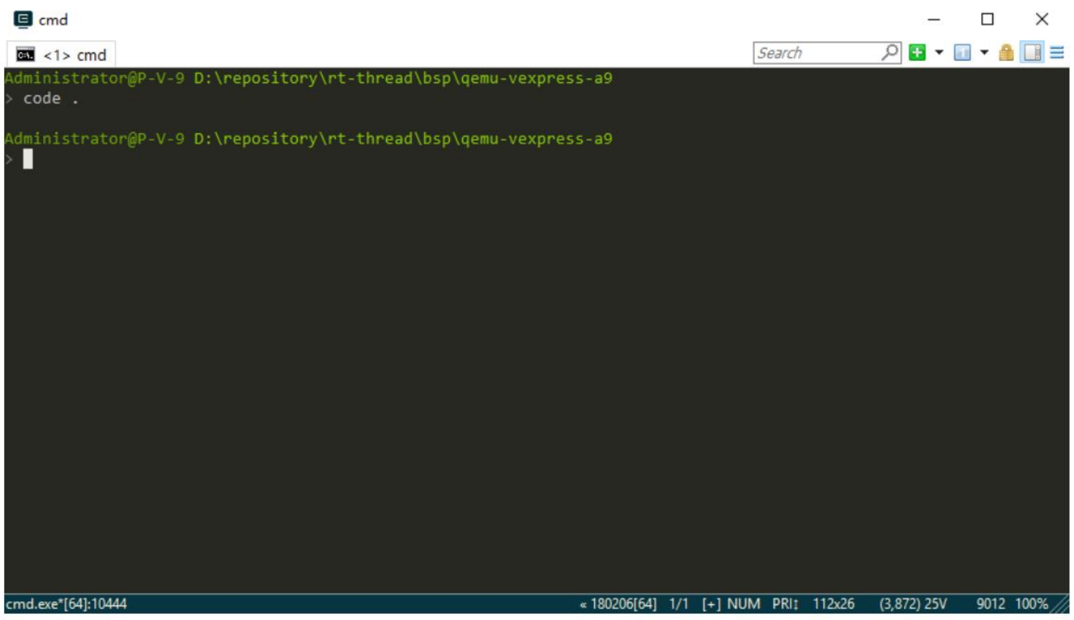

```
Administrator@P-V-9 D:\repository\rt-thread\bsp\qemu-vexpress-a9
> code .

Administrator@P-V-9 D:\repository\rt-thread\bsp\qemu-vexpress-a9
>
```

image 3: Open *Env* Console

When VS Code is opened, the qemu-vexpress-a9 BSP folder will be automatically opened, as shown in the following figure.

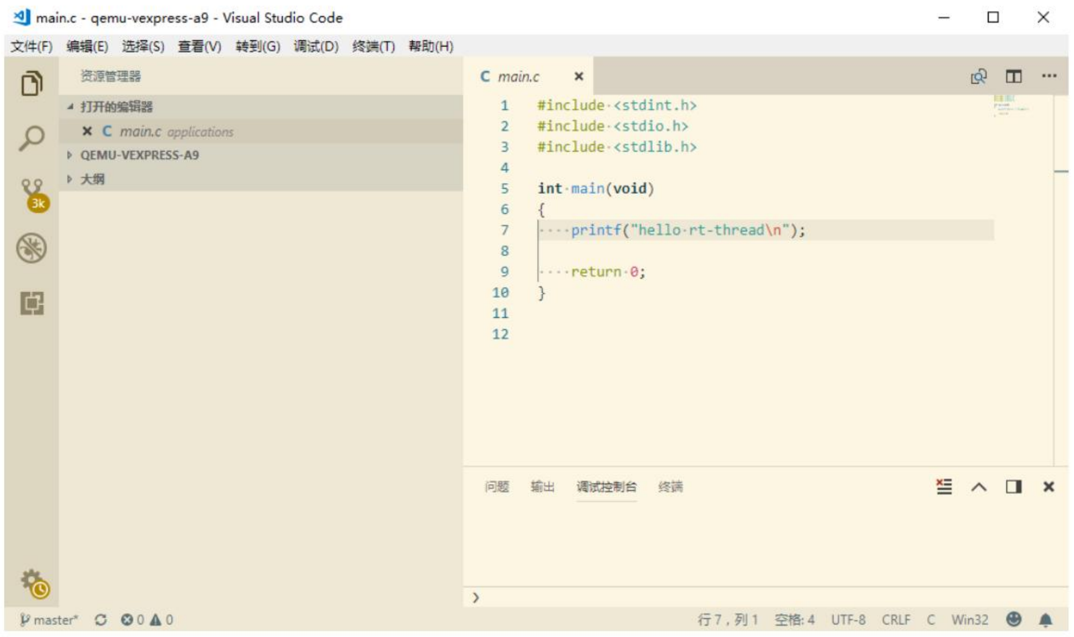

```c
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    printf("hello rt-thread\n");

    return 0;
}
```
main.c - qemu-vexpress-a9 - Visual Studio Code
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 调试(D) 终端(T) 帮助(H)
资源管理器
打开的编辑器
main.c applications
QEMU-VEXPRESS-A9
大纲
main.c
问题 输出 调试控制台 终端
master*
行 7，列 1 空格: 4 UTF-8 CRLF C Win32

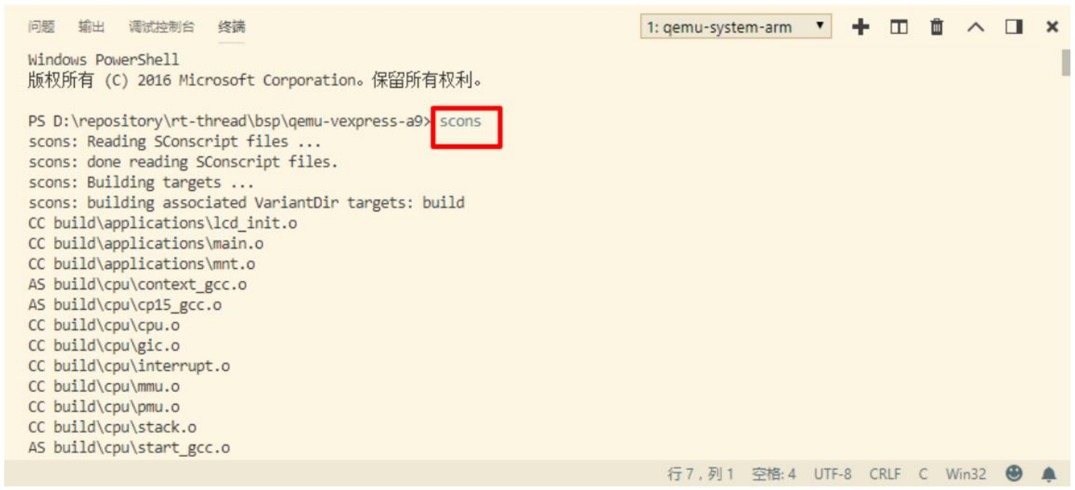
Figure 4: Open *VS Code*

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.

Machine Translated by Google

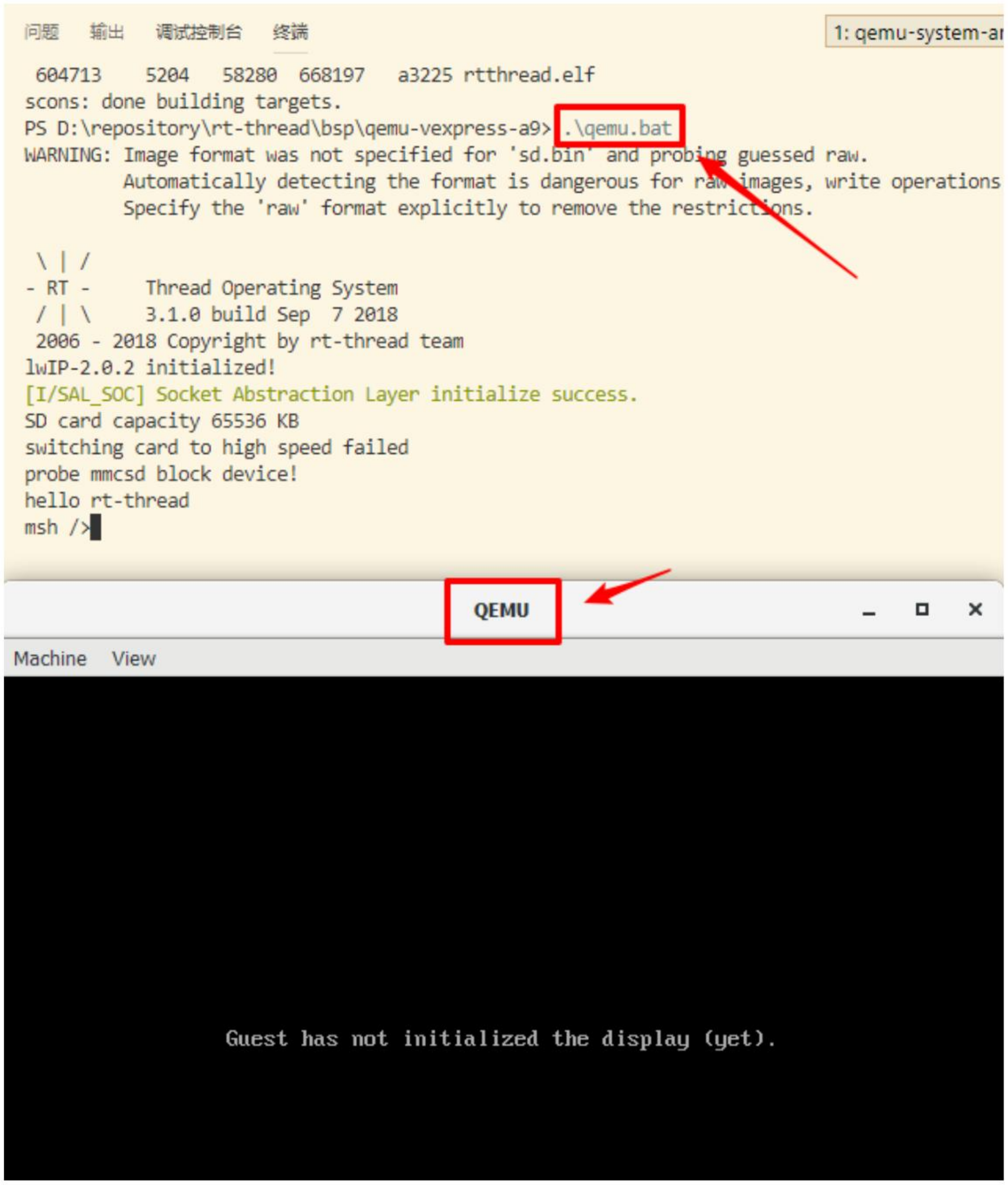


```
Windows PowerShell
版权所有 (C) 2016 Microsoft Corporation。保留所有权利。

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.

Machine Translated by Google



```
 604713    5204   58280  668197   a3225 rtthread.elf
scons: done building targets.
PS D:\repository\rt-thread\bsp\qemu-vexpress-a9> .\qemu.bat
WARNING: Image format was not specified for 'sd.bin' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write operations
         Specify the 'raw' format explicitly to remove the restrictions.

 \ | /
- RT -     Thread Operating System
 / | \     3.1.0 build Sep  7 2018
 2006 - 2018 Copyright by rt-thread team
lwIP-2.0.2 initialized!
[I/SAL_SOC] Socket Abstraction Layer initialize success.
SD card capacity 65536 KB
switching card to high speed failed
probe mmcsd block device!
hello rt-thread
msh />
```

```
Guest has not initialized the display (yet).
```

Figure 6: Run the project

Precautions:

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.

It does not need to be updated every time. It is only updated when menuconfig is used to reconfigure RT-Thread or when

Only needed if the rtconfig.h header file has been modified.

Machine Translated by Google

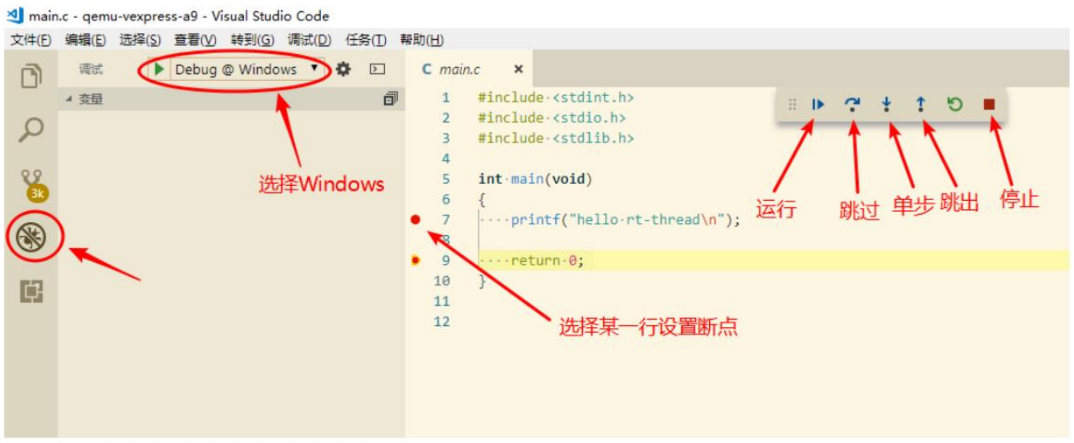
3.4 Step 4 Modify the qemu-dbg.bat file

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

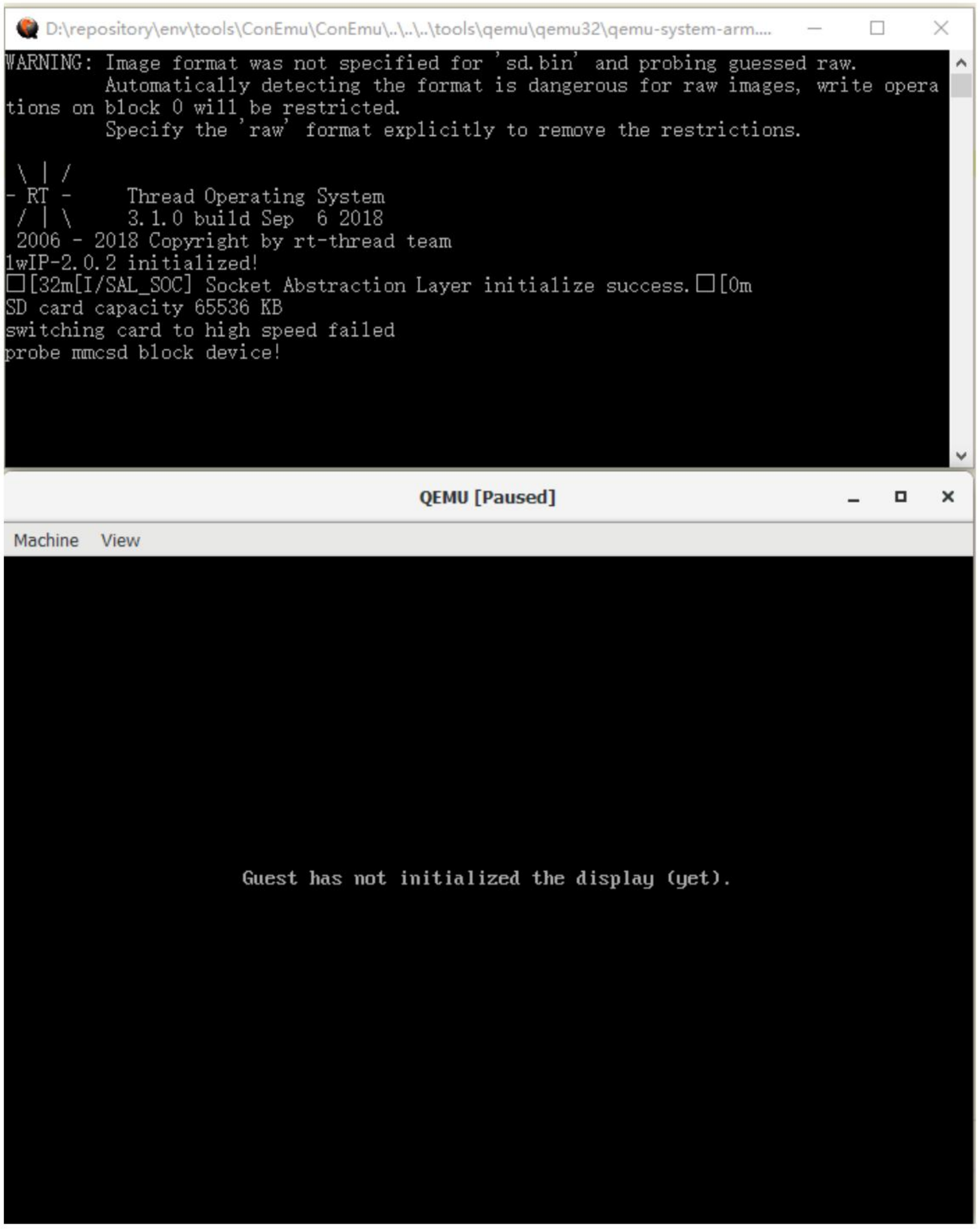
As shown in the figure below, click the debug menu (bug icon) in VS Code, select Windows as the debugging platform, and then press F5 to start QEMU debugging mode, with the breakpoint staying at the main function. The VS Code debugging options are shown in the figure below:


main.c - qemu-vexpress-a9 - Visual Studio Code
文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 调试(D) 任务(T) 帮助(H)
调试
Debug @ Windows
变量
main.c
选择Windows
运行
跳过
单步
跳出
停止
选择某一行设置断点

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    printf("hello rt-thread\n");

    return 0;
}
```

Figure 7: *Env* Debugging surface

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


D:\repository\env\tools\ConEmu\ConEmu\..\..\..\tools\qemu\qemu32\qemu-system-arm....
QEMU [Paused]
Machine
View
Guest has not initialized the display (yet).

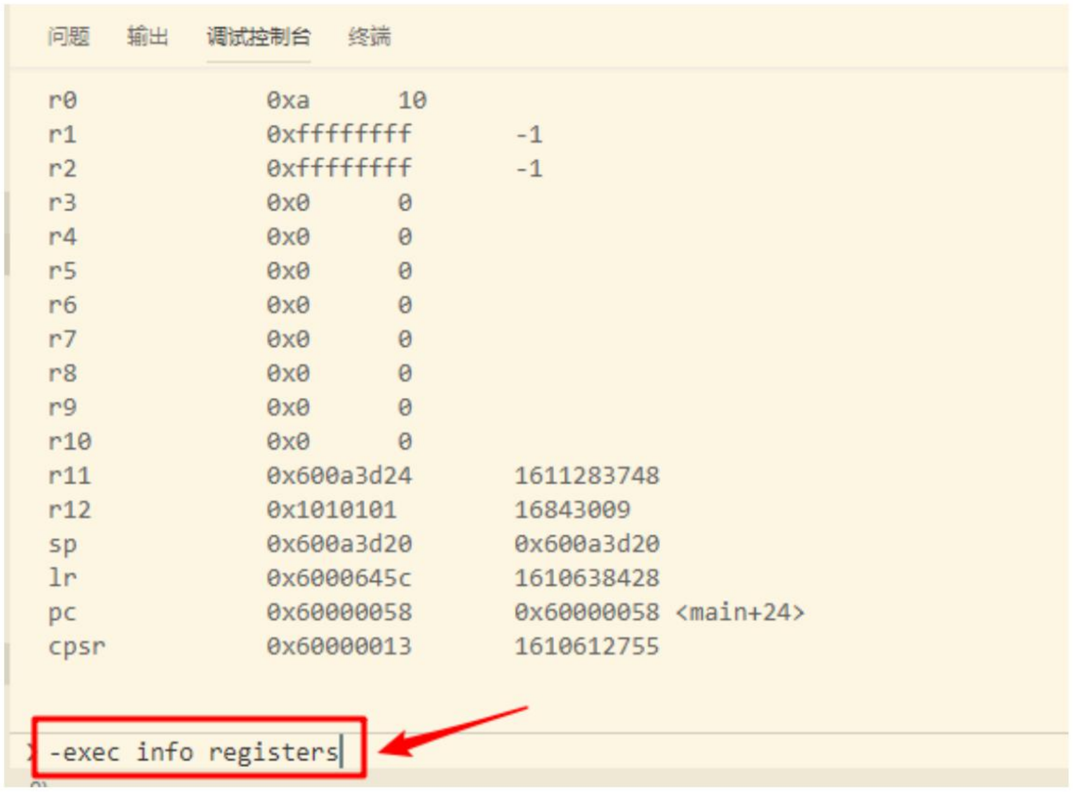
```
WARNING: Image format was not specified for 'sd.bin' and probing guessed raw.
         Automatically detecting the format is dangerous for raw images, write opera
tions on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.

 \ | /
- RT -     Thread Operating System
 / | \     3.1.0 build Sep  6 2018
 2006 - 2018 Copyright by rt-thread team
lwIP-2.0.2 initialized!
□[32m[I/SAL_SOC] Socket Abstraction Layer initialize success.□[0m
SD card capacity 65536 KB
switching card to high speed failed
probe mmcsd block device!
```

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:

Machine Translated by Google



```
r0              0xa      10
r1              0xffffffff       -1
r2              0xffffffff       -1
r3              0x0      0
r4              0x0      0
r5              0x0      0
r6              0x0      0
r7              0x0      0
r8              0x0      0
r9              0x0      0
r10             0x0      0
r11             0x600a3d24       1611283748
r12             0x1010101        16843009
sp              0x600a3d20       0x600a3d20
lr              0x6000645c       1610638428
pc              0x60000058       0x60000058 <main+24>
cpsr            0x60000013       1610612755

-exec info 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 string, the format can be s.

The formula is shown in the following table:

parameter	describe
x	Display variables in hexadecimal format
d	Display variables in decimal format
u	Display unsigned integers 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 requested 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 bytes, and g for eight bytes. When we specify the byte length, GDB will start from the specified memory address, read and write the specified bytes, and take it out as a 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 in 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.

5 References

- 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. Each time you debug, please use **the Env** tool to open **VS Code** in **the BSP** root directory using the code . command .

- The VS Code debugging option does 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 if there is an update to the source code.

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