

---

# 使用 VS CODE + QEMU 调试 RT-THREAD

---

RT-THREAD 文档中心

上海睿赛德电子科技有限公司版权 ©2019



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

Friday 28<sup>th</sup> September, 2018

# 目录

|                           |   |
|---------------------------|---|
| 目录                        | i |
| 1 本文的目的和结构                | 1 |
| 1.1 本文的目的和背景              | 1 |
| 1.2 本文的结构                 | 1 |
| 2 准备工作                    | 1 |
| 3 运行和调试 RT-Thread         | 1 |
| 3.1 步骤一安装调试插件             | 1 |
| 3.2 步骤二打开 VS Code 项目工程    | 2 |
| 3.3 步骤三编译 RT-Thread       | 3 |
| 3.4 步骤四修改 qemu-dbg.bat 文件 | 6 |
| 3.5 步骤五调试工程               | 6 |
| 4 注意事项                    | 9 |
| 5 参考                      | 9 |
| 6 常见问题                    | 9 |

!!! abstract “摘要” 本应用笔记描述了在 Windows 平台使用 VS Code 调试 RT-Thread qemu-vexpress-a9 BSP 工程。

## 1 本文的目的和结构

### 1.1 本文的目的和背景

VS Code（全称 Visual Studio Code）是一个轻量且强大的代码编辑器，支持 Windows，OS X 和 Linux。内置 JavaScript、TypeScript 和 Node.js 支持，而且拥有丰富的插件生态系统，可通过安装插件来支持 C++、C#、Python、PHP 等其他语言。

本文主要介绍在 Windows 平台使用 VS Code 调试 qemu-vexpress-a9 BSP 工程。

### 1.2 本文的结构

本文主要介绍 VS Code 调试准备工作以及如何调试工程。

## 2 准备工作

- 下载 [RT-Thread 源码](#)，注意：一键 VS Code 调试只支持 RT-Thread v3.1.0 及以上版本。
- 下载 [RT-Thread Env 工具](#)，推荐下载 1.0.0 及以上版本。
- 下载 [VS Code](#)

## 3 运行和调试 RT-Thread

### 3.1 步骤一安装调试插件

在 VS Code Extensions 里下载并安装支持 C/C++ 的调试插件：

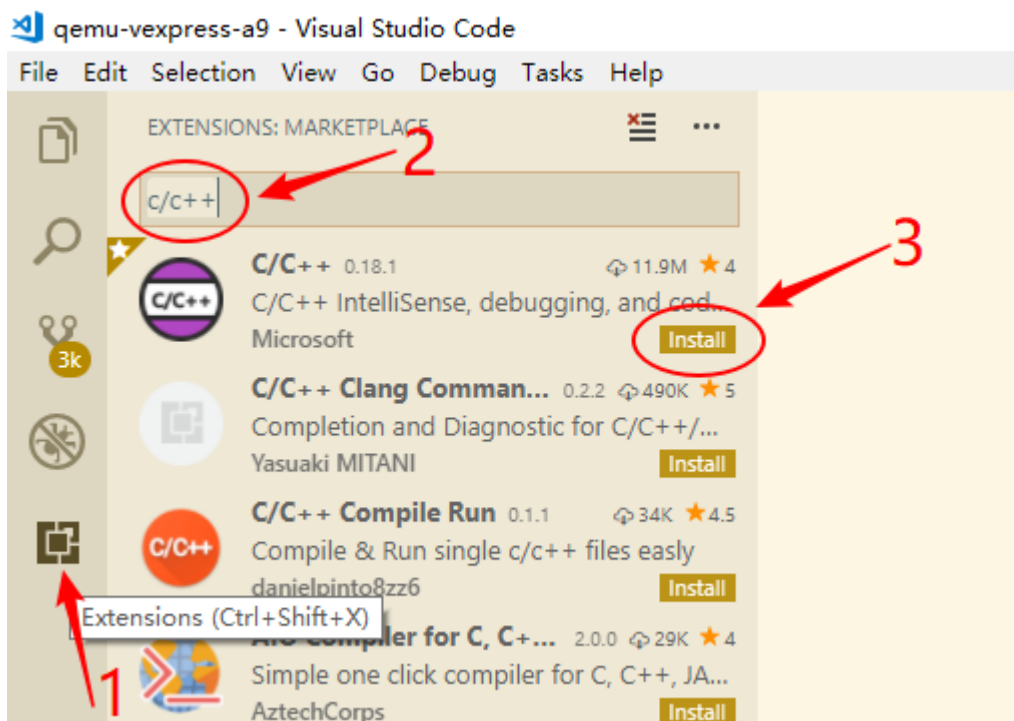


图 1: 安装 c/c++ 插件

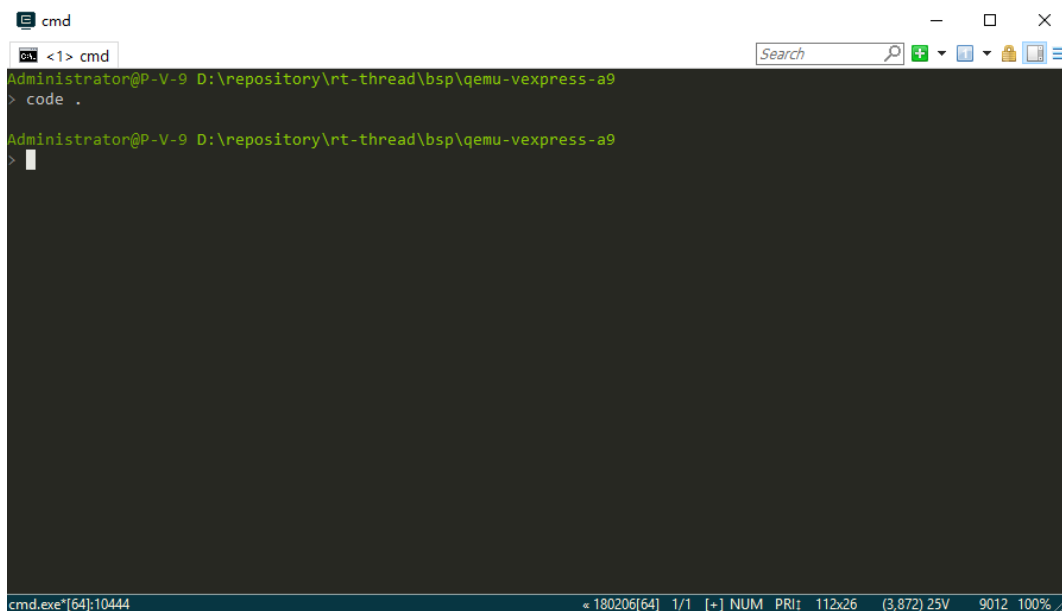
安装好后确认插件为以下状态，如果不是则点击重新加载：



图 2: c/c++ 插件状态

### 3.2 步骤二打开 VS Code 项目工程

在 Env 控制台进入 qemu-vexpress-a9 BSP 根目录，然后输入命令 `code .` (注意: `code` 后面有一个点) 打开 VS Code，表示使用 VS Code 打开当前目录。



**图 3:** 打开 *Env* 控制台

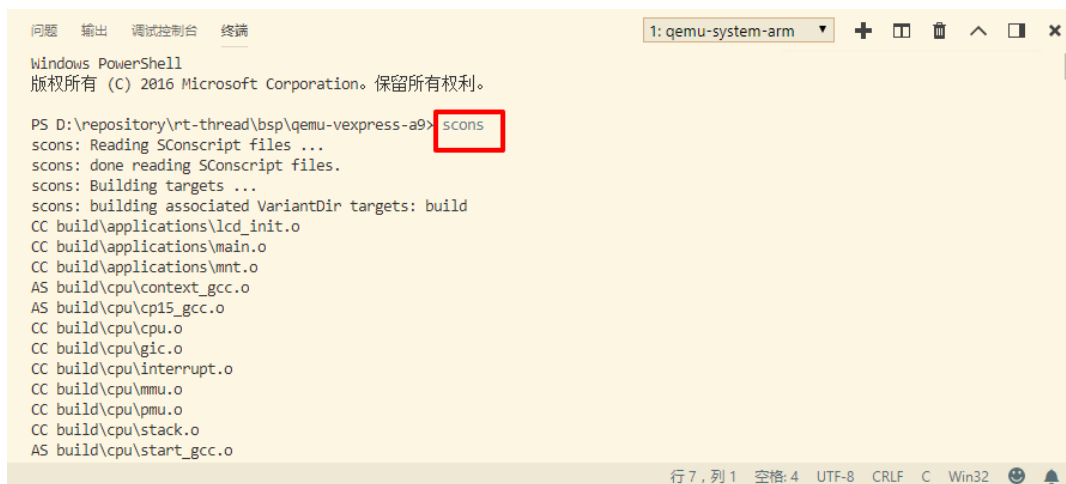
VS Code 打开后会自动打开 qemu-vexpress-a9 BSP 文件夹，如下图所示。



图 4: 打开 VS Code

### 3.3 步骤三编译 RT-Thread

点击 VS Code “查看 -> 终端” 打开 VS Code 内部终端，在终端里输入命令 `scons` 即可编译工程，终端会打印出编译信息。



```
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
```

图 5: 编译工程

编译完成后输入 `.\qemu.bat` 命令就可以运行工程。终端会输出 RT-Thread 启动 logo 信息，QEMU 也运行了起来。

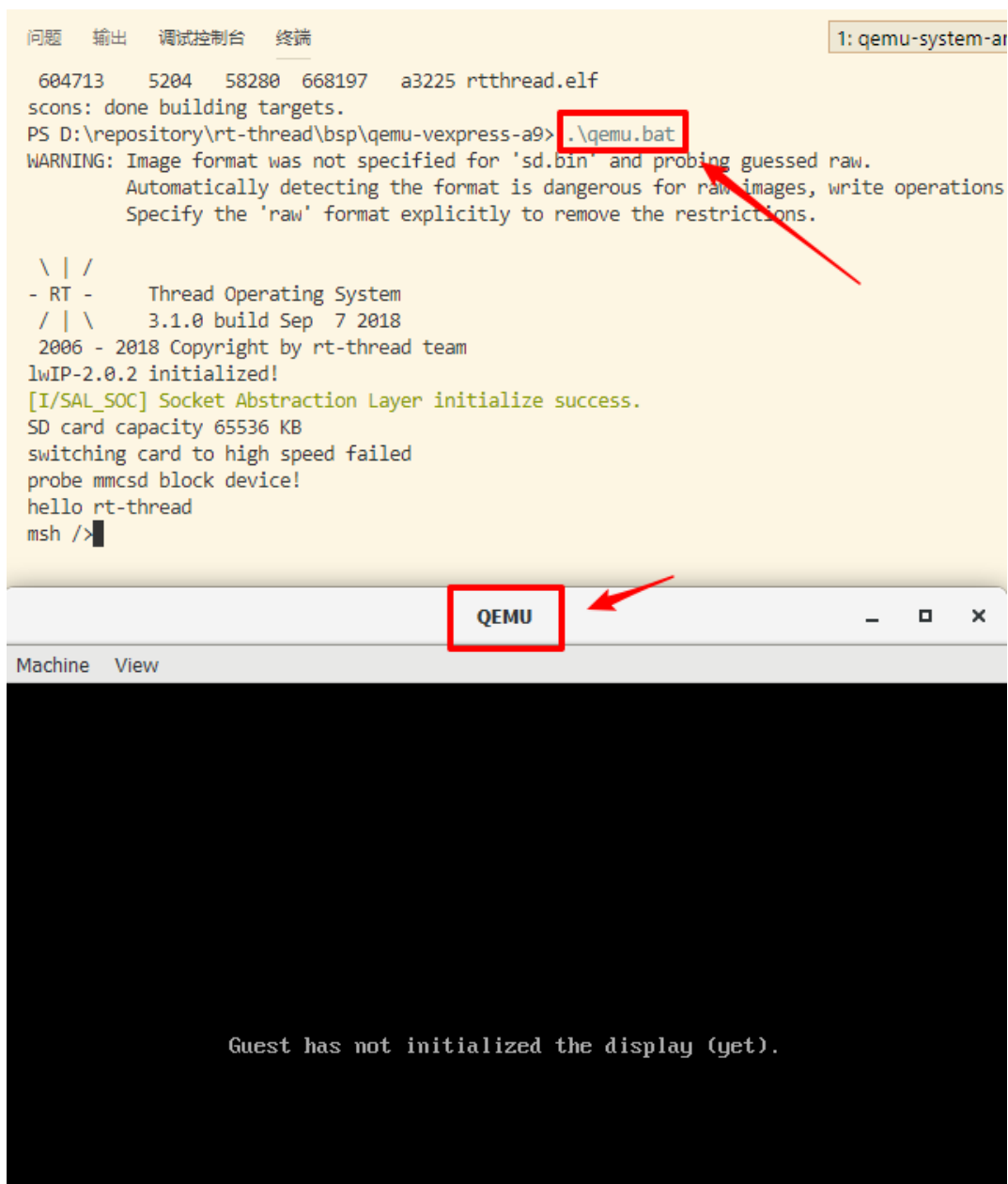


图 6: 运行工程

**注意事项:**

- 1、调试 BSP 工程前需要先编译工程生成 rtthread.elf 文件。
- 2、可以使用 `scons --target=vsc -s` 命令更新 VS Code 需要用到的 C/C++ 头文件搜索路径信息。不是每次都需要更新，只有在使用了 menuconfig 重新配置了 RT-Thread 或更改了 rtconfig.h 头文件时才需要。

### 3.4 步骤四修改 qemu-dbg.bat 文件

开始调试前需要编辑 `qemu-vexpress-a9` 目录下的 `qemu-dbg.bat` 文件，在 `qemu-system-arm` 前加入 `start`：

```
@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 步骤五调试工程

如下图所示，在 VS Code 里点击调试菜单（小虫子图标），调试平台选择 Windows，然后按 F5 就可以开启 QEMU 调试模式，断点停留在 `main` 函数。VS Code 调试选项如下图所示：

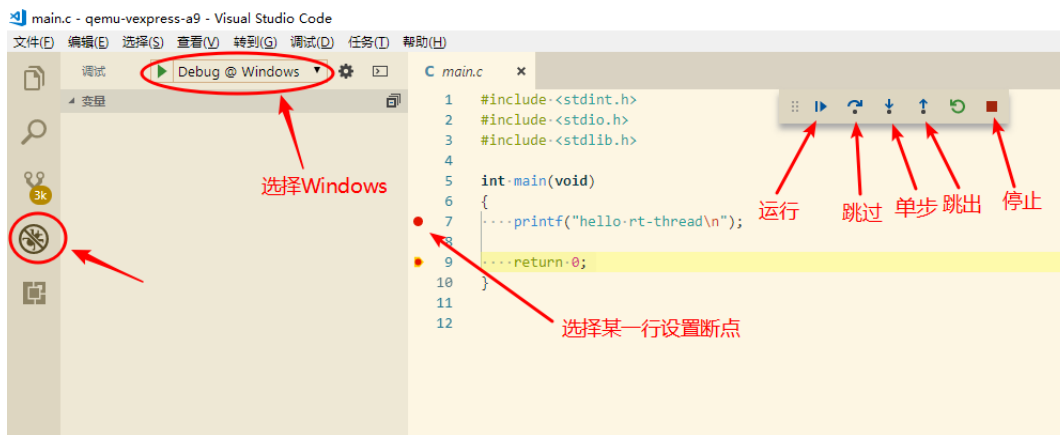


图 7: Env 调试界面

QEMU 也运行了起来，如下图所示。



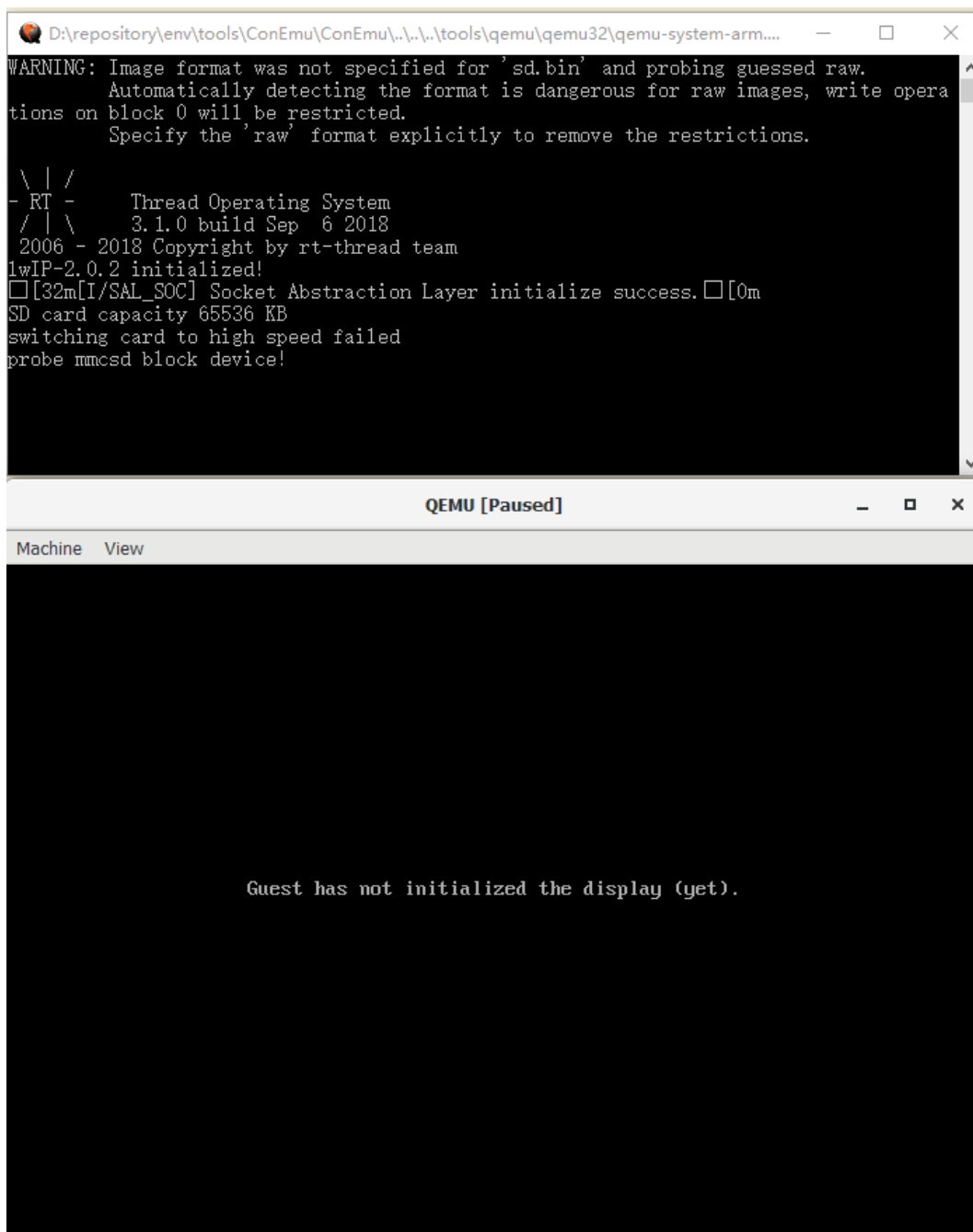


图 8: qemu 调试界面

在 VS Code 里可以使用 GDB 命令，需要在最前面加上 `-exec`。例如 `-exec info registers` 命令可以查看寄存器的内容：



图 9: gdb 查看内存

其他一些主要命令介绍如下所示：

查看内存地址内容：x/<n/f/u> <addr>，各个参数说明如下所示：

- n 是一个正整数，表示需要显示的内存单元的个数，也就是说从当前地址向后显示几个内存单元的内容，一个内存单元的大小由后面的 u 定义
- f 表示显示的格式，参见下面。如果地址所指的是字符串，那么格式可以是 s。其他格式如下表所示：

| 参数 | 描述             |
|----|----------------|
| x  | 按十六进制格式显示变量    |
| d  | 按十进制格式显示变量     |
| u  | 按十六进制格式显示无符号整型 |
| o  | 按八进制格式显示变量     |
| t  | 按二进制格式显示变量     |
| a  | 按十六进制格式显示变量    |
| c  | 按字符格式显示变量      |
| f  | 按浮点数格式显示变量     |

- `u` 表示从当前地址往后请求的字节数，如果不指定的话，GDB 默认是 4 个 bytes。`u` 参数可以用下面的字符来代替，`b` 表示单字节，`h` 表示双字节，`w` 表示四字节，`g` 表示八字节。当我们指定了字节长度后，GDB 会从指内存定的内存地址开始，读写指定字节，并把其当作一个值取出来。
- `addr` 表示一个内存地址。

**注意事项：**严格区分 `n` 和 `u` 的关系，`n` 表示单元个数，`u` 表示每个单元的大小。

示例: `x/3uh 0x54320` 表示从内存地址 `0x54320` 读取内容，`h` 表示以双字节为一个单位，`3` 表示输出三个单位，`u` 表示按十六进制显示。

查看当前程序栈的内容: `x/10x $sp->` 打印 `stack` 的前 10 个元素查看当前程序栈的信息: `info frame`—list general info about the frame 查看当前程序栈的参数: `info args`—lists arguments to the function 查看当前程序栈的局部变量: `info locals`—list variables stored in the frame 查看当前寄存器的值: `info registers`(不包括浮点寄存器) `info all-registers`(包括浮点寄存器) 查看当前栈帧中的异常处理器: `info catch(exception handlers)`

**Tips:** 输入命令时可以只输入每个命令的第一个字母。例如: `info registers` 可以只输入 `ir`。

## 4 注意事项

- 如果在 VS Code 目录中额外添加了文件夹，会导致调试不能够启动。
- 每次开始调试都需要使用 `Env` 工具在 BSP 根目录使用 `code .` 命令打开 VS Code 才能正常调试工程。

## 5 参考

- [Env 工具使用手册](#)

## 6 常见问题

- `Env` 工具的相关问题请参考 `Env` 工具使用手册的常用资料链接小节。
- 提示找不到 `'qemu-system-arm'`。

解决方法: 直接打开 VS Code 调试工程会有这个错误，每次调试请使用 `Env` 工具在 BSP 根目录使用 `code .` 命令打开 VS Code 。

- VS Code 调试选项没有出现 `Debug@windows` 选项或者其他不能调试问题。

解决方法: 请更新 RT-Thread 源代码到 v3.1.0 及以上版本。

- VS Code 出现错误提示: Unable to start debugging.Unexpected GDB output from command “-interpreter-exec console” .....

解决方法: 请修改 qemu-dbg.bat 文件, 特别是有更新源代码的情况下。有问题请按照文档步骤检查一遍是否每一步都做了。