Create a standard **RT-THREAD** project

RT-THREAD Documentation Center

WWW.RT-THREAD.ORG

Friday 28th September, 2018

Table of contents

This application note is used to guide users to create and manage RT-Thread tools in a standard way.

Procedure.

1 Purpose and structure of this paper

1.1 Purpose and Background of this Paper

RT-Thread is completely open source and supports dozens of BSPs, multiple compilers, many basic components, and a growing number of software packages. However, for engineering project development, it only needs to support one or a limited number of MCUs, use a familiar IDE development environment, and use limited peripherals and components. This document is intended to guide users in the full-featured

Based on the RT-Thread version, build the RT-Thread engineering framework according to project requirements.

1.2 Structure of this paper

This article first introduces the necessary preparations, then describes how to use the Env tool provided by RT-Thread to configure the project, and finally describes how to add your own application code and manage your own modules.

2. Preparation

- Download RT-Thread Source code, it is recommended to download the latest version.
- Download RT-Thread Env Tools, it is recommended to download the latest version.
- Install MDK software or IAR software.

3. Standard Engineering Management

3.1 Select **BSP**

After obtaining the RT-Thread source code, users need to find the corresponding BSP according to the development board model they have, and then they can run the default project provided by the BSP. Most BSPs support the MDKfiIAR development environment and GCC compiler, and have provided default MDK and IAR projects.

The following chapters of this article will use the Zhengdian Atom STM32F4 Explorer development board to demonstrate related operations. The MCU model of this development board is STM32F407ZGT6, and the corresponding BSP used is stm32f4xx-HAL, which is in the BSP directory of the RT-Thread source code. This BSP supports development boards with MCU models of stm32f4xx. By default, serial port 2 is used as the shell console output serial port. Users can check the README.md file of the BSP to confirm the serial port used by their development board. This article uses SEGGER JLINK to connect to JTAG debugging and uses USB SLAVE (USART1) for power supply.

Machine Translated by Google

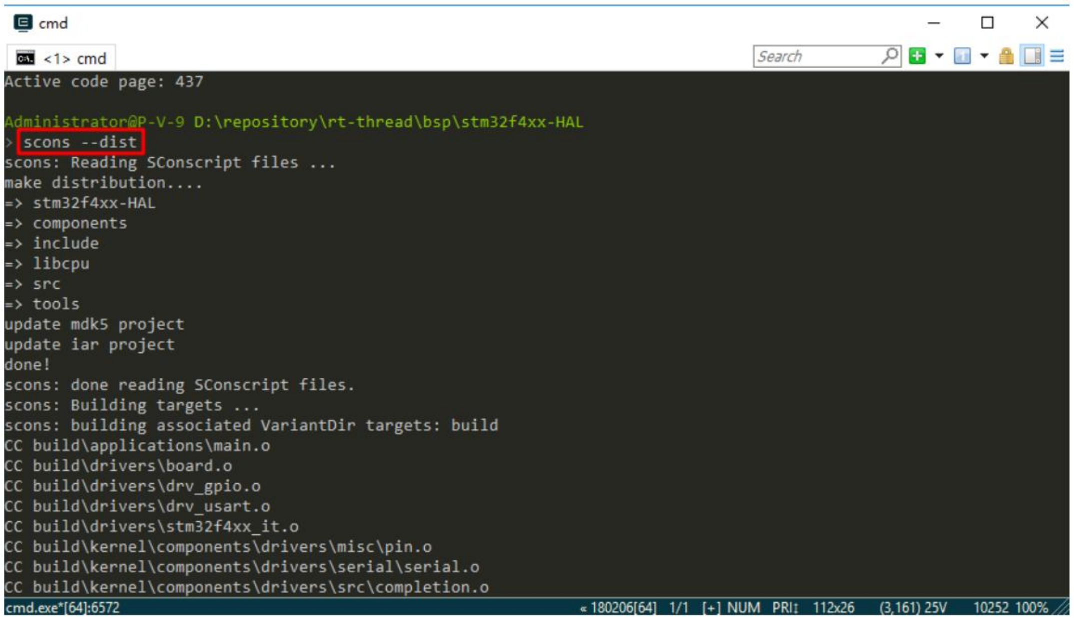
3.2 Build the project framework

Open the Env tool, enter the stm32f4xx-HAL directory, and run the scons --dist command.

The dist directory is generated under the stm32f4xx-HAL BSP directory. This is the directory structure of the development project. The RT-Thread source code

Located in the project folder, it only contains the BSP for stm32f4xx-HAL. You can copy this BSP to any directory at will.

use.

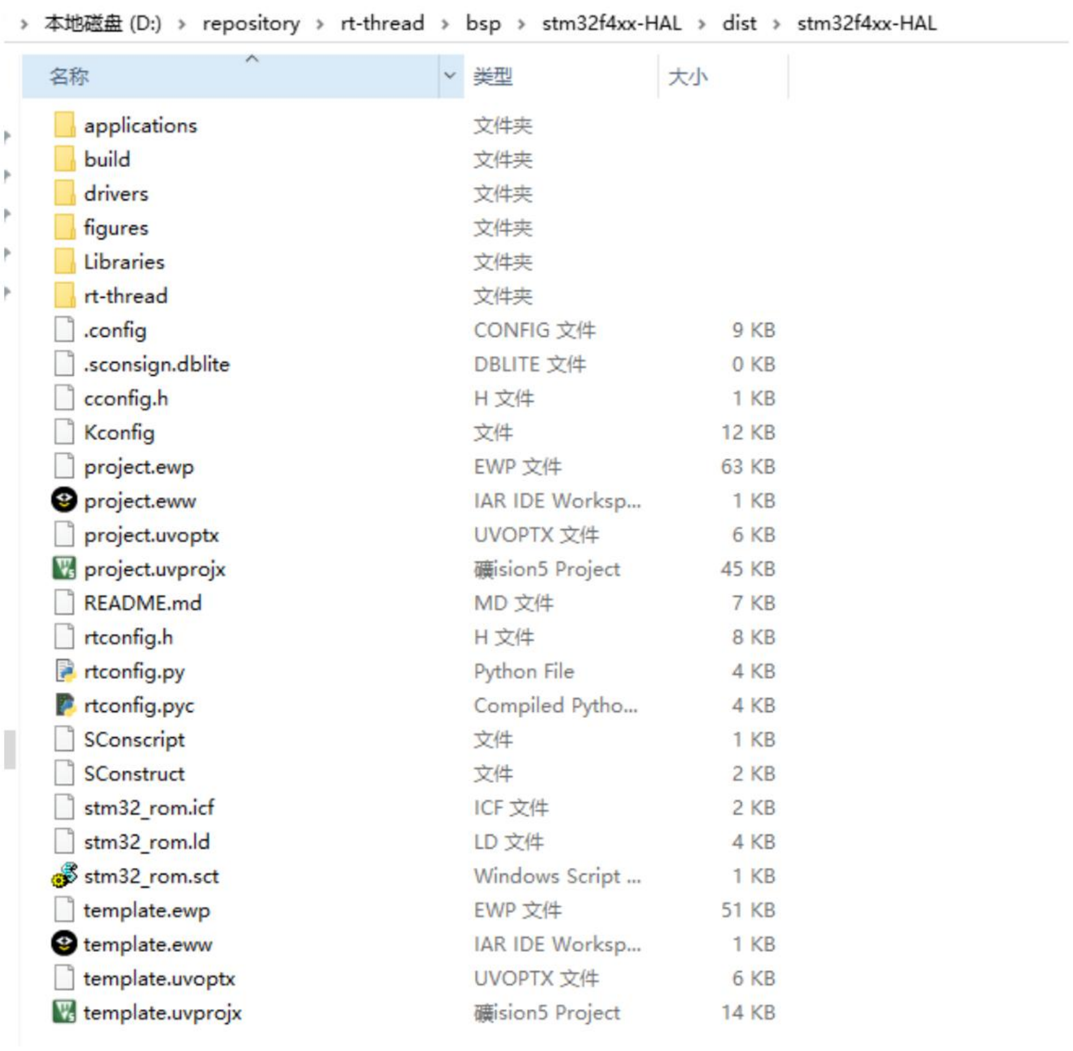


```
Active code page: 437

Administrator@P-V-9 D:\repository\rt-thread\bsp\stm32f4xx-HAL
> scons --dist
scons: Reading SConscript files ...
make distribution....
=> stm32f4xx-HAL
=> components
=> include
=> libcpu
=> src
=> tools
update mdk5 project
update iar project
done!
scons: done reading SConscript files.
scons: Building targets ...
scons: building associated VariantDir targets: build
CC build\applications\main.o
CC build\drivers\board.o
CC build\drivers\drv_gpio.o
CC build\drivers\drv_usart.o
CC build\drivers\stm32f4xx_it.o
CC build\kernel\components\drivers\misc\pin.o
CC build\kernel\components\drivers\serial\serial.o
CC build\kernel\components\drivers\src\completion.o
```

Figure **1:** *scons dist* Order

Enter the stm32f4xx-HAL project directory under the dist directory. The project framework directory structure is shown in the figure below:

Machine Translated by Google



› 本地磁盘 (D:) › repository › rt-thread › bsp › stm32f4xx-HAL › dist › stm32f4xx-HAL

名称	类型	大小
applications	文件夹	
build	文件夹	
drivers	文件夹	
figures	文件夹	
Libraries	文件夹	
rt-thread	文件夹	
.config	CONFIG 文件	9 KB
.sconsign.dblite	DBLITE 文件	0 KB
cconfig.h	H 文件	1 KB
Kconfig	文件	12 KB
project.ewp	EWP 文件	63 KB
project.eww	IAR IDE Worksp...	1 KB
project.uvoptx	UVOPTX 文件	6 KB
project.uvprojx	礦ision5 Project	45 KB
README.md	MD 文件	7 KB
rtconfig.h	H 文件	8 KB
rtconfig.py	Python File	4 KB
rtconfig.pyc	Compiled Pytho...	4 KB
SConscript	文件	1 KB
SConstruct	文件	2 KB
stm32_rom.icf	ICF 文件	2 KB
stm32_rom.ld	LD 文件	4 KB
stm32_rom.sct	Windows Script ...	1 KB
template.ewp	EWP 文件	51 KB
template.eww	IAR IDE Worksp...	1 KB
template.uvoptx	UVOPTX 文件	6 KB
template.uvprojx	礦ision5 Project	14 KB

figure 2: Project framework directory structure

The description of the main directories and files of the project framework is shown in the following table:

File Directory	describe
applications	User application code directory
drivers	The underlying driver provided by RT-Thread
Libraries	Firmware library downloaded from the chip official website
rt-thread	RT-Thread source code
Kconfig	Files used by menuconfig
project.ewww	IAR project files used by users
project.uvprojx	MDK project files used by users
template.uvprojx	MDK project template file
SConscript	Files used by the SCons configuration tool

File Directory	describe
SConstruct	Files used by the SCons configuration tool
README.md	BSP Documentation
rtconfig.h	BSP configuration header file

Note: This command is only supported from the official version of RT-Thread 3.1.0.

3.3 Modify the project template

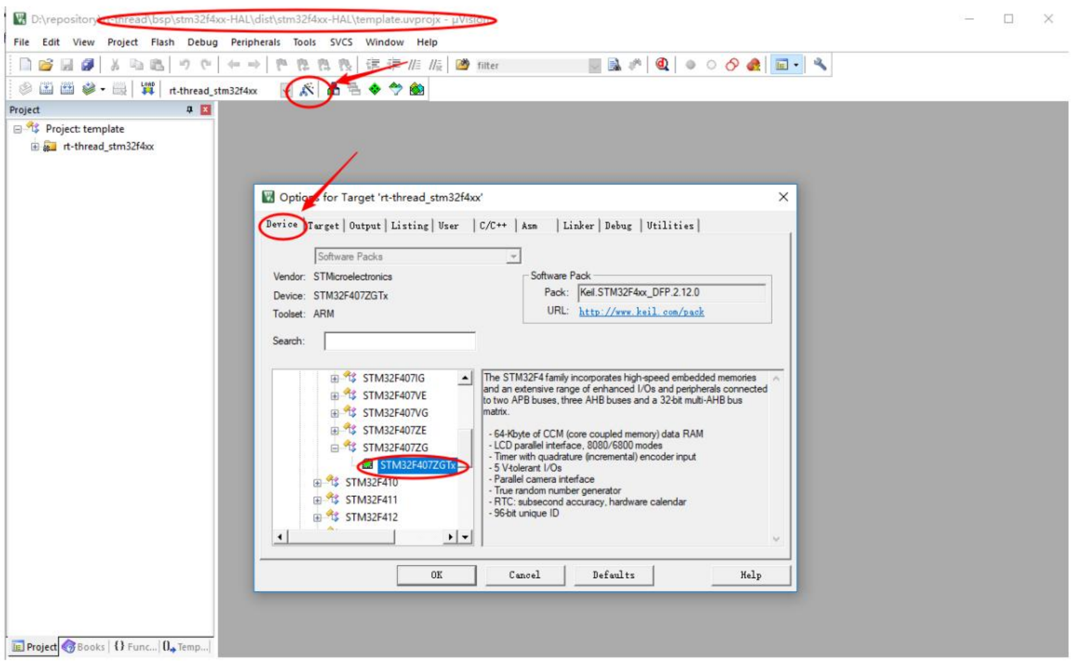
Users generally need to make some project configurations according to their needs, such as configuring the MCU model, setting the

Test options, etc. It is recommended that you modify the project template directly, so that new projects generated using Scons related commands will also include

Modification of templates. The template project of MDK is template.uvprojx. The template project of IAR is template.eww.

Note: Double-clicking to open the IAR project template and modifying it may cause the generated new project to be unusable with the lower version of IAR software.

The following figure shows an example of modifying the chip model of the MDK project template file.


D:\repository\...thread\bsp\stm32f4xx-HAL\dist\stm32f4xx-HAL\template.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
rt-thread_stm32f4xx
Project
Project: template
rt-thread_stm32f4xx
Options for Target 'rt-thread_stm32f4xx'
Device Target Output Listing User C/C++ Asm Linker Debug Utilities
Software Packs
Search:
STM32F407IG
STM32F407VE
STM32F407VG
STM32F407ZE
STM32F407ZG
STM32F407ZGTx
STM32F410
STM32F411
STM32F412
OK Cancel Defaults Help
Project Books Func... Temp...

Vendor: STMicroelectronics
Device: STM32F407ZGTx
Toolset: ARM

Software Pack
Pack: Keil.STM32F4xx_DFP.2.12.0
URL: http://www.keil.com/pack

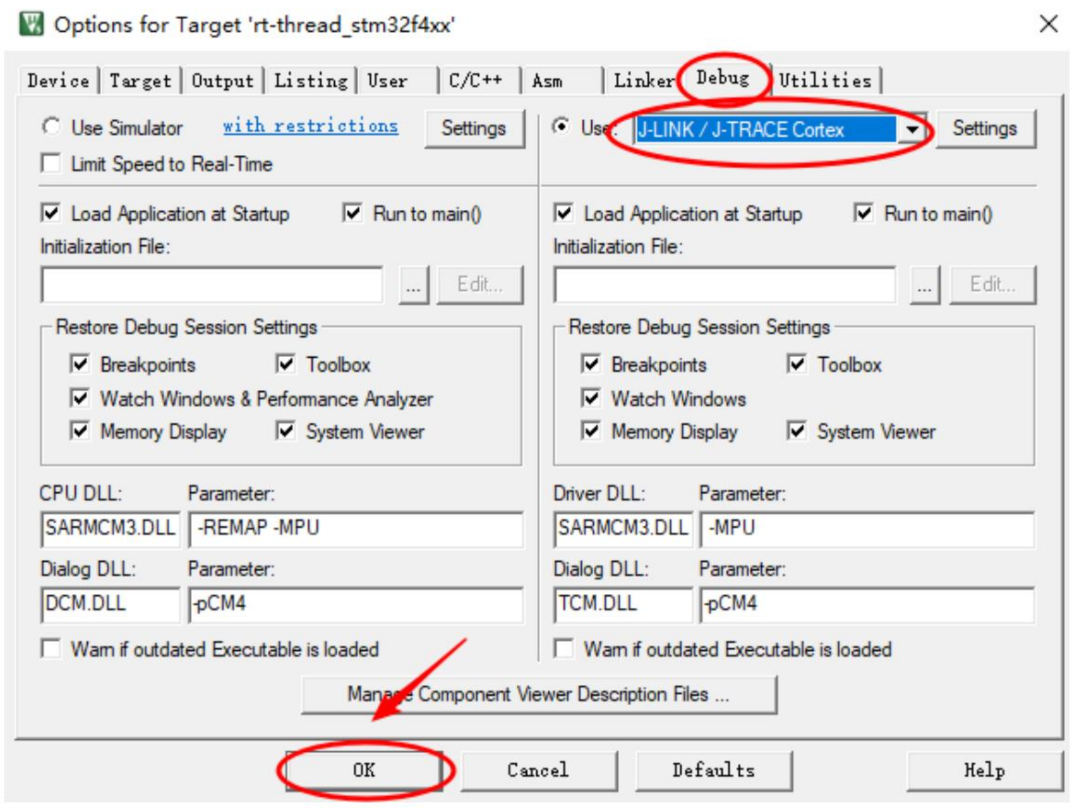
The STM32F4 family incorporates high-speed embedded memories and an extensive range of enhanced I/Os and peripherals connected to two APB buses, three AHB buses and a 32-bit multi-AHB bus matrix.

- 64-Kbyte of CCM (core coupled memory) data RAM
- LCD parallel interface, 8080/6800 modes
- Timer with quadrature (incremental) encoder input
- 5 V-tolerant I/Os
- Parallel camera interface
- True random number generator
- RTC: subsecond accuracy, hardware calendar
- 96-bit unique ID

Figure **3:** *stm32f4xx-HAL* Select chip

The following figure shows the debugging options for the debugging tool you use. You can close it after the relevant configuration is modified.

Formwork engineering.



Options for Target 'rt-thread_stm32f4xx'

Use Simulator with restrictions Settings
Limit Speed to Real-Time
Load Application at Startup ✓ Run to main() ✓
Initialization File:
Restore Debug Session Settings
Breakpoints ✓ Toolbox ✓
Watch Windows & Performance Analyzer ✓
Memory Display ✓ System Viewer ✓
CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU
Dialog DLL: DCM.DLL Parameter: -pCM4
Warn if outdated Executable is loaded

Use: J-LINK / J-TRACE Cortex Settings
Load Application at Startup ✓ Run to main() ✓
Initialization File:
Restore Debug Session Settings
Breakpoints ✓ Toolbox ✓
Watch Windows ✓
Memory Display ✓ System Viewer ✓
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM4
Warn if outdated Executable is loaded

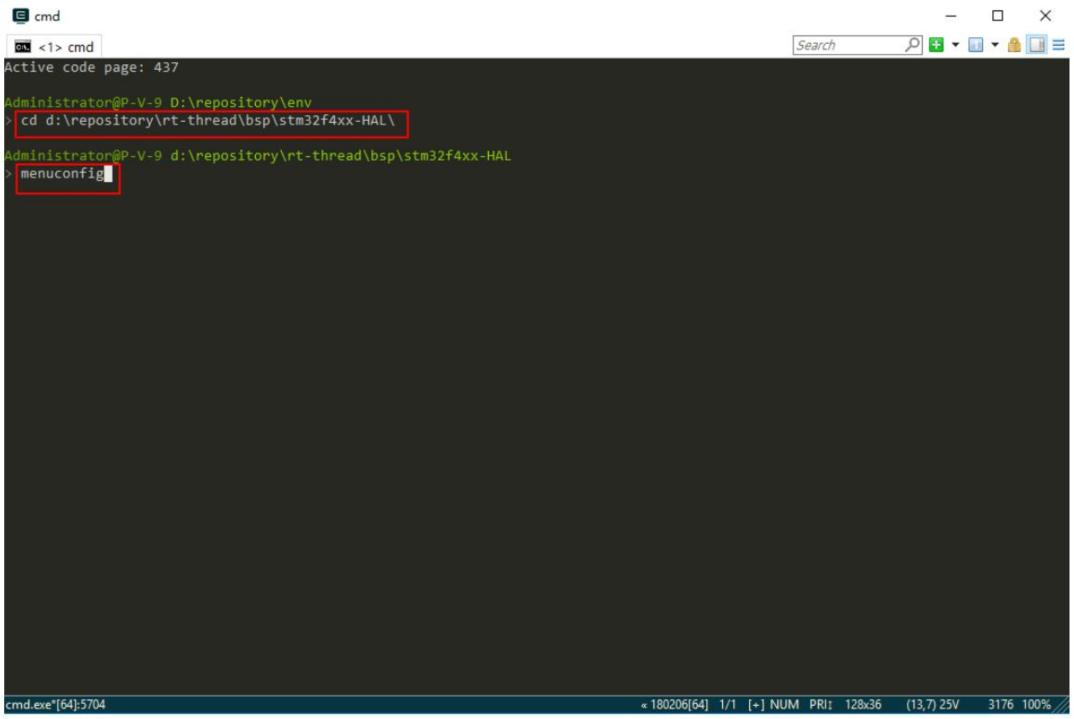
Figure **4:** *stm32f4xx-HAL* Select Debug Options

Note: It is not recommended to modify the project template directly to add source code and header file paths. The following chapters will introduce how to use The Scons tool adds source code and header file paths to the project.

3.4 Configuring and Tailoring **RT-Thread**

Each BSP project has a default configuration, such as the maximum thread priority supported by the system kernel, the system clock frequency, device drivers used, serial ports used by the console, etc. The RT-Thread operating system is highly customizable. Users can use the Env tool to configure and tailor according to their needs.

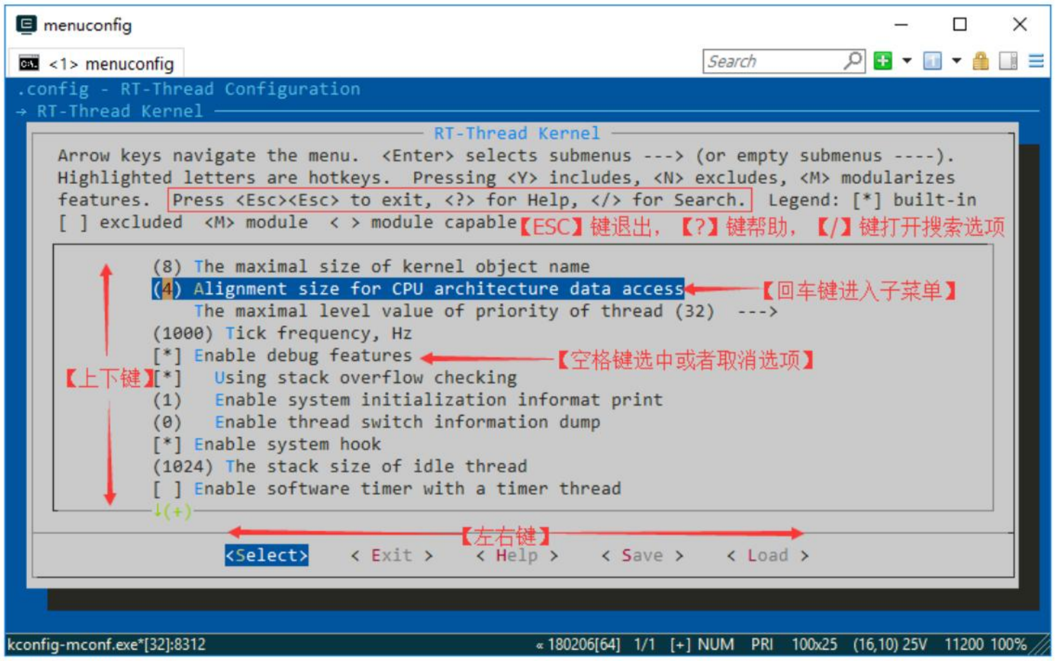
Double-click env.exe to open the configuration interface, and then use the cd d:\repository\rt-thread\bsp\stm32f4xx- HAL command to enter the BSP project directory. Note that cd is followed by the user's own project directory. Then use menuconfig Command to open the configuration interface.



```
Active code page: 437

Administrator@P-V-9 D:\repository\env
> cd d:\repository\rt-thread\bsp\stm32f4xx-HAL\

Administrator@P-V-9 d:\repository\rt-thread\bsp\stm32f4xx-HAL
> menuconfig
```

Figure 5: *Env* Command Line Interface

The commonly used shortcut keys for menuconfig are shown in the figure:



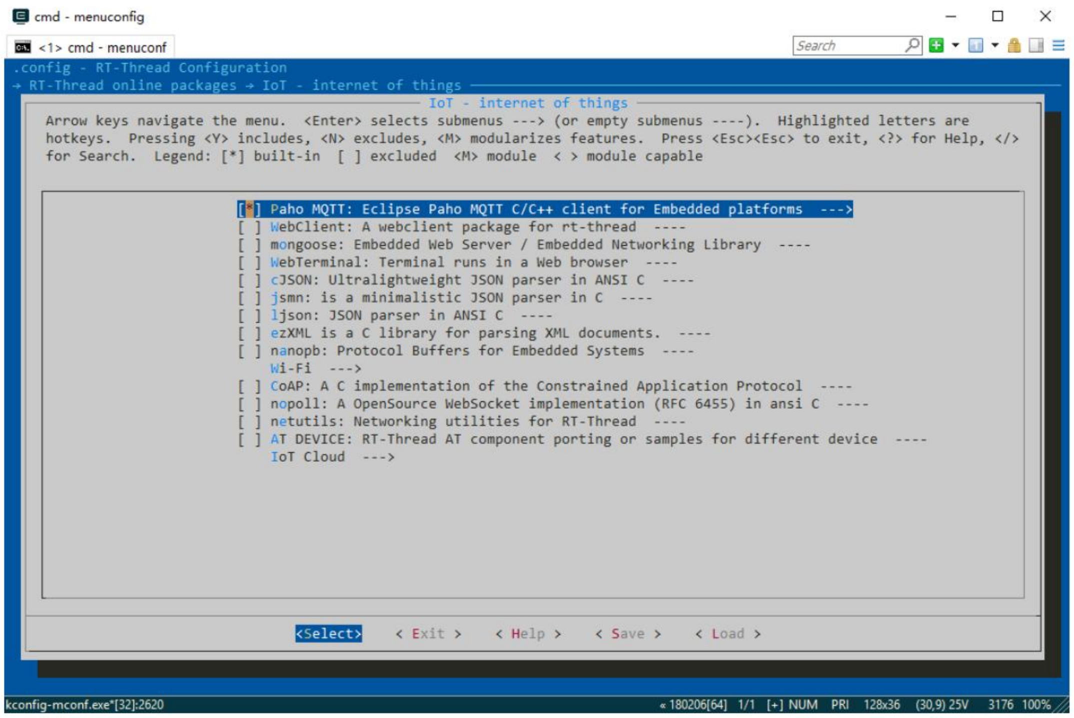
```
.config - RT-Thread Configuration
→ RT-Thread Kernel
                         RT-Thread Kernel
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  [ ] excluded  <M> module  < > module capable

        (8) The maximal size of kernel object name
        (4) Alignment size for CPU architecture data access
            The maximal level value of priority of thread (32)  --->
        (1000) Tick frequency, Hz
        [*] Enable debug features
        [*]   Using stack overflow checking
        (1)   Enable system initialization informat print
        (0)   Enable thread switch information dump
        [*] Enable system hook
        (1024) The stack size of idle thread
        [ ] Enable software timer with a timer thread
        ↓(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 6: *menuconfig* Configuration interface

Machine Translated by Google

3.5 Enabling online software packages

The following figure enables the mqtt related software packages.



```
.config - RT-Thread Configuration
→ RT-Thread online packages → IoT - internet of things
                        IoT - internet of things
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                        [*] Paho MQTT: Eclipse Paho MQTT C/C++ client for Embedded platforms  --->
                        [ ] WebClient: A webclient package for rt-thread  ----
                        [ ] mongoose: Embedded Web Server / Embedded Networking Library  ----
                        [ ] WebTerminal: Terminal runs in a Web browser  ----
                        [ ] cJSON: Ultralightweight JSON parser in ANSI C  ----
                        [ ] jsmn: is a minimalistic JSON parser in C  ----
                        [ ] ljson: JSON parser in ANSI C  ----
                        [ ] ezXML is a C library for parsing XML documents.  ----
                        [ ] nanopb: Protocol Buffers for Embedded Systems  ----
                            Wi-Fi  --->
                        [ ] CoAP: A C implementation of the Constrained Application Protocol  ----
                        [ ] nopoll: A OpenSource WebSocket implementation (RFC 6455) in ansi C  ----
                        [ ] netutils: Networking utilities for RT-Thread  ----
                        [ ] AT DEVICE: RT-Thread AT component porting or samples for different device  ----
                            IoT Cloud  --->

                    <Select>    < Exit >    < Help >    < Save >    < Load >
```

cmd - menuconfig
<1> cmd - menuconf
Search
kconfig-mconf.exe*[32]:2620
« 180206[64] 1/1 [+] NUM PRI 128x36 (30,9) 25V 3176 100%

Figure 7: Enable MQTT Software Packages

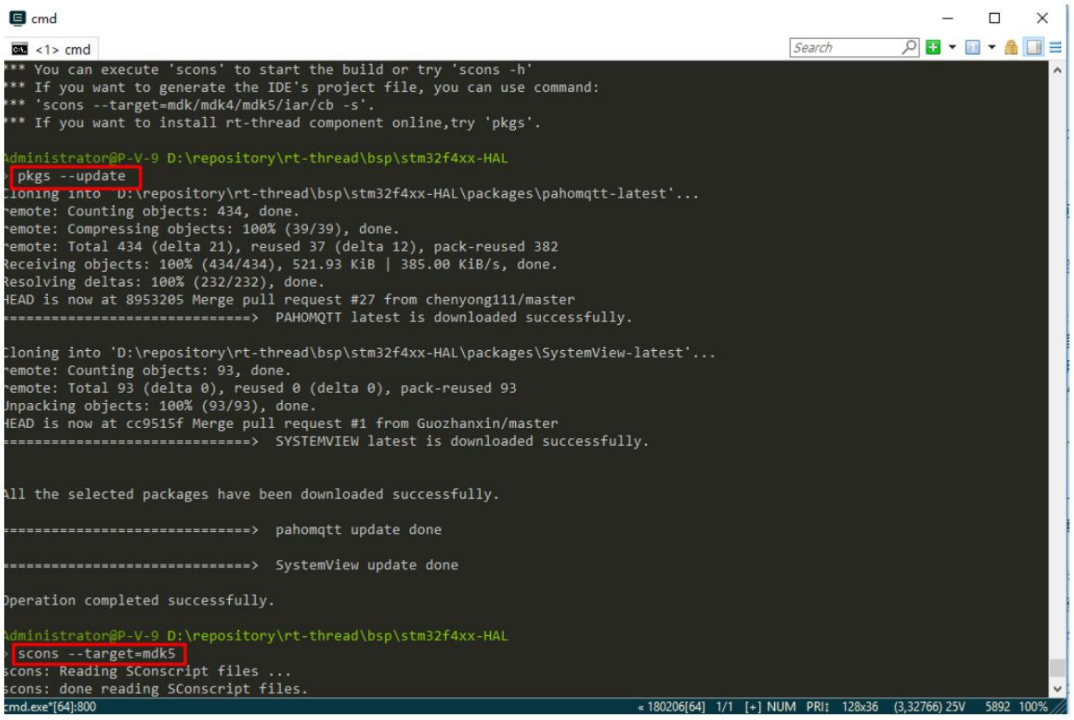
3.6 Generate Project

After selecting the package, you need to use the pkgs --update command to download the package, and then use scons --target=mdk5 command or scons --target=iar command to generate MDK or IAR project.

Add your own code to the project file project.uvprojx or IAR project file project.ewww, or modify

The generated new project will overwrite the previous changes to the project file project.

Machine Translated by Google



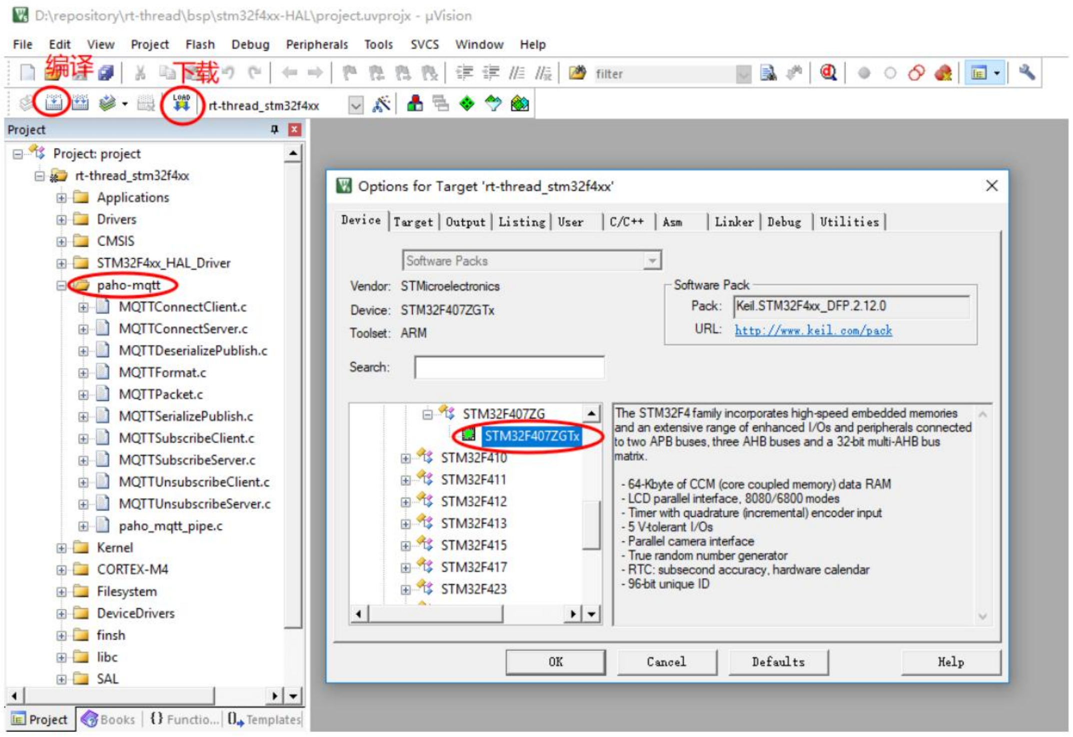
```
*** You can execute 'scons' to start the build or try 'scons -h'
*** If you want to generate the IDE's project file, you can use command:
*** 'scons --target=mdk/mdk4/mdk5/iar/cb -s'.
*** If you want to install rt-thread component online,try 'pkgs'.

Administrator@P-V-9 D:\repository\rt-thread\bsp\stm32f4xx-HAL
> pkgs --update
Cloning into 'D:\repository\rt-thread\bsp\stm32f4xx-HAL\packages\pahomqtt-latest'...
remote: Counting objects: 434, done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 434 (delta 21), reused 37 (delta 12), pack-reused 382
Receiving objects: 100% (434/434), 521.93 KiB | 385.00 KiB/s, done.
Resolving deltas: 100% (232/232), done.
HEAD is now at 8953205 Merge pull request #27 from chenyong111/master
==============================>  PAHOMQTT latest is downloaded successfully.

Cloning into 'D:\repository\rt-thread\bsp\stm32f4xx-HAL\packages\SystemView-latest'...
remote: Counting objects: 93, done.
remote: Total 93 (delta 0), reused 0 (delta 0), pack-reused 93
Unpacking objects: 100% (93/93), done.
HEAD is now at cc9515f Merge pull request #1 from Guozhanxin/master
==============================>  SYSTEMVIEW latest is downloaded successfully.


All the selected packages have been downloaded successfully.

==============================>  pahomqtt update done

==============================>  SystemView update done

Operation completed successfully.

Administrator@P-V-9 D:\repository\rt-thread\bsp\stm32f4xx-HAL
> scons --target=mdk5
scons: Reading SConscript files ...
scons: done reading SConscript files.
```

Figure 8: Download the package and update the project

Open the newly generated MDK project project.uvprojx, and you can see the software related to the paho mqtt we selected.

The source file of the package has been added to the project. The chip model corresponding to the project is also the chip selected based on the project template in the previous article.

model.



The STM32F4 family incorporates high-speed embedded memories and an extensive range of enhanced I/Os and peripherals connected to two APB buses, three AHB buses and a 32-bit multi-AHB bus matrix.

- 64-Kbyte of CCM (core coupled memory) data RAM
- LCD parallel interface, 8080/6800 modes
- Timer with quadrature (incremental) encoder input
- 5 V-tolerant I/Os
- Parallel camera interface
- True random number generator
- RTC: subsecond accuracy, hardware calendar
- 96-bit unique ID

Figure 9: New files have been added to the project

3.7 Verification Engineering

Compile the project, generate the target code, and then download it to the development board to run. This article uses the terminal software PuTTY Receive the data sent by the serial port 2 corresponding to the engineering console, right-click the computer ÿ Properties ÿ Device Manager ÿ Port (COM and LPT), you can view the COM number corresponding to serial port 2, which is COM14 in this article. Open PuTTY and configure it according to the figure below. The baud rate is generally configured to 115200.

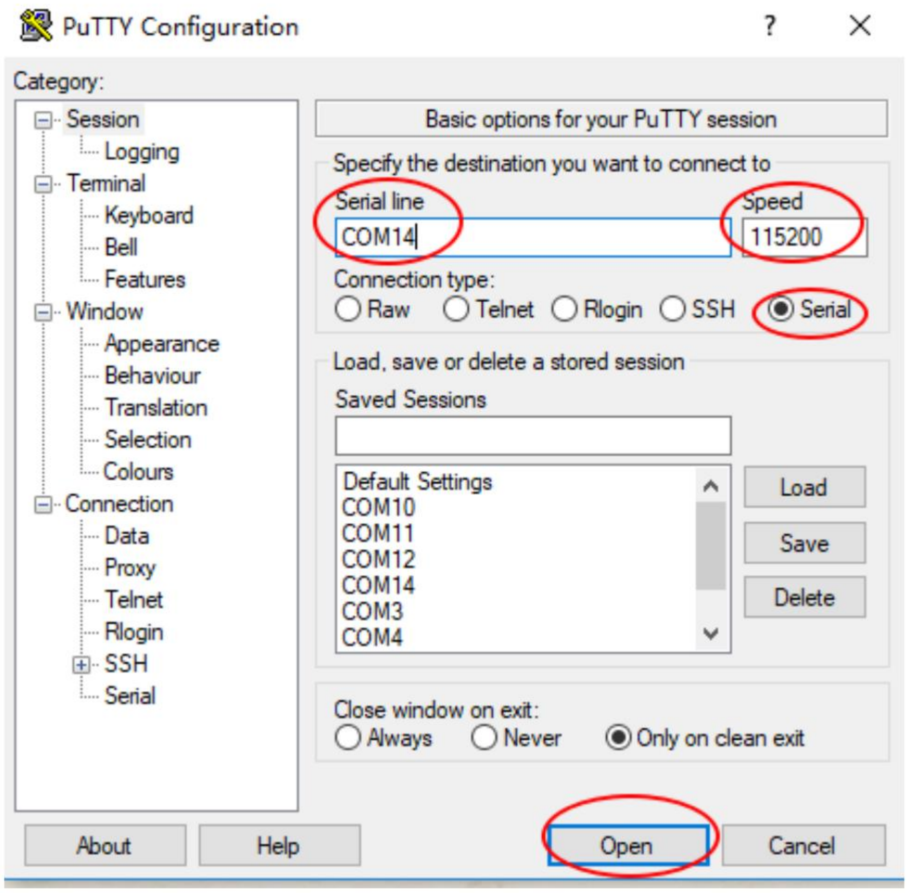
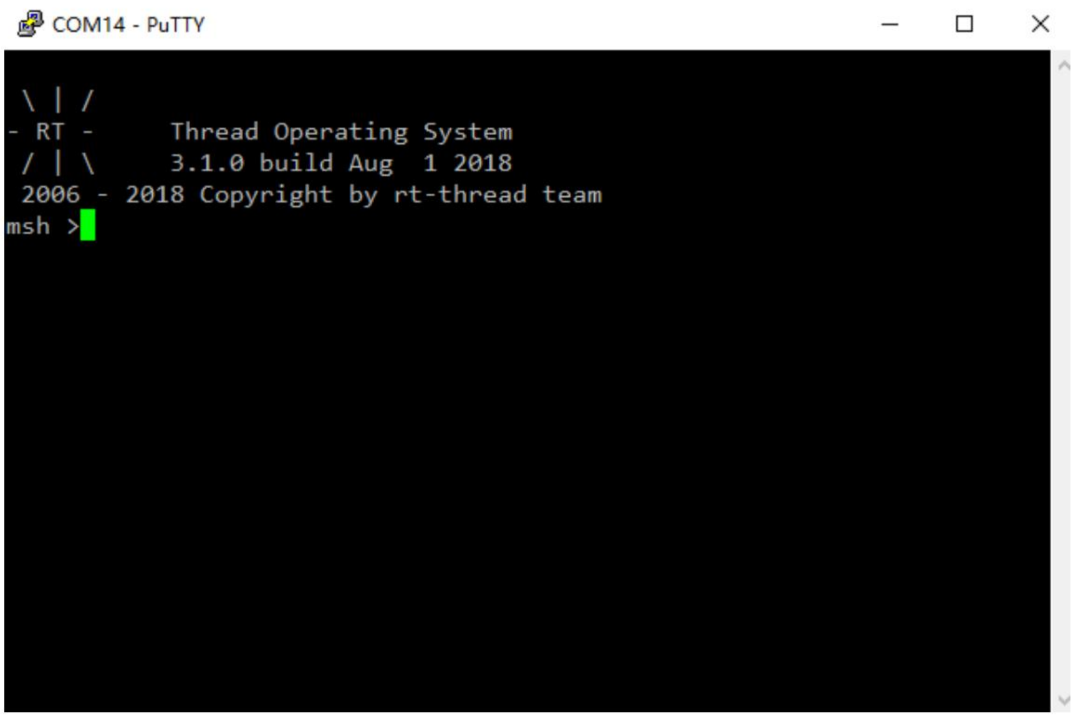

Serial line: COM14
Speed: 115200
Connection type: Raw Telnet Rlogin SSH [✓ Serial]
Close window on exit: Always Never [✓ Only on clean exit]

Figure 10: *PuTTY* Configuration

Click open to open it. After restarting the development board, you will see the startup logo information of RT-Thread.

Machine Translated by Google



```
 \ | /
- RT -     Thread Operating System
 / | \     3.1.0 build Aug  1 2018
 2006 - 2018 Copyright by rt-thread team
msh >
```

Figure **11:** *RT-Thread* Startup *logo* information

3.8 Add files to the project

The applications folder under BSP is used to store the user's own application code. Currently there is only one main.c
If the user's application code is not large, it is recommended that all related source files be placed in this folder.

Two simple files hello.c and hello.h are added to the applications folder.

```
/* file: hello.h */

#ifndef _HELLO_H_
#define _HELLO_H_

int hello_world(void);

#endif /* _HELLO_H_ */
```

```
/* file: hello.c */
#include <stdio.h>
#include <finsh.h>
#include <rtthread.h>

int hello_world(void)
```

```
{
        rt_kprintf("Hello, world!\n");

        return 0;
}

MSH_CMD_EXPORT(hello_world, Hello world!)
```

So how do users add their own application initialization code? RT-Thread uses the main function as the user

User code entry, just add your own code in the main function.

```
#include <rtthread.h>
#include <board.h>
#include "hello.h"

int main(void) {

        /* user app entry */

        hello_world();

        return 0;
}
```

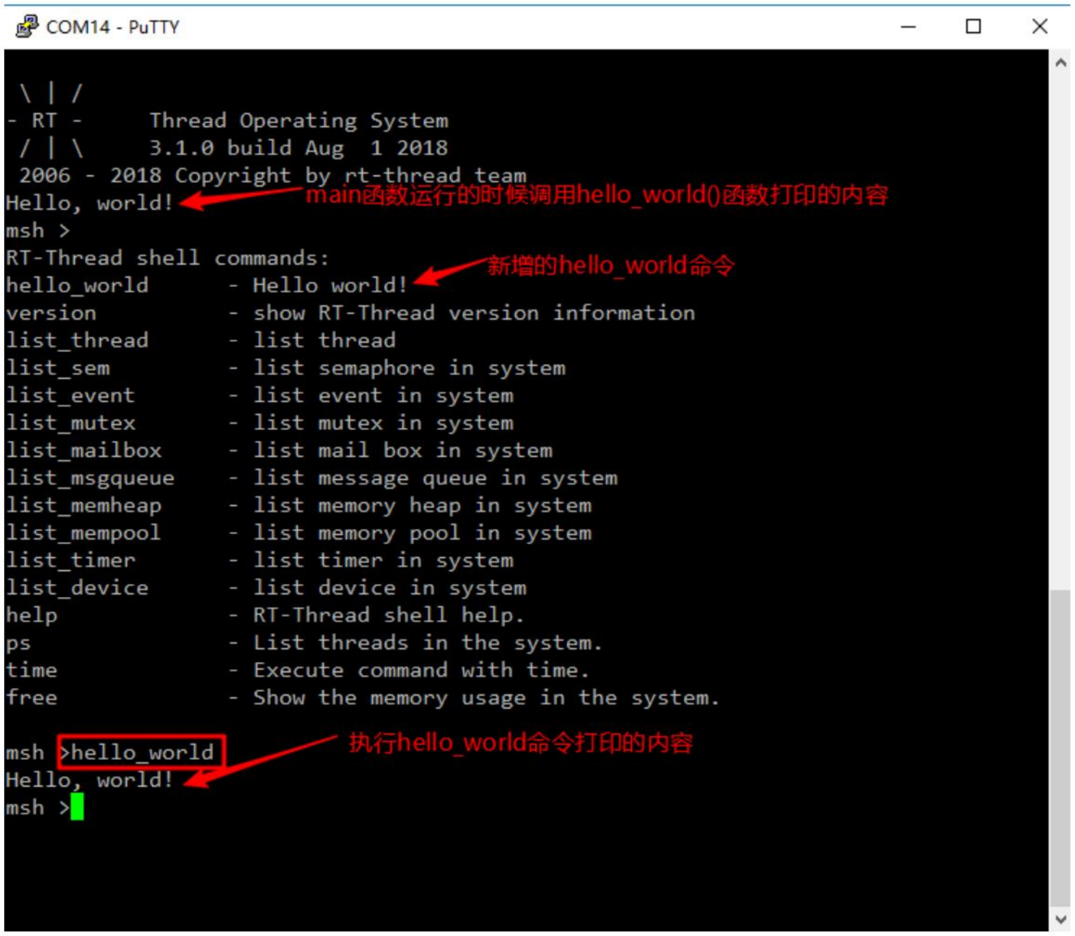
RT-Thread also supports finsh, providing a set of command line interfaces for users to use macros

MSH_CMD_EXPORT() can add a new command and start your own application code on the command line.

Note: The two new files in the applications folder need to use the scons --target=xxx command

Generate a new project to add it to the project. Every time you add a new file, you need to regenerate the project.

The following figure shows two calling results of the application hello_world().


COM14 - PuTTY
main函数运行的时候调用hello_world()函数打印的内容
新增的hello_world命令
执行hello_world命令打印的内容

```
 \ | /
- RT -     Thread Operating System
 / | \     3.1.0 build Aug  1 2018
 2006 - 2018 Copyright by rt-thread team
Hello, world!
msh >
RT-Thread shell commands:
hello_world      - Hello world!
version          - show RT-Thread version information
list_thread      - list thread
list_sem         - list semaphore in system
list_event       - list event in system
list_mutex       - list mutex in system
list_mailbox     - list mail box in system
list_msgqueue    - list message queue in system
list_memheap     - list memory heap in system
list_mempool     - list memory pool in system
list_timer       - list timer in system
list_device      - list device in system
help             - RT-Thread shell help.
ps               - List threads in the system.
time             - Execute command with time.
free             - Show the memory usage in the system.

msh >hello_world
Hello, world!
msh >
```

Figure **12:** *hello-world*

How are the new files in the applications folder added to the project? There is an SConscript file in the applications folder of each BSP, and the functions of these files are similar. RT-Thread uses the SCons build system, and SCons uses SConscript and SConstruct files to organize the source code structure. Generally speaking, a project has only one SConstruct, but there will be multiple SConscripts. In general, an SConscript is placed in each subdirectory where the source code is stored.

The SConscript file is written in Python. The applications file in the stm32f4xx-HAL BSP

The contents of the SConscript file in the folder are as follows. The contents after the # sign are comments.

```
Import('RTT_ROOT')
Import('rtconfig') from building
import *

# str(Dir('#') indicates the directory where the SConstruct of the project is located, that is, D:\repository
    \rt-thread\bsp\stm32f4xx-HAL # os.path.join()
means concatenating the directory represented by `str(Dir('#')` and applications into a complete directory.
```

Machine Translated by Google

```
#That is, get the current directory `D:\repository\rt-thread\bsp\stm32f4xx-HAL\ applications `

cwd              = os.path.join(str(Dir('#')), 'applications')

#Get all C files in the current directory
src = Glob('*.c')

#Save the current directory and the directory where SConstruct is located to the list variable CPPPATH
CPPPATH = [cwd, str(Dir('#'))]

#Create the source files contained in src as the Applications group. If depend is empty, it means that the group is not dependent
     Any macros in rtconfig.h.
# CPPPATH = CPPPATH means adding the directory contained in the CPPPATH variable on the right to the system's header file path
     In the CPPPATH path, the CPPPATH on the left indicates the header file path.
#This Applications group corresponds to the Applications group in the MDK or IAR project.
     Group.
group = DefineGroup('Applications', src, depend = [''], CPPPATH = CPPPATH
     )

Return('group')
```

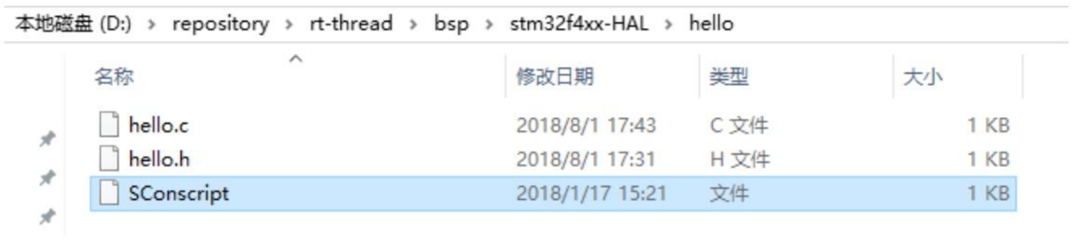
Through this SConscript file, all source files and related header file directories under the applications directory are added to the project without the need for users to add it manually.

3.9 Add modules to the project

As mentioned in the previous article, if you don't have many source code files, it is recommended that all source code files be placed in the applications folder. If the user has a lot of source code and wants to create his own project module, or needs to use other modules he has obtained, what is the appropriate way? The previous article briefly introduced the SConscript file in the applications folder. This section will simply demonstrate how to implement your own SConscript file to manage project files.

3.9.1. Add source code

Taking the hello.c and hello.h mentioned above as an example, they need to be managed as a separate module and have their own group in the MDK or IAR project file, and you can choose whether to use this module through menuconfig. Add a new hello folder under BSP.



本地磁盘 (D:) › repository › rt-thread › bsp › stm32f4xx-HAL › hello

名称	修改日期	类型	大小
hello.c	2018/8/1 17:43	C 文件	1 KB
hello.h	2018/8/1 17:31	H 文件	1 KB
SConscript	2018/1/17 15:21	文件	1 KB

Figure 13: Added *hello* Modules

3.9.2. Writing **SConscript** Files

You will notice that there is an SConscript file in the folder. If you want to add some of your own source code to it,
In the SCons compilation environment, you can generally create or modify existing SConscript files. SConscript files can control
Source code files are added, and the file group can be specified (similar to the concept of group in IDEs such as MDK/IAR)
Similar). The contents of the SConscript file for the newly added hello module are as follows. The contents after the # sign are comments.

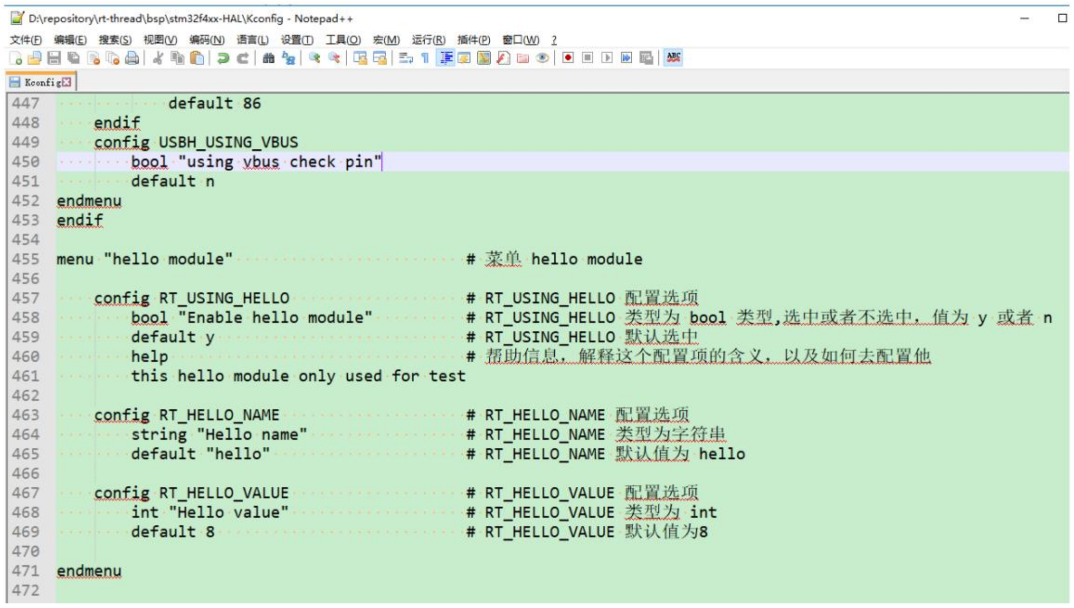
```
# -*- coding: UTF-8 -*- in Chinese                  # The encoding format of the specified file is UTF-8. The file can be

Import('RTT_ROOT')
Import('rtconfig')
from building import *

cwd            = GetCurrentDir()                    #Assign the current path to the string cwd
include_path = [cwd]                                #Save the current header file path to the list variable
     include_path in
src            = []                                 #Define a list variable src

if GetDepend(['RT_USING_HELLO']): # hello.c depends on macro RT_USING_HELLO
    src += ['hello.c']                              #Save the hello.c string to the list variable src
          middle

#Create the source files included in src as the hello group. If depend is empty, it means that the group does not depend on rtconfig.
     h any macro.
# CPPPATH = include_path means adding the current directory to the system's header file path
group = DefineGroup('hello', src, depend = [''], CPPPATH = include_path)

Return('group')
```

A new hello group is created through the above few simple Python codes, and can be defined through macros
Controls the source files to be added to the group. For more information about SCons, please refer to the Scons official documentation Or refer to
Scons build tool manual provided by RT-Thread .

3.9.3. Add **menuconfig** menu

So how is the custom macro RT_USING_HELLO defined? Here we introduce a new
File Kconfig. Kconfig is used to configure the kernel. The menuconfig command reads the various Kconfig files of the project.
Generate a configuration interface for users to configure the kernel, and finally all configuration-related macro definitions will be automatically saved to the BSP directory
In the rtconfig.h file, each BSP has an rtconfig.h file, which is the configuration information of this BSP.

There is already a Kconfig file for this BSP in the stm32f4xx-HAL BSP directory.
If BSP does not have it, you can create one yourself. We can add the configuration options we need based on this file.
The hello module adds the following configuration options. The # sign is followed by a comment.



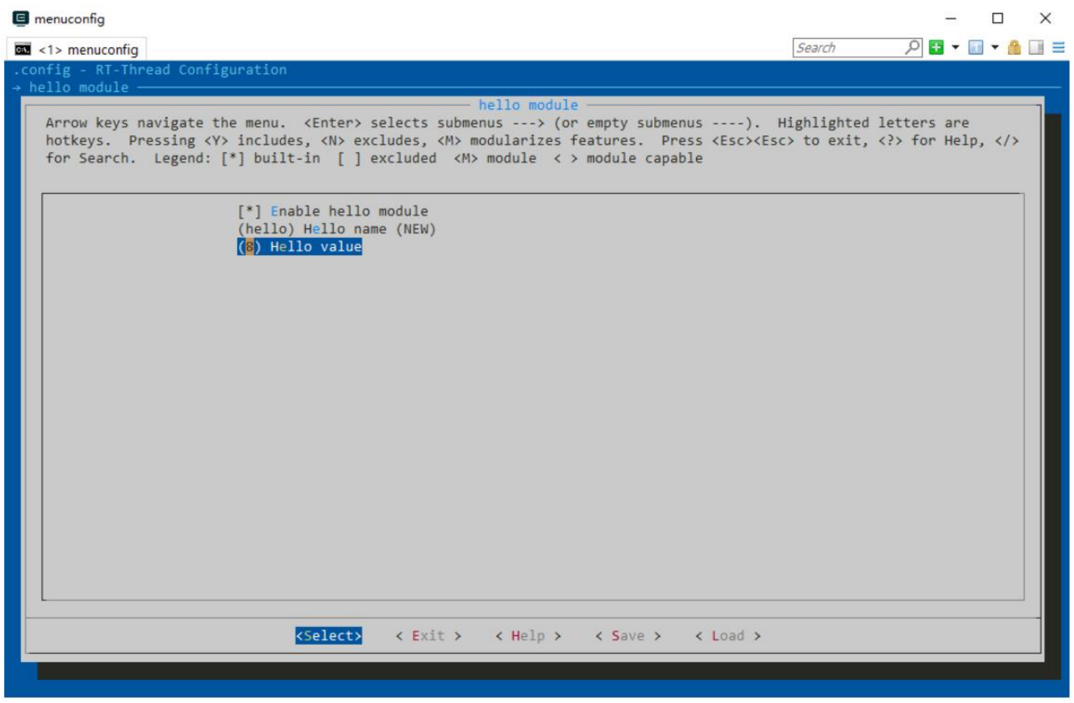
D:\repository\rt-thread\bsp\stm32f4xx-HAL\Kconfig - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
Kconfig

```
                default 86
    endif
    config USBH_USING_VBUS
        bool "using vbus check pin"
        default n
endmenu
endif

menu "hello module"                            # 菜单 hello module

    config RT_USING_HELLO                      # RT_USING_HELLO 配置选项
        bool "Enable hello module"             # RT_USING_HELLO 类型为 bool 类型,选中或者不选中, 值为 y 或者 n
        default y                              # RT_USING_HELLO 默认选中
        help                                   # 帮助信息, 解释这个配置项的含义, 以及如何去配置他
        this hello module only used for test

    config RT_HELLO_NAME                       # RT_HELLO_NAME 配置选项
        string "Hello name"                    # RT_HELLO_NAME 类型为字符串
        default "hello"                        # RT_HELLO_NAME 默认值为 hello

    config RT_HELLO_VALUE                      # RT_HELLO_VALUE 配置选项
        int "Hello value"                      # RT_HELLO_VALUE 类型为 int
        default 8                              # RT_HELLO_VALUE 默认值为8

endmenu
```

Figure 14: Added *hello* Module configuration information

After using the Env tool to enter the stm32f4xx-HAL BSP directory, use the menuconfig command at the bottom of the main page

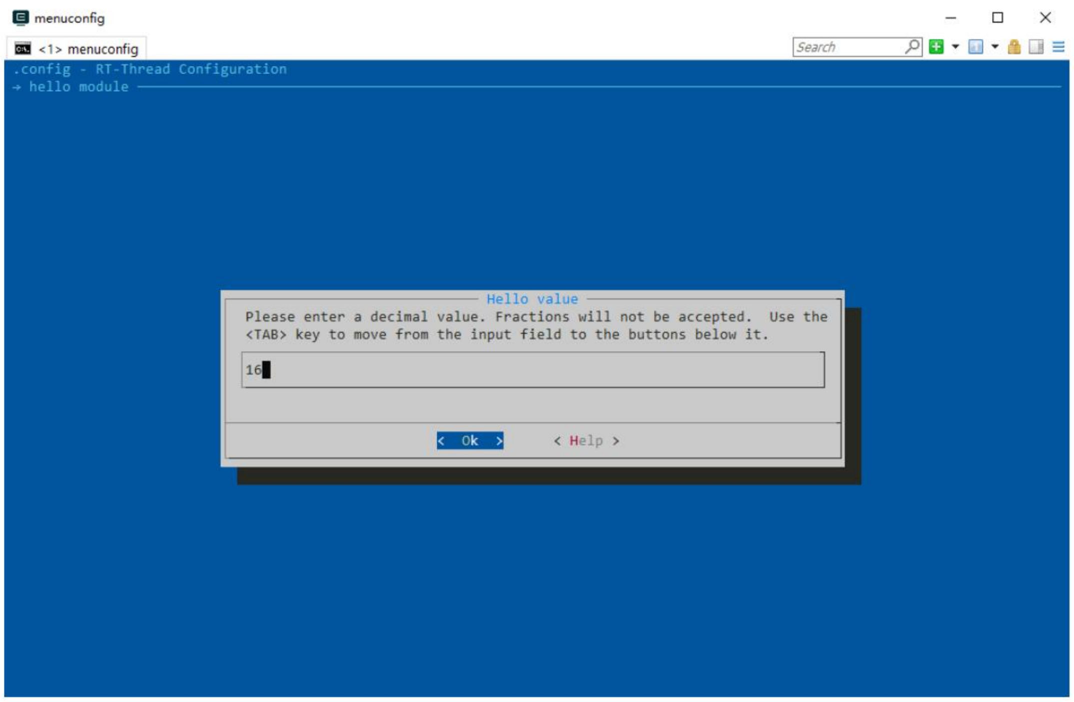
You can see the configuration menu of the newly added hello module. After entering the menu, it will be displayed as shown in the figure below.



```
.config - RT-Thread Configuration
→ hello module
                                   hello module
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                        [*] Enable hello module
                        (hello) Hello name (NEW)
                        (8) Hello value

                  <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 15: Added *hello* Module Configuration Menu

We can also modify the value of hello value.

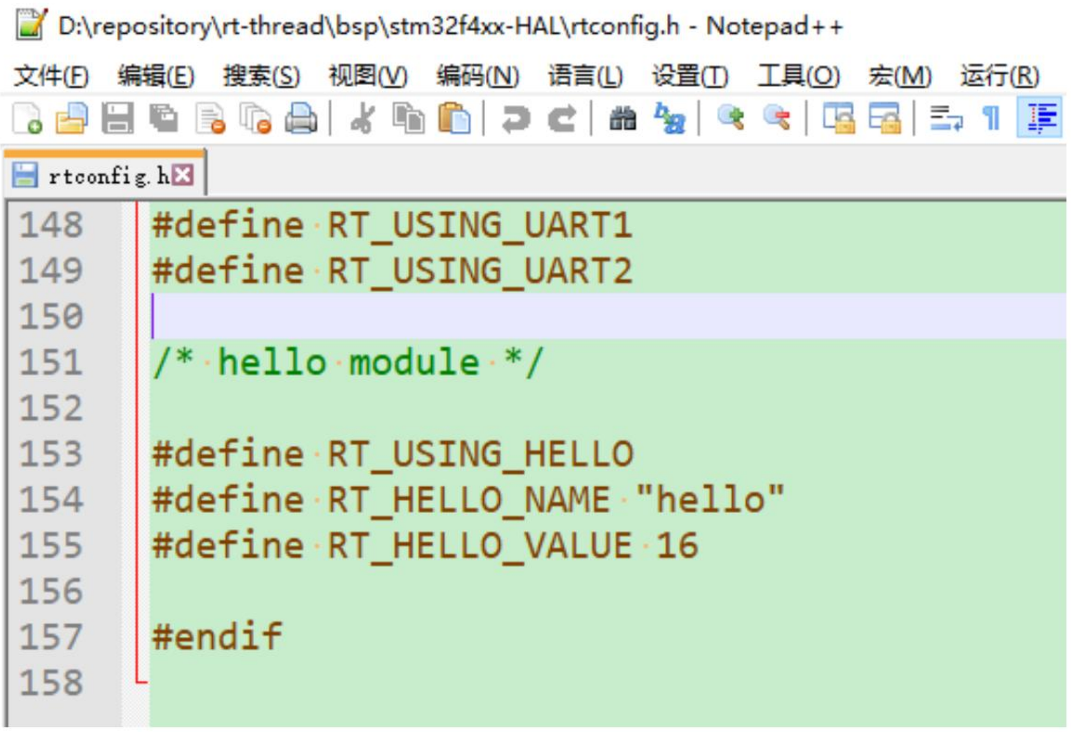


```
.config - RT-Thread Configuration
→ hello module

Hello value
Please enter a decimal value. Fractions will not be accepted.  Use the
<TAB> key to move from the input field to the buttons below it.

16

< Ok >      < Help >
```

Figure 16: Added *hello* Module Configuration Menu

After saving the configuration, exit the configuration interface and open the rtconfig.h file in the stm32f4xx-HAL BSP directory to see

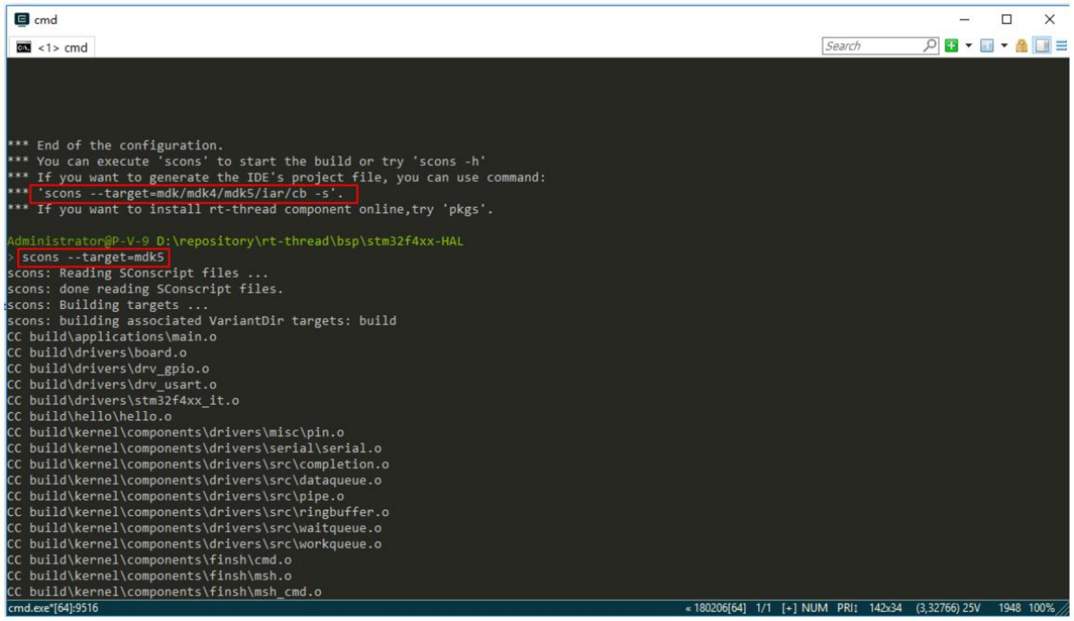
The configuration information for the hello module is already there.


D:\repository\rt-thread\bsp\stm32f4xx-HAL\rtconfig.h - Notepad++
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R)
rtconfig.h

```
#define RT_USING_UART1
#define RT_USING_UART2

/* hello module */

#define RT_USING_HELLO
#define RT_HELLO_NAME "hello"
#define RT_HELLO_VALUE 16

#endif
```

Figure 17: Added *hello* Module configuration information

Note: Each time menuconfig configuration is completed, use scons --target=XXX to generate a new project.

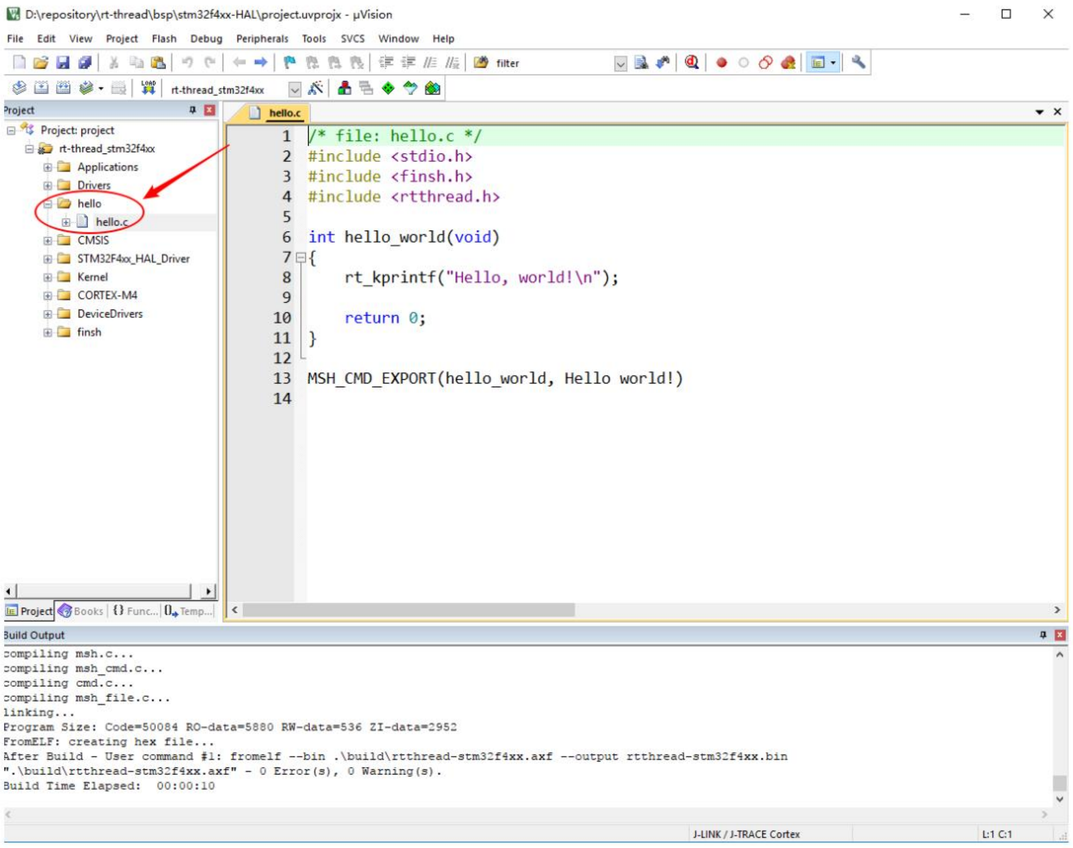


```
*** End of the configuration.
*** You can execute 'scons' to start the build or try 'scons -h'
*** If you want to generate the IDE's project file, you can use command:
*** 'scons --target=mdk/mdk4/mdk5/iar/cb -s'.
*** If you want to install rt-thread component online,try 'pkgs'.

Administrator@P-V-9 D:\repository\rt-thread\bsp\stm32f4xx-HAL
> scons --target=mdk5
scons: Reading SConscript files ...
scons: done reading SConscript files.
scons: Building targets ...
scons: building associated VariantDir targets: build
CC build\applications\main.o
CC build\drivers\board.o
CC build\drivers\drv_gpio.o
CC build\drivers\drv_usart.o
CC build\drivers\stm32f4xx_it.o
CC build\hello\hello.o
CC build\kernel\components\drivers\misc\pin.o
CC build\kernel\components\drivers\serial\serial.o
CC build\kernel\components\drivers\src\completion.o
CC build\kernel\components\drivers\src\dataqueue.o
CC build\kernel\components\drivers\src\pipe.o
CC build\kernel\components\drivers\src\ringbuffer.o
CC build\kernel\components\drivers\src\waitqueue.o
CC build\kernel\components\drivers\src\workqueue.o
CC build\kernel\components\finsh\cmd.o
CC build\kernel\components\finsh\msh.o
CC build\kernel\components\finsh\msh_cmd.o
```

cmd
<1> cmd
Search
cmd.exe*[64]:9516
« 180206[64] 1/1 [+] NUM PRI↕ 142x34 (3,32766) 25V 1948 100%

Figure 18: Create a new project

After opening the newly generated MDK5 project file project.uvprojx, you can see a new hello group, as well as This group contains the hello.c file.



```c
/* file: hello.c */
#include <stdio.h>
#include <finsh.h>
#include <rtthread.h>

int hello_world(void)
{
    rt_kprintf("Hello, world!\n");

    return 0;
}

MSH_CMD_EXPORT(hello_world, Hello world!)
```

```
compiling msh.c...
compiling msh_cmd.c...
compiling cmd.c...
compiling msh_file.c...
linking...
Program Size: Code=50084 RO-data=5880 RW-data=536 ZI-data=2952
FromELF: creating hex file...
After Build - User command #1: fromelf --bin .\build\rtthread-stm32f4xx.axf --output rtthread-stm32f4xx.bin
".\build\rtthread-stm32f4xx.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:10
```

Figure **19**: Create a new project

The above is just a simple list of adding your own configuration options in the Kconfig file. Users can also refer to Env usage.

Use the Using Env in your project section of the manual Or view Kconfig related documents on the Internet to learn more about Kconfig.

Use method.