
Create a standard **RT-THREAD** project

RT-THREAD Document Center

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



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. Standard engineering management.	1
3.1 Select BSP.	1
3.2 Build the project framework.	2
3.3 Modify the project template.	4
3.4 Configuring and tailoring RT-Thread.	5
3.5 Enable online software packages.	7
3.6 Generate the project.	7
3.7 Verification Engineering.	9
3.8 Add files to the project.	10
3.9 Add modules to the project.	13
3.9.1. Add source code.	13
3.9.2. Writing SConscript files.	14
3.9.3. Added menuconfig menu.	14

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, numerous basic components, and a growing number of software packages. However, for engineering project development, it is only necessary 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

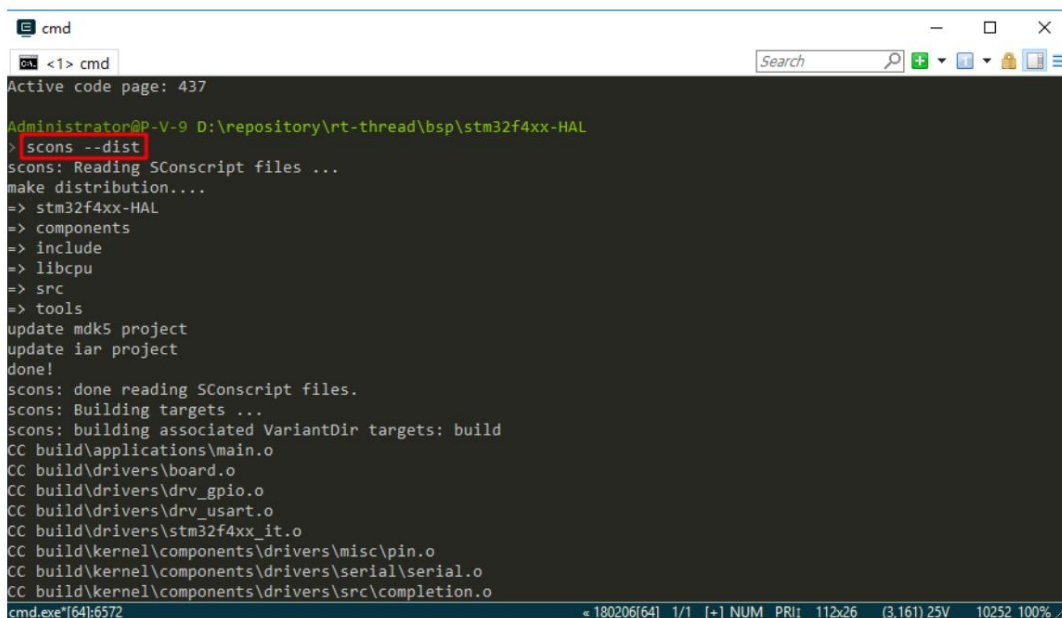
After obtaining the RT-Thread source code, users need to find the corresponding BSP based on their development board model and run the default project provided by the BSP. Most BSPs support the MDK/IAR development environment and the GCC compiler, and already provide default MDK and IAR projects.

The following sections of this article will demonstrate the relevant operations using the Zhengdian Atom STM32F4 Explorer development board. This development board uses the STM32F407ZGT6 MCU and the corresponding BSP, stm32f4xx-HAL, which is available in the BSP directory of the RT-Thread source code. This BSP supports development boards with the stm32f4xx MCU and uses serial port 2 for shell console output by default. Users can check the BSP's README.md file to confirm the serial port used for their development board. This article uses a SEGGER JLINK for JTAG debugging and a USB slave (USART1) for power.

3.2 Building the project framework

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

Generate the dist directory under the stm32f4xx-HAL BSP directory. This is the directory structure of the development project. RT-Thread source code Located in the project folder, it only contains the BSP of stm32f4xx-HAL. You can copy this BSP to any directory at will. use.



```
cmd
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
cmd.exe*164:6572
```

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:

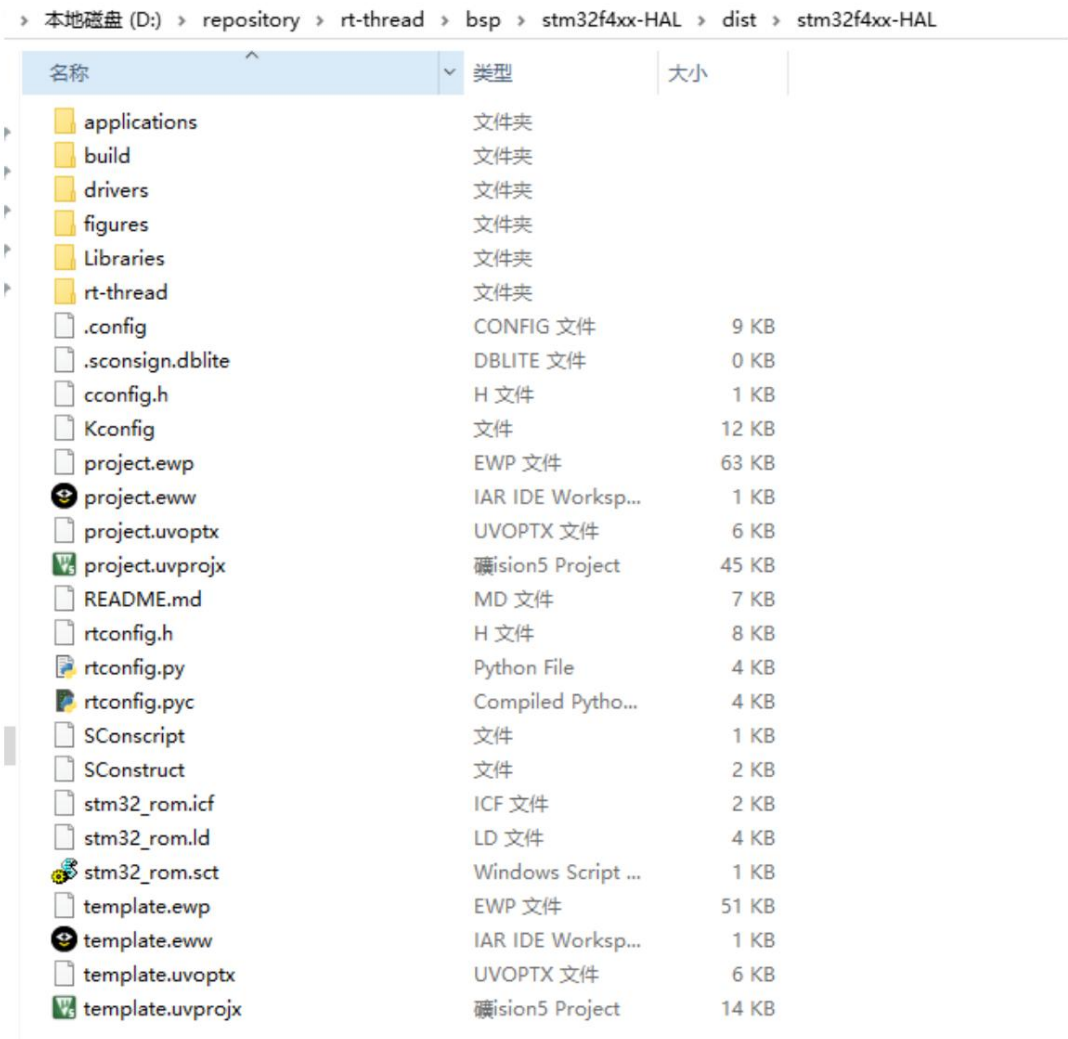


Figure 2: Project framework directory structure

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

Files/Directories	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

Files/Directories	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 starting 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 Modifications to the template. The template project for MDK is template.uvprojx. The template project for 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 by the lower version of IAR software.

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

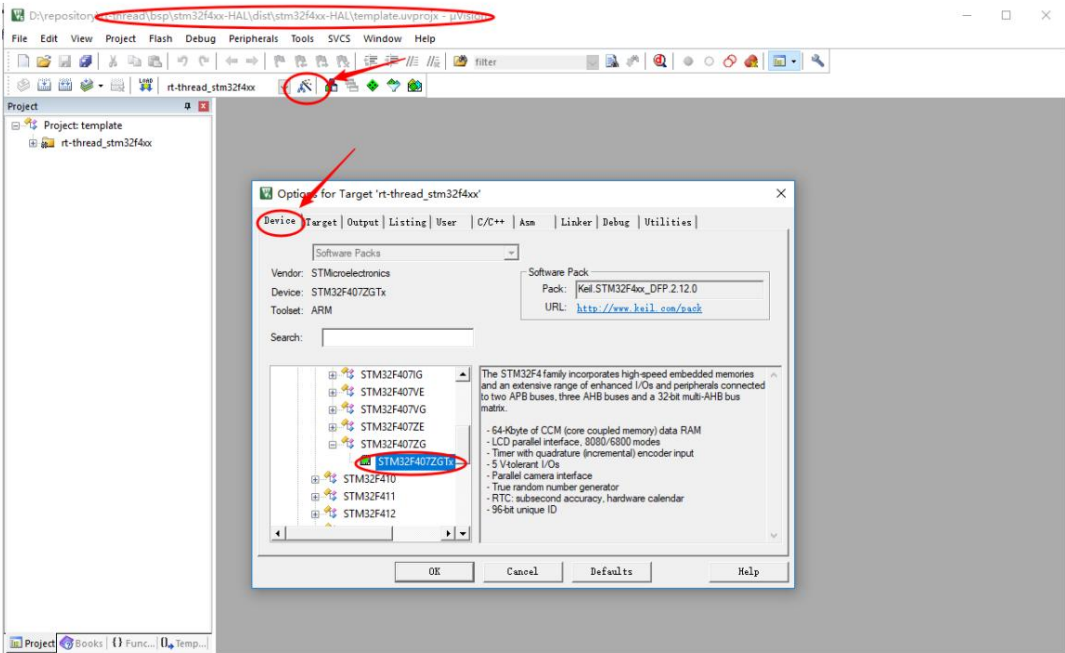


Figure 3: stm32f4xx-HAL Select chip

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

Template engineering.

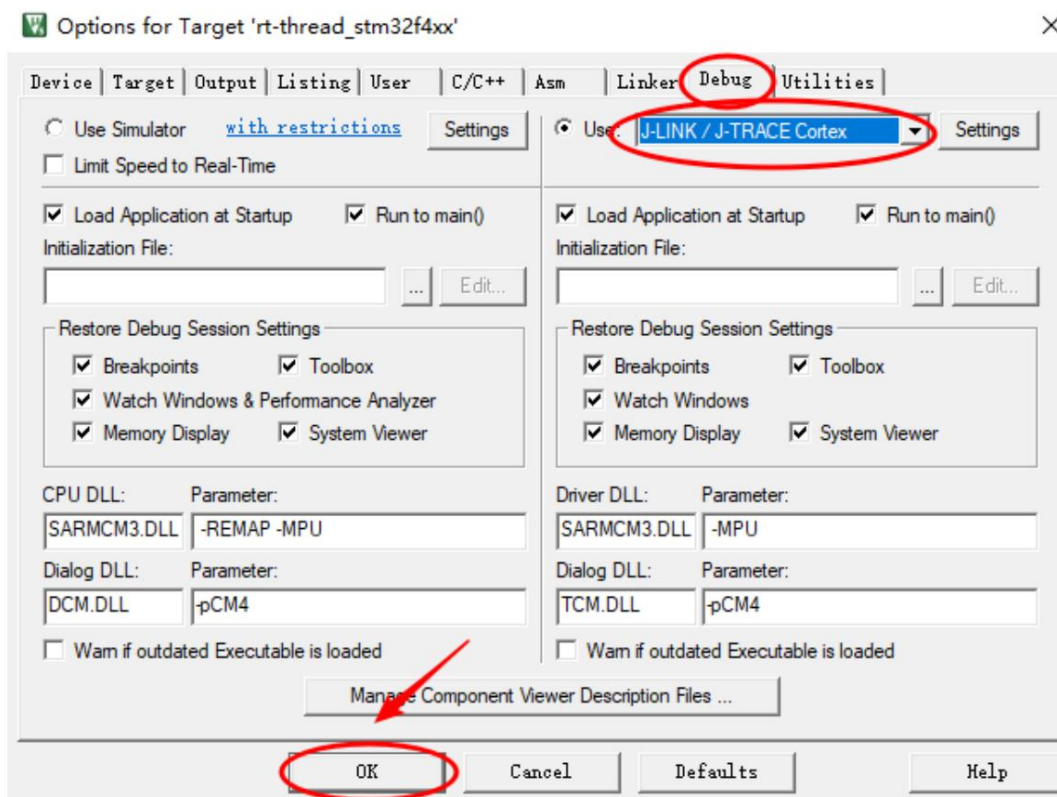


Figure 4: stm32f4xx-HAL

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 the use of The Scons tool adds source code and header file paths to the project.

3.4 Configuring and Tailoring RT-Thread

Each project under BSP has default configuration, such as the maximum thread priority supported by the system kernel, system clock Frequency, device driver used, serial port 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, then use the `cd d:\repository\rt-thread\bsp\stm32f4xx- HAL` command to enter the BSP project directory. Note that the `cd` is followed by the user's own project directory. Then use `menuconfig` command to open the configuration interface.

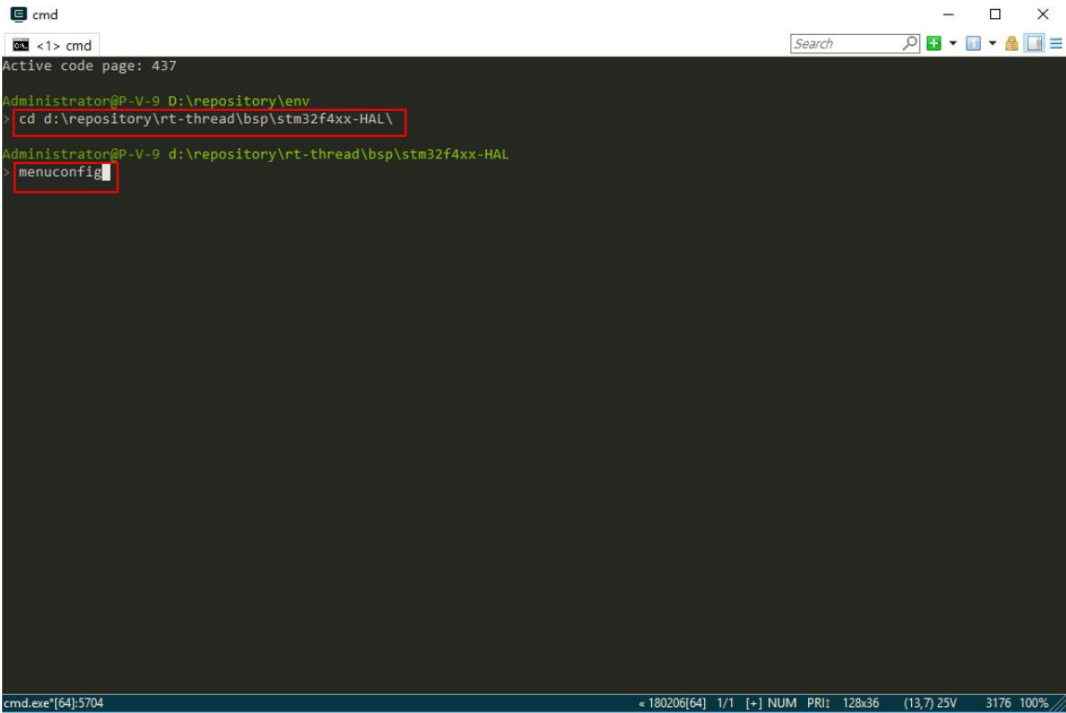


Figure 5: Env Command Line Interface

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

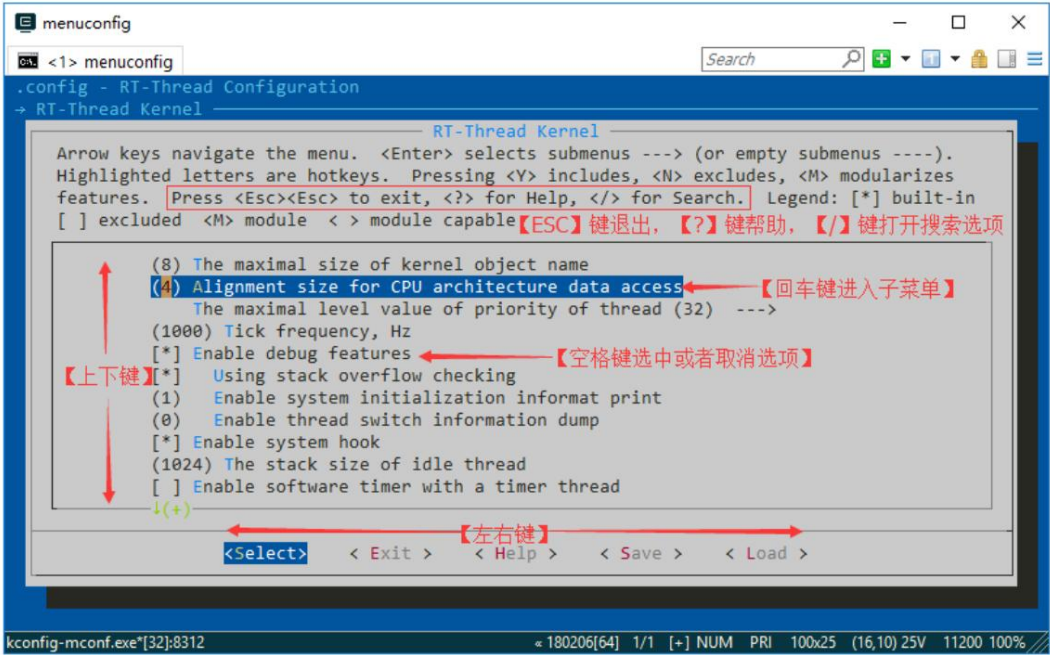


Figure 6: menuconfig Configuration interface

3.5 Enabling the Online Software Package

The following figure enables MQTT related software packages.

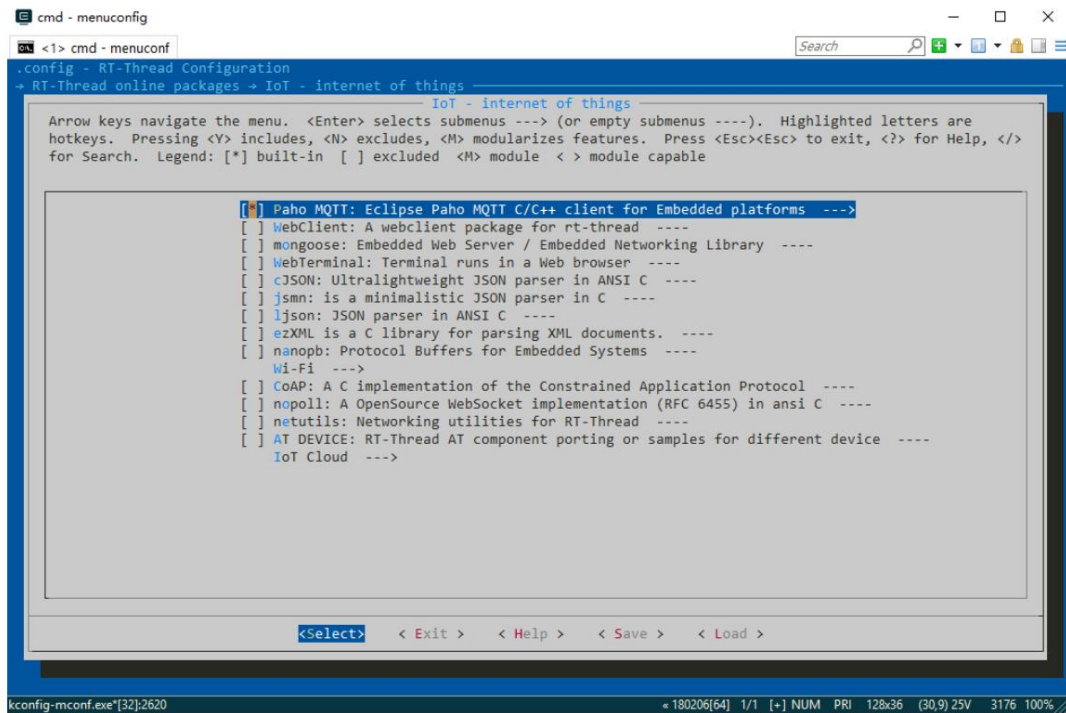


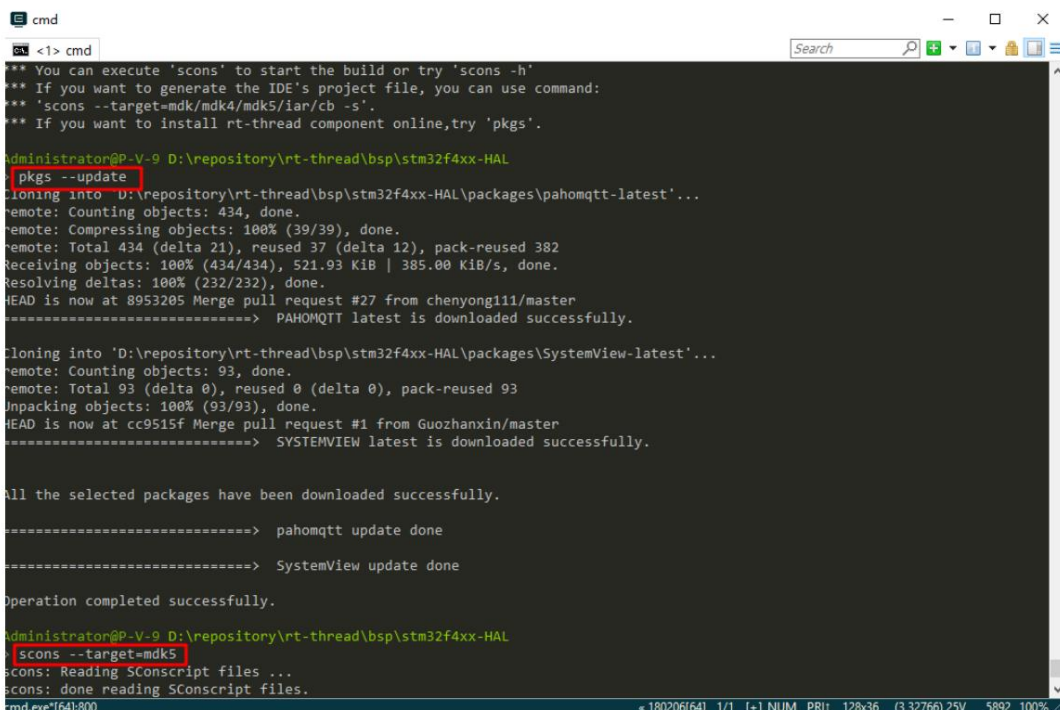
Figure 7: Enable MQTT Software Package

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.



```

cmd
<1> cmd

** 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.
cmd.exe*[64]:800
+ 180206[64] 1/1 [+ ] NUM PRI: 128x36 (3,32766) 25V 5892 100%

```

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 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.

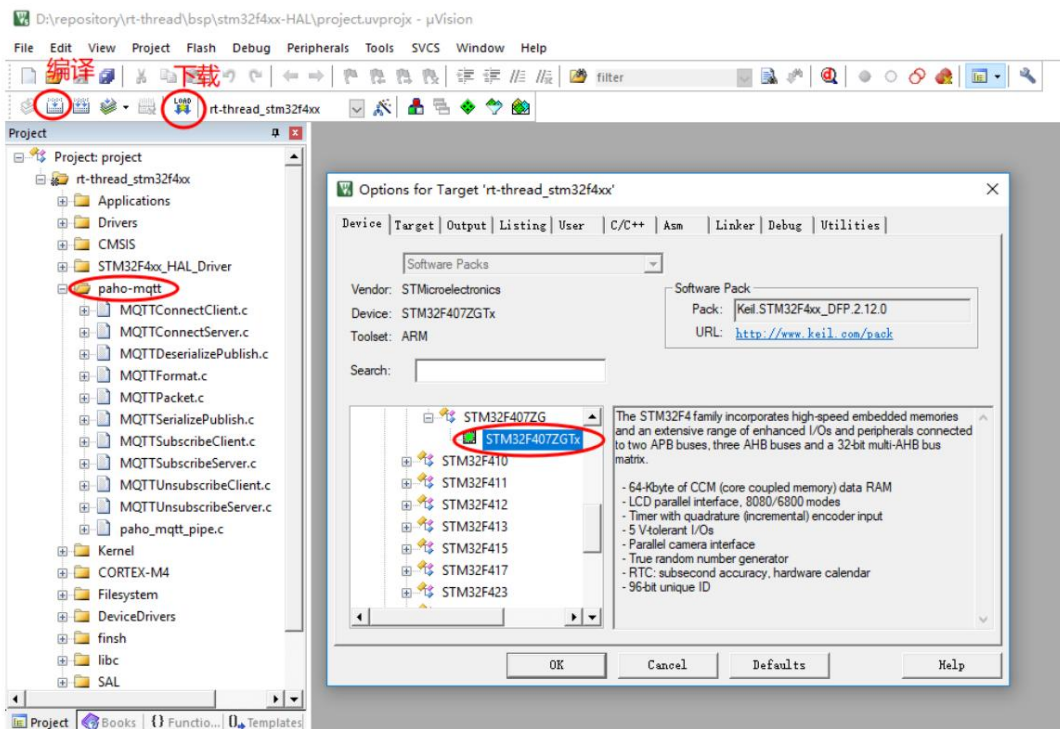


Figure 9: New files have been added to the project

3.7 Verification Project

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 project console, right-click the computer → Properties → Device Manager → 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.

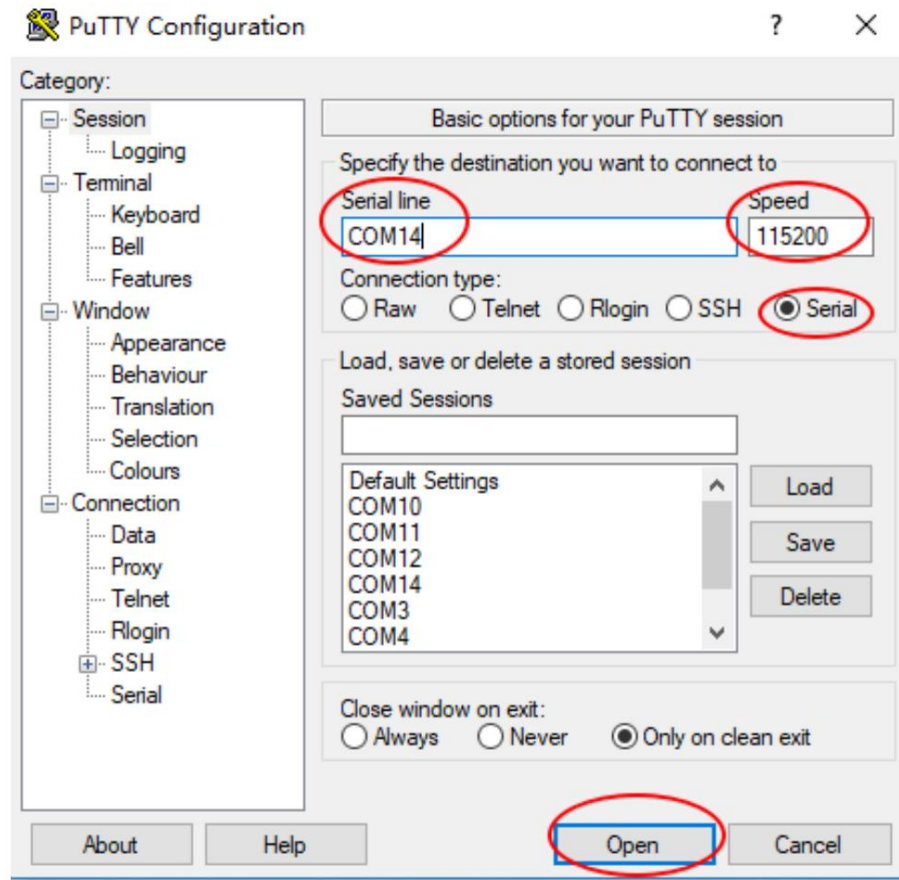


Figure 10: PuTTY Configuration

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

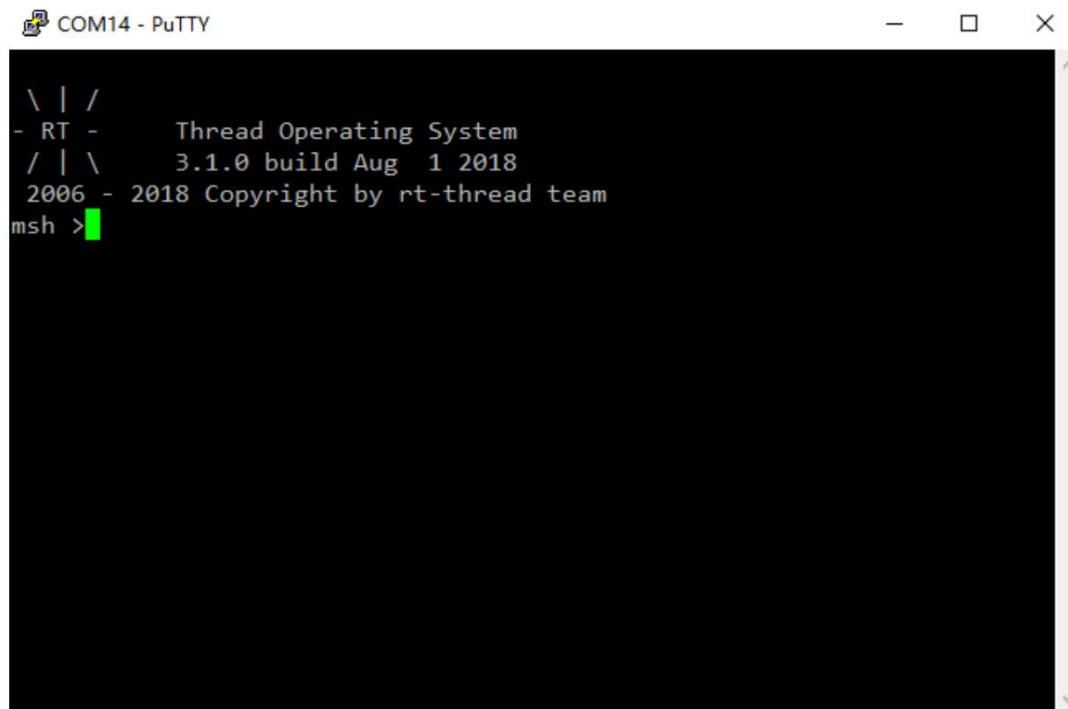


Figure 11: RT-Thread Startup logo information

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 under 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 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`, which provides a set of command line interface 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 to 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()`.

```

\ | /
- 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 >

```

Annotations in the image:

- Red arrow pointing to "Hello, world!" with text: "main函数运行的时候调用hello_world()函数打印的内容"
- Red arrow pointing to "hello_world - Hello world!" with text: "新增的hello_world命令"
- Red arrow pointing to "Hello, world!" (after command) with text: "执行hello_world命令打印的内容"

Figure 12: hello-world

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

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

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.

```

```

#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 does not depend on
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 Adding modules to the project

As mentioned earlier, if you don't have many source code files, it's recommended that you place all of them in the "applications" folder. However, if you have a large amount of source code and want to create your own project modules or use other modules you've acquired, what's the best approach? The previous article briefly introduced the "SConscript" file in the "applications" folder. This section will demonstrate how to implement your own "SConscript" file to manage project files.

3.9.1. Add source code

Again, using the hello.c and hello.h files 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. You can also choose whether to use this module through menuconfig. Add a new hello folder under the 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: New *hello* Module

3.9.2. Writing SConscript Files

You may have noticed 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). 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 -*- to use          #Specify that the encoding format of the file is UTF-8, the file can be
    Chinese
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
    module

#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 lines of simple Python code, 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 a new one. We can add the configuration options we need based on this file.

The following configuration options are added to the hello module. The # sign is followed by a comment.


```

447     default 86
448   endif
449   config USBH_USING_VBUS
450     bool "using vbus check pin"
451     default n
452   endmenu
453 endif
454
455 menu "hello module" ----- # 菜单 hello module
456
457   config RT_USING_HELLO ----- # RT_USING_HELLO 配置选项
458     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
459     default y ----- # RT_USING_HELLO 默认选中
460     help ----- # 帮助信息, 解释这个配置项的含义, 以及如何去配置他
461     this hello module only used for test
462
463   config RT_HELLO_NAME ----- # RT_HELLO_NAME 配置选项
464     string "Hello name" ----- # RT_HELLO_NAME 类型为字符串
465     default "hello" ----- # RT_HELLO_NAME 默认值为 hello
466
467   config RT_HELLO_VALUE ----- # RT_HELLO_VALUE 配置选项
468     int "Hello value" ----- # RT_HELLO_VALUE 类型为 int
469     default 8 ----- # RT_HELLO_VALUE 默认值为8
470
471   endmenu
472

```

Figure 14: New *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 on the page. After entering the menu, it will be displayed as shown below.

```

menuconfig
<1> menuconfig
.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

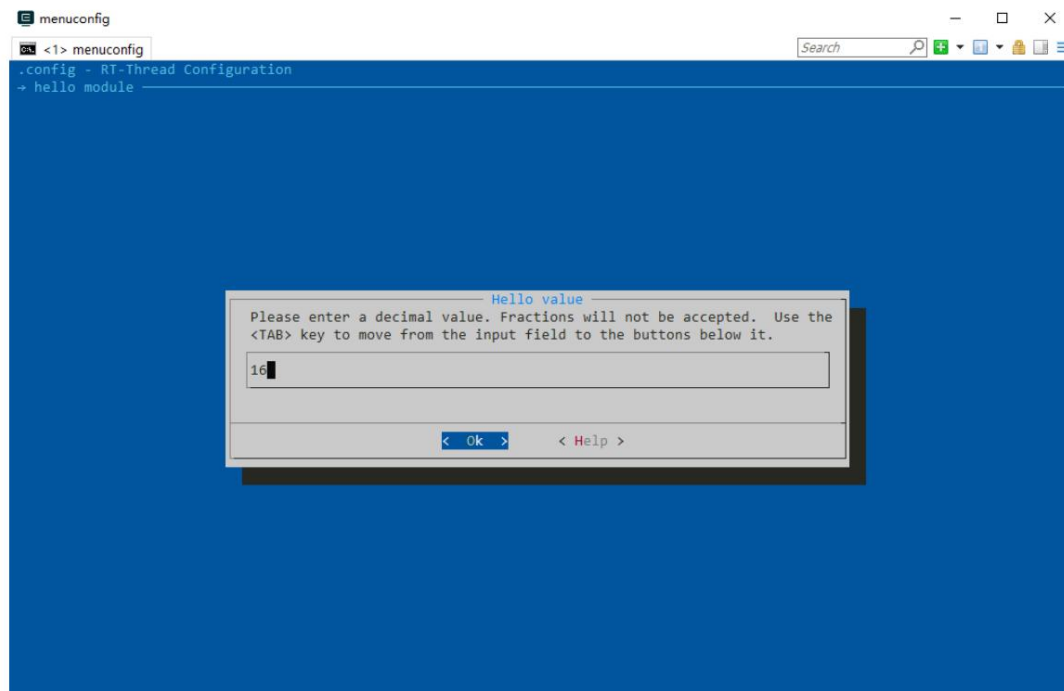
[*] Enable hello module
(hello) Hello name (NEW)
(8) Hello value

<select> < Exit > < Help > < Save > < Load >

```

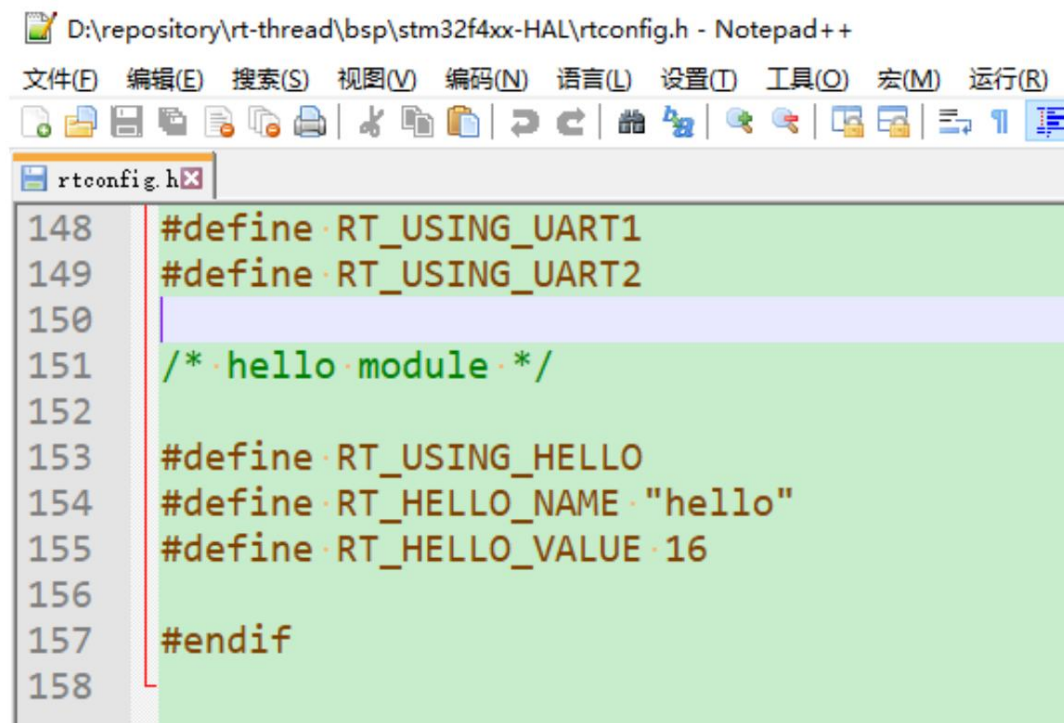
Figure 15: New *hello* Module Configuration Menu

We can also modify the value of hello value.

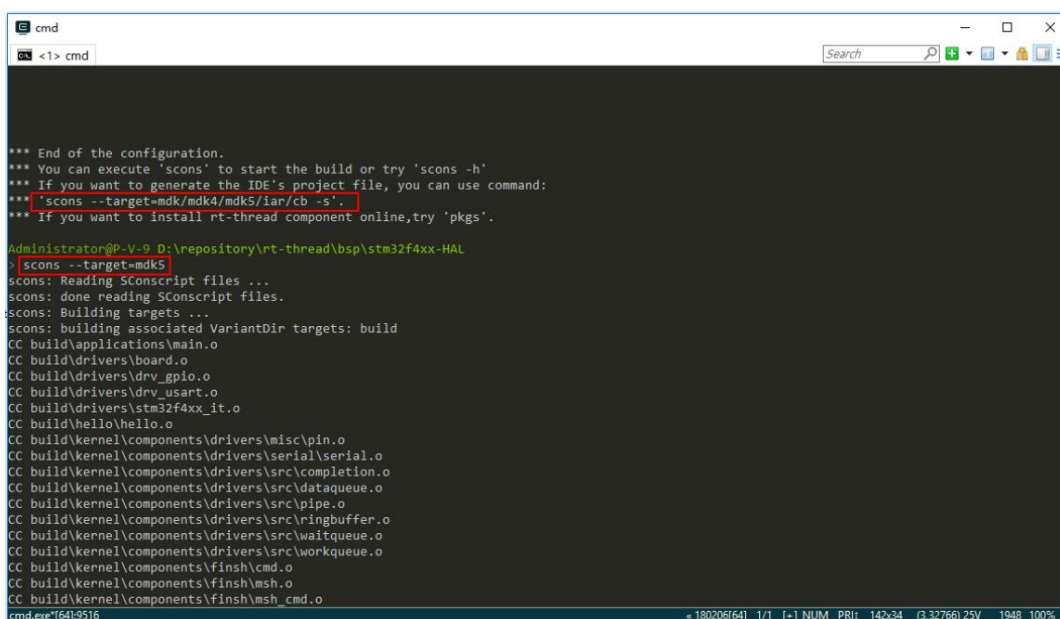
Figure 16: New *hello* Module Configuration Menu

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.

Figure 17: New *hello* Module configuration information

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



```

cmd
<1> cmd

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

```

Figure 18: Generate a new project

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

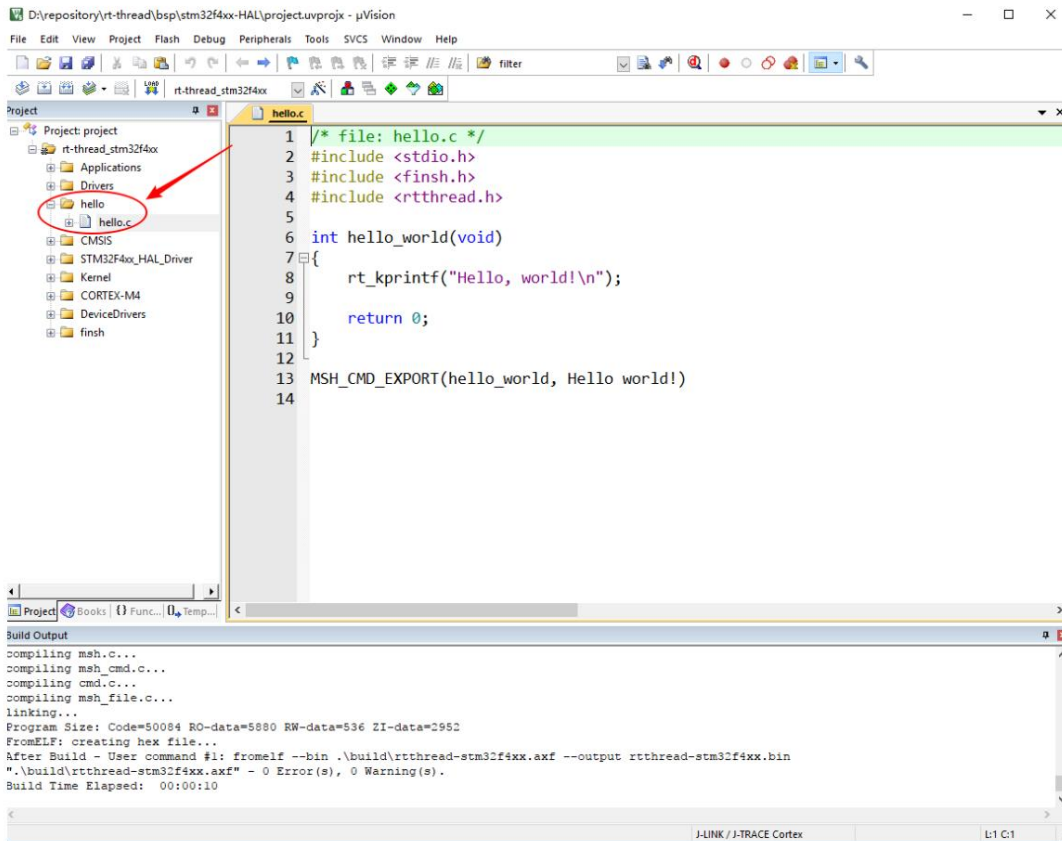


Figure 19: Generate a new project

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

Use [the Using Env in your project section](#) in the manual Or view Kconfig related documents on the Internet to learn more about Kconfig

Use method.