# “国产MCU开发技术与生态建设”
# 第三期线上分享系列课程

## 第3讲：Embedded Studio嵌入式开发入门

主讲人：吕裔枫

2020年/8月/19日

单片机与嵌入式系统应用
Microcontrollers & Embedded Systems
eefocus 与非网

# 目录

CONTENTS

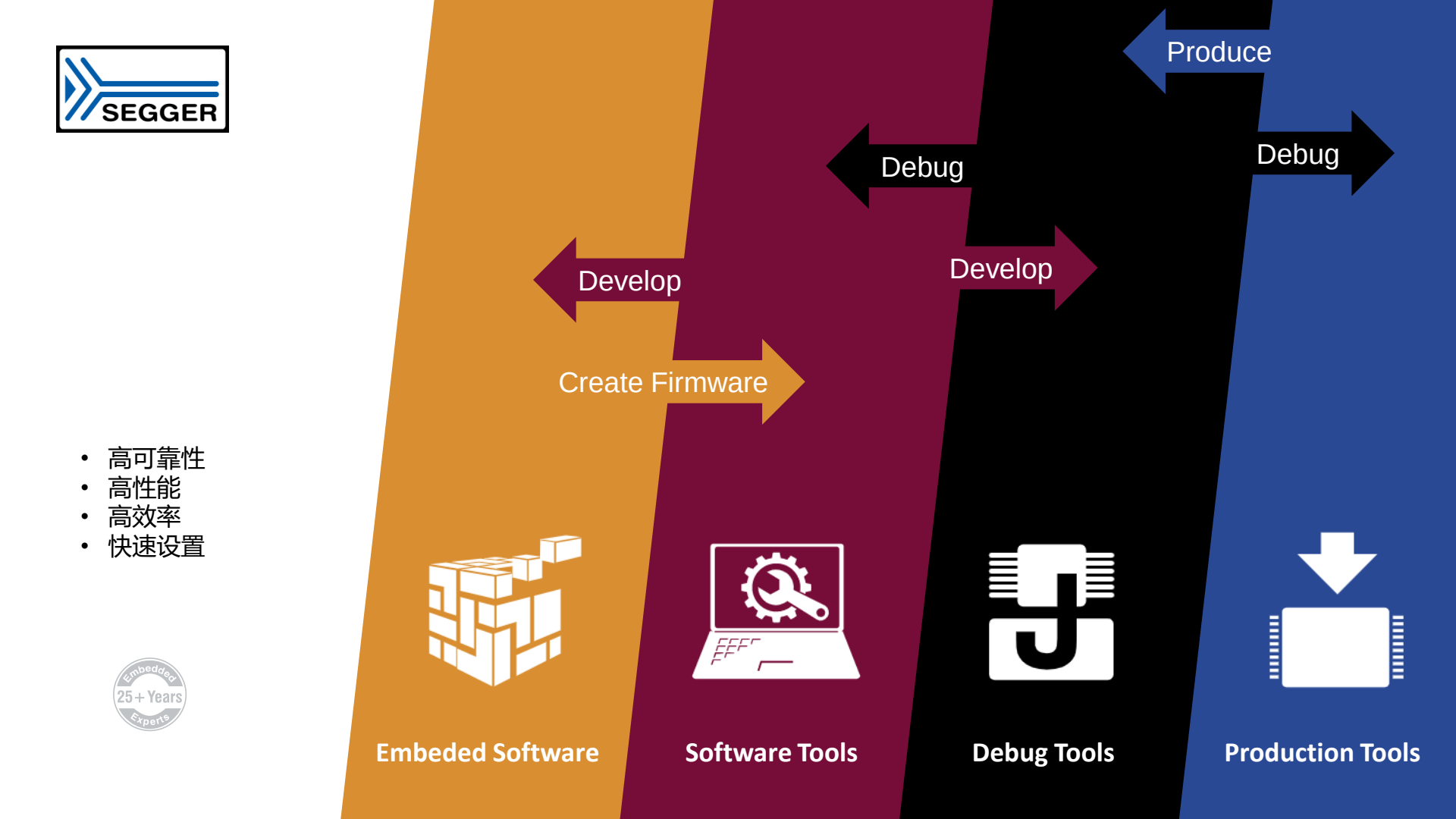
# 01 Segger Embedded Studio 介绍

# Segger公司介绍

德国segger集团总部(Monheim)


Segger公司创始人


Segger公司的产品手册




- 高可靠性
- 高性能
- 高效率
- 快速设置

# Segger Embedded Studio 介绍

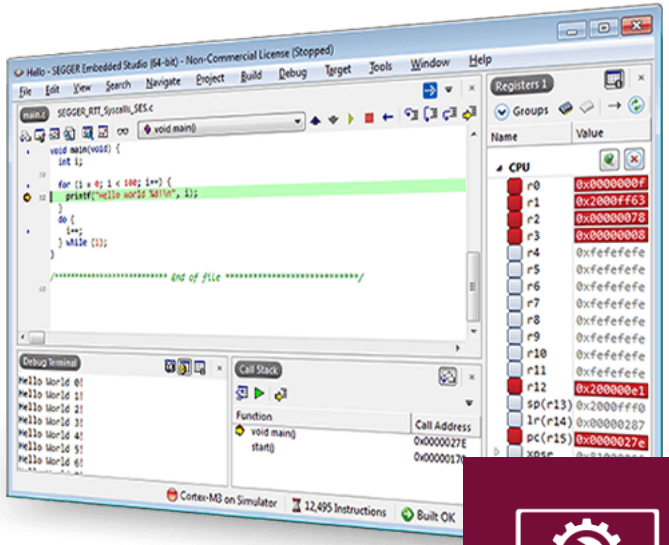
BMR 麦克泰技术

* Embedded Studio集成开发环境

嵌入式系统的C/C++集成开发环境

Embedded Studio 是一款功能强大的嵌入式系统 C/C++ IDE，为用户提供专业的嵌入式C编程的一体化解决方案。

- 轻松启动通用微控制器的项目生成器
- 功能强大的工程管理和源代码编辑器
- 具有高级调试信息窗口的集成调试器
- 直接 J-Link 集成

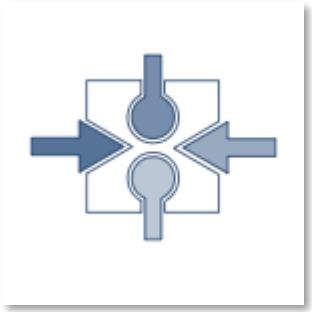


可视化工作室式风格为嵌入式工程世界提供了与 PC 开发人员所熟悉的直观使用方式。

# Segger Embedded Studio 介绍

BMR 麦克泰技术

* Embedded Studio集成开发环境



**内核支持**

Embedded Studio支持基于ARM和RISC-V的微控制器。

**效 率**

快速启动，较短的项目加载时间和并行构建，可最大程度地减少等待时间，并提高效率。

**移植性**

Embedded Studio支持在Windows，macOS和Linux上使用。

**工具链**

Embedded Studio附带的两个工具链GCC和LLVM。

# Segger Embedded Studio 介绍

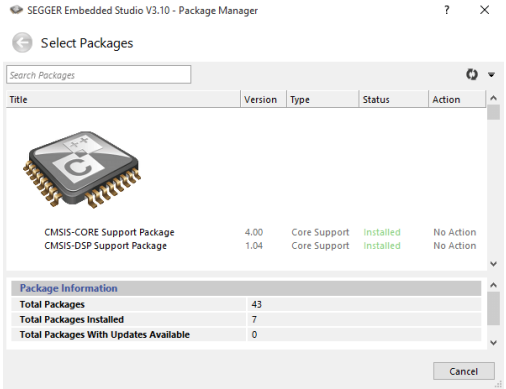
BMR 麦克泰技术

* 内核支持

- Cortex-M0， Cortex-M0+， Cortex-M1， Cortex-M3， Cortex-M4， Cortex-M7
- Cortex-M23和Cortex-M33
- Cortex-A和Cortex-R
- 传统 ARM7、ARM9、ARM11
- RISC-V RV32
- CPU 支持包提供一切，使您轻松入门

ARM®

RISC-V




* 包管理器

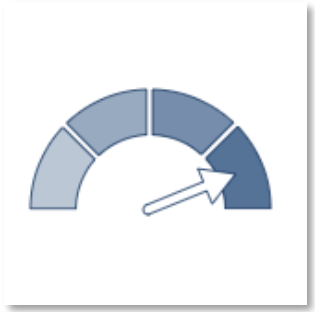
CPU支持包包括了启动文件、内存映射、CPU 初始化等所有文件，用于为您的设备创建一个完整的新项目。此外，预先配置的示例项目也包含在每个包中

# Segger Embedded Studio 介绍

- Embedded Studio集成开发环境

## 内核支持

Embedded Studio支持基于ARM和RISC-V的微控制器。



## 效率

快速启动，较短的项目加载时间和并行构建，可最大程度地减少等待时间，并提高效率。

## 移植性

Embedded Studio支持在Windows，macOS和Linux上使用。

## 工具链

Embedded Studio附带的两个工具链GCC和LLVM。

# Segger Embedded Studio 介绍

•启动与项目加载

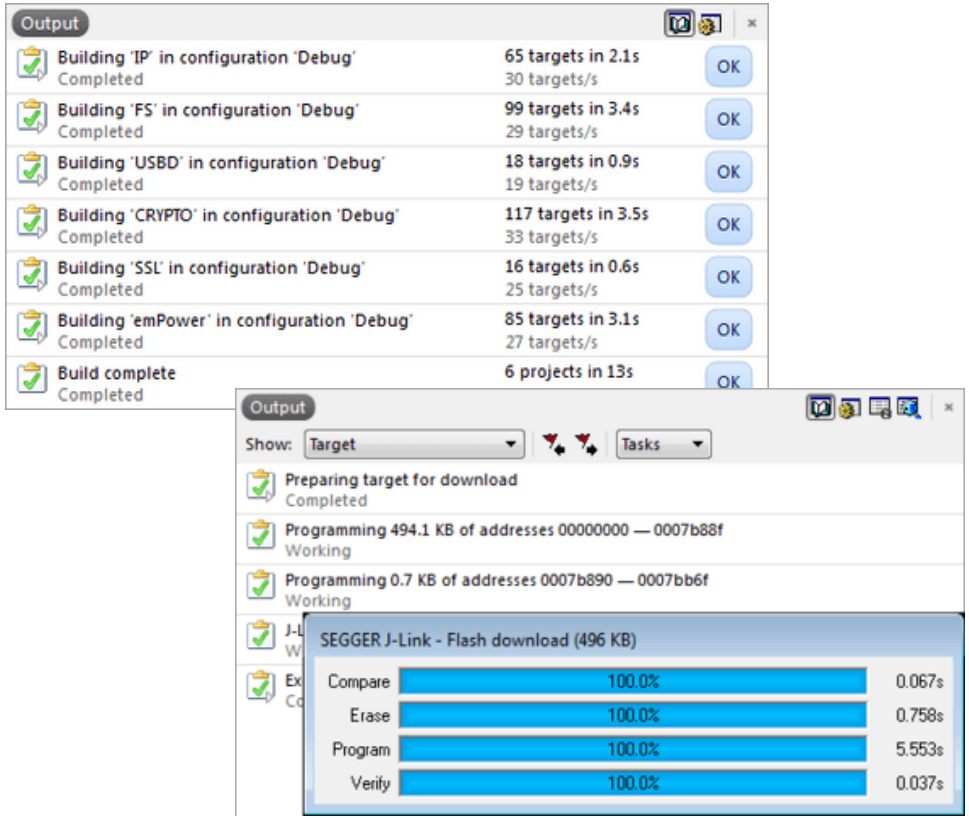
| | |
|---|---|
| **启动** | 1 秒 |
| **项目加载** | 1 秒 |

•多线程构建
•6 个工程中的 400 个源文件，Windows 7 64 位

| | |
|---|---|
| **1线程构建** | 41 秒（10 个文件/秒） |
| **8线程构建** | 13 秒（31 文件/秒） |

•使用J-Link调试
•496 kByTe 应用， Kinetis K60， SWD， 16 MHz

| | |
|---|---|
| **调试启动，空设备** | 7 秒 |
| **调试启动，无源更改** | 2 秒 |
| **重新 启动** | 1 秒 |



# Segger Embedded Studio 介绍

BMR 麦克泰技术

* Embedded Studio集成开发环境

内核支持

Embedded Studio支持基于ARM和RISC-V的微控制器。

效 率

快速启动，较短的项目加载时间和并行构建，可最大程度地减少等待时间，并提高效率。

**移植性**

Embedded Studio支持在Windows，macOS和Linux上使用。

工具链

Embedded Studio附带的两个工具链GCC和LLVM。

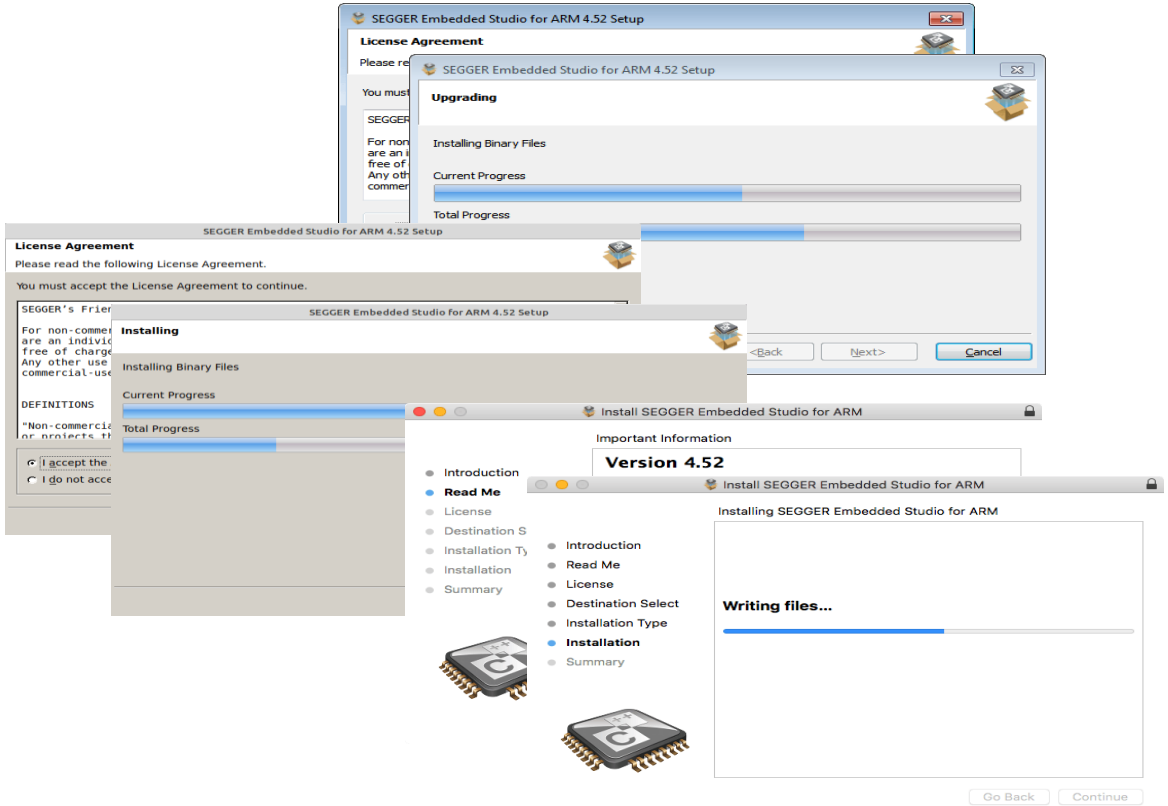
# Segger Embedded Studio 介绍

BMR 麦克泰技术

* 移植性

Embedded Studio支持的操作系统：

- Windows
- Linux
- MacOS



# Segger Embedded Studio 介绍

BMR 麦克泰技术

* Embedded Studio集成开发环境

内核支持

Embedded Studio支持基于ARM和RISC-V的微控制器。

效 率

快速启动，较短的项目加载时间和并行构建，可最大程度地减少等待时间，并提高效率。

移植性

Embedded Studio支持在Windows，macOS和Linux上使用。

**工具链**

Embedded Studio附带的两个工具链GCC和LLVM。

# Segger Embedded Studio 介绍

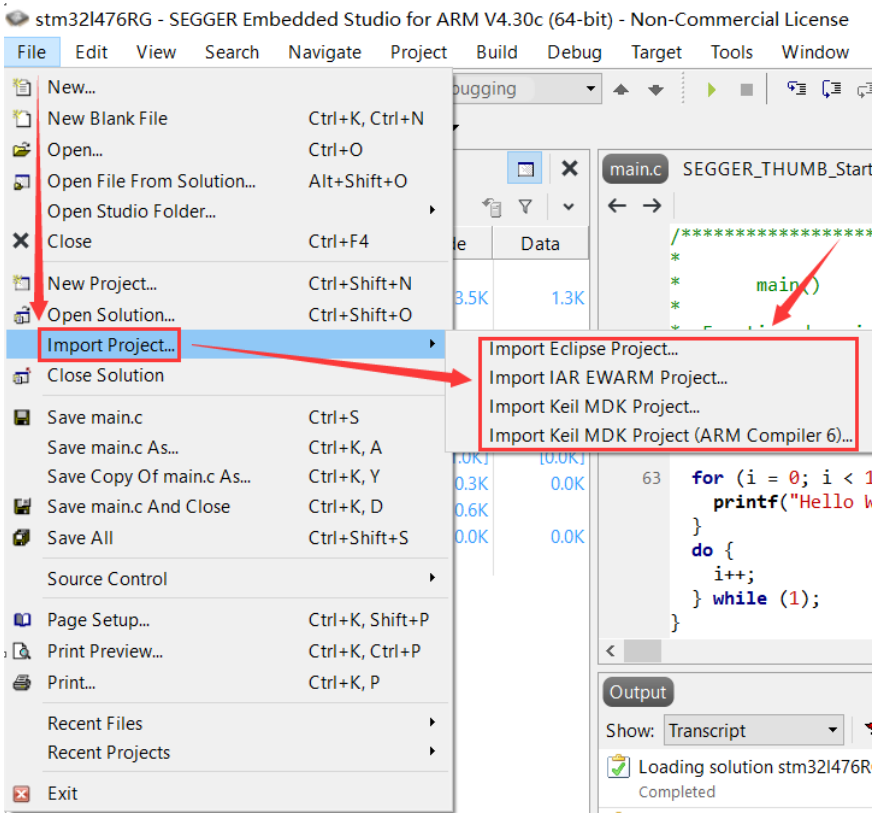
BMR 麦克泰技术

* 工具链

- 包括从编译到生成执行文件所有工具
- GCC
- Clang/LLVM
- GNU连接器与库管理
- 支持使用外部工具链
- 内置优化的标准C库

Embedded Studio支持从其他开发环境导入工程：

- Eclipse
- IAR EWARM
- Keil MDK
- Keil MDK（ARM Compiler 6）

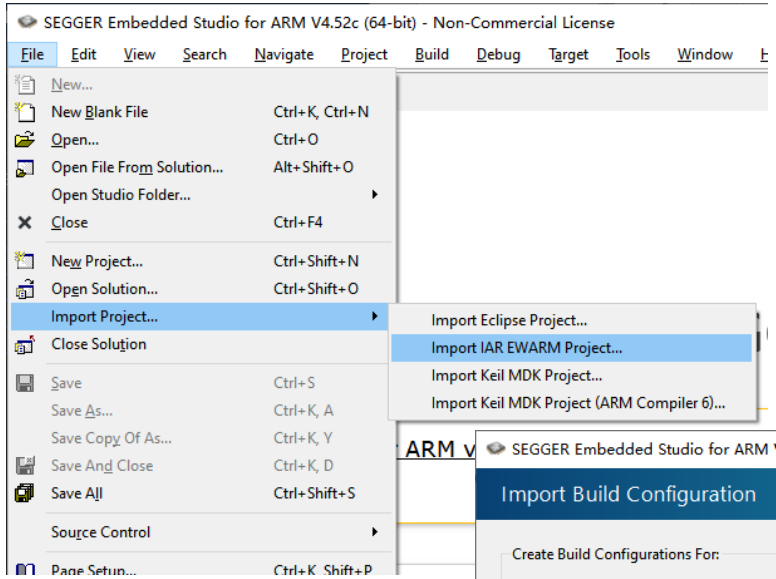


# Segger Embedded Studio 介绍

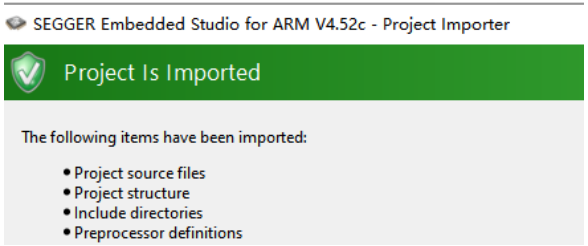
* 工具链

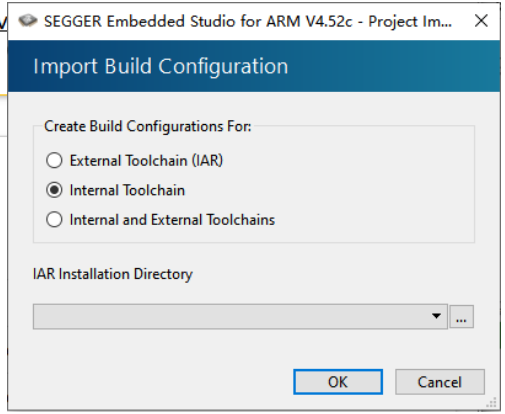
Embedded Studio导入IAR工程：

- 点击Import IAR EWARM Project开始导入
- 定位到IAR工程文件（*.eww/*.ewp）
- 选择工具链
- 开始开发/编译/调试




导入成功



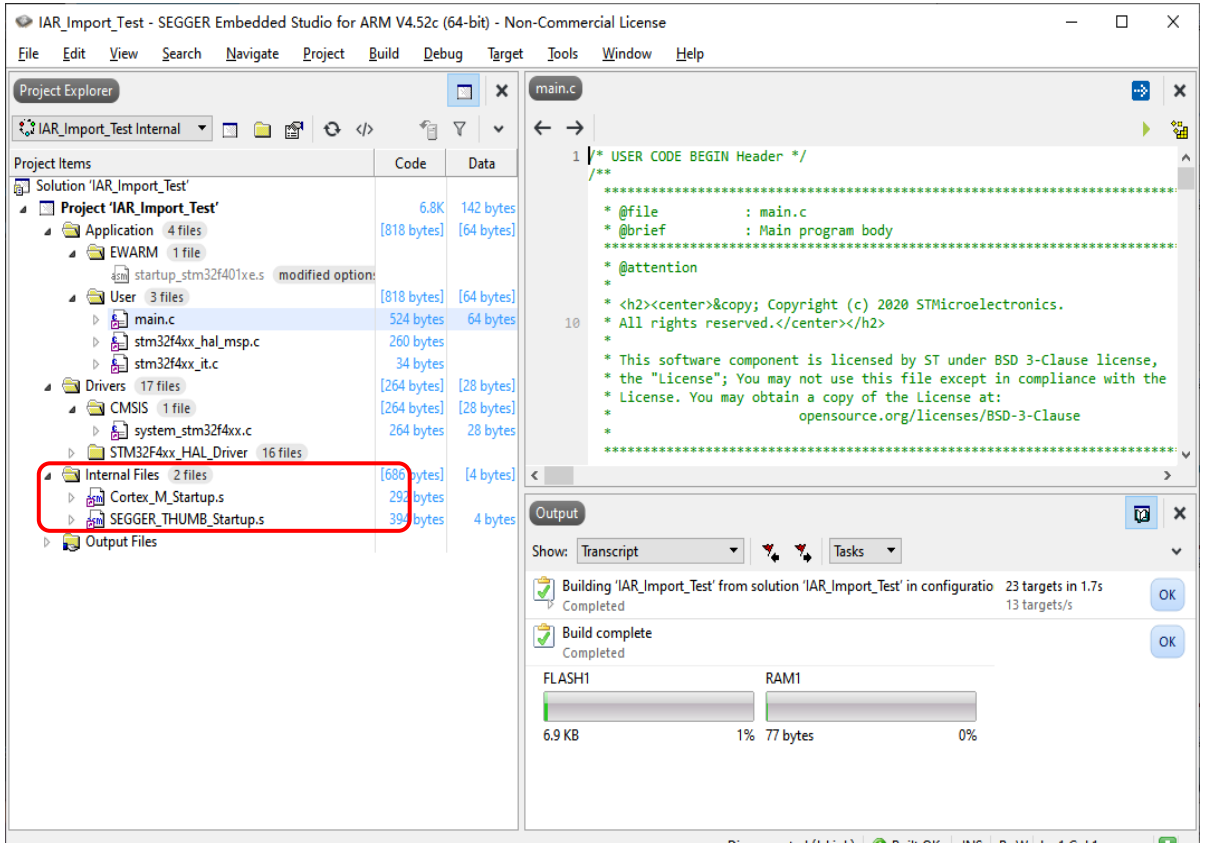



# Segger Embedded Studio 介绍

## 工具链

Embedded Studio导入IAR工程：

- 自动替换汇编文件



# Segger Embedded Studio 介绍

BMR 麦克泰技术

* Embedded Studio集成开发环境

## 免费使用

对于非商业性和非营利性的教育用途，Embedded Studio是免费提供的。

您可以将其用于课程，学校或在家中的业余爱好项目。

## 项目设置

项目管理器可以在一个地方组织和管理项目源代码。

多项目解决方案，动态文件夹和属性继承为您的项目设置提供了灵活性。

## 调试器

J-Link / J-Trace调试仿真器无缝集成到Embedded Studio调试器中，以启用其所有强大功能。

调试器可以提供有关操作系统的信息-当前正在执行的任务，或者每个任务使用堆栈等。

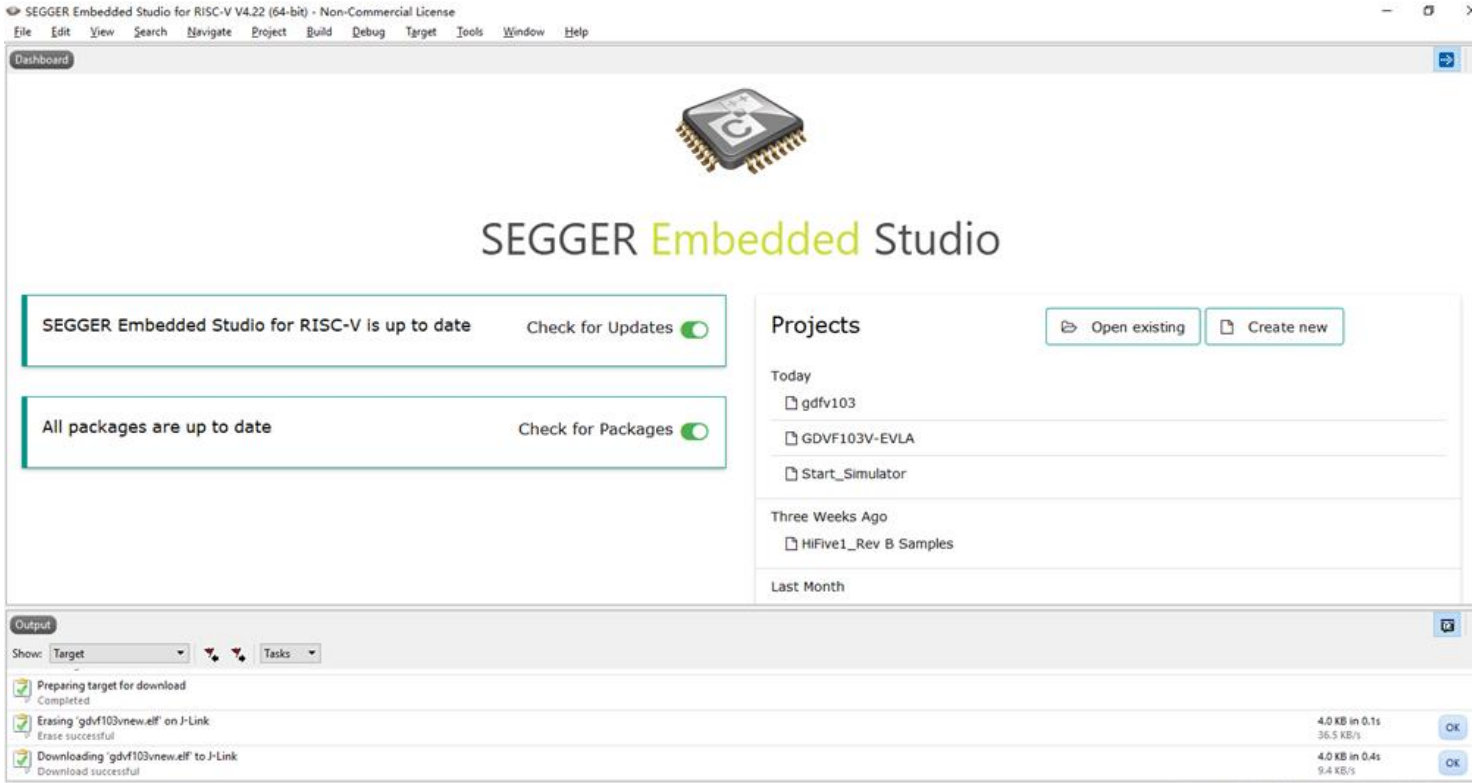
02

# Segger Embedded Studio 快速入门

# Embedded Studio 使用入门

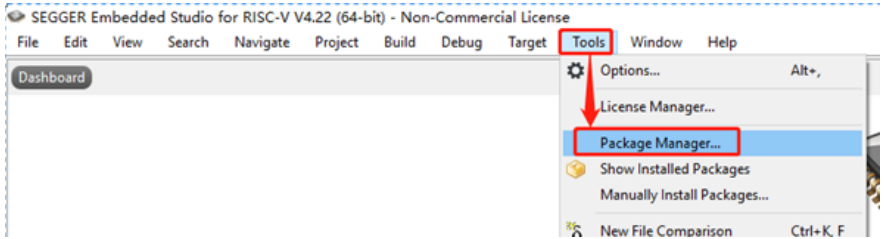
* 启动界面

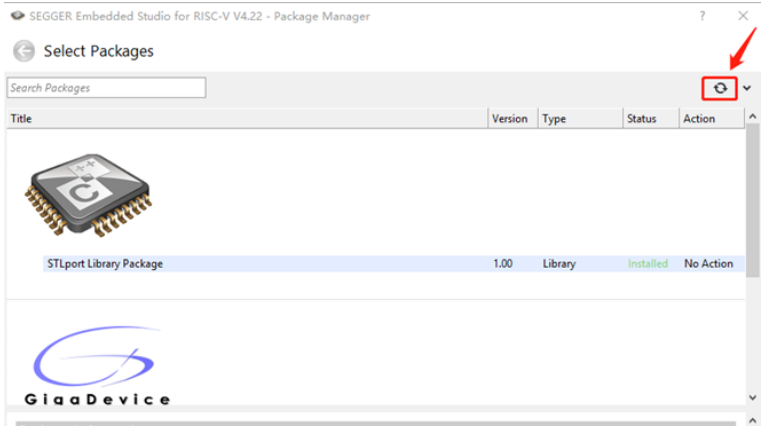
- 处理器基于软件包管理
- 完整的IDE功能
- 强大的调试器
- 跨平台



# Embedded Studio 使用入门

* 下载支持包
  - 进入包管理器
    - Tools->Package Manager



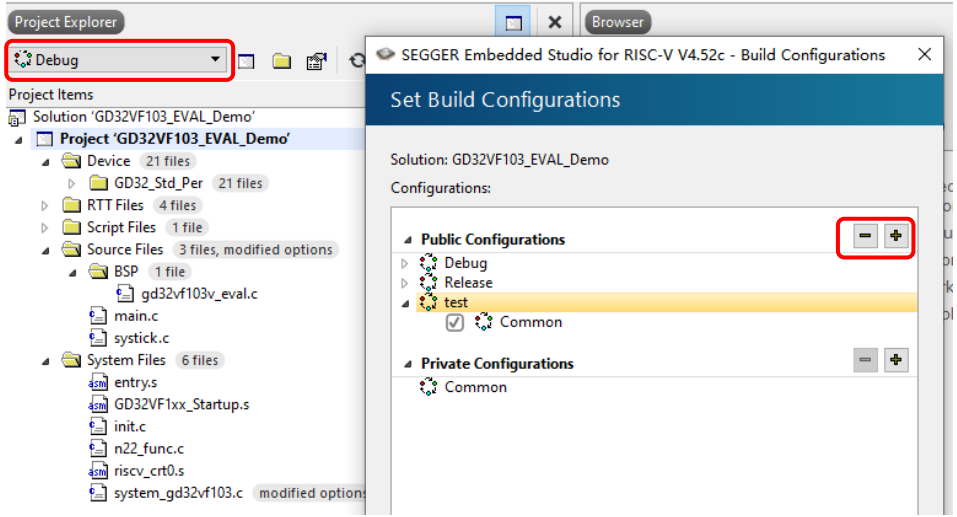




- 选择支持包下载
  - 在Select Packages下面右侧按 刷新 按钮，第一次会显示所有包的列表，如果没有安装的，可以用鼠标点在你需要的包上选择下载安装
  - 双击选中GDVFXX处理器支持包，点击Next进行下载安装

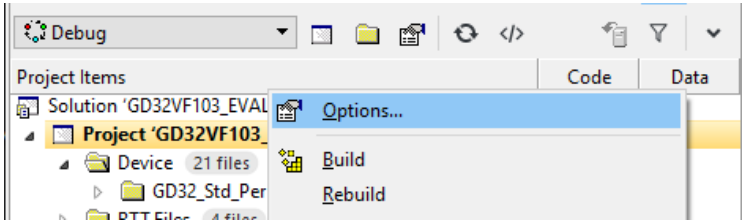
# Embedded Studio 使用入门

* 工程配置管理

保存不同的工程配置方便管理与调试



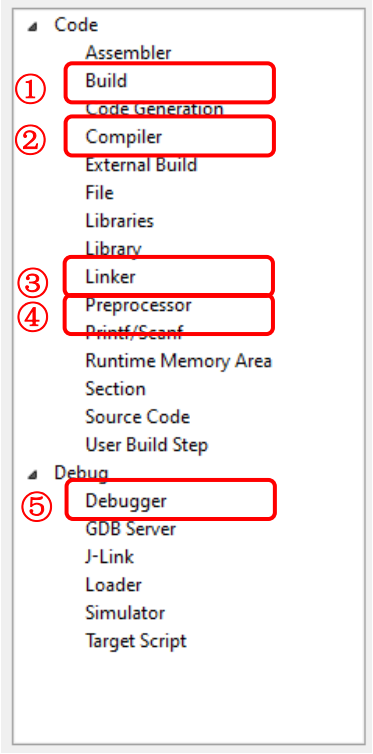

右键工程名，选择Options进入工程配置页面



# Embedded Studio 使用入门

BMR 麦克泰技术

* 基本工程配置




① Build选项：设置编译器与中间文件输出位置

| Build | |
|---|---|
| Always Rebuild | No |
| Batch Build Configurations | |
| Build Quietly | Yes |
| Dependency File Name | *None* |
| Enable Unused Symbol Removal | Yes |
| Exclude From Build | No |
| External Compiler | None |
| Include Debug Information | Yes |
| Intermediate Directory | Output/$(ProjectName) $(Configuration)/Obj |
| Memory Map File | *None* |

② Compiler选项：设置C语言标准与编译器

| Compiler | |
|---|---|
| Additional C Compiler Only Options | |
| Additional C Compiler Only Options From File | *None* |
| Additional C/C++ Compiler Options | |
| Additional C/C++ Compiler Options From File | *None* |
| Additional C++ Compiler Only Options | |
| Additional C++ Compiler Only Options From File | *None* |
| C Language Standard | gnu99 |
| C++ Language Standard | gnu++98 |
| Compiler | gcc |
| Enable All Warnings | No |

③ Linker选项：设置定位文件与二进制文件输出格式

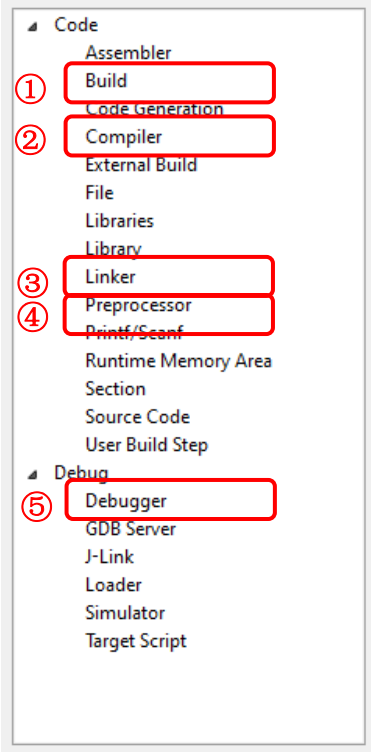
| Linker | |
|---|---|
| Executable File Name | $(OutDir)/$(ProjectName)$(EXE) |
| Additional Input Files | |
| Link Dependent Projects | Yes |
| Use Manual Linker Script | No |
| Section Placement File | $(ProjectDir)/flash_placement_riscv.xml inherits |
| Section Placement Macros | |
| Default Fill Pattern | *None* |
| Additional Output Format | None inherits |
| Generate Map File | Yes |

④ Preprocessor选项：设置宏定义与预处理头文件目录

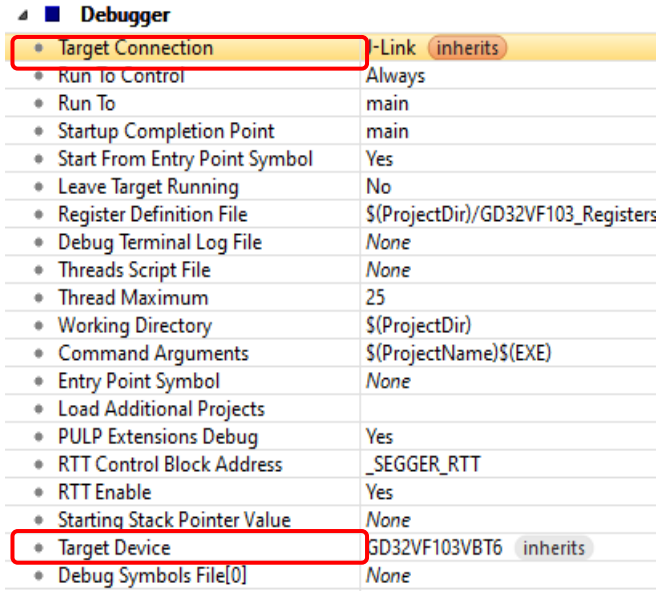
| Preprocessor | |
|---|---|
| Ignore Includes | No |
| Preprocessor Definitions | inherits |
| Preprocessor Undefinitions | |
| System Include Directories | |
| Undefine All Preprocessor Definitions | No |
| User Include Directories | inherits |

# Embedded Studio 使用入门

* 基本工程配置




⑤：Debugger选项：设置调试器与目标芯片



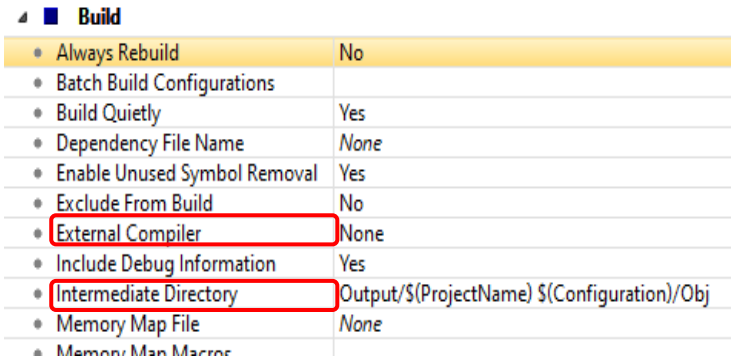
# Embedded Studio 使用入门

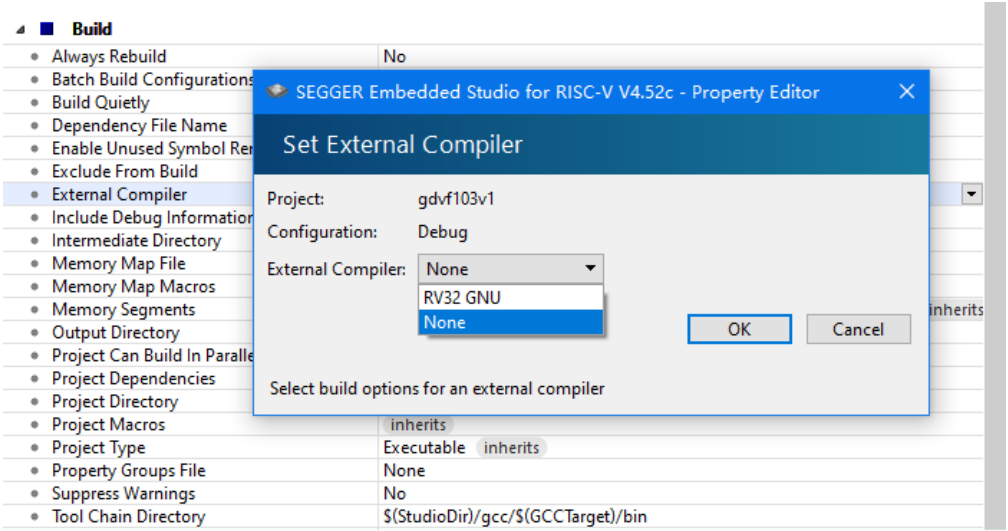
BMR 麦克泰技术

* **①Build选项**

1，选择编译器

2，设置中间文件，与二进制文件输出位置，可使用默认位置



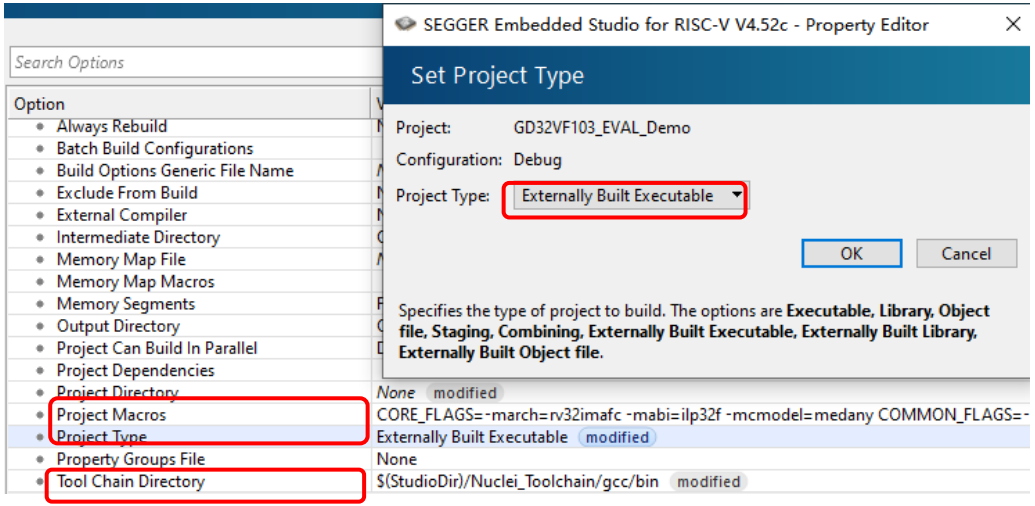
# Embedded Studio 使用入门

* ①Build选项-使用外部工具链（芯来官方工具链为例）

1　修改Project Type为Externally Built Executable

2　修改Tool Chain Directory为\$(StudioDir)/Nuclei_Toolchain/gcc/bin

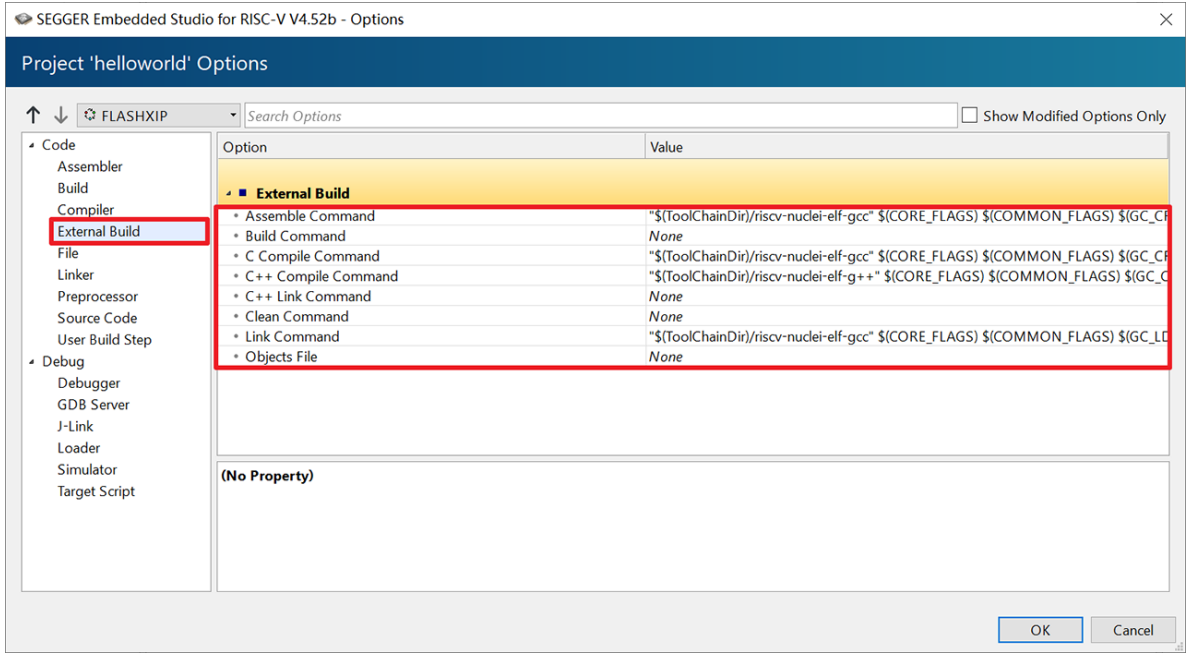
3　修改工程宏



# Embedded Studio 使用入门

* ①Build选项-使用外部工具链（芯来官方工具链为例）

修改编译指令



# Embedded Studio 使用入门

* ①Build选项-使用外部工具链（芯来官方工具链为例）

修改工程宏

CORE_FLAGS=-march=rv32imafc -mabi=ilp32f -mcmodel=medany COMMON_FLAGS=-g -fno-common -DDOWNLOAD_MODE=DOWNLOAD_MODE_FLASHXIP GC_CFLAGS=-ffunction-sections -fdata-sections GC_LDFLAGS=-Wl,--gc-sections -Wl,--check-sections NEWLIB_LDFLAGS=--specs=nano.specs EXTRA_LDFLAGS=-u _isatty -u _write -u _sbrk -u _read -u _close -u _fstat -u _lseek

修改编译指令

Assemble Command：  "$(ToolChainDir)/riscv-nuclei-elf-gcc" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_CFLAGS) $(AsmOptions) $(Defines) $(Includes) -MD -MF "$(RelDependencyPath)" -c -o  "$(RelTargetPath)"  "$(RelInputPath)"

C Compile Command：  "$(ToolChainDir)/riscv-nuclei-elf-gcc" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_CFLAGS) $(COptions) $(COnlyOptions) $(Defines) $(Includes) -MD -MF "$(RelDependencyPath)" -c -o  "$(RelTargetPath)"  "$(RelInputPath)"

C++ Compile Command：  "$(ToolChainDir)/riscv-nuclei-elf-g++" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_CFLAGS) $(COptions) $(CppOnlyOptions) $(Defines) $(Includes) -MD -MF "$(RelDependencyPath)" -c -o  "$(RelTargetPath)"  "$(RelInputPath)"

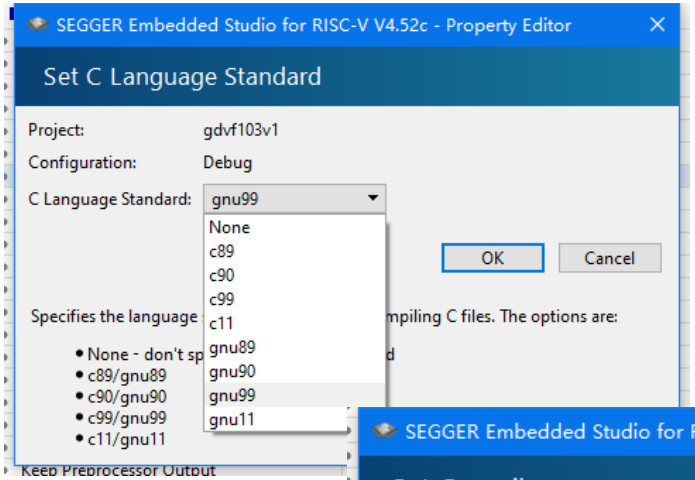
Link Command：  "$(ToolChainDir)/riscv-nuclei-elf-gcc" $(CORE_FLAGS) $(COMMON_FLAGS) $(GC_LDFLAGS) $(NEWLIB_LDFLAGS) $(EXTRA_LDFLAGS)  --specs=nosys.specs -MMD -MT $(ProjectName)$(EXE) -MF $(ProjectName)$(EXE).d $(Objects) -o "$(OutDir)/$(ProjectName)$(EXE)" -T "$(RelLinkerScriptPath)" -lstdc++ -nostartfiles -Wl,-M,-Map="$(RelMapPath)" $(LinkOptions)

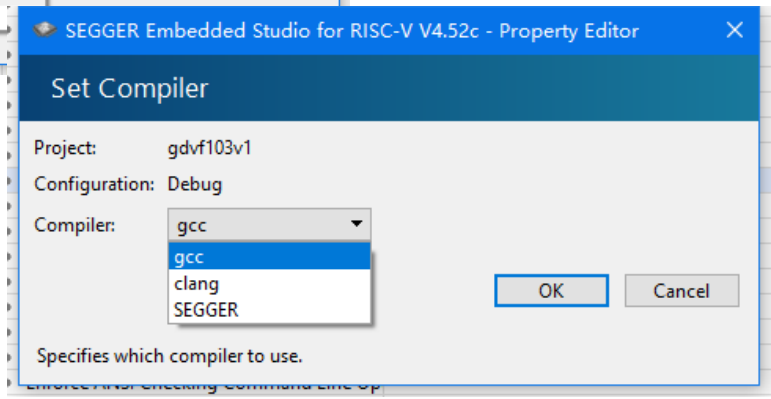
# Embedded Studio 使用入门

* ②Compiler选项

1，选择要使用的C语言标准

2，设置使用的内置编译器




| Compiler | |
|---|---|
| Additional C Compiler Only Options | |
| Additional C Compiler Only Options From File | *None* |
| Additional C/C++ Compiler Options | |
| Additional C/C++ Compiler Options From File | *None* |
| Additional C++ Compiler Only Options | |
| Additional C++ Compiler Only Options From File | *None* |
| C Language Standard | gnu99 |
| C++ Language Standard | gnu++98 |
| Compiler | gcc |
| Enable All Warnings | No |

# Embedded Studio 使用入门

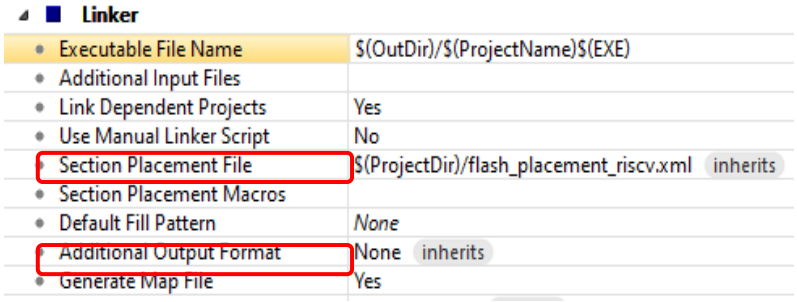
BMR 麦克泰技术

* ③Linker选项

1，设置定位文件目录，程序移植时请确保此定位文件位置正确。

- flash_placement.xml
- sram_placement.xml

2，设置二进制文件格式



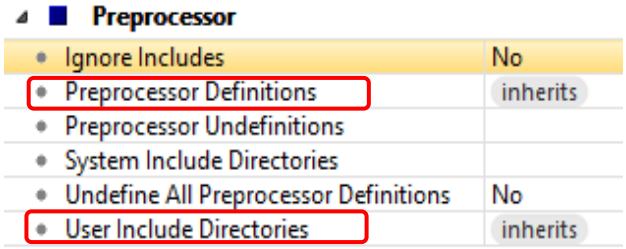
# Embedded Studio 使用入门

BMR 麦克泰技术

* ④Preprocessor选项

1，设置预处理宏定义

2，设置预处理头文件目录



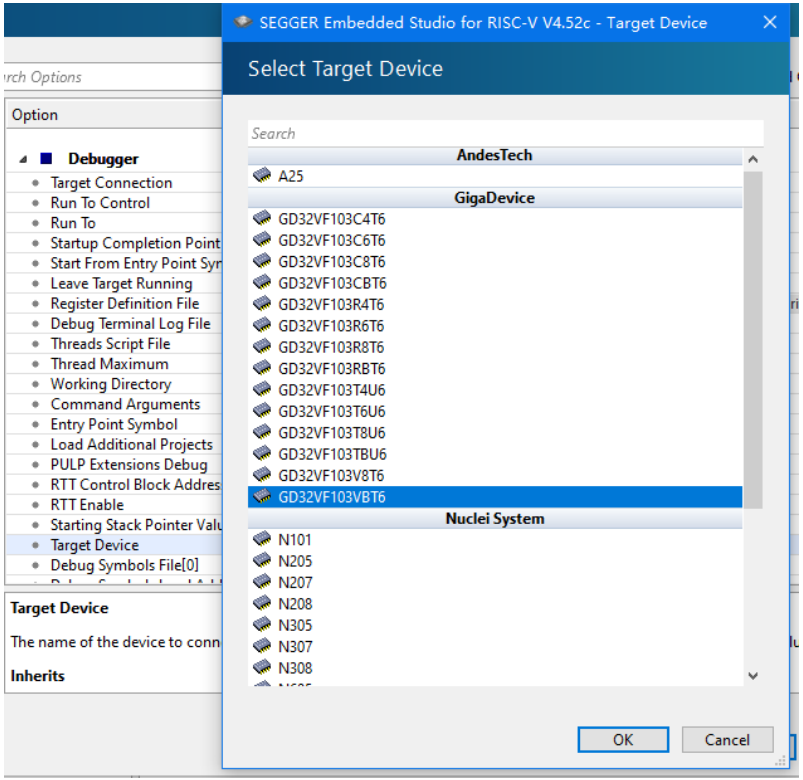
# Embedded Studio 使用入门

* ⑤Debugger选项

1，设置调试器
- Simulator
- J-Link
- GDB Server

2，设置目标芯片

| Debugger | |
|---|---|
| Target Connection | -Link inherits |
| Run To Control | Always |
| Run To | main |
| Startup Completion Point | main |
| Start From Entry Point Symbol | Yes |
| Leave Target Running | No |
| Register Definition File | $(ProjectDir)/GD32VF103_Registers |
| Debug Terminal Log File | *None* |
| Threads Script File | *None* |
| Thread Maximum | 25 |
| Working Directory | $(ProjectDir) |
| Command Arguments | $(ProjectName)$(EXE) |
| Entry Point Symbol | *None* |
| Load Additional Projects | |
| PULP Extensions Debug | Yes |
| RTT Control Block Address | _SEGGER_RTT |
| RTT Enable | Yes |
| Starting Stack Pointer Value | *None* |
| Target Device | GD32VF103VBT6 inherits |
| Debug Symbols File[0] | *None* |



# Embedded Studio 使用入门

BMR 麦克泰技术

* ④Debugger选项

1，J-Link调试器
- 连接类型
- 调试接口
- 调试速度

| J-Link | |
|---|---|
| Host Connection | USB |
| Target Interface Type | JTAG |
| JTAG Instruction Register Size Before Target | Auto Detect |
| JTAG Number Of Devices Before Target | Auto Detect |
| Speed | 4,000 kHz |
| Supply Power | No |
| Show Log Messages In Output Window | Yes |
| Log File | *None* |
| Script File | *None* |
| Exclude Flash Cache Range | *None* |
| Additional J-Link Options | *None* |
| Target Has Cycle Counter | No |

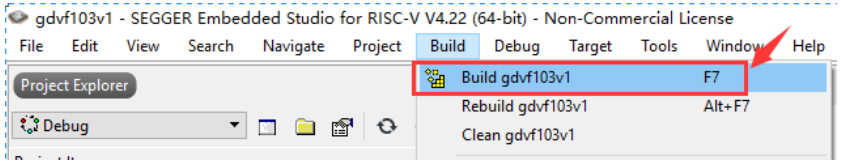
2，GDB Sever（GD-Link为例）
- 驱动下载
- 修改Type为OpenOCD。
- 修改GDB Server Command Line为："\$(StudioDir)/Nuclei_Toolchain/openocd/bin/openocd" -f 并且加上OpenOCD的设置文件路径。
- 修改Auto Start GDB Server为yes。

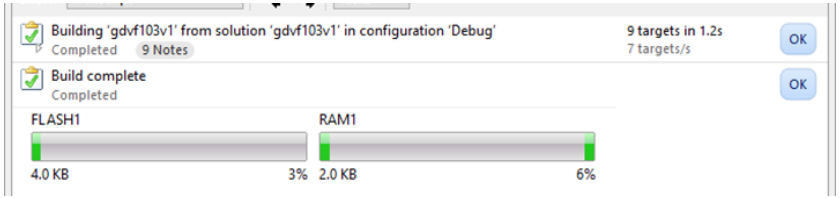
| GDB Server | |
|---|---|
| Host | localhost |
| Type | OpenOCD modified |
| GDB Server Command Line | "$(StudioDir)/Nuclei_Toolchain/openocd/bin/openocd" -f "$(StudioDir)/../n100-sdk/bsp/core/env/openocd_hbird_ilm.cfg" modified |
| Auto Start GDB Server | No modified |
| Port | 3,333 modified |
| Reset and Stop Command | reset halt modified |
| Ignore Checksum Errors | No modified |
| Allow Memory Access During Execution | Yes modified |
| Register Access | General and Individual modified |
| Log File | *None* |
| Target XML File | *None* |
| Connect Timeout | 5 seconds |
| Read Timeout | 60 seconds |
| Write Timeout | 60 seconds |

# Embedded Studio 使用入门

* 编译工程

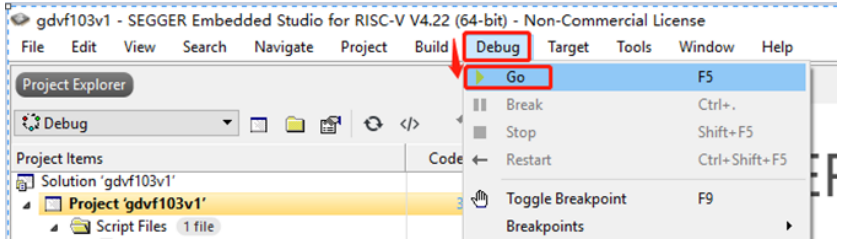
- 选中工程
- Build->Build gdvf103v1




- 编译结果

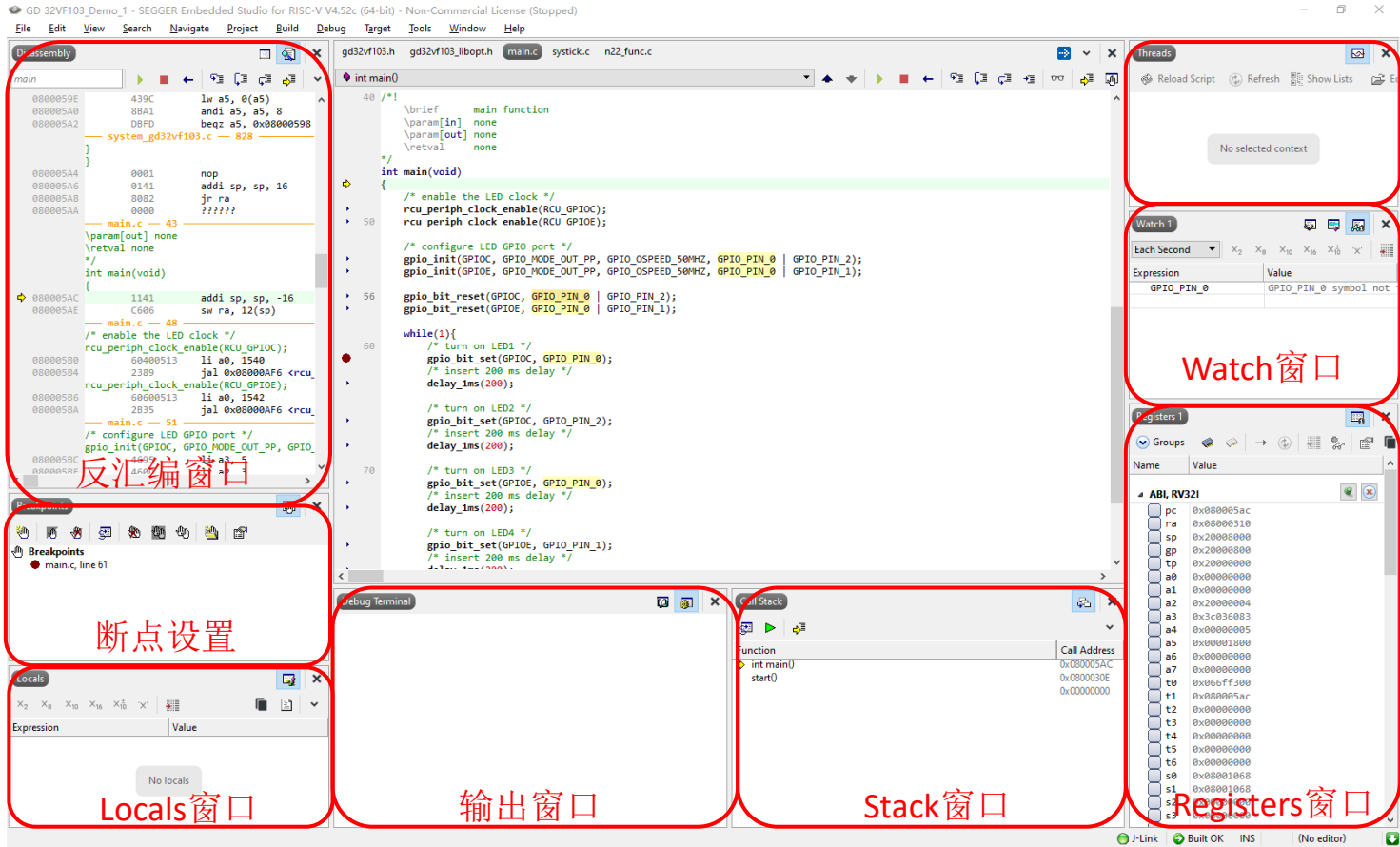
- Debug->Go进入调试



# Embedded Studio 使用入门

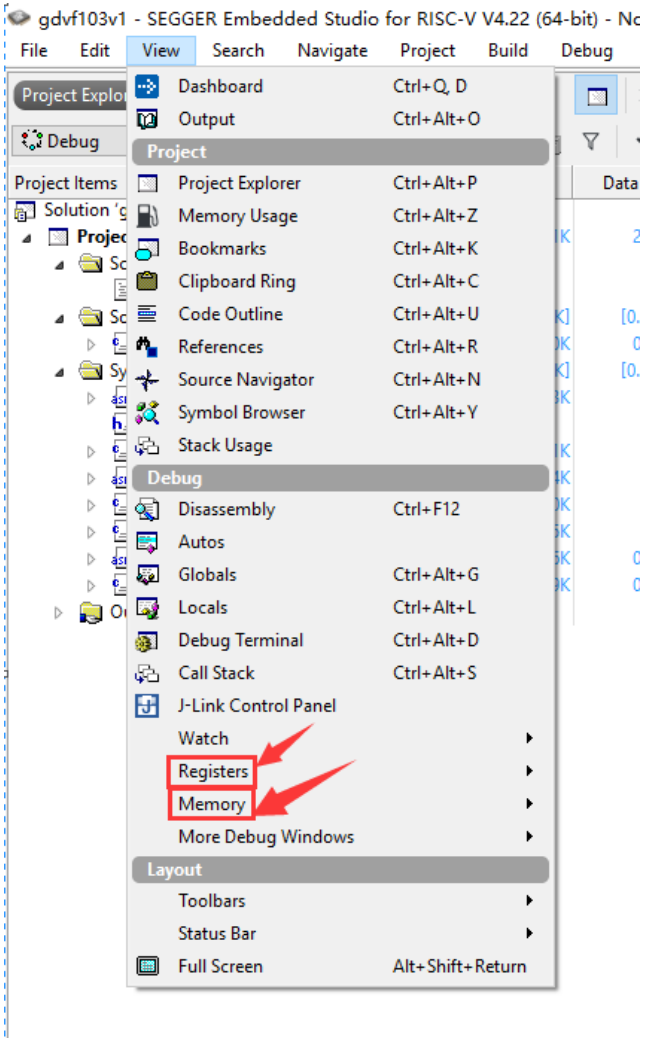
BMR 麦克泰技术

* 调试工程
  - 调试窗口



# Embedded Studio 使用入门

* 调试工程
  - 观察寄存器
    - View->Register
  - 观察存储器
    - View->Memory
  - 观察其它信息
    - View->其它



03

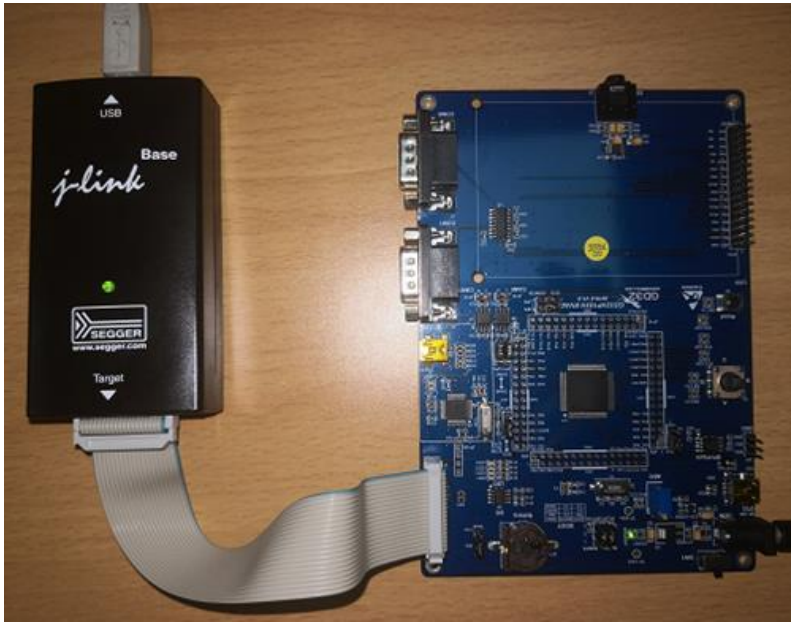
# Segger Embedded Studio 实例演示

# Embedded Studio 实例演示

BMR 麦克泰技术

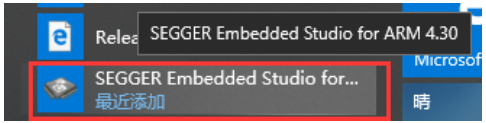
* 使用环境

- 操作系统：Windows 10 64位版
- Embedded Studio for RISC-V：4.52c 64位版
- 仿真器：J-link BASE 驱动 6.80c
- 目标板：GDVF103V-EVAL 2019.06 v1.0版




* 启动软件

- 从windows开始或从桌面图标启动Segger Embedded Studio




# Embedded Studio 实例演示

BMR 麦克泰技术

* 软件下载

  - 打开网址 https://www.segger.com/downloads/embedded-studio/ 根据你使用的计算机操作系统下载适合您的Embedded Studio for RISC-V软件

**Embedded Studio for RISC-V**

| | Version | Date | File size | |
|---|---|---|---|---|
| Embedded Studio for RISC-V, Windows, 64-bit<br>Simply download and run the installer. | V4.52c | [2020-05-18] | 226,210 KB | DOWNLOAD |
| Embedded Studio for RISC-V, Windows, 32-bit<br>Simply download and run the installer. | V4.52c | [2020-05-18] | 217,676 KB | DOWNLOAD |
| Embedded Studio for RISC-V, macOS<br>Download and mount the image, then run the installer. | V4.52c | [2020-05-18] | 258,397 KB | DOWNLOAD |
| Embedded Studio for RISC-V, Linux, 64-bit<br>Download and extract the archive, then run the installer. | V4.52c | [2020-05-18] | 245,166 KB | DOWNLOAD |
| Embedded Studio for RISC-V, Linux, 32-bit<br>Download and extract the archive, then run the installer. | V4.52c | [2020-05-18] | 263,566 KB | DOWNLOAD |

# Embedded Studio 实例演示

* 固件库下载

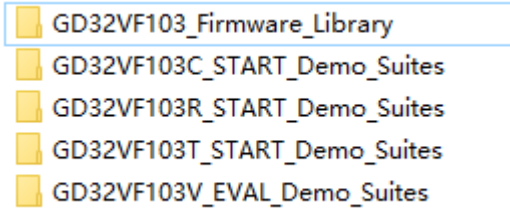
GD32VF103固件库下载：
GD官网：http://www.gd32mcu.com/
资料下载->开发板资料->GD32VF1 MCU->GD32VF103 Demo Suites

# Embedded Studio 实例演示

* 固件库

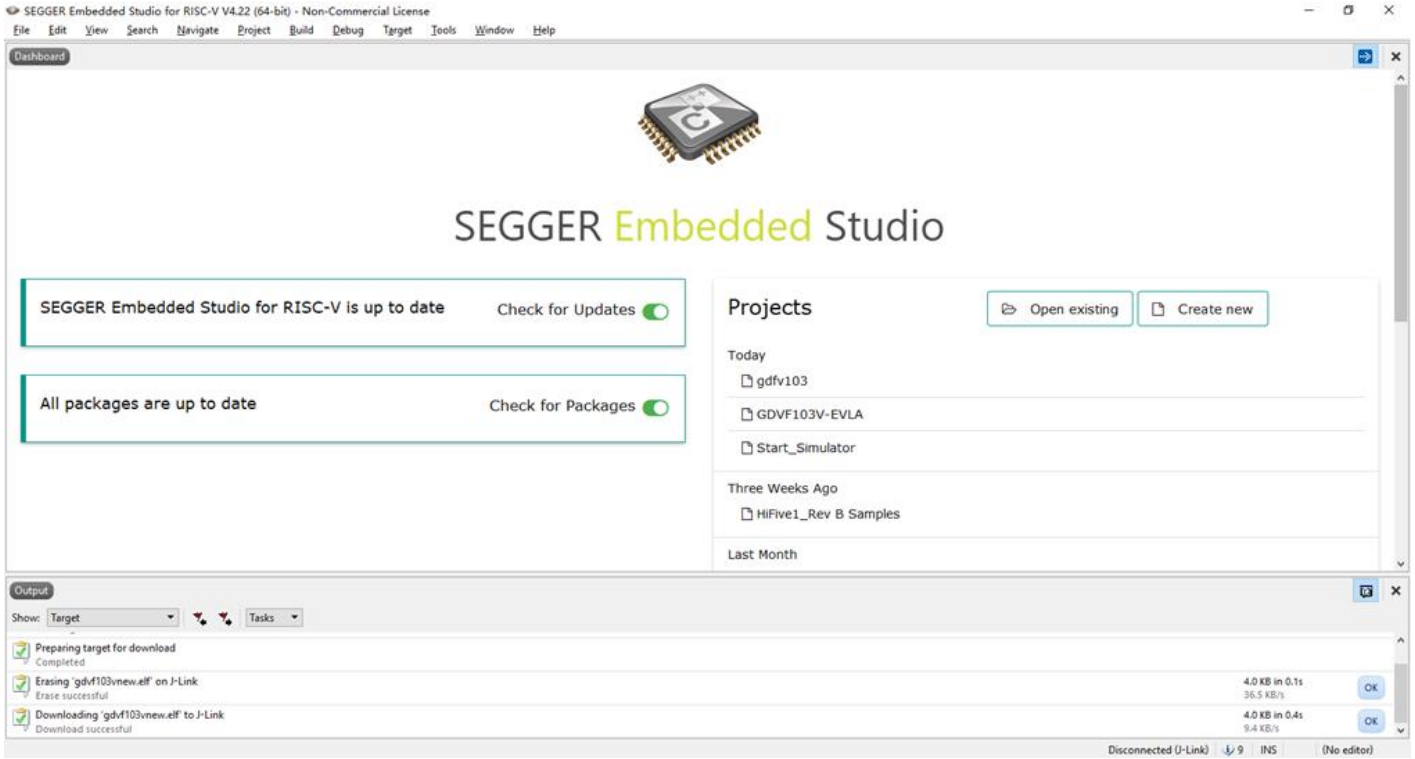
将下载的固件库GD32VF103_Demo_Suites_V1.0.3.rar解压到工程目录下，解压后的文件包括了库文件GD32VF103_Firmware_Library，和开发板例程。

本次实验将要使用的工程是GD32VF103_Demo_Suites_V1.0.3\GD32VF103V_EVAL_Demo_Suites\Projects\01_GPIO_Running_Led的电量LED工程。



# Embedded Studio 实例演示

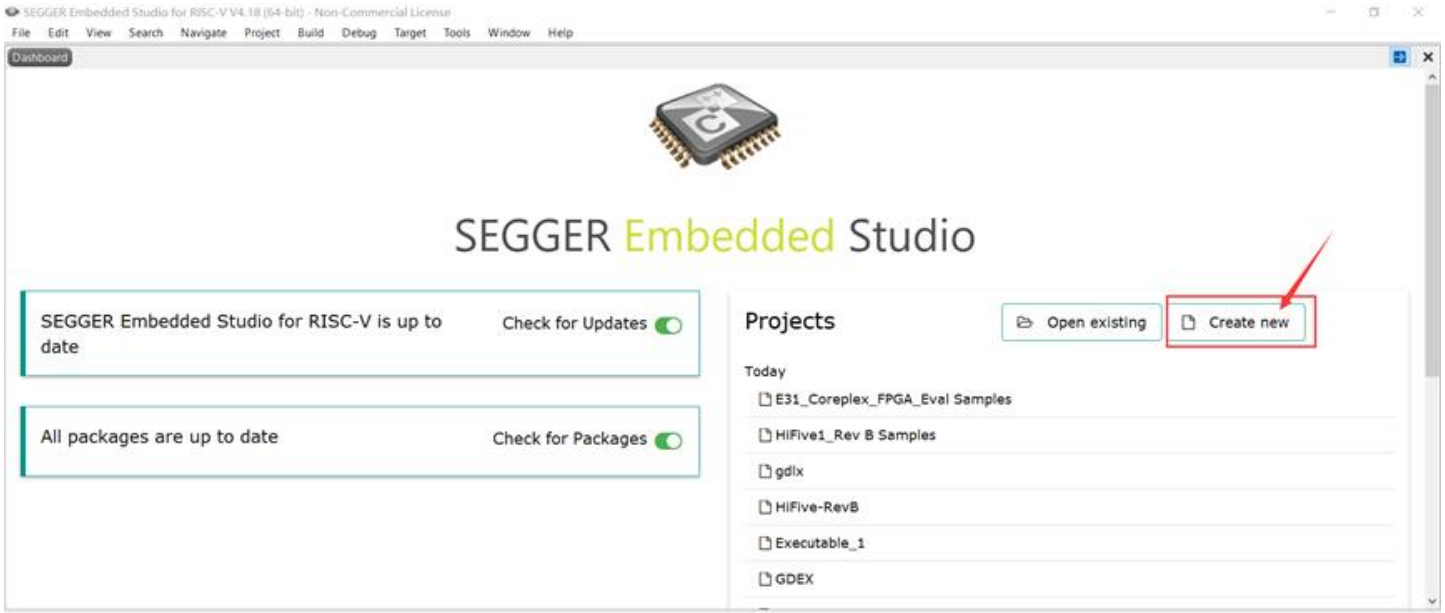
* 软件界面



# Embedded Studio 实例演示

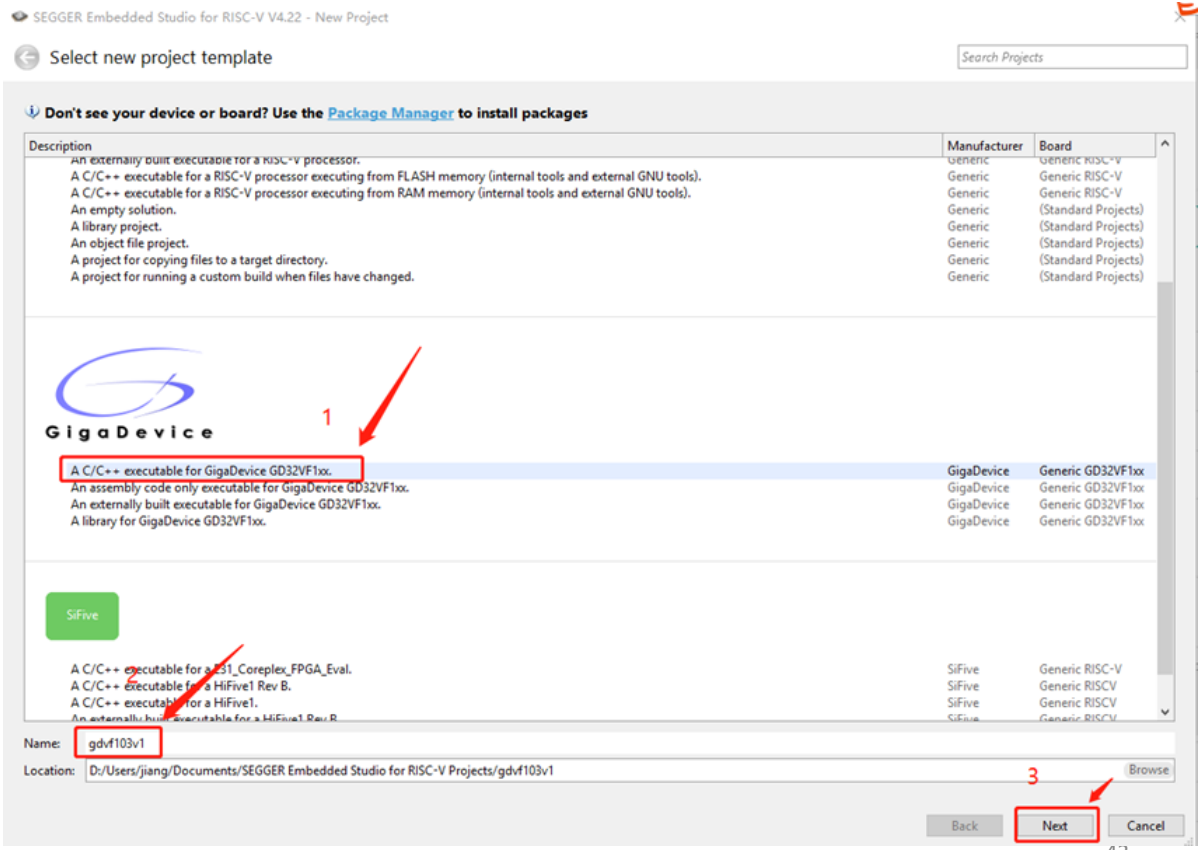
* 创建工程

  - 从右上部按Creat New，创建一个GDVF103V-EVAL板子的工程



# Embedded Studio 实例演示

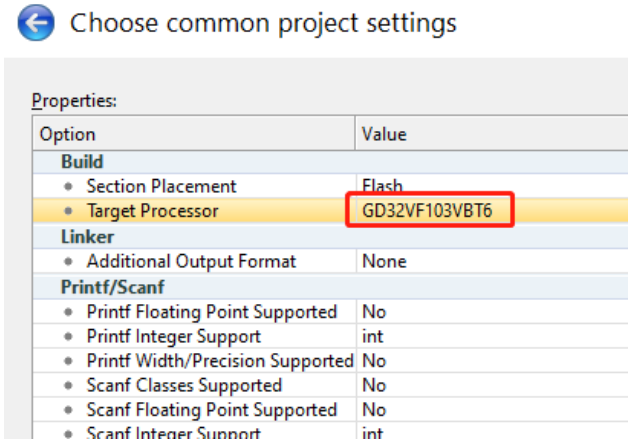
* 创建工程
  - 选择GDVF103V-EVAL板子的包如下：
  - 选择 A C/C++ executable Gige Device GD32VF1xx
  - 在下面Name右侧的框中，给你的项目起一个名字
  - 按 Next

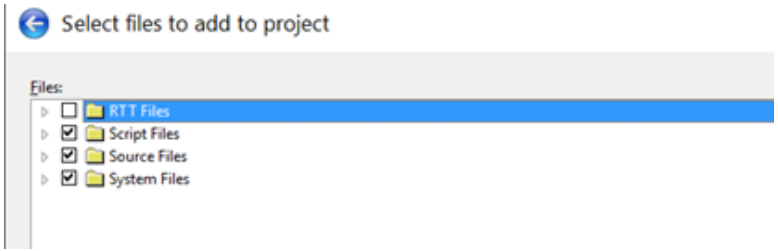


# Embedded Studio 实例演示

* 创建工程

- 弹出 Choose Common project setting窗口
  - 设置程序在哪里执行
  - 设置目标处理器的具体型号
  - 设置输出文件格式
- 按 Next



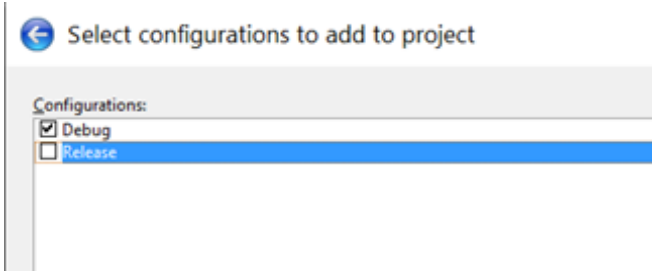

- 弹出 Select files to add to project窗口
- 选择你需要的文件
- 按 Next

# Embedded Studio 实例演示

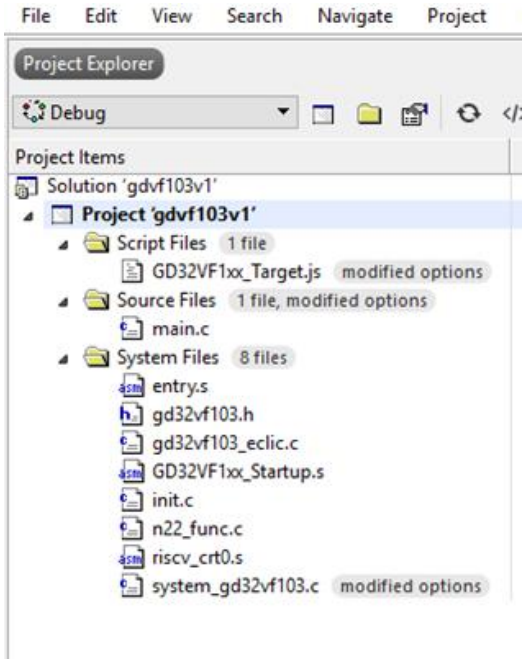
BMR 麦克泰技术

* 创建工程

- 弹出 Select configurations  to add to project窗口
  - Debug
  - Release
- 选择你需要的配置，按 Finish。



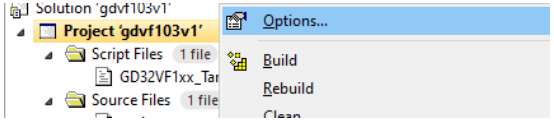

- 生成的工程如图所示

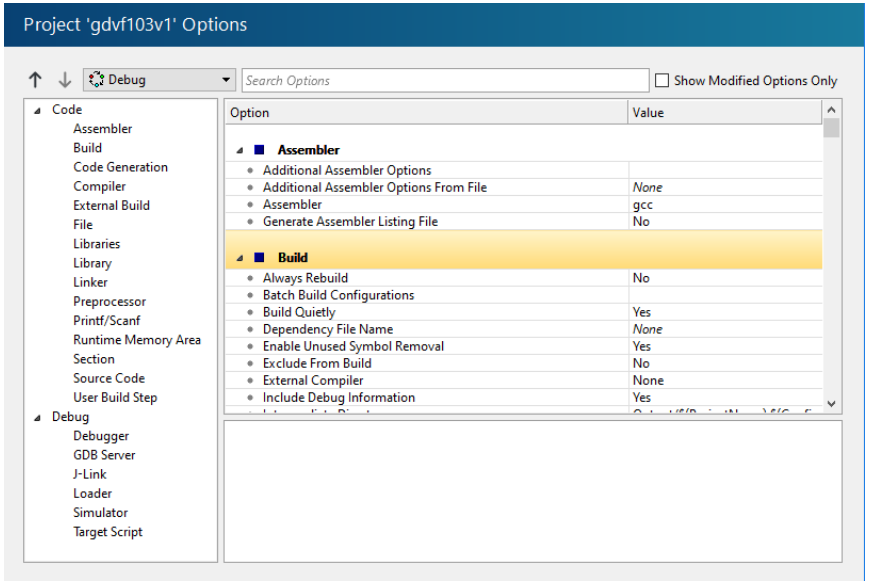


# Embedded Studio 实例演示

* 查看工程选项

- 选中工程后，右键选择Options进入工程选项，或在工具栏Project->Options中进入

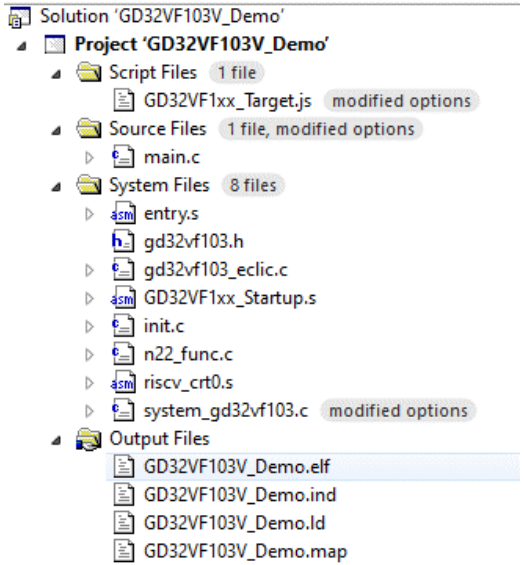



- 工程选项中包括了工程代码配置与调试配置信息

# Embedded Studio 实例演示

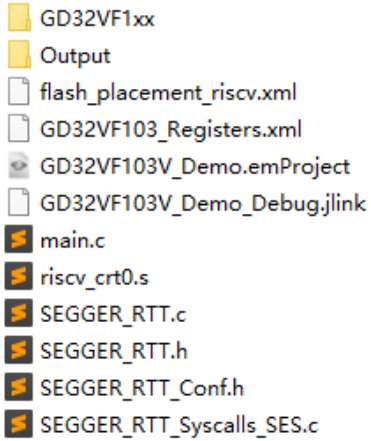
* 裸板程序




- Script Files
  - GD32VF1XX_Target.js //调试脚本
- Source Files
  - main.c
- System Files
  - entry.s
  - gd32vf103.h
  - gd32vf103_eclic.c
  - GD32VF1xx_Stertup.s
  - init.c
  - n22_func.c //内核文件
  - riscv_crt0.s
  - system_gd32vf103.c

# Embedded Studio 实例演示

* 裸板程序




- GD32VF1xx
  - **Device**
    - **Include**
      - **gd32vf103.h**
      - **gd32vf103_xxx.h**
      - n22_eclic.h
      - n22_func.h
      - n22_tmr.h
      - riscv_bits.h
      - riscv_const.h
      - riscv_encoding.h
      - system_gd32vf103.h
    - **Source**
      - entry.s
      - gd32vf103_eclic.c
      - init.c
      - n22_func.c
      - system_gd32vf103.c
  - **Scripts**
    - GD32VF1xx_Target.js
  - **Source**
    - GD32VF1xx_Startup.s

**flash_placement_riscv.xml**

**GD32VF103_Registers.xml**

GD32VF103V_Demo.emProject

GD32VF103V_Demo_Debug.jlink

**main.c**

riscv_crt0.s

# Embedded Studio 实例演示

* 修改ESE下工程结构

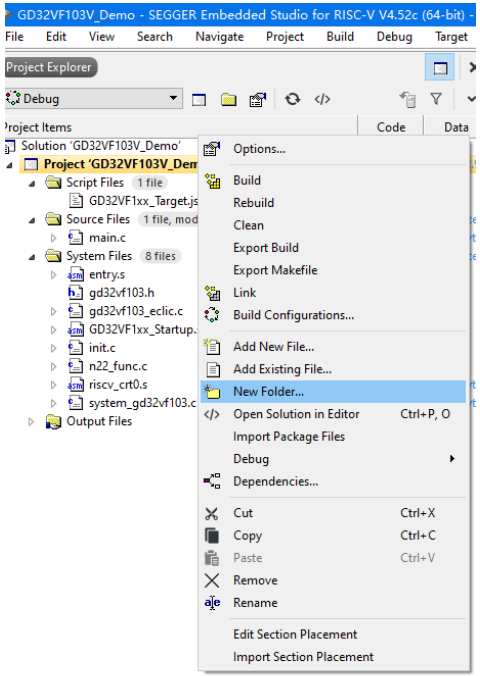
右键工程，点击New Folder添加存放固件库文件的文件夹：

- Device
  - GD32_Std_Per

- 在Source Files下，删除main.c，添加BSP文件夹用于存放开发板支持文件

- 删除System Files下的gd32vf103.h，gd32vf103_eclic.c

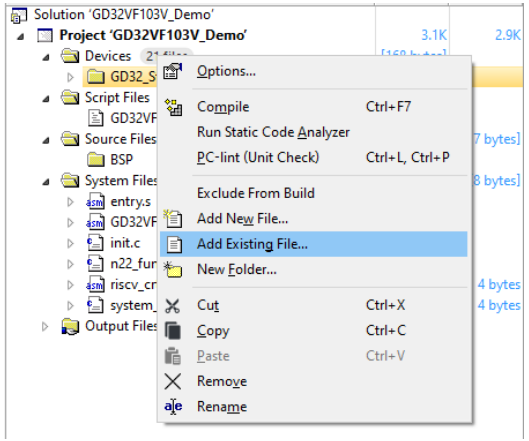
并且删除计算机工程目录中GD32VF1xx\Device\Include文件夹下的gd32vf103.h与gd32vf103_xxx.h文件



# Embedded Studio 实例演示

* 添加源代码

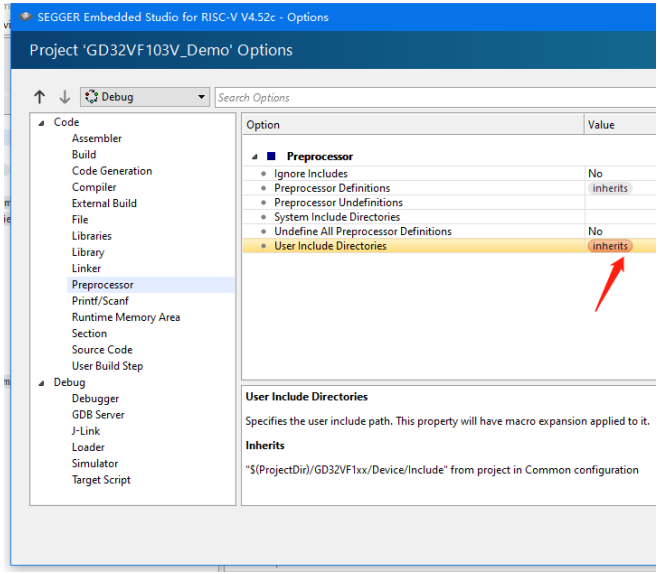
- 右键文件夹，点击Add Existing File添加源代码。
- 将GD32VF103_Firmware_Library\GD32VF103_standard_peripheral\Source文件夹下的C文件添加到工程的Devices\GD32_Std_Per文件夹下
- 将GD32VF103_Demo_Suites_V1.0.3\GD32VF103V_EVAL_Demo_Suites\Projects\01_GPIO_Running_Led下的C文件添加到工程的Source Files文件夹下
- 将GD32VF103_Demo_Suites_V1.0.3\GD32VF103V_EVAL_Demo_Suites\Utilities下的C文件添加到Source Files\BSP文件夹下



# Embedded Studio 实例演示

* 添加头文件路径

右键工程，选择Options在Preprocessor下的User Include Directories下填入头文件路径




$(ProjectDir)\GD32VF103_Demo_Suites_V1.0.3\GD32VF103_Firmware_Library\GD32VF103_standard_peripheral
$(ProjectDir)\GD32VF103_Demo_Suites_V1.0.3\GD32VF103_Firmware_Library\GD32VF103_standard_peripheral\Include
$(ProjectDir)\GD32VF103_Demo_Suites_V1.0.3\GD32VF103V_EVAL_Demo_Suites\Projects\01_GPIO_Running_Led
$(ProjectDir)\GD32VF103_Demo_Suites_V1.0.3\GD32VF103V_EVAL_Demo_Suites\Utilities

# Embedded Studio 实例演示

* 修改文件

移植完成后点击编译，会出现报错，报错显示没有定义开发板，所以这里将开发板定义为GD32VF103V_EVAL

```
/* define value of high speed crystal oscillator (HXTAL) in Hz */
#if !defined  HXTAL_VALUE
  #ifdef GD32VF103R_START
60  #define HXTAL_VALUE    ((uint32_t)25000000) /*!< value of the external oscillato
  #define HXTAL_VALUE_8M  HXTAL_VALUE
62 #else defined(GD32VF103V_EVAL) || defined(GD32VF103C_START) || defined(GD32VF103T_S
  #define HXTAL_VALUE    ((uint32_t)8000000) /*!< value of the external oscillator
  #define HXTAL_VALUE_25M  HXTAL_VALUE
  //#else
  //#error "Please select the target board type used in your application (in gd32v
67  #endif
#endif /* high speed crystal oscillator value */
```

```
#include "gd32vf103_bkp.h"
40 #include "gd32vf103_can.h"
#include "gd32vf103_crc.h"
#include "gd32vf103_dac.h"
#include "gd32vf103_dma.h"
#include "gd32vf103_eclic.h"
#include "gd32vf103_exmc.h"
#include "gd32vf103_exti.h"
#include "gd32vf103_fmc.h"
#include "gd32vf103_gpio.h"
#include "gd32vf103_i2c.h"
50 #include "gd32vf103_fwdgt.h"
#include "gd32vf103_dbg.h"
#include "gd32vf103_pmu.h"
#include "gd32vf103_rcu.h"
#include "gd32vf103_rtc.h"
#include "gd32vf103_spi.h"
#include "gd32vf103_timer.h"
#include "gd32vf103_usart.h"
#include "gd32vf103_wwdgt.h"
59 #include "n200_func.h"
```

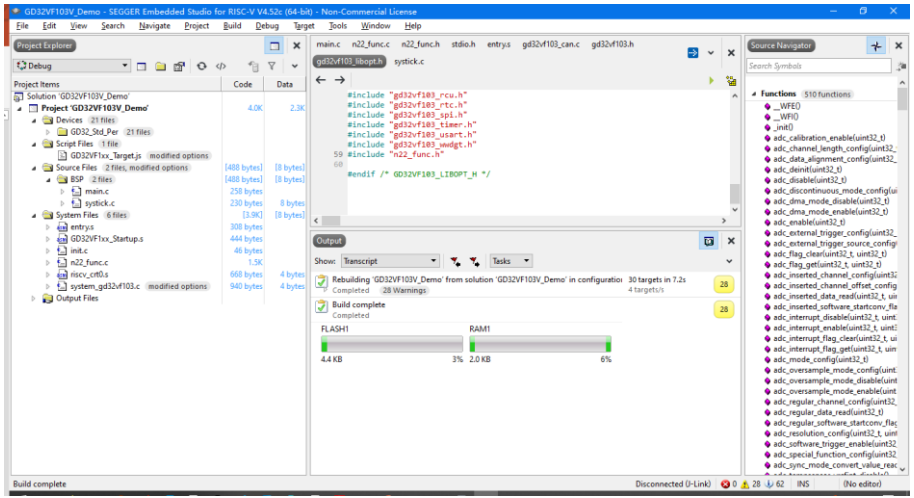
点击重新编译，报错显示没有n200_func.h这个文件

本工程使用的是n22_func.h，所以这里修改为n22_func.h

# Embedded Studio 实例演示

* 下载调试

再点击重新编译，编译通过，可以进行下载调试。



# 解答时间

系列课程后面安排：8月20日，晚8点

第四讲：从零开始学习RTOS 分析工具的使用

欢迎扫码申请入群

演示代码下载链接：

https://eyun.baidu.com/s/3o970fDC

密码：Ma8k

看直播，享福利

单片机与嵌入式系统应用 Microcontrollers & Embedded Systems

eefocus 与非网