

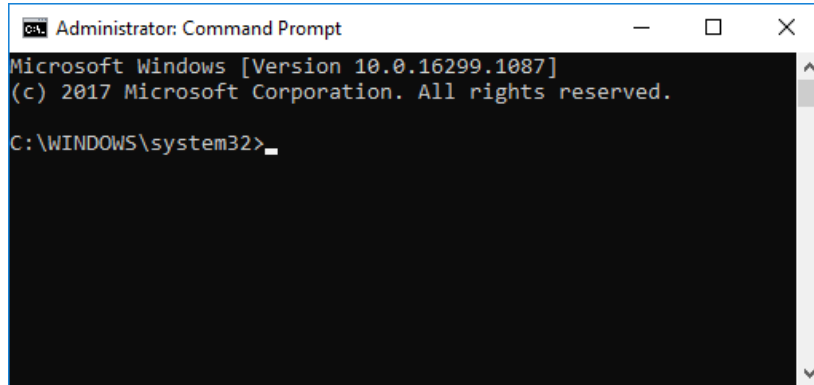
怎样创建 zephyr 工程？

软件要求：Win10, [MCUXpresso11.3.1](#), [LPCScript](#), [jlink v7.22](#), python3

开发板：frdm_k64

1 安装 [Chocolatey](#)

2 以管理员身份打开 cmd (不要使用 powershell)



3 输入以下三条指令

```
>choco feature enable -n allowGlobalConfirmation
> choco install cmake --installargs 'ADD_CMAKE_TO_PATH=System'
> choco install git python ninja gperf
```

4 继续输入

```
>pip3 install -U west
安装 west
```

5 开始 clone 仓库, **userprofile** 是用户目录, 一般是 **c/user/name** 这个目录, name 是用户名

```
> cd %userprofile%
> west init --mr v2.6.0 zephyrproject
> cd zephyrproject
> west update
west update 有的文件可能下不下来, 需要多试几次。
```

6 导出 Zephyr CMake 软件包。允许 CMake 自动加载构建 Zephyr 应用程序所需的样板代码

```
>west zephyr-export
```

7 安装 python 依赖

```
>pip3 install -r zephyr/scripts/requirements.txt
```

8 重新打开 cmd, 不需要管理员权限, 确保已经安装了 mcuxpresso 11.3.1, jlink 版本大于 V7

- 9 在 **userprofile** 下，创建一个文件 **zephyrrc.cmd**，
里面输入如下，根据自己工具安装位置设定
set ZEPHYR_TOOLCHAIN_VARIANT=gnuarmemb
set GNUARMEMB_TOOLCHAIN_PATH=C:\nxp\MCUXpressoIDE_11.3.1_5262\ide\tools
set PATH=%PATH%;C:\Program Files (x86)\SEGGER\JLink_V722

10 设置 build 环境

```
> cd %userprofile%\zephyrproject\zephyr  
> zephyr-env.cmd
```

zephyr-env.cmd 会调用 zephyrrc.cmd，所以要确保有这个文件

11 创建 led 应用

```
> west build -b frdm_k64f -d build\blink samples\basic\blink
```

会生成用于烧写的 elf 文件在 build\blink 目录下

12 west flash -d build\blink 这样就可以烧写这个文件

但是我们实际上需要一个完整的工程来开发代码，这时候可以这样生成工程

13 进入 cd %userprofile%\zephyrproject\zephyr，使用 zephyr-env.cmd，之前做过了可以省略

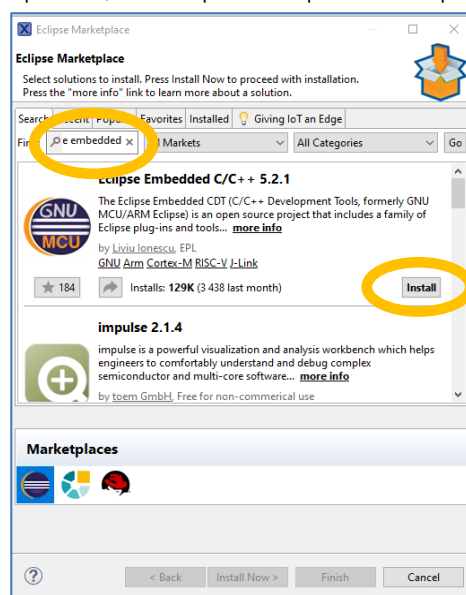
14 进入 userprofile

```
> cd %userprofile%
```

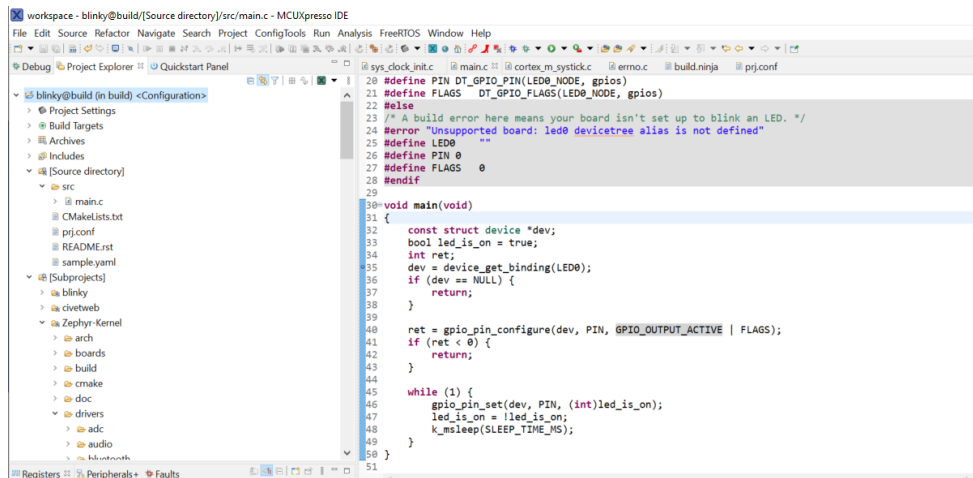
15 生成工程

```
> west build -b frdm_k64f %ZEPHYR_BASE%\samples\hello_world -G"Eclipse CDT4 - Ninja"
```

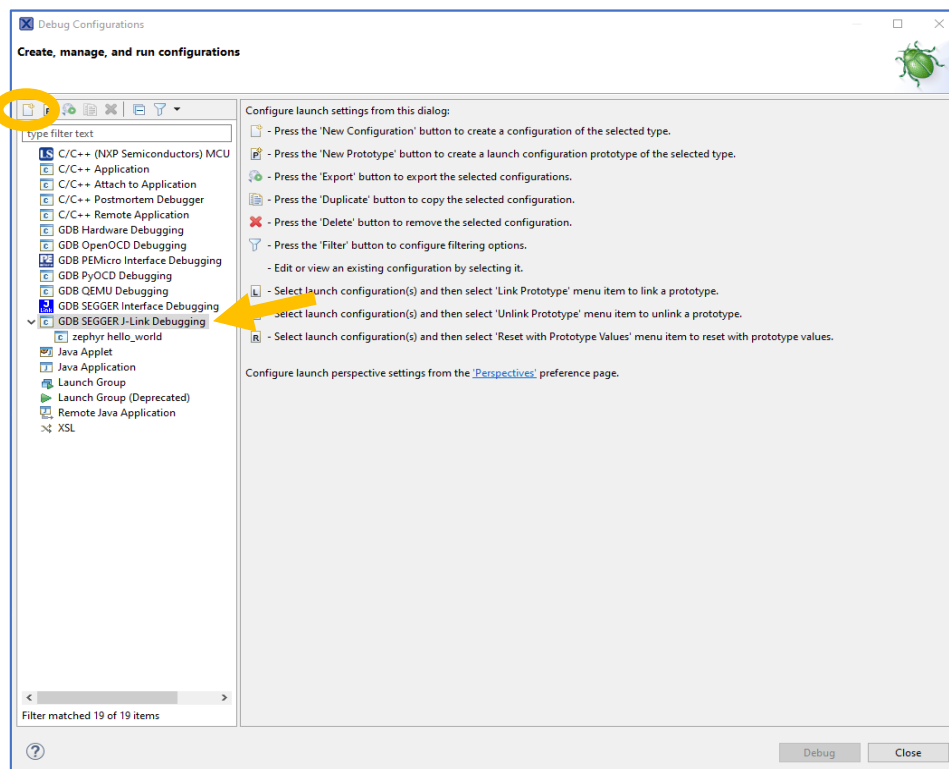
16 进入 mcuxpresso，到 help-> Eclipse Marketplace 找到 *eclipse embedded* 插件，安装



17 打开工程 mcuxpresso, 下面 build 和 debug 的按钮都是灰色的, 无法使用, 要 build 工程, 要右键工程->build project



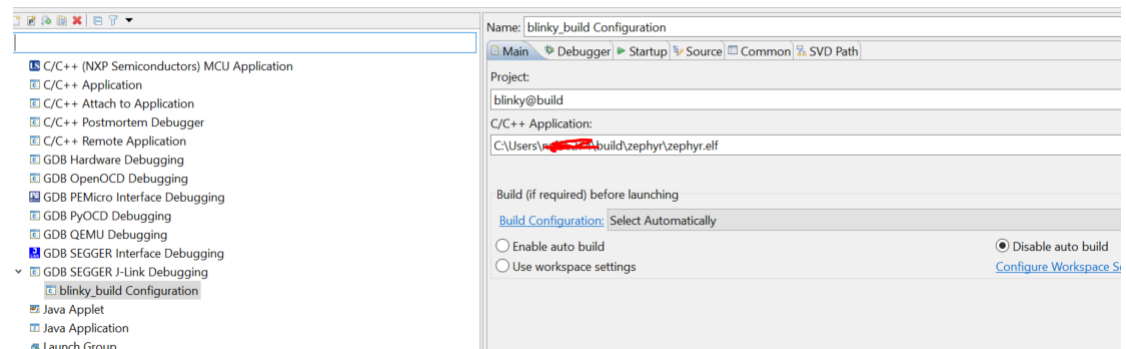
18 打开 debug configuration (run-> Debug Configurations), 新建一个标签在 GDB SEGGER J-Link Debugging



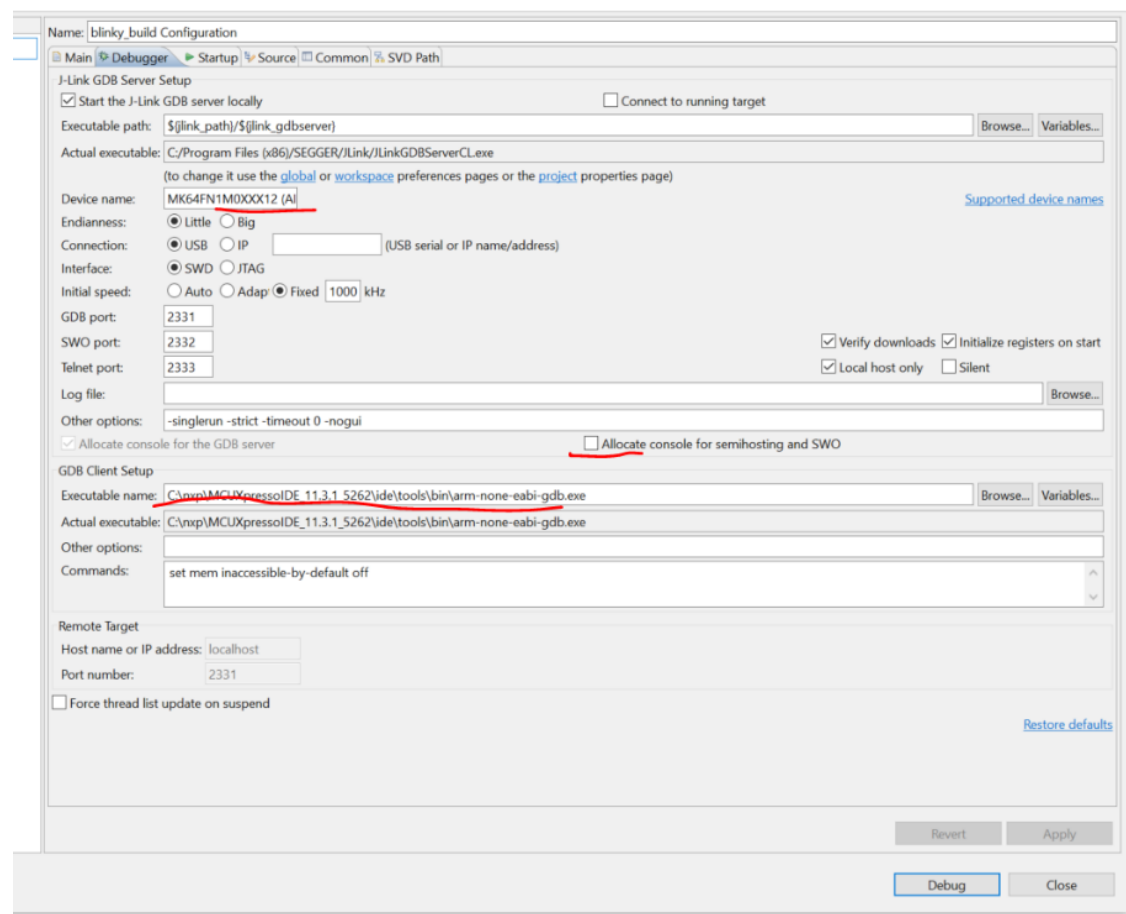
19 新建以后，在 main 标签里，将 Project 和 C/C++ Application 填好

Debug Configurations

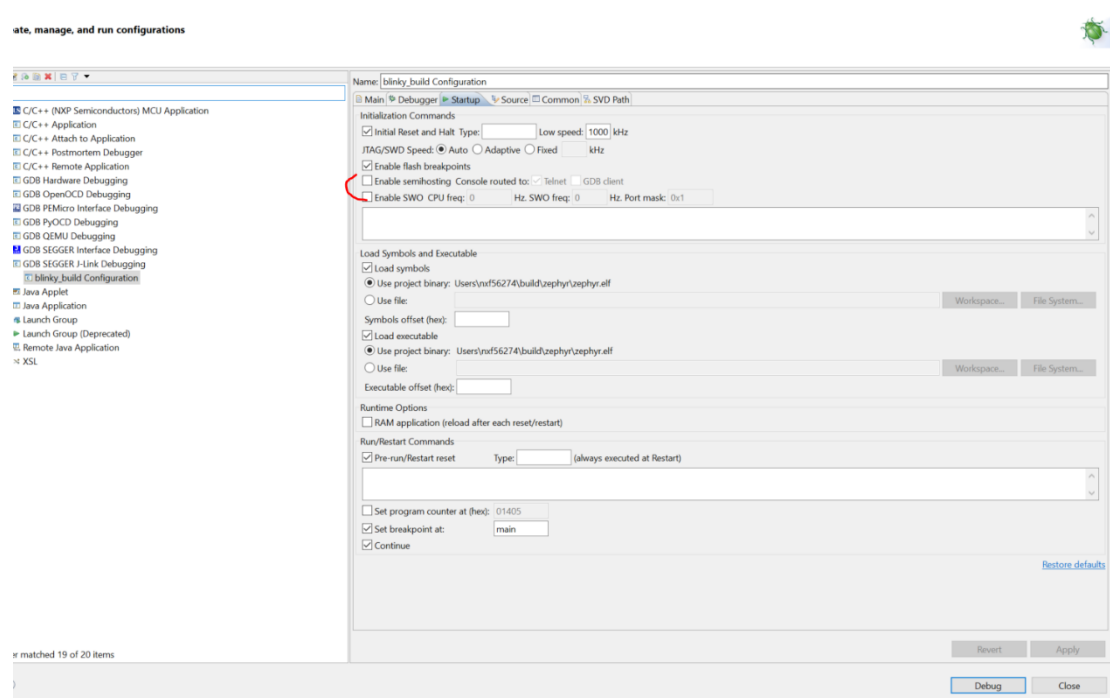
Create, manage, and run configurations



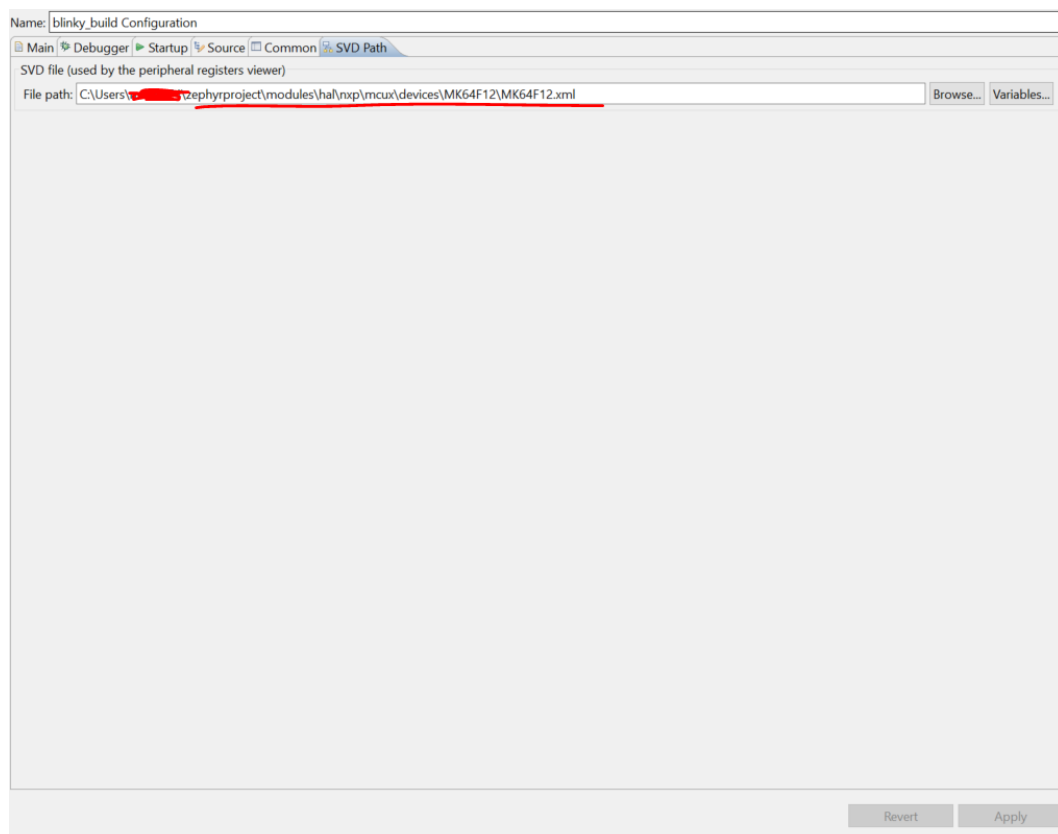
20 在 debugger 选项里，加入 device name，以及其他



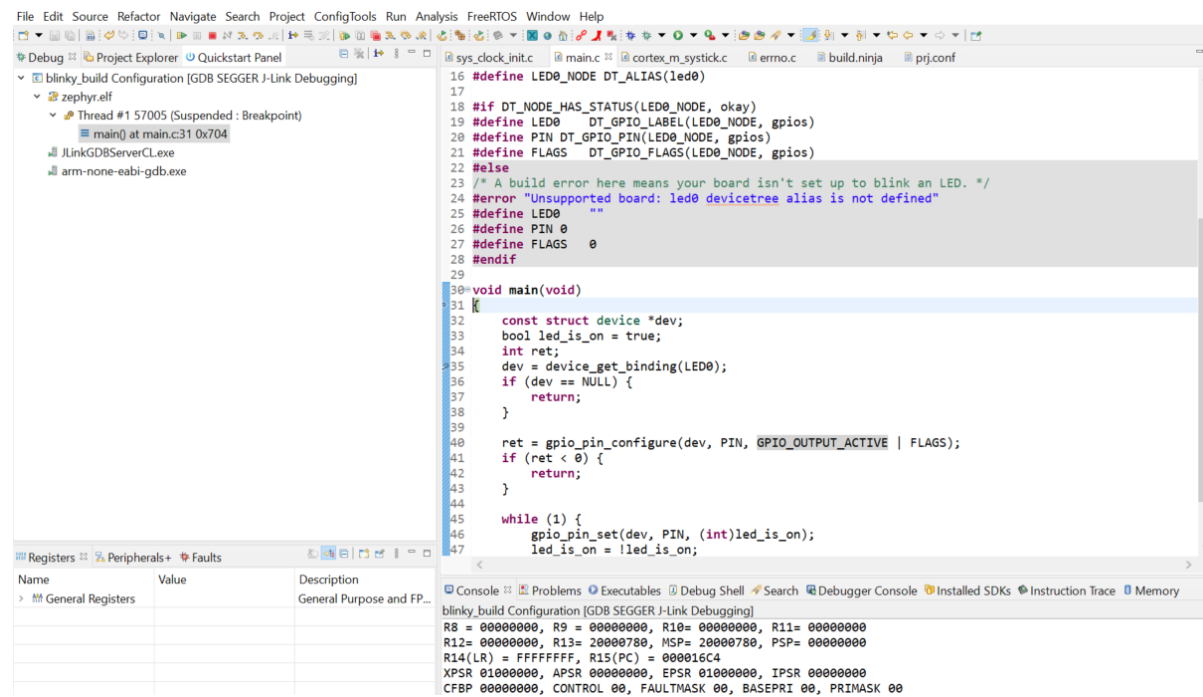
21 在 startup 里，取消勾选



22 添加 SVD 文件，文件在 zephyrproject\modules\hal\nxp\mcux\devices 目录下



23 最后点击 debug



The screenshot shows the Visual Studio Code IDE with the Zephyr project open. The Project Explorer on the left shows the file structure, including 'zephyr.elf' and 'main.c'. The Source Explorer in the center shows the code for 'main.c', with a breakpoint set at line 31. The Console panel at the bottom displays the output of the 'blinky_build Configuration [GDB SEGGER J-Link Debugging]' command, showing various system registers and their values.

```
16 #define LED0_NODE DT_ALIAS(led0)
17
18 #if DT_NODE_HAS_STATUS(LED0_NODE, okay)
19 #define LED0 DT_GPIO_LABEL(LED0_NODE, gprios)
20 #define PIN DT_GPIO_PIN(LED0_NODE, gprios)
21 #define FLAGS DT_GPIO_FLAGS(LED0_NODE, gprios)
22 #else
23 /* A build error here means your board isn't set up to blink an LED. */
24 #error "Unsupported board: led0 devicetree alias is not defined"
25 #define LED0 ""
26 #define PIN 0
27 #define FLAGS 0
28 #endif
29
30 void main(void)
31 {
32     const struct device *dev;
33     bool led_is_on = true;
34     int ret;
35     dev = device_get_binding(LED0);
36     if (dev == NULL) {
37         return;
38     }
39
40     ret = gpio_pin_configure(dev, PIN, GPIO_OUTPUT_ACTIVE | FLAGS);
41     if (ret < 0) {
42         return;
43     }
44
45     while (1) {
46         gpio_pin_set(dev, PIN, (int)led_is_on);
47         led_is_on = !led_is_on;
```

Registers: Peripherals: Faults

Name	Value	Description
General Registers		General Purpose and FP...

Console: Problems: Executables: Debug Shell: Search: Debugger Console: Installed SDKs: Instruction Trace: Memory

blinky_build Configuration [GDB SEGGER J-Link Debugging]

R8 = 00000000, R9 = 00000000, R10 = 00000000, R11 = 00000000
R12 = 00000000, R13 = 20000780, MSP = 20000780, PSP = 00000000
R14(LR) = FFFFFFFF, R15(PC) = 000016C4
XPSR 01000000, APSR 00000000, EPSR 01000000, IPSR 00000000
CFBP 00000000, CONTROL 00, FAULTMASK 00, BASEPRI 00, PRIMASK 00