

Windows下使用makefile + GNU tools for arm + Openocd 开发MSP432P401R

写在前面

正式开始前，我想说一下为何要抛弃Windows一些优秀的IDE，而使用复杂的make来开发。一开始拿到这块开发板的时候，我是用TI官方的CCS（Code Composer Studio）来点灯的。但是我发现这款IDE做的并不是很优秀，体验感并不是很好，具体就是代码补全不如其他的IDE，工程的创建也不如RTT的IDE（RT-Thread Studio）。在网上搜索开发环境后，发现大家有的用MDK，有的用IAR，还有用eclipse的（文末细嗦）。无一例外，都是IDE，然而我想折腾一下，想要了解IDE“一键编译”的背后，熟悉底层的原理，所以我选择了makefile，同时搭配VScode的编辑功能来开发。

目录

1. 简介

- Makefile
- GNU tools for arm
- Openocd

2. 开发环境搭建

- GNU tools for arm 工具的安装
- make 工具的安装
- Openocd 的安装
 - .cfg文件配置
- VScode 的安装
- Simplelink库下载
- 新建点灯工程
 - makefile的编写
 - 链接脚本文件的修改
 - 编写点灯代码

3. 结语

一、简介

• Makefile

简单来说，Makefile就是一个辅助工具，它可以根据设定来执行一些指令，这些指令是在命令行窗口或是shell里执行的。它就像一个脚本，一旦编写完，只需要在一个make命令，就可以完成整个工程的编译。具体语法请大家自行百度，或者参照我文末给出的文档-跟我一起写Makefile。在大家了解学习makefile之前，我想一些基础的C语言编译链接的知识是需要的。

• GNU tools for arm

对于GCC，大家可能并不陌生，它是编译源代码的工具，全称叫GNU Compiler Collection，而这个GNU的全称是GNU's Not UNIX，是GNU的递归缩写。对GNU计划感兴趣的同学可以百度一下GNU，很有意思。回归正题，这个tools for arm也叫交叉编译链，交叉是因为windows/linux的架构和arm不一样，要在x86架构上的主机编译能够在arm架构的机子运行的可执行文件，就要用到交叉编译链。而如果在本机编译出本机的可执行文件，就用不着“交叉”了。有了这个tools，我们才能将写出来的源文件编译成可在msp432上运行的可执行文件。

• Openocd

Openocd 是一个功能强大的软件调试器，并不仅仅可以实现单片机程序的烧录。使用时需配备仿真器如STlink、Jlink或者DAPlink等。Openocd配合cross tools的gdb工具就可以实现对单片机的调试。

二、开发环境搭建

• GNU tools for arm 工具的安装

在官网获取安装包后，双击。根据提示，可以安装在自己指定的文件夹中并记下安装路径，建议在一个总文件夹下存放交叉工具链和make工具的安装。

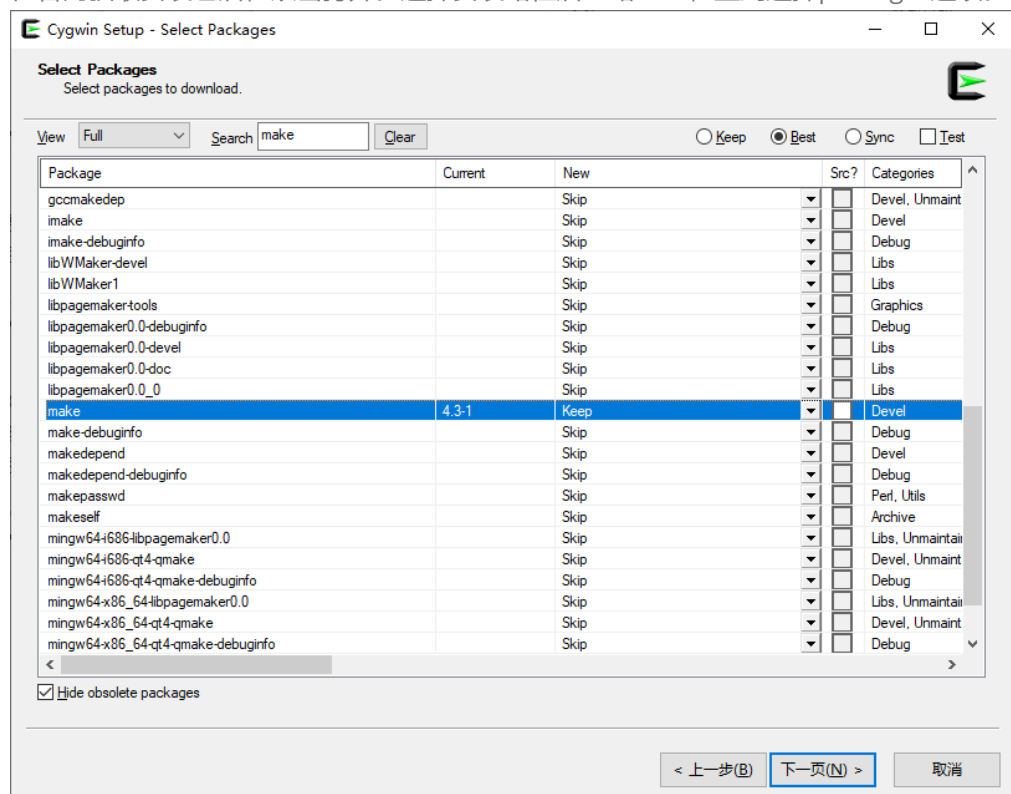
可以在此链接找到安装包[Arm GNU Toolchain | Arm GNU Toolchain Downloads – Arm Developer](#)

下载Windows (mingw-w64-i686) hosted cross toolchains下的[gcc-arm-11.2-2022.02-mingw-w64-i686-arm-none-eabi.exe](#)

安装后将安装路径下的bin文件夹添加到环境变量中。

• make 工具的安装

在官网获取安装包后，双击打开。选择安装路径后一路next，直到选择packages选项。



在make 的new 一栏选择最新版本，之后一路next，并记下安装路径。将安装路径下的bin文件夹添加到环境变量中。

可以在此链接获取安装包[setup-x86_64.exe](#)

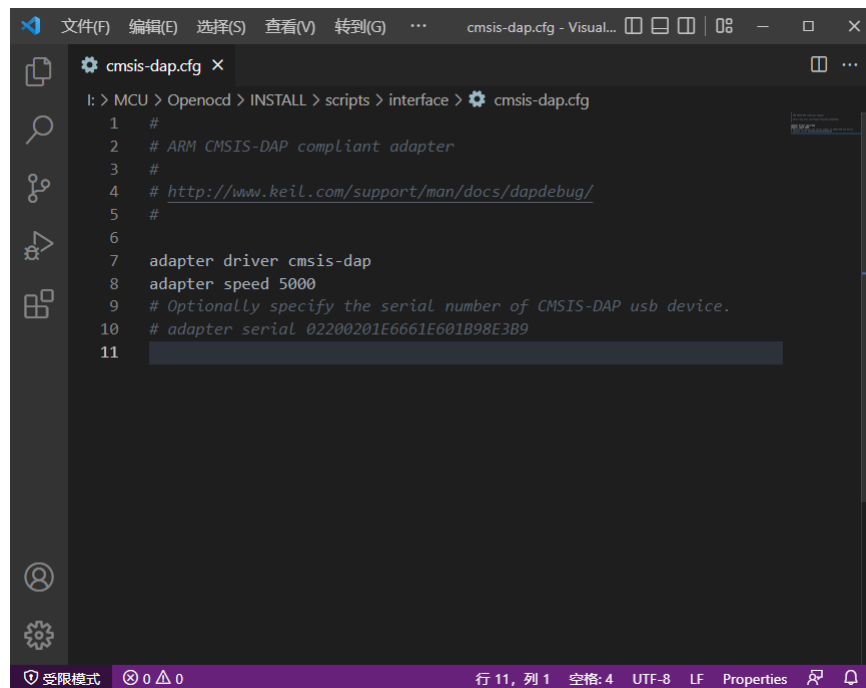
• Openocd 的安装

Openocd可以在github上获取。

[xpack-openocd-0.11.0-4-win32-x64.zip](#)

选择一个路径直接解压，无需安装，并记下解压路径。将解压路径下的bin文件夹添加到环境变量中。

找到安装路径，进入scripts，进入interface，根据你所选的仿真器打开对应的.cfg文件，在其中添加 `adapter speed 5000`。时钟也可以根据你的需求来配置。如：



```
I: > MCU > Openocd > INSTALL > scripts > interface > cmsis-dap.cfg
1 #
2 # ARM CMSIS-DAP compliant adapter
3 #
4 # http://www.keil.com/support/man/docs/dapdebug/
5 #
6
7 adapter driver cmsis-dap
8 adapter speed 5000
9 # Optionally specify the serial number of CMSIS-DAP usb device.
10 # adapter serial 02200201E6661E601B98E3B9
11
```

• VScode 的安装

这个比较简单，官网获取安装包后傻瓜式安装就可以。（全称Visual Studio Code）

可以在此链接获取安装包[direct download link](#)

• Simplelink库下载

simplelink是ti官方推出的SDK，有对应不同开发板的版本，我们可以到官网（[www.ti.com](#)）搜索simplelink msp432的版本。下载完后解压得到一个simplelink_msp432p4_sdk_3_40_01_02的文件夹，里面存放的就是一些官方的例程和所需的头文件和链接脚本文件，还有全部的配置外设的库。例程在examples目录下，头文件在source\ti\devices\msp432p4xx\inc目录下，库文件在\source\ti\devices\msp432p4xx\driverlib目录下，还有必需的启动文件和系统初始化文件在\source\ti\devices\msp432p4xx\startup_system_files目录下。

可以在此链接获取安装包[SIMPLELINK-MSP432-SDK Software development kit \(SDK\). TI.com](#)

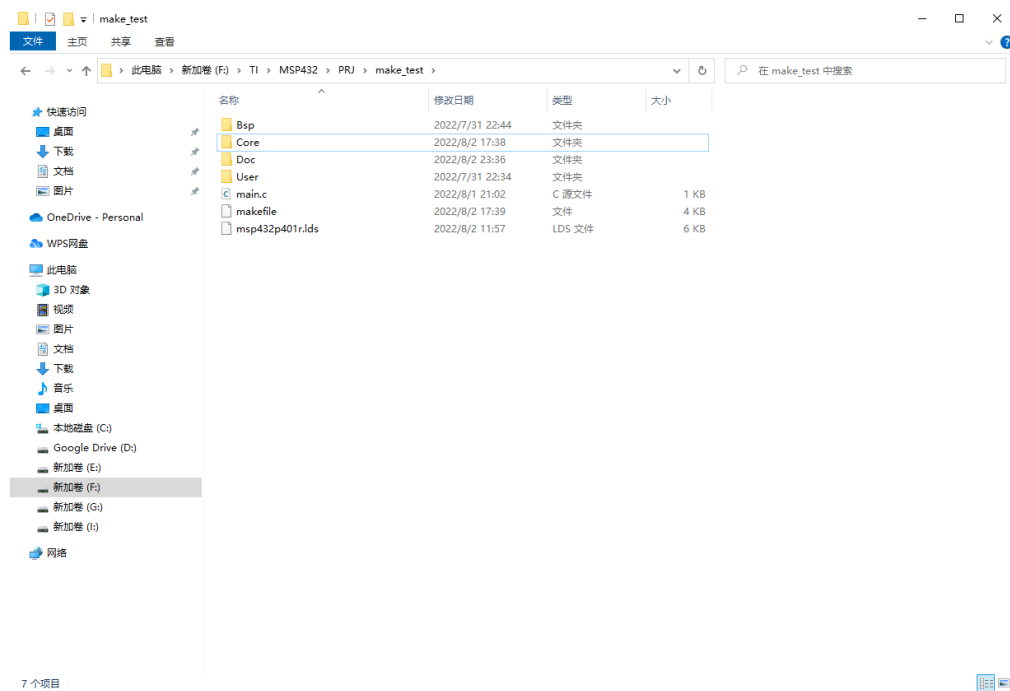
[lockWindows Installer for SimpleLink MSP432P4 SDK — 237908 K](#)

• 新建点灯工程

我们开始建立一个新工程。先建立一个总的工程文件夹，以后开发的别的工程文件夹可放在这个总文件夹里，也可称为工作区。在工作区内新建工程文件夹make_test，工程文件夹下新建四个文件夹Bsp，User，Core和Doc，其中Bsp用来存放板级支持包（也就是simplelink里配置外设的库文件）；User用来存放用户（也就是我们自己）写的代码，例如点灯的代码；Core用来存放与cortex内核有关的文件和官方给的关于msp432的一些文件；Doc用来存放文档；最外层可以放链接文件，makefile等与编译链接有关的文件。

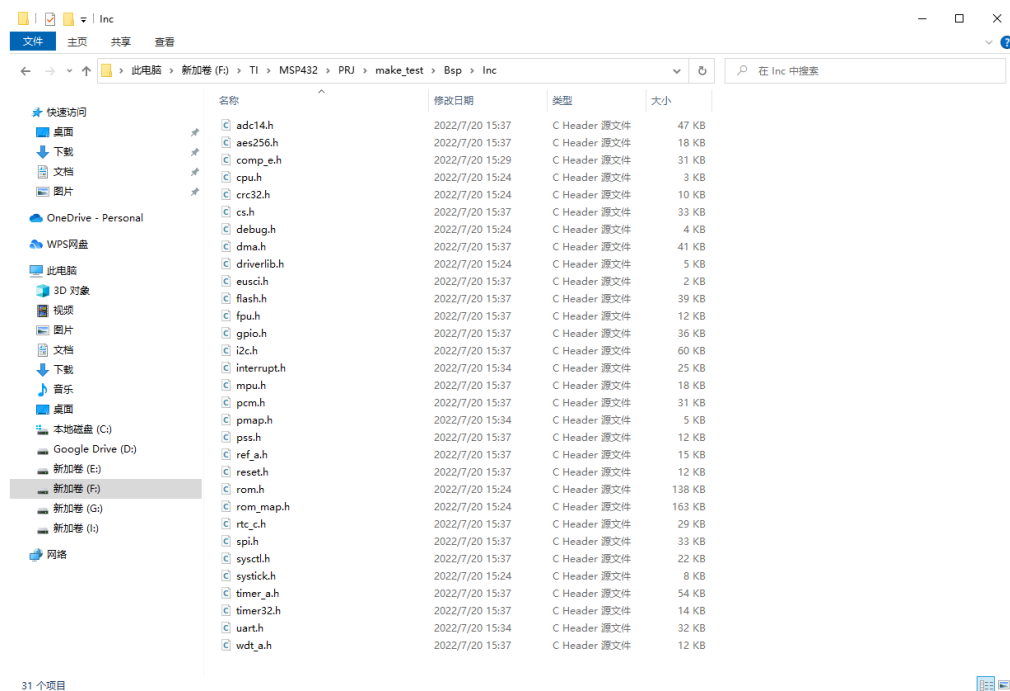
具体如下：

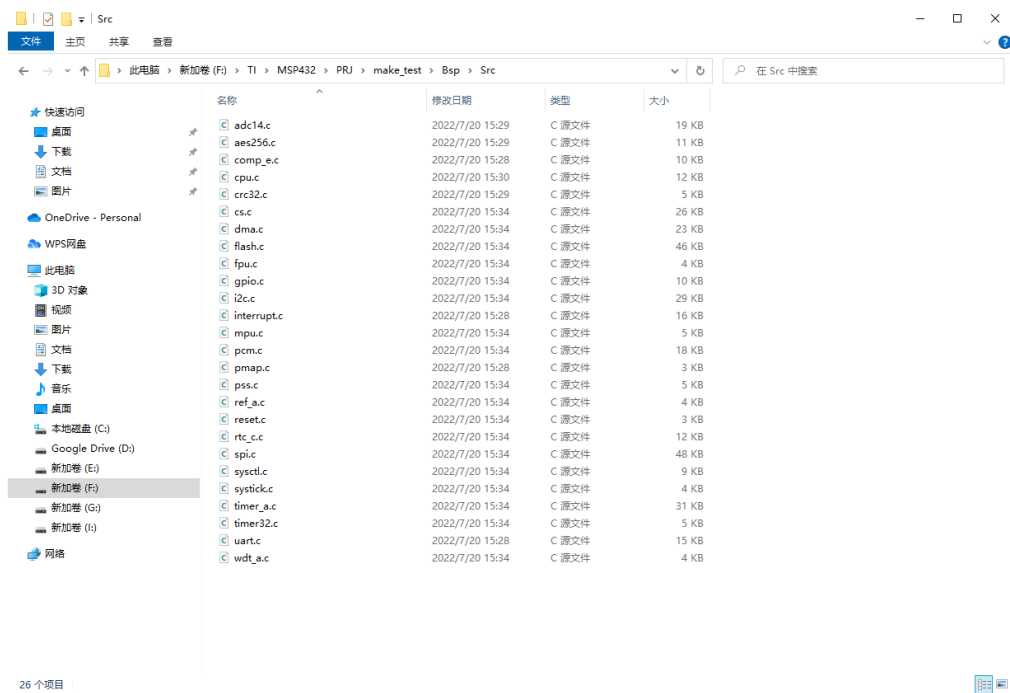
工程文件夹make_test



Bsp文件夹

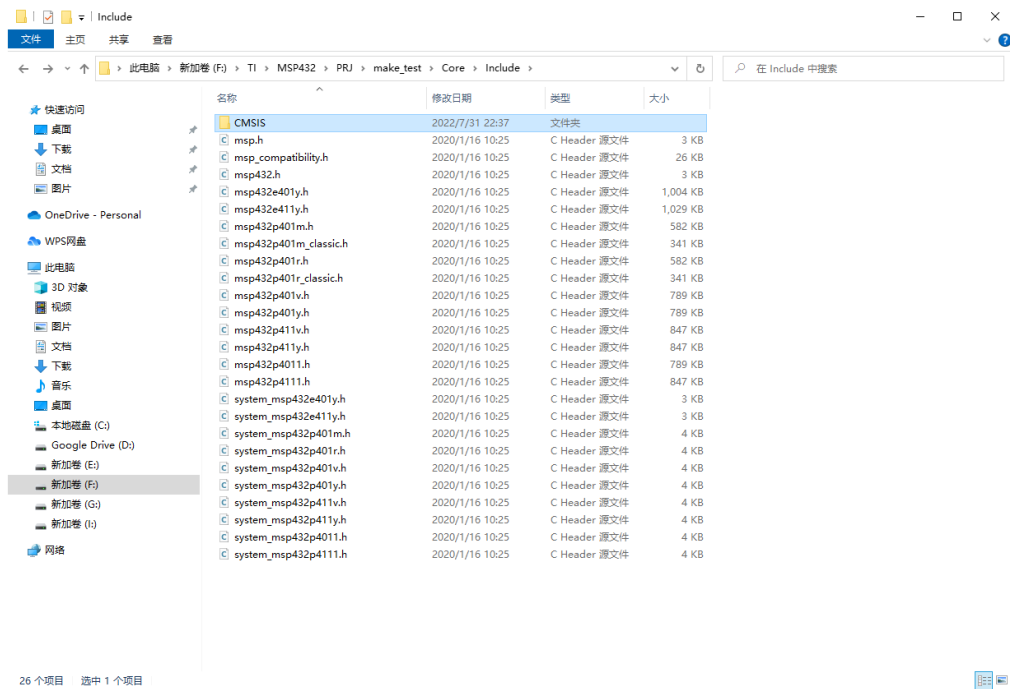
将simple的\source\ti\devices\msp432p4xx\driverlib文件夹下的.c和.h文件复制进来，并分为头文件和源文件。

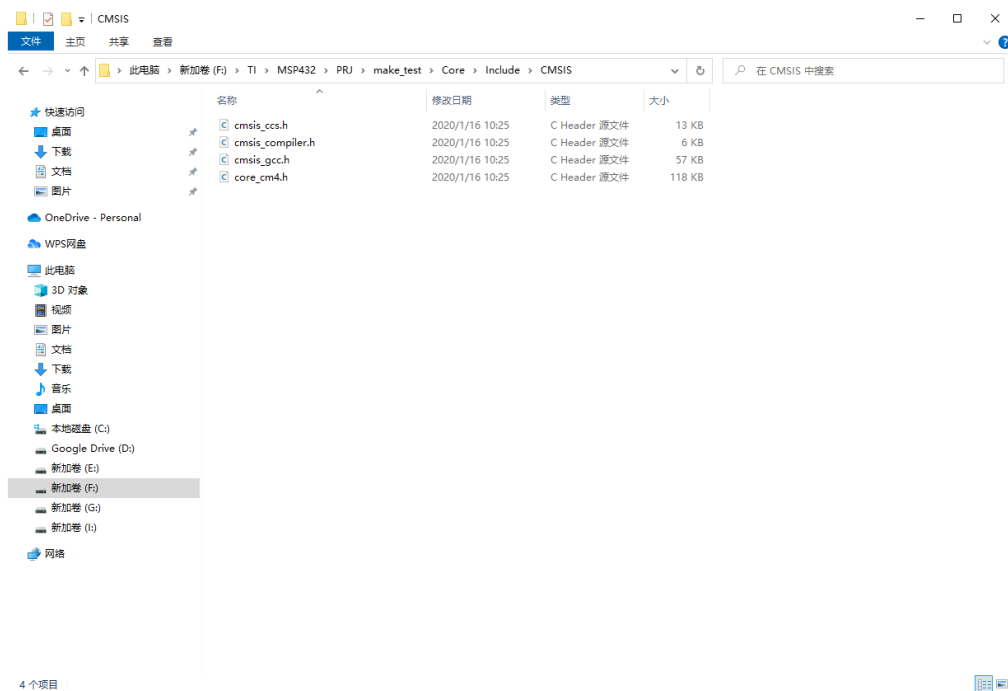




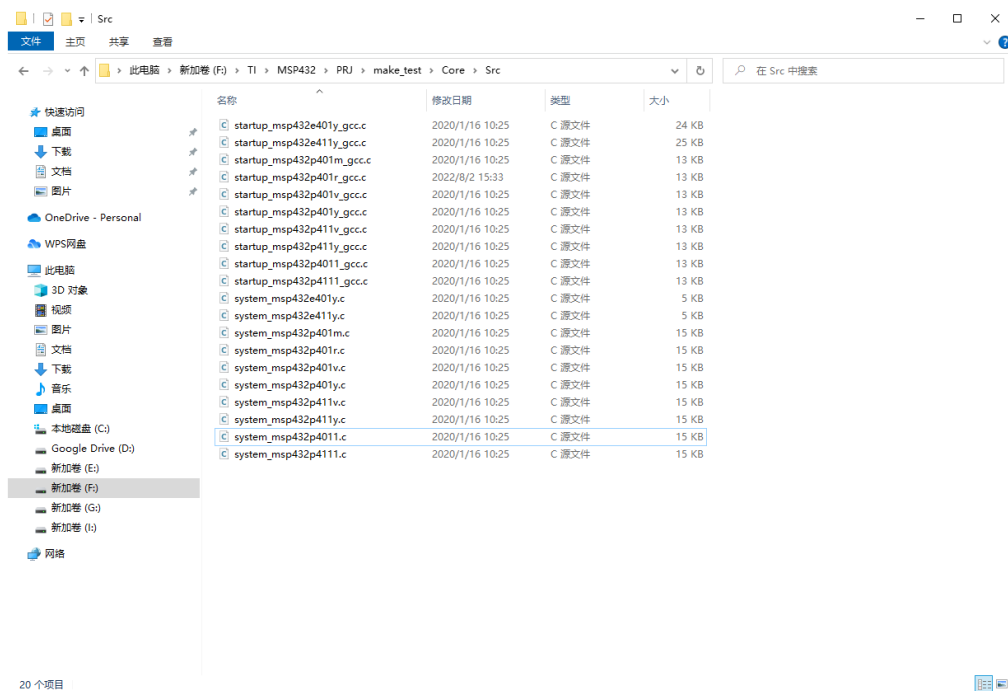
Core文件夹分为Include和Src两个文件夹，其中

Inc：在simplelink里的source\ti\devices\msp432p4xx\inc目录下复制进来所需文件。





Src: 在source\ti\devices\msp432p4xx\startup_system_files下找到所需源文件并复制进来。



User暂时不需要存放文件，点灯代码直接写在main.c里。

Doc也暂时不需要。

最后在simplelink的source\ti\devices\msp432p4xx\linker_files\gcc下找到p401r的.lds文件，复制到工程文件夹下，同时新建main.c文件和makefile文件。

◦ makefile 的编写

这个需要注意，如果按照上面的步骤配置完后，C_SOURCES和C_INCLUDES不用改，需要更改的是最后 make connect 后面的参数，-f后要根据自己的解压路径来。

```
1 #2022/7/31 22:56
2 #make-test
3
```

```
4      #fake target define
5      .PHONY: clean connect
6
7      #define tools
8      PREFIX = arm-none-eabi-
9      # The gcc compiler bin path can be either defined
      in make command via GCC_PATH variable (> make
      GCC_PATH=xxx)
10     # either it can be added to the PATH environment
      variable.
11     ifdef GCC_PATH
12     CC = $(GCC_PATH)/$(PREFIX)gcc
13     AS = $(GCC_PATH)/$(PREFIX)gcc -x assembler-with-cpp
14     CP = $(GCC_PATH)/$(PREFIX)objcopy
15     SZ = $(GCC_PATH)/$(PREFIX)size
16     LD = $(GCC_PATH)/$(PREFIX)ld
17     else
18     CC = $(PREFIX)gcc
19     AS = $(PREFIX)gcc -x assembler-with-cpp
20     CP = $(PREFIX)objcopy
21     SZ = $(PREFIX)size
22     LD = $(PREFIX)ld
23     endif
24
25     # cpu
26     CPU = -mcpu=cortex-m4
27
28     # fpu
29     FPU = -mfpu=fpv4-sp-d16
30
31     # float-abi
32     FLOAT-ABI = -mfloat-abi=hard
33
34     # mcu
35     MCU = $(CPU) -mthumb $(FPU) $(FLOAT-ABI)
36
37     #define project name
38     PRJ_NAME = make_test
39     #build, contents obj file
40     BUILD_DIR = build
41
42     #sources
43     C_SOURCES = \
44     /Bsp/Src/adc14.c \
45     /Bsp/Src/aes256.c \
46     /Bsp/Src/comp_e.c \
47     /Bsp/Src/cpu.c \
48     /Bsp/Src/crc32.c \
49     /Bsp/Src/cs.c \
50     /Bsp/Src/dma.c \
51     /Bsp/Src/flash.c \
52     /Bsp/Src/fpu.c \
53     /Bsp/Src/gpio.c \
54     /Bsp/Src/i2c.c \
55     /Bsp/Src/interrupt.c \
```

```

56 /Bsp/Src/mpu.c \
57 /Bsp/Src/pcm.c \
58 /Bsp/Src/pmap.c \
59 /Bsp/Src/pss.c \
60 /Bsp/Src/ref_a.c \
61 /Bsp/Src/reset.c \
62 /Bsp/Src/rtc_c.c \
63 /Bsp/Src/spi.c \
64 /Bsp/Src/sysctl.c \
65 /Bsp/Src/systick.c \
66 /Bsp/Src/timer32.c \
67 /Bsp/Src/timer_a.c \
68 /Bsp/Src/uart.c \
69 /Bsp/Src/wdt_a.c\
70 /Core/Src/startup_msp432p401r_gcc.c \
71 /Core/Src/system_msp432p401r.c \
72 /main.c \
73 #defines
74 C_DEFS = \
75 -D__MSP432P401R__ \
76 -Dgcc \
77
78 #includes
79 C_INCLUDES = \
80 -ICore/Include \
81 -ICore/Include/CMSIS \
82 -IBsp/Inc
83 #-ICore/tools/ti-cgt-arm_20.2.6.LTS/include
84 #-ICore/tools/ti-cgt-arm_20.2.6.LTS/lib
85
86 #lib link
87 LIB = -lc -lm -lnosys \
88 #-LF:\TI\MSP432\PRJ\make_test\Core\tools\ti-cgt-
89 arm_20.2.6.LTS\lib \
90 -LI:\MCU\GNU_tools_ARM\arm-none-eabi\lib\armv7-m \
91 -LI:\MCU\GNU_tools_ARM\lib\gcc\arm-none-
92 eabi\5.4.1\armv7-m \
93 -LI:\MCU\GNU_tools_ARM\lib\gcc\arm-none-
94 eabi\5.4.1\fpu \
95
96 #link options
97 LDFLAGS = $(MCU) -Tmsp432p401r.lds -
98 specs=nosys.specs $(LIB) -w1,-gc-sections,--cref -
99 w1,-Map=$(BUILD_DIR)/$(PRJ_NAME).map#-nostdlib
100 #compile options
101 CCFLAGS = -c -g $(MCU) $(C_DEFS) $(C_INCLUDES) #
102
103 #path define
104 VPATH = \
105 Core/Include:Core/Include/CMSIS:Bsp/Inc:\
106 Bsp/Src:Core/Src:\
107
108 OBJECTS = $(addprefix $(BUILD_DIR)/,$(notdir
109 $(C_SOURCES:.c=.o)))
110 #vpath %.c $(sort $(dir $(C_SOURCES)))

```



```

105 #OBJECTS += \
106 I:\MCU\GNU_tools_ARM\arm-none-eabi\lib\armv7-
    m\crt0.o \
107 I:\MCU\GNU_tools_ARM\lib\gcc\arm-none-
    eabi\5.4.1\armv7-m\crti.o \
108 I:\MCU\GNU_tools_ARM\lib\gcc\arm-none-
    eabi\5.4.1\armv7-m\crtbegin.o \
109
110 all: $(BUILD_DIR)/$(PRJ_NAME).elf
    $(BUILD_DIR)/$(PRJ_NAME).bin
111
112 $(BUILD_DIR)/$(PRJ_NAME).elf: $(OBJECTS)
113 $(CC) $(OBJECTS) $(LDFLAGS) -o $@
114 $(SZ) $@
115
116 $(BUILD_DIR)/$(PRJ_NAME).bin:
    $(BUILD_DIR)/$(PRJ_NAME).elf
117 $(CP) $< -o binary $@
118
119 $(BUILD_DIR)/%.o: %.c makefile | $(BUILD_DIR)
120 $(CC) $(CCFLAGS) $< -o $@
121
122 $(BUILD_DIR):
123 mkdir $@
124
125 clean:
126 rm -f $(BUILD_DIR)/*.o $(BUILD_DIR)/*.elf
    $(BUILD_DIR)/*.bin
127 connect:
128     openocd -f
    I:\MCU\Openocd\INSTALL\scripts\interface\cmsis-
    dap.cfg -f
    I:\MCU\Openocd\INSTALL\scripts\target\ti_msp432.cfg

```

o 链接脚本文件修改

只需在开头添加 `ENTRY(Reset_Handler)` 指定程序入口即可。

```

1  /*****
    *****/
2  *
3  * Copyright (C) 2012 - 2017 Texas Instruments
    Incorporated - http://www.ti.com/
4  *
5  * Redistribution and use in source and binary
    forms, with or without
6  * modification, are permitted provided that the
    following conditions
7  * are met:
8  *
9  * Redistributions of source code must retain the
    above copyright
10 * notice, this list of conditions and the
    following disclaimer.

```

```

11  *
12  * Redistributions in binary form must reproduce the
    above copyright
13  * notice, this list of conditions and the
    following disclaimer in the
14  * documentation and/or other materials provided
    with the
15  * distribution.
16  *
17  * Neither the name of Texas Instruments
    Incorporated nor the names of
18  * its contributors may be used to endorse or
    promote products derived
19  * from this software without specific prior
    written permission.
20  *
21  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT
    HOLDERS AND CONTRIBUTORS
22  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
    INCLUDING, BUT NOT
23  * LIMITED TO, THE IMPLIED WARRANTIES OF
    MERCHANTABILITY AND FITNESS FOR
24  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
    SHALL THE COPYRIGHT
25  * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
    INDIRECT, INCIDENTAL,
26  * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
    (INCLUDING, BUT NOT
27  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
    SERVICES; LOSS OF USE,
28  * DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
    HOWEVER CAUSED AND ON ANY
29  * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
    LIABILITY, OR TORT
30  * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
    ANY WAY OUT OF THE USE
31  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE
    POSSIBILITY OF SUCH DAMAGE.
32  *
33  * GCC linker script for Texas Instruments
    MSP432P401R
34  *
35  * File creation date: 12/06/17
36  *
37
    *****
    *****/
38  ENTRY(Reset_Handler)
39
40  MEMORY
41  {
42      MAIN_FLASH (RX) : ORIGIN = 0x00000000, LENGTH =
        0x00040000
43      INFO_FLASH (RX) : ORIGIN = 0x00200000, LENGTH =
        0x00004000

```

```

44     SRAM_CODE   (RWX): ORIGIN = 0x01000000, LENGTH =
        0x00010000
45     SRAM_DATA   (RW) : ORIGIN = 0x20000000, LENGTH =
        0x00010000
46 }
47
48     REGION_ALIAS("REGION_TEXT", MAIN_FLASH);
49     REGION_ALIAS("REGION_INFO", INFO_FLASH);
50     REGION_ALIAS("REGION_BSS", SRAM_DATA);
51     REGION_ALIAS("REGION_DATA", SRAM_DATA);
52     REGION_ALIAS("REGION_STACK", SRAM_DATA);
53     REGION_ALIAS("REGION_HEAP", SRAM_DATA);
54     REGION_ALIAS("REGION_ARM_EXIDX", MAIN_FLASH);
55     REGION_ALIAS("REGION_ARM_EXTAB", MAIN_FLASH);

```

```

1  SECTIONS {
2
3      /* section for the interrupt vector area
        */
4      PROVIDE (_intvecs_base_address =
5          DEFINED(_intvecs_base_address) ?
        _intvecs_base_address : 0x0);
6
7      .intvecs (_intvecs_base_address) : AT
        (_intvecs_base_address) {
8          KEEP (*(intvecs))
9      } > REGION_TEXT
10
11     /* The following three sections show the usage of the
        INFO flash memory */
12     /* INFO flash memory is intended to be used for the
        following */
13     /* device specific purposes:
        */
14     /* Flash mailbox for device security operations
        */
15     PROVIDE (_mailbox_base_address = 0x200000);
16
17     .flashMailbox (_mailbox_base_address) : AT
        (_mailbox_base_address) {
18         KEEP (*(flashMailbox))
19     } > REGION_INFO
20
21     /* TLV table for device identification and
        characterization */
22     PROVIDE (_tlv_base_address = 0x00201000);
23
24     .tlvTable (_tlv_base_address) (NOLOAD) : AT
        (_tlv_base_address) {
25         KEEP (*(tlvTable))
26     } > REGION_INFO
27
28     /* BSL area for device bootstrap loader
        */
29     PROVIDE (_bsl_base_address = 0x00202000);

```

```

30
31     .bssArea (_bss_base_address) : AT (_bss_base_address)
32 {
33     KEEP (*( .bssArea))
34 } > REGION_INFO
35
36 PROVIDE (_vtable_base_address =
37     DEFINED(_vtable_base_address) ?
38     _vtable_base_address : 0x20000000);
39
40 .vtable (_vtable_base_address) : AT
41 (_vtable_base_address) {
42     KEEP (*( .vtable))
43 } > REGION_DATA
44
45 .text : {
46     CREATE_OBJECT_SYMBOLS
47     KEEP (*( .text))
48     *( .text.*)
49     . = ALIGN(0x4);
50     KEEP (*( .ctors))
51     . = ALIGN(0x4);
52     KEEP (*( .dtors))
53     . = ALIGN(0x4);
54     __init_array_start = .;
55     KEEP (*( .init_array*))
56     __init_array_end = .;
57     KEEP (*( .init))
58     KEEP (*( .fini*))
59 } > REGION_TEXT AT> REGION_TEXT
60
61 .rodata : {
62     *( .rodata)
63     *( .rodata.*)
64 } > REGION_TEXT AT> REGION_TEXT
65
66 .ARM.exidx : {
67     __exidx_start = .;
68     *( .ARM.exidx* .gnu.linkonce.armexidx.*)
69     __exidx_end = .;
70 } > REGION_ARM_EXIDX AT> REGION_ARM_EXIDX
71
72 .ARM.extab : {
73     KEEP (*( .ARM.extab* .gnu.linkonce.armextab.*))
74 } > REGION_ARM_EXTAB AT> REGION_ARM_EXTAB
75
76 __etext = .;
77
78 .data : {
79     __data_load__ = LOADADDR (.data);
80     __data_start__ = .;
81     KEEP (*( .data))
82     KEEP (*( .data*))
83     . = ALIGN (4);
84     __data_end__ = .;

```

```

82     } > REGION_DATA AT> REGION_TEXT
83
84     .bss : {
85         __bss_start__ = .;
86         *(.shbss)
87         KEEP (*(.bss))
88         *(.bss.*)
89         *(COMMON)
90         . = ALIGN (4);
91         __bss_end__ = .;
92     } > REGION_BSS AT> REGION_BSS
93
94     .heap : {
95         __heap_start__ = .;
96         end = __heap_start__;
97         _end = end;
98         __end = end;
99         KEEP (*(.heap))
100        __heap_end__ = .;
101        __HeapLimit = __heap_end__;
102    } > REGION_HEAP AT> REGION_HEAP
103
104    .stack (NOLOAD) : ALIGN(0x8) {
105        _stack = .;
106        KEEP(*(.stack))
107    } > REGION_STACK AT> REGION_STACK
108
109    __StackTop = ORIGIN(REGION_STACK) +
LENGTH(REGION_STACK);
110    PROVIDE(__stack = __StackTop);
111 }

```

1 | ...

◦ 编写点灯代码

这一部分用到了simplelink库。

```

1  #include "msp.h"
2  #include "gpio.h"
3
4  /**
5   * main.c
6   */
7  int main(void)
8  {
9      WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;
10     // stop watchdog timer
11
12     GPIO_setAsOutputPin(GPIO_PORT_P1, GPIO_PIN0);
13     GPIO_setOutputHighOnPin(GPIO_PORT_P1, GPIO_PIN0);
14     while(1)
15     {

```

```
15  
16     }  
17     return 0;  
18 }  
19
```

做完上面的工作后，整个工程的建立就结束了。要生成可执行文件只需在makefile所在路径打开命令行窗口（或者在makefile所在文件夹右键在VScode中打开，新建终端），输入"make"，就会编译生成.elf 和 .bin文件，再"make connect"，可以看到仿真器和开发板已经connect上了。再打开另一个终端，进入build文件夹，输入 `arm-none-eabi-gdb ./make_test.elf` 即可通过gdb调试目标开发板。进入gdb模式后 `load ./make_test.elf` 可将elf文件烧录进flash， `target extended-remote localhost:3333` 将gdb连接至仿真器， `run` 即可看到开发板灯亮起，说明烧录成功，文件的编译也没有出错。

三、结语

makefile文件我会再出一篇详解，敬请期待！感谢读到这里的人！

2022/08/03 By Budali11