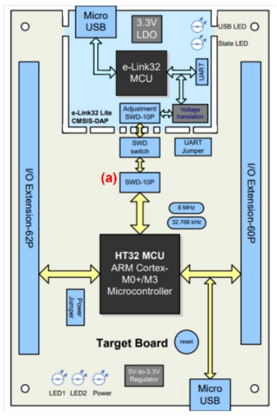
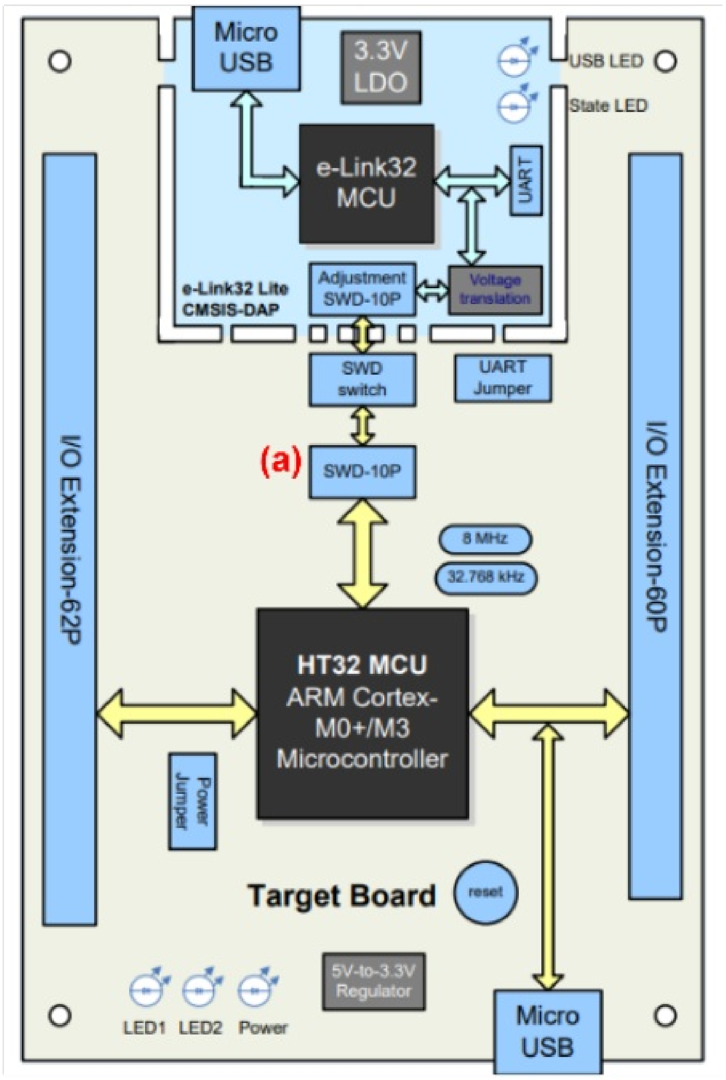


HOLTEK HT32 MCU GNU Arm Compiler User Guide

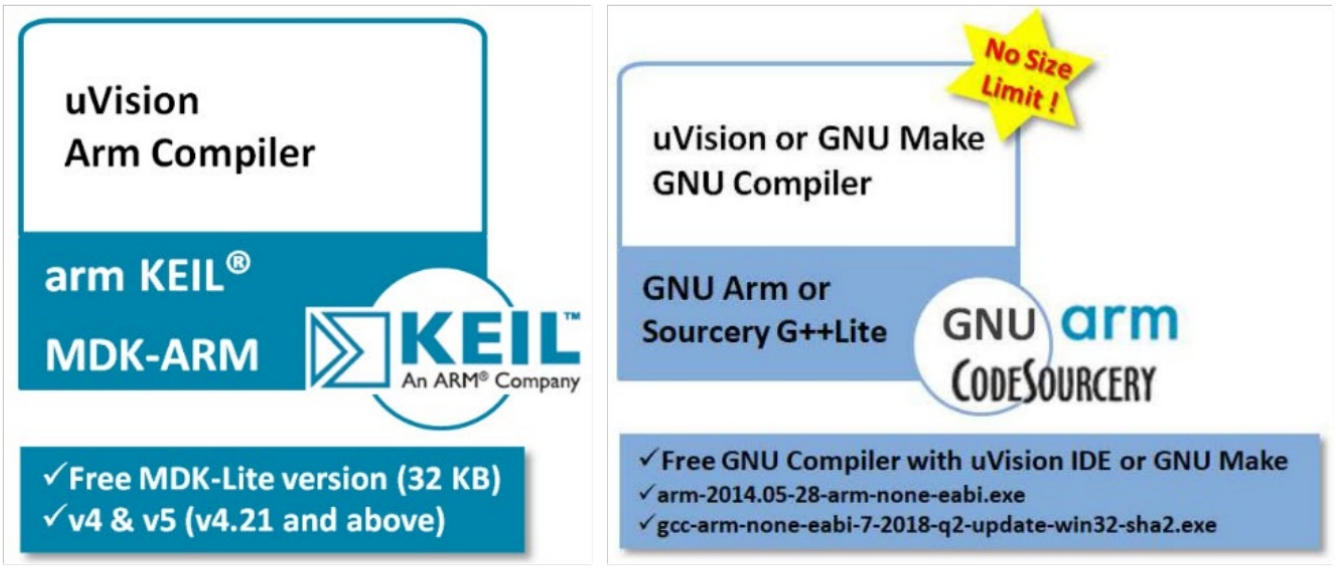
HOLTEK HT32 MCU GNU Arm Compiler


Micro USB
3.3V LDO
USB LED
State LED
e-Link32 MCU
UART
e-Link32 Lite CMSIS-DAP
Adjustment SWD-10P
Voltage translation
SWD switch
UART Jumper
(a)
SWD-10P
8 MHz
32.768 kHz
I/O Extension-62P
I/O Extension-60P
HT32 MCU ARM Cortex-M0+/M3 Microcontroller
Power Jumper
Target Board
reset
5V-to-3.3V Regulator
LED1
LED2
Power
Micro USB

Contents [hide

Introduction

There are many kinds of compiler available, some commonly used ones are the Keil (MDK-ARM), the IAR (EWARM), the GNU (GNU ARM) and so on. If the “GNU” is compared with the “Keil” and “IAR”, the main difference is that the GNU is free to use and the Keil and IAR both have paid for licenses, otherwise there will be program size limit. As can be seen from the following figure, compared with the Keil MDK-ARM, the GNU Arm has the advantage of no size limit and is more convenient to use. This application note will describe how to use the GNU Arm Compiler with HT32 MCUs.
This application note first describes the resource download and preparation. The download file includes a Firmware Library which contains an example program required during the testing process. The example program can transmit messages via the COM port, therefore the terminal software will be used for function selection or status display. The installation and usage of the GNU Arm Compiler will be introduced in sequence and can be used with “GNU Make” or “Keil MDKARM uVision”. Finally, assistance is provided to resolve common problems during installation, allowing users to find a solution when they encounter problems. It also helps users to quickly build an environment for the GNU Arm Compiler to use.


uVision
Arm Compiler
arm KEIL®
MDK-ARM
KEIL™ An ARM® Company
✓Free MDK-Lite version (32 KB)
✓v4 & v5 (v4.21 and above)
No Size Limit !
uVision or GNU Make
GNU Compiler
GNU Arm or
Sourcery G++Lite
GNU arm
CODESOURCERY
✓Free GNU Compiler with uVision IDE or GNU Make
✓arm-2014.05-28-arm-none-eabi.exe
✓gcc-arm-none-eabi-7-2018-q2-update-win32-sha2.exe

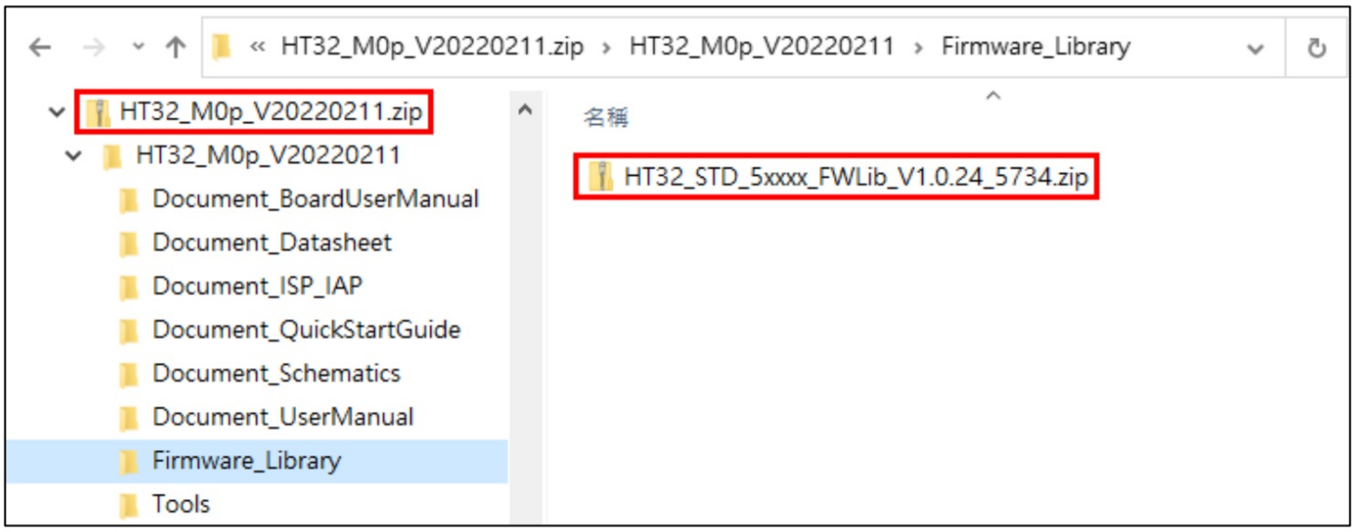
Resource Download and Preparation

This chapter describes the example program and required software tools and explains how to configure the directory and file path.

Firmware Library

Before using the example program, download the latest Holtek HT32 Firmware Library from the following link and then decompress the downloaded file. Ensure that the correct HT32 firmware library has been selected. For example the HT32_M0p_Vyyyymmdd.zip is for the HT32F5xxxx series of MCUs and the HT32_M3_Vyyyymmdd.zip is provided for the HT32F1xxxx series of MCUs.
This compressed file contains several folders which can be categorised as Document, Firmware Library, Tools, etc, which are located in the directory as shown in the following Figure. In the Firmware Library folder is placed the HT32 Firmware Library compressed file named HT32_STD_xxxxx_FWLib_Vm.n.r_s.zip, as shown below.

Download link: https://mcu.holtek.com.tw/ht32/resource/


« HT32_M0p_V20220211.zip › HT32_M0p_V20220211 › Firmware_Library

HT32_M0p_V20220211.zip
- HT32_M0p_V20220211
  - Document_BoardUserManual
  - Document_Datasheet
  - Document_ISP_IAP
  - Document_QuickStartGuide
  - Document_Schematics
  - Document_UserManual
  - Firmware_Library
  - Tools

名稱
HT32_STD_5xxxx_FWLib_V1.0.24_5734.zip

Terminal Program

The Application Code example program can transmit messages via the COM port for function selection or status display. Users can install appropriate communication software on the host PC, such as Tera Term, which is a license free program.

The UART interface configuration in the example program has an 8-bit data format. There is no parity bit. It has one stop bit and a baud rate of 115200.

GNU Arm Compiler Installation

This chapter describes the installation of the GNU Arm Compiler which is explained in the “GNU Arm Installation” and “Test” sections.

GNU Arm Installation

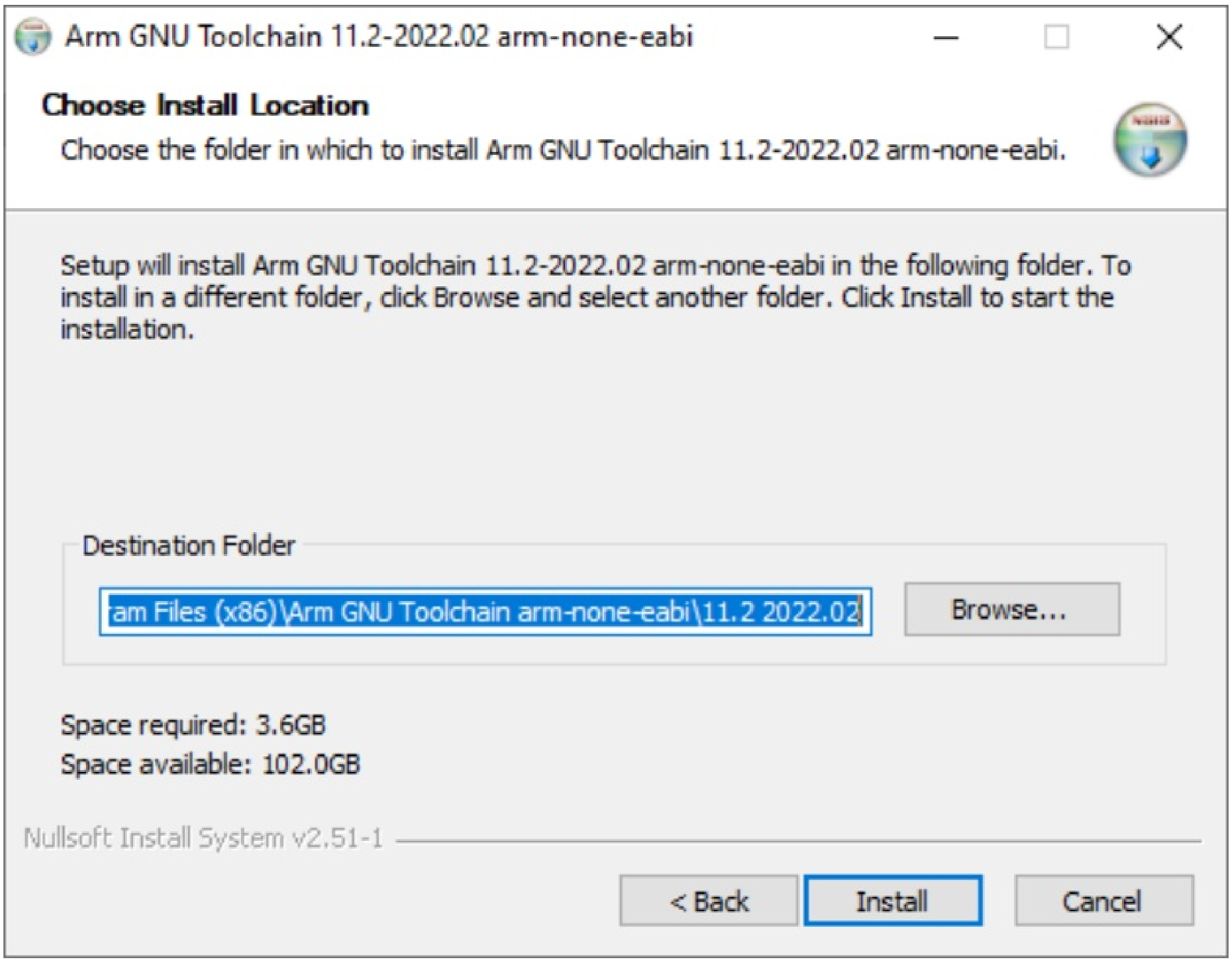
Step 1. Download an EXE file for the GNU Arm installation from the following link.

https://developer.arm.com/open-source/gnu-toolchain/gnu-rm

Note: According to the Arm GNU Toolchain 2022 update information, the previous version is classified as a discontinued file. The file names used in this article and the latest discontinued version are as follows:

The file name used in this article is: “gcc-arm-11.2-2022.02-mingw-w64-i686-arm-none- eabi.exe”.
The file name for the latest discontinued version is: “gcc-arm-none-eabi-10.3-2021.10- win32.exe”.

Step 2. In the “Choose Install Location” step, as shown in the following figure, save the installation path during installation. This path will be configured in Keil in the “Used with Keil MDK- ARM uVision” chapter.
For example:
“C:\Program Files (x86)\Arm GNU Toolchain arm-none-eabi\11.2 2022.02”.


Arm GNU Toolchain 11.2-2022.02 arm-none-eabi

Choose Install Location

Choose the folder in which to install Arm GNU Toolchain 11.2-2022.02 arm-none-eabi.

Setup will install Arm GNU Toolchain 11.2-2022.02 arm-none-eabi in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

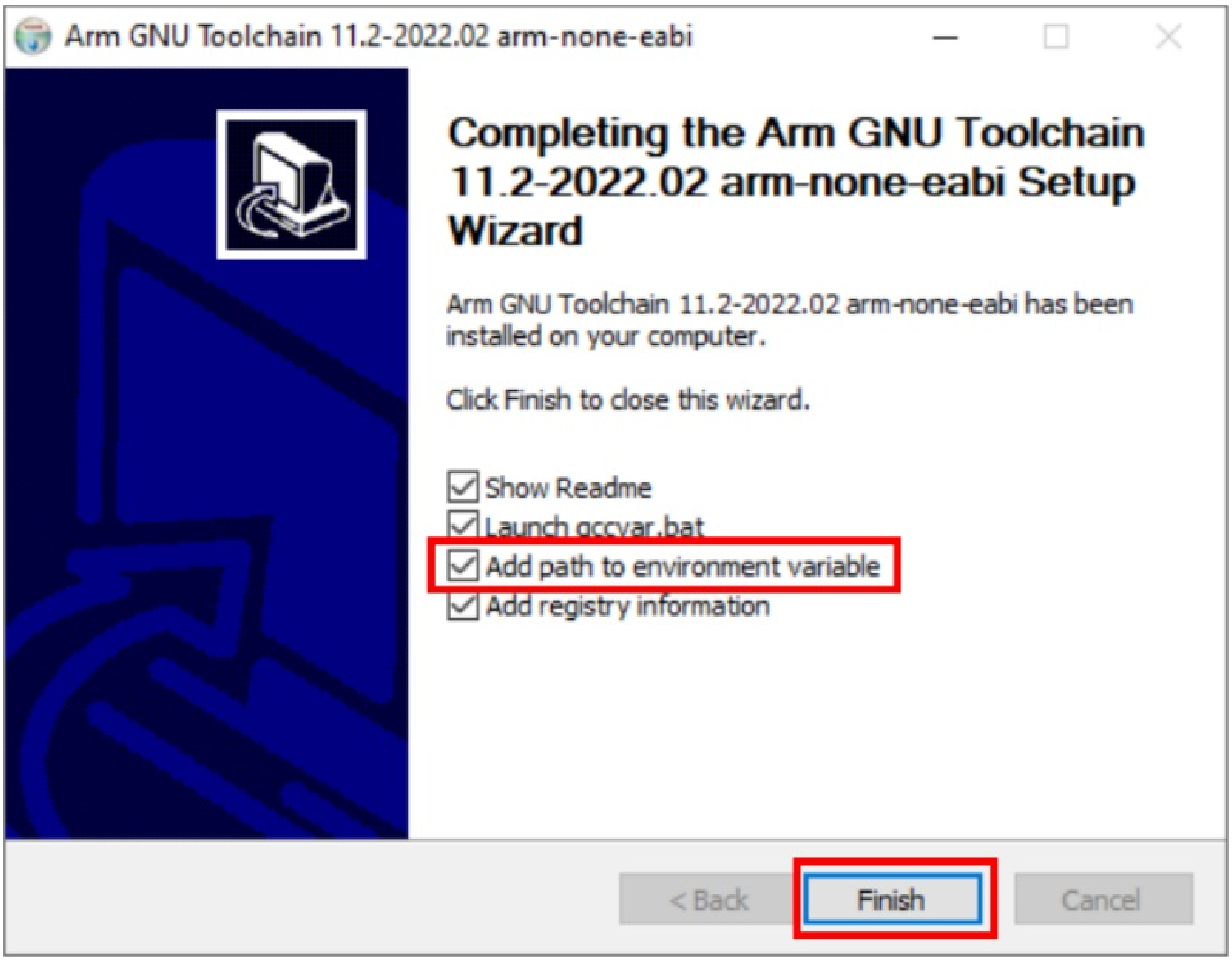
Destination Folder

am Files (x86)\Arm GNU Toolchain arm-none-eabi\11.2 2022.02

Space required: 3.6GB
Space available: 102.0GB

Nullsoft Install System v2.51-1

Step 3. During the last installation step, select “Add path to environment variable” and click “Finish”.


Arm GNU Toolchain 11.2-2022.02 arm-none-eabi

Completing the Arm GNU Toolchain 11.2-2022.02 arm-none-eabi Setup Wizard

Arm GNU Toolchain 11.2-2022.02 arm-none-eabi has been installed on your computer.

Click Finish to close this wizard.

☑ Show Readme
☑ Launch gccvar.bat
☑ Add path to environment variable
☑ Add registry information

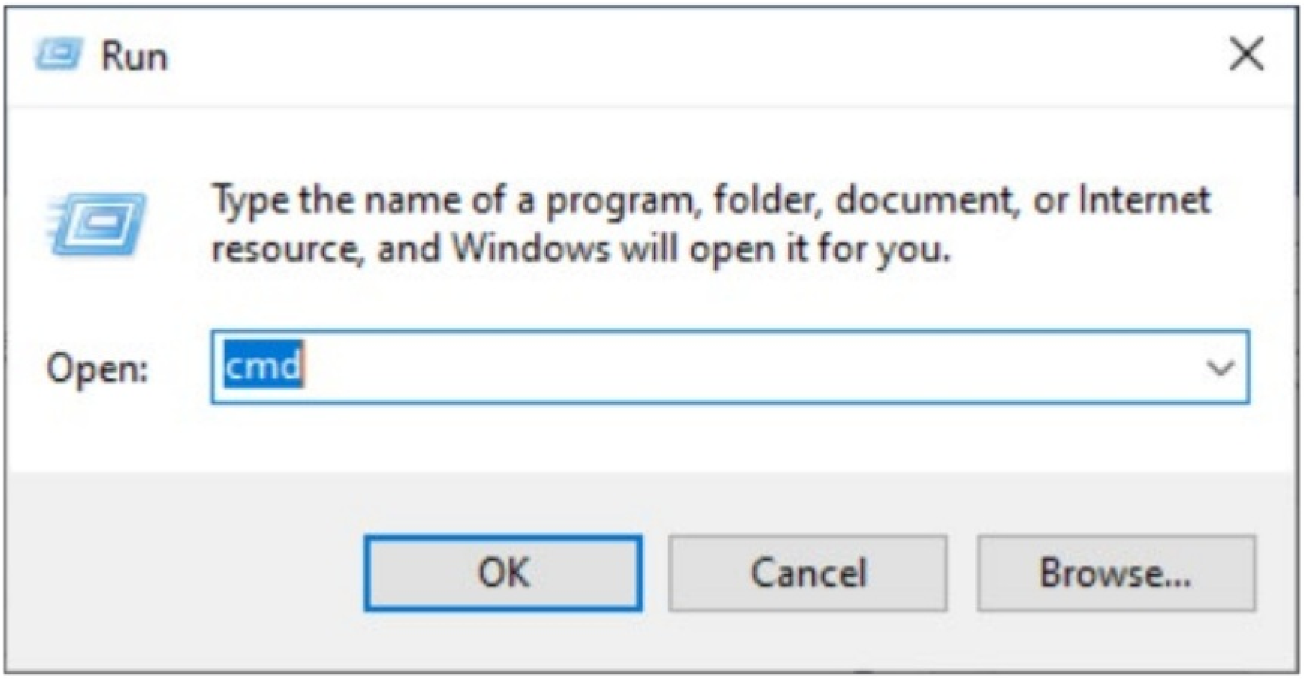
Note: Reboot the computer when the installation has finished.

Test

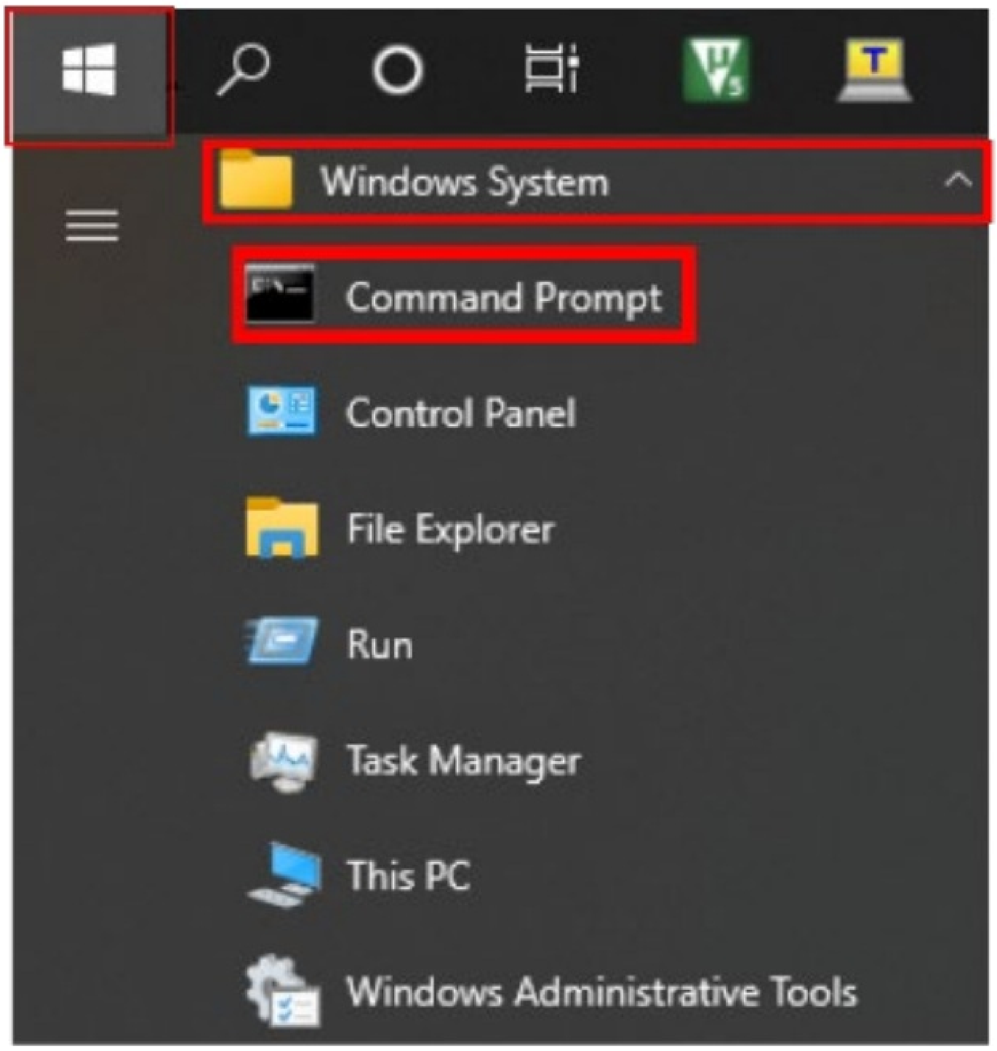
The GNU Arm Compiler adds a path to the environment variable during installation, as shown in Step 3 in the “GNU Arm Installation” section. This section will explain how to use the “Command Prompt” to test whether the GNU Arm installation has finished.

Step 1. Enable the Command Prompt.
There are several methods for enabling the command prompt, which will be explained in the “Run” Window and “Menu” selection in the following section.

- Enable via the “Run” Window: First press “Windows + R” keys on the keyboard and enter “cmd” in the pop-up “Run” window as shown in the following figure. Then press “OK” to enable the Command Prompt.

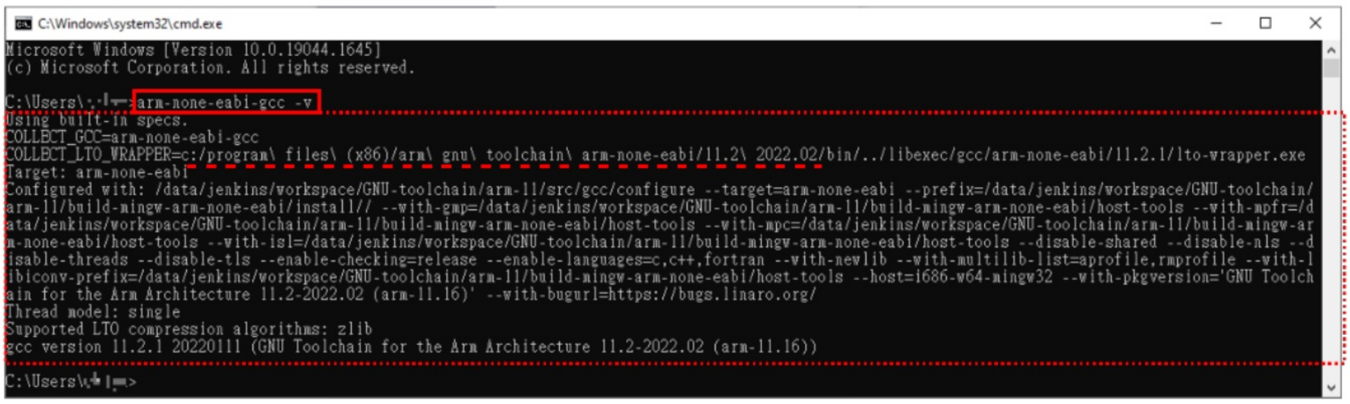

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
cmd
OK
Cancel
Browse...

- Select from the “Menu”: Click the “Start” menu, then find and open the Windows System folder. Click on “Command Prompt”, as shown in the following figure.



Step 2. Enter “arm-none-eabi-gcc -v” on the enabled Command Prompt and the following screen will appear, indicating that the command is valid. This means that the GNU Arm installation has finished and that the Arm program code can be compiled. At the same time, the installation path can be confirmed using the Command

Prompt output. This is shown by the path marked by the dotted line in the red dotted box in the figure.



```
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>arm-none-eabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=c:/program\ files\ (x86)/arm\ gnu\ toolchain\ arm-none-eabi/11.2\ 2022.02/bin/../libexec/gcc/arm-none-eabi/11.2.1/lto-wrapper.exe
Target: arm-none-eabi
Configured with: /data/jenkins/workspace/GNU-toolchain/arm-11/src/gcc/configure --target=arm-none-eabi --prefix=/data/jenkins/workspace/GNU-toolchain/
arm-11/build-mingw-arm-none-eabi/install// --with-gmp=/data/jenkins/workspace/GNU-toolchain/arm-11/build-mingw-arm-none-eabi/host-tools --with-mpfr=/d
ata/jenkins/workspace/GNU-toolchain/arm-11/build-mingw-arm-none-eabi/host-tools --with-mpc=/data/jenkins/workspace/GNU-toolchain/arm-11/build-mingw-ar
m-none-eabi/host-tools --with-isl=/data/jenkins/workspace/GNU-toolchain/arm-11/build-mingw-arm-none-eabi/host-tools --disable-shared --disable-nls --d
isable-threads --disable-tls --enable-checking=release --enable-languages=c,c++,fortran --with-newlib --with-multilib-list=aprofile,rmprofile --with-l
ibiconv-prefix=/data/jenkins/workspace/GNU-toolchain/arm-11/build-mingw-arm-none-eabi/host-tools --host=i686-w64-mingw32 --with-pkgversion='GNU Toolch
ain for the Arm Architecture 11.2-2022.02 (arm-11.16)' --with-bugurl=https://bugs.linaro.org/
Thread model: single
Supported LTO compression algorithms: zlib
gcc version 11.2.1 20220111 (GNU Toolchain for the Arm Architecture 11.2-2022.02 (arm-11.16))

C:\Users\[redacted]>
```

Used with GNU Make

This chapter describes how to use the GNU Arm Compiler with GNU Make.

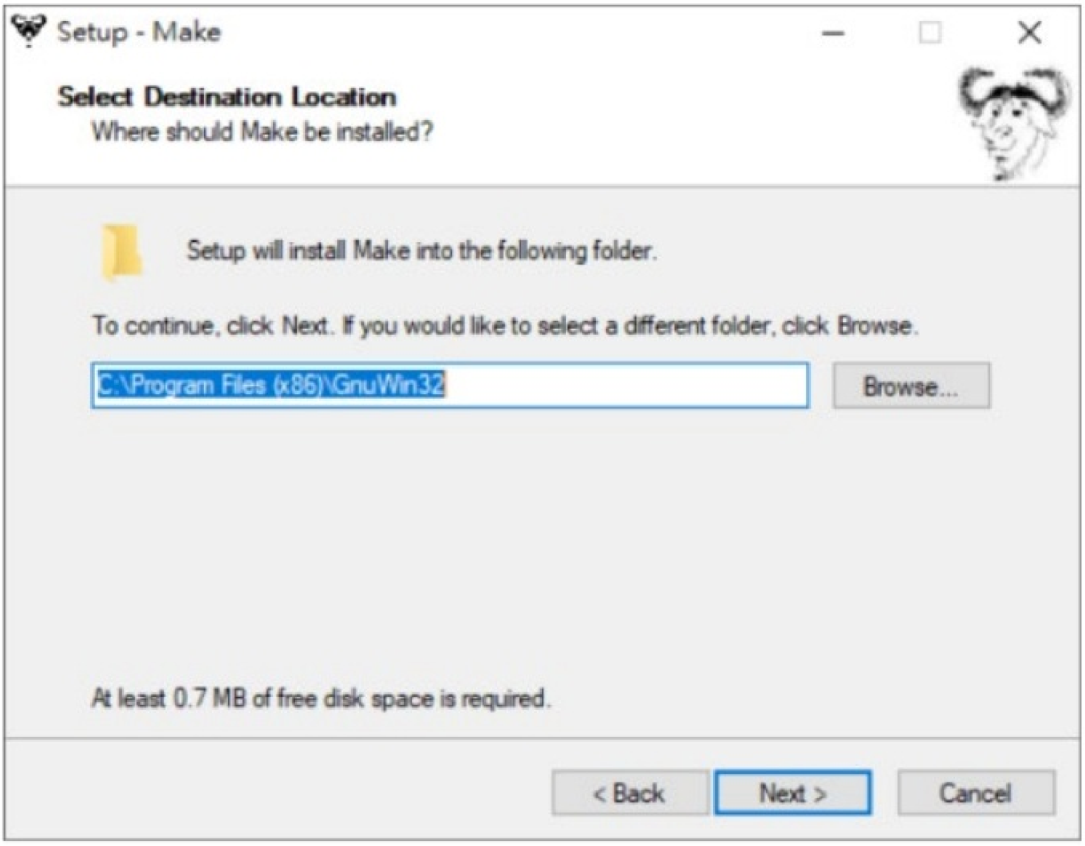
GNU Make Installation

Step 1. Click the following link to download the EXE file for GNU Make installation.

http://gnuwin32.sourceforge.net/packages/make.htm

Note: The file name is similar to “make-3.81.exe

Step 2. In the “Select Destination Location” step, as shown in the following figure, copy and save the installation path during installation. The path will be configured into an environment variable later. For example: “C:\Program Files (x86)\GnuWin32”



Select Destination Location
Where should Make be installed?

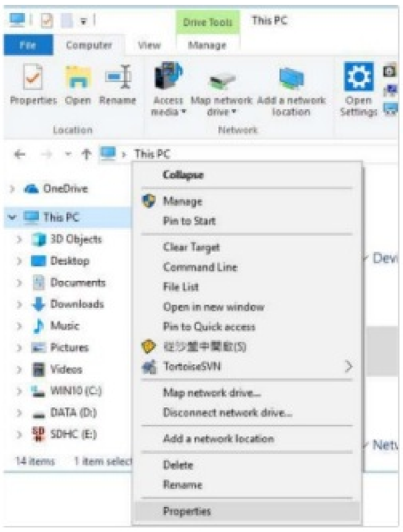
Setup will install Make into the following folder.

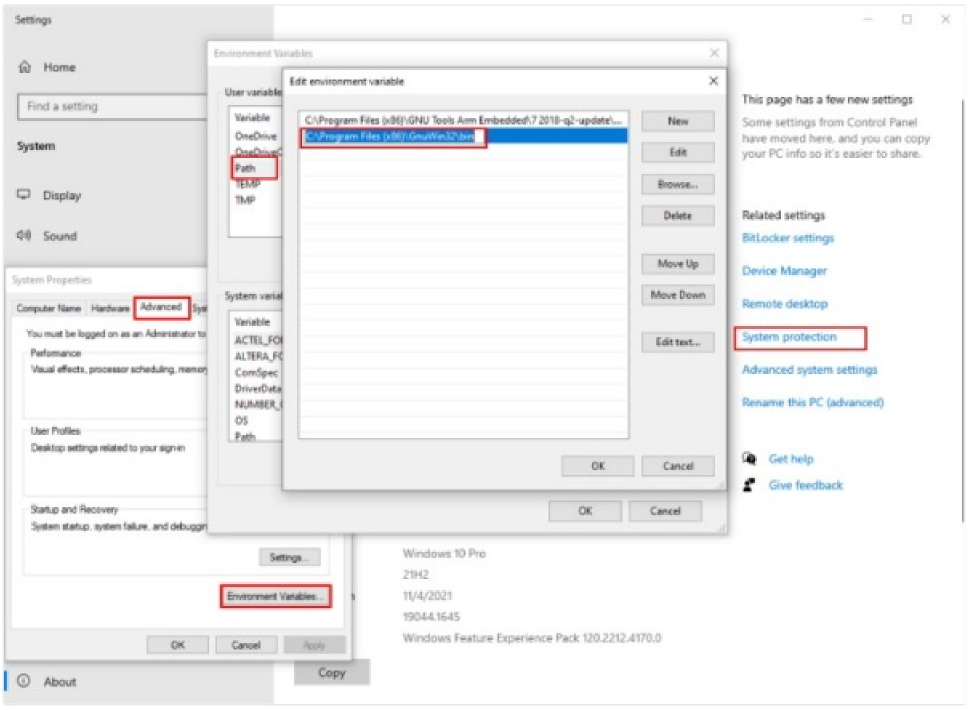
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files (x86)\GnuWin32

At least 0.7 MB of free disk space is required.

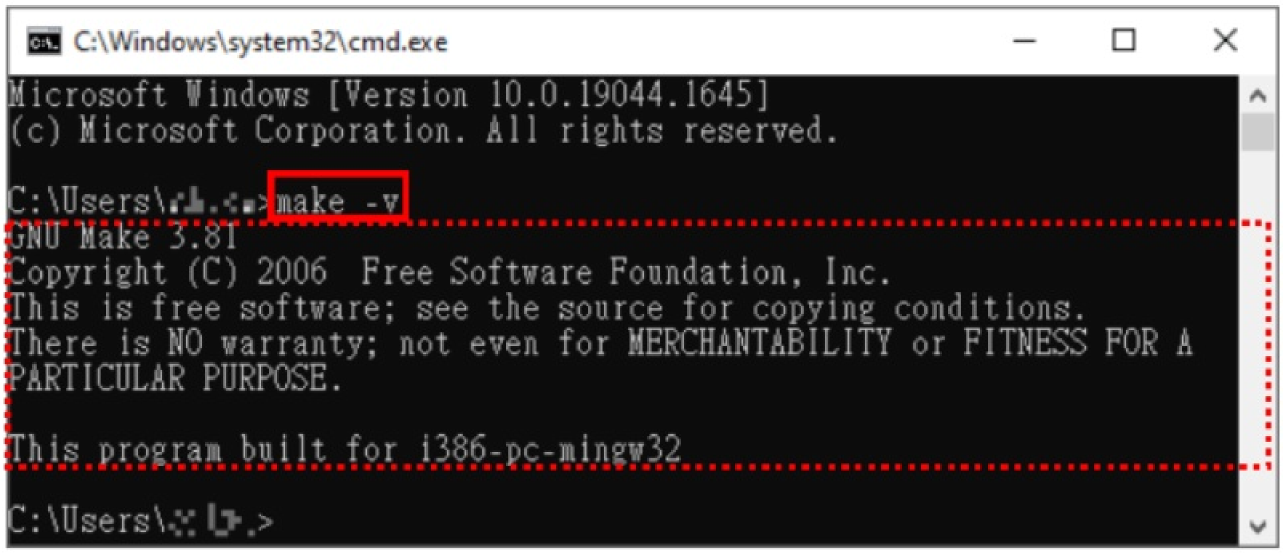
Step 3. Add an extra character “\bin” to the copied path in Step 2 and configure it in the environment variable “Path” to add a path for the GNU Make tool. Refer to the two following figures, which show how to enable the environment variable, and find the “Path” edit and add a path.
Note: The entire path is similar to “C:\Program Files (x86)\GnuWin32\bin”.





C:\Program Files (x86)\GNU Tools Arm Embedded\7 2018-q2-update\...
C:\Program Files (x86)\GnuWin32\bin

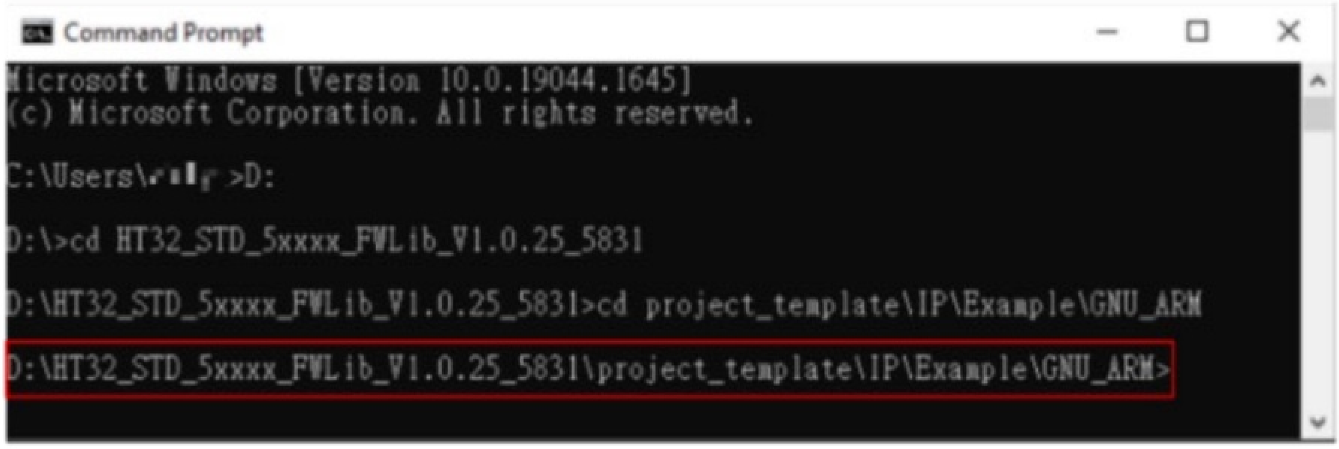
Step 4. Test the “make-v” command using the Command Prompt and the following screen will appear, indicating that the command is valid. This means that the GNU Make installation has finished.
Note: Refer to the “Enable Command Prompt” contents in the “Test” section for the Command Prompt enabling method.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>make -v
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i386-pc-mingw32

C:\Users\[redacted]>
```

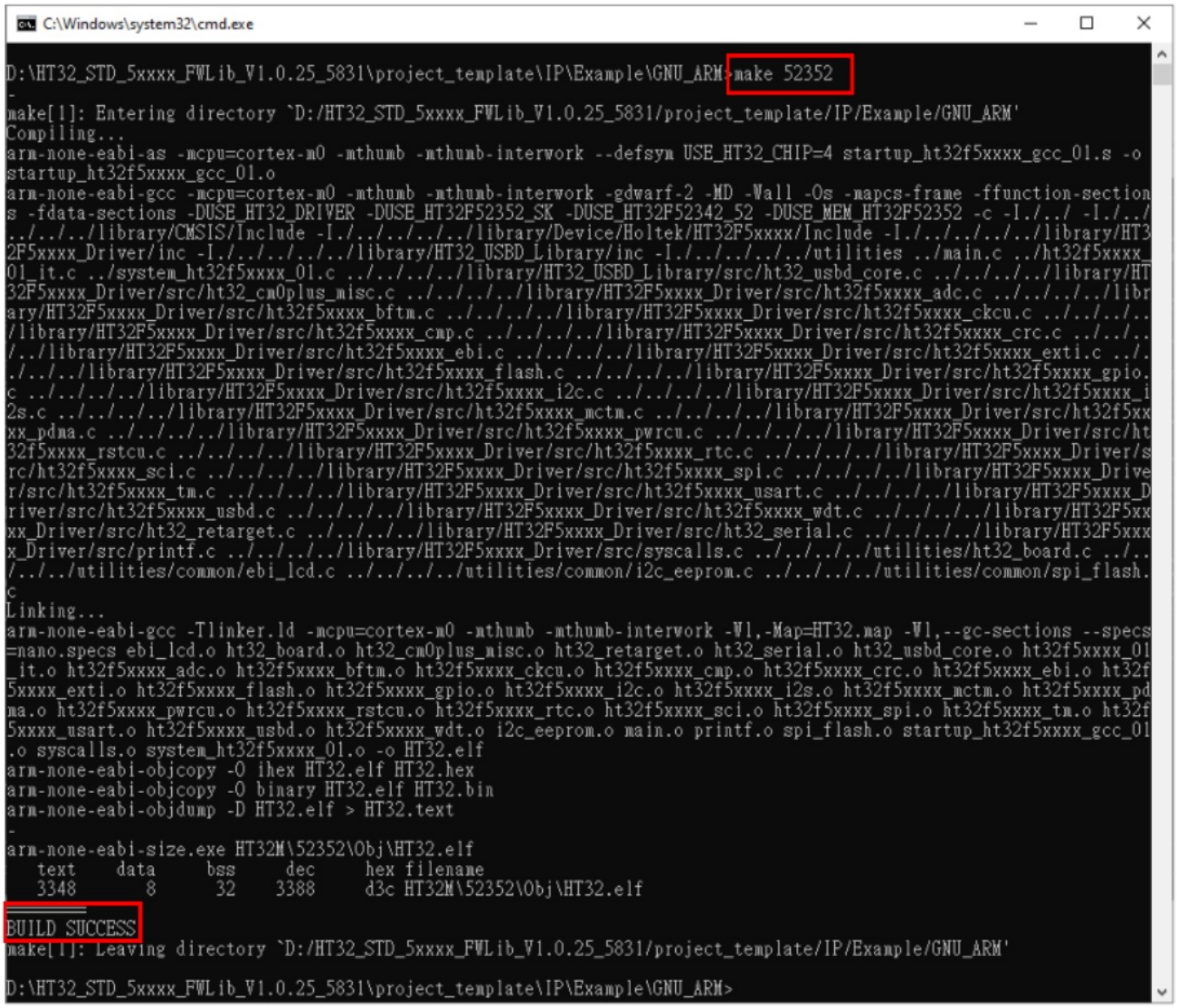
Compile and Output

This section will describe the compilation method and output results using a Firmware Library Project Template (…\project template\IP\Example), which includes compilation commands, output messages and output files etc.
Step 1. Enable the Command Prompt and change the operating directory to the “GNU_ARM” folder in the Firmware Library Project Template.
(\\HT32_STD_5xxxx_FWLib_Vm.n.r_s\project_template\IP\Example\GNU_ARM)



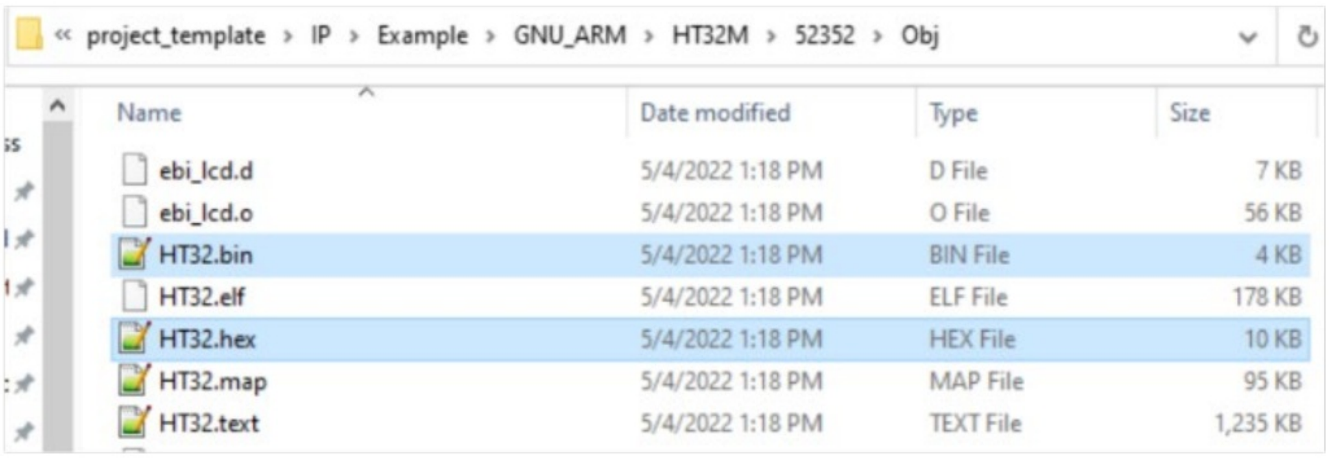
```
Command Prompt
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>D:

D:\>cd HT32_STD_5xxxx_FWLib_V1.0.25_5831

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831>cd project_template\IP\Example\GNU_ARM

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>
```

Step 2. Enter a “make xxxxx” or “make –f xxxxx.mk” command to build a program. When all build operations have finished, a “BUILD SUCCESSFUL” message should appear as shown in the following figure.
(xxxxx is the IC device, here a 52352 is used)


C:\Windows\system32\cmd.exe

```
D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>make 52352
-
make[1]: Entering directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
Compiling...
arm-none-eabi-as -mcpu=cortex-m0 -mthumb -mthumb-interwork --defsym USE_HT32_CHIP=4 startup_ht32f5xxxx_gcc_01.s -o
startup_ht32f5xxxx_gcc_01.o
arm-none-eabi-gcc -mcpu=cortex-m0 -mthumb -mthumb-interwork -gdwarf-2 -MD -Wall -Os -mapcs-frame -ffunction-section
s -fdata-sections -DUSE_HT32_DRIVER -DUSE_HT32F52352_SK -DUSE_HT32F52342_52 -DUSE_MEM_HT32F52352 -c -I./../ -I./../
../../../library/CMSIS/Include -I./../../../../library/Device/Holtek/HT32F5xxxx/Include -I./../../../../library/HT3
2F5xxxx_Driver/inc -I./../../../../library/HT32_USBD_Library/inc -I./../../../../utilities ../main.c ../ht32f5xxxx_
01_it.c ../system_ht32f5xxxx_01.c ../../../../library/HT32_USBD_Library/src/ht32_usbd_core.c ../../../../library/HT
32F5xxxx_Driver/src/ht32_cm0plus_misc.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_adc.c ../../../../libr
ary/HT32F5xxxx_Driver/src/ht32f5xxxx_bftm.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_ckcu.c ../../../..
/library/HT32F5xxxx_Driver/src/ht32f5xxxx_cmp.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_crc.c ../../..
/../library/HT32F5xxxx_Driver/src/ht32f5xxxx_ebi.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_exti.c ../.
./../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_flash.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_gpio.
c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_i2c.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_i
2s.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_mctm.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xx
xx_pdma.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_pwrcu.c ../../../../library/HT32F5xxxx_Driver/src/ht
32f5xxxx_rstcu.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_rtc.c ../../../../library/HT32F5xxxx_Driver/s
rc/ht32f5xxxx_sci.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_spi.c ../../../../library/HT32F5xxxx_Drive
r/src/ht32f5xxxx_tm.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_usart.c ../../../../library/HT32F5xxxx_D
river/src/ht32f5xxxx_usbd.c ../../../../library/HT32F5xxxx_Driver/src/ht32f5xxxx_wdt.c ../../../../library/HT32F5xx
xx_Driver/src/ht32_retarget.c ../../../../library/HT32F5xxxx_Driver/src/ht32_serial.c ../../../../library/HT32F5xxx
x_Driver/src/printf.c ../../../../library/HT32F5xxxx_Driver/src/syscalls.c ../../../../utilities/ht32_board.c ../..
/../../utilities/common/ebi_lcd.c ../../../../utilities/common/i2c_eeprom.c ../../../../utilities/common/spi_flash.
c
Linking...
arm-none-eabi-gcc -Tlinker.ld -mcpu=cortex-m0 -mthumb -mthumb-interwork -Wl,-Map=HT32.map -Wl,--gc-sections --specs
=nano.specs ebi_lcd.o ht32_board.o ht32_cm0plus_misc.o ht32_retarget.o ht32_serial.o ht32_usbd_core.o ht32f5xxxx_01
_it.o ht32f5xxxx_adc.o ht32f5xxxx_bftm.o ht32f5xxxx_ckcu.o ht32f5xxxx_cmp.o ht32f5xxxx_crc.o ht32f5xxxx_ebi.o ht32f
5xxxx_exti.o ht32f5xxxx_flash.o ht32f5xxxx_gpio.o ht32f5xxxx_i2c.o ht32f5xxxx_i2s.o ht32f5xxxx_mctm.o ht32f5xxxx_pd
ma.o ht32f5xxxx_pwrcu.o ht32f5xxxx_rstcu.o ht32f5xxxx_rtc.o ht32f5xxxx_sci.o ht32f5xxxx_spi.o ht32f5xxxx_tm.o ht32f
5xxxx_usart.o ht32f5xxxx_usbd.o ht32f5xxxx_wdt.o i2c_eeprom.o main.o printf.o spi_flash.o startup_ht32f5xxxx_gcc_01
.o syscalls.o system_ht32f5xxxx_01.o -o HT32.elf
arm-none-eabi-objcopy -O ihex HT32.elf HT32.hex
arm-none-eabi-objcopy -O binary HT32.elf HT32.bin
arm-none-eabi-objdump -D HT32.elf > HT32.text
-
arm-none-eabi-size.exe HT32M\52352\Obj\HT32.elf
   text    data     bss     dec     hex filename
   3348       8      32    3388     d3c HT32M\52352\Obj\HT32.elf
========
BUILD SUCCESS
make[1]: Leaving directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>
```

Step 3. After completing Step 2, the Hex and Binary files can be found in the following path. Then update the firmware to the Starter Kit using other programming tools such as e-Writer32, HT32 ICP Tool, e-Link32 Pro/Lite, HT32 Flash Programmer and ISP bootloader. The next section will introduce programming using the e-Link32 Pro / Lite.
"…\GNU_ARM\HT32M\xxxxx\Obj\HT32.bin"
"…\GNU_ARM\HT32M\xxxxx\Obj\HT32.hex"

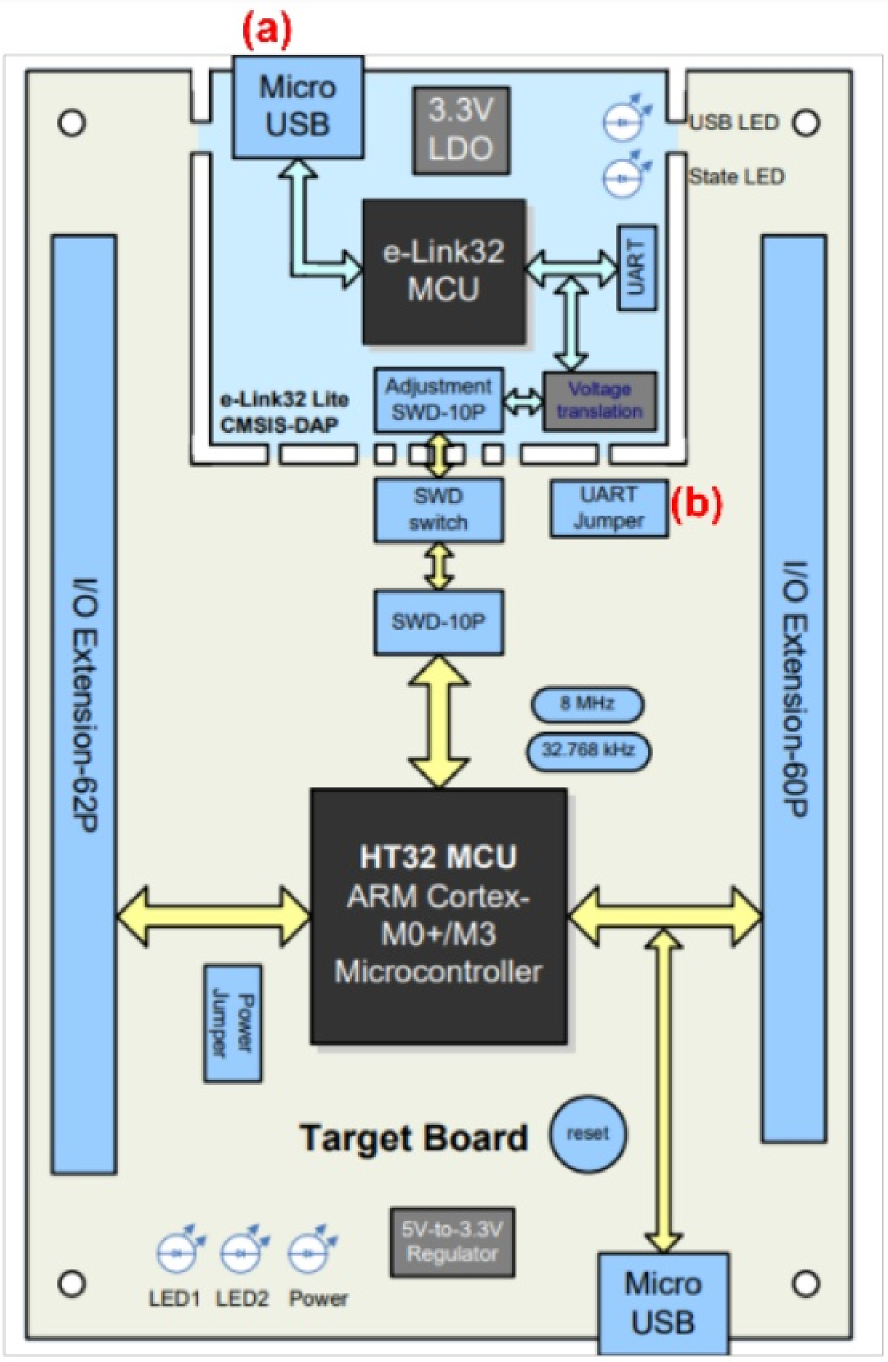

« project_template › IP › Example › GNU_ARM › HT32M › 52352 › Obj

Name	Date modified	Type	Size
ebi_lcd.d	5/4/2022 1:18 PM	D File	7 KB
ebi_lcd.o	5/4/2022 1:18 PM	O File	56 KB
HT32.bin	5/4/2022 1:18 PM	BIN File	4 KB
HT32.elf	5/4/2022 1:18 PM	ELF File	178 KB
HT32.hex	5/4/2022 1:18 PM	HEX File	10 KB
HT32.map	5/4/2022 1:18 PM	MAP File	95 KB
HT32.text	5/4/2022 1:18 PM	TEXT File	1,235 KB

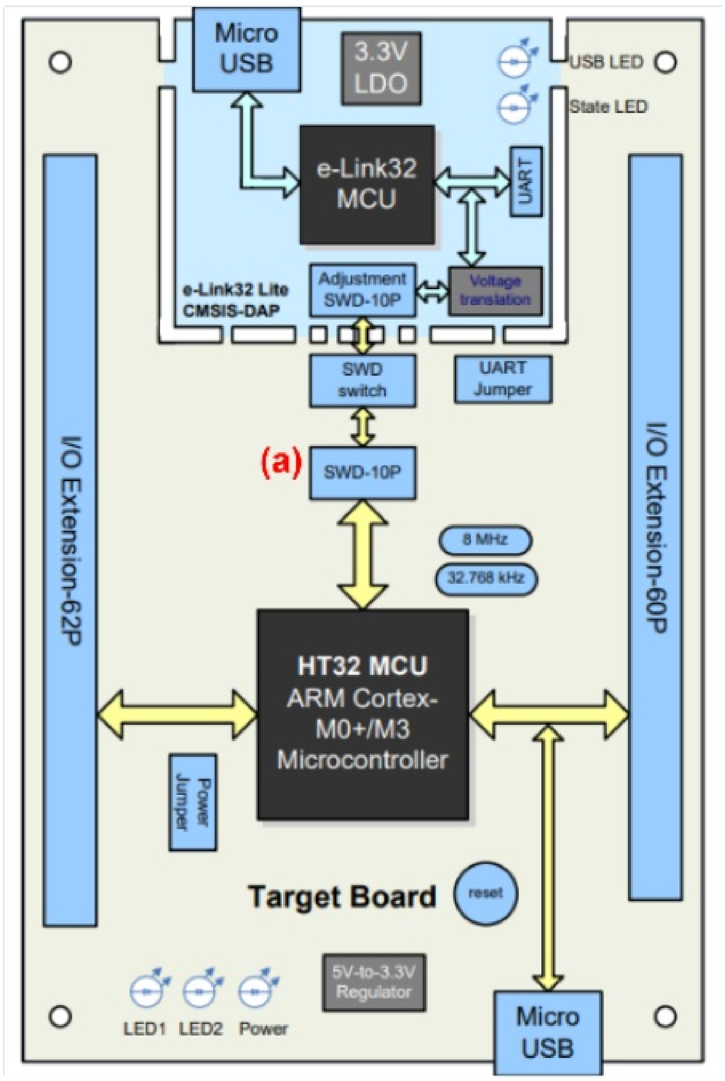
Programming using the e-Link32 Pro/Lite

This section will take the HT32F52352 Starter Kit (SK) as an example. First, it introduces the environment

preparation operations for the Starter Kit (SK) and e-Link32 Pro / Lite, and then explains how to use “make IC=xxxxx eraseall/program/run” and Command Prompt resultsin sequence. Finally it explains how to observe whether the programming is successful or not through the SK status.
The environment preparation operations for the SK and e-Link32 Lite are as follows:
(1) There are two USB COM ports on the board. Here the PC is connected to the e-Link32 Lite port on the board using a USB cable, as shown by (a) in the following figure.
(2) The VCP (Virtual COM Port) function of the e-Link32 Lite is required for programming confirmation. Ensure that the UART Jumper-J2*1 jumper cap shorts the PAx*2 and DAP_Tx pins. The jumper location is shown by (b) in the following figure.
Note: 1. J2 on the SK provides two settings which are to short the PAx and DAP_Tx pins or short the PAx and RS232_Tx pins. Refer to the Starter Kit User Manual for details.
2. The pin is named PAx here because the setting for the MCU UART RX pin varies in different SKs.

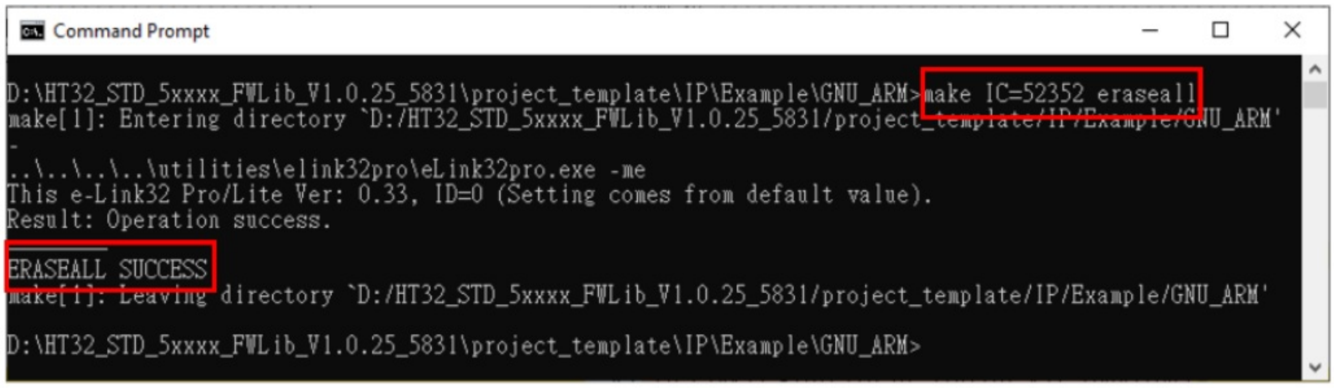


The environment preparation operations for the SK and e-Link32 Pro is as follows: One side of the e-Link32 Pro is connected to the PC using a Mini USB data cable and the other side is for the SWD interface. The e-Link32 Pro is required to connect to the SWD-10P on the SK using a 10-pin gray flat cable, as shown in the following figure (a).



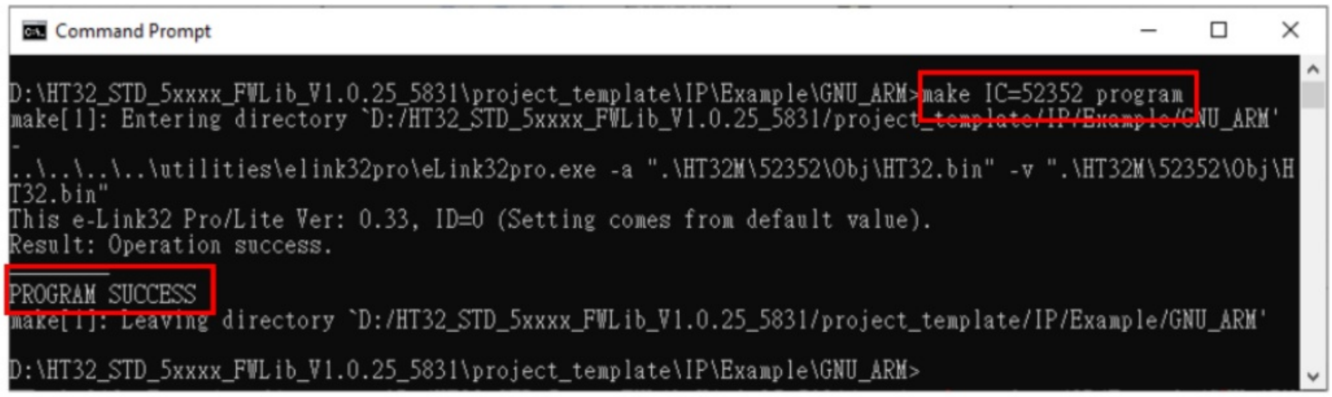
The following section will describe the "make IC=xxxxx eraseall/program/run" command usage and the Command Prompt results in sequence.

Step 1. Enter a "make IC=xxxxx eraseall" or "make-f xxxxx.mk eraseall" command in the "Command Prompt" window. If successful, the "ERASEALL SUCCESS" message will appear on the screen, as shown in the following figure.
(xxxxx is the IC device, here a 52352 is used).
Note: This command is used to execute a Flash Mass Erase operation.



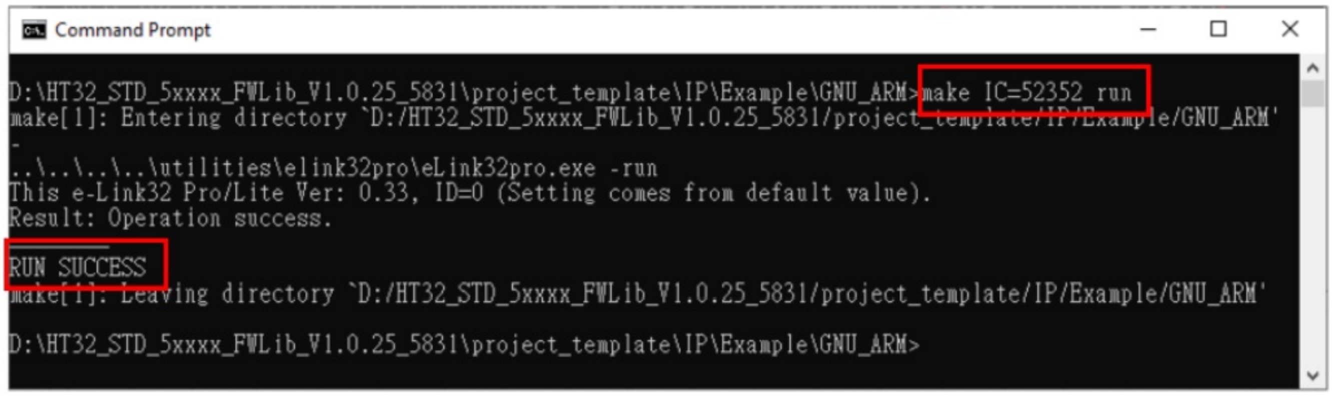
```
D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>make IC=52352 eraseall
make[1]: Entering directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
-
..\..\..\..\utilities\elink32pro\eLink32pro.exe -me
This e-Link32 Pro/Lite Ver: 0.33, ID=0 (Setting comes from default value).
Result: Operation success.

ERASEALL SUCCESS
make[1]: Leaving directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>
```

Step 2. Enter a "make IC=xxxxx program" or "make-f xxxxx.mk program" command in the "Command Prompt" window. If successful, the "PROGRAM SUCCESS" message will appear on the screen, as shown in the following figure.
(xxxxx is the IC device, here a 52352 is used).



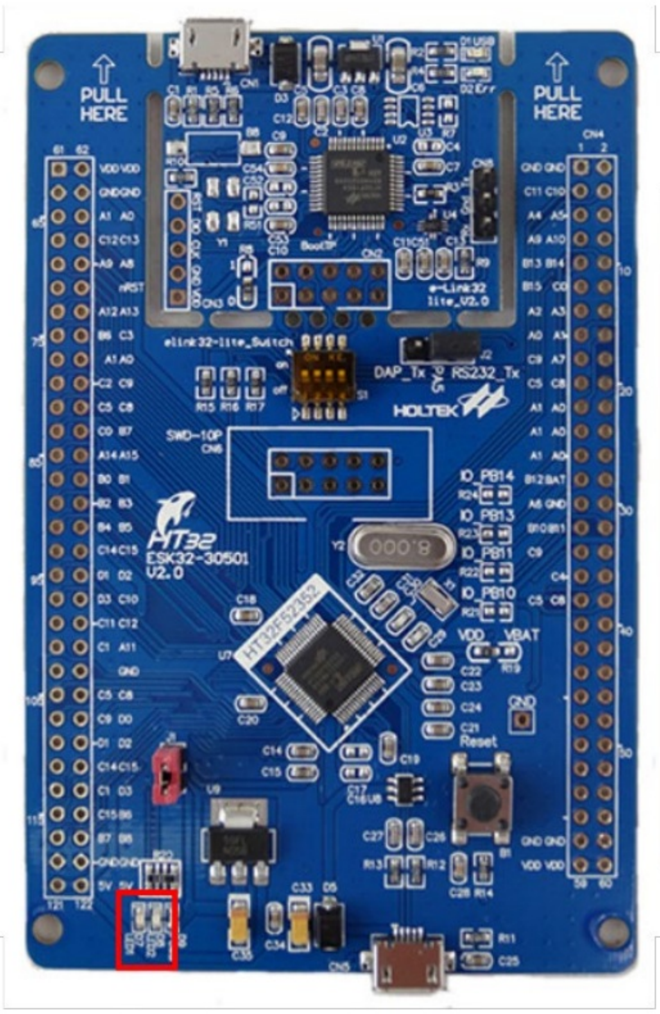
```
D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>make IC=52352 program
make[1]: Entering directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
-
..\..\..\..\utilities\elink32pro\eLink32pro.exe -a ".\HT32M\52352\Obj\HT32.bin" -v ".\HT32M\52352\Obj\H
T32.bin"
This e-Link32 Pro/Lite Ver: 0.33, ID=0 (Setting comes from default value).
Result: Operation success.

PROGRAM SUCCESS
make[1]: Leaving directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>
```

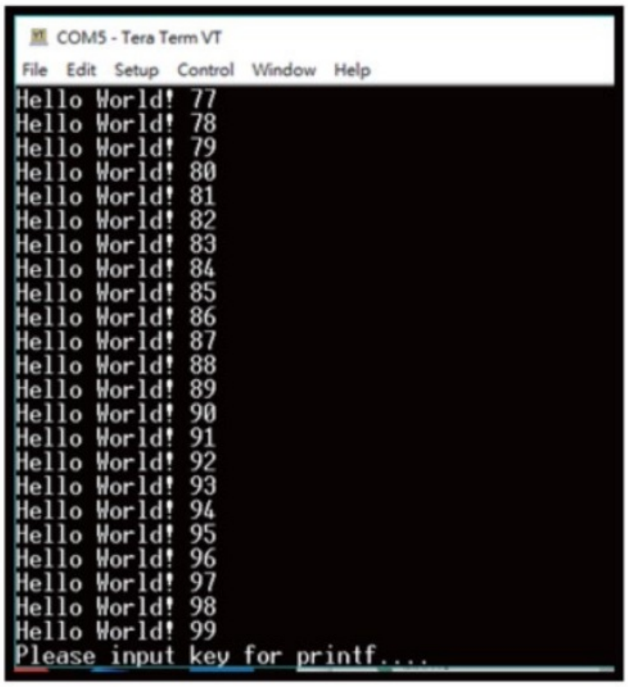
Step 3. Enter a “make IC=xxxxx run” or “make-f xxxxx.mk run” command in the “Command Prompt” window. If successful, the “RUN SUCCESS” message will appear on the screen, as shown in the following figure. The SK will operate according to the example program and its status when programmed successfully is shown in Step 4. (xxxxx is the IC device, here a 52352 is used)



```
D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>make IC=52352 run
make[1]: Entering directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
-
..\..\..\..\utilities\elink32pro\eLink32pro.exe -run
This e-Link32 Pro/Lite Ver: 0.33, ID=0 (Setting comes from default value).
Result: Operation success.

RUN SUCCESS
make[1]: Leaving directory `D:/HT32_STD_5xxxx_FWLib_V1.0.25_5831/project_template/IP/Example/GNU_ARM'

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM>
```

Step 4. When the Step 3 action has finished, this step will be continued to determine whether the programming has been successful by checking the SK status. This can be verified using the LED or terminal software. Refer to the “Terminal Software” section for the terminal software settings. The status description will be given below. When the “RUN SUCCESS” message appears on the screen, both LED1 and LED2 will blink. Their positions are shown at the bottom left of the following figure. The following message will then be displayed “Hello World! 0” ~ “Hello World! 99” on the PC’s terminal software via the Virtual COM Port, as shown on the right side of the following figure. Both can be used to verify that the environment has been successfully used.





```
Hello World! 77
Hello World! 78
Hello World! 79
Hello World! 80
Hello World! 81
Hello World! 82
Hello World! 83
Hello World! 84
Hello World! 85
Hello World! 86
Hello World! 87
Hello World! 88
Hello World! 89
Hello World! 90
Hello World! 91
Hello World! 92
Hello World! 93
Hello World! 94
Hello World! 95
Hello World! 96
Hello World! 97
Hello World! 98
Hello World! 99
Please input key for printf....
```

Setting Description

This section describes the related files purpose in the GNU_ARM directory, as shown in the following table.

Folder/File Name	**Description**
\\project_template\IP\Example\GNU_ARM	
xxxxx.mk	Makefile file, xxxxx is IC device
linker.ld	Linker Script
Makefile	Makefile file
Project_xxxxx.uvprojx	Project, xxxxx is IC device

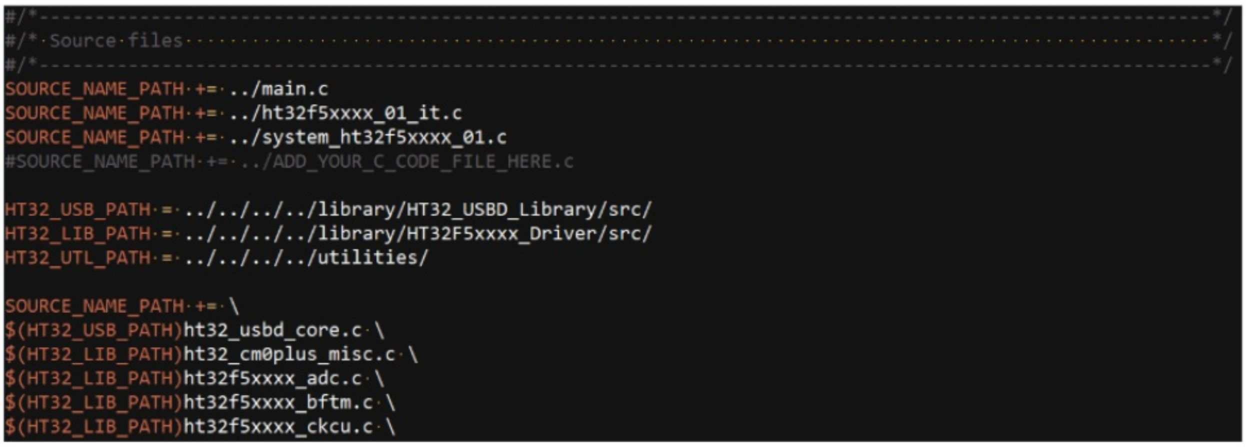
It will now be explained how to add a .c file, include path or C/S Preprocessor Define by modifying the makefile file named “xxxxx.mk”.

Note: This section uses 52352.mk as an illustration.

- Add a .c file. This part is used to set the project .c file, the following method is used.
    - Open the 52352.mk, search for “Source files”, the settings shown below appear on the screen, which can be added using “SOURCE_NAME_PATH +=” append “.c file path and name”.
- Include Path. This part is used to add Include Paths, which provide multiple paths to search for the header file (.h file), the following method is used.
    - Open the 52352.mk, search for “Include Path”, the settings shown below appear on the screen, which can be added using “INCLUDE_PATH += -I./ ” append “Path”.
- C/S Preprocessor Define. This part is used to add a Preprocessor Define message, the following method is used.
    - Open the 52352.mk, search for “Preprocessor Define”, the settings shown below appear on the screen. The adding methods for a .c Preprocessor Define is slightly different from a .s Preprocessor Define. This is arranged as follows.

      ► c Preprocessor Define: “C_Option += -D” + “Define content” For example: C_OPTION += -DUSE_HT32_DRIVER.

      ► .h Preprocessor Define: “S_Option = –defsym” + “Define content” For example: S_OPTION = –defsym USE_HT32_CHIP=4



```
#/*-----------------------------------------------------------------------------------------*/
#/* Source files                                                                            */
#/*-----------------------------------------------------------------------------------------*/
SOURCE_NAME_PATH += ../main.c
SOURCE_NAME_PATH += ../ht32f5xxxx_01_it.c
SOURCE_NAME_PATH += ../system_ht32f5xxxx_01.c
#SOURCE_NAME_PATH += ../ADD_YOUR_C_CODE_FILE_HERE.c

HT32_USB_PATH = ../../../../library/HT32_USBD_Library/src/
HT32_LIB_PATH = ../../../../library/HT32F5xxxx_Driver/src/
HT32_UTL_PATH = ../../../../utilities/

SOURCE_NAME_PATH += \
$(HT32_USB_PATH)ht32_usbd_core.c \
$(HT32_LIB_PATH)ht32_cm0plus_misc.c \
$(HT32_LIB_PATH)ht32f5xxxx_adc.c \
$(HT32_LIB_PATH)ht32f5xxxx_bftm.c \
$(HT32_LIB_PATH)ht32f5xxxx_ckcu.c \
```

Used with the Keil MDK-ARM uVision

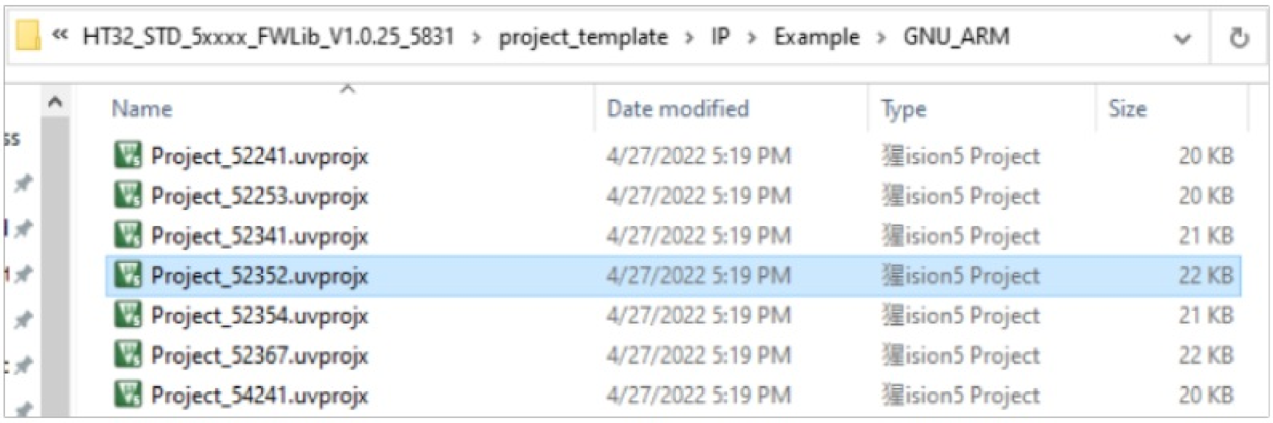
This chapter describes how to use the GNU Arm Compiler with the Keil MDK-ARM uVision. Note: This part requires the use of Keil MDK-ARM. First go to the Keil official website to obtain the EXE file for Keil MDK-ARM installation and complete the installation. The Keil official website installation link is as follows.

https://www.keil.com/demo/eval/arm.htm

Note: The file name is similar to “MDK537.EXE”.

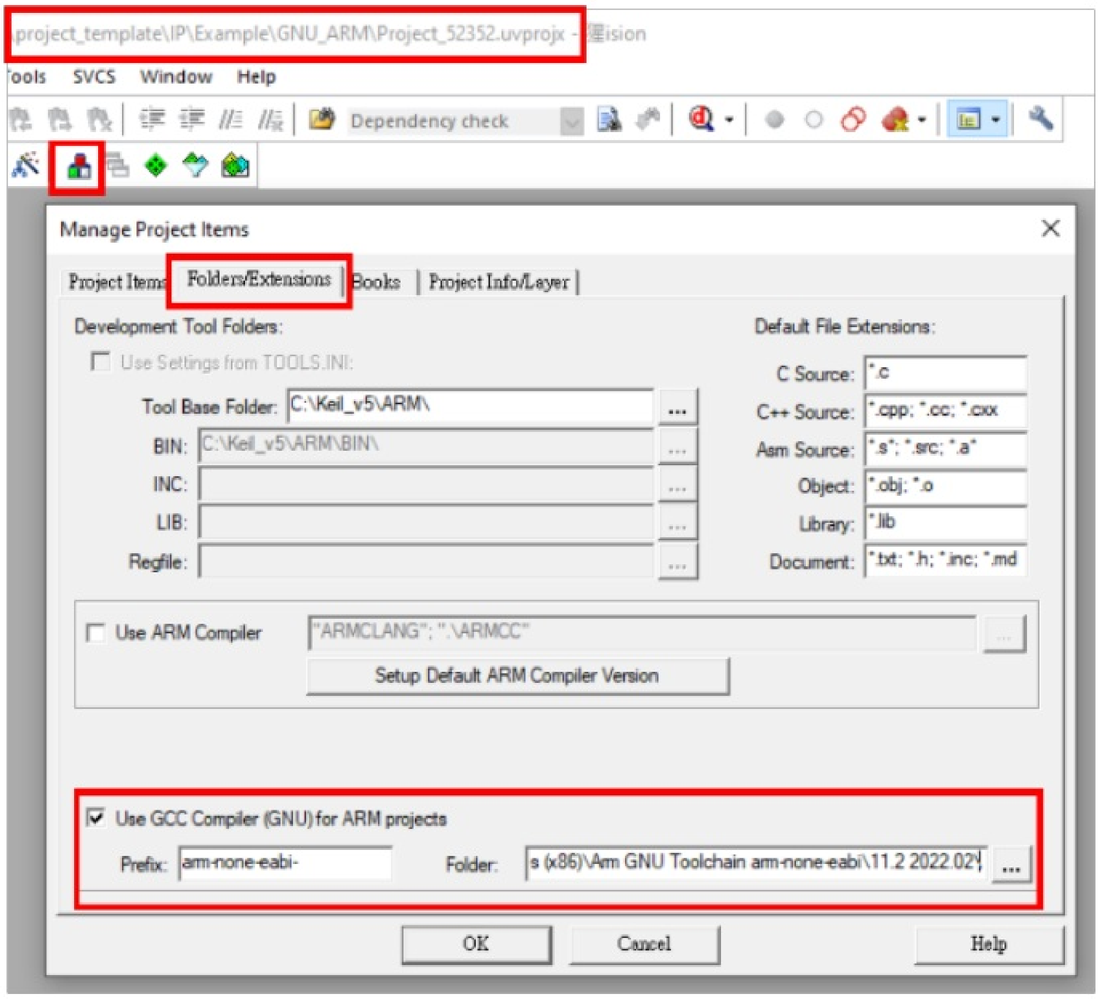
Project Settings

Step 1. Open a Project_xxxxx.uvprojx project file from the Firmware Library. Here a 52352 is used. \\HT32_STD_5xxxx_FWLib_Vm.n.r_s \project_template\IP\Example\GNU_ARM\Project_xxxxx.uvprojx Note: xxxxx is the device name.


« HT32_STD_5xxxx_FWLib_V1.0.25_5831 › project_template › IP › Example › GNU_ARM

Name	Date modified	Type	Size
Project_52241.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	20 KB
Project_52253.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	20 KB
Project_52341.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	21 KB
Project_52352.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	22 KB
Project_52354.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	21 KB
Project_52367.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	22 KB
Project_54241.uvprojx	4/27/2022 5:19 PM	礦ision5 Project	20 KB

Step 2. Click the “Manage Project Items” icon and then click the “Folders/Extensions” option. Select “Use GCC Compiler (GNU) for ARM projects” and then copy the GNU Arm installation path to the “Folder” text box, as shown in the following figure.
Note: For the GNU Arm installation path, refer to the path copied in Step 2 of the “GNU Arm Installation” section in the “GNU Arm Compiler Installation” chapter.


project_template\IP\Example\GNU_ARM\Project_52352.uvprojx - 礦ision

Manage Project Items

Development Tool Folders:

☐ Use Settings from TOOLS.INI:

Tool Base Folder: C:\Keil_v5\ARM\
BIN: C:\Keil_v5\ARM\BIN\
INC:
LIB:
Regfile:

Default File Extensions:

C Source: *.c
C++ Source: *.cpp; *.cc; *.cxx
Asm Source: *.s*; *.src; *.a*
Object: *.obj; *.o
Library: *.lib
Document: *.txt; *.h; *.inc; *.md

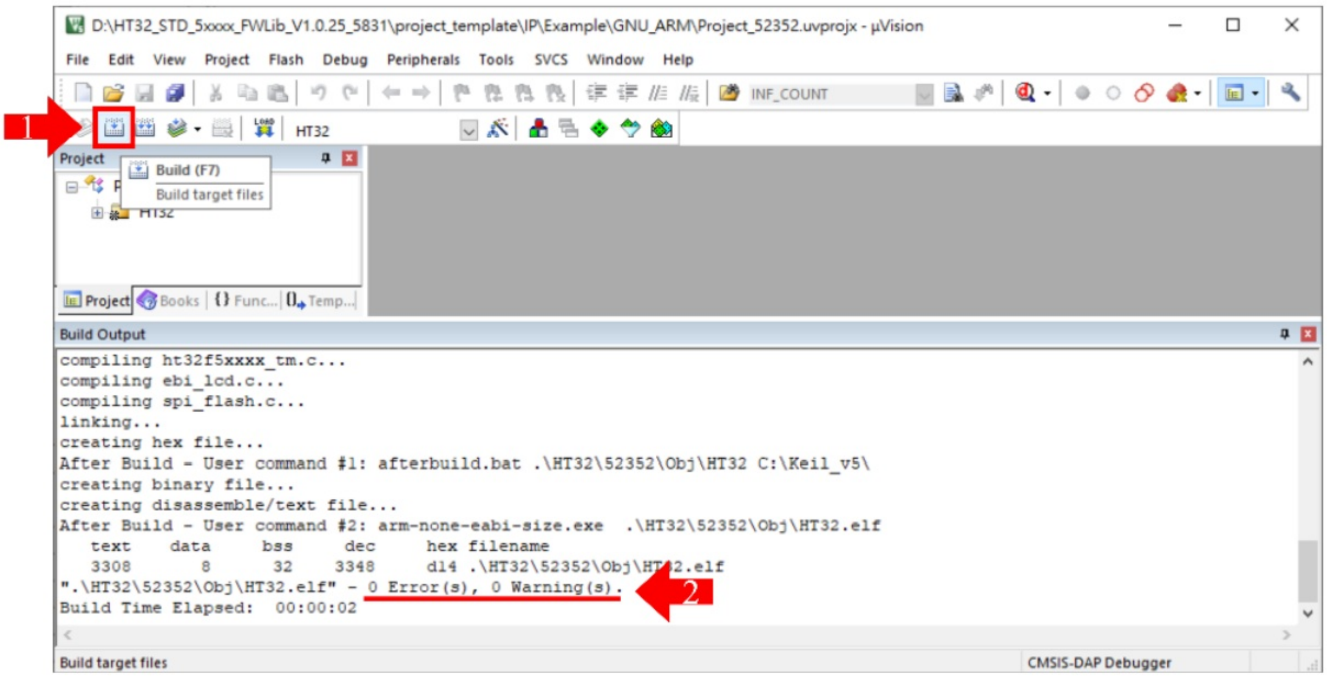
☐ Use ARM Compiler "ARMCLANG"; ".\ARMCC"

☑ Use GCC Compiler (GNU) for ARM projects

Prefix: arm-none-eabi-
Folder: s (x86)\Arm GNU Toolchain arm-none-eabi\11.2 2022.02\

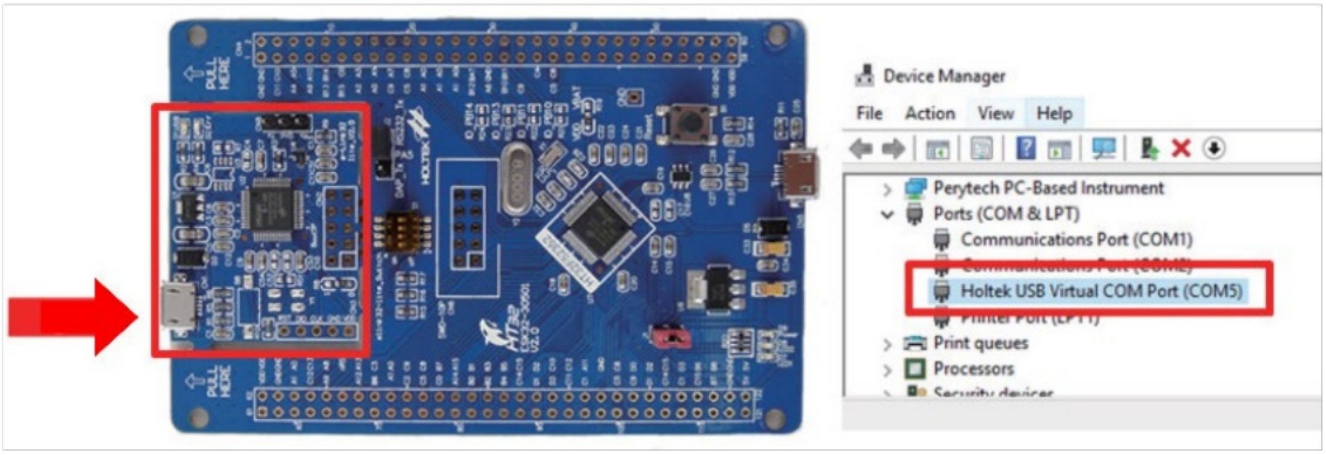
Compile and Test

Step 1. Click “Build (F7)” to build a project.
Step 2. Check “Build Output” window to confirm whether the program has been built correctly.

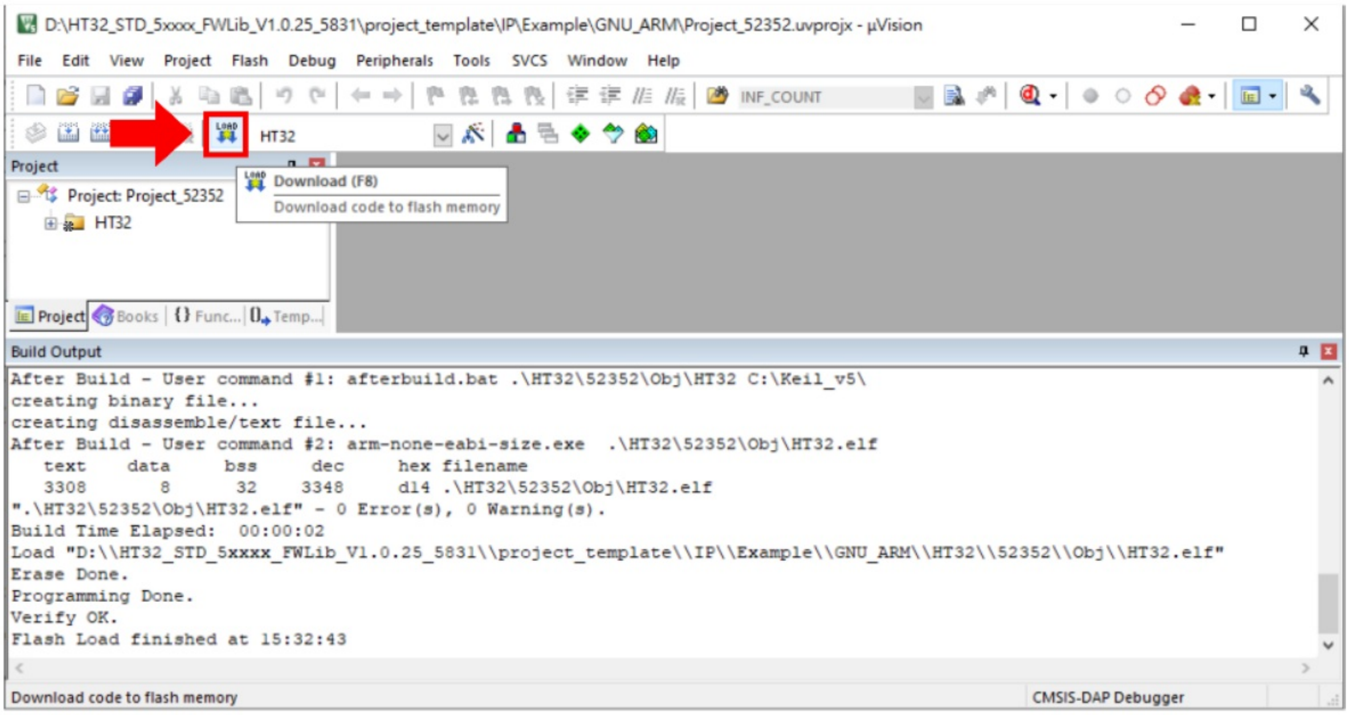

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM\Project_52352.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
1
HT32
Build (F7)
Build target files
Build Output
2
Build target files
CMSIS-DAP Debugger

```
compiling ht32f5xxxx_tm.c...
compiling ebi_lcd.c...
compiling spi_flash.c...
linking...
creating hex file...
After Build - User command #1: afterbuild.bat .\HT32\52352\Obj\HT32 C:\Keil_v5\
creating binary file...
creating disassemble/text file...
After Build - User command #2: arm-none-eabi-size.exe  .\HT32\52352\Obj\HT32.elf
   text    data     bss     dec     hex filename
   3308       8      32    3348     d14 .\HT32\52352\Obj\HT32.elf
".\HT32\52352\Obj\HT32.elf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:02
```

Step 3. Connect the e-link32 lite USB COM port on the HT32F52352 Starter Kit to the PC, as shown in the red box on the left side of the following figure. Confirm that the PC has detected the USB device normally, as shown in the red box on the right side of the following figure.

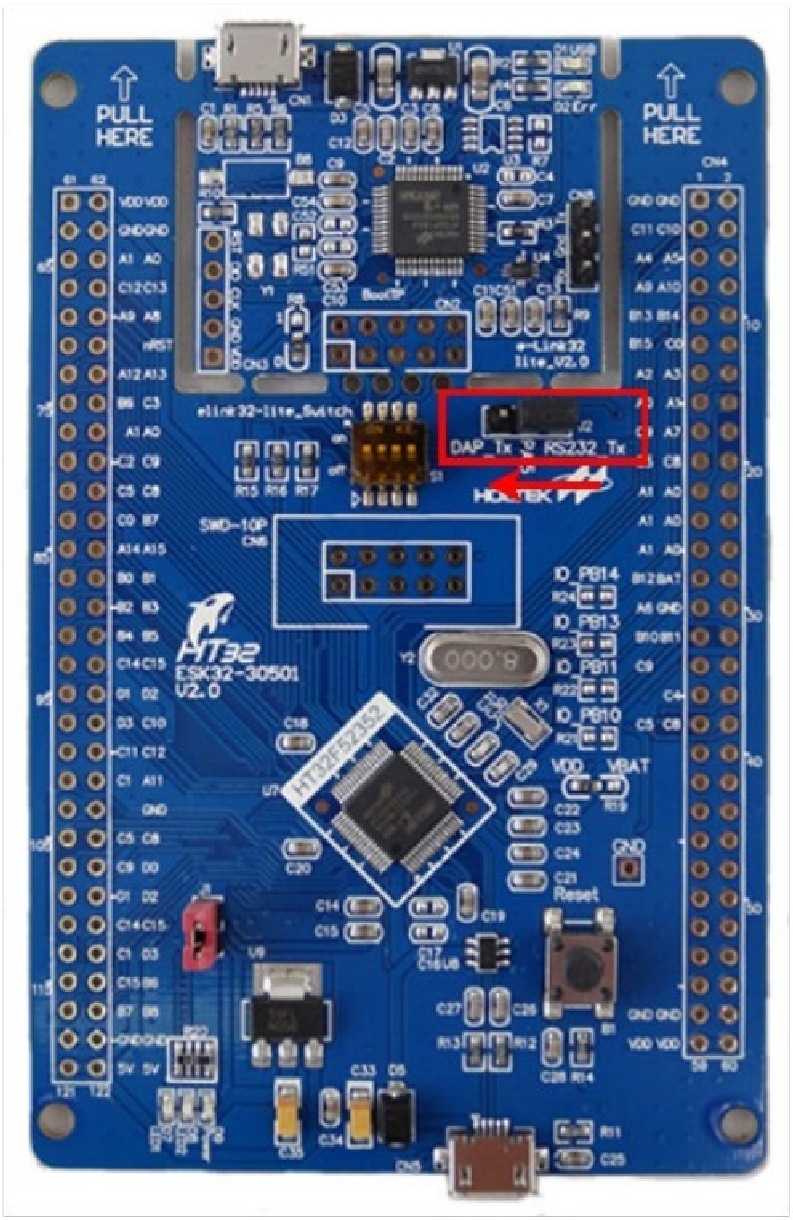

Device Manager
File Action View Help
Perytech PC-Based Instrument
Ports (COM & LPT)
Communications Port (COM1)
Holtek USB Virtual COM Port (COM5)
Print queues
Processors

Step 4. Click “Download (F8)” to download the code to the Flash memory.

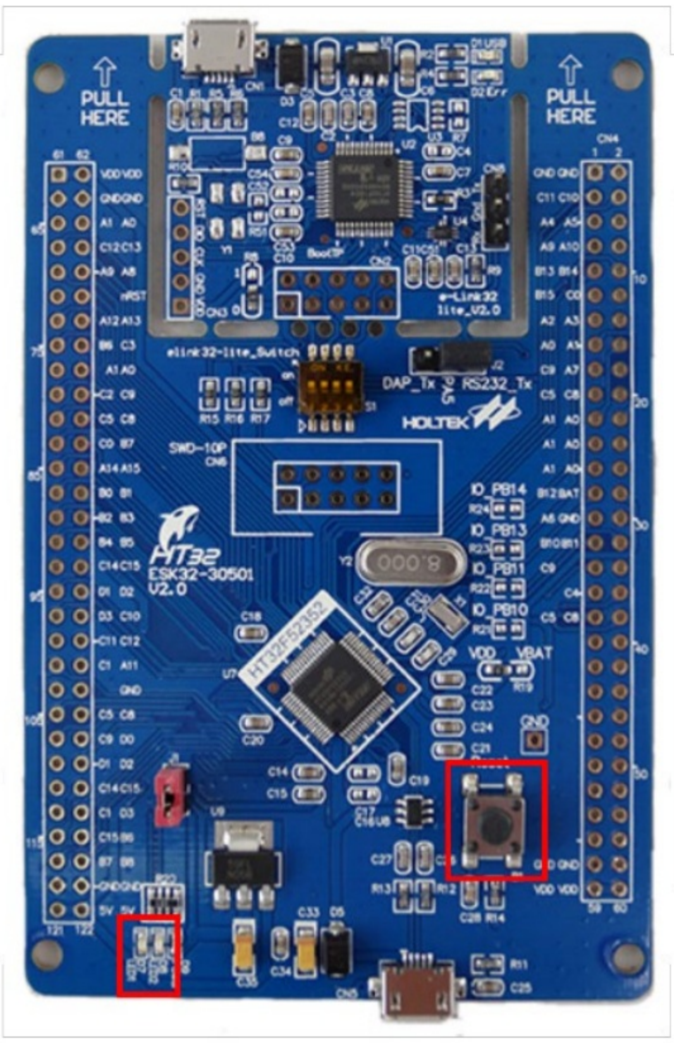

D:\HT32_STD_5xxxx_FWLib_V1.0.25_5831\project_template\IP\Example\GNU_ARM\Project_52352.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
HT32
Download (F8)
Download code to flash memory
Project: Project_52352
HT32
Build Output
Download code to flash memory
CMSIS-DAP Debugger

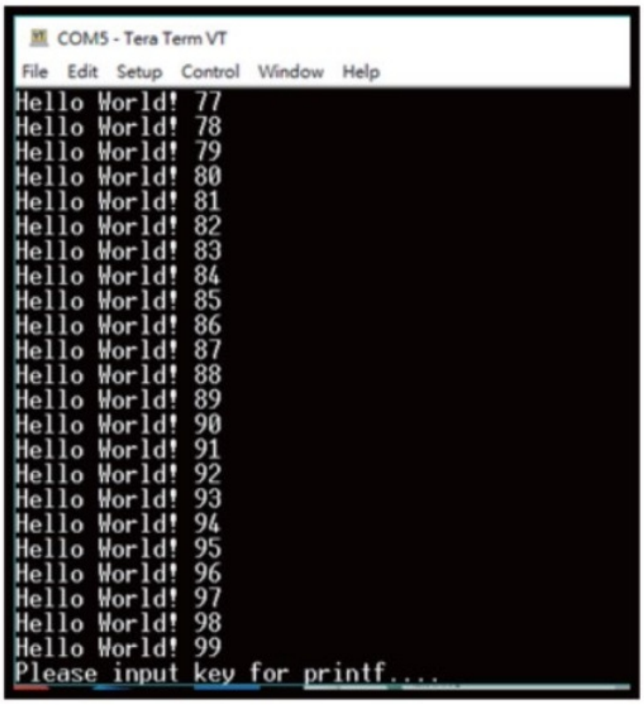
```
After Build - User command #1: afterbuild.bat .\HT32\52352\Obj\HT32 C:\Keil_v5\
creating binary file...
creating disassemble/text file...
After Build - User command #2: arm-none-eabi-size.exe  .\HT32\52352\Obj\HT32.elf
   text    data     bss     dec     hex filename
   3308       8      32    3348     d14 .\HT32\52352\Obj\HT32.elf
".\HT32\52352\Obj\HT32.elf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:02
Load "D:\\HT32_STD_5xxxx_FWLib_V1.0.25_5831\\project_template\\IP\\Example\\GNU_ARM\\HT32\\52352\\Obj\\HT32.elf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 15:32:43
```

Step 5. A jumper cap is placed on the DAP_TX and PA5 pins to short them, as shown in the following figure. Then the PC terminal software (Tera Term) will be configured and the COM Port will be set according to Step3. Refer to the “Terminal Software” section for the detailed Tera Term configuration.



Step 6. When the “Reset” key has been pressed, both LED1 and LED2 will blink as shown on the left side of the following figure. The messages “Hello World! 0” ~ “Hello World! 99” will appear in the “Tera Term” window via the Virtual COM Port, as shown on the right side of the following figure. This is used to verify that it has been successfully used with the Keil MDK-ARM uVision GNU Arm Compiler.





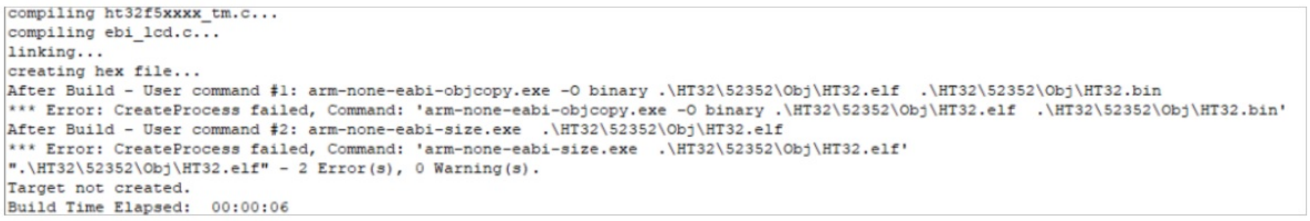
```
Hello World! 77
Hello World! 78
Hello World! 79
Hello World! 80
Hello World! 81
Hello World! 82
Hello World! 83
Hello World! 84
Hello World! 85
Hello World! 86
Hello World! 87
Hello World! 88
Hello World! 89
Hello World! 90
Hello World! 91
Hello World! 92
Hello World! 93
Hello World! 94
Hello World! 95
Hello World! 96
Hello World! 97
Hello World! 98
Hello World! 99
Please input key for printf....
```

Common Problems

This chapter assists with some of the common problems which may be encountered.

Error messages which may appear After Build

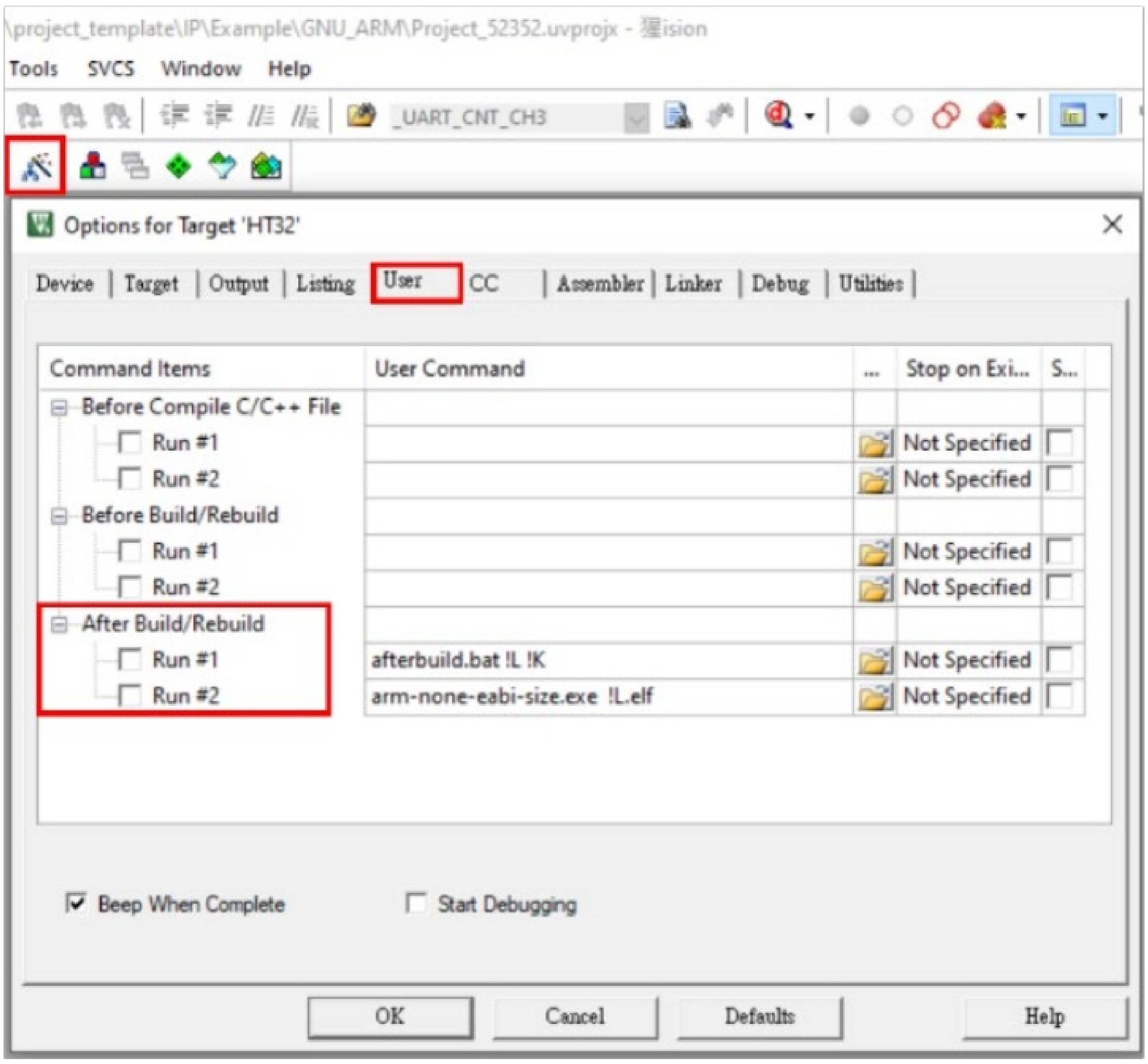
- If executing “After Build”, the following error message will be generated. Try to reboot the computer or run the Keil MDK-ARM as an administrator to make the “After Build” operation successful.



```
compiling ht32f5xxxx_tm.c...
compiling ebi_lcd.c...
linking...
creating hex file...
After Build - User command #1: arm-none-eabi-objcopy.exe -O binary .\HT32\52352\Obj\HT32.elf  .\HT32\52352\Obj\HT32.bin
*** Error: CreateProcess failed, Command: 'arm-none-eabi-objcopy.exe -O binary .\HT32\52352\Obj\HT32.elf  .\HT32\52352\Obj\HT32.bin'
After Build - User command #2: arm-none-eabi-size.exe  .\HT32\52352\Obj\HT32.elf
*** Error: CreateProcess failed, Command: 'arm-none-eabi-size.exe  .\HT32\52352\Obj\HT32.elf'
".\HT32\52352\Obj\HT32.elf" - 2 Error(s), 0 Warning(s).
Target not created.
Build Time Elapsed:  00:00:06
```

If this problem cannot be resolved using the above steps, the user can also disable the “After Build/Rebuild” option as shown in the following figure.

Note: When the “After Build/Rebuild” option is disabled, the Keil will no longer output binary format and Code size messages.



Command Items	User Command	...	Stop on Exi...	S...
Before Compile C/C++ File				
☐ Run #1			Not Specified	☐
☐ Run #2			Not Specified	☐
Before Build/Rebuild				
☐ Run #1			Not Specified	☐
☐ Run #2			Not Specified	☐
After Build/Rebuild				
☐ Run #1	afterbuild.bat !L !K		Not Specified	☐
☐ Run #2	arm-none-eabi-size.exe !L.elf		Not Specified	☐

☑ Beep When Complete ☐ Start Debugging

Firmware Library Version Requirements

If the “GNU Arm Compiler” is used with the “GNU Make” or the “Keil MDK-ARM uVision”, it should be noted that only the following version or higher Firmware Library versions support the GNU Arm project files.

- HT32_STD_5xxxx_FWLib_V1.0.26_nnnn.zip
- HT32_STD_1xxxx_FWLib_V1.0.11_nnnn.zip

Conclusion

This application note first provided a brief description of the GNU Arm. This was followed by an explanation to show users how to install and test the GNU Arm Compiler. It then described how to use it with the “GNU Make” or “Keil MDK-ARM uVision”. Finally, there was an explanation of how to use the GNU Arm Compiler with the HT32 MCUs.

Reference Material

For more information, consult to the Holtek official website: www.holtek.com.

Revision and Modification Information

Date	Author	Issue	Modification Information
2022.05.13		V1.00	First Version

Disclaimer

Limitation of Liability

Governing Law

Update of Disclaimer

Documents / Resources

	HOLTEK HT32 MCU GNU Arm Compiler [pdf] User Guide HT32 MCU, HT32 MCU GNU Arm Compiler, GNU Arm Compiler, Arm Compiler

References

- Make for Windows
- Arm GNU Toolchain
- HT32 Development Resource Download
- MDK-ARM Version 5.38a Evaluation Software Request