

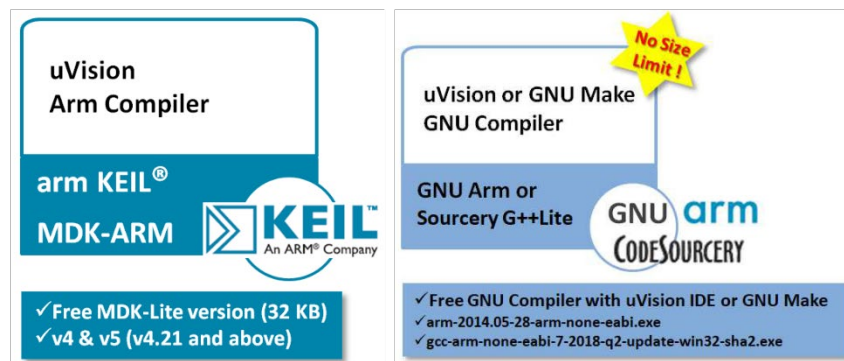
HT32 MCU GNU Arm Compiler User's Guide

D/N: AN0611EN

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 MDK-ARM 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.



Resource Download and Preparation

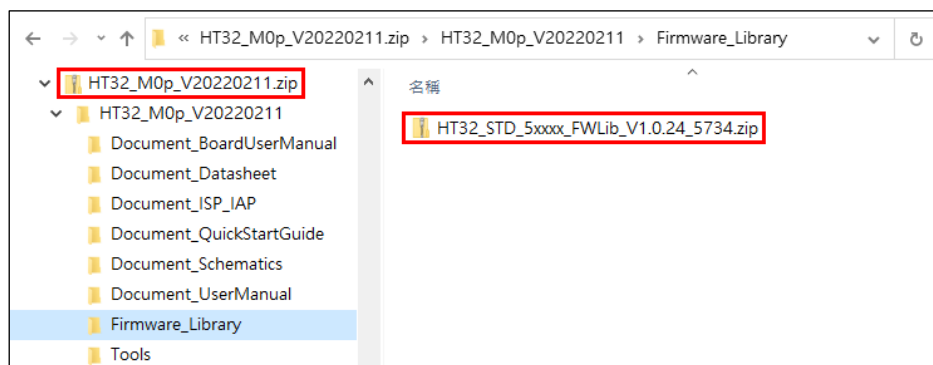
This chapter describes the example program and required software tools and explains how to configure the directory and file path.

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_5xxxx_FWLib_Vm.n.r_s.zip, as shown below.

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



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:

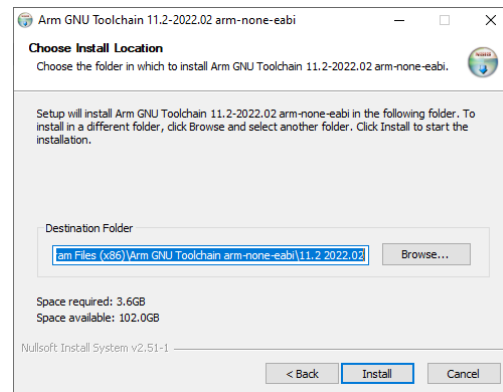
The file name used in this article is: “gcc-arm-11.2-2022.02-mingw-w64-i686-arm-none-eabi.exe”.

The file name for the latest discontinued version is: “gcc-arm-none-eabi-10.3-2021.10-win32.exe”.

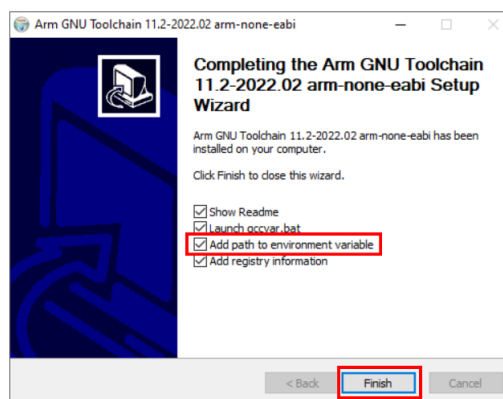
Step 2. In the “Choose Install Location” step, as shown in the following figure, save the installation path during installation. This path will be configured in Keil in the “Used with Keil MDK-ARM uVision” chapter.

For example:

“C:\Program Files (x86)\Arm GNU Toolchain arm-none-eabi\11.2 2022.02”.



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



Note: Reboot the computer when the installation has finished.

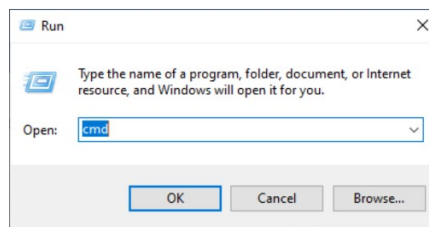
Test

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.

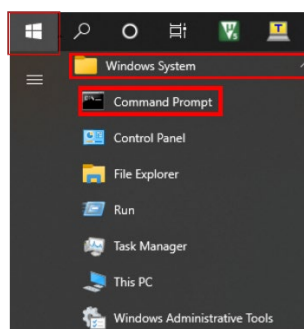
Step 1. Enable the Command Prompt.

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

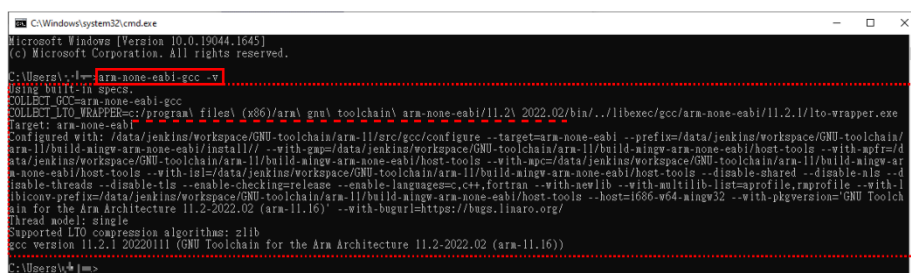
- Enable via the “Run” Window: First press “Windows + R” keys on the keyboard and enter “cmd” in the pop-up “Run” window as shown in the following figure. Then press “OK” to enable the Command Prompt.



- 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.



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.

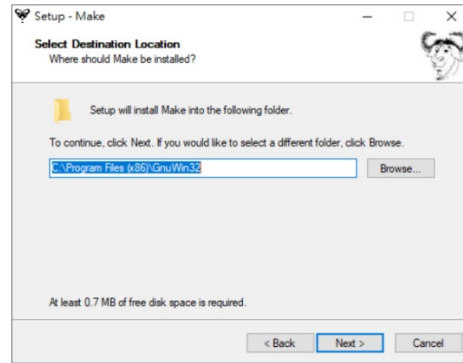
<http://gnuwin32.sourceforge.net/packages/make.htm>

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.

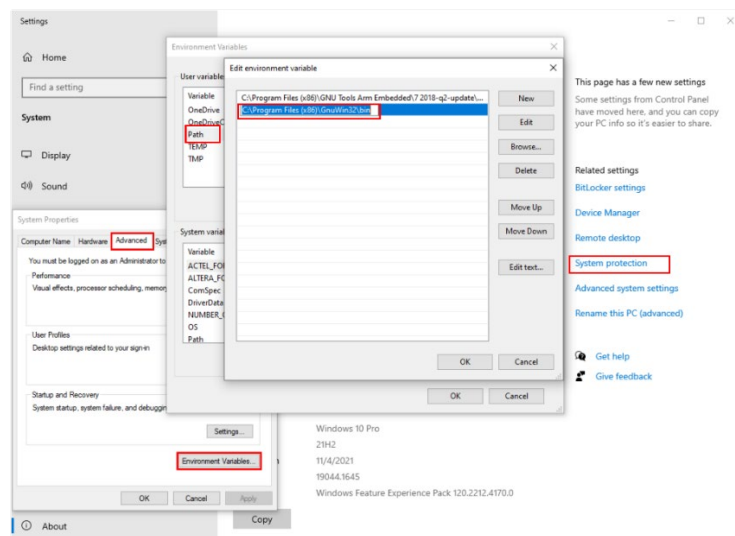
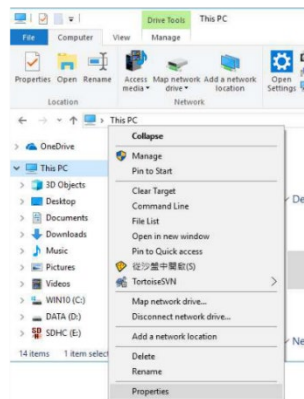
For example:

“C:\Program Files (x86)\GnuWin32”.

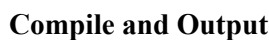


Step 3. Add an extra character “\bin” to the copied path in Step 2 and configure it in the environment variable “Path” to add a path for the GNU Make tool. Refer to the two following figures, which show how to enable the environment variable, and find the “Path” edit and add a path.

Note: The entire path is similar to “C:\Program Files (x86)\GnuWin32\bin”.



Note: Refer to the “Enable Command Prompt” contents in the “Test” section for the Command Prompt enabling method.



Step 1. Enable the Command Prompt and change the operating directory to the “GNU_ARM” folder in the Firmware Library Project Template.

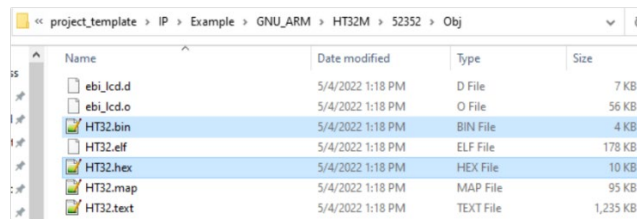
Step 2. Enter a “make xxxxx” or “make -f xxxxx.mk” command to build a program. When all build operations have finished, a “BUILD SUCCESSFUL” message should appear as shown in the following figure.

[illegible]

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”



| 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

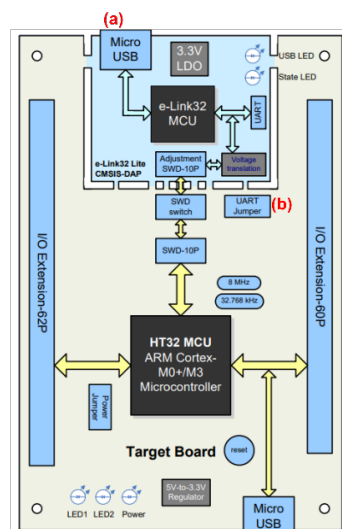
This section will take the HT32F52352 Starter Kit (SK) as an example. First, it introduces the environment preparation operations for the Starter Kit (SK) and e-Link32 Pro / Lite, and then explains how to use “make IC=xxxxx eraseall/program/run” and Command Prompt results in sequence. Finally it explains 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.

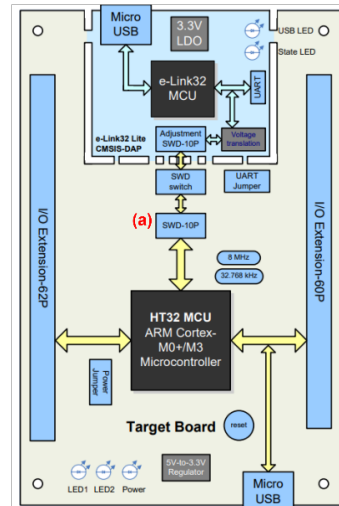
Note: 1. J2 on the SK provides two settings which are to short the PAX and DAP_Tx pins or short the PAX and RS232_Tx pins. Refer to the Starter Kit User Manual for details.

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.

```

Command Prompt
D:\HT32_STD_5xxxx_FWLb_V1.0.25_5831\project_template\IP\Example\GNU_ARM>make IC=52352 eraseall
make[1]: Entering directory 'D:/HT32_STD_5xxxx_FWLb_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
..\..\..\Utilities\link32pro\link32pro.exe -ne
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_FWLb_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
D:\HT32_STD_5xxxx_FWLb_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.

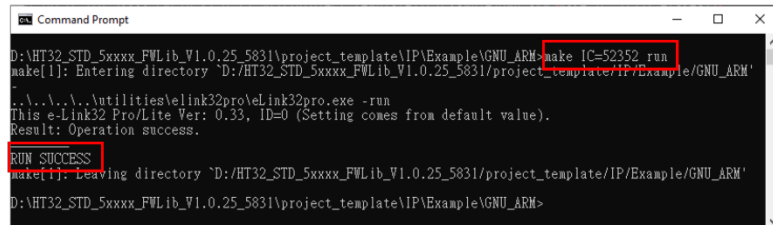
(xxxxx is the IC device, here a 52352 is used).

```

Command Prompt
D:\HT32_STD_5xxxx_FWLb_V1.0.25_5831\project_template\IP\Example\GNU_ARM>make IC=52352 program
make[1]: Entering directory 'D:/HT32_STD_5xxxx_FWLb_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
..\..\..\Utilities\link32pro\link32pro.exe -a ".\HT32M\52352\Obj\HT32.bin" -v ".\HT32M\52352\Obj\HT32.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_FWLb_V1.0.25_5831/project_template/IP/Example/GNU_ARM'
D:\HT32_STD_5xxxx_FWLb_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).

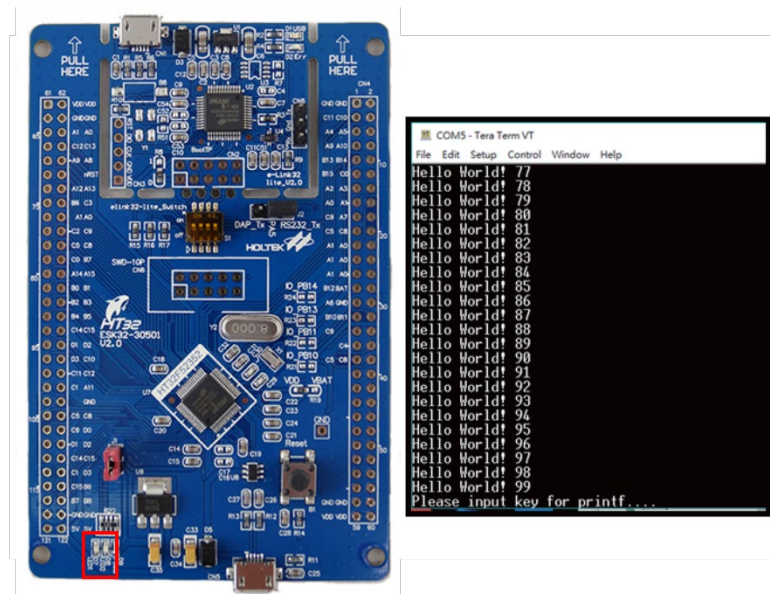


```

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.



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”.

```

/* 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_USB_Driver/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_bfm.c \
$(HT32_LIB_PATH)ht32f5xxxx_ckcu.c \

```

- 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”.

```

/* Include Path..... */
/*
INCLUDE_PATH += -I./
INCLUDE_PATH += -I./../../library/CMSIS/Include
INCLUDE_PATH += -I./../../library/Device/Holtek/HT32F5xxxx/Include
INCLUDE_PATH += -I./../../library/HT32F5xxxx_Driver/inc
INCLUDE_PATH += -I./../../library/HT32_USB_Driver/inc
INCLUDE_PATH += -I./../../utilities

#INCLUDE_PATH += -I$(GNU_PATH)arm-none-eabi/include"
#INCLUDE_PATH += -I$(GNU_PATH)lib/gcc/arm-none-eabi/7.3.1/include"
#INCLUDE_PATH += -I$(GNU_PATH)arm-none-eabi/include/c++/7.3.1"
#INCLUDE_PATH += -I$(GNU_PATH)arm-none-eabi/include/c++/7.3.1/arm-none-eabi"

```

- 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

```

/* Preprocessor Define..... */
/*
C_OPTION += -DUSE_HT32_DRIVER
C_OPTION += -DUSE_HT32F52352_SK
C_OPTION += -DUSE_HT32F52342_S2
C_OPTION += -DUSE_MEM_HT32F52352
S_OPTION = --defsym USE_HT32_CHIP=4

```

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”.

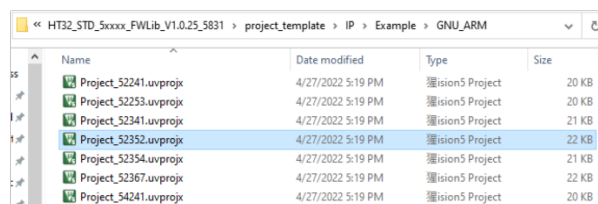
Project Settings

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

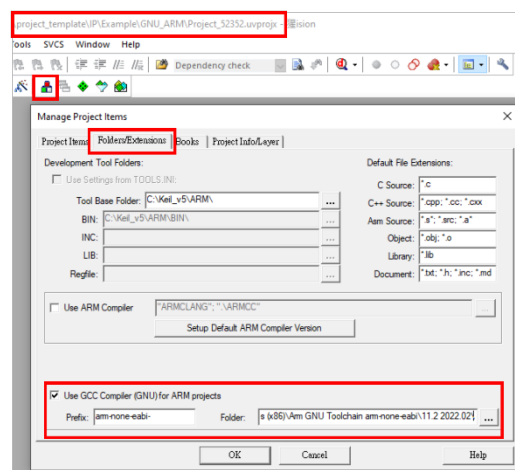
\\project_template\IP\Example\GNU_ARM\Project_XXXXX.uvprojx

Note: XXXXX is the device name.



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.

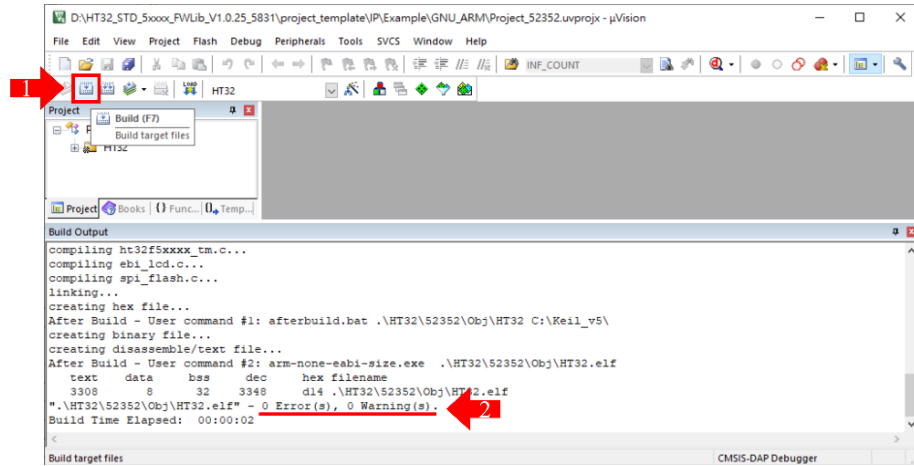
Note: For the GNU Arm installation path, refer to the path copied in Step 2 of the “GNU Arm Installation” section in the “GNU Arm Compiler Installation” chapter.



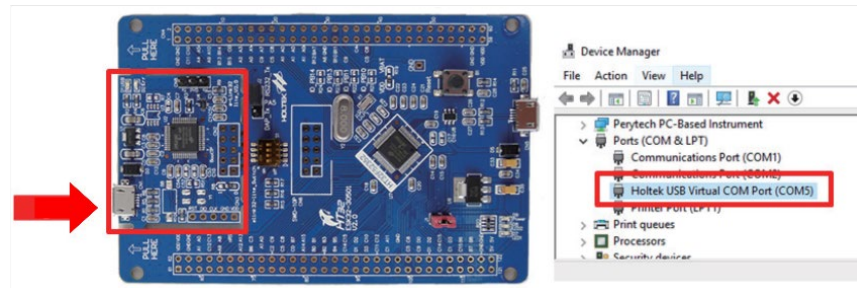
Compile and Test

Step 1. Click “Build (F7)” to build a project.

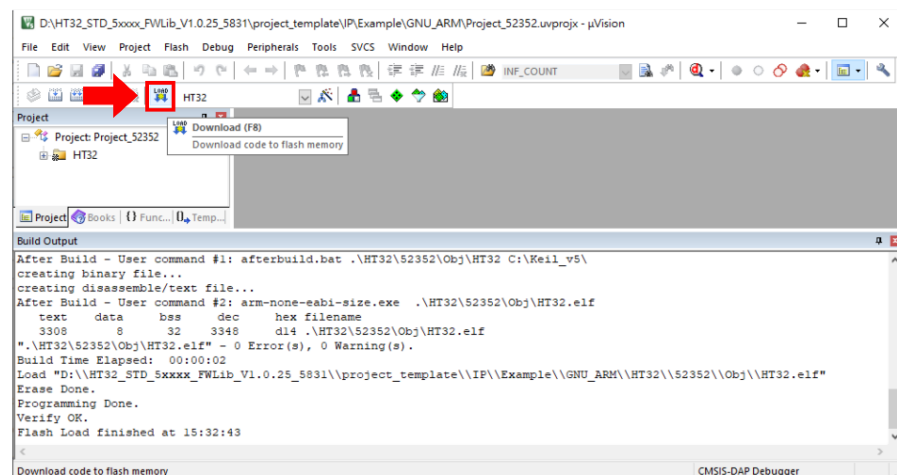
Step 2. Check “Build Output” window to confirm whether the program has been built correctly.



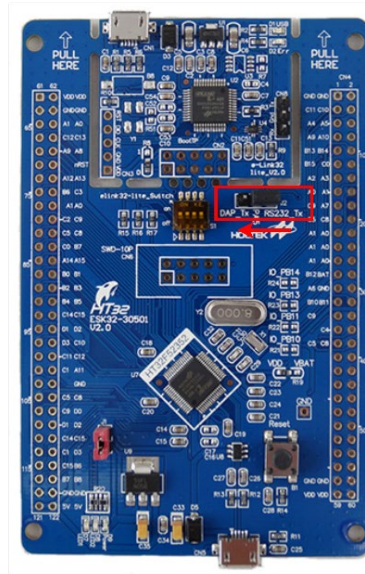
Step 3. Connect the e-link32 lite USB COM port on the HT32F52352 Starter Kit to the PC, as shown in the red box on the left side of the following figure. Confirm that the PC has detected the USB device normally, as shown in the red box on the right side of the following figure.



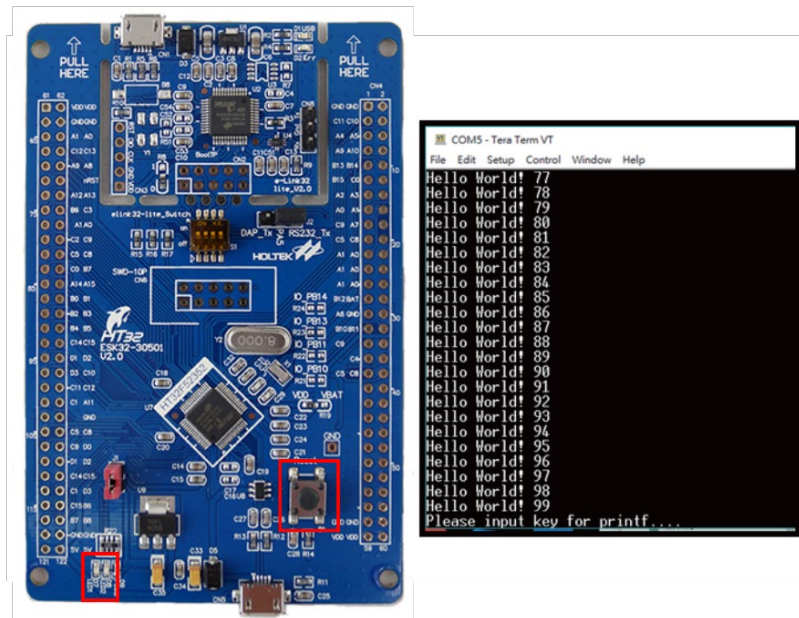
Step 4. Click “Download (F8)” to download the code to the Flash memory.



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.



Common Problems

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

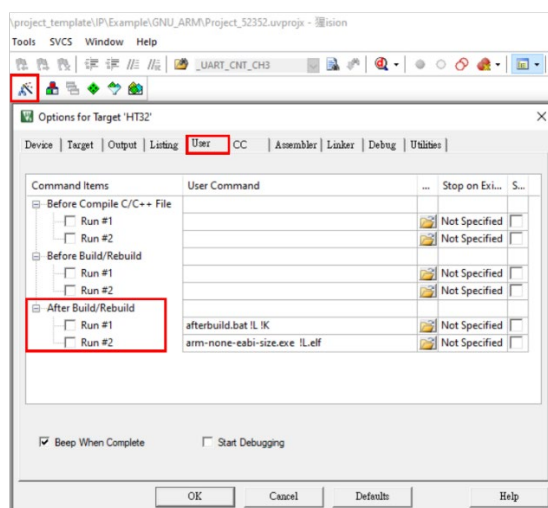
Error messages which may appear After Build

- 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 ht32f5xxx_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.



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 | 蔡期育 | V1.00 | First Version |

Disclaimer

All information, trademarks, logos, graphics, videos, audio clips, links and other items appearing on this website ('Information') are for reference only and is subject to change at any time without prior notice and at the discretion of Holtek Semiconductor Inc. and its related companies (hereinafter 'Holtek', 'the company', 'us', 'we' or 'our'). Whilst Holtek endeavors to ensure the accuracy of the Information on this website, no express or implied warranty is given by Holtek to the accuracy of the Information. Holtek will bear no responsibility for any incorrectness or leakage.

Holtek will not be liable for any damages (including but not limited to computer virus, system problems or data loss) whatsoever arising in using or in connection with the use of this website by any party. There may be links in this area, which allow you to visit the websites of other companies. These websites are not controlled by Holtek. Holtek will bear no responsibility and no guarantee to whatsoever Information displayed at such sites. Hyperlinks to other websites are at your own risk.

Limitation of Liability

In no event shall Holtek Limited be liable to any other party for any loss or damage whatsoever or howsoever caused directly or indirectly in connection with your access to or use of this website, the content thereon or any goods, materials or services.

Governing Law

The Disclaimer contained in the website shall be governed by and interpreted in accordance with the laws of the Republic of China. Users will submit to the non-exclusive jurisdiction of the Republic of China courts.

Update of Disclaimer

Holtek reserves the right to update the Disclaimer at any time with or without prior notice, all changes are effective immediately upon posting to the website.