ASR6601

Program Development Quick Start Guide

Version: 1.3.1

Date: 2024-03-11

About this document

This document describes the development environment settings and programming of the IoT LPWAN SoC chip ASR6601 SDK, which helps users to quickly develop programs for ASR6601.

Target Audience

This document is mainly intended for the following engineers:

- hardware development engineer
- software engineer
- technical support engineer

Product numbering

Product models corresponding to this document:

Model	Flash	SRAM	Core	Package	Frequency
ASR6601SE	256 KB	64 KB	32-bit 48 MHz Arm China STAR-MC1 Processor	QFN68, 8*8 mm	150 ~ 960 MHz
ASR6601CB	128 KB	16 KB	32-bit 48 MHz Arm China STAR-MC1 Processor	QFN48, 6*6 mm	150 ~ 960 MHz
ASR6601SER	256 KB	64 KB	32-bit 48 MHz Arm China STAR-MC1 Processor	QFN68, 8*8 mm	150 ~ 960 MHz
ASR6601CBR	128 KB	16 KB	32-bit 48 MHz Arm China STAR-MC1 Processor	QFN48, 6*6 mm	150 ~ 960 MHz

Copyright Notice

Trademark Notice

ASR, ASR and other ASR logos are trademarks of ASR Technology Co., Ltd.

All other trade names, trademarks and registered trademarks mentioned in this document are the property of their respective owners and are hereby acknowledged.

Disclaimer

ASR Technology Co., Ltd. makes no warranty of any kind for the contents of this document and will update the contents of this document or the products described in this document from time to time.

This document is only used as a guide for use, and all contents of this document do not constitute any form of warranty. The information in this document is subject to change without notice.

This document does not assume any responsibility, including responsibility for infringement of any proprietary rights arising from the use of the information in this document.

ASR Technology Ltd.

Address: 9th Floor, Building 10, Zhangjiang Innovation Park, 399 Keyuan

Road, Pudong New District, Shanghai Postal Code: 201203

URL：http://www.asrmicro.com/

Document Revision History

Date	Version	Release Notes
2020.05	V0.1.0	Initial Release.
2020.08	V0.2.0	● Added Keil environment development instructions. ● Added new Q&A content.
2020.09	V0.3.0	Updated images.
2020.10	V0.4.0	Updated with ASR6601SE-EVAL v2.0 diagram.
2021.01	V1.1.0	Removed the overview from Chapter 1 and merged its contents into the "About This Document" section of the preface.
2021.03	V1.2.0	Modified Chapter 3 to simplify the compilation process.
2022.06	V1.3.0	Updated the methods for obtaining ASR SDK and GCC toolchain.
2024.03	V1.3.1	Updated description of Section 2.1.

Table of contents

List of Tables

List of Figures

1. Hardware

Hardware required:

(1) ASR6601 development board 1 pc
(2) Antenna 1 pc
(3) USB cable 1 pc
(4) PC 1 pc

1.1 ASR6601 Development Board Description

The top and bottom sides of the development board ASR6601SE-EVAL v2.0 are shown in Figure 1-1 and Figure 1-2:

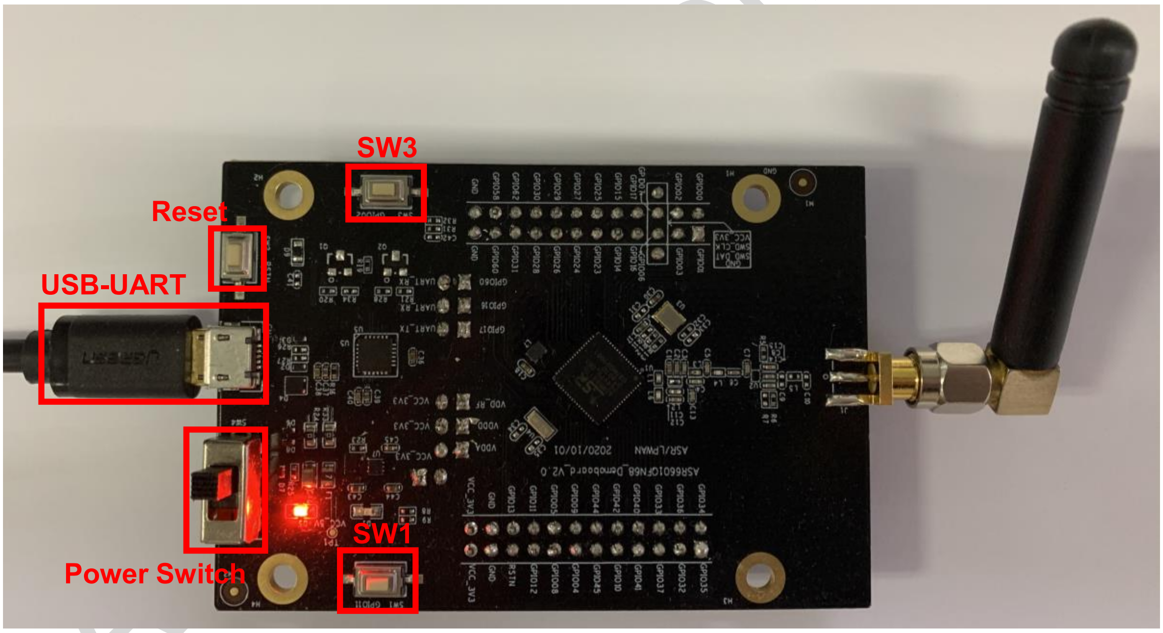


Figure 1-1 ASR6601SE-EVAL v2.0 top view

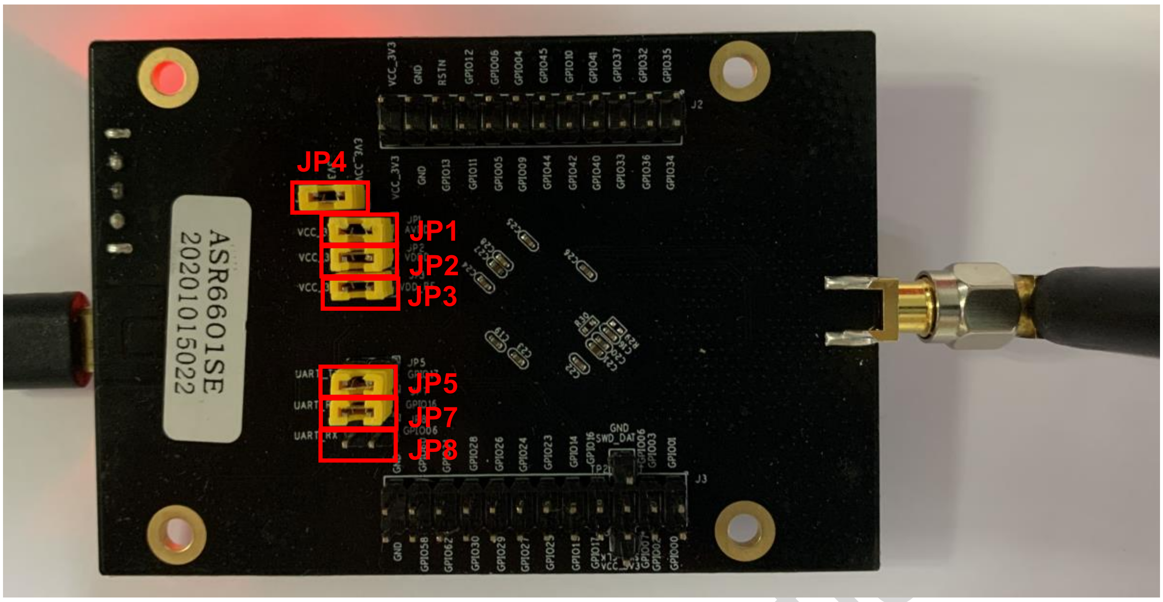


Figure 1-2 ASR6601SE-EVAL v2.0 bottom view

Table 1-1 ASR6601SE-EVAL v2.0 interface description

Interface	Description
USB-UART	USB to Serial adapter
Power Switch	Switch
Reset	Reset button
SW3	Download button, GPIO02 is pulled high if pressed
SW1	User button, GPIO11 is pulled low if pressed
JP1	Power jumper
JP2	Power jumper
JP3	Power jumper
JP4	Power jumper, for current sensor
JP5	UART_TX jumper, selects UART0_TX
JP6 (Only available in ASR6601CB-EVAL)	UART_TX jumper, selects LPUART_TX
JP7	UART_RX jumper, selects UART0_RX
JP8	UART_RX jumper, selects LPUART_RX

1.2 Jumper connection

When operating with ASR6601 development board, please ensure that the following jumpers are in correct state.

Table 1-2 Jumper connection

Jumper	Status
JP1	Close
JP2	Close
JP3	Close
JP4	Close
JP5	Close
JP6（ASR6601CB-EVALonly）	Open
JP7	Close
JP8	Open

2. Using the KEIL development environment

2.1 J-Link connection

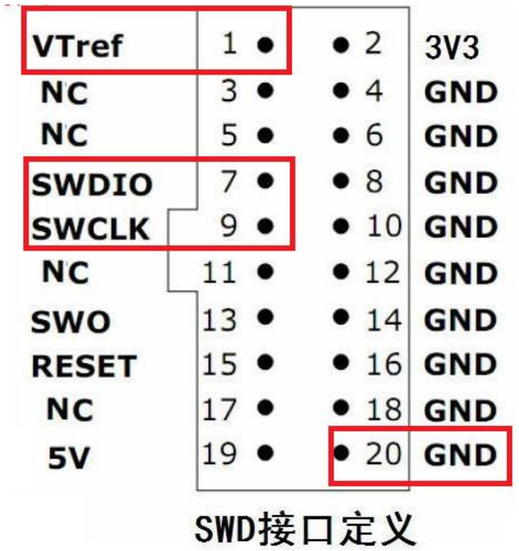


Figure 2-1 SWD interface

When using J-Link with ASR6601, you need to connect 4 wires, just connect pins 1, 7, 9 and 20 in the above figure to the corresponding pins of the board. Please note that it is forbidden to connect the RESET pin of J-LINK to the RESET pin of ASR6601, which may cause the risk of flash erasure.

2.2 Obtaining SDK

You can contact ASR technical support to obtain it, or download it from GitHub using the command below:

```
git clone https://github.com/asrlora/asr_lora_6601.git
```

2.3 KEIL project files generation

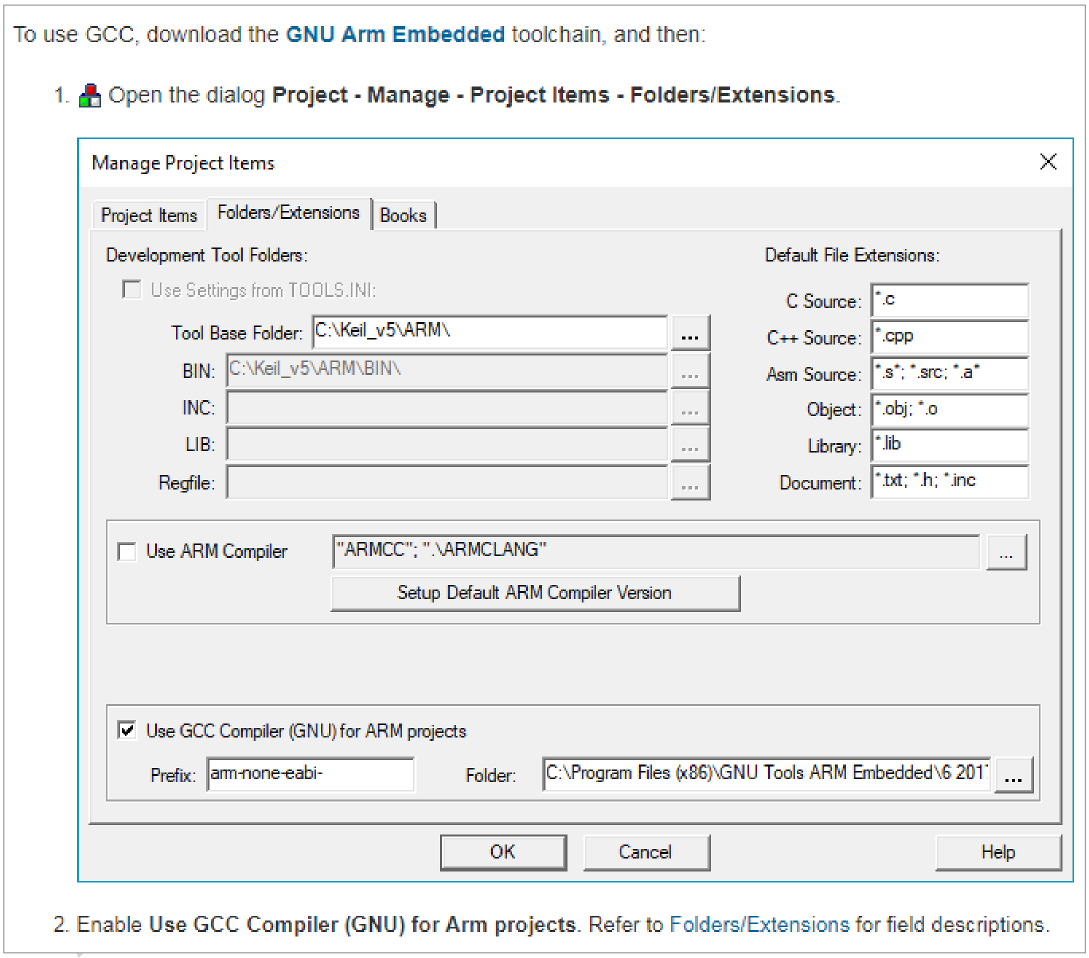
The SDK does not provide KEIL project files. You can run the keil.bat in the sample program to generate KEIL project files.

2.4 GCC toolchain configuration

(1) Download the GNU Arm Embedded Toolchain from the link below and unzip it.
https://developer.arm.com/-/media/Files/downloads/gnu-rm/9-2020q2/gcc-arm-none-eabi-9-2020-q2-update-win32.zip

(2) Follow the KEIL User Guide to set up the GCC toolchain. The link is below:
https://www.keil.com/support/man/docs/uv4/uv4_gnucomp.htm

Set Tool Base Folder to the directory just extracted, for example: D:\ASR6601_rel\tools\toolchain



To use GCC, download the **GNU Arm Embedded** toolchain, and then:

1. Open the dialog **Project - Manage - Project Items - Folders/Extensions**.

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

Asm Source: *.s*; *.src; *.a*

Object: *.obj; *.o

Library: *.lib

Document: *.txt; *.h; *.inc

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

☑ Use GCC Compiler (GNU) for ARM projects

Prefix: arm-none-eabi-

Folder: C:\Program Files (x86)\GNU Tools ARM Embedded\6 201

2. Enable **Use GCC Compiler (GNU) for Arm projects**. Refer to Folders/Extensions for field descriptions.

Figure 2-2 GCC toolchain configuration

2.5 Flash Programming Algorithm configuration

First, copy the ASR6601.FLM file in the \tools\FLM directory to the Keil Flash directory, for example: C:\Keil_v5\ARM\Flash.

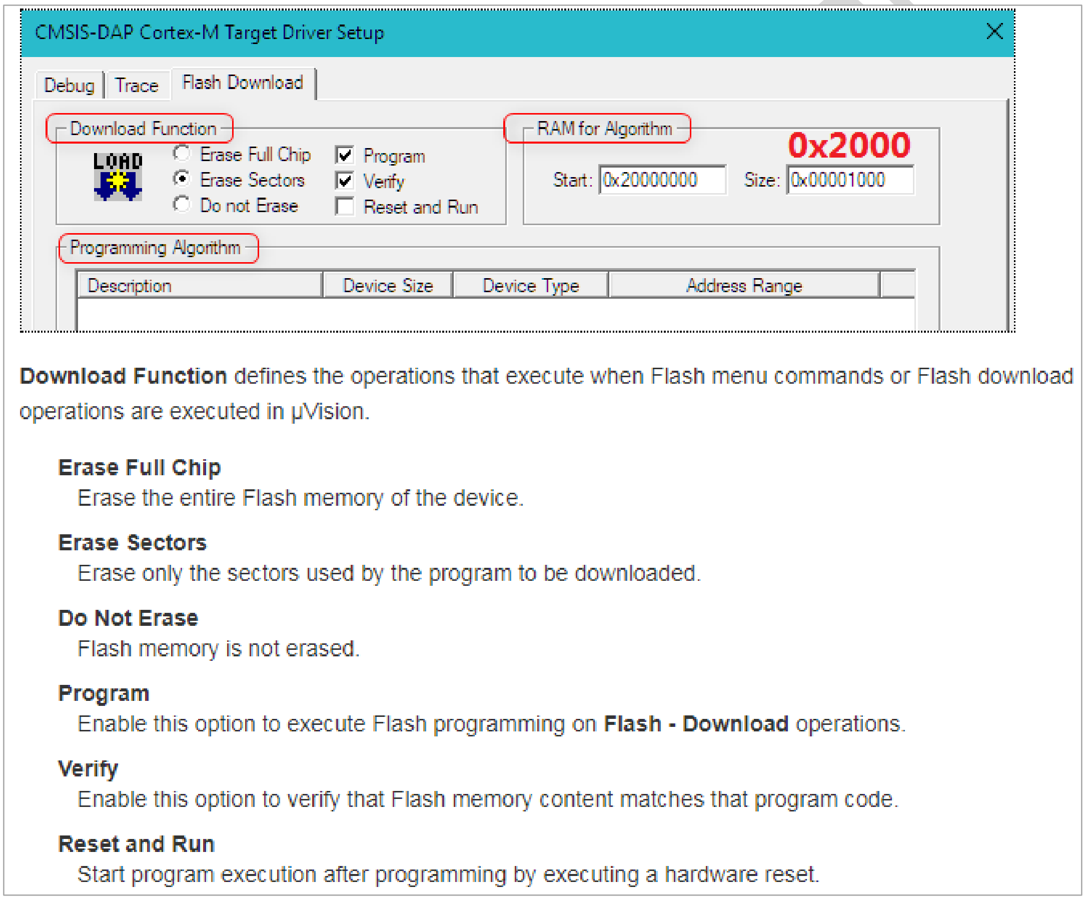
If you still cannot flash SoC, please modify the Flash Download configuration according to the KEIL user guide document. The document link is below: https://www.keil.com/support/man/docs/uv4/uv4_fl_dlconfiguration.htm.

The main configuration is as follows: ：

(1) **Download Function:** Check “Erase Sectors”, “Program” and “Verify”

(2) **RAM for Algorithm:** Set Start to 0x20000000 and Size to 0x2000

(3) **Programming Algorithm**: Added Flash Programming Algorithm file for ASR6601.



Download Function
○ Erase Full Chip
● Erase Sectors
○ Do not Erase
☑ Program
☑ Verify
☐ Reset and Run

RAM for Algorithm
Start: 0x20000000 Size: 0x00001000
0x2000

Programming Algorithm

Description	Device Size	Device Type	Address Range

Download Function defines the operations that execute when Flash menu commands or Flash download operations are executed in µVision.

Erase Full Chip
Erase the entire Flash memory of the device.

Erase Sectors
Erase only the sectors used by the program to be downloaded.

Do Not Erase
Flash memory is not erased.

Program
Enable this option to execute Flash programming on **Flash - Download** operations.

Verify
Enable this option to verify that Flash memory content matches that program code.

Reset and Run
Start program execution after programming by executing a hardware reset.

Figure 2-3 Flash Programming Algorithm configuration

2.6 Compilation and flashing

(1) Click the "Build" button to compile

(2) Click the "Download" button to flash

3. Working with Make Command Line

3.1 Prerequisities

3.1.1 Development environment installation

3.1.1.1 Ubuntu environment (Ubuntu 18.04)

Run the following command to install Python and other necessary software:

```
sudo apt-get install gcc-arm-none-eabi git vim python python-pip
pip install pyserial configparser
```

3.1.1.2 Windows environment

(1) **Install MSYS2**

Log in to the official website（https://www.msys2.org/）

Download the installation package and install it

(2) **Install related software**

Open MSYS2 and install related software: pacman -S git vim make unzip python python-pip wget

Install using pip pyserial：pip install pyserial configparser

3.1.2 Obtaining SDK

You can contact ASR technical support to obtain it, or download it from GitHub using the command below.

```
git clone https://github.com/asrlora/asr_lora_6601.git
```

3.2 Compilation and flashing

The following takes the uart_printf project as an example to explain the software compilation and flashing process.

3.2.1 Project compilation

Compile the project:

(1) Execute command to configure the environment variables: source build/envsetup.sh

(2) Enter the uart_printf directory: cd projects/ASR6601CB-EVAL/examples/uart/uart_printf (3)

Execute make command: make

After successful compilation, the results are as follows:

```
Build completed.

arm-none-eabi-size out/uart_printf.elf

    text       data        bss        dec        hex filename

    9972       1080       4164      15216       3b70 out/uart_printf.elf

Please run 'make flash' or the following command to download the app

python  /home/ruilinhao/work/ASR6601_rel/build/scripts/tremo_loader.py  -p  /dev/ttyUSB0  -b
921600 flash 0x08000000 out/uart_printf.bin
```

3.2.2 Flashing

There are two ways to flash:

● Use the programming tool to flash. For details, please refer to the document "ASR6601_ Programming Tool Manual".

● Use command line to flash.

The following focuses on the steps of command line flashing:

(1) **Serial port configuration**

First execute the command ls /dev/ to view the serial port used by the development board.

Usually under MSYS2, there will be ttyS* devices, which are serial port devices. They correspond to the COM port numbers under Windows. For example, COM6 corresponds to /dev/ttyS5 in MSYS2; under Ubuntu, serial port devices are usually /dev/ttyUSB*.

After finding the serial port device, modify the Makefile of the uart_printf project, remove the "#" symbol in front of SERIAL_PORT, and change SERIAL_PORT to the corresponding serial port number.

If there is no special requirement, SERIAL_BUADRATE and $(PROJECT)_ADDRESS do not need to be modified and the default values are used.

```
SERIAL_PORT             :=/dev/ttyS5
#SERIAL_BAUDRATE     :=
#$(PROJECT)_ADDRESS :=
```

(2) **Enter download mode**

Before flashing, please press and hold the SW3 button on the board to pull GPIO02 high, then click the Reset button to restart and enter download mode.

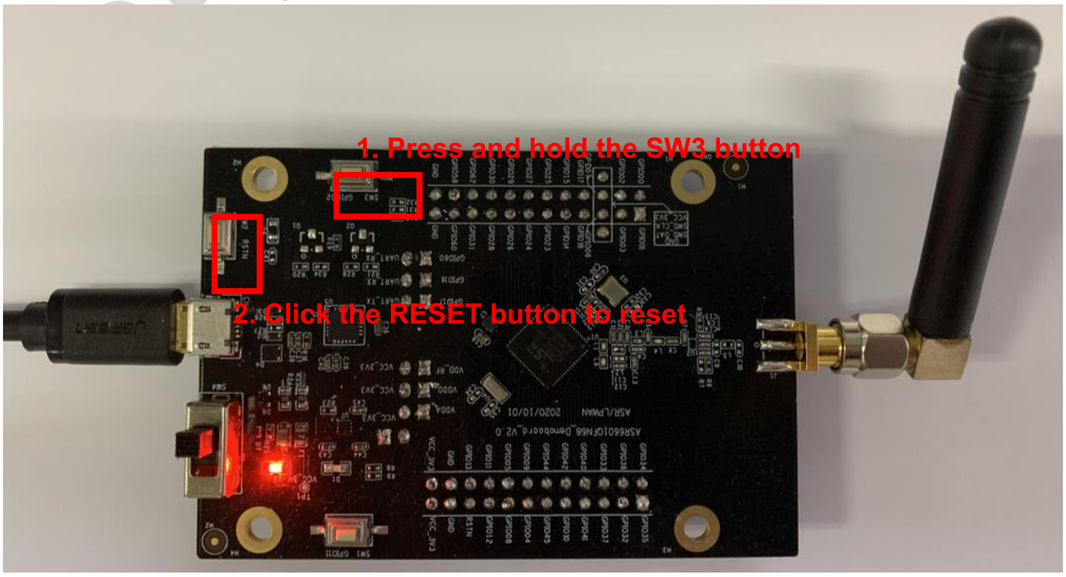

1. Press and hold the SW3 button
2. Click the RESET button to reset

Figure 3-1 Download mode entering diagram

(3) **Start flashing**

Finally, execute the make flash command or use the tremo_loader.py custom command to flash.

If the flashing is successful, the following information will be displayed. If the flashing is still unsuccessful, please refer to the relevant QA content in Chapter 4.

```
Connecting...
Connected
('send: ', 512)
('send: ', 1024)
('send: ', 1536)
('send: ', 2048)
('send: ', 2560)
('send: ', 3072)
('send: ', 3584)
('send: ', 4096)
('send: ', 4100)
Download files successfully
```

3.2.3 Run

After the flashing is completed, restart to run the program smoothly. The serial port tool interface prints: hello world

4. Q&A

4.1 When using KEIL to flash, SW Deice never appears. What should I do?

The absence of SW Device may be caused by the following two situations:

1. Hardware connection problem, you need to check the wiring, power supply and other hardware connections.

2. If a hard fault occurs or the MCU enters low power consumption, the SW device will not appear. In this case, you can use a Dupont line to pull the GPIO02 pin high, then restart the MCU to enter the bootloader, and the SW device will appear and can be flashed again.

4.2 When using MSYS2 environment for flashing, the corresponding serial port device cannot be found. What should I do?

There is a maximum number of serial port devices in MSYS2 (64 or 128 depending on the version). If the serial port device port number is too large, it will not be found in MSYS2. You can change the serial port number to a smaller one, and it will appear in MSYS2.

4.3 When using MSYS2 environment for flashing, I can see the serial port device, but the flashing is always unsuccessful. What should I do?

1. Check whether the serial port is opened by other software, such as serial port tools.

2. On some Windows versions, using /dev/ttyS* directly will fail. Try changing SERIAL_PORT to COM* in the Makefile.