AN0015

Application Note

AT32 Printf Debug Demo

Introduction

It is common for the user to check the debug process information in the process of application code debug. In most cases, this action can be done through the serial port debugging assistant. However, it would be a headache for users to try to observe the testing process information when the serial port assistant is not supported by the hardware.

To address the above concerns, this application note provides a complete set of example codes describing how to output the debug process information, especially when the serial port debugging assistant is not available.

Note: The corresponding code in this application note is developed on the basis of V2.x.x BSP provided by Artery. For other versions of BSP, please pay attention to the differences in usage.

Applicable products:

Part number	All AT32F series

Contents

List of Tables

List of Figures

1 Overview

This application note describes how to use the printf function of AT chips in the environment of Keil and IAR. There are six methods to be listed in the table below, with each of them being detailed in this document.

Table 1. AT MCUs printf function use methods

AT MCUs printf function use methods			
No.	**Environment**	**Description**	**Remark**
Method 1	IAR	Printf via Terminal I/O	See 3.1
Method 2		Redirect Printf as serial ports	See 3.2
Method 3	Keil	Printf via Debug (printf) Viewer	See 3.3
Method 4		Redirect Printf as serial ports (using MicroLIB)	See 3.4
Method 5		Redirect Printf as serial ports (not use MicroLIB)	See 3.5
Method 6	IAR/Keil	Printf via JLinkRTTClient window	See 3.6

2 Detailed information

2.1 Printf via Terminal I/O in IAR

2.1.1 Introduction

The IAR driver linked to the Terminal contains standard input and output driver functions such as scanf and printf so that the information interaction of the project files can be done via the Terminal I/O.

2.1.2 Example code

001_Printf_Test_IAR_Terminal\project\iar_v8.2

2.1.3 Environment and hardware

2.1.3.1 Development environment

This method is used in the IAR environment. The compiling environment used in the example code is IAR_V8, with the hardware board AT-START-F403A_V1.2.

2.1.3.2 Hardware connection

J-Link/AT-Link&... connection

Table 2. Hardware connection (without JTDO)

Hardware connection (without JTDO)			
No.	**AT-START-F403A_V1.2**	**J-Link/AT-Link&…**	**Attention**
1	3.3V	3.3V	None
2	PA13	SWDIO	Must Pull up external
3	PA14	SWCLK	Must Pull down external
4	NRST	RSTn	None
5	GND	GND	None

2.1.4 Software

2.1.4.1 Header file

Add the “stdio.h” to the code project files.

2.1.4.2 Redirection settings

Unlock the redirection of Printf (shield the Printf from the actual serial interface).

2.1.5 Debug and download

Compile and download to the MCU, then enter the debug environment to call up the virtual terminal through View->Terminal I/O (Figure 1), and run the code at full speed, then the “Hello World” is visible in the Output column (Figure 2), and the data in the Input column is also displayed in this window.

Figure 1. Virtual terminal window path

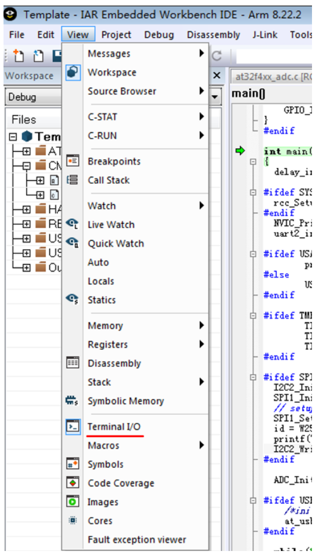
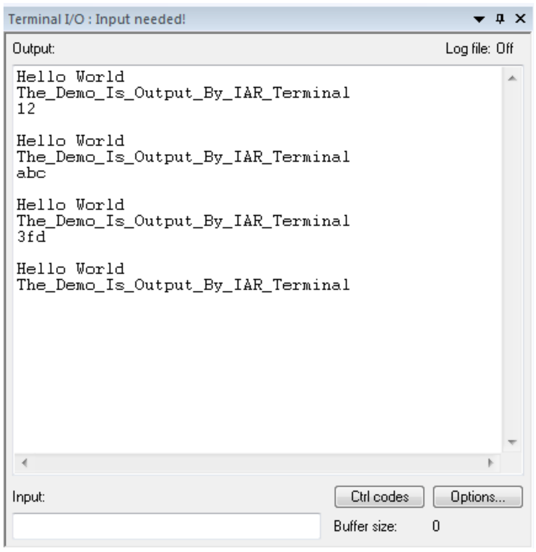


Figure 2. Virtual terminal interaction window



```
Hello World
The_Demo_Is_Output_By_IAR_Terminal
12

Hello World
The_Demo_Is_Output_By_IAR_Terminal
abc

Hello World
The_Demo_Is_Output_By_IAR_Terminal
3fd

Hello World
The_Demo_Is_Output_By_IAR_Terminal
```

2.2 Redirect Printf as serial ports in IAR

2.2.1 Introduction

Redirect the Printf function to a set of actual serial ports in the chip, and output via TX pin and finally implement information interaction through the serial port debugging assistant.

2.2.2 Example code

002_Printf_Test_IAR_USART2\project\iar_v8.2Environment and hardware

2.2.3 Development environment

2.2.3.1 Hardware connection

This method is used in the IAR environment. The compiling environment used in the example code is IAR_V8, with the hardware board AT-START-F403A_V1.2.

2.2.3.2 Hardware connection

2.2.3.2.1 J-Link/AT-Link&... connection

J-Link/AT-Link&... connection

Table 3. Hardware connection (without JTDO)

Hardware connection (without JTDO)			
No.	**AT-START-F403A_V1.2**	**J-Link/AT-Link&…**	**Attention**
1	3.3V	3.3V	None
2	PA13	SWDIO	Must Pull up external
3	PA14	SWCLK	Must Pull down external
4	NRST	RSTn	None
5	GND	GND	None

2.2.3.2.2 USART2 connection

Table 4. Hardware connection (USART)

Hardware connection (USART)			
No.	**AT-START-F403_V1.2**	**USB_To_TTL(CH340)**	**Attention**
1	GND	GMD	None
2	PA2	RXD	None
3	PA3	TXD	None

2.2.4 Software

2.2.4.1 Header file

Add the “stdio.h” to the code project files.

2.2.4.2 Redirection settings

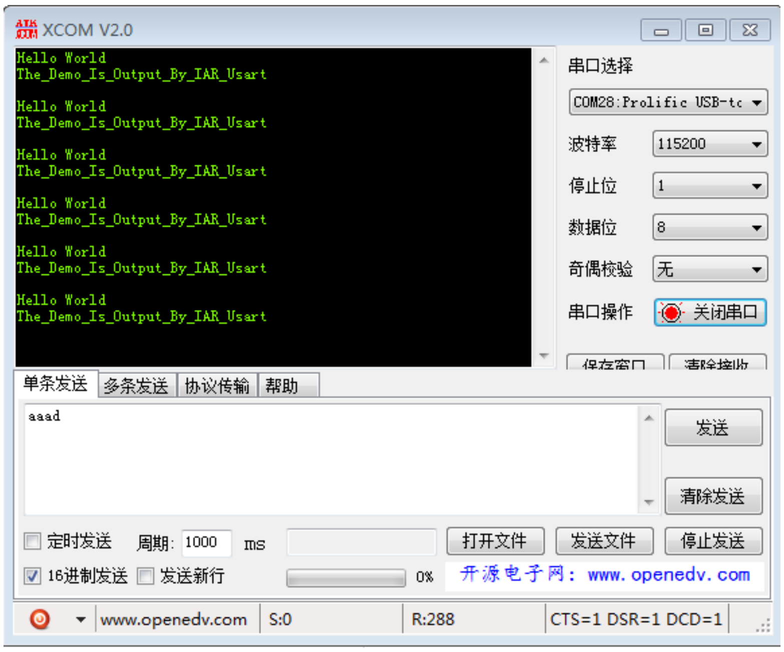
Initialize the serial ports and redirect the Printf function to the actual serial ports. The redirection function is as follows:

```
PUTCHAR_PROTOTYPE
{
    USART_SendData( USART2, ch);
    while ( USART_GetFlagStatus( USART2, USART_FLAG_TRAC) == RESET );
    return ch;
}
```

2.2.5 Debug and download

Compile the code and download to the MCU, and run it at full speed, then you can see that the “Hello World” is displayed in the terminal interaction window (Figure 3).

Figure 3. Serial port debugging assistant window



XCOM V2.0

```
Hello World
The_Demo_Is_Output_By_IAR_Usart

Hello World
The_Demo_Is_Output_By_IAR_Usart

Hello World
The_Demo_Is_Output_By_IAR_Usart

Hello World
The_Demo_Is_Output_By_IAR_Usart

Hello World
The_Demo_Is_Output_By_IAR_Usart

Hello World
The_Demo_Is_Output_By_IAR_Usart
```

串口选择: COM28:Prolific USB-tc
波特率: 115200
停止位: 1
数据位: 8
奇偶校验: 无
串口操作: 关闭串口

aaad

定时发送 周期: 1000 ms
[✓] 16进制发送 发送新行
0%
开源电子网: www.openedv.com
www.openedv.com S:0 R:288 CTS=1 DSR=1 DCD=1

2.3 Printf via Debug(printf) Viewer in Keil

2.3.1 Introduction

The Keil platform comes with a Debug(printf) Viewer that can be used for standard Printf interaction on the premise that the ARM core integrates standard input and output driver functions such as scanf and printf.

2.3.2 Example code location

003_Printf_Test_Keil_JTDO\project\mdk_v 5Environment and hardware

2.3.3 Development environment

2.3.3.1 Environment

This method is used in the Keil environment. The compiling environment used in the example code is Keil_V5, with the hardware board AT-START-F403A_V1.2.

2.3.3.2 Hardware connection

J-Link/AT-Link&... connection

Table 5. Hardware connection (with JTDO)

Hardware connection (with JTDO)			
No.	**AT-START-F403A_V1.2**	**J-Link/AT-Link&…**	**Attention**
1	3.3V	3.3V	None
2	PA13	SWDIO	Must Pull up external
3	PA14	SWCLK	Must Pull down external
4	NRST	RSTn	None
5	PB3	JTDO	Must Pull up external
6	GND	GND	None

2.3.4 Software

2.3.4.1 Header files

Add the “stdio.h” to the code project files.

2.3.4.2 Trace pin assignment

```
DEBUG->ctrl_bit.trace_ioen = FALSE;
DEBUG->ctrl_bit.trace_ioen = TRUE;
```

2.3.4.3 Printf mapping

```
int fputc(int c, FILE *f)
{
if (c == '\n')
{
SER_PutChar('\r');
}
return (SER_PutChar(c));
}
int SER_PutChar (int c)
{
ITM_SendChar(c);
return (c);
}
```

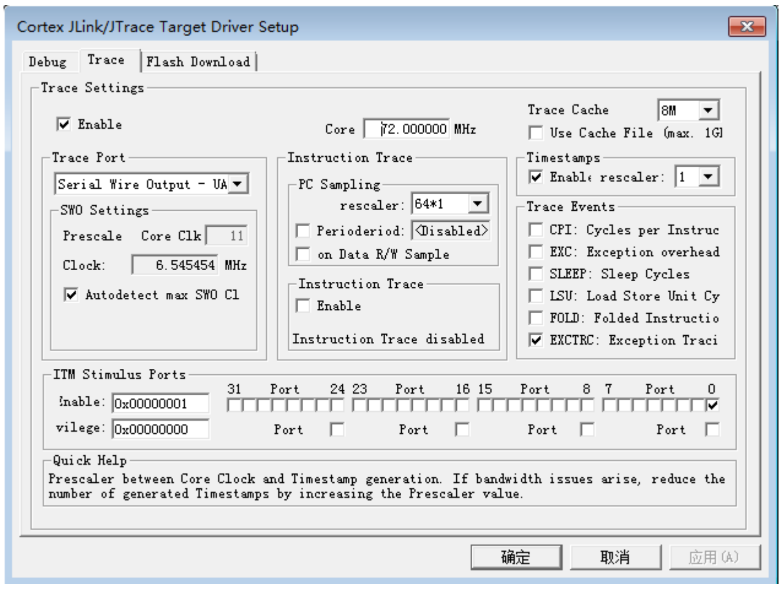
2.3.5 Debug and download

Tick the Enable box (Figure 4) and set the Core value, and the Core value must be equal to the system clock.

Configure the serial clock by checking the Autodetect max SWO C1 box (Figure 4). If garbled characters are displayed, untick the Autodetect max SWO C1 box and manually modify the Prescale Core Clk to ensure that the printed information is correct.

Compile the code and download to the MCU, and then enter debug environment to call up the virtual terminal window through View->Serial Windows->Debug (printf) Viewer (Figure 5); then run the code, and "Hello World" is visible in the terminal interaction window (Figure 6).

Figure 4. Trace target driver setup



Trace Settings
[✓] Enable
Core: 72.000000 MHz
Trace Cache: 8M
[ ] Use Cache File (max. 1G)
Trace Port: Serial Wire Output - UA
SWO Settings
Prescale Core Clk: 11
Clock: 6.545454 MHz
[✓] Autodetect max SWO Cl
Instruction Trace
PC Sampling
rescaler: 64*1
[ ] Perioderiod: <Disabled>
[ ] on Data R/W Sample
Instruction Trace
[ ] Enable
Instruction Trace disabled
Timestamps
[✓] Enabl rescaler: 1
Trace Events
[ ] CPI: Cycles per Instruc
[ ] EXC: Exception overhead
[ ] SLEEP: Sleep Cycles
[ ] LSU: Load Store Unit Cy
[ ] FOLD: Folded Instructio
[✓] EXCTRC: Exception Traci
ITM Stimulus Ports
Enable: 0x00000001
Privilege: 0x00000000
31 Port 24 23 Port 16 15 Port 8 7 Port 0
Quick Help
Prescaler between Core Clock and Timestamp generation. If bandwidth issues arise, reduce the number of generated Timestamps by increasing the Prescaler value.

Figure 5. Keil virtual terminal window path

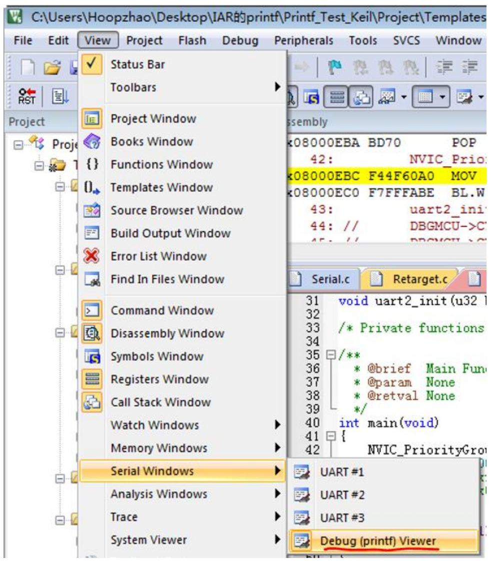
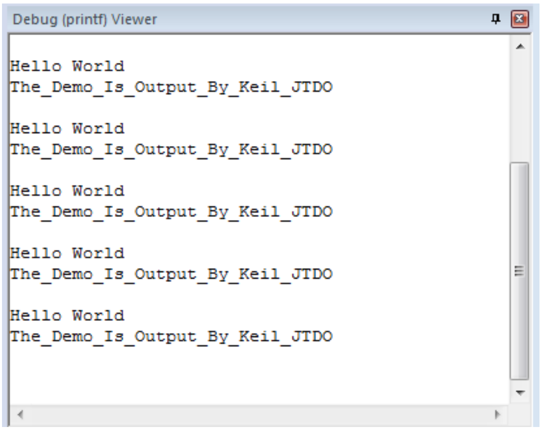


Figure 6. Keil virtual terminal interaction window



```
Hello World
The_Demo_Is_Output_By_Keil_JTDO

Hello World
The_Demo_Is_Output_By_Keil_JTDO

Hello World
The_Demo_Is_Output_By_Keil_JTDO

Hello World
The_Demo_Is_Output_By_Keil_JTDO

Hello World
The_Demo_Is_Output_By_Keil_JTDO
```

2.4 Redirect Printf as serial ports in Keil (using MicroLIB)

2.4.1 Introduction

The Keil environment comes with a MicroLIB that contains some codes supporting Printf function. Tick the MicroLIB option to implement information interaction through the serial port debugging assistant when the Printf is redirected to the serial ports.

2.4.2 Example code

004_Printf_Test_Keil_USART2_MicroLIB\project\mdk_v5

2.4.3 Environment and hardware

2.4.3.1 Environment

This method is used in the Keil environment. The compiling environment used in the example code is Keil_V5, with the hardware board AT-START-F403A_V1.2.

2.4.3.2 Hardware connection

2.4.3.2.1 J-Link/AT-Link&... connection

Table 6. Hardware connection (without JTDO)

Hardware connection (without JTDO)			
No.	**AT-START-F403A_V1.2**	**J-Link/AT-Link&…**	**Attention**
1	3.3V	3.3V	None
2	PA13	SWDIO	Must Pull up external
3	PA14	SWCLK	Must Pull down external
4	NRST	RSTn	None
5	GND	GND	None

2.4.3.2.2 USART2 connection

Table 7. Hardware connection (USART)

Hardware connection (USART)			
No.	**AT-START-F403_V1.2**	**USB_To_TTL(CH340)**	**Attention**
1	GND	GMD	None
2	PA2	RXD	None
3	PA3	TXD	None

2.4.4 Software

2.4.4.1 Header files

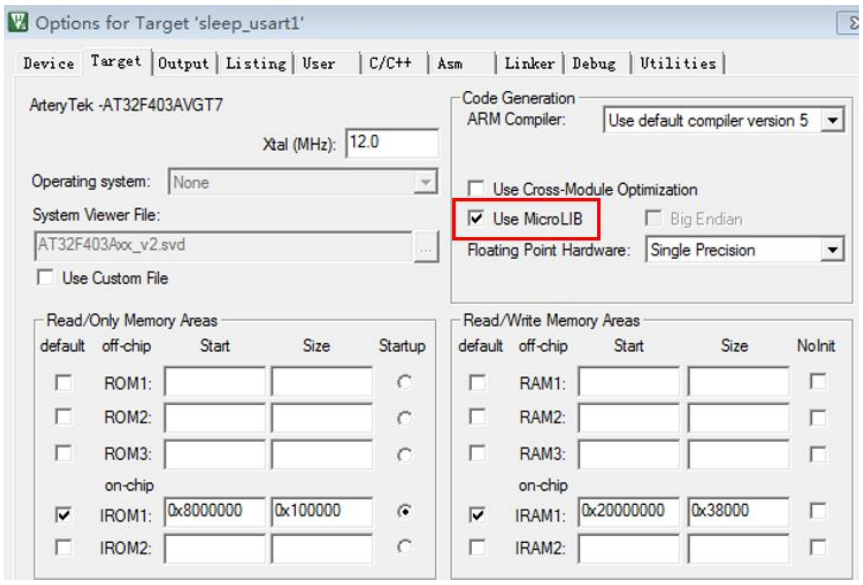
Add the “stdio.h” to the code project files.

2.4.4.2 Redirect Printf

```
PUTCHAR_PROTOTYPE
{
   USART_SendData( USART2, ch);
   while ( USART_GetFlagStatus( USART2, USART_FLAG_TRAC) == RESET );
   return ch;
}
```

2.4.4.3 MicroLIB settings

Figure 7. MicroLIB settings



Options for Target 'sleep_usart1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

ArteryTek -AT32F403AVGT7

Xtal (MHz): 12.0

Operating system: None

System Viewer File: AT32F403Axx_v2.svd

☐ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

☐ Use Cross-Module Optimization

☑ Use MicroLIB ☐ Big Endian

Floating Point Hardware: Single Precision

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x100000	●
☐	IROM2:			○

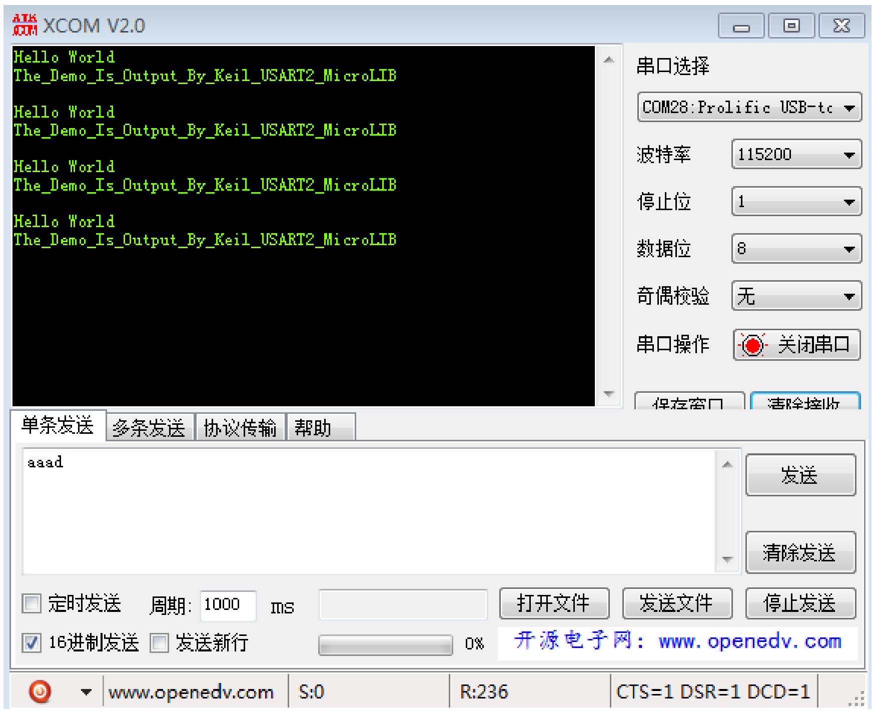
Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x20000000	0x38000	☐
☐	IRAM2:			☐

2.4.5 Debug and download

Compile the code and download to the MCU; then run the code at full speed, and you can see that “Hello World” is displayed in the terminal interaction window (Figure 8).

Figure 8. Serial port interaction window



```
Hello World
The_Demo_Is_Output_By_Keil_USART2_MicroLIB

Hello World
The_Demo_Is_Output_By_Keil_USART2_MicroLIB

Hello World
The_Demo_Is_Output_By_Keil_USART2_MicroLIB

Hello World
The_Demo_Is_Output_By_Keil_USART2_MicroLIB
```

XCOM V2.0

串口选择: COM28:Prolific USB-tc

波特率: 115200

停止位: 1

数据位: 8

奇偶校验: 无

串口操作: 关闭串口

aaad

定时发送 周期: 1000 ms

[✓ 16进制发送] 发送新行 0%

开源电子网: www.openedv.com

www.openedv.com S:0 R:236 CTS=1 DSR=1 DCD=1

2.5 Redirect Printf as serial ports in Keil (not use MicroLIB)

2.5.1 Introduction

The Keil environment comes with a MicroLIB that contains some codes supporting Printf function. When the Printf is redirected to the serial ports and the MicroLIB option box is not ticked, the information interaction can be done through the serial port debugging assistant after adding the codes that supports Printf to the project files.

2.5.2 Example code

005_Printf_Test_Keil_USART2__Without_MicroLIB\project\mdk_v5

2.5.3 Environment and hardware

2.5.3.1 Environment

This method is used in the Keil environment. The compiling environment used in the example code is Keil_V5, with the hardware board AT-START-F403A_V1.2.

2.5.3.2 Hardware connection

2.5.3.2.1 J-Link/AT-Link&... connection

Table 8. Hardware connection (without JTDO)

Hardware connection (without JTDO)			
No.	**AT-START-F403A_V1.2**	**J-Link/AT-Link&…**	**Attention**
1	3.3V	3.3V	None
2	PA13	SWDIO	Must Pull up external
3	PA14	SWCLK	Must Pull down external
4	NRST	RSTn	None
5	GND	GND	None

2.5.3.2.2 USART2 connection

Table 9. Hardware connection (USART)

Hardware connection (USART)			
No.	**AT-START-F403A_V1.2**	**USB_To_TTL(CH340)**	**Attention**
1	GND	GMD	None
2	PA2	RXD	None
3	PA3	TXD	None

2.5.4 Software

2.5.4.1 Header files

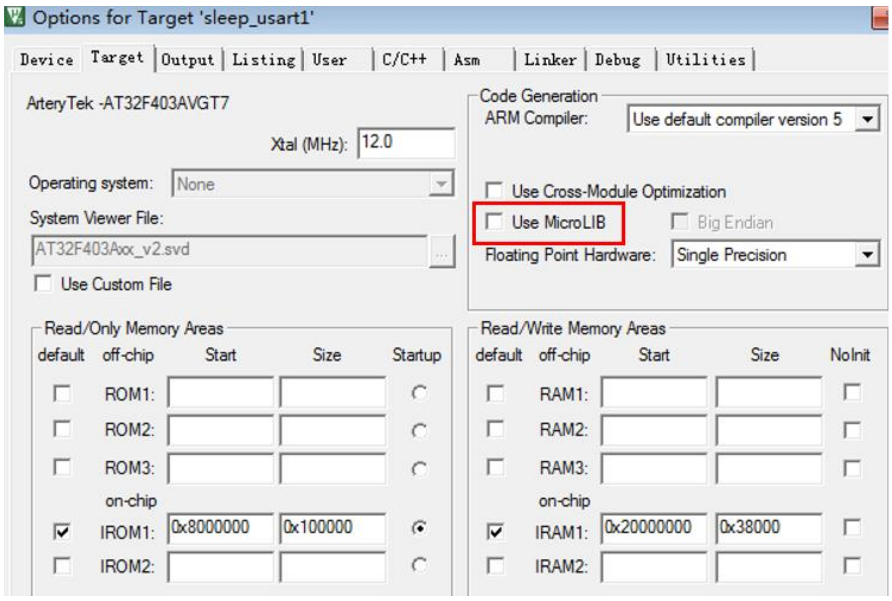
Add the “stdio.h” to the code project files.

2.5.4.2 Redirect Printf

```
PUTCHAR_PROTOTYPE
{
    USART_SendData( USART2, ch);
    while ( USART_GetFlagStatus( USART2, USART_FLAG_TRAC) == RESET );
    return ch;
}
```

2.5.4.3 MicroLIB settings

Figure 9. MicroLIB settings



Options for Target 'sleep_usart1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

ArteryTek -AT32F403AVGT7

Xtal (MHz): 12.0

Operating system: None

System Viewer File: AT32F403Axx_v2.svd

☐ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

☐ Use Cross-Module Optimization

☐ Use MicroLIB ☐ Big Endian

Floating Point Hardware: Single Precision

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
✓	IROM1:	0x8000000	0x100000	●
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
✓	IRAM1:	0x20000000	0x38000	☐
☐	IRAM2:			☐

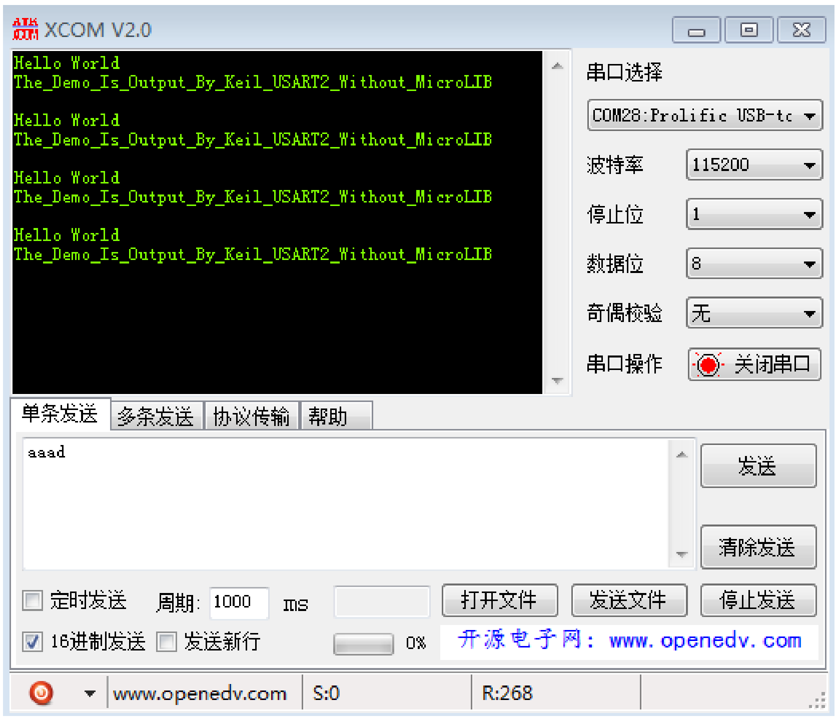
2.5.4.4 Add codes supporting Printf function

```
#if (__ARMCC_VERSION > 6000000)
    __asm (".global __use_no_semihosting\n\t");
    void _sys_exit(int x)
    {
        x = x;
    }
    /* __use_no_semihosting was requested, but _ttywrch was */
    void _ttywrch(int ch)
    {
        ch = ch;
    }
    FILE __stdout;
#else
  #ifdef __CC_ARM
    #pragma import(__use_no_semihosting)
    struct __FILE
    {
      int handle;
    };
    FILE __stdout;
    void _sys_exit(int x)
    {
      x = x;
    }
  #endif
#endif#if (__ARMCC_VERSION > 6000000)
    __asm (".global __use_no_semihosting\n\t");
    void _sys_exit(int x)
    {
      x = x;
    }
    /* __use_no_semihosting was requested, but _ttywrch was */
    void _ttywrch(int ch)
    {
      ch = ch;
    }
    FILE __stdout;
#else
  #ifdef __CC_ARM
    #pragma import(__use_no_semihosting)
    struct __FILE
    {
      int handle;
    };
    FILE __stdout;
    void _sys_exit(int x)
    {
      x = x;
    }
  #endif
#endif
```

2.5.5 Debug and download

Compile the code and download to the MCU; then run the code at full speed, you can find that the “Hello World” is displayed in the terminal interaction window (Figure 10).

Figure 10. Serial port interaction window



```
Hello World
The_Demo_Is_Output_By_Keil_USART2_Without_MicroLIB

Hello World
The_Demo_Is_Output_By_Keil_USART2_Without_MicroLIB

Hello World
The_Demo_Is_Output_By_Keil_USART2_Without_MicroLIB

Hello World
The_Demo_Is_Output_By_Keil_USART2_Without_MicroLIB
```

串口选择: COM28:Prolific USB-tc
波特率: 115200
停止位: 1
数据位: 8
奇偶校验: 无
串口操作: 关闭串口

aaad

定时发送 周期: 1000 ms
☑ 16进制发送 ☐ 发送新行
开源电子网: www.openedv.com

www.openedv.com S:0 R:268

2.6 Printf via JLinkRTT

2.6.1 Introduction

JLink has its own debug output function that can debug the code and output to the corresponding window according to the specified instructions after the JLink RTT library code is added.

2.6.2 Example code

006_Printf_Test_Jlink_RTT\project\mdk_v5

2.6.3 Environment and hardware

2.6.3.1 Environment

This method is used in both IAR and Keil environment. The compiling environment used in the example code is IAR_V8 or Keil_V5, with the hardware board AT-START-F403A_V1.2.

2.6.3.2 Hardware connection

J-Link connection

Table 10. Hardware connection (without JTDO)

Hardware connection (without JTDO)			
No.	**AT-START-F403A_V1.2**	**J-Link**	**Attention**
1	3.3V	3.3V	None
2	PA13	SWDIO	Must Pull up external
3	PA14	SWCLK	Must Pull down external
4	NRST	RSTn	None
5	GND	GND	None

2.6.4 Software

2.6.4.1 Header file

Add the “stdio.h” to the code project files.

2.6.4.2 Add JLink RTT library code

- Add the SEGGER_RTT.c and SEGGER_RTT_printf.c in the JLink RTT library code to the project files;
- Add SEGGER_RTT_Syscalls_IAR.c or SEGGER_RTT_Syscalls_KEIL.c to the project files according to the compiling environment.

2.6.4.3 Output to PC

Call the SEGGER_RTT_WriteString or SEGGER_RTT_printf command from the code and output to PC.

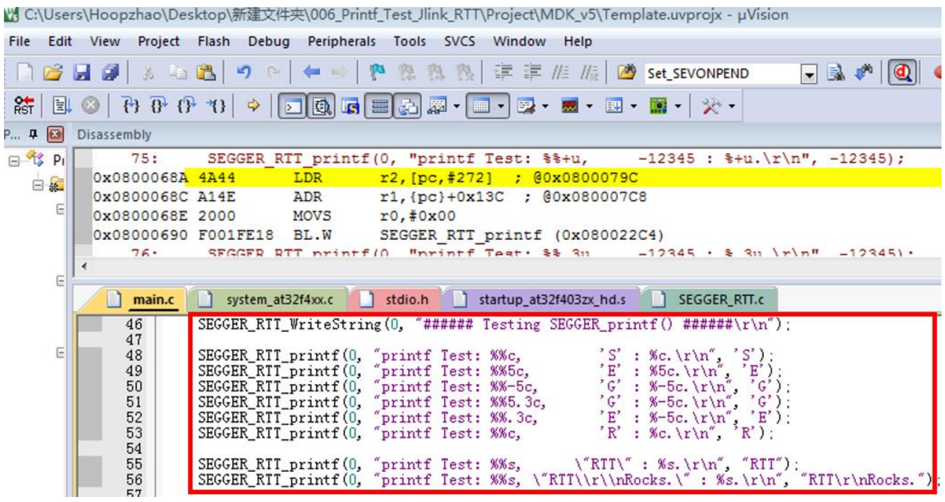
```
SEGGER_RTT_WriteString(0, "SEGGER Real-Time-Terminal Sample\r\n\r\n");
SEGGER_RTT_printf(0, "printf Test: %%c, 'S' : %c.\r\n", 'S');
```

2.6.5 Debug and download

2.6.5.1 Printf via JLinkRTTClient window

Compile the code and download to the MCU, and enter the debug environment, open the JLinkRTTClient application in the JLink installation path. Run the code step by step, and you can find the print information is output to the JLinkRTTClient window, as shown in Figure 11 and Figure 12.

Figure 11. Code debug

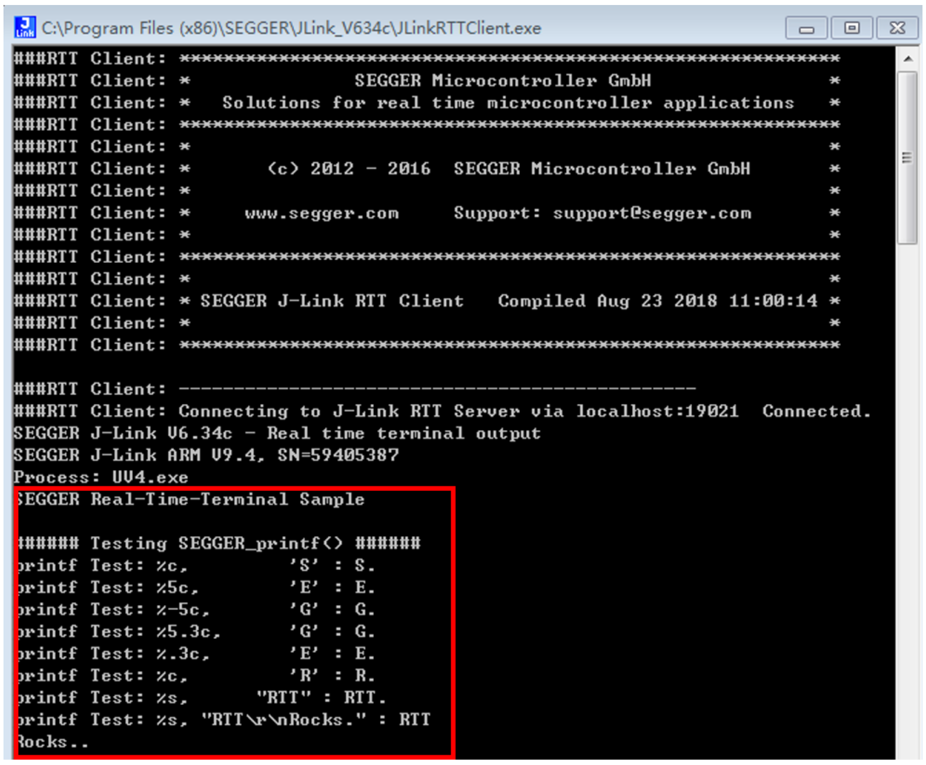


C:\Users\Hoopzhao\Desktop\新建文件夹\006_Printf_Test_Jlink_RTT\Project\MDK_v5\Template.uvprojx - µVision

```
    75:     SEGGER_RTT_printf(0, "printf Test: %%+u,     -12345 : %+u.\r\n", -12345);
0x0800068A 4A44      LDR      r2,[pc,#272]   ; @0x0800079C
0x0800068C A14E      ADR      r1,{pc}+0x13C  ; @0x080007C8
0x0800068E 2000      MOVS     r0,#0x00
0x08000690 F001FE18  BL.W     SEGGER_RTT_printf (0x080022C4)
```

```
SEGGER_RTT_WriteString(0, "###### Testing SEGGER_printf() ######\r\n");

SEGGER_RTT_printf(0, "printf Test: %%c,         'S' : %c.\r\n", 'S');
SEGGER_RTT_printf(0, "printf Test: %%5c,        'E' : %5c.\r\n", 'E');
SEGGER_RTT_printf(0, "printf Test: %%-5c,       'G' : %-5c.\r\n", 'G');
SEGGER_RTT_printf(0, "printf Test: %%5.3c,      'G' : %-5c.\r\n", 'G');
SEGGER_RTT_printf(0, "printf Test: %%.3c,       'E' : %-5c.\r\n", 'E');
SEGGER_RTT_printf(0, "printf Test: %%c,         'R' : %c.\r\n", 'R');

SEGGER_RTT_printf(0, "printf Test: %%s,      \"RTT\" : %s.\r\n", "RTT");
SEGGER_RTT_printf(0, "printf Test: %%s, \"RTT\\r\\nRocks.\" : %s.\r\n", "RTT\r\nRocks.");
```

Figure 12. JLinkRTTClient window output information

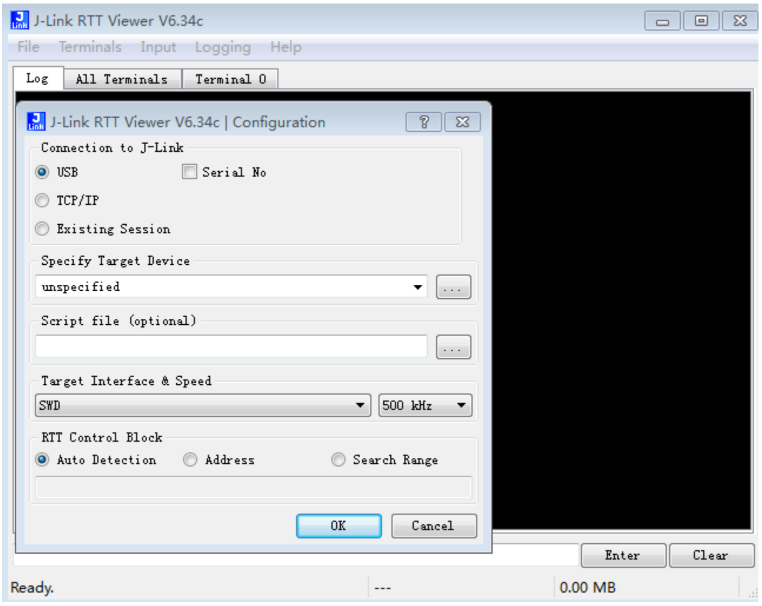


```
###RTT Client: ************************************************************
###RTT Client: *               SEGGER Microcontroller GmbH                *
###RTT Client: *   Solutions for real time microcontroller applications   *
###RTT Client: ************************************************************
###RTT Client: *                                                          *
###RTT Client: *       (c) 2012 - 2016  SEGGER Microcontroller GmbH       *
###RTT Client: *                                                          *
###RTT Client: *     www.segger.com     Support: support@segger.com       *
###RTT Client: *                                                          *
###RTT Client: ************************************************************
###RTT Client: *                                                          *
###RTT Client: * SEGGER J-Link RTT Client   Compiled Aug 23 2018 11:00:14 *
###RTT Client: *                                                          *
###RTT Client: ************************************************************

###RTT Client: -----------------------------------------------
###RTT Client: Connecting to J-Link RTT Server via localhost:19021  Connected.
SEGGER J-Link V6.34c - Real time terminal output
SEGGER J-Link ARM V9.4, SN=59405387
Process: UV4.exe
SEGGER Real-Time-Terminal Sample

###### Testing SEGGER_printf() ######
printf Test: %c,         'S' : S.
printf Test: %5c,        'E' : E.
printf Test: %-5c,       'G' : G.
printf Test: %5.3c,      'G' : G.
printf Test: %.3c,       'E' : E.
printf Test: %c,         'R' : R.
printf Test: %s,      "RTT" : RTT.
printf Test: %s, "RTT\r\nRocks." : RTT
Rocks..
```

2.6.5.2 Printf via JLinkRTTViewer window

1. Compile the code and download to the MCU, and then open the JLinkRTTViewer window, as shown in Figure 13.

Figure 13. Open JLinkRTTViewer window



J-Link RTT Viewer V6.34c | Configuration

Connection to J-Link: [✓ USB] Serial No / TCP/IP / Existing Session

Specify Target Device: unspecified

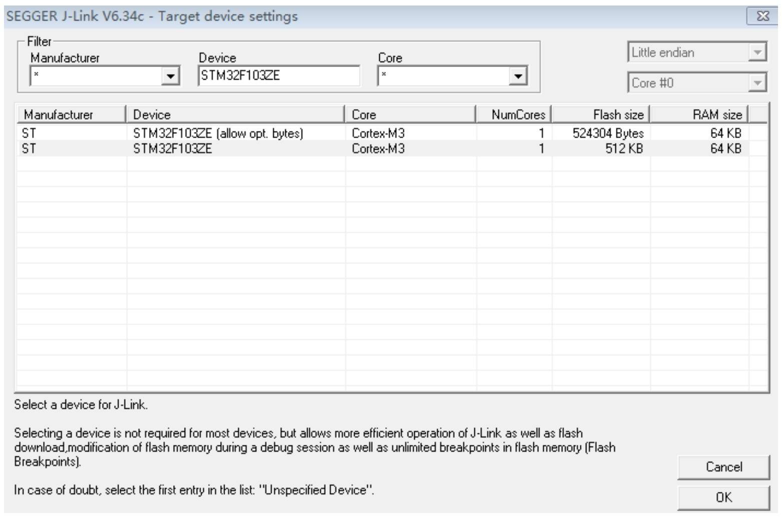
Script file (optional):

Target Interface & Speed: SWD, 500 kHz

RTT Control Block: [✓ Auto Detection] Address / Search Range

2. Click on "OK" and then click on "OK" again in the pop-up window; then, input and select the flowing information (taking ZE series as an example) and click on "OK", as shown in Figure 14.

Figure 14. Device selection window


SEGGER J-Link V6.34c - Target device settings

Filter: Manufacturer: * Device: STM32F103ZE Core: *

Little endian / Core #0

Manufacturer	Device	Core	NumCores	Flash size	RAM size
ST	STM32F103ZE (allow opt. bytes)	Cortex-M3	1	524304 Bytes	64 KB
ST	STM32F103ZE	Cortex-M3	1	512 KB	64 KB

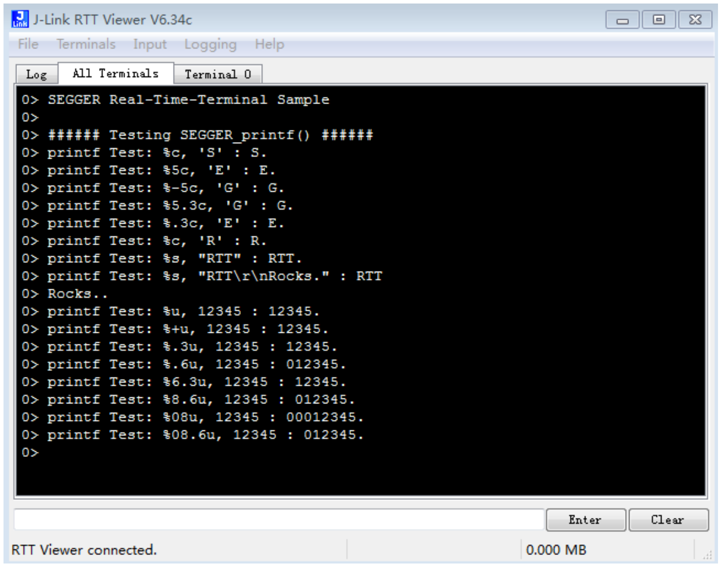
Select a device for J-Link.

Selecting a device is not required for most devices, but allows more efficient operation of J-Link as well as flash download,modification of flash memory during a debug session as well as unlimited breakpoints in flash memory (Flash Breakpoints).

In case of doubt, select the first entry in the list: "Unspecified Device".

3. Use the aforementioned code to enter the debug environment, and run the code step by step, and then the print information will be displayed in the JLinkRTTViewer window, as shown in Figure 15.

Figure 15. JLinkRTTViewer window output information



```
0> SEGGER Real-Time-Terminal Sample
0>
0> ###### Testing SEGGER_printf() ######
0> printf Test: %c, 'S' : S.
0> printf Test: %5c, 'E' : E.
0> printf Test: %-5c, 'G' : G.
0> printf Test: %5.3c, 'G' : G.
0> printf Test: %.3c, 'E' : E.
0> printf Test: %c, 'R' : R.
0> printf Test: %s, "RTT" : RTT.
0> printf Test: %s, "RTT\r\nRocks." : RTT
0> Rocks..
0> printf Test: %u, 12345 : 12345.
0> printf Test: %+u, 12345 : 12345.
0> printf Test: %.3u, 12345 : 12345.
0> printf Test: %.6u, 12345 : 012345.
0> printf Test: %6.3u, 12345 : 12345.
0> printf Test: %8.6u, 12345 : 012345.
0> printf Test: %08u, 12345 : 00012345.
0> printf Test: %08.6u, 12345 : 012345.
0>
```

RTT Viewer connected. 0.000 MB

3 Notes

- Only J-Link can be used for the testing in sections 2.3 and 2.6; AT-Link is not supported.
- For the testing in section 2.1 and 2.2, if AT-Link is used, in the Options→CMSIS DAP→Reset, either Hardware or System must be selected; otherwise, it will not be able to connect and download the code.
- For the testing in section 2.2, in the Options→General Options→Library Configuration→Library, Full must be selected; otherwise, it cannot be output. Only when the Full is selected, can the codes that support Printf be available.
- For the testing in section 2.6.5.1, the program in the output window cannot designate the device temporarily, so the ST part number have to be selected in the Device option at present, and the “Use MicroLIB” in “Options” must be ticked; otherwise, the code compiling may be abnormal.

4 Revision history

Table 11. Document revision history

Date	Version	Revision note
2021.12.07	2.0.0	Initial release