

# 使用指南

## SPC1068 ULINK2 使用指南

## 概述

本文主要介绍 ULINK2 调试适配器的使用。

注意: 本文档主要以 SPC1068 为例进行介绍。



# 目录

1	ULINK2 与 SPC1068 连接	7
2	KEIL 环境下 ULINK2 配置	9
3	KEIL 环境下使用 ULINK2 调试	14
3.1	单步调试	
3.2	断点设置	
3.3	观察变量值	
3.4	观察外设寄存器	
3.5	Memory 窗口	



# 图片列表

图	1-1:	ULINK2 适配器接口
图	1-2:	ARM 20-PIN 接口
图	1-3:	ULINK2 与 SPC1068 实物连接8
图	2-1:	Options for Target 对话框9
图	2-2:	Debug 配置界面9
图	2-3:	ULINK2 设置对话框10
图	2-4:	勾选 Run to main()选项运行结果11
图	2-5:	未勾选 Run to main()选项运行结果11
图	2-6:	未勾选 Run to main()选项执行至断点情形12
图	2-7:	Flash Download 设置12
图	2-8:	Add Flash Programming Algorithm13
图	2-9:	Build Output 窗口信息13
图	3-1:	Update Target before Debugging 设置14
图	3-2:	启动 Debug 后的界面15
图	3-3:	Run to Cursor Line 实现
图	3-4:	设置断点
图	3-5:	程序执行到断点
图	3-6:	添加变量到观察窗口18
图	3-7:	添加变量到观察窗口的结果19
图	3-8:	i=5 执行结果19
图	3-9:	i++执行结果20
图	3-10:	System Viewer File 设置界面21
图	3-11:	添加 PWM0 到 System Viewer 窗口22
图	3-12:	添加 PWM0 到 System Viewer 窗口的结果
图	3-13:	TBCTL=0 执行结果
图	3-14:	TBCTL=0x1234 执行结果
图	3-15:	打开 Memory 观察窗口
图	3-16:	Memory 窗口中观察到的 TBCTL 初始值
图	3-17:	Memory 窗口结果(TBCTL=0)25
图	3-18:	Memory 窗口结果(TBCTL=0x1234)25



# 表格列表

表 1-1:	标准 JTAG 接口信号定义	8
表 1-2:	ULINK2 与 SPC1068 管脚连接	8
表 3-1:	调试菜单和命令	15



# 版本历史

版本	日期	作者	状态	变更
C/0	2024-02-27	周佳莉	Released	首次发布。



# 术语或缩写

术语或缩写	描述
/	/



## 1 ULINK2 与 SPC1068 连接

ULINK2 适配器支持 5 种 JTAG 接口,如图 1-1 所示。其中,ARM 20-pin, 2.54mm 接口是用于 ARM 芯片调试的标准 JTAG 接口。该接口信号定义如图 1-2 和表 1-1 所示。



图 1-1: ULINK2 适配器接口

图 1-2: ARM 20-PIN 接口

ARI	M 20	)-PIN I	nte	iterface	
vcc	1		2	2 VCC (optional)	
TRST	3		4	4 GND	
TDI	5		6	6 GND	
TMS	7		8	8 GND	
TCLK	9		1	10 GND	
RTCK	11		1	12 GND	
тро	13		1	14 GND	
RESET	15		1	16 GND	
N/C	17		1	18 GND	
N/C	19		2	20 GND	



Signal	Connects to
TMS	Test Mode State pin — Use 100K Ohm pull-up resistor to VCC
TDO	Test Data Out pin
RTCK	JTAG Return Test Clock
TDI	Test Data In pin — Use 100K Ohm pull-up resistor to VCC
TRST	Test Reset/ pin — Use 100K Ohm pull-up resistor to VCC
TCLK	Test Clock pin — Use 100K Ohm pull-down resistor to GND
VCC	Positive Supply Voltage — Power supply for JTAG interface drivers
GND	Digital ground
RESET	RSTIN/ pin — Connect this pin to the (active low) reset input of the target
	CPU

## 表 1-1:标准 JTAG 接口信号定义

在采用 SPC1068 芯片进行应用开发的过程中,需要经常使用 ULINK2 进行程序的调试。ULINK2 与 SPC1068 的硬件连接如图 1-3 所示。表 1-2 中为具体的 PIN 脚连接关系。



## 图 1-3: ULINK2 与 SPC1068 实物连接

#### 表 1-2: ULINK2 与 SPC1068 管脚连接

ULINK2	SPC1068
TMS	GPIO16
TDO	GPIO17
RTCK	1
TDI	GPIO15
TRST	TRSTn
TCLK	GPIO18
VCC	VDD (+3.3V)
GND	GND
RESET	/



## 2 KEIL 环境下 ULINK2 配置

在安装 KEIL MDK 时,软件会默认安装 ULINK2 设备的驱动。按照表 1-1 将 ULINK2 与 SPC1068 连接,然后给芯片上电。这时打开 KEIL 软件,鼠标左键单击图标≤,弹出界 面如下:

图 2-1: Options for Target 对话框

•	t installed version	Use latest in	Generation — 1 Compiler:	Code (				ex-M3	ADMC-4
•	t installed version	Use latest in	Compiler:					0.00	ARM COR
				Aniv	2.0	Xtal (MHz):			
					•		None	system:	Operating
	zation	odule Optimiza	lse Cross-Mo	🗆 🗆 U				iewer File:	System V
	Big Endian	в Г	lse MicroLIB	□ □ U					
							e	Custom Fil	Use Use
		ory Areas	Write Memor	Read/			ry Areas	Only Memo	Read/0
Nolnit	Size	Start	off-chip	default	Startup	Size	Start	off-chip	default
			RAM1:		0			ROM1:	
			RAM2:		0			ROM2:	
			RAM3:		С			ROM3:	
			on-chip		_			on-chip	
	0x4000	0×20000000	IRAM1: 0	▼	۰	0x8000	0x1FFF8000	IROM1:	
			IRAM2:		0			IROM2:	
Help	0x4000	0x20000000	on-chip IRAM1: 0 IRAM2: 0 Defa	Cel	Can	0x8000	0x1FFF8000	on-chip IROM1: IROM2:	

选择 Debug 选项卡, 会看到如图 2-2 所示的界面。 红色矩形框标记的内容是 Debug 时需要设置的选项。

图 2-2: Debug 配置界面

C       Use Simulator       Settings
Image: Construction at Startup
Restore Debug Session Settings       Image: Construct of the set of th
Dialog DLL: Parameter: Dialog DLL: Parameter: Dialog DLL: Parameter:

图 2-2 所示界面中, 左侧是仿真调试相关的配置选项, 右侧则是与硬件调试相关的 选项。根据实际情形, 选择使用 ULINK2/ME Cortex Debugger 选项。单击 Settings 按钮, 会弹出与 ULINK2 相关的设置, 如图 2-3 所示。可以看到, 红色矩形框中出现 Debug targets 的信息, 表明 ULINK2 设备此时是正常工作的; 否则,则表明 ULINK2 设备不可 用。因此, 在用 ULINK2 调试程序时, 常常用此方法检查 ULINK2 设备是否正常。此外, 建议用户按照图 2-3 配置 Connect & Reset Options, Reset 方式选择 SYSRESETREQ。此外, SPC1068 芯片支持 JTAG 和 SWD 两种 Debug 协议, 用户可以根据需要进行配置。

ULINK USB - JTAG/SW Adapter -	JTAG Device Chain
Serial No: V0168AVR	IDCODE Device Name IR len Move
ULINK Version: ULINK2	DD Ox4BA00477 ARM CoreSight JTAG-DP 4
Device Family: Cortex-M	Down
Firmware Version: V2.03	Automatic Detection     ID CODE:
SWJ Port: JTAG	Manual Configuration     Device Name:
Max Clock: 1MHz	Add Delete Update IR len: AP: 0x00
Debug	
Connect: Normal TRes	set: SYSRESETREO V Cache Code
Reset after Connect	Cache Memory Download to Flash

图 2-3: ULINK2 设置对话框

接下来,在图 2-2 中,我们看到有两个选项: Load Application at Startup 和 Run to main()。其中 Load Application at Startup 选项是必须要勾选的,Run to main()选项根据需要决定要不要勾选。如果勾选 Run to main()选项,当启动 Debug 调试后,程序会直接 Run 到 main 函数的入口,如图 2-4 所示;相反,如果没有勾选 Run to main()选项,程序会停在 Boot Loader 程序的入口,如图 2-5 所示,此时在应用程序 main 中设置一个断点,单击 按钮或者按下 F5 键,程序就会快速执行到断点处,如图 2-6 所示。



Image:	<b>3</b>
Posisemby     7: Sys_Init();     7: Sys_Init();     9:     9osiFFR938 F7FFF93 BL.N Sys_Init (0xFFF885C)     9: CLOCK_INITWINHCO(CLOCK_HCLE_24M2);     10:     0xFFF83E 22000 NOVS r0,0x00     10:     manc	a (
7: Sys_Init(); 8: \$0:JFFFEBS0 FFFFEFS0 BLN Sys_Init (0x1FFF685C) 9: CLOCK_InitWithKO(CLOCK_HCLK_24MHZ); 10: 0x1FFFEBS1C 2000 MOV5 r0,40x00 C INFERSIC 2000 NOV5 R0 C INFERSIC 20	
Operifyressis fifterson BL.N Sys Int: (Valifyressi)         9       Clock_Int:WithRC0(Clock_HCLK_24MHZ);         10:       Operifyressic 2000         0:       France	
9: CLOCK_InitWithRCO(CLOCK_HCLK_24NHZ); 10: 0x1FFF8BiC 2000 MOVS r0.#0x00 Contemposite States of the states of	Ę
10: CALIFFEBIC 2000 NOVS 10,0000 (ALIFFEDICA)	
Imainc	
main.c	,
	<b>▼</b> ×
1 #include "spc1068.h"	
2 #include < <stdic.h></stdic.h>	
3	
5 int-main()	
e 甲 (	
7   Sys_Init();	1
9 CLOCK InitWithRCO(CLOCK HCLK 24MHZ);	
10	
11 ·· Delay_Init();	
14 ··/*·Set·GPIO·function·as·UART··*/	
15 • GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);	
16 GPIO_SetPinChannel (GPIO_35, GPIO35_UART_RXD);	
18 ···/*·Enable·UART·Clock·*/	
19 CLOCK_EnableModule(UART_MODULE);	-
(	Þ
a 🔝 Call Stack + Locals	# 🖲
2C1068\\Project\\Debug\\Objects\\PICA_FWLlb.axf" Name Location/Value Type	
*	
III F	
	<pre>int main() if () if</pre>

### 图 2-4: 勾选 Run to main()选项运行结果

## 图 2-5: 未勾选 Run to main()选项运行结果

	9 ℃ ← → 1 № 熟 熟 課 課 服 服 個	
Kar   E   O   O   O   O   O   O   O   O   O		
Registers	4 Disassembly	Ф <u>(</u>
Register Value	Cox0000030 4804 LDR r0.[pc,#16]; @0x00000044	· · · · · · · · · · · · · · · · · · ·
□         Care         0         0.00000000           □         1         0.00000000         0.00000000           □         2         0.00000000         0.00000000           □         3.0         0.00000000         0.00000000           □         5.0         0.00000000         0.00000000           □         5.0         0.00000000         0.00000000           □         1.0         0.00000000         0.00000000           □         1.1         0.00000000         0.00000000           □         1.1         0.00000000         0.00000000           □         1.1         0.00000000         0.00000000           □         1.1         0.00000000         0.00000000           □         1.1         0.00000000         0.00000000           □         1.1         0.00000000         0.00000000           □         1.0         0.000000000         0.00000000           □         1.0         0.00000000         0.00000000           □         1.0         0.00000000         0.00000000           □         1.0         0.000000000         0.000000000           □         1.0         0.000000000         0.00000	Deconcords4 4904 LDR r0.(pc,416) ; @0x000004@ Ox0000003 F7FE B Ox000003A Ox0000003 F7FE B Ox000003A Ox0000003 F7FE B Ox000003C Ox0000040 F7FE B Ox0000040	
🔚 Project 🛛 🚟 Registers	< III.	•
Command	a 🖻 Call Stack + Locals	<b>4</b> E
Load "D:\\SpintrolAE\\Her	gFang\\SPC1068\\Project\\Debug\\Objects\\PICA_FWLib.axf" Name Location/Value Type	
· [	н Т	
2 NGCICN BurnhDischle Burnh	Enable Brankfill Brankfist Brankforger COURDACE	



20000000000000000000000000000000000000		
Registers	Image:	<b># </b>
Register         Value           Corr         0.00007714           11         0.00000700           12         0.00000000           13         0.00000000           14         0.00000000           15         0.00000000           16         0.00000000           17         0.00000000           18         0.00000000           19         0.00000000           10         0.000000000           112         0.000000000           112         0.000000000           112         0.000000000           112         0.000000000           112         0.00000000           113         0.010000000           114         0.010000000           115         0.010000000           116         0.010000000           116         0.010000000           116         10.0000000           116         10.0000000           116         10.0000000           116         10.0000000           116         10.0000000           116         10.0000000           116         10.0000000           116         10.0000000	<pre>5: viffF8Bls PTFFF8Bls PTFFF8Bls viffF8Bls PTFFF8Bls PTFFF8Bls viffF8Bls PTFFF6ls BL.W Sys_Init (0xiFFF8Bls) viffF8Bls PTFF6ls BL.W CLOCK_HCLK_24MHZ); viffF8Bls PTFF6ls BL.W CLOCK_HCLK_24MHZ); viffels viffel</pre>	, , , , , , , , , , , , , , , , , , ,
E Project Registers	<pre>// /* Emable UMAT_CLock */ // Emable UMAT_CLock */ // /* Emable UMAT_CLock */ // // /* Emable UMAT_MODULE); // // // // // // // // // // // // //</pre>	*
Command	Gall Stack + Locals     Call Stack + Locals	* 🗵
<	Name Location/Value Type	

#### 图 2-6: 未勾选 Run to main()选项执行至断点情形

在使用 ULINK2 调试程序之前,还需要设置 Flash Download 选项,如图 2-7 所示。 其中,SPC1068 Programming Algorithm 可以通过点击 Add 按钮来添加,如图 2-8 所示。 (注意:需要将本目录下的 SPC1068.FLM 文件复制到 KEIL 软件安装路径下的目录 Keil\_v5\ARM\Flash\)

图 2-7: Flash Download 设置

Download Function C Erase Full Chip Erase Sectors C Do not Erase	<ul> <li>✓ Program</li> <li>✓ Verify</li> <li>☐ Reset and Run</li> </ul>	RAM for Algorithm Start: 0x200020	000 Size: 0x2000	
Programming Algorithm				
Description	Device Type	Device Size	Address Range	
SPC1068 48KB Flash	On-chip Flash	48k 1FF	F8000H - 20003FFFH	
		Start: 0x1FFF80	000 Size: 0x0000C000	
	Add	Remove		

Г



SN32F240 64KB User ROM         On-chip Flash         64k           SN32F700 32kB User ROM         On-chip Flash         32k           SN32F710 16kB User ROM         On-chip Flash         16k           SN32F720 8kB User ROM         On-chip Flash         16k           SN32F730 8kB User ROM         On-chip Flash         8k           SN32F740 16kB User ROM         On-chip Flash         8k           SN32F750 32kB User ROM         On-chip Flash         16k           SN32F750 32kB User ROM         On-chip Flash         32k           SN32F750 32kB User ROM         On-chip Flash         48k           SN32F760 64kB User ROM         On-chip Flash         64k           SPC1068 49kB Flash         On-chip Flash         128k           STM32F0xx 128kB Flash         On-chip Flash         16k           STM32F0xx 128kB Flash         On-chip Flash         16k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 44kB Flash         On-chip Flash         32k           STM32F0xx 64kB Flash         On-chip Flash         64k           STM32F0xx 16ak Options         On-chip Flash         64k	Description	Device Type	Device Size	
SN32F700 32kB User ROM         On-chip Flash         32k           SN32F710 16kB User ROM         On-chip Flash         16k           SN32F720 8kB User ROM         On-chip Flash         8k           SN32F730 8KB User ROM         On-chip Flash         8k           SN32F730 8KB User ROM         On-chip Flash         8k           SN32F750 32KB User ROM         On-chip Flash         16k           SN32F760 16KB User ROM         On-chip Flash         16k           SN32F760 64KB User ROM         On-chip Flash         64k           SPC1068 48KB Flash         On-chip Flash         64k           STM32F0xx 128kB Flash         On-chip Flash         128k           STM32F0xx 128kB Flash         On-chip Flash         16k           STM32F0xx 128kB Flash         On-chip Flash         32k           STM32F0xx 128kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 44kB Flash         On-chip Flash         32k           STM32F0xx 128kB Flash         On-chip Flash         32k           STM32F0xx 728kB Flash         On-chip Flash         64k           STM32F0xx 74kB Flash         On-chip Flash         64k           STM32F0xx 74kB Flash         On-chip	SN32F240 64KB User ROM	On-chip Flash	64k	
SN32F710 16kB User ROM         On-chip Flash         16k           SN32F720 8kB User ROM         On-chip Flash         8k           SN32F730 8KB User ROM         On-chip Flash         8k           SN32F740 16KB User ROM         On-chip Flash         16k           SN32F750 32KB User ROM         On-chip Flash         16k           SN32F760 16KB User ROM         On-chip Flash         32k           SN32F760 64KB User ROM         On-chip Flash         64k           SPC1068 48KB Flash         On-chip Flash         64k           STM32F0xx 128kB Flash         On-chip Flash         128k           STM32F0xx 128kB Flash         On-chip Flash         16k           STM32F0xx 128kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 44kB Flash         On-chip Flash         64k           STM32F0xx 54kB Flash         On-chip Flash         64k           STM32F0xx 74kB Flash         On-chip Flash         64k	SN32F700 32kB User ROM	On-chip Flash	32k	
SN32F720         8kB         User         On         Chip         Flash         8k           SN32F730         8KB         User         ROM         On-chip         Flash         8k           SN32F740         16KB         User         ROM         On-chip         Flash         16k           SN32F750         32KB         User         ROM         On-chip         Flash         32k           SN32F750         32KB         User         ROM         On-chip         Flash         32k           SN32F750         64KB         User         ROM         On-chip         Flash         64k           SPC1068         48KB         Flash         On-chip         Flash         64k           STM32F0x         128kB         Flash         On-chip         Flash         128k           STM32F0xx         128kB         Flash         On-chip         Flash         16k           STM32F0xx         32kB         Flash         On-chip         Flash         32k           STM32F0xx         32kB         Flash         On-chip         Flash         64k           STM32F0xx         48kB         Non-chip         Flash         64k         STM32F0x         Flash	SN32F710 16kB User ROM	On-chip Flash	16k	
SN32F730 8KB User ROM         On-chip Flash         8k           SN32F740 16KB User ROM         On-chip Flash         16k           SN32F750 32KB User ROM         On-chip Flash         32k           SN32F760 64KB User ROM         On-chip Flash         64k           SPC1068 48KB Flash         On-chip Flash         48k           STM32F0xx 128kB Flash         On-chip Flash         128k           STM32F0xx 128kB Flash         On-chip Flash         16k           STM32F0xx 32kB Flash         On-chip Flash         16k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         64k           STM32F0xx 32kB Flash         On-chip Flash         64k           STM32F0xx 32kB Flash         On-chip Flash         64k           STM32F0xx 32kB Flash         On-chip Flash         16k	SN32F720 8kB User ROM	On-chip Flash	8k	
SN32F740 16KB User ROM         On-chip Flash         16k           SN32F750 32KB User ROM         On-chip Flash         32k           SN32F760 64KB User ROM         On-chip Flash         64k           SPC1068 48KB Flash         On-chip Flash         64k           STM32F0xx 128kB Flash         On-chip Flash         128k           STM32F0xx 128kB Flash         On-chip Flash         16k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         64k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 32kB Flash         On-chip Flash         32k	SN32F730 8KB User ROM	On-chip Flash	8k	
SN32F750         32KB         User         ROM         On-chip         Flash         32k           SN32F760         64KB         User         ROM         On-chip         Flash         64k           SPC1068         48KB         Flash         On-chip         Flash         48k           STM32F0xx         128kB         Flash         On-chip         Flash         128k           STM32F0xx         128kB         Flash         On-chip         Flash         16k	SN32F740 16KB User ROM	On-chip Flash	16k	
SN32F760 64KB User ROM         On-chip Flash         64k           SPC1058 49KB Flash         On-chip Flash         48k           STM32F0xx 128kB Flash         On-chip Flash         128k           STM32F0xx 128kB Flash         On-chip Flash         16k	SN32F750 32KB User ROM	On-chip Flash	32k	
SPC1068 48KB Flash         On-chip Flash         48k           STM32F0xx 128kB Flash         On-chip Flash         128k           STM32F0xx 16kB Flash         On-chip Flash         16k           STM32F0xx 32kB Flash         On-chip Flash         32k           STM32F0xx 44kB Flash         On-chip Flash         32k           STM32F0xx 54kB Flash         On-chip Flash         64k           STM32F0xx 74kB flash         On-chip Flash         64k	SN32F760 64KB User ROM	On-chip Flash	64k	
STM32F0xx         128kB         Flash         On-chip         Flash         128k           STM32F0xx         16kB         Flash         On-chip         Flash         16k         Image: STM32F0xx         32k           STM32F0xx         32kB         Flash         On-chip         Flash         32k           STM32F0xx         64kB         Flash         On-chip         Flash         64k           STM32F0xx         Flash         On-chip         Flash         64k           STM32F0xx         Flash         On-chip         Flash         64k	SPC1068 48KB Flash	On-chip Flash	48k	
STM32F0xx         16kB         Flash         On-chip         Flash         16k            STM32F0xx         32kB         Flash         On-chip         Flash         32k           STM32F0xx         54kB         Flash         On-chip         Flash         32k           STM32F0xx         64kB         Flash         On-chip         Flash         64k           STM32F0xx         Flash         On-chip         Flash         16	STM32F0xx 128kB Flash	On-chip Flash	128k	_
STM32F0xx 32kB Flash On-chip Flash 32k STM32F0xx 64kB Flash On-chip Flash 64k STM32F0xx Flash Options On-chip Flash 16	STM32F0xx 16kB Flash	On-chip Flash	16k	
STM32F0xx 64kB Flash On-chip Flash 64k STM32F0xx Flash Options On-chip Flash 16	STM32F0xx 32kB Flash	On-chip Flash	32k	
STM32F0xx Flash Options On-chip Flash 16	STM32F0xx 64kB Flash	On-chip Flash	64k	
	STM32F0xx Flash Options	On-chip Flash	16	
STM32F10x XL-density Flash On-chip Flash 1M	STM32F10x XL-density Flash	On-chip Flash	1M	
STM32F10x Med-density Flash On-chip Flash 128k 🔻	STM32F10x Med-density Flash	On-chip Flash	128k	Ψ.

#### 图 2-8: Add Flash Programming Algorithm

Flash Download 设置完成之后,将应用程序编译,然后点击 KEIL 软件工具栏上的罩按钮,就可以将应用程序下载到芯片中。用户可以在 Build Output 窗口中查看具体的 Download 过程信息,如图 2-9 所示。

### 图 2-9: Build Output 窗口信息





## 3 KEIL 环境下使用 ULINK2 调试

根据前面的介绍,将 ULINK2 设备与 SPC1068 正确连接后,按照图 2-2、图 2-3 以及图 2-7 设置 Debug 的相关选项,就可以使用 ULINK2 设备调试程序了。使用 ULINK2 调试程序时,必须 保证 Flash 存储器中的程序与当前程序一致。这就需要用户每次修改代码后,都要点击罩按钮 将程序下载到 Flash 存储器中。值得一提的是,KEIL 软件提供了一个功能,可以自动上述动作, 如图 3-1 所示。用户只需勾选 Update Target before Debugging 选项,那么在每次启动 Debug 会 话时,KEIL 软件会自动通过 ULINK2 设备将程序下载到 Flash 中,从而保证了 Flash 中的程序与 当前调试的程序一致。

	+ Asm Linker Debug Utilities
Configure Flash Menu Command	
<ul> <li>Use Target Driver for Flash Programming</li> </ul>	Vse Debug Driver
Use Debug Driver	Settings Update Target before Debugging
Init File:	Edit
O Use External Tool for Flash Programming	
Command:	
Arguments:	
Run Independent	
- Configure Image File Processing (ECARM):	
configure integer no ricecearing (i or i inti).	Add Output File to Group:
Output File:	
Output File:	User
Output File: Image Files Root Folder:	User
Output File:	User  Generate Listing

图 3-1: Update Target before Debugging 设置

单击工具栏上的按钮 ④进入 Debug 状态,程序界面如图 3-2 所示。程序执行到 main 函数 入口处后停止,等待用户的进一步操作。此时,KEIL 软件的界面也发生了变化:除了用户源代 码窗口,还出现了汇编代码窗口和 CPU 寄存器窗口。在汇编代码窗口中,黄色底纹的汇编代码 对应于用户代码窗口中光标所在位置的 C 代码;此外,菜单栏上也出现了一些与 Debug 相关 的菜单选项,如表 3-1 所示。





## 图 3-2: 启动 Debug 后的界面

Debug Menu	Toolbar	Shortcut	Description
Start/Stop Debug Session	<b>(d)</b>	Ctrl+F5	Starts or stops a debugging session.
Reset CPU	RST		Sets the CPU to RESET state.
Run		F5	Continues executing the program until the next active breakpoint is reached.
Stop	8		Stops the program execution immediately.
Step	<del>[</del> +]	F11	Executes a single-step into a function; Executes the current instruction line.
Step Over	0	F10	Executes a single-step over a function.
Step Out	() <sup>1</sup>	Ctrl+F11	Finishes executing the current function and stops afterwards.
Run to Cursor Line	*{}	Ctrl+F10	Executes the program until the current cursor line is reached.
Show Next Statement	4		Shows the next executable statement/instruction.
Breakpoints		Ctrl+B	Opens the dialog Breakpoints.
Insert/Remove Breakpoint	٠	F9	Toggles the breakpoint on the current line.
Enable/Disable Breakpoint	0	Ctrl+F9	Enables/disables the breakpoint on the current line.
Disable All Breakpoints	8		Disables all breakpoints in the program.
Kill All Breakpoints	æ	Ctrl+Shift+F9	Removes all breakpoints in the program.

### 表 3-1: 调试菜单和命令



### 3.1 单步调试

单击工具栏上的 ④按钮后,程序进入 Debug 会话状态。此时单击工具栏上的 ⑦按钮或者按下快捷键 F10 就可以单步执行程序。在单步调试的时候,用户代码窗口左侧边框处有两个三角箭头: ▶表示当前光标所在的位置; ▶表示当前位置的代码为下一次要执行的语句。因此,可以通过这两个三角箭头快速判断程序执行到哪条语句以及光标的位置。

有时候,我们希望程序能够快速地执行到某个位置,再进行单步调试。这时我们可以将光标定位到该位置,然后单击工具栏上的制按钮,程序就会立即执行到当前光标处。该功能也可以通过单击鼠标右键,在弹出的快捷菜单中选择 Run to Cursor Line 实现,如图 3-3 所示。此外,我们也可以通过设置断点的方式来实现上述功能。



图 3-3: Run to Cursor Line 实现



## 3.2 断点设置

当启动 Debug 会话后,在用户代码所在行左侧边框处单击鼠标左键,即可快速地设置断 点,此时左侧边框会出现一个红色的圆形标记,如图 3-4 所示。当然,也可以通过工具栏上的 ●按钮设置断点,具体做法是:将光标定位到欲设置断点的代码行,然后单击●按钮,即可设 置断点。此时,单击工具栏上的 按钮或者按下快捷键 F5,程序就会执行起来,一直执行到断 点处停下来,如图 3-5 所示。



图 3-4: 设置断点

#### 图 3-5: 程序执行到断点

	main.c	<b>▼</b> ×
	1 #include "spc1068.h"	A
	<pre>2 #include <stdio.h></stdio.h></pre>	
	3	
	4	
	5 int main()	
	6 🖵 {	-
	7 Sys_Init();	-
	8	
	9 CLOCK_InitWithRCO(CLOCK_H	CLK_24MHZ);
	10	
	<pre>11 Delay_Init();</pre>	
	12	
	14 /* Set GPIO function as 0	ART ···*/
NN	15 GPIO_SetPinchannel(GPIO_3	4, GPI034_UARI_IXD);
	16 GPIO_SetPinchannel(GPIO_3	5, GPI035_UART_RXD);
	10 (* Frable UNDT Clark */	
	10 CLOCK EnableMedule (UNPT M	
	IS CLOCK ENADIEMODULE (UARI M	DDDLE);

设置断点除了可以让程序快速的执行到某个位置外,在 Debug 程序时也非常有用。例如,可以在中断服务函数中设置断点,用来判断相应的中断是否发生。



## 3.3 观察变量值

在调试程序的时候,常常需要观察变量的值。这个可以通过 KEIL 软件提供的 Watch Window 来实现。具体实现过程如下:将光标定位到要观察的变量(光标移到变量名左侧),单击鼠标 右键,在弹出的快捷菜单中即可将变量添加到观察窗口中,如图 3-6 所示。

从图 3-6 可以看出,我们将变量 i 添加到观察窗口 Watch1 里,结果如图 3-7 所示。从图 中可以看出在代码区下方出现了 Watch1 窗口,在 Watch1 中可以看到刚刚添加的变量 i。



图 3-6: 添加变量到观察窗口



Image: Constant of the second sec	mgFang\SPC1068\Project\Debug1\Pro       Project     Flash     Debug     Peripherals     Td       Image: State of the	<pre>jectuvproj - µVision bols SVCS Window Help</pre>	4MHZ);		
Froject Regist	ters 19	} III		4	
Load "D:\\Spin WS 1, `i <	trolAE\\HengFang\\SFC1068\\F	Image: Second	Type unsigned int nory 1 ULINK2/ME Cortex Debugger 11:	0.24622170 sí	

图 3-7: 添加变量到观察窗口的结果

接下来,我们就通过单步执行观察变量 i 的值。

第一步:单步执行语句 i = 5,执行结果如图 3-8 所示,可以看出变量 i 的值变为 5; 第二步:单步执行语句 i++,执行结果如图 3-9 所示,可以看出变量 i 的值变为 6。

main.c			<b>▼</b> ×
11 12 13 14 15 16 17 18 19 20 21 ✓ Ⅲ	CLOCK_InitWithRCC i.=.5; i++; while(1);	(CLOCK_HCLK_24MHZ);	
Watch 1			д 📧
Name	Value	Туре	
i Enter express Call Stack + Log	0x00000005 ssion> ocals Watch 1 Memo	unsigned int	

图 3-8: i=5 执行结果



图 3-9: i++执行结果

main.c			▼ ×
11 12 ··· CL	OCK_InitWithRCO	(CLOCK_HCLK_24MHZ);	*
14 ··i· 15 ·· 16 ··i+	=·5; +;		
17 17 18 19 20 21	ile(1);		E .
•			F
Watch 1			Ф 💽
Name	Value	Туре	
i Enter expression	0x00000006	unsigned int	
Call Stack + Loca	s Watch 1 Memo	ry 1	



## 3.4 观察外设寄存器

在调试程序的时候,我们不仅需要观察变量的值,也需要查看芯片外设 Register 的值。本节以芯片 SPC1068 外设模块 PWM0 为例介绍实现过程。

- 1. 通过 KEIL 软件添加芯片 System Viewer File。单击图标系,在弹出的界面中勾选 Use Custom File 选项,然后单击图标,在弹出的对话框中选中芯片的 System Viewer File。设置结果如 图 3-10 所示。
- 2. 单击 ④按钮,进入 Debug 模式,将芯片外设 PWMO 添加到 System Viewer 窗口,如图 3-11 所示。添加后的结果如图 3-12 所示。从图 3-12 可以看出,在 System Viewer 窗口中,不 仅可以看到 PWMO 模块各个 Register 的值,而且还可以看到 Register 各个位段的值。

按照上面的步骤将 PWMO 添加到 System Viewer 窗口后,就可以在 Debug 程序的过程中观 察到 PWMO 各个寄存器的值。我们单步执行图 3-12 中 main 函数的程序,分别将 TBCTL 寄存 器赋值为 0 和 0x1234,结果分别如图 3-13 和图 3-14 所示。

Option	s for Tar	get 'FWLib'							
Device	Target (	Output   List	ing   User	C/C++   J	Asm  I	inker   I	)ebug   Utili	ities	
ARM Cort	ex-M3								
				2.0	-Code G	ieneration			
			Xtal (MHz):	2.0	ARM	Compiler:	Use latest in	stalled version	_
Operating	g system:	None		-					
System V	iewer File:				🗆 U:	se Cross-M	lodule Optimiza	tion	
.E\Heng	Fang\SPC	1068\Project\	Debug\SPC10	68.SFF	🗆 U:	se MicroLl	вГ	Big Endian	
I Use	Custom Fil	e							
⊢ Read/	Only Memo	nv Areas —			- Read∕\	Nrite Mem	orv Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit
	ROM1-			0		RAM1			
	DOM2			-		DAM2			
	RUM2:			0		RAM2:			
	ROM3:			0		RAM3:			
	on-chip			_		on-chip			_
	IROM1:	0x1FFF8000	0x8000	۲		IRAM1:	0x20000000	0x4000	
	IROM2:			0		IRAM2:			
		,					-		
			OK	Can	cel	Def	aults		Help

图 3-10: System Viewer File 设置界面







### 图 3-12: 添加 PWM0 到 System Viewer 窗口的结果





* * * *	Property TBPRD TBPHS TBCTR TBCTL FREE SOFT	Value           0           0           0           0           0           0	-
• • • ×	Property TBPRD TBPHS TBCTR TBCTL FREE SOFT	Value 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
+ + + ×	TBPRD     TBPRD     TBPHS     TBCTR     TBCTL     FREE     SOFT	Value 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
* * * X	TBPRD     TBPHS     TBCTR     TBCTL     FREE     SOFT	0 0 0 0x00000000	_
+ + × X		0 0 0x00000000	
+ • ×		0 0x00000000	
• x	TBCTL FREE SOFT	0x0000000	
▼ ×			
*	SOFT		
Î	SOFT		
	PHSDIR		
	TBCLKDIV	0x00	
	SWFSYNC		
	SYNCOSEL	0x00	
	DRDLD	<b>E</b>	
	PRULU	I	
	PHSEN	L	
E	CTRMODE	0x00	
		0x0000001	
		0	
	-CMPB	0	
	CMPCTI	0	
	трсті		
		1000700 C) TR CTI	
-	[BILS 31.0] KM (@ 0%	NOOVOC) IBCIL	
	E		TBCLKDIV       0x00         SWFSYNC       -         SYNCOSEL       0x00         PRDLD       -         PHSEN       -         CTRMODE       0x00         TBSTS       0x00000001         CMPA       0         CMPB       0         CMPCTI       0         TBCTL       (Bits 31.0) RW (@ 0x4000700C) TBCTL

## 图 3-13: TBCTL=0 执行结果

图 3-14: TBCTL=0x1234 执行结果

<b>₽ ⊠</b>	Property	▼ Value	д 🖪
<b>^</b>	Property	▼ Value	
	Property	Value	
		value	
	TRPRD		_
		0	<b>^</b>
-	TBPHS	0	
		0	
	B TBCTL	0x00001234	
<b>▼</b> ×	FREE		
*	SOFT		
	PHSDIR		
	TBCLKDIV	0x04	
	SWFSYNC		
	SYNCOSEL	0x03	
	PRDLD		
	PHSEN	<b>v</b>	
E	CTRMODE	0x00	
	<b>TBSTS</b>	0x0000001	
	+ CMPA	0	
	-CMPB	0	
		0	
	твсті		
	[Bits 31, 0] RW (@ 0v4	000700C) TRCTI	
*	[513 5110] 111 (@ 014	our out for the feature	
	× X		<ul> <li></li></ul>



## 3.5 Memory 窗口

在 Debug 程序的过程中,我们还可以通过 Memory 窗口观察芯片内任一存储单元的地址。 我们以章节 3.4 中的程序为例,其中 SPC1068 芯片 PWM0 模块 TBCTL 寄存器的地址为 0x4000700C。首先,打开一个 Memory 观察窗口(Memory1),如图 3-15 所示。



图 3-15: 打开 Memory 观察窗口

在 Memory1 窗口中输入地址 0x4000700C 后回车,结果如下:

图 3-16: Memory 窗口中观察到的 TBCTL 初始值

0x1FFF89C6       7:       Sys_Init():         3:       Property       Value         0x1FFF89C8       F7FFFDE BL.W       Sys_Init()x1FFF8789)         9:       CLOCK_InitWithRC0(CLOCK_HCLK_24MHZ);       TBCTR         1       #include "spcl068.h"       *         2       #include "spcl068.h"       *         3       4       *         4       5       int main()         6       (       SOFT         9       ·CLOCK_InitWithRC0(CLOCK_HCLK_24MHZ);       -         11       ·CLOCK_InitWithRC0(CLOCK_HCLK_24MHZ);       -         12       #include 'spcl068.h"       -         3       -       -       -         4       5       int main()       -         6       (       -       -         11       ·CLOCK_InitWithRC0(CLOCK_HCLK_24MHZ);       -       -         11       ·CLOCK_EnableFWMModule(FWM0);       -       -         12       ·FWNO->TBCTL.all = 0;       -       -         14       ·FWNO->TBCTL.all = 0x1234;       -       -         16       ·       -       -       -         14       ·       -       -       -	Disassembly 🛛 🖡 🔯	PWM0 🗜 🕅
<pre>8:</pre>	0x1FFF89C6         E7FE         B         0x1FFF89C6         A           7:         Sys_Init();         Init();         Init();	
15       PWM0->TBCTL.all.=.0x1234;         16      while(l);         17      while(l);         18      while(l);         16      while(l);         17      while(l);         18      while(l);         19      while(l);         10      while(l);         11      while(l);         12      while(l);         13      while(l);         14      while(l);         15      while(l);         15      while(l);         16      while(l);         17      while(l);         18      while(l);         19      while(l);         10      while(l);         11      while(l);         12      while(l);         13      while(l);         14      while(l);	<pre>7: Sys_Init(); 8: 0x1FFF89C8 F7FFFEDE BL.W Sys_Init (0x1FFF8788) 9: CLOCK_InitWithRCO(CLOCK_HCLK_24MHZ); 1</pre>	Property         Value           ■ TBPRD         0         ▲           ■ TBPHS         0         ▲           ■ TBCTR         0         ▲           ■ TBCTR         0         ▲           ■ TBCTR         0         ▲           ■ FREE         □         →           ■ FREE         □         →           ■ FREI         □         →           ■ SOFT         □         →           ■ FRELKDIV         0x00         →           ■ SWFSYNC         □         →           ■ SYNCOSEL         0x00         →           ■ PRDLD         □         →           ■ TBSTS         0x00000001         ●           ● CMPA         0         ●
Memory 1         Address:         0x4000700C         0 <th>15 ··· PWMO-&gt;TBCTL.all ·=· 0x1234; 16 ··· 17 ···while(1); 18 }</th> <th>⊕ CMPB 0 ▼ <b>TBCTL</b> [Bits 310] RW (@ 0x4000700C) TBCTL</th>	15 ··· PWMO->TBCTL.all ·=· 0x1234; 16 ··· 17 ···while(1); 18 }	⊕ CMPB 0 ▼ <b>TBCTL</b> [Bits 310] RW (@ 0x4000700C) TBCTL
Address: [0x4000700C 0x4000700C: 03 00 00 01 00 00 01 00 00 00 00 00 00 00	Memory 1	<b>⊅</b> ⊠
0x4000700C:       03       00       00       01       00	Address: 0x4000700C	^∩ ≜
	0x4000700C:       03       00       00       01       00	00         00<



分别单步执行图 3-16 中 main 函数的程序,分别将 TBCTL 寄存器赋值为 0 和 0x1234,结 果分别如图 3-17 和图 3-18 所示。

Disassembly Ox1FFF89DC 60C8 STR r0, [r1, #0x0C] 15: FWM0->TBCTL.all = 0x1234; 16: <pre></pre>	A     PWM0     A       Property     Value       B     TBPRD       C     TBPHS       0     B       TBCTL     0.00000000       FREE     -       -SOFT     -       -PHSDIR     -       -SVRCOSEL     0x00       -PHSEN     -       CTRMODE     0x00       B     -       CTRMODE     0x00       B     CMPB       CMPB     0       TBCTL     [Bits 310] RW (@ 0x4000700C) TBCTL
Memory 1	•
Address: 0x4000700C	∎ Â
0x4000700C:         00         00         00         01         00	

图 3-17: Memory 窗口结果(TBCTL=0)

## 图 3-18: Memory 窗口结果(TBCTL=0x1234)

Disassembly	д 🔝	PWMO 🕈 🖬
0x1FFF89E2 60C8 STR r0,[r1,#0x0C] 17: while(1);		
C×1FFF89E4 BF00 NOP		Property Value
0x1FFF89E6 E7FE B 0x1FFF89E6		TBPRD 0 ▲
0x1FFF89E8 7000 DCW 0x7000	-	TBPHS 0
	•	■ TBCTR 0
main.c ) pga.c ) pga.h ) spc1068_reg.h	<b>▼</b> ×	□ TBCTL 0x00001234
3		FREE
4		SOFT
5 int main()		PHSDIR
0 ⊟i 7 L. Sug Thit():		TBCLKDIV 0x04
8		SWFSYNC
9 CLOCK InitWithRCO (CLOCK HCLK 24MHZ);		SYNCOSEL 0x03
10		PRDI D
11 CLOCK_EnablePWMModule(PWM0);		
	E	
13 ···PWMU->IBCIL.all·=·0;		
15 PWM0->TBCTL.all = 0x1234;		CMDA 0
16		
17 · while(1);		
18 }		TBCTL
19 -	-	[Bits 310] RW (@ 0x4000700C) TBCTL
<	- F	
Memory 1		<b>4</b> 🔀
Address: 0x4000700C		
0x4000700C: 34 12 00 00 01 00 00 00 00 00 00 00 00 00	00 00	
0x40007025: 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00	00 00 00 00 00 00 00 00 00 00
0x4000703E: 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00	00 00 00 00 00 00 00 00 00 00
	00 00	00 00 00 00 00 00 00 00 00 00
0x40007057: 00 00 00 00 00 00 00 00 00 00 00 00 00		
0x40007057: 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00	- 00 00 00 00 00 00 00 00 -