

旋智有限公司 Spintrol Limited Co.

SPINTROL 软件调试 使用指南

V01-20200311

上海市浦东新区龙东大道 3000 号张江集电港 4 号楼 601 室 Room 601, Building 4, 3000 Long dong Avenue, Pudong District, Shanghai Tel: 86-21-58212991 E-mail: info@spintrol.com





SPINTROL 软件调试 使用指南	1
目录	2
1. KEIL 工具配置导引	4
1.1. 版本检测	4
1.2. 打开项目文件	
1.3. 打开 OPTIONAL 配置	
1.4. OPTIONAL 配置 DEVICE 选项	
1.5. OPTIONAL 配置 TARGET 选项	6
1.6. Optional 配置 Output 选项	8
1.7. Optional 配置 Listing 选项	8
1.8. Optional 配置 Listing 选项	9
1.9. OPTIONAL 配置 DEBUG 选项	9
1.10. OPTIONAL 配置 DEBUG 的 FLASH 算法选项	11
1.11. 保存配置	
1.12. 编译结果	
1.13. 下载程序	
1.14. 调试程序	
1.15. 仿真注意事项	
2. JLINK/ULINK2 工具使用指南	15
2.1. 调试工具与开发板 JTAG 连接	
2.2. 硬件管脚 JTAG/ULINK/JLNK 配置	
2.3. KEIL 环境下 JLINK 配置	
2.4. 配置 ALGORITHM	24
2.5. KEIL 环境下使用 ULINK2/JLNK 调试	
1) 单步调试	
2) 断点设置	
3) 观察变量值	31
4) 观察外设寄存器值	34
5) Memory 窗口	
3. J-FLASH 烧录下载指南	
31 基本要求	41
3.2 更新 FIM 文件	Δ1
33 更新 SPINTROI 产品信息	
3.4. 用 J-FI ASH 软件烧录 HEX 文件了	43
1) 运行 JFLASH.EXE,新建一个工程,选择要烧录的芯片	43
2) 选择要下载的 Hex 文件, File ->OPEN DATA FILE	
3) 烧录芯片, TARGET -> PRODUCTION PROGRAMMING	
4. I>Y 甲凵丄,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	45

©2014-2025, Spintrol Limited Corporation

Page 2



4.1. 工具栏	
4.2. 状态栏	
4.3. PROGRAM 文件	
4.4. 代码信息	48
4.5. LOG 窗口	
4.6. 下载程序	
1) 硬件 ISP 模式选择	
2) SPINTROL ISP TOOL 配置	
4.7. SECURITY 功能	
4.8. 代码加密切能	
4.9. UART 迪信父互	
5. 蓝牙调试	
51 蓝牙模块使用	55
5.2. 转接板	
5.3. 波特率设置	
5.4. 配置软件波特率与蓝牙波特率一致	
5.5. 添加蓝牙串口进行串口下载	
6. 放此框如	59
9. 次门 框未	
7. 数据采集功能	
8. 电机参数	
9. 电机保护功能	
10. 开环电流采集测试	
11. 开环算法适用验证	
12. 闭环调试	
13. 电机启动调试	
14. 串口指令集	
15. MEMDUMP 工具使用	71
15.1	71
15.1. 相大工兵的国仇 15.2 再新 MiniSPI iniz 的国仇	
15.2.	73
1) 接线如下	73
2) MEMDUMP 配置	74
3) 连接串口	
4) 复位 MCU	
, 5) 点击 READ,数据读取完毕。	
6) 保存 HEX 文件	
16. 持续更新中	



1. Keil 工具配置导引

1.1. 版本检测

建议安装 Keil 5 之后的版本

About µVision	dit View Project Flash C dit View Project F	Debug Peripherals Tools SVCS Windon Peripherals Tools SVCS Windon State of the second secon	Heip ♥ µVision Help ♥ Open Books Winkow Simulated Periphenis for 'Cortex-M3' Contact Support Check for Update About µVision GmbH. All rights reserved	d.
Toolchain: Toolchain Path: C Compiler: Assembler: Linker/Locator: Library Manager Hex Converter: CPU DLL: Dialog DLL: Target DLL: Dialog DLL:	MDK-/ C:\Keil_v5\ARM Armcc.exe Armasm.exe ArmLink.exe ArmAr.exe FromElf.exe SARM DCM.DLL Segger\JL2CM TCM.DLL	ARM Professional Versio V5.04 update V5.04 update V5.11.0.0	n: 5.11.0.0 1 (build 49) 1 (build 49) 1 (build 49) 1 (build 49) 1 (build 49) 1 (build 49)	

1.2. 打开项目文件

点击*.proj 格式的文件,就可以打开项目工程



新增資料夾		0
名稱	修改日期	類型
🚴 Арр	2017/5/16 下午 0	檔案資料夾
🔊 del	2015/10/9 下午 0	Windows 批3
🔊 isr	2016/11/26下午	C檔案
 isr.h 	2016/11/26下午	VisualStudio.
🛃 main	2017/5/16 下午 1	C檔案
Project.uvgui_Maxwell_SSD.bak	2017/5/16 下午 0	BAK 檔案
🛃 Project	2017/5/16 上午 1	猩ision4 Proj
 Project_FWLib.dep 	2017/5/16 下午 1	DEP 檔案
Project_uvopt.bak	2017/5/16 上午 1	BAK 檔案
Project_uvproj.bak	2016/5/23 下午 0	BAK 檔案

1.3. 打开 Optional 配置

请点开魔法棒,开启 Optional 设定窗口

1 🗃 🖬 🖉 3 🖧 🛍 🖌 e	
🖹 🎬 🧼 🔜 🛛 🛒 🛛 FWLib	
ject	🕈 🔯 🔚 spc=068.h 📄 global.h 🖆 main.c 📑 isr.c 🖃 motor_hardw
Periph_Driver	Options for Target 'FWLib'
u adc.c	Device Target Output Listing User C/C++ Asm Linker Debug Utilities
aes.c	Device Database 🗸
⊕ 🖹 gpio.c	Vendor: ARM
⊞– 🔛 i2c.c	Device: Cortex-M3
	Toolset ARM
timer.c	Search:



1.4. Optional 配置 Device 选项

请选择相应的芯片对应的 Device.

chip	Device
SPC1068	Cortex-M3 下的 ARMCM3
SPD1078	Cortex-M3 下的 ARMCM3
SPC1158	Cortex-M4 下的 ARMCM4_FP
SPC1168	Cortex-M4 下的 ARMCM4_FP
SPC1178	Cortex-M4 下的 ARMCM4 FP
/endor: ARM Device: ARMCM4_FP Foolset: ARM	Software Pack Pack: ARM.CMSIS.4.5.0 URL: http://www.keil.com/pack/
ARM Cortex M0 ARM Cortex M0 ARM Cortex M3 ARMCM3 ARMCM3 ARMCM4 ARMCM4 ARMCM4 ARMCM4_FF ARM Cortex M7 ARM SC000 ARM SC300	plus The Cortex-M4 processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including: - simple, easy-to-use programmers model - highly efficient ultra-low power operation - excellent code density - deterministic, high-performance interrupt handling - upward compatibility with the rest of the Cortex-M processor family.

1.5. Optional 配置 Target 选项

chip	System Viewer File
SPC1068	SPC1068.SFR
SPD1078	SPC1068.SFR
SPC1158	SPC1168.SFR
SPC1168	SPC1168.SFR
SPC1178	SPC1168.SFR
注: 文件一般放在此文件放	(置在 SDK 内的 Utilities 内,如 \SDK\SPC1068\Utilities\SPC1068.SFR



chip	IROM1	SIZE	IRAM1	SIZE	IRAM2	SIZE
SPC1068	0x1FFF8000	0x8000	0x20000000	0x4000		
SPD1078	0x1FFF8000	0x8000	0x20000000	0x4000		
SPC1158	0x10000000	0x10000	0x1FFFC000	0x8000		
SPC1168	0x10000000	0x20000	0x20000000	0x4000	0x1FFF4000	0xC000
SPC1178	0x10000000	0x20000	0x20000000	0x4000	0x1FFF4000	0xC000
详细请参考 Tec	chnical Reference	Manual 里面的	Memory map			

	reet C	utout Listing	Lilear 1	夾					
		athat Fismus	0361	名稱		^		修改日期	A
ARM Cone	CIVID		Xtal (MHz):	SPC:	1068.SFR			2016/5/	/23 下午 0.
Operating s	system:	None		_	1 00		ano opanieou		
System Vie	wer File:				I Us	MicroLIB	ſ	Big Endian	
C:\XXXXX	DK\SPC	1068\Utilities\SP	C1068.SFR	- ()	\rightarrow				
Use Cu	ustom File	3		-					
-Read/Or	ly Memo	rv Areas			ReadM	Vrite Memory	Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit
	ROM1:			с		RAM1:			
	ROM2:		1	c		RAM2:			
	ROM3:			с		RAM3:			
	on-chip	-			-	on-chip			
	IROM1:	0x1FFF8000	0x8000	(•)	(7	IRAM1:	x20000000	0x4000	- r)
				C	Г	IRAM2:			
	IROM2:								



1.6. Optional 配置 Output 选项

vice larget Output Listing User C/C++ Asm Link	ter Debug Utilities
Select Folder for Objects Name of Executa	able: Project
Create Executable: .\Objects\Project	
Debug Information	Create Batch File
Create HEX File	
✓ Browse Information	

1.7. Optional 配置 Listing 选项

ice larget Output Lisu	ng User C/C++ Asm	Linker Debug Utilities
Select Folder for Listings	Page	Width: 79 + Page Length: 66 +
 ✓ Assembler Listing: .\Listing ✓ Cross Reference 	gs*.lst	
C Compiler Listing: .\Listing C Preprocessor Listing: .\L	gs*.bxt .istings*.i	
	ojectmap	
Linker Listing: .\Listings\Pr		
 ✓ Linker Listing: .\Listings\Pr ✓ Memory Map 	Symbols	Size Info
 ✓ Linker Listing: .\Listings\Pr ✓ Memory Map ✓ Callgraph 	Symbols Cross Reference	✓ Size Info✓ Totals Info
 ✓ Linker Listing: .\Listings\Pr ✓ Memory Map ✓ Callgraph 	✓ Symbols✓ Cross Reference	 ✓ Size Info ✓ Totals Info ✓ Unused Sections Info



ptions for Target 'FWLib'	A prove of some of the	
vice Target Output Listing User	C/C++ Asm Linker Debug Uti	lities
Undefine:		
Language / Code Generation Execute-only Code Optimization: Level 3 (-03)	 Strict ANSI C Enum Container always int Plain Char is Signed 	Warnings: All Warnings
Split Load and Store Multiple One ELF Section per Function 	Read-Only Position Independent Read-Write Position Independent	☐ No Auto Includes ☐ C99 Mode
Include Paths Misc Controls	\Libraries\CMSIS\device;\\Libraries\CMSIS\dev	rice\startup;\\Libraries\drivers
Compiler ccpu Cortex-M3 -D_MICI control \Libraries\CMSIS\device -L. string \inc\reg -L.\\Utilities -L.\App\	ROLIBli -g -O3apcs=interworksplit_sections - \Libraries\CMSIS\device\startup -l\\Libraries\d inc -l\\Libraries\IQMath	I\\Libraries\CMSIS\cm3 -I\ rivers\inc -I\\Libraries\drivers

Tips:

开发初期, C 的 Optimization 建议选择 OO。使用 O2 与 O3 虽然有较佳的代码大小或是 运算效率,但是在调试时变量数值不一定与预期相符合,即使运算结果正确,却容易 造成开发困难。

Optional 配置 Debug 选项 1.9.

选择你所需要用的 JLINK 或者 ulink 调试工具



C Use <u>S</u> imulator	with restrictions Settings		J-TRACE Cortex Settings
 Load Applicat Initialization File: 	ion at Startup 🔽 Run to main()	 Load Applicat Initialization File: 	ion at Starlup 🔽 Run to main()
	E dit		Edit
Restore Debug	Session Settings	Restore Debug	Session Settings
Breakpoin	ts 🔽 Toolbox	Breakpoint	ts 🔽 Toolbox
₩ Watch Wi	ndows & Performance Analyzer	Watch Wi	ndows
Memory D	isplay 🔽 System Viewer	Memory D	isplay 🔽 System Viewer
CPU DLL:	Parameter:	Driver DLL:	Parameter:
SARMCM3.DLL	-MPU	SARMCM3.DLL	-MPU
Dialog DLL:	Parameter:	Dialog DLL:	Parameter:
DCM.DLL	-pCM3	TCM.DLL	-pCM3
	Manage Component View	ver Description File	es



Use Simula	tor Ito Real-Time	Setting	sm Linker s @ Use:	ULINK2/ME Cortex Deb	ougger	Settings
Load Applie	cation at Startup 🔽 Ru 1 Target Driver Setup	in to main()	V Load A	pplication at Startup	Run to r	nain()
Debug F – ULIN Se	Trace Flash Download IK USB - JTAG/SW Adapter rial No: V0010M9E VLINK Version: ULINK2	SW Dev	Vice IDCODE Ox2BA01477	Device Name ARM CoreSight SW	-DP	Move Up
P Firm	Device Family: Cortex-M ware Version: V2.03 SWJ Port SW Max Clock: 2MHz	Aut Aut Aut Add	omatic Detection nual Configuratio	ID CODE Device Name	: 	AP: 0x00
ie Debi Co Co	ug onnect & Reset Options onnect Normal 💌 F Reset after Connect	Reset Autodete	ect 💌	Cache Options	Download Verify C Downlo	Options Code Download ad to Flash

建议选择 SW 模式进行 Debug,请注意 SW Device 必须要有显示芯片 IDCODE 才能算是连结成功。

此步骤最常出现无法与芯片沟通之状况,若发生问题,可以检查: (1) 芯片的 TRSTn 是否没有拉到高电位,拉到高电位后,单击板子上的 Reset 键。 (2) JTAG 或是 SWD 走在线的滤波电容是否过大,建议可先去除滤波电容

1.10. Optional 配置 Debug 的 Flash 算法选项





Note :

若在图中没有拷贝算法程序的档案,此处不会出现 相关芯片的选项。

chip	算法文件					
SPC1068	SPC1068.FLM					
SPD1078	SPC1068.FLM					
SPC1158	SPC1168.FLM					
SPC1168	SPC1168.FLM					
SPC1178	SPC1168.FLM					
注: 文件一般放在此文件放置在 SDK 内的 Utilities 内,如 \SDK\SPC1068\Utilities\SPC1068.FLM						
请把相关算法文件拷贝到相应的目录下 C:\Keil_v5\ARM\Flash						



1.11. 保存配置

按下 Save All,请务必记得此步骤,当 Keil 重启时才会纪录之前所有的步骤



1.12. 编译结果



Tips:

除非项目出现严重错误需要 Rebuild all,建议按下中间的按钮即可,compiler 只会编译 改动过的代码,减少等待时间 Note: 编译之后,专案的资料夹会产生编译后的 object 档案,会导致资料夹过大,不方便夹

编译之后,专条的资料夹层广生编译后的 object 档条,层等政资料夹过入,不力使 带在电邮中,请参考 Appendix 錯誤!找不到參照來源。的方法进行解决。

1.13. 下载程序



le Edit View Project Flash Debug	Peripherals	Tools SVCS W	indow Help	
ો 🚰 🖬 🗿 💰 🥹 🖏 🗖 ભે 🌼	1 m 1 m	图图 谭谭//	/ 🖾 сом	P_ClearAllO
🖻 🕮 🧼 🌙 🙀 FWLib	- *	1 🗟 🔶 🗇 ಖ		
oject	a 💌	spc1068.h	global.h	🛓 mai
🖕 🔄 Periph_Driver		412	myMotor[0].sOth
↓ Build Output			-	E
Load "D:\\SpintrolAE\\HengFang Erase Done. Programming Done. Verify OK. Flash Load finished at 14:36:0	g\\SPC1068\\ 07	Project\\Debug\\(Objects\\Proje	ct.axf"

1.14. 调试程序



1.15. 仿真注意事项

Jlink 在线仿真时,遇到无法正确复位的问题。此处是因为在点复位后需要在 4s 内点运行程序才能正常的在线 仿真,否则会出错!



2. JLINK/ULINK2 工具使用指南

2.1. 调试工具与开发板 JTAG 连接



图 2-1: ULINK2 适配器接口





图 2-1: JLINK OB 调试器为推荐的调试工具,淘宝搜索一下 jlink ob 就可以 看到。ulink 适配器支持 5 种 JTAG 接口,如图 2-1 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 所示。ulink 或者 jlink 只需要选择一种就可以了。

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



ARM 20-PIN Interface

	 1
VCC 1	2 VCC (optional)
TRST 3	4 GND
TDI 5	6 GND
TMS 7	8 GND
TCLK 9	10 GND
RTCK 11	12 GND
TDO 13	14 GND
RESET 15	16 GND
N/C 17	18 GND
N/C 19	20 GND

图 3.1-2: ARM 20-PIN 接口

在采用 SPINTROL 芯片进行应用开发的过程的中,需要经常使用 JTAG 进行程序的调试。调试工具与 SPINTROL 的硬件连接如图 1-3 所示。表 3.1-1 中为具体的 PIN 脚连接关系。

JTAG	SWD	SPC10 68	SPD107 8	SPC1168 SPC1158 SPC1178 SPC1188	SPC1148	持续更 新
TMS	SWDIO	GPIO16	GPIO30	GPIO38	GPIO38	
TDO	/	GPIO17	GPIO29	GPIO36	GPIO36	
RTCK	/	1	1	1	1	
TDI	/	GPIO15	GPIO28	GPIO37	GPIO37	
TRST	/	TRSTn	TRSTn	TRST	TRST	
TCLK	SWDCLK	GPIO18	GPIO31	GPIO39	GPIO39	
VCC	VCC	+3.3V	+3.3V	+3.3V	+5V	
GND	GND	GND	GND	GND	GND	
RESET	/	1	1	1	1	
注: 相关管	脚位置可以参考	考 datasheet	的 Pin-outs	and pin descr	iption 章节	

表 3.1-1: 与硬件连接图



VCC	1		2	VCC (optional)	VCC	1			2	VCC (optional)
TRST	3		4	GND	N/U	3			4	GND
TDI	5		6	GND	N/U	5			6	GND
TMS	7		8	GND	SWDIO	7			8	GND
TCLK	9		10	GND	SWCLK	9			10	GND
RTCK	11		12	GND	N/U	11			12	GND
TDO	13		14	GND	swo	13			14	GND
RESET	15		16	GND	RESET	15			16	GND
N/C	17		18	GND	N/C	17			18	GND
N/C	19		20	GND	N/C	19			20	GND
		Λ.				C				
	J	4	J		1	C	V	٧L	J	
							htt	tp:/	/b	log.csdn.net/

2.2. 硬件管脚 JTAG/ULINK/JLNK 配置

使用 ULINK/JLINK 工具下载程序时,首先需要设置 Boot Pin 管脚,以及 TRSTN, 然后按下 RESET 按键,芯片处于正常模式,不然将处于 ISP 模式。此时不能进行 JTAG/ULINK/JLINK 调试。

芯片	Boot PIN	TRSTn	RESET 按键					
SPC1068/SPD1078	GPI00 需要拉高	TRSTn 需要拉高	最后按下 reset 按键,进入下载模式					
SPC1158/SPD1148	GPI040 需要拉高	TRSTn 需要拉高	最后按下 reset 按键,进入下载模式					
SPC1168/SPC1178/SPC1188								
注: 详细可以查看芯片的 datasheet 的 Boot mode 章节,具体管脚参考 Pinout and pin description 章节								

2.3. KEIL 环境下 JLINK 配置

在安装 KEIL MDK 时,软件会默认安装 JLINK 设备的驱动。按照上面步骤将 Jlink 与硬件连接,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:



RM Cort	ex-M3		Xtal (MHz):	12.0		Generation I Compiler:	Use latest ir	nstalled version	_
Operating	system:	None		-			1		_
System Vi	ewer File:				ΠU	se Cross-N	Nodule Optimiza	tion	
(ΓU	se MicroL	в Г	Big Endian	
Use (Custom Fil	e							
-Read/0	Only Memo	ory Areas			- Read/	Write Merr	iory Areas		
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	Nolnit
Г	ROM1:			- c	Г	RAM1:			Γ
	ROM2:			- c	Г	RAM2:			Г
Г	ROM3:	Ì	í	- c	Г	RAM3:	Ĺ	<u></u>	Г
	on-chip					on-chip	,	1	
◄	IROM1:	0x1FFF8000	0x8000	œ	•	IRAM1:	0x20000000	0x4000	
Γ	IROM2:			с		IRAM2:			

图 2-1: Options for Target

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 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,如果是使用 ULINK2,就选择使用 ULINK2/ME Cortex Debugger 选项,如果是 JLNK 就选择 J-LINK/J-TRACE Cortes 选项。单击 Settings 按钮,会弹出相关的设置,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/J-LNK 设备此时是正常工作的;否则,则表明设备不可用。因此,在用 ULINK2/J-LNK 调试程序时,常常用此方法检查 ULINK2/J-LNK



设备是否正常。目前, SPINTROL 芯片的 Debug 模块只能支持 VECTRESET 功能, 即在 Debug 的时候只能 Reset 芯片的内核。因此,必须按照图 2-2 配置 Connect & Reset Options。此外, SPINTROL 芯片支持 JTAG 和 SWD 两种 Debug 协议,用 户可以根据需要进行配置。

C Use Simulato	or Settings	● Use: ULINK2/ME Cortex Debugger Settings
Limit Speed	o Real-Time	
✓ Load Applica Initialization File:	ation at Startup 🔽 Run to main()	✓ Load Application at Startup ✓ Run to main() Initialization File:
	Edit	Edit
Restore Debug	g Session Settings	Restore Debug Session Settings
🔽 Breakpoi	nts 🔽 Toolbox	I Breakpoints I Toolbox
Watch W	/indows & Performance Analyzer	Watch Windows
Memory I	Display IV System Viewer	I✓ Memory Display I✓ System Viewer
CPU DLL:	Parameter:	Driver DLL: Parameter:
SARMCM3.DLL		SARMCM3.DLL
Dialog DLL:	Parameter:	Dialog DLL: Parameter:
DCM.DLL	pCM3	TCM.DLL pCM3
	OK C	ancel Defaults Help



g Periphera	als Tools SVCS Window Help		
\leftarrow	陀 🏗 🥂 🕀 🛊 🗊 //注 //注 🖄 i16ThetaMerge	🗟 🐝 🎯 र 🕘 📀 🔗 📽 र 🔲 🖬 र	
	🖂 💑 📇 🗣 🗇 🎰		
4	🗑 Ontions for Target 'FWLib'	×	
	Device Target Output Listing User C/C++ Asm	h Linker Debug Utilities	
	C Use Simulator <u>with restrictions</u> Settings G	♥ Use: J-LINK / J-TRACE Cortex ▼ Settings	
	✓ Load Application at Startup ✓ Run to main() Initialization File:	✓ Load Application at Startup	
	Edit	Edit	
	Restore Debug Session Settings	Restore Debug Session Settings	
	Image: wide of the set of the s	Breakpoints Toolbox	2
	Watch Windows & Performance Analyzer	Watch Windows	
	I✓ Memory Display I✓ System Viewer	V Memory Display V System Viewer	
	CPU DLL: Parameter: D	Driver DLL: Parameter:	
	SARMCM3.DLL -MPU	SARMCM3.DLL -MPU	
are.h	Dialog DLL: Parameter: D	Dialog DLL: Parameter:	
	DCM.DLL pCM3	TCM.DLL pCM3	
	图 2.3.2: JLINK Debu	ug 配置界面	

Debug Trace Flash Download ULINK USB - JTAG/SW Adapter JTAG Device Chain Serial No: V0010M9E IDCODE Device Name IDCODE Device Name
ULINK Version: ULINK2 Device Family: Cortex-M Firmware Version: V2.03 Image: SWJ Port: JTAG Image: SWJ Port: Max Clock: 1MHz Image: Add Image: SWJ Port:
Debug Connect & Reset Options Connect: Normal ✓ Reset: VECTRESET ✓ ✓ Cache Options ✓ Download Options ✓ Cache Code ✓ Cache Memory ✓ Download to Flash
OK Cancel Help





接下来,在图 2-3.1 或者图 2-3.2,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,单击副按钮或者按下 F5 键,程序就会快速 执行到断点处,如图 2-3.7 所示。



D:\SpintrolAE\H	lengFang\SPC1068\Project\De	ebug\PICA_FWLibuvproj - µVision	- 0 <u>- X</u>
<u>F</u> ile <u>E</u> dit <u>V</u> iew	Project Flash Debug Peris	pherals <u>T</u> ools <u>SVCS</u> <u>Window</u> <u>H</u> elp	
	8 4 🖄 🔊 e 👄	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	
RST 🗄 🧐 {	9 8+ 8+ +8 ♀ ☑ 🖾		_
Registers	Ф (<u>с</u>	a Disasembly	Ф 🛛
Register	Value	7: Sys_Init();	*
□ Core ■ 00 11 12 13 12 14 12 15 16 16 13 16 14 17 16 18 15 16 114 115 15 16 114 17 114 18 15 16 Parks 16 Parked 17 Parked	0x20000088 0x20000485 0x20000485 0x20000485 0x20000000 0x20000000 0x20000000 0x20000000 0x219779000 0x219779000 0x219779010 0x20000008 0x20000008 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x20000088 0x200000088 0x200000088 0x200000088 0x20000000000	<pre> S: ONIFFEB18 F7FFE90 BL.W Sys_Init (0x1FFF88SC) 9: CLOCK_InitWithRCO(CLOCK_RCLK_24MHZ); 10: Dx1FFFBB1C 2000 MOVS r0,#0x00</pre>	• • • •
Mode Privilege Stack States Sec	Threed Privileged MSP 2494794 0.24947940	<pre>11</pre>	
Project Bis Regi	sters		· · · ·
Command		4 🖸 Call Stack + Locals	p 🖬
Load "D:\\Spin	ntrolAE\\HengFang\\SPC	Ild68\\Project\\Debug\\Objects\\PICA_FWLib.axf" ~ Name Location/Value Type	
ASSIGN BreakD	isable BreakEnable Bre	rakKill BreakList BreakSet BreakAccess COVERAGE	
		ULINK2/ME Cortex Debugger 11: 0.24947940 sec L7 C1 CA	P NUM SCRL OVR R/W

图 2-3.5: 勾选 Run to main()选项运行结果

D:\SpintrolAE\H	engFang\SPC1068\Project\De	ebug\PICA_FWLib.uvproj - µVision	
<u>File Edit View</u>	Project Flash Debug Perip	vherals Iools SVCS <u>Wi</u> ndow <u>H</u> elp	
🗋 🖸 🐸 🖉	* 🔤 🕮 🖌 😁 🖛 =	◇ 隆 黎 雅 微 律 準 准 版 🖄 📃 🔍 🔍 🐼 👰 🔍 ◇ ◇ 🔗 🍓 💷 🔍	
🛛 🗱 🖹 💷 🚳 🛛 74) () () *() ⇒		
Registers	д) Disassembly	д 🖬
Register	Value	cox00000030 4804 LDR r0,[pc,#16] ; @0x00000044	*
Core R0 R1 R2 R3 R4 R5	0x00000000 0x0000000 0x0000000 0x0000000	OKOUDU032 4780 BLX FU 0x0000034 4804 LDR r0 [pc, #16] ; @0x00000048 0x0000036 4700 BX r0 0x00000036 0x00000038 0x00000038 E7FE B 0x00000038 0x00000038 0x0000003A 0x00000036 E7FE B 0x0000003C 0x0000003E 0x0000003E	
R6	0x00000000	0x00000040 E7FE B 0x00000040	-
R8	0x00000000	4 m00000043 ETEE B 0*00000043	•
	0x00000000		
R11	0x00000000	main.c	▼ X
R12	0x00000000	1 #include "spc1068.h"	<u>^</u>
R13 (SP) R14 (LR) R15 (PC)	0x20003980 0xFFFFFFF 0x00000030	2 #include <stdio.h> 3</stdio.h>	
Banked System Internal Mode Privilege Stack	Thread Privileged MSP	<pre>5 int main() 6 = { 7</pre>	E
States Sec	0 0.00000000	<pre>11 ···Delay_Init(); 12 ··· 13 ·· 14 ···/*·Set·GPIO·function·as·UART···*/ 15 ···GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD); 16 ···GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD); </pre>	-
m Project m Regis	ters		· _
Command		Call Stack + Locals	д 🖬
Load "D:\\Spin	trolAE\\HengFang\\SPC	1068\\Project\\Debug\\Objects\\FICA_FWLib.axf" Mame Location/Value Type	
ASSIGN BreakDi	sable BreakEnable Brea	akKill BreakList BreakSet BreakAccess COVERAGE	
Dictabl		III INFECTION OF THE CONTROL OF THE	TAD NILIM SCOL OVD DAV
		Stankeyme concer bebugget it. 0.00000000 set E./ C.I	Stand Sche Ovid KVW

图 2-3.6: 未勾选 Run to main()选项运行结果



D:\SpintrolAE\H	engFang\SPC1068\	\Project\Debug\PICA_FWLib.uvproj - µVision	- 0 - X -
Eile Edit View	Project Flash De	ibug Peripherals Iools SVCS Window Help	
n 🖻 🖬 🥥	s 🕹 🚨 🔊	▷ ← ⇒ 巻 急 急 谋 準 進 版 20 ・ ○ ▲ ■ ● ◇ ◇ ▲ ■ ● ◇	
👫 🗉 🚳 🤻) () () () () () ()		
Registers		4 2 Disassembly	4
Register	Value		*
- Core	000007814	9: CLOCK INITWICHCO (CLOCK HCLK 24MHZ);	
R1	0x4000B000	10:	
R2	0x00000020	Ox1FFF8B1C 2000 MOVS r0,≢0x00	
R3	0x00000020	0x1FFF8B1E F7FFC43 BL.W CLOCK_InitWithRCO (0x1FFF83A8)	
	0x20000088	11: Delay_Init();	
R6	0x00000000	13:	
R7 R8	0x00000000 0x00000000	14. /* C-* CDTO European TEDT */	
	0x1FFF8000		
R10	0x1FFF8BAC	main.c	▼ ×
R12	0x00000000	4	
	0x200008E8	5 int main()	
R14 (LR)	0x1FFF8729	6 □ (
KIS UUJ	0x1FFF8B1C	7 Sys_Init();	
+ Banked	0.1000000		
🗄 System		y s clock_initwithkco(clock_clk_24mz);	Ξ
- Internal Mode	Thread	11 ···Delay Init();	
Privilege	Privileged	12	
Stack	MSP	13 ***	
States	2497292	14 ···/*·Set·GPIO·function·as·UART··*/	
Sec	0.24512520	15 GPIO SetPinChannel(GPIO 34, GPIO34 UART TXD);	
		18 GPIO_SetFinchannel(GPIO_35, GPIO35_UART_RXD);	
		18/*·Enable·UART·Clock·*/	
		19 CLOCK EnableModule (UART MODULE);	-
Project Regi	sters		•
Command		a Call Stack + Locals	a 🖬
Load "D:\\Spir	trolAE\\HengFa	ang\\SPC1068\\Project\\Debug\\Objects\\FICA_FWLib.axf" ^ Name Location/Value Type	
		• •	
>			
ASSIGN BreakDi	sable BreakEna	able BreakKill BreakList BreakSet BreakAccess COVERAGE 🛛 🚰 Call Stack + Locals 🐺 Trace Exceptions 🐺 Event Counters 🗔 Memory 1	
		ULINK2/ME Cortex Debugger t1: 0.24972920 sec L9 C:1 CAP	NUM SCRL OVR R/W

图 2-3.7: 未勾选 Run to main()选项执行至断点情形

2.4. 配置 ALGorithm

在使用 ULINK2/J-LNK 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\) 各种芯片对应的 FLM 文件,都能在相应的 SDK 里面找到。



ownload Function C Erase Full Chip Erase Sectors C Do not Erase	 ✓ Program ✓ Verify ✓ Reset and Run 	RAM for Alg	orithm 20002000 Size: 0x2000	
rogramming Algorithm				
Description	Device Type	Device Size	Address Range	
SPC1068 48KB Flash	On-chip Flash	48k	1FFF8000H - 20003FFFH	
SPC1068 48KB Flash	On-chip Flash	48k Start: Qx1	1FFF8000H - 20003FFFH FFF8000 Size: 0x0000C0	00

图 2-7: Flash Download 设置

Description	Device Type	Device Size	
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	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	
SN32F760 64KB User ROM	On-chip Flash	64k	
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 64kB Flash	On-chip Flash	64k	
STM32F0xx Flash Options	On-chip Flash	16	
STM32F10x XL-density Flash	On-chip Flash	1M	
STM32F10x Med-density Flash	On-chip Flash	128k	-

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

2.5. KEIL 环境下使用 ULINK2/JLNK 调试

根据前面的介绍,将ULINK2/Jlink 设备与板子正确连接后,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/Jlink 设备调试程序了。使 用 ULINK2/Jlink 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。

wice farger output Listing oser	C/C++ Asm Linker Debug Utilities
Configure Flash Menu Command	
Use Target Driver for Flash Programming	Vise Debug Driver
Use Debug Driver	Settings Update Target before Debugging
Init File:	Edit
C Lise External Tool for Flash Programming	
Command()	
Arguments:	
Arguments:	
Arguments: Run Independent Configure Image File Processing (FCARM):	
Arguments: Run Independent Configure Image File Processing (FCARM): Output File:	Add Output File to Group:
Arguments: Run Independent Configure Image File Processing (FCARM): Output File:	Add Output File to Group:
Arguments: Run Independent Configure Image File Processing (FCARM): Output File: Image Files Root Folder:	Add Output File to Group:

图 3-1: Update Target before Debugging 设置

单击工具栏上的 ④ 按钮进入 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。

注意: (1) 目前版本的 SPC1068 芯片 Reset CPU 命令只支持 VECTRESET 功能。

(2) 在程序进入 Debug 状态后,代码是不可以修改的。如果想修改代码,需要单击按钮 2 退出 Debug 模式,然后才能修改代码。修改后的代码编译通过后,将代码 重新下载到 Flash 中,用户可以继续单击按钮 3 进行 Debug。

Debug Menu	Toolbar	Shortcut	Description
Start/Stop Debug	0	Ctrl+F5	Starts or stops a debugging session.
Session			
Reset CPU	RST		Sets the CPU to RESET state.
Run	E.	F5	Continues executing the program until
			the next active breakpoint is reached.
Stop	8		Stops the program execution
			immediately.
Step	(+)	F11	Executes a single-step into a function;
			Executes the current instruction line.
Step Over	8	F10	Executes a single-step over a function.
Step Out	{} }	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	۲	F9	Toggles the breakpoint on the current
Breakpoint			line.
Enable/Disable	0	Ctrl+F9	Enables/disables the breakpoint on the
Breakpoint			current line.
Disable All	S		Disables all breakpoints in the

表 3-1: Debug Menu and Commands



Breakpoints		program.
Kill All Breakpoints	Ctrl+Shift+F9	Removes all breakpoints in the program.

D:\SpintrolAE\F	lengFang\SPC1068\Project\Deb	ug(Project.uvproj - µVision	
Eile Edit View	Project Flash Debug Periph	erals Iools SVCS Window Help	
	8 43 13 19 19 19 19 19 19 19 19 19 19 19 19 19	M % % % ‡ # /// /// @ m	
RST 🗐 🥨 {	}}+{}+{}++{}) ⇒ ⊡®		
Registers	4	Bisassembly	‡ 🔝
Register	Value	7: Sys_Init();	*
E-Core			
RO	0x200000E8	A CLOSE TAST MARKADON CLOSE MET A AMAZA	
KI P2	0x20000488	10.	
	0x200004E8	0x1FFF8B1C 2000 MOVS r0.#0x00	
	0x00000000		
R5	0x20000088	11: Delay Init(); 汇编代码窗口	
R6	0x00000000	12:	
R8	0x00000000	13:	
R9	0x1FFF8000	14: /* Set GPIO function as UART */	*
R10	0x1FFF8BAC		F.
R11	0x0000000		
B13 (SP)	0x200000008	main.c	¥ X
R14 (LR)		2 #include <stdio.h></stdio.h>	*
R15 (PC)		3	
± RPSR	0x21000000	4	
+ System		5 int main ()	-
E Internal	PU句仔岙图L		=
Mode	Thread	· · · · · · · · · · · · · · · · · · ·	
Privilege	Privileged		
Stack	MSP 2494794		
States	0.24947940	11 ···Delay Toit():	
10000			
		13	
		14 ··/*·Set·GPIO·function·as·UART··*/	
		15 GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);	
	-	16 GPIO SetPinChannel (GPIO 35, GPIO35 UART RXD);	*
🔃 Project 🛛 🗮 Regi	sters		•
Command		a 🔯 Call Stack + Locals	P 🖸
Load "D:\\Spin	htrolAE\\HengFang\\SPC1	.068\\Project\\Debug\\Objects\\Project.axf" ^ Name Location/Value Type	
		v	
•		,	
>			
ASSIGN BreakD:	isable BreakEnable Brea	kKill BreakList BreakSet BreakAccess COVERAGE 🛛 🖓 Call Stack + Locals 🔲 Memory 1	
		ULINK2/ME Cortex Debugger 11: 0,24947940 sec L:7 C:1	CAP NUM SCRL OVR R /W

图 3-2: 启动 Debug 后的界面

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 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70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。

D:\SpintrolAE\	lengFang\SPC1068\Project\Debug	g\Project.uvproj	- μVision			- 0 <u>- x</u>
File Edit View	Project Flash Debug Periphera	als Tools SVCS	Window Help			
🗋 🖸 🚰 🗃 🗃	3 43 😤 47 Pr 🗲 44	P 2 2 2	Split Window horizontally		💽 🗟 🥐 🕘 🔹 🔗 🏡 🔳 🔹 🔦	
RST 🗉 🚳 🗍	₽₽₽₽	- 🖓 🕄	Go to Headerfile			
Registers	4 E	Disassembly	Show Dicassembly at 0x1 EEE	8838		4
Register	Value	16:	Set Program Counter	0000	O35_UART_RXD);	-
- Core	000000078	18:	#3 Pup to Curror line	Ctrl+E10		
	0x200004E8	0x1FFF8B	{} Kunto cursor line	Cuittio	; @Ox1FFF8B70	
R2 R3	0x200004E8 0x200004E8	0x1FFF8B	Insert/Remove Breakpoint	F9	01	
	0x00000000	0x1FFF8B	O Enable/Disable Breakpoint	Ctrl+F9	0]	-
R5 R6	0x20000088 0x00000000	I I I	Insert Tracepoint at line 16	+		P
	0x00000000	main.	Enable/Disable Tracepoint			▼ ×
R8 R9	0x1FFF8000	1				
	0x1FFF8BAC	2	re Insert/Remove Bookmark	Ctrl+F2		
R12	0x200000008	3	=) Undo	Ctrl+Z		
R13 (SP)	0x200008E8	5		Ctrl+Y		
R14 (LK)	0x1FFF8B18	6 🖯	∦ Cut	Ctrl+X		
*PSR	0x21000000	2 7	Ца Сору	Ctrl+C		=
+ System		9	Paste	Ctrl+V	iMHZ);	
- Internal	Thursd	10	Select All	Ctrl+A		
Privilege	Privileged	11	Execution Profiling	+		
Stack	MSP 2494794	13				
Sec	0. 24947940	14	Outlining			
		15	Advanced	FIU 35, GF	1034_0ARI_IXD); 1035_UART_RXD);	
		17	·			
		18	/* Enable UART Clock	*/		
		19	CLOCK_EnableModule (0/	ARI_MODULE);	-
me Project me Reg	isters	1.				
Command				4	Call Stack + Locals	P
Load "D:\\Spi	ntrolAE\\HengFang\\SPC106	58\\Project\	\Debug\\Objects\\Projec	ct.axf"	Name Location/Value Type	
					w.	
<						
>						
ASSIGN BreakD	isable BreakEnable BreakF	(ill BreakLi:	st BreakSet BreakAccess	COVERAGE	Call Stack + Locals Memory 1	
Run to the current of	ursor line				ULINK2/ME Cortex Debugger 11: 0.24947940 sec L:16 C:1 CAP NUM	SCRL OVR R/W

图 3-3: Run to Cursor Line 实现

此外,我们也可以通过设置断点的方式来实现上述功能。

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,单击工具栏上的副按钮或者按下快 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

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) 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。



D:\SpintrolAE\H	engFang\SPC1068\Proje Project Flash Debug 3 4 12 1 - 0 - 1 3 17 17 1 + 10 1 - 0 1 - 10	ct\Debug1\Projec Peripherals Tool: Colored Colored	-91	Split Window horizontally Go to Headerfile Show Disassembly at 0x1FFF8976 Set Program Counter Pun to Curror line Ctria	10	- A (Q) •	• 0 2
Register Core R1 R2 R3 R4	Value 0x20000080 0x200004E0 0x200004E0 0x200004E0 0x200004E0 0x200004E0 0x200004E0	14: 15: 0x1FFF8976 0x1FFF8978 0x1FFF897A 16:	•	Insert/Remove Breakpoint Enable/Disable Breakpoint Go To Definition Of 'T Go To Reference To 'T	F9 F9	; @0x1FFF8988	* 0 *
15 - R5 - R7 - R3 - R1 - R12 - R13 - R14 - R15 - Stack - Stack - Stack - Stack - Stack - Stack	0x2000007C 0x40000000 0x40000000 0x1FFF8000 0x1FFF8000 0x20000000 0x20000000 0x20000000 0x20000000 0x20000000 0x1FFF8185 0x1FFF8185 0x1FFF886C 0x21000000 Thread Privileged MSP 2482217 0.24822170	main.c 1 #i 2 #i 3 4 5 ui 6 7 8 in 9 { 10 12 13 14 15 16 17 18	5 C & 1 B	Add 'i' to Insert Tracepoint at 'i' Enable/Disable Tracepoint Undo Ctrl+ Undo Ctrl Redo Ctrl Cut Ctrl Copy Ctrl+ Select All Ctrl+ Execution Profiling Ctrl+ Outlining Advanced	+ F2 +Z +Y +X +C +V +A +	Watch 1 Watch 2 Memory 1 Memory 2 Memory 3 Memory 4 Logic Analyzer	▼ X
Deproject Begi	sters	4	3	Call Stack + Locals			۲ ب
Load "D:\\Spir WS 1, `i	sable BreakEnable	\SPC1068\\Pr .	-	Name Location/Valu	e	Type ULINK2/ME Cortex Debugger 11: 0.24	822170 st

图 3-6: 添加变量到观察窗口

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。



D:\SpintrolAE\H	engFang\SPC1068\Pro	iject\Debug1\Project.uvproj - μVision	×
File Edit View	Project Flash Debug	g Peripherals Tools SVCS Window Help	
🗋 🖸 🐸 🖉 🖉	3 4 🖪 🕫 🖻	← → 隆 🎕 🍇 ‡ ‡ //: //¿ 🖄 BIT 🕢 💽 🗟 🛷 💽 🌢 •	0
👫 📃 🔕 🥂	} 0 () *() ⇒ [:	S (a)	
Registers	4 💽	Disassembly	4
Register	Value	14: i = 5;	•
🚍 Core	and the second se	15:	_
RO	0x200000E0	0x1FFF8976 2005 MOVS r0, #0x05	
RI	0x200004E0	0x1FFF8978 4903 LDK r1,[pc,#12] ; @0x1FFF8988	
	0x200004E0	16: j++:	+
R4	0x00000000	< I	•
R5	0x2000007C		
R6 R7	0x0000000	main.c	• •
R8	0x00000000	1 #include "spc1068.h"	^
R9	0x1FFF8000	2 #include <stdic.h></stdic.h>	
R10	Ox1FFF89AC	3	
RI1	0x200000BC	5 uint 32 t.i. = 0	
R13 (SP)		6 amesiz e 1 = 0,	
R14 (LR)	0x1FFF818B	7	
R15 (FC)	0x1FFF896C	8 int main()	
+ Banked	0821000000	9 🗏 1	=
🗄 System		10 Sys_Init();	
- Internal	T1	11	
Privilege	Privileged	12 CLOCK_INITWITHRCO(CLOCK_HCLK_24MHZ);	
Stack	MSP	N 14	
States	2482217	15	
Sec	0.24822170	16 i++;	
		17	
		18 while(1);	
		19 }	-
🔃 Project 📰 Regis	sters		•
Command		4 Watch 1	P 🖸
Load "D:\\Spin	trolAE\\HengFang	r\\SPC1068\\Pr A Name Value Type	
WS 1, 1		i 0x00000000 unsigned int	
		<enter expression=""></enter>	
>			
ASSIGN BreakDi	sable BreakEnabl	e BreakKill 🕼 Call Stack + Locals Watch 1 🛄 Memory 1	
		ULINK2/ME Cortex Debugger t1: 0.2482	2170 s

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。



图 3-8: i=5 执行结果

图 3-9: i++执行结果

4) 观察外设寄存器值

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[∞],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。



	ev-M3									
				10.0	Code	Generation				
			Xtal (MHz):	12.0	ARM Compiler: Use latest installed version					
Operating	g system:	None		_						
System Viewer File:						Use Cross-Module Optimization				
.E\Heng	Fang\SP(C1068\Project\[Debug SPC10)68.SFF	Γι	lse MicroL	IB ľ	Big Endian		
🔽 Use	Custom Fi	le								
Read/Only Memory Areas					Read/Write Memory Areas					
default	off-chip	Start	Size	Startup	default	off-chip	Start	Size	NoInit	
П	ROM1:			C	Г	RAM1:				
Г	ROM2:			с	Г	RAM2:	[
	ROM3:			- c	Г	RAM3:				
Г	on-chip					on-chip				
Г		0x1FFF8000	0x8000	¢	V	IRAM1:	0x20000000	0x4000		
<u>ح</u> ا	IROM1:	Press and a second second second		_	-	ID AMO	-			

图 3-10: System Viewer File 设置界面



©2014-2025, Spintrol Limited Corporation







图 3-12: 添加 PWM0 到 System Viewer 窗口的结果



图 3-13: TBCTL=0 执行结果


-				
<pre>Dx1FFF89E2 60C8 STR r0,[r1,#0x0C] 17: while(1);</pre>	Â			
0x1FFF89E4 BF00 NOP		Property	Value	
Dx1FFF89E6 E7FE B 0x1FFF89E6 Dx1FFF89E8 7000 DCW 0x7000 Dx1FFF89EA 4000 DCW 0x4000	, - , -	 → TBPRD → TBPHS → TBCTR → TBCTL 	0 0 0 0x00001234	
3 4 4	<u>, , ,</u>	FREE SOFT PHSDIR		
6 □ { 7 · Sys_Init(); 8 9 · CLOCK_InitWithRCO(CLOCK_HCLK_24MHZ); 10 · · 11 · CLOCK EnableFWMModule(FWMO);		TBCLKDIV SWFSYNC SYNCOSEL PRDLD PHSEN	0x04	
12 13 PWMO->TBCTL.all = 0; 14 15 PWMO->TBCTL.all = 0x1234; 16 17 while(1);	H	CTRMODE TBSTS CMPA CMPB CMPCTI	0x00 0x00000001 0 0	
18 }		TBCTL [Bits 310] RW (@ 0x4	4000700C) TBCTL	

图 3-14: TBCTL=0x1234 执行结果

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 PWMO 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。



D:\SpintrolAE\He	engFang\SPC1068\Project\De	ibug2\Project.uvproj - µVision					
File Edit View	File Edit View Project Flash Debug Peripherals Tools SVCS Window Help						
N 😂 🖬 🥥	* 🔤 🖪 🔊 🗠 📥	- 🍖 🎕 🎕 澤 澤 /// //(: 🎯 BIT 🕢 🕞 🗟 🥐 阈 🖕 🔹 🔗 🌨					
🚼 🗟 🤇) 🖓 () 🐴 () 🍕 🚺						
Registers	4 🖬	Disassembly Memory 1 📮 🖬					
Register	Value	0x1FFF89C6 Memory 2 0x1FFF89C6					
R1 R2 R3 R4	0x2000000B 0x200004D8 0x200004D8 0x200004D8 0x200004D8 0x00000000	8: Memory 3 Ox1FFF89C8 Memory 4 Sys_Init (0x1FFF8788) 9: CLOCK_INITWITHRCO(CLOCK_HCLK_24MHZ); 10:					
R5 R6 R7 R8	0x20000078 0x00000000 0x00000000 0x00000000 0x000000	main.c pga.c pga.h spc1068_reg.h * 2 #include <stdio.h> *</stdio.h>					
R9 R10 R11 R12 R13 (SP) R14 (LR) R15 (PC)	0x1FFF8000 0x1FFF80C 0x20000000 0x200000B8 0x20000BB8 0x1FFF818B 0x1FFF818B	4 5 int main() 6 ☐ { 7 ···Sys_Init(); 8					
wPSR wrSR wrSR wrSystem Thernal wrMode Privilage	Ox21000000	<pre>9</pre>					
Stack States Sec	MSP 2484413 0.24844130	14 ··· 15 ···PWMO->TBCTL.all =·Ox1234; 16 ··· 17 ··while(1); 18 }					
🖭 Project 🛛 🧮 Regis	ters	• <u> </u>					
Command		4 🖻 Call Stack + Locals 4 🖻					
Load "D:\\Spin	trolAE\\HengFang\\SPC	1068\\Project\\Del A Name Location/Value Type					
×							
ASSIGN BreakDi	sable BreakEnable Bre	akKill BreakList					
_L		Guinz/Mc Concer Debugger 11:0:24044150 Sec 3					

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:

SPIN TROL



图 3-16: Memory 窗口中观察到的 TBCTL 初始值

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。

SPIN TROL



图 3-17: Memory 窗口结果(TBCTL=0)

SPIN TROL



图 3-18: Memory 窗口结果(TBCTL=0x1234)

3. J-FLASH 烧录下载指南

3.1. 基本要求

J-Link 的驱动版本要是 6.16 以后的;

3.2. 更新 FLM 文件

在 J-Link 驱动的安装目录下,在 Devices 文件夹下新建文件夹 SPINTROL,并将附 件中的 SPC1168.FLM 文件放到 SPINTROL 文件夹中;

三修成,口红文》。 (19][10][7][2]	
芯片	所需文件
SPC2168	SPC2168.FLM
SPD1148	SPC1168.FLM
SPC1158	SPC1168.FLM
SPC1168	SPC1168.FLM
SPD1178	SPC1168.FLM
SPD1188	SPC1168.FLM

注: SPC1168.FLM 是烧录中用到的 Flash 编程算法文件,不过为了适应 J-Flash 软件,做了一些修改,已经更新。不同的芯片对应的.FLM 文件有所不同,如下:

3.3. 更新 Spintrol 产品信息

J-Link 驱动的安装目录下找到 JLinkDevices.xml,将 Spintrol 产品信息添加到里面,具体可参见附件中 JLinkDevices.xml 文件的最后的配置语句。如下:

<device></device>
<chipinfo <="" name="SPC1168" td="" vendor="Spintrol" workramaddr="0x20000000" workramsize="0x4000"></chipinfo>
Core="JLINK_CORE_CORTEX_M4"/>
<flashbankinfo <="" baseaddr="0x10000000" maxsize="0x20000" name="Internal Flash" td=""></flashbankinfo>
Loader="Devices/SPINTROL/SPC1168.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" AlwaysPresent="1"/>
<device></device>
<chipinfo <="" name="SPC2168" td="" vendor="Spintrol" workramaddr="0x20000000" workramsize="0x4000"></chipinfo>
Core="JLINK_CORE_CORTEX_M4"/>
<flashbankinfo <="" baseaddr="0x10000000" maxsize="0x80000" name="Internal Flash" td=""></flashbankinfo>
Loader="Devices/SPINTROL/SPC2168.FLM" LoaderType="FLASH_ALGO_TYPE_OPEN" AlwaysPresent="1"/>



3.4. 用 J-Flash 软件烧录 Hex 文件了

1) 运行 JFlash.exe,新建一个工程,选择要烧录的芯片

SEGGER J	-Flash V6.35d (beta)	×
₽] SEGGER J File Edit	I-Flash V6.35d (beta) View Target Options Window Help	
List of MCU List of MCU File Edit	Llog startel W. S4 (JrF1 ach compiled Sep 17 2018 12:37:29) I. dll V6.35d (DLL compiled Sep 17 2018 12:37:09) devices read successfully (6561 Devices) I-Flash V6.35d (beta) View Target Options Window Help Select device	
	Manufacturer Spinnol Manufacturer Device Core Spinnol SPC168 Contex-M4 Spinnol SPC2168 Contex-M4	Flash size RAM size 128 KB 16 KB 512 KB 16 KB
Application - J-Flash - JLindow		DK Cancel
Line - EMCLI	de deux and annual de 16561 De deux	

2) 选择要下载的 Hex 文件, File ->Open data file



- trunk & Desiret	N Gramplar N Tamelete		- 4.	ter# Objects	0	
W trunk > Project	 U_Examples Template - I 	BJ4- ► MIDK-ARMI ► Objects	• • • • •	搜索 Objects	۷	
组织 ▼ 新建文件夹				= •		
🗟 Git 🔷	名称	修改日期	类型	大小		
Subversion	Project.hex	2019/11/27 18:08	HEX 文件	9 KB		
💾 视频						
■ 文档						
22.5						
□₩ 计算机						
System (C:)						
Data (D:)						
hfhuang (\\192.168.9.10						
PDK Document (\\192.1)						
spintrol (\\192.168.9.10)				r		
文件名(N): Pro	oject.hex		•	Intel Hex files (*.hex)	-	
				打开(O)	取消	3
					d a	-
- JLinkARM. dll V6.35d (ULL compile Creating new project	d Sep 17 2018 12:37:08)					-
- New project created successfully						

3) 烧录芯片, Target -> Production Programming

	w p 🗖 🔲 🔀	D:\Spintrol Work\SpintrolSVN\CHAM\A0\AE\01_FIRMWARE\Customer\trunk\Project\0_Ex
Name	Value	Address: [0x10000000 [x1 <u>x2</u> x4]
Host connection	USB [Device 0]	Address 0 1 2 3 4 5 6 7 8 9 8 B C D E F ASCI
Target interface	SWD	
Init SWD speed	4000 kHz	
SWD speed	4000 kHz	
HOL	C 11 10000100	
I Cere	Spintrol SPU2168	10000030 7D 07 00 10 00 00 00 53 0A 00 10 65 0A 00 10 }Se
Endian	Little	10000040 2B 0A 00 10 31 0A 00 10 DD 05 00 10 F5 0A 00 10 +1
Check core ID	No	10000050 23 0A 00 10 21 0A 00 10 55 0A 00 10 57 0A 00 10 #
Use target RAM	16 KB @ 0x20000000	10000060 59 0A 00 10 5B 0A 00 10 5D 0A 00 10 5F 0A 00 10 Y[]
		10000070 8F 0A 00 10 61 0A 00 10 27 0A 00 10 25 04 00 10a'
Flash memory	Internal bank 0	L-Flach V6 35d X 10 10 3 5 7 9
Base address	0x1000000	
Flash size	512 NB	
		larget erased, programmed and verified successfully -
		Completed after 1.077 sec
		I arget erased, programmed and verified successfully - I arget erased, programmed and verified successfully - Completed after 1.077 sec I 00 10 G1k III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed and verified successfully - III arget erased, programmed arg
		I arget erased, programmed and verified successfully - 100 10 / 1 E Completed after 1.077 sec 00 10 C 2 1 E 00 10 C 2 1 E 00 10 G 2 3 5 00 10 G 2 5 00 10 G 5 5 00 10 G 5
		I arget erased, programmed and verified successfully - 100 10
		Iarget erased, programmed and verified successfully - 10
		I arget erased, programmed and verified successfully - Completed after 1.077 sec III arget erased, programmed and verified successfully - III arget erased, programmed arget erased, programmed arget erased, programmed erased, programed erased, programmed erased, programmed erased, progr
		I arget erased, programmed and verified successfully- Completed after 1.077 sec III 0
		I arget erased, programmed and verified successfully- Completed after 1.077 sec 00 10
LOG		I arget erased, programmed and verified successfully- Completed after 1.077 sec Image: Image
LOG - End of flash	programming mains nuclear 1	I arget erased, programmed and verified successfully - Completed after 1.077 sec Image:
LOG - End of flash - Viab progr 0x1000000	programming mming performed for 1 0x1000DFF (7 Sect.	I arget erased, programmed and verified successfully- Completed after 1.077 sec III 0 IIII 0 III 0 <
End of flack - End of flack - Start of ver	programming mming performed for 1 Ox10000DFF (7 Secti fying flash	I arget erased, programmed and verified successfully- Completed after 1.077 sec Image (3584 bytes)
LOG - End of flack Flack progra- - Start of ver- - Start of ver- - End of verif	programming maning performed for 1 Ox100000FF (7 Secte ifying flash tive verify function (ving flash	Iarget erased, programmed and verified successfully- Completed after 1.077 sec 00 10 G1KM 00 10 G2 00 10 G40
I LOG - Ind of flash - Flash progr - Ok10000000 - Start of ver - Start of	programming mming performed for 1 0x10000TPF (7 Sector ifying flash tive verify function e ying flash toring	I arget erased, programmed and verified successfully- Completed after 1.077 sec III and a 120 120 120 120 120 120 120 120 120 120
LOG - End of flash - Start of veri - Start of veri - Start of veri - Start of rest - End of rest	programming mming performed for 1 Ox10000DFF (7 Secti fying flash tive verify function o toring ring	Iarget erased, programmed and verified successfully- 00 10 GIKM Completed after 1.077 sec 00 10 GIKM 00 10 GIKM 00 10 GIKM 00 10 JKM 00 10 GKO 00 10 GKO 00 00 00 00
LOG - End of flack - Nach of flack - Start of veri - Start of veri - End of veri - End of rest - The of rest - End of	programming mming performed for 1 Ox100000FF (7 Secter ifying flash tive verify function of toring it sequence it requence schulty	Iarget erased, programmed and verified successfully- Completed after 1.077 sec 00 10 GIKM 00 10 GKO 01 10 GKO 02 10 GKO 03 10 GKO 04 10 GKO 05 10 GKO 05 10 GKO 06 10 GKO 07 GKO 08 10
Derivities - Ind of flash - Isah progra- - Oku1000000 - Start of ver - Start of ver - Start of rer - The of rest - The of rest - Target erise	programming mming parformed for 1 Ox10000TPF (7 Sector ifying flash tive verify function of ying flash tive verify function ying it sequence lized successfully d, programmed and veri	Iarget erased, programmed and verified successfully - 00 10

4. ISP 串口工具概述

用户可以使用 ULINK 工具将应用程序下载到芯片中。但是这种方式不支持代码的保护等功能。Spintrol 公司提供了专门的 SPINTROL ISP 工具,帮助用户通过 UART 实现代码的下载、调试以及加密保护等功能。

SPINTROL ISP 工具运行推荐的环境配置为:

- Windows 7 操作系统
- Microsoft .NET Framework 4.0 及以上版本

如果用户电脑安装的操作系统是 Windows XP, 用户需要到 Microsoft 的官网下载并安装 Microsoft .NET Framework 4.0 组件。否则,无法运行 SPINTROL ISP 工具。

🚺 Flash Download Tool v2.	1.1				
Device SPC1068 - Po	rt COM1	8 🕞 🚺 🗳 🏟	↓ ② 🗾 🛈 工具栏		
Program Chip Vart Communi	cation				
Program File Prog D:\SpintrolAE\HengFang Select File Download Options Down ③ Download to Flash Code Info Start: 0x1FFF6000	ram文化 \spc1068\J ▼ nload选 ⑤	Project\Examples\f Auto Reload File 项 Download to SRAM e: Ox0000126C	Security选项 12345 Security Mode Password Gen Encrypt 代码加密 KEY0: KEY4: KEY1: KEY5: KEY2: KEY6: KEY3: KEY7: Encrypt Key Gen		
Ta Tino	Lorral	Harrage			
10 11me	Level T-f-	message	50		
99 16:55:52 579	Info	Rx: 79	35		
100 16:55:52 579	Success	Verify OK!			
101 16:55:52 579	Info	Tx: 32 CD			
102 16:55:52 595	Info	Rx: 79	Log窗口		
103 16:55:52 595	Info	Tx: 02 FD			
104 16:55:52 611	Info	Rx: 79			
105 16:55:52 617	Success	Download code to F.	lash successfully!		
106 16:55:52 617	Info	Set BOOT PIN high	and reset chip to run!		
			4		
COM18 = [38400,8,None,]] Comp	lete! 状态栏	100%		



图 1-1: SPINTROL ISP 工具界面

用户运行 SPINTROL ISP 工具后,可以看到如图 1-1 所示的界面。在图 1-1 中, 每个功能组件都用红色矩形框进行了标记。下面就逐一介绍各个功能组件的功能及 使用方法。

4.1. 工具栏

工具栏 Device 列表中包含 ISP 工具目前能够支持的芯片名称, Port 列表中包 含用户电脑上连接的所有可用的串口信息,用户需要从中选择所需的串口。工具栏 中其他图标功能说明见表 1-1。

Toolbar Icon	Description
0	表明串口处于关闭状态,单击该按钮则会打开相应的串口,同时该图标变为🕛
0	表明串口处于打开状态,单击该按钮则会关闭相应的串口,同时该图标变为▶
2	该按钮在串口关闭状态下有效,单击该按钮,ISP 工具会重新搜索所有可用的串口
Ø	设置当前串口的通信参数,单击该按钮会弹出图 1-2 所示的对话框
4	下载程序
2	清空 Log 窗口
X	表明 Log Auto-Scroll 功能开启,单击该按钮则会关闭 Auto-Scroll 功能,同时图标变为
X	表明 Log Auto-Scroll 功能关闭,单击该按钮则会开启 Auto-Scroll 功能,同时图标变为
i	单击该按钮会弹出 ISP 工具的相关说明信息

表 1-1: 工具栏图标功能说明

4.2. 状态栏

状态栏中各个图标的含义如图 1-2 所示。





图 1-2: 状态栏图标含义说明

图 1-2 中所示串口状态图标表示串口已被成功打开,串口其他状态的图标如表 1-2 所示。

Status Icon	Description
•	表明 ISP 工具未发现可用的串口
0	表明串口状态未知,一般 ISP 工具打开后为该图标
8	表明串口打开或者关闭时出错
0	表明串口处于关闭状态
0	表明串口处于打开状态

表 1-2: 串口状态图标说明

此外, 串口通信参数[38400, 8, None, 1]含义为: 波特率 38400bps、Data Bits 为 8、无校验、Stop Bit 为 1。在下载程序时, UART 参数配置为 <u>8 Data Bits、None</u> <u>Parity、1 Stop Bit</u>。因此, 用户需要确保串口相关参数的配置正确。通信波特率 参数可由用户根据需要自行设定(默认设置为 38400, 如果通信不正常, 请联系技 术支持确认波特率)。用户可以单击工具栏上的图标**登**进行串口参数的设置, 对话 框如图 1-3 所示。



图 1-3: 串口通信参数设置

4.3. Program 文件

Program 文件选项用来选择要下载到芯片中的 HEX 格式文件。单击图 1-1 中的 Select File 按钮, 会弹出如图 1-4 所示的文件对话框。用户选中相应的 HEX 文件, 确认即可。另外,如果勾选 Auto Reload File,则 ISP 工具在每次下载程序时,都 会重新装载选中的 HEX 文件并提取文件中的数据,如果未勾选,那么每次下载到芯 片中的数据都是第一次选择 HEX 文件时的数据,即使后面 HEX 文件被更新过,也不 会被下载到芯片中。

▶ 打开	- + 1 +	73.0	×
🔾 🗢 📕 « Project	▶ Template ▶ Objects	▼ 4 / 搜索 Object	<u>م</u> ع
组织 ▼ 新建文件夹			•
🔛 视频 🔦	名称	修改日期	类型
	🔊 Project.hex	2016/5/21 19:25	HEX 文件
● 文日			
INTERPORT IN INTERPORT INTERPORT IN INTERPORT INTERPOR	< [Þ
文件名	(N): Project.hex		d) •
		打开(0)	取消

图 1-4: 添加 Program 文件对话框

4.4. 代码信息

代码信息是用来给用户提供程序的起始地址以及程序大小(字节)信息。这些信息都是从选中的 HEX 文件中提取的。

4.5. Log 窗口

Log 窗口用来显示 ISP 操作信息、错误提示以及程序下载等信息。用户需要特别留意黄色和红色背景的信息:黄色背景 Log 代表警告信息;红色背景 Log 代表错误信息。

Download 选项、Security 选项以及代码加密将在下面的章节进行介绍。

4.6. 下载程序

SPINTROL ISP 工具提供两种 Download 选项: Download to Flash 和 Download to SRAM。Download to Flash 意味着用户的程序会被下载到芯片的内部 Flash; 而 Download to SRAM 则意味着用户的程序被直接下载到芯片内部 SRAM 中。但是有些 芯片不支持 Download to SRAM 这种模式。

(1)如果用户选择 Download to Flash 选项,那么当用户按下 ISP 工具上的下载
按键后,Boot Loader 会将通过 UART 接收到的数据写到 Flash 中。当程序下载
成功后,用户需要将 Boot Pin 接高电平,然后按下 RESET 按键,Boot Loader 就会
将 Flash 中的程序装载到 SRAM 中执行。

(2)如果用户选择 Download to SRAM 选项,那么当用户按下 ISP 工具上的下载↓按键后,Boot Loader 会将通过 UART 接收到的数据写到 SRAM 中。当程序下载成功后,Boot Loader 就会直接执行 SRAM 中的程序。

详细下载过程如下:

1) 硬件 ISP 模式选择

使用 SPINTROL ISP 工具下载程序时,需要设置 Boot Pin 管脚,然后按下 RESET 按键,此时 Boot Loader 就进入程序下载模式:



芯片	Boot PIN	TRSTn	RESET 按键
SPC1068\SPD1078	GPI00 需要拉低	TRSTn 需要拉低	最后按下 reset 按键,进入下载模式
SPC1158\SPD1148	GPI040 需要拉低	TRSTn 需要拉低	最后按下 reset 按键,进入下载模式
SPC1168\SPD1178\SPD1188			
注: 详细可以查看芯片的 d	latasheet 的 Boot	mode 章节,具体管	脚参考 Pinout and pin description 章节



SPINTROL ISP Tool	v2.3.0	4	- 🗆 X
Device SPC1168 -	Port COM10 -	🏵 🔶 😫 🔮 🖉 ≚ 🔇)
rogram Chip Varte Com	nmunication Param ⁴ Decode	Memory Area	
Program File	2	Download Options	
or Sample codes VO3	30\Objects\PICA_FWLib.hex	💿 Program to Fla	sh 🗹 Jump and Run
		Program to SRA	^M 3
🗹 Auto-Reload	🗌 Multi-Zone	1	
Code Info		Erase Options	
Start: 0x10000000	Size: 0x00007C0	C O Erase Chip	Erase Sectors
Start:	Size:		
Operations GetID GetC	onfig Erase NVR	Write NVR Read NVR	• NVR3 O NVR4
Operations GetID GetC Id Time	Config Erase NVR	Write NVR Read NVR	• NVR3 O NVR4
Operations GetID GetC Id Time 1 17:31:55 51	Config Erase NVR Level Message 2 Success Successful	Write NVR Read NVR	● NVR3 ○ NVR4
Operations GetID GetC Id Time 0 17:31:55 51 1. 选择目 2. 选择言 3.选择下 4 史口志	ionfig Erase NVR Level Message 2 Success Successful 目标芯片 需要下载的文件 S载到Flash还是下载到SF	Write NVR Read NVR ly Open COMIO RAM. (有些芯片不支持下	● NVR3 ○ NVR4 载到SRAM)
Operations GetID GetC Id Time 0 17:31:55 51 1. 选择目 2. 选择評 3.选择下 4.串口面	eonfig Erase NVR Level Message 2 Success Successful 目标芯片 需要下载的文件 S载到Flash还是下载到SF 2置 (选择相应Port端口	Write NVR Read NVR Ly Open COMIO RAM. (有些芯片不支持下 L,再点击串口配置按钮进行	 NVR3 NVR4 教到SRAM) 方相应的配置)
Operations GetID GetC Id Time 0 17:31:55 51 1. 选择目 2. 选择言 3.选择下 4.串口配 < 5.打开串	eonfig Erase NVR Level Message 2 Success Successful 目标芯片 需要下载的文件 S载到Flash还是下载到SF 2置 (选择相应Port端口	Write NVR Read NVR Ly Open COM10 RAM. (有些芯片不支持下 1,再点击串口配置按钮进行	 NVR3 NVR4 载到SRAM) 訪相应的配置)
Operations GetID GetC Id Time 1.选择目 2.选择言 3.选择下 4.串口酝 c 5.打开串 6.点击下	ionfig Erase NVR Level Message 2 Success Successful 目标芯片 需要下载的文件 S载到Flash还是下载到SF 2置 (选择相应Port端口 コロー S载按钮开始下载	Write NVR Read NVR Ly Open COM10 RAM. (有些芯片不支持下 D,再点击串口配置按钮进行	 NVR3 NVR4 载到SRAM) 方相应的配置)

4.7. Security 功能

Security功能主要是用来控制芯片的 Debug 模块。如果用户勾选了图 1-1 中的 Security选项,同时用户还需要设定一个最大长度为 32 字节的密码,那么 ISP 工 具会在下载程序时将这些信息传递给芯片,芯片会将这些信息加密后存储于芯片内 部。当芯片再次启动后, Debug 模块就会处于关闭状态, 防止其他人员通过 Debug 接口获取芯片内部的数据。

当芯片的 Security 功能开启后,如果需要更新芯片中的程序,那么用户需要事 先在 ISP 工具中输入密码。在启动程序下载功能时,ISP 工具会将该密码传递给 Boot Loader 进行校验。如果密码校验正确,那么 Boot Loader 会继续进行程序的下载; 如果密码校验失败,那么 Boot Loader 就会拒绝程序下载请求并擦除芯片 Flash 中 的原有数据。

注意: (1) Security 功能仅在用户选择 Download to Flash 选项时有效;

(2)当 SPC1068 Security 功能开启后, Boot Loader 会拒绝 Download to SRAM 请求;

(3) 密码格式限定为英文大小写字母和数字 0~9。

4.8. 代码加密功能

代码加密功能是用来对用户的原始程序进行加密,防止其他人员通过反汇编技术获取用户源程序。如果用户勾选了图 1-1 中的 Encrypt 选项,那么 ISP 工具会在下载程序时将加密用的 KEY (32 字节)传递给芯片,Boot Loader 会利用这些 KEY 将用户程序进行加密后写入 Flash 中。程序下载完成后,芯片会将这些 KEY 加密后存储于芯片内部。当芯片再次上电后,Boot Loader 会将 Flash 中的程序进行解密,然后装载到 SRAM 中,最后执行 SRAM 中的程序。

注意: (1) Encrypt 功能仅在用户选择 Download to Flash 选项时有效;

(2)密钥 KEY 为 32 字节数据,如果 Boot Loader 解密失败, Flash 中原有的程序数据会被擦除。



4.9. UART 通信交互

SPINTROL ISP 工具为了方便用户使用 UART 调试程序,还特别集成了 UART 通信 交互功能,如图 5-1 所示。



图 5-1: ISP 工具 UART 交互界面

©2014-2025, Spintrol Limited Corporation

从图 5-1 可以看出, UART 交互界面主要分为两大部分:上半部分为 Receive 选项,下半部分为 Transmit 选项。

Receive 选项包含以下功能:

- (1) Receive 窗口:接收芯片 UART 发送出来的数据,以 ASCII 格式进行显示;
- (2) Clear 按钮: 单击该按钮则会清空 Receive 窗口中的信息;
- (3) Save 按钮:单击该按钮则会将 Receive 窗口中的信息以文本形式保存在本地;

(4) Stop 勾选项:如果勾选该选项, ISP 工具会停止显示从芯片 UART 接收到的数据。

Transmit 选项包含以下功能:

- (1) Transmit 窗口: 接收用户要发送的数据;
- (2) Clear 按钮: 单击该按钮则会清空 Transmit 窗口中的数据;
- (3) Send 按钮: 单击该按钮则会将 Transmit 窗口中的数据发送给芯片的 UART;

(4) HEX 数据:如果勾选该选项,则表明 Transmit 窗口中的数据是 HEX 数据,用 户输入数据时,需要用空格将各个 HEX 字节数据分开;如果未勾选该选项,则表明 Transmit 窗口中的数据为字符数据;

(5) 换行符:

5. 蓝牙调试





5.1. 蓝牙模块使用

蓝牙模块是为了方便隔离调试,推荐购买渠道是淘宝店搜索 HC05 模块,就可以 看到相应的模块,图 5-1 就是 HC05。购买时候请跟淘宝店主确认模块波特率以及 模块密码。多数默认密码为 1234,波特率为 38400。蓝牙连接名称可能为 HC05, 或者 HC01, HC02 字样。

5.2. 转接板

如果使用 SPINTROL 的开发板,接口可以接入转接板,可以直接使用转接板 跟蓝牙相接入。请确保转接口可以接入转接板。图 5-2.1 为转接模块,



©2014-2025, Spintrol Limited Corporation





图 5-2.2: 蓝牙与转接口



5.3. 波特率设置

蓝牙模块如果与程序的串口波特率不一致,会导致通信不上。需要进行修改相应的波特率。 使用串口模块与蓝牙模块相连接,进行设置波特率。





图 5-4: 蓝牙模块与 USB 转串口模块

蓝牙修改波特率

蓝牙的硬件连接如下:

蓝牙模块	USB 转串口模块
STATE	不需要接
RXD	TXD
TXD	RXD
GND	GND
VCC	VCC
KEY	不需要接

按下蓝牙的按键,然后进行上电,上电的时候后可以松开按键。此时蓝牙灯慢闪,通过串口工具 发送相应的指令可以进行波特率修改。指令如下:

串口指令	是否需要发送换行符	功能
AT+UART=57600,0,0	是	修改波特率为 57600
AT	是	返回 OK 表示通信正常
AT+PSWD=1234	是	设置蓝牙连接密码为 1234
AT+NAME=SPINTROL	是	设置蓝牙名称为 SPINTROL

注: 串口的波特率设置 38400,可以使用发送 AT+换行符进行测试是否通信 OK。



5.4. 配置软件波特率与蓝牙波特率一致



5.5. 添加蓝牙串口进行串口下载





蓝牙和其他设备 ^{添加设备}	
+ 添加蓝牙或其他 添加设备 施用他的公公司打开并可能发现 在工在发展现在1400	
蓝牙 · · · · · · · · · · · · · · · · · · ·	
打开此选项,设备 SPI	
前入 sg57600 的 PIN。 「T232R USB UA 1234 ×	
USB Receiver 连接 取消	
图 5-4.2 win10 添加蓝牙名称为 sq57600 的蓝牙	



蓝牙按照图 5-4.1,图 5-4.2,图 5-4.3 操作后,可以看到再串口上多出了两个串口设备,此时就可以按照 ISP 串口工具的 4.5 章节进行下载程序。 注意:选择哪个 com 口,此时可以先进行连接,如果蓝牙能够慢闪,证明蓝牙连接成功。

6. 软件框架



软件的算法框架大概如下框图。









7. 数据采集功能

多少个PWM周期采集一 次数据
采集变量的数据总量
采集变量的个数
采集变量初始化
开始采集数据
开始打印数据
所需要变量 lgBuffer);进行采集数据。 居。

图 7-2 数据采集功能



8. 电机参数

所	在文件: motor_sys_config_basic.h	5
	U8_MOTOR_POLES	电机极数,一般为2极(1对极)、4 极(2对极)、6极(3对极)。。。。
	F32_MOTOR_REAL_PHASE_R_OHM	电机相电阻阻值,单位为欧姆
	F32_MOTOR_REAL_PHASE_LD_H	电机D轴相电感,单位为亨利
	F32_MOTOR_REAL_PHASE_LQ_H	电机Q轴相电感,单位为亨利
	MOTOR_LOWSPEED_BEMF_FREQ_HZ	电机反电势的频率
	MOTOR_LOWSPEED_BEMF_VL2L_VOLT	电机两个波峰之间的电压差
	BEMF_COEF_AUTO	是否打开反电势系数自动计算(根 据经验数据),1:使用;0:禁用。 默认为0,当条件不允许的时候,可 以打开这个选项。
	图 8-1: 国	电机参数对应的宏





9. 电机保护功能

用户参数	描述	
参数调试档案位置:motor_sys_config_basic.h		
STALL_DETECTION_EANBLE	是否启用堵转保护	
OVER_CURRENT_ENABLE	是否启用过流保护	
OVER_VOLTAGE_ENABLE	是否启用过压保护	
UNDER_VOLTAGE_ENABLE	是否启用欠压保护	
PHASE_LACK_DETECTION_ENABLE	是否启用缺相保护,其中默认开启的是运行过程中的缺相保护,启 动前的缺相保护取决于是否有相电压检查电路	



用户参数	描述
参数调试档案位置:motor_sys_config_basic.h	
32_MOTOR_SYS_PHASE_OVERCURRENT_A	过流保护相电流大小,当任意相电流大于这个值,触发过 流保护
32_MOTOR_SYS_OVER_VOLTAGE_V	过压保护电压大小,当电压大于这个值,触发过压保护
32_MOTOR_SYS_UNDER_VOLTAGE_V	欠压保护电压大小,当电压小于这个值,触欠过压保护

10. 开环电流采集测试

- 1) 首先需要了解第7章节的数据采集功能,然后开启开环 VDVQ 控制的相关宏,具体相关的宏参考图 10-1:电机电流验证的相关宏。
- 2) 采集到相应的结果后参考图 10-2: 电机电流采集验证。

正常的情况,需要看看从 5% - 30% - 70% - 95%的电压占空比下,电流采样波形是否连续,在这过程中,电流可能会非常大,可以通过降低母线电压,或者使用电阻较大的电机来做对应测试。

- 3)如果电流采集正常,则可以通过 MOTOR_RAMP_ENABLE为1,PURE_VOLTAGE_RAMP_ENABLE设置为0, 进行开环电流环控制,通过电流增加或者减少指令操作(查询第15章节串口指令集),通 过示波器观测电流是否增大减少。如果正常,则开环电流测试正常。
- 注意调试完成后, 启用的宏需要关闭。
 MOTOR_RAMP_ENABLE 与 PURE_VOLTAGE_RAMP_ENABLE 务必需要恢复为 0。



采集三相电流以及电压变量进行excel处理。 motor_sys_config.h	
MOTOR_RAMP_ENABLE	
F32_MOTOR_RAMP_UP_CURRENT_START_A	额定电流20%
F32_MOTOR_RAMP_UP_CURRENT_FINAL_A	额定电流20%
MOTOR_RAMP_UP_FORCE_TIME_MS	推荐5000ms
MOTOR_RAMP_UP_SPEED_START_RPM	0
MOTOR_RAMP_UP_SPEED_FINAL_RPM	额定转速10%
MOTOR_RAMP_UP_SPEED_TIME_MS	推荐5000ms
PURE_VOLTAGE_RAMP_ENABLE	开环电压控制使能, 需要使能1
F32_MOTOR_RAMP_UP_VOLTAGE_DUTY_START	初始Duty1%, 建议7%
F32_MOTOR_RAMP_UP_VOLTAGE_DUTY_FINAL	最终pwm Duty , 建议7%

图 10-1: 电机电流验证的相关宏



11. 开环算法适用验证

再第 10 章节验证电流正常后,可以采集如下两个变量,进行算法适用验证。算法不适用, 可以联系 SPINTROL 的 fae,让其帮忙修改算法的频率滤波参数进行适配。





12. 闭环调试

闭环电流环测试正常后,现在开始外环调计 外环包含speed, power, duty, voltage,这里 相关宏配置如下:	试。 且使用 <u>速度环</u> 来调试参考。
MOTOR_RAMP_ENABLE	开环关闭使能0
PURE_VOLTAGE_RAMP_ENABLE	0
TORQUE_LOOP_ENABLE	0
SPEED_LOOP_ENABLE	1
POWER_LOOP_ENABLE	0
DUTY_LOOP_ENABLE	0
VOLTAGE_LOOP_ENABLE	0
CURRENT_LOOP_BANDWIDTH_HZ	电流环带宽,大小为载波的1/20,1/60之间
速度环PI参数: PI_CONTROL_SPEED_I_GAIN, PI_CONTROL_SPEED_P_GAIN DUTY环的PI参数: PI_CONTROL_DUTY_I_GAIN PI_CONTROL_DUTY_P_GAIN 功率环的PI参数: PI_CONTROL_POWER_I_GAIN PI_CONTROL_POWER_P_GAIN	
图 12-1: 电机算法对比验证	





13. 电机启动调试

如图 13-1:电机启动参数配置,其中 fRs fLs fOneOverFlux 等为电机参数。 Alpha0 和 Lambda 是观测其中的增益。所以电机参数确认正常的情况下,Alpha0 和 Lambda 保 持默认的情况下启动参数就配置完了。





参数配置完后,进行启动调试。

配置完lowspeed的参数,下面进行闭	环电流环调试。
MOTOR_RAMP_ENABLE	0
PURE_VOLTAGE_RAMP_ENABLE	0
TORQUE_LOOP_ENABLE	1
SPEED_LOOP_ENABLE	0
POWER_LOOP_ENABLE	0
DUTY_LOOP_ENABLE	0
> VOLTAGE_LOOP_ENABLE	0
CURRENT_LOOP_BANDWIDTH_HZ	电流环带宽,大小为载波的1/20,1/60之间
串口指令给定相应的电流('[',']','o' 试。如果启动不成功,则尝试打开平滑启z	,'p'),然后给定电机启动指令't',进行启动测 动。
如果启动还是各种不顺,可以按修改下面下 myMotor[0].sThetaMerge.u16ToSMOThresh myMotor[0].sThetaMerge.u16ToLowThresh u16ToSMOThreshold的值可以修改为额定转	两个阀值。 nold (HZ) nold (HZ) 速频率的20%附近。
图 13-2:	电机启动调试
如果启动还不是很顺畅,可以进行再调节平滑。	电流:

MOTOR_STANDSTILL_SMOOTH_ENABLE	是否打开平滑启动,打开:1, 关闭:0	
STANDSTILL_SMOOTH_MAX_CURRENT_A	平滑启动电流	
STANDSTILL_SMOOTH_DONE_SPEED_THRESHOLD_RPM	平滑启动结束的速度阈值	
用电流钳观察相电流,把这个值从小到大调节,直到保证每没 过额定时候的最大电流。	欠都可能快速启动。这个值不要超	
图 12 2 . 由 扣 亚 裡 白 动 浬	4:	
四155. 电机工作内列调风		

14. 串口指令集



串口命令	描述
t	电机启动
+ p	增加给定(电流/速度/功率): +0.1A/+100RPM/+1.0w
- 0	减小给定(电流/速度/功率): -0.1A/-100RPM/-1.0w
I	增加给定(电流/速度/功率): +0.01A/+10RPM/+0.1w
1	减小给定(电流/速度/功率): -0.01A/-10RPM/-0.1w
8	增加给定(电流/速度/功率): +0.001A/+1RPM/+0.01w
9	减小给定(电流/速度/功率): -0.001A/-1RPM/-0.01w
5	电机停止
r	打印记录数据
q	打印运行相关状态数据

15. MemDump 工具使用

MemDump 工具是用于把烧入芯片的固件读取出来,然后与现有的固件进行 CRC 判断,从 而可以判断固件是否烧写正确或者完整。

15.1. 相关工具

硬件	usb 转串口工具 / MiniSPLink 工具 / SPC1068 开发板(目标板子)
软件	MemDump.exe
MiniSPLink 的固件	MiniSPLink_MemDump.hex
测试固件	测试的 hex 文件

表 15-1:相关工具

15.2. 更新 MiniSPLink 的固件

需要把 MiniSPLink_MemDump.hex 下载进到 MiniSPLink 工具。 接线如下:





接线完成, usb 转串口接入电脑, 使用 ISP 工具进行下载, 选择相应的串口, 波特率使用 38400。 注意: MiniSPLink 的芯片为 1068, 需要把 boot0 拉低, 才能进行下载, 所以需要跳冒进行短接 才能进行下载.




15.3. 接入开发板

1) 接线如下

一端接 USB 转串口,一段接口接板子 SWD。 接完后按一下 MinSPLink 的复位按键,绿灯常亮表示连接正常。 注意: MiniSPLink 的 boot0 不能短接。 如图:





2) memdump 配置

打开 memdump 软件,选择相应的串口,波特率选择 115200,选择相应的芯片,以及设置 相应的地址。



Memory Dump Tool v1.0.0 Chig SPC1068 Port COM25 1	ö 🔞 🕒 🗖 🗙 🛛 🕇	- 🗆 X
Read Control Start: 0x1fff8000 Sire: 0x1000	■ erial port settings for [□ × 1. 设置相应的串口,波特率使用115200 Fort: COM25 ✓ OK Baud rate: 115200 Date: 8 ✓	DD bps, 8 data bits, None, 1 stop 🕠
2. 远择相应的芯片 Address 0 1 2 3 4 5 6 7 3. 设置相应的地址,以及需要的大小	Parity: None V Stop: 1 V .d	
图 15	·3.2 : Memdump 工具设置	

3) 连接串口

配置完串口后,选择连接串口.

4) 复位 MCU

如图点击红色的按钮,在然后选择 Reset Target MCU.,此步很重要。



Read Control								R	eset	MiniS	PLink	U		40 F1	0 F2 01 12 10 43 48 62 70 4 F 1F 2D E9 F0 47 23 49 05 4 8 63 22 49 08 63 23 4C 22 4	7 00 A0 3 05 20 8 E0 64	00 4	0 12 3 22	E5 49 F7	,			
Start: Ux1	III0.	00		21	ize:	08100	10			1	R	eset	Both	1	-		D6 64 B3 06 E0	6 FF 00 20 02 23 A1 68 69 0 4 28 F8 DB 99 46 1C 4B 00 2 3 F9 28 60 5F 6A DB 6A 08 4 8 80 5F EA 88 78 FA D5 D4 F 9 44 08 FB 07 F8 01 EB E8 3	7 FC D5 1 OA 46 5 4F F4 3 98 80 1 D4 F8	E3 6 B3 H 80 6 C8 H 9C 8	10 40 19 20 1A D4 13 0C	1C C0 F8 08 F8	
										复	位M	CU		1			00 40 81 00	C 90 C8 F3 0C 08 B0 44 08 F 0 1C 50 45 E3 DB 88 12 E8 8 D E8 F0 87 00 00 00 71 00 4 0 40 C6 25 49 CA 00 A0 00 4 0 F8 AC 1F 41 F0 80 01 40 F	03 F8 7 90 12 0 00 72 0 4C 0A 8 9C 1B	02 H A5 H 00 4 00 2 01 6	B E8 8 44 0 00 0 29 8 21	32 00 73 48 F0	
Address	0	1	2	3	4	5	6	7	8	9	a	b	с	4	e	f	^ 20	0 01 x: 21 01 00 8E FF 1F 00 01	30 00 B	0			
lfff8ec0	44	06	c5	fl	01	05	05	eb	45	05	05	eb	44	04	cl	£3	Ra 50	x: 61 01 60 25 48 01 68 41 0 F8 84 1C 41 F4 80 41 40 F	70 20 0 8 84 1C	1 40	F8 8	4 1H F8	8
lfff8ed0	00	15	c5	fl	01	05	65	£3	9f	01	0d	36	bl	40	02	25	70	C 2F 0B 7C 63 F3 D6 42 02 6 2 32 02 60 8B 7C 02 68 63 F) 4B 7C	02 6	8 63	F3	
lfff8ee0	0d	34	a5	40	0b	43	2b	43	43	ea	42	61	02	4 a	02	eb	81	B 8A 02 68 63 F3 1F 42 02 6	CB 8A	02 6	8 63	F3	
lfff8ef0	80	00	cl	64	70	bd	00	00	00	a 0	00	40	70	b5	04	46	01	r 02 04 CO 05 85 02 68 63 r 1 68 62 F3 08 01 40 F8 8C 1	3 51 22 3 01 68	21 H	10 4A	61	
lfff8f00	15	46	0e	46	06	20	00	fO	al	f8	10	4a	42	£2	10	73	01	1 F1 40 71 01 60 01 68 21 F 1 60 01 68 41 F4 00 61 01 6	4 CO 61 0 01 68	01 H	5 00	71 41	
lfff8f10	b0	fb	f2	fl	b0	fb	£2	fO	68	43	71	43	ь0	fb	£3	fO	01	1 60 70 47 00 00 00 B0 00 4 0 20 70 B5 49 18 C1 83 80 0	0 00 A0	00 4	10 4C	OA F3	
lfff8f20	bl	fb	£3	fl	40	lc	7f	29	00	d3	7£	21	3f	28	00	d3	C	0 04 4F F0 00 43 05 EB 45 0	5 E3 40	04 H	B 44	04	
lfff8f30	3f	20	07	4a	02	eb	84	02	d3	6c	61	£3	8c	13	d3	64	Ci	6 EB 44 06 C5 F1 01 05 05 E 1 F3 00 15 C5 F1 01 05 65 F	3 45 05 3 9F 01	0D 3	16 B1	40	
lfff8f40	dl	6c	60	£3	05	01	dl	64	70	bd	00	00	a0	86	01	00	02	2 25 0D 34 A5 40 0B 43 2B 4 2 EB 80 00 C1 64 70 BD 00 0	3 43 EA	42 6	1 02	4A 85	
lfff8f50	00	a0	00	40	70	47	00	00	10	b5	07	4c	60	78	40	bl	04	4 46	00.00 8	1	2013	155	
lfff8f60	00	20	60	70	e0	69	80	47	04	fl	2c	00	01	88	49	lc	R	x: 61 15 46 0E 46 06 20 00	FO A1 F	8 10	4A 4	2 F2	2
lfff8f70	01	80	02	48	60	61	10	bd	00	00	00	20	bd	8f	ff	lf	10 F3	0 73 BO FB F2 F1 BO FB F2 F 3 F0 B1 FB F3 F1 40 1C 7F 2) 68 43 9 00 D3	71 4 7F 2	13 BO	FB 28	
lfff8f80	70	b5	02	23	00	25	29	46	10	e0	00	bf	00	eb	cl	02	00	0 D3 3F 20 07 4A 02 EB 84 0 3 64 D1 6C 60 F3 05 01 D1 6	2 D3 6C 4 70 BD	61 H	3 8C	13	
lfff8f90	d6	68	54	68	b4	42	08	dd	50	f8	31	50	56	60	96	68	Ő	1 00 00 A0 00 40 70 47 00 0	0 10 B5	07 4	IC 60	78	
lfff8fa0	40	£8	31	60	c2	e9	02	54	01	25	49	lc	99	42	ed	db	49	9 1C 01 80 02 48 60 61 10 B	0 00 00	00 2	O BD	8F	
lfff8fb0	00	2d	02	d0	5b	le	00	2b	e4	dc	70	bd	10	b5	07	4c	FI	F 1F 70 B5 02 23 00 25 29 4 1 02 D6 68 54 68 B4 42 08 D	5 10 E0 5 50 F8	00 H 31 E	JF 00 50 56	EB 60	
lfff8fc0	a 0	78	40	bl	00	20	a0	70	20	6a	80	47	04	fl	34	00	96	6 68 40 F8 31 60 C2 E9 02 5 D DB 00 2D 02 D0 5B 1F 00 2	4 01 25 8 E4 DC	49 1 70 1	C 99	42 85	
lfff8fd0	01	88	49	lc	01	80	02	48	60	61	10	bd	00	00	00	20	0	7 4C AO 78 40 B1 00 20 AO 7	20 6A	80 4	17 04	F1	
lfff8fe0	95	8c	ff	1f	10	b5	08	49	08	4b	00	20	0a	5c	40	lc	- 34	4 00 01 88 49 10 01 80 02 4 0 20 95 8C FF 1F 10 B5 08 4	9 08 4B	00 2	A0 05	50	i
lfff8ff0	03	eb	82	02	c0	b2	52	£8	94	4f	10	28	24	fO	80	04	40	0 1C 03 EB 82 02 CO B2 52 F 0 04	3 94 4F	10 2	8 24	FO	

5) 点击 read,数据读取完毕。

注意:如果 read 不成功,接线也正常的话,可能的原因为电压不够,需要外接 3.3V 电压给 MiniSPLink 小板子。





6) 保存 hex 文件



选择保存,保存后使用 beyong compare 等对比工具进行对比。



16. 持续更新中