

# 使用指南

## SPC2168 CAU 双核调试指南

## 概述

本文主要介绍了 SPC2168 的 CAU 调试方法。



# 目录

1	CAU 朴	既述	6
1.1	CAU 代	码执行过程	6
2	双核	代码烧录	7
2.1	采用 IS	P Download Tool 烧录镜像	7
	2.1.1	CPU 工程配置	8
	2.1.2	CAU 工程配置	8
	2.1.3	CPU 搬运 CAU 镜像	9
	2.1.4	烧录1	.0
2.2	采用儿	INK 烧录镜像1	.1
	2.2.1	CPU 工程配置1	.1
	2.2.2	CAU 工程配置1	.1
2.3	采用儿	INK 调试程序1	.3
	2.3.1	CPU 调试1	.3
	2.3.2	CAU 调试1	.3
2.4	JLINK រ៉	周试接口说明1	.6



# 图片列表

图	1-1:	Mailbox 架构	6
图	1-2:	Main CPU 与 CAU 框图	6
图	2-1:	SPC2168 内存映射图	7
图	2-2:	ISP Download Tool 烧录时 CPU 工程配置	8
图	2-3:	ISP Download Tool 烧录时 CAU 工程配置	8
图	2-4:	CPU 搬运程序	9
图	2-5:	ISP Download Tool	.10
图	2-6:	CAU 工程添加 Flash 算法文件	.11
图	2-7:	CAU 工程烧录地址及执行地址配置	.12
图	2-8:	CAU 工程选择 ini 文件	.15
图	2-9:	ini 文件内容	.15
图	2-10:	Debug in RAM 下"Flash Download"配置	.15



# 表格列表

表 2-	1:	JTAG 设置				16
------	----	---------	--	--	--	----



# 版本历史

版本	日期	作者	状态	变更
1	2022-01-04	-	Outdated	1. 初始版本。
C/0	2024-06-27	LemengZhou	Released	<ol> <li>修改 2.2 章节为"采用 JLINK 烧录镜像", 并将本节所有 JTAG 相关修改为 JLINK, 增加注意项,以便说明烧录所需的具体 SWD 接口。</li> <li>增加章节 2.3 采用 JLINK 调试程序。</li> <li>原章节 2.3 变为章节 2.4,更名为"JLINK 调试接口说明"。</li> </ol>



# 1 CAU 概述

SPC2168 是双核 CPU,除了主 CPU,另一颗基于 ARM CM4 的 CPU 被称为 CAU。CPU 和 CAU 之间通过 Mailbox 进行握手和共享数据,CPU 和 CAU 通过中断来相互请求,请求的消息内容通 过 Message RAM 传递。Mailbox 架构和地址映射如图 1-1 所示。



图 1-1: Mailbox 架构

### 1.1 CAU 代码执行过程

如图 1-2 所示,对于存储区域而言,CAU 只能访问属于自身的一段 RAM 控件,并没有访问 Flash 的能力,所以 CAU 的代码在存储到 Flash 之后,只能依靠 CPU 将其代码搬运到 CAU 的 RAM 区域,然后有 CPU 配置 CAU 使能位,让 CAU 开始执行其代码。

本指南后续章节将对如何烧录 CAU 程序以及调试方法作详细说明。





# 2 双核代码烧录

## 2.1 采用 ISP Download Tool 烧录镜像

ISP 工具是 Spintrol 开发的一款烧录工具,此工具的详细说明请参考《ISP 工具使用指南》。 以下将对使用 ISP Download Tool 烧录程序时,CPU 以及 CAU 的 KEIL 工程配置情况进行说明。

范例采用 Demo Code 的 DualCore\_CAU\_ToggleGPIO 以及 DualCore\_MainCPU\_ToggleGPIO 两 个示例程序,来说明 CAU 以及 CPU 的工程配置。为了方便后续说明,默认芯片的设置为使能了 Cache 以及 CAU,其内存映射如下图所示:



图 2-1: SPC2168 内存映射图

上图中, CPU 可以访问的实际物理 RAM 的地址范围为: 0x1FFF8000~0x20000000(以 CPU 的视角看此地址),共 48K 大小,其中 0x1FFF8000~0x1FFFC000,大小为 16K 的地址区间与 CAU 共享;而 CAU 能够访问的实际物理 RAM 的地址范围为: 0x0000000~0x4000、0x20000000~0x20004000(从 CAU 的视角看这两段 RAM 地址),共 32K 大小,其中 0x0000000~0x4000 这段 RAM 是与 CPU 共享的那段物理 RAM。





## 2.1.1 CPU 工程配置

由下图可知, CPU 代码的加载域和执行域是以 0x10000000 开始的长度为 0x2000 的 Flash 区域,数据堆栈区域则为 0x2000000~0x20004000 的 DRAM 区域。

ARM ARMCM4_FP	Device Target	arget 'FWLib' Output   List:	ing   User	C/C++   A	lsm I	linker   1	Debug   Vt:	ilities	×	
Operating system: None     System Viewer File:     ARMCM4.svd     Use Custom File     Use Custom File     Bead/Only Memory Areas   default off-chip   Start   Size   RoM1:   C   ROM2:   On-chip   IROM1:   Ox1000000   Ox20000   IRAM1:   IRAM1:   IRAM2:	ARM ARMCM4_FF	N	<u>X</u> tal (MHz): 12	.0	Code G	eneration Compiler:	Use defa	ult compiler versior	n 5 💌	
Read/Only Memory Areas         default off-chip       Start         Size       Startup         ROM1:       C         ROM2:       C         ROM3:       C         on-chip       C         IROM1:       0x1000000         0x20000       C         IROM2:       C         IROM2:       Image: C         IRAM2:       Image: C	System Viewer File ARMCM4.svd	None : le			□ U □ U Floatir	se Cross-N se MicroLl ng Point H	Module Optim IB łardware:	ization Big Endian Single Precision	•	
ROM1:       C       RAM1:       C         ROM2:       C       RAM2:       C         ROM3:       C       RAM3:       C         on-chip       C       RAM3:       C         IROM1:       0x1000000       0x20000       C         IROM2:       C       IRAM1:       0x200000         IRAM2:       C       IRAM1:       0x2000000         IRAM2:       C       IRAM1:       0x2000000	Read/Only Mem default off-chip	ory Areas	Size	Startup	Read/	Write Merr off-chip	nory Areas — Start	Size	Nolnit	
ROM3:       C       RAM3:       C         on-chip       on-chip       on-chip         IROM1:       0x1000000       0x20000       IRAM1:       0x2000000       0x40000         IROM2:       C       IRAM2:       C       IRAM2:       C	ROM1: ROM2:			C C		RAM1: RAM2:				
IROM1:     JX 20000000     JX 20000000     JX 4000     JX 4000       IROM2:     IRAM2:     IRAM2:     IRAM2:	ROM3:	0.1000000	0.20000	0		RAM3: on-chip	0.2000000	0.4000		
	IROM1: □ IROM2:		0x20000	C		IRAM1: IRAM2:		0,000		

#### 图 2-2: ISP Download Tool 烧录时 CPU 工程配置

### 2.1.2 CAU 工程配置

图 2-3: ISP Download Tool 烧录时 CAU 工程配置

Device Target Outpu	ut   Listing   User	C/C++   A	sm ]]	Linker   I	)ebug   Vtil	lities	
ARM ARMCM4_FP	Xtal (MHz): 12	.0	Code C	Generation Compiler:	Use defau	t compiler version	5 💌
System Viewer File:		<u> </u>		se Cross-M	lodule Optimiz	ation	
ARMCM4.svd				se MicroLl	В	Big Endian	
Use Custom File			Floati	ng Point H	ardware:	Single Precision	•
Read/Only Memory Are	eas		-Read/	Write Mem	ory Areas —		
default off-chip	Start Size	Startup	default	off-chip	Start	Size	NoInit
ROM1:		0		RAM1:			
ROM2:		0		RAM2:			
ROM3:		0		RAM3:			
on-chip				on-chip			
IROM1: 0x0	0x1000	•	$\mathbf{V}$	IRAM1:	0×20000000	0x4000	
IROM2:		0		IRAM2:			



上图的配置表示: CAU 的执行域是以 CAU 的视角看到的 0x0~0x1000 的地址空间(设置的 这个范围是以 CAU 的视角看到的地址范围,如果以 CPU 的视角看,则是 0x1FFF8000~0x1FFF9 000), CAU 代码的加载域稍后再说明,数据堆栈区域则为 0x20000000~0x20004000。

#### 2.1.3 CPU 搬运 CAU 镜像

如章节 1.1 描述, CAU 需要 CPU 将其代码搬运至属于自己的 RAM 控件中,在 DualCore\_MainCPU\_ToggleGPIO 示例程序中我们可以从代码中看到蛛丝马迹。如下图所示,● 搬运完代码之后,●使能 CAU,此时 CAU 开始执行程序。







#### 2.1.4 烧录

在 2.1.1 以及 2.1.2 已经介绍了 CPU 以及 CAU 的工程配置情况,在工程编译之后,就可以 使用 ISP Download Tool 烧录镜像。

如图 2-5 所示,当选择 Decie 为 SPC2168 后,在●和●分别选择 CPU 和 CAU 工程编译之后 的 hex 文件,然后在●手动更改属于 CAU 的 hex 文件的加载地址(CPU 的加载地址无需更改)。

E SPINTROL ISP Tool v2.3.7 - □ × Device SPC2168 • Port COM4 • ○ ↓ ◆ ☆ ☆ ☆ ☆ ☆ ③ ◎ Frogram Chip Uart Communication EEPROM Burner Memory Area
Program File       Download Options         Image: Construction of the state of th
Id Time Level Message
 COM4 = [38400,8,None,1]

图 2-5: ISP Download Tool

- 注意: 1. CPU 搬运代码起搬运的源地址是 2.1.4 节人为填写的 CAU 加载地址,而目的地址是从 CPU 角度看到的 CAU 0x0~0x4000 的地址。
  - 2. CPU 无需更改的原因是: 在 CPU 的 KEIL 工程中已经设置了加载域和执行域, 但 是在 CAU 的工程中, 设置的以 0x0 开始的地址在 ARM Cortex-M 体系结构中是 不能作为加载域的, 所以只能在 ISP Download Tool 中重新制定加载域。

至此,就可以点击运行按钮,使用 ISP Download Tool 烧录镜像。



### 2.2 采用 JLINK 烧录镜像

在开始介绍 JLINK 烧录镜像之前,建议用户首先阅读《快速上手指南》,了解 JLINK 的基本使用方法,本指南将着重介绍使用 JLINK 方式烧录的方法。

#### 2.2.1 CPU 工程配置

对于 CPU 而言,使用 JLINK 的方式进行烧录可以完全参照《快速上手指南》中的步骤进行操作,本文不再赘述。

#### 2.2.2 CAU 工程配置

在详细介绍 CAU 的详细步骤之前,首先需要把 SPC2168\_CAU.FLM 这个文件拷贝至 Keil 安装目录下的 ARM\Flash\文件夹下,另外,SPC2168\_CAU.FLM 算法文件将会把 CAU 的镜像烧录到 Flash 0x10078000~0x1007FFF 32KB 大小的区域内。

在 CAU 的工程中,在工程配置上我们需要注意以下几点:

- 添加 Flash 算法文件

如图 2-6 所示,选择正确的调试工具,然后再选择 SPC2168\_CAU.FLM 算法文件,这个文件 在 KEIL 中显示的 Flash Size 大小很大,这个无需担心,这个只是 KEIL IDE 的显示,并不会 影响烧录过程。

V       Load Application at Statup Initialization File:       Debug       Trace       Flash Download         P       Debug       Session Settings V       Debug       Trace       Flash Download         V       Debug       Session Settings V       CPU DLI:       Programming Algorithm       Start:       0x20000000       Size:       0x1000         Dalog DLL:       Parameter:       DCM DLL       PC168 CAU Flash       Device Size       Device Type       Address Range         SPC2168 CAU Flash       S24304k       On-chip Flash       00000000H - 20003FFFH         Add Flash Programming Algorithm       X       Description       Rath Size       Device Type       Address Range         SPC2168 CAU Flash       S24304k       On-chip Flash       00000000H - 20003FFFH       X         Description       Device Size       Device Type       Address Range       N         SPC168 CAU Flash       S24304k       On-chip Flash       00000000H - 20003FFFH         Add Flash Programming Algorithm       X       Description       Size: Size Size       N         SPC1168 INVE Paae       File       Add Flash Programming Algorithm       X       N         SPC1168 INVE Paae       File       Size: Size Flash       Size: Size Flash       N       N	t   Target   Output   Listing   User   C/C++   Asm   Linker Det se Smulator <u>with restrictions</u> Settings   @ Use: JJ NK/J mit Speed to Real-Time Contau   Link/Utarget Turget During Satu	bug Utilities	×
Wam f outdated Executable is k     Description     Rash Size     Device Type     Orgin       SN32F750 32KB User ROM     32k     On-chip Rash     MDK Core       SYC1168 12KB Rash     12kk     On-chip Rash     MDK Core       SPC1168 12KB Rash     12kk     On-chip Rash     MDK Core       SPC2168 312K Rash     512k     On-chip Rash     MDK Core       SPC2168 12KB Rash     512k     On-chip Rash     MDK Core       SPC3168 NUR Rash     12k     On-chip Rash     MDK Core       SPC3168 NUR Rash     12k     On-chip Rash     MDK Core       SPC3168 NUR Rash     12k     On-chip Rash     MDK Core       STM32Fbox 12kB Rash     12k     On-chip Rash     MDK Core       STM32Fbox 52kB Rash     13k     On-chip Rash     MDK Core       STM32Fbox 12kB Rash     16k     On-chip Rash     MDK Core       STM32Fbox 72kB Rash     16k     On-chip Rash     MDK Core       STM32Fbox 72kB Rash     16k     On-chip Rash     MDK Core	ad Application at Startup     Corres/Link//Trace Flash Download       zation File:     Debug     Trace       tore Debug Session Settings- * Breakpoints     Obwnload Function       * Breakpoints     Tor     © Erase Sectors       * Wach Windows 8 Performs     C Do not Erase       * Memory Display     \$ Syst       DLL:     Parameter:       DDLL:     Parameter:       DLL:     Parameter:       DLL:     PCM4	Program Verify RAM for Algorithm Start: 0x2000000 Size: 0x1000 vevice Size Device Type Address Range S24304k On-chip Flash 0000000H - 20003FFFH Add Elash Programming Algorithm	×
STM32F10k Mod densty Rash 128k On-chip Rash MDK Core STM32F10k Mod-ensty Rash 16k On-chip Rash MDK Core STM32F10k High-densty Rash 512k On-chip Rash MDK Core V C:\Keil_v5\ARM\tash\SPC2168_CAU.FLM	am if outdated Executable is k	Description         Raith Size         Device Type         Origin           SN32F780 3XK8 User ROM         33x         On-chip Rash         MDK Core           SN32F780 54K8 User ROM         64x         On-chip Rash         MDK Core           SPC1168 128k8 Rash         128k         On-chip Rash         MDK Core           SPC1168 1124k Rash         128k         On-chip Rash         MDK Core           SPC1168 1124k Rash         11k         On-chip Rash         MDK Core           SPC2168 04/0 Rash         524304         On-chip Rash         MDK Core           SPC2168 04/0 Rash         12k         On-chip Rash         MDK Core           SPC2168 04/0 Rash         12k         On-chip Rash         MDK Core           STM32F0x 16k B Rash         12k         On-chip Rash         MDK Core           STM32F0x 16k B Rash         18k         On-chip Rash         MDK Core           STM32F0x 16k Rash         64k         On-chip Rash         MDK Core           STM32F0x 16k Rash Cash Options         168         On-chip Rash         MDK Core           STM32F0x 16k Rash Cash Dations         168         On-chip Rash         MDK Core           STM32F0x 16k Rash Cash Dations         168         On-chip Rash         MDK Core <td< td=""><td></td></td<>	

#### 图 2-6: CAU 工程添加 Flash 算法文件

- 配置加载地址以及执行地址



CAU 工程中,加载地址必须为 0x00000000,执行地址,需要设置为 CAU 能够访问到的地址空间,具体 CAU 的访问地址空间范围,需要根据芯片的是否开启 CAU 以及是否开启 Cache 而定,具体细节请参考《技术参考手册》。

在 DualCore\_CAU\_ToggleGPIO 示例工程中, CAU 的加载地址以及执行地址配置如图 2-7 所示:

🔣 Options for Targ	get 'FWLib'		×	
Device Target Out	tput   Listing   Vser   C/C++	Asm   Linker   Debug   Vtilit	ies	1
ARM ARMCM4_FP		Code Generation		
	<u>X</u> tal (MHz): 12.0	ARM Compiler: Use default co	ompiler version 5	
Operating system:	Vone 💌			
System Viewer File:		Use Cross-Module Optimizatio	n	
ARMCM4.svd		Use MicroLIB	Big Endian	
Use Custom File		Floating Point Hardware: Sin	ngle Precision 💌	
Read/Only Memory	Areas	Read/Write Memory Areas		
default off-chip	Start Size Startup	default off-chip Start	Size Nolnit	
ROM1:	• • • • • • • • • • • • • • • • • • •	RAM1:		
ROM2:	O	RAM2:		
ROM3:	0	RAM3:		
on-chip	,	on-chip		
IROM1: Ox	<0 0x1000 (©	✓ IRAM1: 0x20000000	0x4000	
IROM2:	0	IRAM2:		
				1
	OKCa	ncel Defaults	Help	

当以上步骤都操作完毕,就可以使用 JLINK 的 SWD 接口烧录 CAU 镜像。

 注意: 1. 因为芯片系统架构上只有 CPU 与 FLASH 连接,本章节介绍的 JLINK 烧录程序的 方法中,JLINK 的 SWD 接口均需连接到芯片的 CPU 的调试接口(例如: SPC2168 CPU 的 SWD 接口为 GPIO48 GPIO49)



### 2.3 采用 JLINK 调试程序

在开始介绍 JLINK 烧录镜像之前,建议用户首先阅读《快速上手指南》手册,了解 JLINK 的基本使用方法。本手册将着重介绍使用 JLINK 方式调试的方法。

#### 2.3.1 CPU 调试

对于 SPC2168 的 CPU 而言,应使用 Debug On FLASH 的方式进行调试。调试步骤为,先使用 JLINK 将程序烧录进 FLASH,再将 FLASH 中的程序进行调试。其烧录及调试可以完全参照《快速上手指南》文档中的步骤进行操作,在此就不再赘述。

#### 2.3.2 CAU 调试

对于 SPC2168 的 CAU 而言,可以采用 Debug On FLASH 的方式,也可采用 Debug On RAM 的方式进行调试,两者的区别体现在:

- 调试接口

Debug On FLASH 的方式需要通过 CPU 的调试接口烧录程序,再通过 CAU 的调试接口调试 程序。Debug On RAM 的方式只需通过 CAU 的调试接口调试程序。

- 代码是否固化

Debug On FLASH 方式调试的程序会固化到 FLASH 上。Debug On RAM 的代码,不会固化在 FLASH 上,芯片掉电后,RAM 中的程序会消失。

对于两种 CAU 的调试方法,对于应用开发人员,有如下使用建议:

- CAU 代码无需频繁修改

对于只是监测代码逻辑以及变量值,CAU代码无需频繁修改的情况,应使用 Debug On FLASH 的方式进行调试。

- CAU 代码需频繁修改

对于需要根据每次调试不同反馈状态,进行 CAU 代码频繁修改的情况,应使用 Debug On RAM 的方式进行调试。

注意: 1. Debug On RAM 方式下,调试成功的代码,需进行烧录,才会固化到 FLASH 中。
2. SPC2168 中 CAU 的调试接口为 GPIO50、GPIO51。



#### 2.3.2.1 Debug On Flash 的方式进行 CAU 调试

- 前提条件

确保当前 CPU 烧录的程序中,具有如下代码动作:

示例代码

SYSTEM\_EnableDebugForCAU() 或 SYSTEM\_EnableDebugForDualCore() GPIO\_SetPinChannel(GPIO\_50,GPIO50\_CAU\_SWCK) GPIO\_SetPinChannel(GPIO\_51,GPIO51\_CAU\_SWD) SYSTEM EnableCAU()

- 程序烧录

如本文 2.2.2 章节所述,通过 CPU 的调试接口,将程序烧录到 FLASH 中。

- 程序调试

如《快速上手指南》中的 3.3.1 章节所述,通过 CAU 的调试接口,进行 FLASH 中程序的调试工作。

#### 2.3.2.2 Debug On RAM 的方式进行 CAU 调试

- 前提条件

确保当前 CPU 烧录的程序中,具有如下代码动作:

示例代码

```
SYSTEM_EnableDebugForCAU() 或 SYSTEM_EnableDebugForDualCore()
GPIO_SetPinChannel(GPIO_50,GPIO50_CAU_SWCK)
GPIO_SetPinChannel(GPIO_51,GPIO51_CAU_SWD)
SYSTEM EnableCAU()
```

- 配置修改

在本文 2.2.2 章节所述基础上,进行如下修改。

第一步, 配置 ini 文件, 选择"Debug"配置项, 在红色框部分输入 ini 文件的路径, 如下图 所示:



图 2-8: CAU 工程选择 ini 文件

Device   Target   Output   Listing   User   C/C++   C Use Simulator <u>vith restrictions</u> Settings I Limit Speed to Real-Time	Asm Linker Debug Vtilities
Load Application at Startup     Ivit Run to main() Initialization File:     Ivit Run to main() Initialization File:     Ivit Run to main() Iv	Image: Construction at Statup     Image: Construction at Statup       Initialization File:     Image: Construction at Statup       INRAM ini     Image: Construction at Statup
Restore Debug Session Settings C Breakpoints C Toolbox C Watch Windows & Performance Analyzer C Memory Display C System Viewer	Restore Debug Session Settings C Breakpoints C Toolbox Watch Windows Memory Display C System Viewer
CPU DLL: Parameter:	Unver DLL: Parameter:
Dialog DLL: Parameter: [DCM DLL  _pCM4	Dialog DLL: Parameter: TCM.DLL [=cCM4
Manage Component \	Viewer Description Files

第二步,点击"Edit..."按钮,在打开的文件中添加如下所示内容:

图 2-9: ini 文件内容

```
FUNC void Setup (void)
{
   SP = _RDWORD(0x0000);
   PC = _RDWORD(0x0004); // Set Program Counter
   _WDWORD(0xE000ED08, 0x0000);// Set Vector Table Offset Register
   loAD %L INCREMENTAL // Download to RAM
   Setup(); // Setup for debug running
   g, main
```

第三步,取消 Flash 配置,选择"Flash Download"选项,在"Download Function"功能栏中选择"Do not Erase"选项,并取消"Program"、"Verify"、"Reset and Run"选项,如下所示:

#### 图 2-10: Debug in RAM 下"Flash Download"配置

Download	Function		
	C Erase Full Chip	□ Program	
	<ul> <li>Do not Erase</li> </ul>	Reset and Run	

- 程序调试

上述前提条件及配置修改正确操作后,点击"Debug"按钮,可以进行 Debug On RAM 方式下的 CAU 程序调试。



### 2.4 JLINK 调试接口说明

SPC2168 用于调试的 PIN 只有 4 个, 分别是: GPIO48 (TCK/SWCK)、GPIO49 (TMS/SWD)、 GPIO50 (TDI/CAU\_SWCK)、GPIO51 (TDO/CAU\_SWD), 通过 SYSTEM->CAUCTL 寄存器设置 JTAG 的功能。

CAUCTL 寄存器数值	DEBUG 接口功能
0x0	JTAG/SWD 只能由 CPU 使用
0xACCE51	JTAG/SWD 只能由 CAU 使用
0xACCE52	作为两个 SWD 功能,其中 GPIO48 和 GPIO49 是 CPU 的 SWD 接口,GPIO50 和 GPIO51 是 CAU 的 SWD 接口

表 2-1: JTAG 设置