

使用指南

IAR 使用指南

概述

IAR IDE 是一款较为通用的嵌入式开发 IDE,本文对其使用进行了较为全面的描述。

本手册适用范围:

适用范围。				
1125 系列	SPC1125, SPC1128			
1168 系列	SPC1155, SPC1156, SPC1158, SPC1168,			
	SPD1148, SPD1178, SPD1188, SPD1163,			
	SPM1173			
2168 系列	SPC2168, SPC2165, SPC2166, SPC1198			
1169 系列	SPC1169, SPD1179, SPD1176			
2188 系列	SPC2188, SPC1185			



目录

1	新建 IAR 工程	6
1.1	准备工作	6
1.2	创建新工程	6
1.3	添加源文件	7
2	加载 IAR 现有工程	8
2.1	配置工程	8
3	J-LINK 调试	13
3.1	J-LINK 与目标板硬件连接	13
3.2	单步调试	15
3.3	观察外设寄存器	16
3.4	Memory 窗口	17
4	J-LINK 下载	18
5	IAR 界面介绍	19
5.1	主窗口界面	19
5.2	工具栏	19
6	IAR ICF 文件指令介绍	22
6.1	定义 symbol 指令	22
6.2	定义 memory 指令	22
6.3	定义 region 指令	22
6.4	block 指令	23
6.5	定义 initialize 指令	23
6.6	定义 Do not initialize 指令	24
6.7	定义 place at 指令	24
6.8	定义 place in 指令	24
6.9	概述	25
7	IAR ICF 文件使用示例	26
7.1	对单个函数进行重定向	26
	7.1.1 使用ramfunc 关键字进行重定向	26
	7.1.2 使用 section 修饰进行重定向	26
7.2	对多个函数进行重定向	28
7.3	对整个文件进行重定向	29



图片列表

图 1-1:	创建新工程	6
图 1-2:	工程中添加组合源文件	7
图 2-1:	使用 IAR 打开已有工程	8
图 2-2:	配置工程选择芯片内核	8
图 2-3:	库配置	9
图 2-4:	预处理 Preprocessor 添加路径	9
图 2-5 :	预处理 Preprocessor - 预定义	10
图 2-6 :	输出 Hex 文件和链接配置文件	10
图 2-7:	配置下载调试工具	11
图 2-8 :	设置 Debugger Download	12
图 3-1:	J-LINK 接口	13
图 3-2:	未指定设备	14
图 3-3:	调试设备选择	14
图 3-4:	启动 Debug 后的界面	15
图 3-5 :	查看芯片外设寄存器	16
图 3-6 :	Memory 观察窗口	17
图 4-1:	Flash 擦除	18
图 5-1 :	主窗口界面	19
图 5-2 :	工具栏	20
图 5-3 :	主工具栏	20
图 5-4 :	调试工具栏	21
图 7-1 :	ICF 文件与重定向内容	26
图 7-2:	ICF 文件与重定向内容	27
图 7-3 :	ICF 文件与重定向内容	28
图 7-4:	ICF 文件与重定向内容	29
图 7-5 :	IAR 错误弹框窗口	31
图 7-6 :	IAR 错误信息	31
图 7-7:	IAR ICF 文件警告信息	32
图 7-8:	IAR 示例	32



表格列表

表 2-1:	flash 仿真复位类型选择	12
表 3-1:	SW 接口信号定义	13
表 3-2:	芯片与 SWD 管脚	13



版本历史

版本	日期	作者	状态	变更
A/0	2023-09-01	X.He	Outdated	1. 首次发布。
C/0	2024-08-21	X.He	Released	1. 修改为全系列通用文档。



1 新建 IAR 工程

1.1 准备工作

在开始使用 IAR 软件新建工程前,首先需要安装 IAR EW for Arm 8.32.4,本文是基于此版本对 IAR 软件的使用进行介绍。IAR 软件可前往 IAR 官网(https://www.iar.com/)进行下载。

1.2 创建新工程

使用 IAR 软件创建新工程(Project —> Create New Project —> Empty project),具体操作如 图 1-1 所示。



图 1-1: 创建新工程

至此,一个空的基础工程就创建完成,接下来需要进一步添加文件到工程和配置工程。



1.3 添加源文件

完成空的基础工程创建后,向该工程中添加组(文件夹)和添加源文件。也就是就将源代码(驱动库、新建的源文件等)添加到此工程中。此处的工程项目管理可根据用户自行定义(类似于自己分类、命名文件夹和文件),本文将按照常规的方式进行管理项目。

IAR 和 Keil 组管理的区别:

- IAR 可以添加多级组,类似于文件夹下可以再建文件夹,一直下去。
- Keil 只能添加单级组,类似于文件夹下面只能添加文件,而不能在添加文件夹。

为了简单、遵循 Keil 组结构,我们在 IAR 中分组方式也按照 Keil 方式分组,如图 1-2 所示进行操作,先在工程中添加组,再在组中添加文件,依次循环下去直到完成所有源文件的添加。



图 1-2: 工程中添加组合源文件



2 加载 IAR 现有工程

首先找到如图 2-1 所示 Template 工程,双击打开此工程。





2.1 配置工程

1. 首先根据如图 2-2 所示完成芯片内核选择。

Werkspec Debug Files Debug	・ * * * * * * * * * * * * * * * * * * *	Options for node*	"brow" X
	(a)		(b)

图 2-2: 配置工程选择芯片内核

- 2. 根据如图 2-3 所示进行库配置 Library Configuration
- Library: 如果需要使用某些标准的库函数接口(如 printf and scanf),就需要选择 Full。
- CMSIS: 微控制器软件接口标准(Cortex Microcontroller Software Interface Standard), IAR for ARM 使用新版本 IAR 需要将其勾选。





图 2-3: 库配置

Uptions for node "Demo" X
Catopoy Catopoy Catopoy Certe Context Decorptions for Acade Server Control Badd Data Actors Control Badd Control Control Control Badd Control Control Control Badd Control Control Con

3. 预处理 Preprocessor - 添加路径

添加的路径最好是相对路径,而不是绝对路径。使用绝对路径工程位置改变之后就找不到 文件,就会出错。可以点击按钮选择路径,也可以通过复制文件路径进行配置,如图 2-4 所示 进行添加路径。



图 2-4: 预处理 Preprocessor 添加路径



4. 预处理 Preprocessor - 预定义

这里的预定义类似于在源代码中的#define xxx 这种宏定义,如图 2-5 所示可进行设置。

Options fo Category: State Anal Runtime O Output C Custon B Build Acti Linker Debugge Simulat CADI CMSIS CADI CMSIS CADI T Stelle Nu-Link Pemore ST-LINK T MAP- T TMP- T TMP- T TMP- T TMP-	node "Project" X Addition Set
	Cancel

图 2-5: 预处理 Preprocessor - 预定义

5. 输出 Hex 文件和链接配置文件

在编译完程序之后可以通过输出 Hex 文件进行程序烧录,可按照如图 2-6 所示完成配置即可输出 Hex 文件。也可通过链接配置文件查看链接程序时所产生的信息(定义内存位置、内存 大小和堆栈大小等重要信息)。

Options for node "Project" Cdegogy Generad Obford Suthing Charling Cit + 4 Conglet Assember Cutato Build Build Action Lindbogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Obfogor Offogor Offogor Offogor Sindalor Offogor Sindalor Offogor Sindalor Offogor Sindalor Offogor Sindalor Sindalor Sindalor Sindalor Offogor Sindalor Offogor Sindalor Options file Options file Op	Factory Settings	Options for node "Project"	Factory Settrage Encodings Extra Options s Advanced Output Est
I	OK Cancel	C	OK Cancel
(a)		(b)	

图 2-6: 输出 Hex 文件和链接配置文件

6. 选择下载调试工具

根据实际情况选择下载调试工具,本文选择 J-Link 作为下载调试工具,配置如图 2-7 所示。



图 2-7: 配置下载调试工具



在 flash 仿真时复位类型需要按照如表 2-1 所示进行选择,图 2-7 的(b)图选择复位 core 类型。在 RAM 仿真时 Core、Normal、Core and peripherals 复位所有平台都支持。



表 2-1: flash 仿真复位类型选择

芯片	复位类型选择
SPC1168 系列,SPC2168 系列,SPC1169 系列, SPC1125 系列	Core、Normal、Core and peripherals
SPC2188 系列	Core

在使用 J-Link 进行下载调试程序之前,还需要设置 Debugger Download 选项,如图 2-8 所示进行添加 board 文件。

注意: 在开始 Debug 之前,需要将 IDE_Support\EWARM\flashloader 目录下的算法文件复制到 IAR 软件安装目录下的 arm\config\flashloader\SPINTROL 目录下,否则会报错。

图 2-8: 设置 Debugger Download

Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debuoger	Setup <mark>Download</mark> Images Extra Options Multicore Plugins ⊠Yerify download □Suppress download	Factory Settings	SPINTROL 共享 查看				
Smulator CADI CMSIS DAP CDB Server 1_set/JTAGet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third-Parky Driver TI MSP-FET TI MSP-FET TI MSD	Configuration House Configuration Configuration and Figuration Configuration and Figuration Configuration Configuration Configuration Configuration Configuration Configuration	 ← → < 1 ★ 快速访问 ■ 点面 ◆ 下数 □ 文档 ■ 図片 	« IAR Systems > Embedded Workbench 8.2 > Storm - FlashSPC2188.board FlashSPC2188.lash FlashSPC2188.out	arm > config > flashloade > 修改日期 2023/4/17 18:47 2023/4/17 18:47 2023/4/17 18:47	SPINTROL 类型 BOARD 文件 FLASH 文件 OUT 文件	マクレンジャンクロン 1 KB 1 KB 28 KB	P E SPIN
	OK Cancel			(b)			

[1] 图中所示为 SPC2188 的算法文件。



3 J-LINK 调试

3.1 J-LINK 与目标板硬件连接

J-LINK 适配器支持 2 种接口,如图 3-1 所示。推荐使用 SWD 接口,因为更省引脚而且调试 功能不受影响,该接口如表 3-1 所示。

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
	JTAG			SV	VD.	
						~

图 3-1: J-LINK 接口

表 3-1: SW 接口信号定义

Signal	Connects to
SWDIO	Data I/O pin
SWCLK	Clock pin
VCC	Positive Supply Voltage, the pin is optional.
GND	Digital ground
RESET	RSTIN pin, the pin is optional.
SWO	Serial data output, the pin is optional.

在使用芯片进行应用开发的过程的中,需要经常使用 J-LINK 进行程序的调试,各个芯片对 应的 SWD 管脚如表 3-2 所示。

表 3-2: 芯片与 SWD 管脚

芯片型号	SWD 引脚			
	SWDIO	SWCLK		
SPC1169 系列	GPIO17	GPIO18		
SPC1168 系列	GPIO38	GPIO39		
SPC2168 系列(仅 CPU 核)	GPIO49	GPIO48		
SPC2168 系列(仅 CAU 核)	GPIO51	GPIO50		



芯片型号	SWD 引脚			
	SWDIO	SWCLK		
SPC2188_CPU0/SPC2188_CPU1(单核 SWD 模 式)或者 SPC2188_CPU0 (双核 SWD 模式)	GPIO80	GPIO81		
SPC2188_CPU1(双核 SWD 模式)	GPIO78	GPIO79		
SPC1185	GPIO80	GPIO81		
SPC1125 系列	GPIO38	GPIO39		

注意: J-LINK 调试时,芯片的 TRSTN 必须拉高, boot 引脚电平请参考 TRM 启动引导配置 章节;

J-LINK 调试时, J-LINK 的 GND 必须连上调试板的 GND;

J-LINK 下载器端口电压需要与芯片端口电压一致。

根据前面的介绍,将 J-LINK 设备与 spintrol 芯片正确连接后,按照图 2-7、图 2-8 设置 Debug 的相关选项,就可以使用 J-LINK 设备调试程序了。

单击工具栏上的●(进行下载和仿真)或者 · (仿真但不下载)进入 Debug 状态,如果是 第一次进行调试会出现如图 3-2 弹框,点击 "OK"进行选择调试设备选择如图 3-3 所示,选择 进行仿真的设备。

图 3-2: 未指定设备

🔜 J-Lir	nk V6.14b Device Selection	
	The selected device "UNSPECIFIED" is unknown to this version of the J-Link software. Please make sure that at least the core J-Link shall connect to, is selected. Proper device selection is required to use the J-Link internal flash loaders for flash download or unlimited flash breakpoints. For some devices which require a special handling, selection of the correct device is important	
	<u> </u>	

图 3-3: 调试设备选择

Filter Manufacturer *	~ Device	Core *	~	Little er	ndian v
Manufacturer	Device	Core	NumCores	Flash size	RAM size 🔨
Unspecified	Cortex-M1	Cortex-M1	1		
Unspecified	Cortex-M3	Cortex-M3	1		
Unspecified	Cortex-M4	Cortex-M4	1		-
Unspecified	Cortex-M7	Cortex-M7	1		
Unspecified	Cortex-M23	Cortex-M23	1		
Unspecified	Cortex-M33	Cortex-M33	1		
Unspecified	Cortex-R4	Cortex-R4	1		
Unspecified	Cortex-R5	Cortex-R5	1		
Unspecified	PIC32MK	MIPS	1		
Unspecified	PIC32MZ	MIPS MicroAptiv	1		
Unspecified	PIC32WK	MIPS M14K	1		
Unspecified	RX	RX	1	•	
Abov	AC33M6128L	Cortex-M3	1	128 KB	12 KB
Abov	AC33M8128	Cortex-M3	1	128 KB	12 KB 🗸

当设备选择正确,调试界面将会进入如图 3-4 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,IAR 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,绿色底纹的汇编代码对应于用户代 码窗口中的 C 代码;此外,菜单栏上也出现了一些与 Debug 相关的菜单选项。

注意: 在程序进入 Debug 状态后,如果想修改代码,可以通过单击按钮 重新下载程序 并进入 Debug 模式。

Project - IAR Embed	lded Workbench ID	E - Arm 8.32.4												- 0
File Edit View Proje	ect Debug Disasse	embly J-Link T	ools Window Help					_						
🏠 🙆 🔛 🔒 🔚	X 🛍 🗂 🗅 O	*	- < Q > \$ HE < 🕻) > 🔹 🕨	📄 🌒 🛥 🕝 C 🔾 📮 📭	그 바위 전 💌	🕕 🏛 📲 📮 ETM SUA	to						
Registers 2	- 1 X	Disassembly												-
clind register>	Print (Ctrl+P)	Go to	 Memory 	~ 🗈										
Ourrent ODU De	Value.	Disasse	mhly											
Current CPO Re	Alife	FLAS	- SetTiming(200000000)											
RU D1	0x10000000	0x1	0000ac4: 0x4d12	LDR .N	R5. [PC. #0×48]	: Oxbebc	200 (200000000)							
RI	0x10000CAC	0v1	1000ac6: 0x0028	MOVS	PO. P5	, 0200000	(200000000)		汇编代	「山密口				
82	0=10000085	0x1	0000ac8: 0xf7ff 0xfe0e	BL.	FLASH SetTiming	: 0×1000	16e8							
RJ D4	0x10000003	FLAS	(WDIS():			,								
R4 DC	0=20002594	0x1	000acc: 0x2000	MOVS	R0. #0									
R5 D6	0x20003304	<												:
87	0=000000000	main.c x fla	sh.c											
PB	0×00000000													
29	0x03E4BEEE	13 •	* · LIABLE · FOR · THE · USE	· OF · THE	SOFTWARE. · SPINTROL · D	OES · NOT · GUAR	ANTEE · THE ·							
R10	0x00001F8C	14 .	* · CORRECTNESS · OF · THI	S.SOFTW	ARE · AND · RESERVES · THE ·	RIGHT · TO · MOL	IFY . THE . SOFTWA	RE-						
R11	0+000000000	15 -	* · WITHOUT · NOTIFICATI	ON.										
R12	0x200035C4	16 •	*											
APSP	0x60000000	17		******		**********	********	****/						
IPSR	0x00000000	18	and Hereitico bu											
FPSR	0×01000000	20	nclude. spciics.n											
PC	0x10000ABC	21	cherune (sectro.ms											
SP	0x200004B0	22												
LR	0x10000CE7	23												
PRIMASK	0x00000000	24							田白海伊和					
BASEPRI	0x00000000	🕈 25 ii	nt·main(void)											
BASEPRI MAX	0x00000000	26日(
FAULTMASK	0x00000000	27	PINCH WALLOW().	/										
CONTROL	0x00000004	29	FLASH SetTiming (200	000000										
CYCLECOUNTER	111377	30	FLASH WDIS();	,,										
CCTIMER1	111377	31 .												
CCTIMER2	111377	32 .	/*.Clock.init.*/											
CCSTEP	111377	33 •	CLOCK_InitWithRCO(C	FOCK HCI	K_200MHZ);									
		34 .	•											
		35 .	/*.Delay.init.*/											
		30 .	Delay Init();											2
		Debug Log					- a x	Stack 1						•
CPII寄左	哭窗口	Log					^	CSTACK						
CIOBITI		Fri Ma	r 25, 2022 10:31:10: Hardwa	are reset wi	th strategy 8 was performed			Locatio	n Data	Variable	Value	Туре	Frame	
		Fri Ma	r 25, 2022 10:31:10: Target i	reset										
		Fri Ma	r 25, 2022 10:31:10: — Ru	nning boot	oader before execute applica	tion —								
		Fri Ma	r 25, 2022 10:31:10: Reset: I	Halt core a	fter reset via DEMCR.VC_CC	DRERESET.								
		Fri Ma	r 25, 2022 10:31:10: Reset: I	Reset devi	ce via AIRCR.SYSRESETRE	Q.								
	>	Fri Ma	r 25, 2022 10:31:10: Hardwa	are reset wi	th strategy 8 was performed		~							
egisters 2 🚯 Workspac	ce	Build Debug	Log											
int the active document												Ln 20	Col 19	UTF-8 大写 数字 改写

图 3-4: 启动 Debug 后的界面

3.2 单步调试

单击工具栏上的**●**或者 ▶ 按钮后,程序进入 Debug 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的绿色箭 头 ◆ 表示当前位置的代码为下一次要执行的语句。因此,可以通过这个绿色箭头快速判断程序 执行到了哪条语句。

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「按钮,程序就会立即执行到当前光标处。此外,我们也可以通过设置断点的方式来实现上述功能。



3.3 观察外设寄存器

在调试程序的时候,当我们需要查看芯片外设 Register 的值时,可通过如图 3-5 所示方法 进行查看。



图 3-5: 查看芯片外设寄存器



3.4 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。 我们以芯片的 UART 模块为例,通过芯片技术参考手册可以得到 UARTDLH 寄存器和 UARTIER 寄存器的地址为 0x40004004。首先,打开一个 Memory 观察窗口 (Memory1),如图 3-6 所示。



图 3-6: Memory 观察窗口



4 J-LINK 下载

使用 J-LINK 进行下载 Flash 之前需要先使用 J-LINK 对 Flash 进行擦除,如图 4-1 所示。

File Edit View Pr	piect J-Link Tools Window Help	
	Add Files	
	Add Group	
Norspace	Import File List	
Debug	Add Project Connection	
Files	Edit Configurations	main.c
E Project - D		Main program body
- E Perinh D	Remove	Ion VXX.XX.XX
	Create New Project	
- ⊕ isr.c	Add Existing Project	
under under Littlitice	0-tions Att. 57	Ight (C) 2022 Spintrol Electronic Tec
	Options Alt+F/	ntion
	Version Control System	OFTWARE JUST PROVIDES CUSTOMERS WITH
	Make F7	FOR THE USE OF THE SOFTWARE, SPINTF
	Compile Ctrl+F7	TNESS OF THIS SOFTWARE AND RESERVES
	Bebuild All	IT NOTIFICATION.
	Clean	
	Batch build E8	
		"spc2188.h"
	C-STAT Static Analysis	<stdio.h></stdio.h>
8	Stop Build Ctrl+Break	
C	Download and Debug Ctrl+D	void)
•	Debug without Downloading	
	Attach to Running Target	Init();
G	Make & Restart Debugger Ctrl+R	[- ···
c	Restart Debugger Ctrl+Shift+R	etChannel (PIN_GPI062, PIN_GPI062_UAF
	Do <u>w</u> nload	Download active application
L	STD Colum	Download file
	SFK SELUP	Erase memory
	CMSIS-Pack Manager	(1)
	Open Device Description File	
	Save List of Registers	
		a.

图 4-1: Flash 擦除

在擦除 Flash 之后可以对 Flash 进行写操作,下载 HEX 文件有两种方式,一种是选择如图 4-1 所示 "Download active application"下载当前工程的 HEX 文件,另一种是选择 "Download file..."指定某个 HEX 文件进行下载。

注意: 在开始下载之前,需要将 IDE_Support\EWARM\flashloader 目录下的算法文件复制 到 IAR 软件安装目录下的 arm\config\flashloader\SPINTROL 目录下如图 2-8(b)图 所示,否则会报错。



5 IAR 界面介绍

5.1 主窗口界面

这里简单介绍一下 IAR 软件主界面下的各个窗口,如图 5-1 所示。

- Menu Bar 菜单栏: 该窗口是 IAR 比较重要的一个窗口, 里面包含 IAR 所有操作及内容
- Tool Bar 工具栏: 该窗口是一些常见的快捷按钮
- Workspace Window 工作空间窗口:一个工作空间可以包含多个工程,该窗口主要显示工作空间下面工程项目的内容
- Edit Window 编辑空间:代码编辑区域
- Message Window 信息窗口: 该窗口包括编译信息、调试信息、查找信息等信息窗口
- Status Bar 状态栏: 该窗口包含错误警告、光标行列等一些状态信息



图 5-1: 主窗口界面

5.2 工具栏

IAR 软件中的 Tool Bar 工具栏一共有两个: Main 主工具栏和 Debug 调试工具栏。在编辑(默认)状态下只显示 Main 工具栏,在进入调试模式后才会显示 Debug 工具栏。

工具栏可以在通过菜单打开: Window --> Tool Bar, 如图 5-2 所示。



图 5-2: 工具栏



1. 主工具栏

如图 5-3 所示,在编辑(默认)状态下,只有主工具栏,这个工具栏的内容也是在编辑状态下常用的快捷按钮。其中"Download and Debug"和"Debug without Downloading"这两个按钮的区别需要注意。

Download and Debug: 是下载代码之后再进行调试。

Debug without Downloading:只调试不下载,即如果之前下载过了代码,只需要再点击该按钮即可,否则会出现错误。

这两个按钮图标在编辑和调试模式下略有差异,在调试模式下可以再次下载程序后继续调试。





2. 调试工具栏

调试工具栏则是在进行程序调试时才有效的一个快捷按钮,在编辑状态下,这些按钮是无效的。

如图 5-4 所示,以下是调试模式中常用的快捷按钮:

Reset 复位

Break 停止运行





Step Over 逐行运行 F10 Step Into 跳入运行 F11 Step Out 跳出运行 F11 Next Statement 运行到下一语句 Run to Cursor 运行到光标行 Go 全速运行 F5 Stop Debugging 停止调试 Ctrl + Shift + D

图 5-4: 调试工具栏





6 IAR ICF 文件指令介绍

6.1 定义 symbol 指令

用法:

define [exported] symbol name = expr;

参数:

exported:导出该 symbol,使其对可执行镜像可用

name: 符号名

expr: 符号值

示例:

define symbol __ICFEDIT_region_ROM1_start__ = 0x10000000;

作用:

指定某个符号的值

6.2 定义 memory 指令

用法:

define memory [name] with size = size_expr;

参数:

Name: memory 的名称

size_expr: 地址空间的大小

示例:

define memory mem with size = 4G;

作用:

定义一个可编址的存储地址空间

6.3 定义 region 指令

用法:

define region name = region-expr;

参数:

Name: region 的名称

region-expr: memory_name:[from expr to expr],可以定义起止范围,也可以定义起始地址和 region 的大小。

示例:



define region ROM1_region = mem:[from __ICFEDIT_region_ROM1_start__ to __ICFEDIT_ region_ROM1_end__];

define region RAM2_region = mem:[from __ICFEDIT_region_RAM2_start__ to __ICFEDIT_ region_RAM2_end__];

作用:

定义一个存储地址区域(region)。一个区域可由一个或多个范围组成,每个范围内地址 必须连续,但几个范围之间不必是连续的

6.4 block 指令

用法:

define block name[with param, param...]

参数:

Name: block 的名称

Param:

```
size =expr(块的大小)
```

```
maximum size = expr(块大小的上限)
```

alignment = expr(最小对齐字节数)

```
fixed order (按照固定顺序放置 sections)
```

示例:

```
define block CSTACK with alignment = 8, size = __ICFEDIT_size_cstack___ { };
```

作用:

定义一个地址块(block);它可以是个空块,比如栈、堆;

6.5 定义 initialize 指令

用法:

```
initialize { by copy | manually } { section-selectors }
```

参数:

```
by copy: 在程序启动时 IAR 软件进行自动拷贝。
```

Manually: 在程序启动时 IAR 软件不进行自动拷贝。

示例:

initialize by copy { readwrite, };

作用:

初始化 section 将指定 section 从 ROM 拷贝到 RAM



6.6 定义 Do not initialize 指令

用法:

do not initialize { section-selectors }

参数:

section-selectors: section 选择器 [section-attribute][section sectionname][object filename] 示例:

do not initialize { section .noinit };

作用:

规定在程序启动时不需要初始化的 sections。一般用于__no_init 声明的变量段(.noinit)

6.7 定义 place at 指令

用法:

place at { address [memory:] expr | start of region_expr | end of region_expr }

{

section-selectors

}

参数:

address [memory:] expr: 特定内存中的特定地址。memory 地址必须在由 define memory 指令定义的提供的内存中可用。

start of region_expr: region 的起始地址。

end of region_expr: region 的结束地址。

section-selectors: section 选择器 [section-attribute][section sectionname][object filename] 示例:

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

作用:

把一系列 sections 和 blocks 放置在某个具体的地址,或者一个 region 的开始或者结束处

6.8 定义 place in 指令

用法:

place in region-expr { section -selectors }

参数:

section-selectors: section 选择器 [section-attribute][section sectionname][object filename] 示例:



place in ROM1_region { readonly };

作用:

把一系列 sections 和 blocks 放置在某个 region 中。sections 和 blocks 将按任意顺序放置。

6.9 概述

Section	描述
.bss	保存初始化为0的静态和全局变量
CSTACK	保存 C 或 C++ 程序使用的堆栈
.cstart	保存 start_up 代码
.data	保存静态和全局初始化变量包括初始值设定项
.data_init	保存.data section 的初始值设定项。
.difunct	保存指向代码(通常 C++构造函数)的指针,这些代码应在调用 main 之前由系统启动代码执行
HEAP	保存用于动态分配数据的堆
.iar.dynexit	保存 atexit table
.intvec	保存复位和中断向量
IRQ_STACK	保存中断请求、IRQ 和异常的栈
.noinit	保存no_init 静态和全局变量
.rodata	保存常量数据
.text	保存程序代码

如需查询指定的目标文件包含的 section 内容,可以使用 IAR 编译工具下面的 ielfdumparm.exe 可执行文件查看。 ielfdumparm.exe 文件路径在安装 IAR 软件 IAR Systems\Embedded Workbench 8.2\arm\bin 目录下。



7 IAR ICF 文件使用示例

7.1 对单个函数进行重定向

7.1.1 使用__ramfunc 关键字进行重定向

使用__ramfunc 关键字会将关键字修饰的函数重定向到 ICF 文件中包含 readwrite 属性的 Region 包含的地址范围内,并且关键字修饰的函数不能调用没有被该关键字修饰的函数以及带 有 const 修饰的全局变量。

对函数代码而言,将会被重定向到 ICF 文件中包含有 readwrite 的 region;

对于全局变量而言,非 const 全局变量将会被重定向到 ICF 文件中包含有 readwrite 的 region;

对于局部变量而言,非 static 局部变量将会被重定向到 ICF 文件中包含有 block CSTACK 的 region; static 局部变量将会被重定向到 ICF 文件中包含有 readwrite 的 region;

如图 7-1 所示, add_function_test3 函数代码被重定向到 0x1FFF8000~0x1FFFFFF RAM2_region 地址,是因为 ICF 文件中 RAM2_region 是包含有 readwrite 属性段的 region。形参 a 以及 b 将会被重定向到 RAM_region 中,因为这个 region 包含有 block CSTACK。非 const 且非 static 的全局变量 add_test2 将会被重定向到 RAM2_region 中,因为这个 region 包含有 readwrite。

do not initialize {	<pre>section .noinit };</pre>							
place at address mem:	ICFEDIT_intvec_st	art { readonly s	ection .intv	ec };				
<pre>place in ROM_region };</pre>	{ readonly,							
place in RAM_region	{							
};	block CSTACK, block HEAP,							
define region RAM2_re place in RAM2_region	gion = mem:[from 0x { readwrite.	1FFF8000 to 0x1FFF	FFFF];					
};	(countree)							
		R1 R2 R3	0x0000 0008 0x1fff 80d8 0x0000 0001	ReadWrite ReadWrite ReadWrite	0x1fff'8064 0x1fff'8068	iar_ttio_n Dxffff'ffff Dx0000'0001 add_function	DC32 DC32 DC32 _test3:	_1 1
0)		R4 R5 R6 R7	0x1000'1140 0x0beb'c200 0x0000'0000 0x0000'0000	ReadWrite ReadWrite ReadWrite ReadWrite	0x1fff'806c; 0x1fff'806c; 0x1fff'8070; 0x1fff'8072;	0x4a03 0x6010 0x6051 0x6810	LDR_N STR STR LDR	R2, 22add funct R0, [R2] R1, [R2, #0x4] R0, [R2]
		R8 R9	0x0000'0000 0x1000'10ed	ReadWrite ReadWrite	0x1fff'8074 0x1fff'8076	0x6851 0x1808	LDR ADDS	R1, [R2, #0x4] R0, R1, R0
	<pre>do not initialize { place at address mem: place in ROM_region }; place in RAM_region }; define region RAM2_region }; b)</pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_st place in ROM_region { readonly, }; place in RAM_region { block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x place in RAM2_region { readwrite,] }; b)</pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start_ { readonly se place in ROM_region { readonly, }; place in RAM_region { block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFF1 place in RAM2_region { readwrite,] }; </pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start { readonly section .intv place in ROM_region { readonly, }; place in RAM_region { block CSTACK, block HEAP, }; define region RAM12_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region { readwrite, }; b) </pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvcc_start_ { readonly section .intvec }; place in ROM_region { readonly, ; place in RAM_region { block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region { readwrite, }; b) </pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start { readonly section .intvec }; place in ROM_region { readonly, ; place in RAM_region { block CSTACK, block HEAP, }; define region RAM12_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region { readwrite,] ; } b) define region RAM12_region { readwrite,] readwrite,] readwrite,] } </pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start { readonly section .intvec }; place in ROM_region { readonly, ; place in RAM_region { block CSTACK, block HEAP, }; define region RAM12_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region { readwrite,] ; } b) b) define region RAM12_region { readwrite,] rea</pre>	<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start { readonly section .intvec }; place in ROM_region { readonly, ; place in RAM_region { block CSTACK, block HEAP, }; define region RAM12_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region { readwrite, readwrite, }; b) b) b) control to block do bloc</pre>

图 7-1: ICF 文件与重定向内容

注意: 该方法的重定向适用所有函数包含中断服务函数及中断中调用的函数。

7.1.2 使用 section 修饰进行重定向

对某个函数进行重定向到 ICF 文件中 place in 指令指定的"section .add_test2_func_section" 的 Region 包含的地址范围内,在重定向的函数体中可以调用其他地址范围的函数。

对函数代码而言,将被重定向到 ICF 文件中包含有"add_test2_func_sectio"标识符的 region。

对于全局变量而言,非 const 全局变量将会被重定向到 ICF 文件中包含有 readwrite 的 region; const 全局变量将会被重定向到 ICF 文件中包含有 readonly 的 region;



对于局部变量而言,非 static 局部变量将会被重定向到 ICF 文件中包含有 block CSTACK 的 region; static 局部变量将会被重定向到 ICF 文件中包含有 readwrite 的 region;

如图 7-2 所示, add_function_test2 函数被重定向到 0x1FFF8000~0x1FFFFFF RAM2_region 地址,是因为 ICF 文件 RAM2_region 是包含有"add_test2_func_sectio"的 region。形参 a 以及 b 将会被重定向到 RAM_region 中,因为这个 region 包含有 block CSTACK。非 const 且非 static 的 全局变量 add_test2 将会被重定向到 RAM2_region 中,因为这个 region 包含有 readwrite。

图 7-2: ICF 文件与重定向内容

	<pre>define symbolLCFEDI1_InTVec_Start = 0x10000000; /*-Memory Regions-*/ define symbolICFEDIT_region_ROM_end = 0x1000FFFF; define symbolICFEDIT_region_RAM_start = 0x20000000; define symbolICFEDIT_region_RAM_end = 0x20003FFF; /*-Sizes-*/ define symbolICFEDIT_size_cstack = 0x400; define symbolICFEDIT_size_neap = 0x200; /**** End of ICF editor section. ###ICF###*/ define memory mem with size = 4G; define region ROM_region = mem:[fromICFEDIT_region_ROM_start toICFEDIT_region_ROM define region RAM_region = mem:[fromICFEDIT_region_RAM_start toICFEDIT_region_RAM define block CSTACK with alignment = 8, size =ICFEDIT_size_cstack { }; define block KEAP with alignment = 8, size =ICFEDIT_size_neap { }; initialize by copy { readwrite.</pre>
	section .add_test2_func_section,
	<pre>do not initialize { section .noinit };</pre>
	<pre>place at address mem:ICFEDIT_intvec_start { readonly section .intvec };</pre>
	place in ROM_region {
	readonly,
	· ,
	<pre>place in RAM_region { readwrite, block CSTACK,</pre>
	block HEAP,
	define region KAMZ_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region {
	section .add_test2_tunc_section,
	D
spc2168.idf sub_test.c add_test.c × main.c	Registers 1 · S X: Disasembly
add_function_test2[int, int]	fti) Prefe 🗸 Groups Current CPU Registers Go Se 🖉 🗸 Zones (Manory 🗸 💽
<pre>add_test1.data1 = a + add_const; add test1.data2 = a + add const1:</pre>	Name Value Cocces Re Coc0000 10066 Read/With
add_test1.data1 = a + add_static;	R1 Operative Operative Operative Operative Control LDR FC FC LDR FC FC FC FC FC FC FC FC
<pre>add_test1.data2 = a + add_static1; printf("string = %s\n", < string);</pre>	R3 Opu000010001 Read/Whit add_function_test: R4 Opu00010004 Read/Whit 0x1fff10001 Read/Whit
<pre>printf_funcl();</pre>	H5 Ox00ebb*1c200 ReadWhit Outliff*1000. Ox404 LDR.N R4. 72edd_fmuchican. H6 0x00000 ReadWhit Outliff*1000. 0x400* 0x5TR R0. [R4]
return add_test1.data1 + add_test1.data }	2 + ad_test3.deta1; R7 0x0000'0000 ReadVMH <u>0xtfff18Be 0xt6st 37</u> R1 FR / FVA1 R8 0x000'0000 ReadVMH 0xtff10101 0xff10 fttf16 EL ?Penner1 1(6) ftor .
<pre>int add_function_test2(int a, int b) @".a </pre>	B9 0x100*104 ReadWht 0x11ff*1014: 0x6820 LDR R0. [R4] dd_test2_func_section* R10 0x000*0144 ReadWht 0x11ff*1014: 0x6820 LDR R0. [R4]
add test2.data1 - a:	R11 0x0000*0000 Read/Write 0x11ff*0818 0x1000 RD
<pre>printf_func2();</pre>	(# AFSR 0x0000*1000 Read/Writ 77xdd_function_test2.0: a IFSR 0x0000*1000 Read/Writ 0x1fff*001:: 0x2000*0000 b020
return add_test2.data1 + add_test2.data	22 PC 0x110*1000 PeadWink 0x111*1000 C22 0 PC 0x11*1*1000 PeadWink 0x11*1*1000 C22 0
<pre>int add function test3(ie* = in* b)</pre>	SP 0x2000'0464 0x40ff'1023 0x000'0000 CC2 0 IR 0x100'06ef ReadVMM 0x1ff'1022 0
add_test2.data1 = a;	IF PETMASIK 0x0000*0000 0x01ff*10305 0x0000*0000 CC22 0 IM BASEPERT 0x0000*0000 ReadVMM 0x1ff*10305 0x0000*0000 CC22 0
add_test2.data2 = b;	NBASEPERT_KAX 0x0000'0000 ReadVMWR 0x1ff'0001'0000 CC22 0 # PARLTMASK 0x0000'0000 ReadVMWR 0x1ff'0001'0000 CC22 0
return add_test2.data1 + add_test2.data }	2; 0001FR01. 0x0000*0004 GendWink 0x1ff*0404: 0x0000*0000 CC22 0 CYCLECCOUNTER 129*672 Read/ork 0x1ff*0404: 0x0000*0000 CC22 0

注意: 该方法的重定向不适用中断服务函数及中断服务函数中调用的函数。



7.2 对多个函数进行重定向

对多个函数进行重定向到 ICF 文件中 place in 指令指定的"section .add_test1_func_section" 的 Region 包含的地址范围内,在重定向的函数体中可以调用其他地址范围的函数,重定向多个函数时,重定向起始使用"#pragma default_function_attributes = @ "section_name"",重定向 结 束 使 用 "#pragma default_function_attributes =" 。 如 果 需 要 重 定 向 变 量 可 以 用"default_variable_attributes"。

对函数代码而言,将被重定向到 ICF 文件中包含有"section_name"的 region。

对于全局变量而言,非 const 全局变量将会被重定向到 ICF 文件中包含有 readwrite 的 region; const 全局变量将会被重定向到 ICF 文件中包含有 readonly 的 region;

对于局部变量而言,非 static 局部变量将会被重定向到 ICF 文件中包含有 block CSTACK 的 region; static 局部变量将会被重定向到 ICF 文件中包含有 readwrite 的 region;

如图 7-3 所示, add_function_test2 和 add_function_test3 函数被重定向到 0x1FFF8000~0x1FFFFFF RAM2_region 地址, 是因为 ICF 文件 RAM2_region 是包含 有"add_test1_func_section"的 region。形参 a 以及 b 将会被重定向到 RAM_region 中, 因为这个 region 包含有 block CSTACK。非 const 且非 static 的全局变量 add_test2 将会被重定向到 RAM2_region 中, 因为这个 region 包含有 readwrite。

<pre>fife grads</pre>	<pre>price upon l_f(TENDT_inter_time</pre>							
<pre>retrom ymaniC(T); // **: retrom ymaniC(</pre>	<pre>define speciCCVVinter_text = bulkerese // fine speciCCVVinter_text = bulkerese define speciCCVVI_region_MV:erd = iblerYFFF; define speciCCVVI_region_MV:erd = bulkerYFFF; define bulkerKappi = semi_ffrem_ICFDUT_region_MV:tart to _CFDUT_region_MV:erd = bulkerYFF define bulkerKappi = semi_ffrem_ICFDUT_region_MV:erd = bulkerYFFF; define bulkerKappi = semi_ffrem_ICFDUT_region_MV:erd = bulkerYFFF; define bulkerKappi = semi_ffrem_ICFDUT_region_MV:erd = bulkerYFFF; define specie_CCVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV</pre>							
<pre>//*dec:spid:/</pre>	<pre>/* Amony Region =: #</pre>		define symbolICFEDIT_intvec_start	= 0x1000000;				
<pre>define spublCFUBL_region_Md_tstr = 0.40000000000000000000000000000000000</pre>	<pre>drine yeadsCCOUT_region Add_tata = 0.00000000000000000000000000000000</pre>		/*-Memory Regions-*/					
<pre>strine symbolCFUIL_region_MOV_red = elements; /*-Sizes-/ / drive symbolCFUIL_region_MOV_red = elements; /*-Sizes-/ drive symbolCFUIL_region_MOV_red = elements; /*-Sizes-/ drive symbolCFUIL_region_MOV_red = elements; // drive region MOV_region = ese:[fromLCFUIT_region_MOV_rtart toCCFUIT_region_MOV_end_]; drive symbolCFUIL_region_MOV_red = is :: = _CCFUIT_region_MOV_red]; drive statistication = elements; initialize by copy { rederite; section .eduit); place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; drive statistication = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_NOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_NOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_NOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_MOV_red]; place in MOV_region = ese:[from @LCFUIT_region_NOV_red]; place in MOV_region_region_region_region_region_region_region_region_region_region_region_region_region_region_re</pre>	<pre>define spublCFRUIT_stige.mov_end = delegative /*-Sizes-/ /*-Sizes-/ /*-Sizes-/ /*-Sizes-/ / define semony mew with size - 46; define semony mew with size - 40; define semony mew with size - 46; define semony mew with size - 46; sectiondot center/reasony mew seminy mew seminy memory mew seminy memory mew seminy memory mew seminy memory m</pre>		define symbolICFEDIT_region_ROM_star	rt = 0x10000000;				
<pre>define synchLCFDIT_regin_MALtert = 0.40000000; define synchLCFDIT_ist=cttek = 0.4000; define synchLCFDIT_ist=cttek = 0.4000; define searcy me with size - 40; define regin MALreginemailter define regin MALregi</pre>	<pre>define synbolCFEDT_region_MULtart = 0.000000000000000000000000000000000</pre>		define symbolICFEDIT_region_ROM_end	= 0×1007FFFF;				
<pre>strate symbolCFUET_region_MAR_red 0.2000JFF; define symbolCFUET_region_MAR_red 0.2000JFF; define symbolCFUET_region_MAR_red 0.2000JFF; define symbolCFUET_region_MAR_red 0.2000JFF; define symbolCFUET_region_MAR_red 0.2000JFF; define region MAR_region = exe:[fromCFUET_region_MAR_start toCFUEDT_region_MAR_red]; define from back thus = exe:[fromCFUET_region_MAR_red]; define back thus = exe:[from CFUET_region_MAR_red]; define back thus = exe:[from chullence.text[],</pre>	<pre>error symbolCFEDUT_region_AM(_red 0.0000000000000000000000000000000000</pre>		define symbolICFEDIT_region_RAM_star	rt = 0x20000000;				
<pre>//~~fire year/ define year/ define year/ define year/ define year/ define year/ define year/ define year/ define year/ year/ define year/ define year/ year/ define year/ year/ define year/ year/ define year/ define year/ year/ define year/ define year/ year/ define year/ ye</pre>	<pre>/*-size=?/ define symbolCFEDIT_size_cstack_ = 0x400; define symbolCFEDIT_size_cstack_ = 0x400; define secory ese with size = 40; define region MG(region = sec:[fromLCFEDIT_region_NGU_start_ toCFEDIT_region_NGU_end_]; define lock CSTACK with alignment = 8, size = _CFEDIT_size_tstack_ = { } }; define block CSTACK with alignment = 8, size = _CFEDIT_size_tstack_ = { } }; define block CSTACK with alignment = 8, size = _CFEDIT_size_tstack_ = { } }; do not initialize { section .noint }; place in AGM_region = esc:[from duff#F8000 to duff#FFFFFF; }; do not initialize { section .noint }; place in SMM_region = esc:[from duff#F8000 to duff#FFFFFF; }; define region MMU_region = esc:[from duff#F8000 to duff#FFFFFF; }; place in SMM_region = esc:[from section; ; treads finati_toutin_stringstringstring =</pre>		define symbolICFEDIT_region_RAM_end	= 0x20003FFF;				
<pre>/*ister=// define secory and with size = 40; define secory and size = 40; define secory = 10; find size = 1, secony; find size = 1, secon</pre>	<pre>/*_:::::/ /*::::::::::::::::::::::::::::</pre>		(n = 1 = n /					
<pre>define typesd</pre>	<pre>define years if welliter_iter_iter_iter_iter_iter define region R00;region = sem:[fromiCFEDIT_region_R00;tert toCFEDIT_region_R04_end_]; define region R00;region = sem:[fromICFEDIT_region_R04_end_]; define block STACK with alignment = 0, size =ICFEDIT_region_R04_end_]; define block STACK with alignment = 0, size =ICFEDIT_size_texp {};; sectionR04_end(); j = sectionR04_end(); j = sectionR04_end(); j = sectionR04_end(); j = sectionR04_end(); j = sectionR04_end(); sectionR04_end(</pre>		/*-51zes-*/					
<pre>define region RAM1_region *= #0.200; /*** To un'of ICT editor section ##ICT###P # 00.200; / define region RAM1_region *= #00.200; / define region RAM1_region *= #00.200; / define block USTACK with alignment *= &, size =ICTEDIT_region_RAM1_start toICTEDIT_region_RAM1_end]; / define block USTACK with alignment *= &, size =ICTEDIT_size_ctack {}; / initialize by copy { section .editor section section .editor section .editor section .editor =ICTEDIT_size_ctack {}; / section .editor section .editor section .editor =ICTEDIT_size_ctack {}; / define block USTACK with alignment *= &, size =ICTEDIT_size_ctack {}; / define block USTACK with alignment *= &, size =ICTEDIT_size_ctack {}; / place at address memICTEDIT_intvestart { readomits,</pre>	<pre>define symbolCloseliste_edg. define memory mem with size = 40; define memory mem with size = 40; place in MMU_region { readerity, j; place in MMU_region { readerity, block STACK, block STACK,</pre>		define symbolICFEDII_size_cstack_ =	= 0x400;				
<pre>/*** to of it vector settion.settine**/ define memory mem with size = 4G; define block HAPA</pre>	<pre>//*** rue of if end/setion.setion.set/setion.setion.setions/setion_setion_col_start to _CTEDIT_region_ROM_end;; define negoon ROM_region = seti[from _LTEDIT_region_ROM_start to _CTEDIT_region_ROM_end;; define block HARA = with alignment = 0, size = _LTEDIT_size_heap { } { }; initialize by copy { reducing_ent = 0, size = _LTEDIT_size_heap { } { }; si</pre>		define symbolICFEDII_size_heap	= 0x200;				
<pre>define region MM1 region</pre>	<pre>define memory memo</pre>		/**** End of ICF editor section. ###IC	-###*/				
<pre>define region RAWIregion * eme:[from _LIFEDIT_region_RAWistart_ to _LFEDIT_region_RAWiend_]; define region RAWIregion * eme:[from _LIFEDIT_region_RAWiend_]; define region RAWIregion * eme:[from _LIFEDIT_region_RAWiend_]; define region RAWIregion * eme:[from _LIFEDIT_region_RAWiend_]; initialize by copy { section .moint }; place at address mem:_LIFEDIT_region_RAWiend_;; place at address mem:_LIFEDIT_region_RAWIstart_ { (readonly section .intruce); place in RAWIregion * eme:[from @LIFFEEDIT_emissions_ place in RAWIregion = mem:[from @LIFFEEDIT_size_readon]; j; place in RAWIregion = mem:[from @LIFFEEDIT_size_readon]; gettint_section_statistics * @ *.edu(smit_from_section_); j; place in RAWIregion = mem:[from @LIFFEEDIT_size_readon]; define region RAWIregion = mem:[from @LIFFEEDIT_size_readon]; define region RAWIregion = mem:[from @LIFFEEDIT_size_readon]; if medit_section_statistics * @ *.edu(smit_from_section_); j;</pre>	<pre>offine memory mes with size + 49; define region RAM1_region</pre>							
<pre>define region ReW_region =</pre>	<pre>idefine region BOU_region to</pre>		define memory mem with size - 46:					
<pre>define region RAWLregion = mem:[from _ICFEDIT_region_RAWLtert_ to _ICFEDIT_region_RAWLend_]; define to lock CSTACK with alignment = 0, size = _ICFEDIT_size_cstack_ { };; initialize by copy { coderite, section .ndmitesting, section .ndmitesting, section .ndmitesting, section .ndmitesting, section .ndmitesting, place in RAW_region { readwrite, block CSTACK, block CSTACK,</pre>	<pre>define region RAW_region = mes:[freeICFEDIT_region_RAW_tent toICFEDIT_region_RAW_end_]; define block CSTACK with alignment = 0, size =ICFEDIT_size_cstack {}; initialize by copy { section .hdd_testim_unux_meeting, ;; do not initialize (section .noinit); place at address mesICFEDIT_Intvec_start { readonly section .intvec }; place in RAW_region { readonly, ;; place in RAW_region { readonly, ;; place in RAW_region { readonly, ;;</pre>		define region ROM region = mem:[fr	com ICEEDIT region	ROM start	to TO	FEDIT region POM end 1	
<pre>define block Kigger with alignment = 0, size =ICFEDIT_isze_cstack {}; define block KEAP with alignment = 0, size =ICFEDIT_isze_cstack {}; initialize by copy { redurite; section .noint }; place at address messICFEDIT_intvec_start_ { readonly section .intvec }; place in R04_region { readonly, }; define region R042_region = mess[from 0x1FFFE000 to 0x1FFFFFF]; place in R04_region = mess[from 0x1FFFE000 to 0x1FFFFFF]; place in R042_region = mess[from 0x1FFFE000 to 0x1FFFFFF]; messed introduces = 0 = mess[from 0x1FFFE000 to 0x1FFFFFF]; messed introduces = mess[from 0x1FFFE000 to 0x1FFFFFF]; messed introduces = mess[from 0x1FFFE000 to 0x1FFFFFF]; messed internation = mess[from 0x1FFFE000 to 0x1FFFFFF]; messed internation = mess[from 0x1FFFFF]; messed internation = mess[from 0x1FFFFF]; messed internation = mess[from 0x1FFFFFF]; messed int</pre>	<pre>define lock CSTACK with alignment = 0, size = _ICFEDIT_size_cstack{}; define block HEAP with alignment = 0, size = _ICFEDIT_size_heap{}; initialize by copy { section _BML_testB_torosection; } do not initialize (section .noinit); place in R0M_region { readonly, place in R0M_region = xem:[frem doIFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFFFFFFFFFFFFF]; place in R0M_region = xem:[frem doIFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</pre>		define region RAM region = mem:[fr	com ICEEDIT_region	RAM start	to	FEDIT region RAM end 1:	
<pre>define block CSTACK with alignment = 0, size =ICFEDIT_size_cstack { };; initialize by copy { readomite. readomite. readomite. readomite. readomite. readomite. readomite. readomite. readomite. place at address memICFEDIT_intvec.start_ { readonly section .intvec }; place in RAM_region { readomite. readomite. readomite. readomite. readomite. readomite. place in RAM_region { readomite. readomite. block KIRAP, block KIRAP, bl</pre>	<pre>define block CSTACK with alignment = 0, size =ICFEDIT_size_cstack {}; define block CSTACK with alignment = 0, size =ICFEDIT_size_heap {}; initialize by copy { readerite. section .noint }; place at address mes:ICFEDIT_intve_start { readonly section .intvec }; place at address mes:ICFEDIT_intve_start { readonly section .intvec }; place in RAW_region { readonly, ; j; define region RAW2_region = mes:[from 0xIFFFF0000 to 0xIFFFFFF]; place in RAW_region = mes:[from 0xIFFF0000 to 0xIFFFFFF]; place in RAW_region = mes:[from 0xIFFF0000 to 0xIFFFFFF]; place in RAW_region = mes:[from 0xIFFF0000 to 0xIFFFFFF]; place in RAW2_region = mes:[from 0xIFFF0000 to 0xIFFFFFF]; prestriction_text[data + sid text],</pre>				in jour c_	··· _··	i cori_i cgron_iai_cita_];	
<pre>define block HEAP with alignment = 0, size = _ICFEDIT_size_heap {}; initialize by copy { readwrite, section : [d]_readwrite(); } do not initialize { section .noinit }; place in RAM_region { readwrite, block trafk, blo</pre>	<pre>define block HEAP with alignment = 0, size = _ICFEDIT_iIze_neap { }; initialize by copy {</pre>		define block CSTACK with alignment	= 8. size = ICFEDIT	size cstack	{ }:		
<pre>initialize by copy { readwrite, section , wid (westifund, section,); do not initialize (section .nonit); place in ROM_region { readwrite, black draws mesLICEDUT_intex_start_ { readonly section .intexc }; place in ROM_region { readwrite, black draws, b</pre>	<pre>initialize by copy { readorize, j; readorize, j; do not initialize { section .noint }; place at address memICFEDT_Intvec_start_ { readonly section .intvec }; place in RMM_region { readorize, block CSTACK, block CSTACK,</pre>		define block HEAP with alignment	= 8, size = ICFEDIT	size heap	- 83		
<pre>initialize ty copy { section _sold_testi func_section; section _sold_testi func_section; section _sold_testi func_section; place at address mem:_ICFEDIT_intvec_start_ { readonly, place in ROM_region { readonly, place in ROM_region { readonly, }; place in ROM_region { readonly, }; place in ROM_region { readonly, }; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFFF]; place in ROM_region =</pre>	<pre>initialize by copy { section .modifies.section; section .modifies.func.section; section .modifies.func.section; place in ROM_region { readomite, re</pre>		in the second seco					
<pre>// 'readwrite, section .mddfsest3fong section, } do not initialize { section .mdii(section; place an address mes:_ICFEDIT_intvec_start_ { readonly section .intvec }; place in R0M_region { readwrite, black tin R0M_region = see:[from 0x1FFF8000 to 0x1FFF8000 to 0x1FFF8FF]; place in R0M_region = see:[from 0x1FFF8000 to 0x1FFF8000 to 0x1FFF8FF]; place in R0M_region = see:[from 0x1FFF8000 to 0x1FF8000 to 0x1F8000 to 0x1F</pre>	<pre>// 'readwrite section .mdd(section,); do not initialize (section .noinit); place at address memICFEDIT_intvestart_ (readonly section .intvec); place in ROM_region { readonly, j; place in ROM_region { readon'ite, black CSTACK, black CSTACK, bl</pre>		initialize by copy {					
<pre>section .bdd_text1 func_section, } d on ot initialize { section .noint }; place at address mem:ICFEDIT_intvec_start_ { readonly section .intvec }; place in RAM1_region { readonly, ; place in RAM2_region = mem:[from 8x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 8x1FF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 8x1FFF8000 to 0x1FFF8000 to 0x1FF8000 to 0x1FF8000</pre>	<pre>section .bdd_testlifume_section; } } do not initialize { section .noini }; place at address mes:_ICFEDIT_intec_start_ { readonly section .intec }; place in ROM_region { readonly, ; place in ROM_region { readonly, ; place in ROM_region { readonly, }; place in ROM_region = mes:[from 0x1FFFF0000 to 0x1FFFFFFF]; place in ROM_region = mes:[from 0x1FFFF0000 to 0x1FFFFFFF]; place in ROM_region = mes:[from 0x1FFFF0000 to 0x1FFFFFFFF]; place in ROM_region = mes:[from 0x1FFF0000 to 0x1FFFFFFF]; place in ROM_region = mes:[from 0x1FFF0000 to 0x1FFFF000 to 0x1FFFFFFF]; place in ROM_region = mes:[from 0x1FFF0000 to 0x1FFFF000 to 0x1FFFFF000 to 0x1FFFFF000 to 0x1FFFFF000 to 0x1FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</pre>		readwrite.					
<pre>}; do not initialize { section .noinit }; place at address mess_ICFEDTT_intvec_start { readonly section .intvec }; place in RAMI_region { readonly, ; place in RAMI_region { freedomite, block tHAP, }; define region RAMI_region = sex:[from 0x1FFF80000 to 0x1FFFFFFF]; place in RAMI_region = sex:[from 0x1FFF80000 to 0x1FFFFFFFF]; place in RAMI_region = sex:[from 0x1FFF80000 to 0x1FFFFFFFF]; start = sex:[from 0x1FFF80000 to 0x1FFFFFFFF]; start = sex:[from 0x1FFF80000 to 0x1FFFFFFFF]; start = sex:[from 0x1FFF80000 to 0x1FFFF80000 to 0x1FFFF80000 to 0x1FFF80000 to 0x1FFF800</pre>	<pre>}; d on to initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start_ { readonly section .intvec }; place in RAMI_region { readonly, ; place in RAMI_region { readonly, }; place in RAMI_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAMI_region = mem:[from 0x1FFF8000 to 0x1FFFFFFFF]; place in RAMI_region = mem:[from 0x1FFF8000 to 0x1FFFFFFFF]; place in RAMI_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAMI_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAMI_region = mem:[from 0x1FFF8000 to 0</pre>		section .add test1	func section,				
<pre>do not initialize { section .noinit }; place at address mes:_ICFEDIT_intvec_start_ { readonly section .intvec }; place in ROM_region { readonly, ; place in ROM_region { readonly, }; place in ROM_region { readonly, }; place in ROM_region { readonly, }; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1FFFFFF; place in ROM_region = mem:[from 0x1FFFB000 to 0x1FFFFB000 to 0x1</pre>	<pre>do not initialize { section .noint }; place at address mem:_ICFEDIT_intvec_start_ { readonly section .intvec }; place in ROW_region { readonly, ; place in ROW_region { readonly, ; place in ROW_region { readonly, ; place in ROW_region { readonly, ; place in ROW_region at the ROW_region {</pre>		};					
<pre>do not initialize { section .noinit }; place at address mem:_ICFEDIT_intvec_start_ { readonly section .intvec }; place in ROM_region { readonly, ;; place in RAM_region { readonly, ;; block tSTACK, block tSTACK,</pre>	<pre>do not initialize { section .noinit }; place at address mems_ICFEDIT_intvec_start_ { readonly section .intvec }; place in ROM_region { readonly, ;; place in ROM_region { readonly, ;; place in ROM_region { readonly, }; define region ROM2_region = mems[from 0x1FFF8000 to 0x1FFFFFFF]; place in ROM2_region = mems[from 0x1FFF8000 to 0x1FFFFFFF]; proper default_feetLater 0x1FFFFFFF; section .madd_testI_feetLater 0x1FFFFFFF; section .madd_testI_feetLater 0x1FFFFFF; section .madd_testI_feetLater 0x1FFFFFF; section .madd_testI_feetLater 0x1FFFFFFF; section .madd_t</pre>							
<pre>place at address mem:_ICFEDIT_intvec_start_ { readonly section .intvec }; place in ROM_region { readonix, j; place in RAM_region { readonix, block KIRAP, }; define region RAM2_region = mem:[from 8x1FFF88000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 8x1FFF88000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from sciffer section, }; forgend default_function_stributes - 0*dd_testI_func_section; j; forgend default_function_test2(det s, int b) forgend default_function_test2(det s, int b) forgend</pre>	<pre>place at address mem:_ICFEDI_intvec_start_ { readonly section .intvec }; place in RAM_region { readonly, ; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFF6000 to 0xIFFFF6000 to 0xIFFFFFF;; place in RAM_region = test:[fom 0xIFFF6000 to 0xIFFFF6000 to 0xIFFFF6000 to 0xIFFFF6000 to 0xIFFF6000 to 0xIFF600 to 0</pre>		<pre>do not initialize { section .noinit };</pre>	;				
<pre>place at address mem:_ICFEDT_intvec_start { readonly section .intvec }; place in ROM_region { readonly, ; place in ROM_region { readonly, ; place in ROM_region = mem:[from @x1FFF80000 to @x1FFFFFFF; block CSTACK, block HEAP, }; definite region RAM2_region = mem:[from @x1FFF80000 to @x1FFFFFFF; place in RAM2_region = mem:[from @x1FFF8000 to @x1FFFFFFF; place in RAM2_region = mem:[from @x1FFF8000 to @x1FFFFFFF; place in RAM2_region = mem:[from @x1FFF8000 to @x1FFFFFFF; place in RAM2_region = mem:[from @x1FFFFFF; place in RAM2_region = mem:[from @x1FFFFF; place in RAM2_region = mem:[from @x1FFFF; place in RAM3_region = mem:[from @x1FFFF; place in RAM3_region = mem:[from @x1FFFF; place in RAM3_region = mem:[from @x1FFF; place in RAM3_region = mem:[f</pre>	<pre>place at address memICFEDT_intvec_start { readonly section .intvec }; place in ROM_region { readonly, ; place in ROM_region { readonly, }; define region RAM2_region = mem:[from 0xIFFF8000 to 0xIFFFFFFF]; place in RAM2_region = mem:[from 0xIFFF8000 to 0xIFFFFFFF]; prome 0xIFFF8000 to 0xIFFFFFFF];</pre>							
<pre>place in ROM_region { readonly, }; place in RAM_region { readonly, re</pre>	<pre>place in ROM_region { readonly, ; place in ROM_region { readonly, ; place in ROM_region { readonly, }; place in ROM_region { readonly, readonly, }; place in ROM_region { readonly, readonly, }; place in ROM_region { readonly, readonly,</pre>		place at address mem:ICFEDIT_intvec_:	start { readonly se	ction .intve	c };		
<pre>place in ROM_region { readonly, ; place in ROM_region { readonly, ; place in ROM_region { readonly, }; place in ROM_region { readonly, }; place in ROM_region { readonly, }; place in ROM_region { readonly, readonly, }; place in ROM_region { readonly, readonly, }; place in ROM_region #mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in ROM_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in ROM_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in ROM_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in ROM_region = mem:[from 0x1FFF8000 to 0x1FFFFFFFF]; place in ROM_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFF800 to 0x1FF800 to 0x1FF800 to 0</pre>	<pre>place in ROM_region { readonly, ; place in RAM_region { readonly, ; place in RAM_region { readonly, }; place in RAM_region { readonly, readonly, }; }; define region RAM2_region { resting of solution is add_testl_func_section; }; }; define region RAM2_region { resting of solution is add_testl_func_section; }; return add_testl_function_strike = 0; return add_testl_func_section; return add_testl_func_section; return add_testl_func_section; return add_testl_function_settilite = s; add_testl_func_section; return add_testl_function_settilite = s; return add_testl_function_settilites = i; return add_testl_functin_settilites = i;</pre>							
<pre>readonly, }; place in RAM_region { readonly, place in RAM_region { readonly, block CSTACK, block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFFF800 to 0x1FFF8000 to 0x1FFFF8000 to 0x1FFF8000 to 0x1FFF8000 to 0x1FFF8000 to 0x1FFF8000 to 0x1FFF8000 to 0x1FF8000 to 0x1F8000 to 0x1F8</pre>	<pre>readonly, }; place in RAM_region { readorite, block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x100 to 0x1</pre>		place in ROM_region {					
<pre></pre>	<pre> f ; place in RAM_region { readwrite, block CSTACK, block CSTACK, block CSTACK, block ISTACK, block ISTACK,</pre>		readonly,					
<pre>place in RAW1_region { readwrite, block CSTACX, block HEAP, }; define region RAW2_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF]; place in RAW2_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF]; place in RAW2_region = mem:[from stributes = 0 *.add_test1_func_section, }; */ */ *****************************</pre>	<pre>place in RAM1_region { readwrite; block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8FF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 t</pre>		};					
<pre>place in kwr_region { feredwrite, block CSTACK, block CSTACK, block CSTACK, block CSTACK, block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region { section .add_test1_func_section, }; foregas default_function_stributes = 0 *.add_test1_func_section* int add_function_test2(afm *, int b) for add_test2_data2 * 0; printinf_mac2(); return add_test2_data2; int add_function_test2(afm *, int b) int add_function_test3(afm *, int b) int add_function_</pre>	<pre>place in NAM_region { readwrite, block CSTACK, block CSTACK, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFFF8000 to 0x1FFF8000 to 0x1FF8000 to 0x1FFF8000 to 0x1FF8000 to 0x1F</pre>		alars in Daw series (
<pre>block HEAP, block HEAP, }; define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; j; fprages default_function_stributes - # ".add_testi_func_section" int add_function_test2(int a, int b) for int dd_function_test2(int a, int b) for int dd_function_test3(int a, int b) f add_test2.dstal + add_test2.dsta2; if add_function_test3(int a, int b) f add_test2.dstal + add_test2.dsta2; if add_test2.dsta1 + a</pre>	<pre>block HEAP, }; define region RAM2_region = mem:[from 0x1FFFF88000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFFF88000 to 0x1FFFFFF]; place in RAM2_region { section .add_test1_func_section; };</pre>		place in RAM_region {					
<pre>block HEAP; }; define region RAM2_region = mem:[from 0x1FFFB000 to 0x1FFFFFF]; j; define region RAM2_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF]; j; section .add_test1_func_section; j; */ forgen default_function_stributes = 0*.add_test1_func_section* int add_function_test2(int a, int b) { define region RAM2_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF]; j; define region RAM2_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF]; j; for add_test2.deta1 = add_test2_deta2; return add_test2.deta1 = add_test2_deta2; return add_test2.deta1 = add_test2_deta2; return add_test2.deta1 = add_test2_deta2; } for add_test2.deta1 = add_test2_deta2; return add_test2.deta1 = add_test2_deta2; return add_test2.deta1 = add_test2_deta2; return add_test2.deta2 = b; return add_test2.deta2 = b; return add_test2.deta2 = b; return add_test2.deta2 = c; return add_test2.deta1 = add_test2.deta2; return add_test2.deta2 = c; return add_test2.d</pre>	<pre>block clink.c, block HEAP, }; define region RAW2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAW2_region { section .add_test1_func_section, };</pre>		hlack CETACK					
<pre>}; define region RAM2_region = mem:[from 0x1FFFF8000 to 0x1FFFFFFF]; place in RAM2_region { section .add_test1_func_section, };</pre>	<pre>}; define region RAM2_region = mem:[from 0x1FFFB000 to 0x1FFFFFFF; place in RAM2_region { section .add_test1_func_section; }; */ fint add_fmction_test2(int a, int b) { dd_test2_data1 + add_test2_data2; return add_test2_data2 + b; printf_func2(); return add_test2_data1 + add_test2_data2; return add_test2_d</pre>		block HEAD					
<pre>// define region RAM2_region = mem:[from 0x1FFF58000 to 0x1FFFFFFF;; place in RAM2_region { section .add_test1_func_section; }; // fprogma default_function_attributes = 0 *.add_test1_func_section* int add_function_test2(ist a, ist b) (add_test2.dsta1 * add_test2.dsta2; return add_test2.dsta1 * add_test2.dsta2; return add_test2.dsta2 * b; add_test2.dsta1 * add_test2.dsta2; return add_test2.dsta2; return add_test2.dsta2 * b; add_test2.dsta1 * add_test2.dsta2; return add_test2.dsta2;</pre>	<pre>// define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; place in RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF]; }; // section .add_test1_func_section; // section .add_test2_deta1 = add_test2_deta2; // return add_test2_deta1 = add_test2_deta2; // return add_test2_deta2 = s; // return add_test2_deta1 = add_test2_deta2; // return add_test2_deta3; // return add_test2_deta</pre>		}.					
<pre>define region RAW2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF; place in RAW2_region { section .add_test1_func_section, }; // forage default_function_attributes = 0 ".add_test1_func_section" if add_fenction_test2(ist a, int b) if direct data1 = add_test2_data2; if add_fenct2.data1 + add_test2.data2; if add_fenct2.data1 + add_test2.data2; if return add_test2.data1 + add_test2.data2; if return add_test2.data2; if return add_test2.data1 + add_test2.data2; if return add_test2.data2; if return add_test2.data1 + add_test2.data2; if return add_test2.data2; if return add_test2.data1 + add_test2.data2; if return add_test2.data1 + add_test2.data2; if return add_tes</pre>	<pre>define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFFF; place in RAM2_region { section .add_test1_func_section, };</pre>		13					
<pre>place in RA42_region { section .add_test1_func_section; set 0 close() 0000 Read/Wit 0 close() Read/Wit 0 close() 0000 Read/Wit 0 close() cl</pre>	<pre>place in RAP12_region { section .add_test1_func_section; section .add_test1_func_section; section .add_test1_func_section; section .add_test1_func_section; section .add_test1_func_section; int add_function_test2(int a, int b) { dd_test2.dsta1 = add_test2.dsta2; return add_test2.dsta2; int add_function_test3(int a, int b) dd_test2.dsta2 = b; return add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta2; return add_test2.dsta2; int add_test2.dsta2 = b; return add_test2.dsta2 = b; return add_test2.dsta2; int add_test2.dsta2 = b; return add_test2.dsta1 = add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta1 = add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta1 = add_test2.dsta2; int add_test2.dsta1 = add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta1 = add_test2.dsta2; int add_test2.dsta2; int add_test2.dsta2;</pre>		define region RAM2 region = mem:[from (0x1FFF8000 to 0x1FFFF	FFF1:			
<pre>section .add_test1_func_section, }; section .add_test1_func_section, }; section .add_test1_func_section, int add_fenction_attributes = @ ".add_test1_func_section" int add_fenction_test2(int a, int b) section 2.dst1 = add_test2.dst2; return add_test2.dst2 = b; a</pre>	<pre>section .add_test1_func_section, }; section .add_test1_func_section, }; section .add_test1_func_section, }; section .add_test1_func_section, }; section .add_test1_func_section, section .add_test1_function_test3[func_section, .est] section .add_test2_funct_section, section .add_te</pre>		place in RAM2 region {					
<pre>}; foreges default_function_attributes = @ ".add_test1_func_section" int add_function_test2(int *, int b) foreges default_function_test2(int *, int b) foreges default_function_test3(int *, int *</pre>	<pre>}; forages default_function_stributes = @ ".add_test1_func_section" int add_function_test2(ist a, int b) forages default_function_test2(ist a, int b) forages default_function_attributes = forages default_function_attributes</pre>		section .add te	st1 func section,				
<pre>></pre>	<pre>>; progge default_function_attributes = @ ".add_test1_func_section" progge default_function_attributes = @ ".add_test1_func_section" progge default_function_test2(int a, int b) { progge default_function_test2(int a, int b) printf_func2(); return add_test2.data1 + add_test2.data2; return add_test2.data1 = a; add_test2.data1 = a; add_test2.data2 = b; return add_test2.data2; return add_test2</pre>		};					
<pre></pre>	<pre> # # # # # # # # # # # # # # # # # # # # #</pre>							
# # 0x200*0446 PeadWet 0x1111*0044 Pailor 0004 Pailor 0004 # # 0x200*0467 PeadWet 0x1111*0044 Pailor 0004 Pailor 0004 # # 0x200*0467 PeadWet 0x1111*0044 Pailor 0004 Pailor 0004 # # 0x200*0467 PeadWet 0x1111*0044 Pailor 0004 Pailor 0004 # # 0x200*0467 PeadWet 0x1111*0044 Pailor 0004 Pailor 0004 # # 0x200*0467 PeadWet 0x1111*0044 Pailor 0004 Pailor 0004 # # # # 0x100*0407 Pailor 0004 Pailor 0004 </th <th>*/ # 0x500 %440 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x111 % PeedWa PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa PeedWa 0x1111 % PeedWa PeedWa # PeedWa 0x100 % PeedWa 0x1111 % PeedWa PeedWa</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>	*/ # 0x500 %440 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x111 % PeedWa PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa PeedWa 0x1111 % PeedWa # 0x100 % PeedWa 0x1111 % PeedWa PeedWa 0x1111 % PeedWa PeedWa # PeedWa 0x100 % PeedWa 0x1111 % PeedWa PeedWa							
program default_function_attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @ ".add_test1_func_section" profit feed of the section attributes = @	# 22 0x2000 '0440 PeedVma 0x11ff '004 0x100 '0x7 Cc2 prant_[unc2] #pragma default_function_stributes = @ ".add_test1_func_section" 82 0x2000 '0440 PeedVma 0x11ff '004 0x100 '0x7 Cc2 prant_[unc2] int add_function_test2(int a, int b) 84 0x1000 '140 PeedVma 0x11ff '004 0x100 '0x7 Cc2 prant_[unc2] dd_test2.data1 exd status 87 0x2000 '140 PeedVma 0x11ff '004 0x100 '0x7 Cc2 prant_[unc2] dd_test2.data2 = b; printf_func2(); return add_test2.data2; 89 0x1000 '140 PeedVma 0x11ff '001 0x7 f1 0x1ff '011 0x5 0x6 0x1 0x7 f1 0x1ff '011 0x1ff '0111 0x1ff '011 0x1ff '011 0x1ff '011 0x1ff '011	F 1						
#progens default_function_stributes = @ ".add_text1_func_section" #3 000000 0000 Feed/With Feed	<pre>\$7 0x800 0000 000 000 000 000 000 000 000</pre>			R2	0x2000'04f8	ReadWrite	0x1fff'8004: 0x1000'0be7 DC32	printf_func2
int add_function_test2(int a, int b) is 0.atbcb.ic200 PeadWam Diff(1 State at-2 int b) Diff(1 State at-2 int b) id_dtest2.deta2 = b; id_dtest2.deta2 = b; int add_test2.deta2 = b; Diff(1 State at-2 int b) Diff(1 State at-2 int b) Diff(1 State at-2 int b) int add_test2.deta2 = b; Point function_test3(int a, int b) Pin(1 State at-2 int b) Diff(1 State at-2 int b) Diff(1 State at-2 int b) int add_test2.deta1 = a; add_test2.deta1 = a; add_test2.deta2 = b; Pin(1 State at-2 int b) Diff(1 State at-2 int b) Diff(1 State at-2 int b) if 0.atbcb.ic200 PeadWam Diff(1 State at-2 int b) Diff(1 State at-2 int b) Diff(1 State at-2 int b) int add_function_test3(int a, int b) Pin(1 State at-2 int b) Pin(1 State at-2 int b) Pin(1 State at-2 int b) Diff(1 State at-2 int b) id_dtest2.deta1 = a; add_test2.deta2; SP Outcol State at-2 int b) Diff(1 State at-2 int b) Diff(1 State at-2 int b) if add_test2.deta1 + add_test2.deta2; SP Outcol State at-2 int b) Diff(1 State	int add_function_test2(int a, int b) 9 Outbab 2:00 PeedWm Viiii 2:00 PeedWm Viiii 2:00 PeedWm adest2_dstal = a add_est2_dstal = a(add_est2_dstal = a) B6 Outbab 2:00 PeedWm Viiii 2:00 PeedWm	<pre>#pragma default_function_attributes</pre>	<pre>= @ ".add_test1_func_section"</pre>	R4	0x0000'0001 0x1000'1140	ReadWrite	add_runction_test2: 0x1fff'8008: 0xb510 PUSH	(R4, LR)
# feed_unit con_texture a, int or/ # 6 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win # finite con_texture a, int or/ # 8 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win printf_func2(); # 8 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win printf_func2(); # 8 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win # NFSR 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win Outlit 1000C 10000/0000 int add_function_text3(int a, int b) # NFSR 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win id_text2.data1 = a; add_text2.data1 = a; add_text2.data1 = a; add_text2.data1 + add_text2.data2; # NFSR 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win # RFINAR 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win int add_function_text3(int a, int b) # NFSR 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 Peed/Win int add_function_text3(int a, int b) # RFINAR 0x0000/0000 Peed/Win Outlit 1000C 10000/0000 <td># feed_unition_test_data =</td> <td>dat add function tost?/dat - dat b</td> <td>A</td> <td>R5</td> <td>0x0beb'c200</td> <td>ReadWrite</td> <td>0x1fff'800a: 0x4c08 LDR.N</td> <td>R4, ??DataTable16</td>	# feed_unition_test_data =	dat add function tost?/dat - dat b	A	R5	0x0beb'c200	ReadWrite	0x1fff'800a: 0x4c08 LDR.N	R4, ??DataTable16
wid text	ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = 0; ist test2.deta1 = add_test2.deta2; ist test2.deta2 = 0; ist test2.deta1 = a; ist test2.deta2 = 0; ist test2.deta1 = a; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta1 = add_test2.deta2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta1 = add_test2.deta2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta1 = add_test2.deta2; ist test2.deta2 = 2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta1 = add_test2.deta2; ist test2.deta2 = 2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta1 = add_test2.deta2; ist test2.deta2 = 2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta1 = add_test2.deta2 = 0; ist test2.deta2 = 2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta2 = 2; ist test2.deta2 = 2; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta2 = 0; ist test2.deta2 = 2; ist test2.deta2 = 2; ist test2.de	<pre>int add_function_test2(int a, int b </pre>	7	R6	0x0000'0000	ReadWrite ReadWrite	0x1111 800C: 0x8020 SIK	RU, [R4] P1 [P4 #0+4]
add_test2.data2 = b; P9 0x10001084 Read/Wm 0x1111*0014 0x6800 LDR R1 R1 printf_fun2(); return add_test2.data1 + add_test2.data2; R10 0x00001000 Read/Wm 0x1111*001.6 %x6401 LDR R1 R1 0x00001000 Read/Wm 0x1111*001.6 %x6401 LDR R1 R1 0x00001000 Read/Wm Dx1111*001.6 %x6401 LDR R1 R1 <td>add_test2.data2 = b; P9 0x100*1084 RecM/mc 0x11ff*0014 0x620 LDR R0 [R4.f0x1] printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 0x660*0000 R0 R1.f R0.f0x1 0x620 LDR R0.fR4.f0x1 printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 0x660*0000 R0 R1.fR4.f0x1 printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 R0.fR4.f0x1 R1.fR4.f0x1 printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 R0.fR4.f0x1 R1.fR2 R0.fR4.f0x1 printf_fun2(); R1FR6 0x000*0000 RecM/mc 0x11ff*0016 R0.fR4.f0x1 R1.fR2 R0.fR4.f0x1 int add_function_test3(int a, int b) FFR6 0x100*0000 RecM/mc 0x11ff*0016 R0.fR1.fR2 R1.fR2 Pox1 idd_test2.data1 = a; add_test2.data1 = a; 0x100*0407 RR.fR2 0x100*0407 RR.fR2 R1.fR2 R1.fR2<</td> <td>add_test2.data1 = a;</td> <td></td> <td>R8</td> <td>0x0000 0000</td> <td>ReadWrite</td> <td>0x1fff'8010: 0xf7ff 0xfff6 BL</td> <td>?Veneer 2 (6) for</td>	add_test2.data2 = b; P9 0x100*1084 RecM/mc 0x11ff*0014 0x620 LDR R0 [R4.f0x1] printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 0x660*0000 R0 R1.f R0.f0x1 0x620 LDR R0.fR4.f0x1 printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 0x660*0000 R0 R1.fR4.f0x1 printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 R0.fR4.f0x1 R1.fR4.f0x1 printf_fun2(); R11 0x000*0000 RecM/mc 0x11ff*0016 R0.fR4.f0x1 R1.fR2 R0.fR4.f0x1 printf_fun2(); R1FR6 0x000*0000 RecM/mc 0x11ff*0016 R0.fR4.f0x1 R1.fR2 R0.fR4.f0x1 int add_function_test3(int a, int b) FFR6 0x100*0000 RecM/mc 0x11ff*0016 R0.fR1.fR2 R1.fR2 Pox1 idd_test2.data1 = a; add_test2.data1 = a; 0x100*0407 RR.fR2 0x100*0407 RR.fR2 R1.fR2 R1.fR2<	add_test2.data1 = a;		R8	0x0000 0000	ReadWrite	0x1fff'8010: 0xf7ff 0xfff6 BL	?Veneer 2 (6) for
printf_fun2(); R10 0x0000/1444 Peed/MM 0x1ff*00.6 0x6661 LDR R1. [R4. #0x4] preturn add_test2.data1 + add_test2.data2; R4PGR 0x0000/1444 Peed/MM 0x1ff*00.6 0x6615	printf_fun2(); print	add_test2.data2 = b;		R9	0x1000'108d	ReadWrite	0x1fff'8014: 0x6820 LDR	R0. [R4]
Peturn add_test2.datal + add_test2.data2; Pipe Own2000 Feed/Weight Diff #Dia_ Dat/10 Diff #Dia_ Dat/10 Diff #Dia_ Dat/10 int add_test2.datal + add_test2.data2; int add_function_test3(int a, int b) PC 0x11ff #Dia_ Dat/10 Diff #Dia_ Dat/10 <td>File Openotion Proc int add_test2.datal + add_test2.data2; int add_fenttim_test3(int a, int b) Int add_fenttim_test3(int a, int b) int add_test2.datal = a; int add_fenttim_test3(int a, int b) PeedWm Outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; outiff*800 ex600 STM No R23 ADG intetern add_test2.datal = a; outiff*800 ex600</td> <td>printf func2():</td> <td></td> <td>R10 R11</td> <td>0x0000'1f44</td> <td>ReadWrite BeadWrite</td> <td>0x1fff'8016: 0x6861 IDR 0x1fff'8018: 0x1808 ADDS</td> <td>R1, [R4, #0x4] R0 R1 R0</td>	File Openotion Proc int add_test2.datal + add_test2.data2; int add_fenttim_test3(int a, int b) Int add_fenttim_test3(int a, int b) int add_test2.datal = a; int add_fenttim_test3(int a, int b) PeedWm Outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; add_test2.datal = a; outiff*800 ex600 STM No R23 int add_test2.datal = a; outiff*800 ex600 STM No R23 ADG intetern add_test2.datal = a; outiff*800 ex600	printf func2():		R10 R11	0x0000'1f44	ReadWrite BeadWrite	0x1fff'8016: 0x6861 IDR 0x1fff'8018: 0x1808 ADDS	R1, [R4, #0x4] R0 R1 R0
stars 0x0000/0000 ReadVim dd_tuction_test3: int add_function_test3(int a, int b) int sdg_function_test3(int a, int b) 0x1107/0000 FeedVim 0x1117/0010 FeedVim FeedVim FeedVim 0x1117/0010 FeedVim Feed	return add_test2.datal + add_test2.datal; # AFGR 0x0000'0000 PeadVMm add_tunction_test3: int add_function_test3(int a, int b) # IFGR 0x0000'0000 PeadVMm 0x11ff'00ic 0x4600 STR R1 (R2, 10x4) id	prance (/)		R12	0x0000'0000	ReadWrite	Ox1fff'801a: 0xbd10 POP	(R4 PC)
<pre></pre>	<pre></pre>	<pre>return add_test2.data1 + add_test</pre>	2.data2;	* APSR	0x0000'0000x0	ReadWrite	add_function_test3:	P2 22P-4-T-11-16
FC Oxiff*000 PeadWark Oxiff*002 DesD/3 STR El. [82. 90x] add_test2.dstal = a; add_test2.dstal = b; add_test2.dstal = b; return add_test2.dstal + add_test2.dstal; IDR R0. [R2] IDR R0. [R2] return add_test2.dstal = add_test2.dstal; BASEPRI_MAX 0x1000*0407 ReadWark Ox11f*002* ReadWark Ox11f*002* ReadWark DDIF*000* ReadWark return add_test2.dstal + add_test2.dstal; BASEPRI_MAX 0x0000*0000 ReadWark Ox11f*000* RAF70 BX RR return add_test2.dstal + add_test2.dstal; BASEPRI_MAX 0x0000*0000 ReadWark Ox11f*000* RAF70 BX RR RE return add_test2.dstal + add_test2.dstal; BASEPRI_MAX 0x0000*0000 ReadWark Ox11f*000* RV70 BX BR RE return add_test2.dstal + add_test2.dstal; BASEPRI_MAX 0x0000*0000 ReadWark Ox11f*000* RV70 BX BR RE RE <td< td=""><td>ist add_function_test3(ist a, int b) PC Oxifif '000 ReadWar Oxifif '002 Strift E1, [22, 20k1] { {</td><td>- 3</td><td></td><td>± IPSR ± EPSR</td><td>0x0000'0000</td><td>ReadWrite</td><td>0x1fff 801c: 0x4803 IDR.N 0x1fff 801e: 0x6010 STR</td><td>R0. [R2]</td></td<>	ist add_function_test3(ist a, int b) PC Oxifif '000 ReadWar Oxifif '002 Strift E1, [22, 20k1] { {	- 3		± IPSR ± EPSR	0x0000'0000	ReadWrite	0x1fff 801c: 0x4803 IDR.N 0x1fff 801e: 0x6010 STR	R0. [R2]
B Control (100)	BP 0x200°0410 PeedVmm Ox111f*022 0x6810 LDR R0. R22 add_test2.data1 = a; add_test2.data2 = b; LR 0x100°0000 PeedVmm Ox111f*022 0x6810 LDR R0. R22.0 return add_test2.data2 = b; BASEPERT_KAX 0x0000°0000 PeedVmm Ox111f*022 0x6810 LDR R0. R22.0 return add_test2.data1 + add_test2.data2; ************************************	int add function test3(int a. int b)	PC	0x1fff'800a	ReadWrite	0x1fff'8020: 0x6051 STR	R1. [R2. #0x4]
add_test2.data1 = a; add_test2.data1 + add_test2.data2; ************************************	add_test2.dstal = a; add_test2.dstal = a;	₽{		SP	0x2000'04f8	ReadWrite ReadWrite	0x1fff'8022: 0x6810 LDR 0x1fff'8024: 0x6851 TDD	R0, [R2] R1 [R2 #0x4]
BaseRer Outpoint	#BacEFPI 0x0000'0000 Res/Mm 0x111f'1020 0x4770 EX LR return add_test2.data1 + add_test2.data2; *BacEFPI + add_test2.data1 + add_test2.data2; 0x0000'0000 Res/Mm 0x111f'1020 0x4770 EX LR *BacEFPI + add_test2.data1 + add_test2.data2; *BacEFPI + add_test2.data1 + add_test2.data2; Res/Mm 0x111f'1020 0x4700 EX LR *Fpragma default_function_attributes = CCTIMER 129'271 Res/Mm 0x111f'1020 6x000'0000 C22 add_test2 CCTIMER2 129'271 Res/Mm 0x111f'1020 6x000'0000 C22 0	add_test2.data1 = a;		PRIMASK	0x0000 '0000	ReadWrite	0x1fff'8026: 0x1808 ADDS	R0, R1, R0
return add_test2.data1 + add_test2.data2; m # ADLTTAGA 0x0000/0000 Peedv/mc 0x111: 0x2.ex0000 NOVS NO NO # #ADLTTAGA 0x0000/0000 Peedv/mc 0x111: 0x2.ex0000 NOVS NO NO # #ADLTTAGA 0x0000/0000 Peedv/mc 0x111: 0x2.ex0000 NOVS NO NO # pragma default_function_attributes = CYCLESCONFER 129'271 Peedv/mc Ox11f' 1002: 6x2000'0000 CC2 0 CYCLESCONFER 129'271 Peedv/mc Ox1ff' 1002: 6x2000'0000 CC2 0	#pragma default_function_attributes = #pragma default_function_attributes 0x1000 red/min 0x100 red/min 0x	aud_test2.uataz = 0;		BASEPRI DICEPPI NY	0x0000'0000x0	ReadWrite	0x1fff'8028: 0x4770 BX	LR Do Do
*) outrool root Pead/Mm Outfff102:0: 02:000 00:64 CC2 edutes12 #progma default_function_attributes = CCTLEREW 129:271 Read/Mm Outfff102:0: 02:000 00:600 00:000 CC22 0 CCTLEREW 129:271 Read/Mm Outfff102:0: 02:000 00:000 00:00 CC22 0	* CONTROL 0x0000'0004 ReadWint 0x11ff'802c: 8x000'00c4 BC32 add_test2 #pragma default_function_attributes = CCTLRECONTER 129'271 ReadWint 0x11ff'8030: 8x000'0000 DC32 0 CCTLRECONTER 129'271 ReadWint 0x11ff'8030: 8x000'0000 DC32 0 CCTLRER2 129'271 ReadWint 0x11ff'8030: 8x000'0000 DC32 0	<pre>return add_test2.data1 + add_test</pre>	2.data2;	BASEPRI_NAX FAULTWASK	0x0000'0000	ReadWrite ReadWrite	0x1ff'802a: 0x0000 HOVS ??DataTable16:	NU, NU
#progma default_function_attributes = CYCLECOUFTER 129'271 ReadOwly Ox1ff(18030 0x0000'0000 CC22 0 CCTIERER 129'271 ReadWark 0x1ff(18030 0x0000'0000 CC22 0 CCTIERER 129'271 ReadWark 0x1ff(18030 0x0000'0000 CC22 0	#pragma default_function_attributes = CCTIMER 129°221 ReadOmly Oxiff*030 Ex0000*0000 DC22 0 CCTIMER2 129°271 ReadWink Oxiff*030 8x0000*0000 DC22 0	L }		* CONTROL	0x0000'0004	ReadWrite	0x1fff'802c: 0x2000'00c4 DC32	add_test2
CLIERKI 127 274 PWBWYMW UXIII 8034 0X000 000 DC22 0	CCTIMER2 129'271 ReadWink 0x11ft 00x4 0x0000 0000 DC32 0	#pragma default function attributes	-	CYCLECOUNTER	129'271	ReadOnly	0x1fff'8030: 0x0000'0000 DC32	0
				CCTIMER2	129'271	ReadWrite	0x1fff'8038: 0x0000'0000 DC32	0

图 7-3: ICF 文件与重定向内容



注意: 使用#pragma 方法无法重定向中断服务函数及中断服务函数调用的函数, 仅_ramfunc 关键字可以重定向中断服务函数及中断服务函数调用的函数。

7.3 对整个文件进行重定向

假设目标文件为 test1.o、test2.o,将两个文件中的数据及代码进行重定向,此时将需要用 到 place in 指令,并且在重定向的文件中可以调用其他地址范围的函数。

对代码而言,将被重定向到 ICF 文件中包含有"section .text object test1.o"和"section .text object test2.o"的 region。

对于全局变量而言,非 const 全局变量将会被重定向到 ICF 文件中包含有 readwrite 的 region; const 全局变量将会被重定向到 ICF 文件中包含有 "section .rodata object test1.o" 和"section .rodata object test2.o"的 region;

对于局部变量而言,非 static 局部变量将会被重定向到 ICF 文件中包含有 block CSTACK 的 region; static 局部变量将会被重定向到 ICF 文件中包含有 readwrite 的 region;

如图 7-4 所示,程序代码被重定向到 0x1FFF8000~0x1FFFFFF 的 RAM 地址,是因为 ICF 文件 RAM2_region 是包含有 section .text object add_test.o 和 section .text object sub_test.o 的 region。形参 a 以及 b 将会被重定向到 RAM_region 中,因为这个 region 包含有 block CSTACK。非 const 且非 static 和 static 的全局变量 s_data 和 add_static1 将会被重定向到 RAM2_region 中,因为这个 region 包含有 readwrite。const 的全局变量 add_const1 将会被重定向到 RAM2_region 中,因为这个 region 包含有 "section .rodata object test1.o"。

(* Manager Depinger #/
/ - nemory Regions- //
define symbol ICFEDIT region ROM end = 0x1007FFF;
define symbolICFEDIT_region_RAM_start = 0x20000000;
define symbolICFEDIT_region_RAM_end = 0x20003FFF;
/# Since #/
/ -31425-/ define symbol ICFEDIT size cstack = 0x400:
define symbol ICFEDIT size heap = 0x200;
/**** End of ICF editor section. ###ICF###*/
define memory mem with size = 46.
define region ROM region = mem:[from ICFEDIT region ROM start to ICFEDIT region ROM end];
define region RAM_region = mem:[fromICFEDIT_region_RAM_starttoICFEDIT_region_RAM_end];
define block CSIACK with alignment = 0, Size =ICFEDI_Size_SSLACK { }; define block HEAP with alignment = 8, Size = ICFEDI_Size_SSLACK { };
initialize by copy {
readwrite,
section readta object add test.o.
section .text object sub_test.o,
section .rodata object sub_test.o,
D D
do not initialize { section .noinit };
place at address mem: ICFEDIT intvec start { readonly section .intvec };
place in ROM_region {
readonly,
, , , , , , , , , , , , , , , , , , ,
place in RAM_region {
readwrite,
section .rodata object add_test.o,
section .rodata object sub_test.o,
define region RAM2_region = mem:[from 0x1FFF8000 to 0x1FFFFFF];
place in RAM2_region {
section .text object a00_test.0,
};

图 7-4: ICF 文件与重定向内容



spc2168.idf sub_test.c add_test.c x	•	Registers 1		* û X	Disassembly			
add function tectlinit int	for	6-4	Comment COLUDer		0.11		a confidence of the second sec	
ada_idireitoin_cestifiire ind	10	Pind:	Group: Current CPO Keg	oters	Go to:	~	Zone: Memory	
static int add_static;	^	Name	Value	Access	Disessembly			
<pre>static int add_static1 = 9;</pre>			T LINE	muess	and a second sec	-		
static int add_static0 = 0;		RU	0×0000.000	6 Readwint	e Uxitti 802e: Ux6022	STR	R2, [R4]	
should add date back add backty		R1	0x2000'015	BeadWrit	0x1fff'8030: 0x181b	ADDS	R3, R3, R0	
struct add data test add testi;		R2	0x0000'000	 ReadWrit 	0x1fff'8032: 0x6063	STR	R3, [R4, #0x4	
struct dou_data_test dou_test2;		R3	0x0000'000	d ReadWrit	0x1fff'8034: 0x491f	LDR.	N R1, ??DataTa	
int a data.		R4	0x2000'011	 ReadWrit 	e 🔴 0x1fff'8036: 0x6809	LDR	R1, [R1]	
int s_dota)		R5	0x0beb'c20	0 ReadWrit	<pre>0x1fff'8038: 0x1809</pre>	ADDS	R1. R1. R0	
$\operatorname{Anc} S_{\operatorname{obc}} = 0$		R6	0x0000'000	0 ReadWrit	0x1fff'803a: 0x6021	STR	R1. [R4]	
THE PORTY - X)		R7	0x0000.000	0 ReadWhit	0x1fff'803c: 0x491c	TDR	N R1 22DataTab	
char "s string = "123456789\n";		DO	0+0000 ' 000	0 Readivit	0#1666 2003# 0#6809	TDP	P1 (P1)	
		DO	0x0000 000	E Doodithit	0=166610040. 0=1000	LDR	DI DI DO	
int add function test1(int a. int b)		R7	0x1000 010	5 Reduvville	0x1111 0040: 0x1005	ADDS	RI, RI, RU	
		R10	Ux0000'114	4 Readwint	Ux1ttt 8042: 0x6021	STR	R1, [R4]	
T int c = 6;		R11	0x0000,000	0 ReadWrit	e 0x1fff'8044: 0x491d	LDR	N R1, ??DataTal	
int d = 7;		R12	0x4000'882	8 ReadWrit	e 0x1fff'8046: 0x6809	LDR	R1, [R1]	
<pre>struct add_data_test add_test3;</pre>			0x0000,000	0 ReadWrit	6 0x1fff'8048: 0x1809	ADDS	R1, R1, R0	
		IPSR	0x0000'000	0 ReadWrit	e 0x1fff'804a: 0x6061	STR	R1, [R4, #0x4	
add_test3.data1 = a + b;		EPSR	0x0100'000	0 ReadWrit	0x1fff'804c: 0x491c	LDR .	N R1, ??DataTab	
_		PC	0x1fff'803	6 ReadWrit	6 0x1fff'804e: 0x6809	LDR	R1, [R1]	
add_test1.data1 = a;		SP	0x2000'055	0 ReadWrit	0x1fff'8050: 0x1809	ADDS	R1 R1 R0	
add_test1.data2 = b;	- 1	TP	0x1000'0b1	9 ReadWhit	0x1fff'8052 0x6021	STR	P1 [P4]	
		T DDTWACK	0×0000 ' 000	DearfW/it	Ow1fff'00E4 Ow491b	TTP	N P1 22DataTab	
add_test1.data1 = a + c;		T RIBBOR	0.00001000	0 Destablish	0-1666100056 0-6000	TDD	n ni, indicator	
add_test1.data2 = a + d;		DECEMPT N'S	0:00000 000	0 Read	0-166410050; 0 1000	LDR	DI DI DO	
		* BASEPRI_BAX	0x0000.000	U Readwint	UXIEFF 8058: UXI809	AUUS	RI, RI, RU	
add_test1.data1 = a + s_data;		# FAULTHASK	0x0000.000	0 Readwrit	0x1fff'805a: 0x6061	STR	R1, [R4, #0x4	
		* CONTROL	0x0000,000	4 ReadWrit	e Ox1fff'805c: 0x491a	LDR .	N R1, ??DataTab	
add_test1.dat21 = a + 5_data0;		CYCLECOUNTER	126'237	ReadOnly	/ 0x1fff'805e: 0x6809	LDR	R1, [R1]	
add_test1.dataz = a + s_data1;		CCTIMER1	126'237	ReadWrit	6 0x1fff'8060: 0x1809	ADDS	R1, R1, R0	
add test1 data1 - a + add coast0.		CCTIMER2	126'237	ReadWrit	6 0x1fff'8062: 0x6021	STR	R1. [R4]	
add_{test}		CCSTEP	3	ReadOnly	0x1fff'8064: 0x4919	LDR .	N R1, ??DataTab	
add_cestriatar = a + add_statics;								
add_function_test1 (int, int) f()) Er	ind: V Gr	roup: Current CPU Registers	G	a to:	Zone	e: Memory V 🕅	
static int add static:	Ne	ome	Value	Access	Disassembly			
static int add static1 = 9:	F	RO	0x0000'0006	ReadWrite	0x1fff'802e: 0x6022	STR	R2. [R4]	
<pre>static int add static0 = 0;</pre>		P1	0x200010060	BeadWhite	0x1fff'8030: 0x181b	ADDS	P3 P3 P0	
		P2	0x0000'000c	ReadWhite	0w1fff'9022: 0w6062	CTD	P3 [P4 #0+4]	
<pre>struct add_data_test add_test1;</pre>		R2	0x0000'000d	DeadWrite	0w1fff'9024 0w491f	TOP N	P1 22DataTable2 2	
<pre>struct add_data_test add_test2;</pre>	1	n J D A	0x0000 0000	Readblitte	0m1466'0026; 0m6000	TDD	P1 (P1)	
		K4	0x2000 0110	Pleadyvnie	URITET 8036: 0x6809	LDR	RI, [RI]	
int s_data;		85	UXUDED C200	Pleadwrite	UNITEL SU38: DE1809	ADDS	RI, RI, RU	
int s_data0 = 0;	F	R6	0x0000'0000	Headwinte	0x1fff'803a: 0x6021	STR	R1, [R4]	
int s_data1 = 1;	I F	R7	0x0000,0000	ReadWrite	0x1fff'803c: 0x491e	LDR.N	R1, ??DataTable3_3	
	F	R8	0x0000,0000x0	ReadWrite	0x1fff'803e: 0x6809	LDR	R1, [R1]	
char 's_string = 123456/89(n';	F	R9	0x1000'0fd5	ReadWrite	0x1fff'8040: 0x1809	ADDS	R1, R1, R0	
and a state of the	1	R10	0x0000'1f44	ReadWrite	0x1fff'8042: 0x6021	STR	R1, [R4]	
In add_infection_testifing a, inc b)	F	R11	0x0000'0000	ReadWrite	0x1fff'8044: 0x491d	LDR.N	R1, ??DataTable3_4	
int c = 6:	F	R12	0x4000'8828	ReadWrite	0x1fff'8046: 0x6809	LDR	R1, [R1]	
int d = 7:	E 2	APSR	0x0000'0000x0	ReadWrite	0x1fff'8048: 0x1809	ADDS	R1, R1, R0	
struct add data test add test3;	B 1	IPSR	0x0000'0000	ReadWrite	0x1fff'804a: 0x6061	STR	R1, [R4, #0x4]	
	EF	EPSR	0x0100'0000	ReadWrite	0x1fff'804c: 0x491c	LDR.N	R1, ??DataTable3 5	
add_test3.data1 = a + b;	I	PC	0x1fff'8076	ReadWrite	0x1fff'804e: 0x6809	LDR	R1, [R1]	
	9	SP	0x2000'0550	ReadWrite	0x1fff'8050: 0x1809	ADDS	R1, R1, R0	
add_test1.data1 = a;	1	LR	0x1000'0b19	ReadWrite	0x1fff'8052: 0x6021	STR	R1. [R4]	
add_test1.data2 = b;	I I III	PRTNASK	0×0000'0000	BeadWrite	0x1fff'8054: 0x491b	LDR N	R1 22DataTable3 6	
	a F	BASEPRI	0x0000'0000	ReadWhite	0x1fff'8056: 0x6809	TDR	P1 [P1]	
$auu_uvs(1,0373) = a + C;$		BASEPPT NAT	0x0000'0000	BeadWrite	0x1fff'8058 0x1809	ADDS	P1 P1 P0	
euo_test1.0#t82 = a + d;		FAILT TAASA	0×0000.0000	ReedWhite	0x1fff'905a: 0x6061	CTD	P1 (P4 #0~4)	
add test1.data1 = a + s data;		CONTROL	0#0000 0000	Read	0x1466'90Ec: 0x491-	TOP N	P1 22DataTabl=2 2	
000_CCF4408084 - 0 T 5_0805	1	CACIFCOUNTER	1261201	RoadOnho	0x1111 005C: 0x4718	TDP. R	p1 (p1)	
add testl.datal = a + s data0:	1	CICLECOURIER	126 301	Deadlibit	0x1111 0058: 0x0009	ADDC	RI, [RI] D1 D1 D0	
add_test1.data2 = a + s_data1;	1	COLLEGERI	126 301	Descritte	0x1111 0000: 0x1809	AUUS CTD	RI, RI, RU	
	9	CCITRER2	126.301	Pueadwrite	UMIIII 8062: UM6021	STR	R1, [K4]	
add_test1.data1 = a + add_const0;	0	CCSIEP	3	HeadUnly	UX1111 8064: UX4919	LDR . N	RI, 77DataTable3_8	
<pre>add_test1.data2 = a + add_static0;</pre>					Ux11ff 8066: 0x6809	LDR	R1, [R1]	
					0x1fff'8068: 0x1809	ADDS	R1, R1, R0	
<pre>add_test1.data1 = a + add_const;</pre>					0x1fff'806a: 0x6061	STR	R1, [R4, #0x4]	
add_test1.data2 = a + add_const1;					0x1fff'806c: 0x4918	LDR.N	R1, ??DataTable3_9	
					0x1fff'806e: 0x6809	LDR	R1, [R1]	
and rest1.data1 = a + add static;					0x1fff'8070: 0x1809	ADDS	R1, R1, R0	
<pre>auu_ttst1.0atd2 = a + add_stat101;</pre>					0x1fff'8072 0x6021	STR	R1 [R4]	
printf("string = %s\n", s string);					0x1fff'8074: 0x4917	LDR.N	R1, ??DataTable3_1	
printf funct():					0x1fff'8076: 0x6809	LDR	R1, [R1]	
					0x1fff'8078: 0x1808	ADDS	R0, R1, R0	
return add test1.data1 + add test1.data2 + add test3.data1;					0x1fff'807a: 0x6060	STR	R0, [R4, #0x4]	
4	1							

注意: 该方法的重定向不适用中断服务函数及中断服务函数中调用的函数。



7.4 重定向失败错误提示

1. 编写 ICF 文件后点击 Download and Debug 时,如发生图 7-5 所示情况,有可能是因为 ICF 配置错误导致重定向失败,需要检查 ICF 文件。

图 7-5: IAR	错误弹框窗口
------------	--------

ne ріоск немч with alignment = 8, size = _llftuli_size .tialize by copy { readwrite, readonly object add_test.o,	:_neap { };	
<pre>not initialize { section .noinit }; cc at address mem:_ICFEDIT_intvc_start_ { readonly section cc in ROM_region { readonly,</pre>	IarldePm × There were warnings while generating flash loader input. See the Debug Log window for details. OK	
ice in RAM_region { readwrite, block CSTACK, block HEAP,		
'ine region kwwi_region = mem:[from UXIFF8000 to UXIFFFFF; ice in RAM2_region { readonly object add_test.o,		

2. 如果点击 Download and Debug 没有产生错误弹框,重定向还是失败时,可以检查 Project.map 文件如果如图 7-6,需要重新检查 ICF 文件。

图 7-6: IAR 错误信息

spc2168.icf sub_test.c add_test.c main.c Project.map 🗙	
The basic heap was selected becauseadvanced_heap was not specified and the application did not appear to be primarily optimized for speed.	
**** PLACEMENT SUMMARY ***	
<pre>"A0": place at address 0x1000'0000 { ro section .intvec }; "P1": place in [from 0x1000'0000 to 0x1007'ffff] { ro }; define block CSTACK with size = 1K, alignment = 8 { }; define block HEAP with size = 512, alignment = 8 { }; "P2": place in [from 0x2000'0000 to 0x2000'3fff] { rw, block CSTACK, block HEAP, ro object add_test.o section .rodata, object sub_test.o section .rodata }; "P3": place in [from 0x1fff'8000 to 0x1fff'ffff] { ro object add_test.o section .text, object sub_test.o section .text }; initialize by copy { rw, ro object add_test.o section .text, ro object add_test.o section .text, ro object sub_test.o section .rodata };</pre>	
No sections matched the following patterns: ro object add_test.o section .rodata in "P2" ro object add_test.o section .text in "P3"	



3. 在编译程序时,如出现产生图 7-7 所示的 IAR 警告,可能是由于中断服务函数调用通过 ICF 文件配置的重定向函数导致 IAR Link 失败,则需要将中断服务函数调用在 ICF 重定向 的函数前加上"__ramfunc"关键字,如图 7-8 所示。

图 7-7: IAR ICF 文件警告信息

/*	IcfEditorFile="\$TOOLKIT DIR\$\confiq\ide\IcfEditor\cortex v1 0.xml" */
/*-	-Specials-*/
det	<pre>ine symbolICFEDIT_intvec_start_ = 0x10000000;</pre>
/*-	Memory Regions-//
def	The symbolICFEDI_IcgIon_KK_State = 0x10070FFF:
def	fine symbol ICFEDIT region RAM start = 0x20000000;
def	fine symbolICFEDIT_region_RAM_end = 0x20003FFF;
/*-	Sizes-*/
det	ine symbolLCFEDH_Size_cstack_ = 0x1000; fine symbol _LCFEDH_Size_heap = 0x1000;
/**	*** End of ICF editor section. ###ICF###*/
det	fine memory mem with size = 4G.
def	The region ROM region = mem:[from ICFEDIT region ROM start to ICFEDIT region ROM end];
def	fine region RAM_region = mem:[from _ICFEDIT_region_RAM_start_ to _ICFEDIT_region_RAM_end_];
2.04	
det	ine block CSLACK with alignment = 6, size = _llepul_size_cstack_ { };
ini	tialize by copy { readwrite ,
	section .text object SVFMM 1.0,
	section .text object Svrwm_2.0 ,
	};
4.0	
ao	not initialize { section .noint;;
pla	ace at address mem:ICFEDIT_intvec_start_ { readonly section .intvec };
	the in DOM merican (merican).
pla	the in Rom_region { readonity }; toe in Rom region { readonite.
	block CSTACK, block HEAP,
	section .text object SVEWM_1.0 ,
	section .text object SVPWM_2.0 ,
	v vala kasteri (vala)
	visa 2001, J26kmSter visa60
	Type_intsite_io;
Project	
Massace	
system_sp uart c	2월 20 10 10 10 10 10 10 10 10 10 10 10 10 10
timer.c wdt.c	
Warning[L symbols (offil) glocover incluios am of sections with origination (particular section section section SPAM), La syndoxis (EVPMA), relations, 17) and sections whom control (particular section, data in syntam, spc21181.0 D) (
Warring[L [SVPWM	(AID) generation as mic whether sensor (paramptic hrowing sensor), and an intermatic paramptic hrowing sensor (paramptic hrowing sensor), and an intermatic paramptic hrowing sensor (paramptic hrowing sensor), and an intermatic parameters (paramptic hrowing sensor), and an intermatic parameters (parameters hrowing sensor), and and an intermatic parameters (parameters hrowing sensor), and an intermatic parameters (parameters hrowing sensor), and an intermatic parameters (parameters hrowing sensor), and and an intermatic parameters (parameters hrowing sensor), and an intermatic parameters (parameters here here here here here here here h
Converting	
٤	

图 7-8: IAR 示例

main.c SVPWM_1.c x SVPWM_2.c spc2168.id
<pre>#include <spc2168.h> #include "SVERM_1.h"</spc2168.h></pre>
/* Model initialize function */ ramfunc void SVFWM_initialize_1(void)