		在 SAM7	s 上运行	FreeRTOS		
		C				
又档编号	MAN2003A_CH	$ \rightarrow $				
又档版本	Rev. A				-	
<u> </u>	以 SAM7S 为例,	说明 FreeRTOS	在 SAM7 上的	运行,升发环境,	是 Keil MDK	
关键词	SAM7S FreeRTOS	Keil MDK	1			
创建日期	2009-12-10	创建人员	Dracula	軍核人员	<u>Hotislandn</u>	
文档类型	公开发布/开发机	和套文件				
版权信息	Mcuzone 原创文	档,转载请注明	目出处			

更新历史

版本	时间	更新	作者
Rev. A	2009-12-10	初始创建	Dracula
微控电	日子 乐微电子		
杭州市	1 登云路 639 号 2B:	143	
田田	EL: +80-5/1-88908	3133 13770507006	
又付 I FΔX•••	EL: 10913989100 +86-571-88908193	13110201020	
<u>ww</u> w.r	ncuzone.com www	.atarm.com	

1.概述

<u>FreeRTOS</u>是一个免费的 <u>RTOS</u>。其体积很小,核心只有 3 个 c 文件,目前已经移植到了各种体系架构上,包括 ARM7。

使用 RTOS 可以简化软件系统的设计,分解系统中的各种任务,并且代码便于移植与重用。

使用 RTOS 会带来额外的性能开销,包括处理能力,ROM(FLASH), RAM 的占用。

ATMEL的 SAM75_MCU 基于 ARM7TDMI 架构,有足够的空间运行 RTOS,以简化软件设计。基于开源与易用的考虑,选择 FreeRTOS。

开发环境选择 Keil MDK, 界面直观, 且使用 ARM 官方的工具链, 性能有保证。

本站的 wiki 中有一篇讲解 FreeRTOS 移植的文章,请参考<u>《FreeRTOS 在 S32 上的移</u>标 FreeRTOS 的结构图如下:



2. 设置环境

2.1 安装 Keil MDK

下载 Keil MDK 并安装。

本文档涉及的项目工程基于 MDK 3.80a 创建。

磺ision3 Copyrig	3 V3.90 ht (c) Keil Elektronik GmbH / Keil Software, Inc. 1995 - 2009	
Toolchain: Toolchain Path: C Compiler: Assembler: Linker/Locator: Librarian: Hex Converter: CPU DLL: Dialog DLL: Target DLL: Dialog DLL:	RealView MDK-ARM Version: 3.80a BIN40\ Armcc.Exe V4.0.0.524 Armasm.Exe V4.0.0.524 ArmLink.Exe V4.0.0.524 ArmAr.Exe V4.0.0.524 FromElf.Exe V4.0.0.524 SARM.DLL V3.80a DARMATS.DLL V1.16 BINVAGDIRDI.DLL V1.08 TARMATS.DLL V1.13	S. C.

2.2 打开工程

下载项目文件 MAN2003_SAM7S_base_freertos.rar, 并展开:

RamDisk (J:) 🕨	
<u> 土具(1)</u> 帮助(<u>1</u>)	
共享 ▼ 刻录	新建文件夹
	311223311312
	A
治称	
MAN2002 SAM7	S base freetes
INAN2003_3AN	5_base_neentos
temp	
jii temp	
MAN2003 SAM7	/S hase freertos rar
- INA 2005_SAM	o_base_rreerresitar

使用 keil 打开工程 sam7s_base_freertos.Uv2:

🏠 🚅 🖬 🕼 👗 🛍 🛱 🗅 으		車 / 3 % % %
🥙 🕮 🝏 🚿 👗 🙀 💦 SAM7S_	_base_	freertos 🗾 🚽
Project Workspace 🔹 👻	085	#define TRACE
⊡ 🔁 SAM7S_base_freertos 🔹 🔺	086	#define TRACE
⊢ APP	087	#define TRACE
the main c	088	#define TRACE
	089	#define TRACE
	090	
	091	// By default
⊞… 🔛 board_lowlevel.c	092	fil idefine TRACE
⊞ 🗄 dbgu.c	033	tendif
	095	
— ───── 🖃 aic.c 🗉	096	// By default
The second secon	097	<pre>#if !defined()</pre>
	098	#define DYN_TI
	099	#endif
FreeRIOS	100	
⊕ 🗄 croutine.c	101	#if defined (NO
⊞… 🔛 list.c	102	ferror "Lrror
⊕ 🖈 queue.c	103	Tenuir
⊕ 🔝 tasks.c	105	#undef NOTRACN
⊡ — — OS Port	106	#if (TRACE LEV
	107	Adding NOTDA
	*	
\blacksquare		

2.3 配置工程

为了保证工程能正确编译,需要核对工程的相关设置。 首先检查芯片设置:

Options for Target 'SAM7S_base_freertos'				
Device Target Output Listing User C/C++ Asm Linker Debug Utilities				
Database: Generic CPU Data Base				
Vendor: Atmel				
Device: AT91SAM7S64				
Toolset: ARM				
AT91SAM7S256 AT91SAM7S32 AT91SAM7S321 AT91SAM7S321 AT91SAM7S512 AT91SAM7S526 AT91SAM7SE256 AT91SAM7SE512 AT91SAM7SE512 AT91SAM7SE512 AT91SAM7SE512 AT91SAM7X256 AT91SAM7X256 AT91SAM7X256 AT91SAM7XC128 AT91SAM7XC256 A				
AT91SAM9260				
OK Cancel Defaults Help				

然后是时钟及处理器模式,时钟设置为 18.432MHz,只是 ATEML 默认的时钟配置。使用 Thumb 来生成代码,这样处理特别声明的代码,都将编译为 Thumb,可以节省 flash 空间,并提高代码在 Flash 中运行的速度。

Options for Target 'SAM7S_base_freertos'	X					
Device Target Output Listing Vser C/C++ Asm Linker Debug Utilities						
Atmel AT91SAM7S64	- Code Generation					
<u>X</u> tal (MHz): 18.432	Thumb-Mode					
Operating system: None	✓ Use Cross-Module Optimization					
	🗖 Use MicroLIB 🔲 Big Endian					
	Use Link-Time Code Generation					
Read/Only Memory Areas	Read/Write Memory Areas					
default off-chip Start Size Startup	default off-chip Start Size NoInit					
□ ROM1: □ O	□ RAM1: □ □					
□ ROM2: ○	RAM2:					
□ ROM3: □ O	RAM3:					
on-chip	on-chip					
OK Car	cel Defaults Help					

配置输出为工程目录的 output 文件夹,可以避免在工程的根目录生成中间文件,使得工程比较整洁:

Options for Target 'SAM7S_	_base_freertos'	×
Device Target Output I	Listing Vser C/C++ Asm Linker Debug Vtilities	
Select Folder for Objec	ts <u>N</u> ame of Executable: sam7s_base_freertos	
	output\sam7s_base_freertos	
Debug Information	Crea	ate Batch File
Create HEX File		
Browse Information		
C Create Library: .\output	ut\sam7s_base_freertos.LIB	
	OK Cancel Defaults	Help

选择输出 map 文件,便于对代码的 memmap 的分析:

Options for Target 'SAM7S_base_	freertos'	×		
Device Target Output Listin	ng User C/C++ Asm Linker Debug Utilities			
Select Folder for Listings	Page Width: 79 + Page Length: 66 +			
Assembler Listing: .\output*	lst			
C Preprocessor Listing: .\output*	`.txt put*.i			
✓ Linker Listing: .\output\sam7s_base_freertos.map ✓ Memory Map ✓ Symbols ✓ Callgraph ✓ Cross Reference ✓ Unused Sections Info ✓ Veneers Info				
	OK Cancel Defaults	Help		

配置在编译完成后运行一个用户命令,使用 fromelf 工具输出 bin 文件,便于 <u>SAM-BA</u>烧写:

Options for Target 'SAM7S_	_base_freertos'	×
Device Target Output I	Listing Vser C/C++ Asm Linker Debug Vtiliti	es
Run User Programs Before (Compilation of a C/C++ File	
🗖 Run #1:		🗆 DOS16
🗖 Run #2:		🗖 DOS16
Run User Programs Before B	Build/Rebuild	
🗖 Run #1:		🗖 DOS16
🔲 Run #2:		🗆 DOS16
– Run User Programs After Bu	uild/Rebuild	
Run #1: Kam/bin	40/fromelf.exebinoutput #l.bin #l	🗆 DOS16
🗆 Run #2:		🗖 DOS16
☑ Beep When Complete	Start Debugging	
	OK Cancel Defaults	Help

在 C/C++选项卡中可以定义一些宏,控制优化,头文件位置及特殊的编译选项:

Options for Target 'SAM7S_base_freertos'		×
Device Target Output Listing User	C/C++ Asm Linker Debug	Vtilities
Preprocessor Symbols		
Define: at91sam7s64 PORT_INT_FUNC	TION NOFPUT	
U <u>n</u> define:		
Language / Code Generation		
Enable ARM/Thumb Interworking	Strict ANSI C	Wamings:
Optimization: Level 0 (-00)	Enum <u>C</u> ontainer always int	<unspecified></unspecified>
Optimize <u>f</u> or Time	Plain Char is Signed	Thum <u>b</u> Mode
Split Load and Store Multiple	Read-Only Position Independent	
One ELF Section per Function	<u>Read-Write Position Independent</u>	
Include Paths Misc Controls Compiler control string	S_Port;.\at91lib\peripherals;.\at91lib\boar e_freertos.fed'' -thumb -cdevice DARMA TOS -I.\include\OS_Port -I.\at91lib\periph	ds\at91sam7s-ek;.\at91lib \TS -g -O0 erals -l.\at91lib\boards
OK	Cancel Defaults	Help

ASM 选项卡与 C/C++的类似,但是是为控制汇编文件准备的:

Options for Target 'SAM7S_base_freertos'	x
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Conditional Assembly Control Symbols	
Define:	
Undefine:	
Language / Code Generation	
Read-Only Position Independent Read-Write Position Independent	
□ <u>T</u> humb Mode	
No Wamings	
Include	
Misc	- 1
Assembler control string	A T
OK Cancel Defaults	Help

Linker 选项卡控制代码的连接,比较重要。本工程中需要比较复杂的 memmap(部分代码位于 RAM),所以使用 scatter loader file 来控制:

Options for Targ	rget 'SAM7S_base_freertos'	×
Device Targe	get Output Listing Vser C/C++ Asm Linker Debug Vtilities	
☐ <u>U</u> se Memo ☐ Ma <u>k</u> e F ☐ M <u>a</u> ke F ☐ Do <u>n</u> t S ☞ Report	iory Layout from Target Dialog RW Sections Position Independent RO Sections Position Independent RO Sections Position Independent R/W Base Search Standard Libraries Int 'might fail' Conditions as Errors	
Scatter File	Noutput\sam7s_base_freertos.sct	dit
Misc controls		* *
Linker control string	device DARMATS *.o -feedback ".\output\sam7s_base_freertos.fed" strictscatter ".\output\sam7s_base_freertos.sct"	r T
	OK Cancel Defaults	Help

Debug 选项卡中配置使用软件仿真:

Options for Target 'SAM7S_base_freertos'	×
Device Target Output Listing User C/C++ A	sm Linker Debug Utilities
 ✓ Use Simulator ✓ Limit Speed to Real-Time 	© <u>U</u> se: RDI Interface Driver
✓ Load Application at Startup Run to main() Initialization File: NROMstart.ini Restore Debug Session Settings ✓ Breakpoints ✓ Toolbox ✓ Watchpoints & PA ✓ Memory Display CPU DLL: Parameter: SARM.DLL -cAT91SAM7S Dialog DLL: Parameter: DARMATS.DLL -p91SAM7S64	✓ Load Application at Startup Run to main() Initialization File: Restore Debug Session Settings ✓ Breakpoints ✓ Toolbox ✓ Watchpoints ✓ Memory Display Driver DLL: Parameter: SARM.DLL Dialog DLL: Parameter: TARMATS.DLL _p91SAM7S64
OKCan	cel Defaults Help
金查无误后,可以选择菜单 clean 工程:	ndow. Help
New µ <u>V</u> ision Project New Project <u>W</u> orkspace Import µVision1 Project Open Project Close Project	
Manage <u>S</u> elect Device for Target 'SAM7S_base_freertos' Re <u>m</u> ove Item Options for Target 'SAM7S_base_freertos'	•
Clean <u>t</u> arget	

然后重新编译工程:

2	<u>F</u> ile	<u>E</u> dit	<u>V</u> iev	v <u>P</u> r	oject	<u>D</u> ebi	ug F	l <u>a</u> sh	Pe <u>r</u> i	pheral	s <u>T</u> o	ols 🖁						
×) 🖻	H	Ø	X	Ēð	C	2	<u>e</u>	ŧ.	∉ ∧	%	%						
\$			۲	X	LOAD	<u> </u>	AM7	'S_b	ase_	freert	DS	-						
roj	ect W	Build AM7	targe S_ba	et se_fre	erto	5	- X		085 086	#def #def	ine ine	TRA TRA						
无	误的	话将生	主成	axf ⊅	て件及	έ bin 🕽	文件:	:										
:	\outpu \outpu	it\sam it\sam	17s_ba 17s_ba	ase_fi ase_fi	reerto	s.sct	(36): (37):	warn warn	ing: ing:	L6314W: L6314W: L6329W.	No s No s	ection ection	matches matches	pattern pattern	rtt.o(adc.o((RO). (RO).	unused	eections

注意编译后的 memmap:

Ė

Ė

Program Size: Code=7680 RO-data=124 RW-data=76 ZI-data=9320
User command #1: D:\ Dsetup\keil\arm/bin40/fromelf.exe --bin -".\output\sam7s_base_freertos.axf" - 0 Error(s), 8 Warning(s).

⊞ itasks.c							
	Execution Re	gion RAM_VECT	(Base:	0x0020000	0, Size:	0x00000c0, Max: 0x0	0000300, ABSOLUTE)
🗄 🔛 heap_1.c	Deen John	R 4	T	3.5.5.1	Toles T	Contine News	
🗄 🗄 port.c	base Addr	5126	туре	AUUT	Iax P	Section Name	Object
⊕ 🔄 portISR.c	0x00200000	0x00000c0	Code	RO	350	RAM VECTOR	ram vect.o
portasm.s						—	—
	Des set i se De		(D	0	0 01	0	
🖳 🗄 Cstartup_rv.S	Execution Re	gion RAM_CODE	(Base:	0X002000C	0, Size:	UXUUUUU128, Max: UXI	TITITI, ABSOLUTE)
ram_vect.S	Base Addr	Size	Type	Attr	Idx E	Section Name	Object
am7s base freertos.sct	0x002000c0	0x0000004	Ven	RO	343	FreeRTOS_PORT_ASM	portasm.o
sam7s base freertos ma	0x002000c4	0x00000124	Code	RO	343	FreeRTOS_PORT_ASM	portasm.o
KOIVIstart.ini							

Execution Region ER_data_ram (Base: 0x002001e8, Size: 0x000020b4, Max: 0xffffffff, ABSOLUTE)

portasm.s 中的代码在 RAM 中运行。

3. 模拟仿真

3.1 简单仿真

点击 debug 按钮:

🛤 🕂 🔶 🖭 🔶 🗮 🕅	
Start/Sto	op Debug Session
进入 debug 模式: ————————————————————————————————————	ARLA SIARI, CODE, READONLI ARM ENTRY RESET LDR PC, ARM_REST_VECT ; Reset B UDFHANDLER ; UNDEFINED B SWIHANDLER ; SWI B PABTHANDLER ; PREFETCH ABORT B DABTHANDLER ; DATA ABORT
⊡ SPSR 0x00000000 067 ⊡ User/System 068 ⊡ Fast Interrupt 069 □ Interrupt 070	B . ; RESERVED LDR PC, [PC,#-0xF20] ; Vector From AIC_IVR LDR PC, _CPU_FIQ_ISR ; FIQ
黄色箭头就是 PC 指针,也就是当前程序 由于代码部分运行于 RAM,所以首先要 Debug Flash Peripherals Tools SVCS @ Start/Stop Debug Session Ctrl+	P运行的位置。 修改 Keil 默认的 mem 权限: <u>W</u> indo F5
Debug Settings	
Enable/Disable Trace Recording	
★ View Trace Records Ctrl+T Execution Profiling Setup Logic Analyzer	r •
Memory Map	
Per <u>f</u> ormance Analyzer	

首先删除内部 RAM:

Memory Map	×
Current Mapped	
000: 0x00000000 - 0x0000432F exec read write 001: 0x00004330 - 0x0000FFFF exec read 002: 0x00100000 - 0x0010432F exec read write 003: 0x00104330 - 0x0010FFFF exec read 004: 0x00200000 - 0x002001E7 exec read	
005: 0x002001E8 - 0x00203FFF read write 006: 0xFFFA0000 - 0xFFFA3FFF read write	
<u>Kill Selected Range</u> Map Range - Example: 0x40000000, 0x4000FFFF	
, □ <u>R</u> ead □ <u>W</u> rite	
Execute Map Range	
<u>Q</u> lose He	
为 RAM 新建一个属性,支持所有操作:	N.
Memory Map	
Current Mapped	
000: 0x00000000 - 0x0000432F exec read write 001: 0x00004330 - 0x0000FFFF exec read 002: 0x00100000 - 0x0010432F exec read write 003: 0x00104330 - 0x0010FFFF exec read 004: 0x00200000 - 0x0203FFFF exec read write	
005: 0xFFFA0000 - 0xFFFA3FFF read write 006: 0xFFFB8000 - 0xFFFBBFFF read write	-
<u>Kill</u> Selected Range	
Map Range - Example: 0x40000000, 0x4000FFFF	
Dx00200000 .0x00203FFFF	
I Read	
IV <u>vv</u> nte IV <u>Execute</u> Map Range	
<u>Q</u> ose He	elp

选择 Map Range 新建新的 memmap。

打开文件 main.c,并设置断点:

while(1) 24 25 Ł 26 LED_Set(0); vTaskDelay(250); 27 LED Set(1); 28 vTaskDelay(250); 29 30 LED Set(2); vTaskDelay(250); 31 32 33 LED Clear(0); LED Clear(1); 34 35 LED Clear(2); vTaskDelay(248); 36 37 } 38

打开外设窗口的 PIO 窗口:



模拟的 PIO 出现:

PIOA: Parallel I/O Controller A				
PIO / Output / Input Filter / Output Data / Multi Driver / Pull-up / AB Select / Output Write				
PIOA_OWSR: 0x00000000				
Pin Data Status				
I/O Pins 1/O Pins 10 15 0 7 0 7				
Pins: 100000080 31 Bits 24 23 Bits 16 15 Bits 8 7 Bits 0				
Interrupt Mask & Status				
PIOA_IMR: 0x00000000 31 Bits 24 23 Bits 16 15 Bits 8 7 Bits 0				
点击运行按钮:				
Eile Edit View Project Debug Fl				
🛚 👫 🔄 🔁 🔂 🔂 🖓 🤹				
Project Wd Run ce				
Register Value				
代码运行到第一个断点处:				
61 int main (void)				
62⊟ {				
63 /* Setup the ports. */				
65				
66 /* LED task */				
68				

再次点击运行,停在 LED 任务中:

16 vo	id vLEDTask(void *pvPar	ameters)
18	pvParameters = pvParam	eters;
20	LED_Configure(0);	PIOA: Parallel I/O Controller A
21	LED_Configure(1); LED_Configure(2);	PIO / Output / Input Filter / Output Data / Multi Driver / Pull-up / AB Select / Output Write
23 24	while(1)	
25	{ LED Set(0):	
27	vTaskDelay(250);	
28 29	vTaskDelay(250);	
30 31	LED_Set(2); vTaskDelay(250);	
32	IED Clear(0):	
34	LED_Clear(1);	- Pio Data Statue
35 36 37	LED_Clear(2); vTaskDelay(248);	PIOA_PDSR: 0x09000087 31 Bits 24 23 Bits 16 15 Bits 8 7 Bits 0
注意此	时 LED 对应的 PIO 状态	
/// L	ED #0 pin definition	(PAO).
#defi /// T	ne PIN_LED_DS1 {1 · FD #1 pin_definition	<< 0, AT91C_BASE_PIOA, AT91C_ID_PIOA, PIO_OUTPUT_1, PIO_DEFAULT}
#defi	ne PIN_LED_DS2 {1	<< 1, AT91C_BASE_PIOA, AT91C_ID_PIOA, PIO_OUTPUT_1, PIO_DEFAULT}
/// L	ED #2 pin definition	(PA2).
#dell	ED #3 pin definition	(PA3).
#defi	ne PIN_LED_DS4 {1	<< 3, AT91C_BASE_PIOA, AT91C_ID_PIOA, PIO_OUTPUT_1, PIO_DEFAULT}
#defi	ne PINS LEDS PIN L	ED DS1, PIN LED DS2, PIN LED DS3, PIN LED DS4
按下 F	10,运行过 LED,可以看	f到 PIOA0 的状态改变:
19	LED Configure(0);	PIOA: Parallel I/O Controller A
21	LED_Configure(1);	PIO / Output / Input Filter / Output Data / Multi Driver / Pull-up / AB Select / Output Write
23	htp_configure(2);	
24 25	while(1) {	
26	LED_Set(0); vTaskDelav(250);	
28	LED_Set(1);	
29	vTaskDelay(250); LED_Set(2);	
31	vTaskDelay(250);	
33	LED_Clear(0);	
34 35	LED_Clear(1); LED_Clear(2);	Pin Data Status
36 37	<pre>vTaskDelay(248); }</pre>	
38 }		Pins: 0x09000086 31 Bits 24 23 Bits 16 15 Bits 8 7 Bits 0
39 -		

Template A.1

PIOAO 变成了低电平。再次运行可以看到对应引脚的变化:



MAN2003A

FreeRTOS 中有 PIT 中断,为了调试中断,可以在 portsam.S 中的 IRQ 异常服务程序处打断点:



PIT 状态为:

FII (八述/以:	
PIT: Periodic Interval Timer	
Mode	
PIT_MR: 0x03000883 PIV: 0x000883	PITEN PITIEN
Status	
PIT_SR: 0x00000001	
Periodic Interval Value	
PIT_PIVR: 0x00100000 CPIV: 0x000000 PICNT:	0x0001
□ Periodic Interval Image	
PIT_PIIR: 0x00100000 CPIV: 0x000000 PICNT:	0x0001
DITC 已经罢位	
中国 山红直世。 按 [11 可以单步跟踪代码的运行 法音奏方界和标	
按 FII 可以毕少戚际们将的运行, 注息可行命种位 大下面的位置打入断占 并入违行行到断占位置	的文化,结合代码工的注种型牌 RIUS IRQ 运行的过程。
在下面的位直打了呦点,开至迷运打到呦点位直:	
	MOV LR, PC
	BX RO
221 228 : 13	. Find next ready task
	SR may wake some high priority tasks
	LDR RO, =vTaskSwitchContext ; Sel
	MOV LR, PC
232 233	
按卜F11,程序跳转到具止的C语言与的对应的S	R:
OS_Port\portISR.c]	And the second sec
Fl <u>a</u> sh Pe <u>r</u> ipherals <u>T</u> ools <u>S</u> VCS <u>W</u> indow <u>H</u> elp	
2 🕮 連 🦽 % % % 🐃	2) + + M M
2 of R 🔊 🌱 2 🗉 🗄 🚾 强 🕻	J 🥕
× 111 }	
112 #else /* configUSE_PREEMPTIC	N == 1 * /
113 Void vPortPreemptivelick (V	01d)
115 volatile unsigned portLC	NG ulDummy;
116 /* Increment the tick co	unt - this may make a de
117 to run - but a context s	witch is not performed.
119	
120 /* Clear the PIT interru	pt. */
121 ulDummy = AT91C_BASE_PIT	C->PITC_PIVR;
	ON */
123 #enuli /* configure_PREEMPII	

3.3 调试任务

任务(task)可以任务是 RTOS 的一个独立执行单元,一个任务(task)认为它自己独享了系统资源。但实际的任务被调度器调度运行。

复位处理器,并清除前面的断点。

任务的创建可以跟踪如下代码来了解。

在 port.c 的如下位置打个断点:

P → a sicc 118 -*/ P → a pio.c 119 portSTACK_TYH P → a pio.c 120 - (P → a portSTACK_TYH 120 - (P → a portSTACK_TYH 120 - (P → a portSTACK_TYH 121 portSTACK_TYH P → a portSTACK_TYH 122 /* Setup P → a portSTACK_TYH 122 /* Setup P → a portSTACK_TYH 124 expected P → a portSTACK_TYH 125 px0crigins P → a port 128 px0rigins P → b portc 129 px0rigins 131 start of 131 start of	<pre>PE *pxPortInitialiseStack(portSTACK_TYPE *pxTopOfStack, pdTASK_CODE pxCode, void *pvParameters) PE *pxOriginalTOS; the initial stack of the task. The stack is set exactly as by the portRESTORE_CONTEXT() macro. where the top of the (simulated) stack is before we place on it. */ alTOS = pxTopOfStack; on the stack is the return address - which in this case is the the task. The offset is added to make the return address appear T bl == Table = 24 bb th </pre>
全球执行判断点位直,从受重卤口口 × Name ・ pxTopOfStack ・ pxCode ・ pvParameters ・ pxOriginalTOS	リ以看到当前的值: Value 0x002004EC 0x001001A9 0x00000000 0x00100FB0
Y 打开 mem 窗口,并否看此位置,	
Address: 0x00200480 0x00200480: A5A5A5A5 A5A5A5 0x00200490: A5A5A5A5 A5A5A5	A5 A5A5A5A5 A5A5A5A5 A5 A5A5A5A5 A5A5A5A5
0x002004A0: A5A5A5A5 A5A5A5 0x002004B0: A5A5A5A5 A5A5A5 0x002004C0: A5A5A5A5 A5A5A5 0x002004D0: A5A5A5A5 A5A5A5	A5 A5A5A5A5 A5A5A5A5 A5 A5A5A5A5 A5A5A5A5
0x002004E0: A5A5A5A5 A5A5A5	A5 A5A5A5A5 A5A5A5

按 F11 单步执行到函数结束,可以看到系统构造的任务栈及新的栈位置:



按 F11 单步跟入,系统进入启动第一个任务的代码: VAN2003_SAM7S_base_freertos\OS_Port\portasm.s] ash Peripherals Tools SVCS Window Help 連連 な % % % 骗 **▼**|| 繜 ヅ 🛃 🗉 🔚 🚟 🚾 🗞 🗔 🗡 達 ()金 R 闷 063 064 :/ 065 vPortStartFirstTask ; Start the first ready task 066 ; 067 ;/*-068 portRESTORE CONTEXT 069 vPortStartFirstTask 070 RUN NEXT TASK 071 ; make sure this function is called in SV(; 1. Set the LR to the task stack 072 073 LDR R0, =pxCurrentTCB R0, [R0] 074 LDR LR, [RO] ; First Item is pxTop 075 LDR 076 The critical nesting depth is the fi 077 078 ; 2. Load it into the ulCriticalNesting va R0, =ulCriticalNesting 079 LDR 080 LDMFD LR!, {R1} R1, [R0] 081 STR 082 083 ; 3. Get the SPSR from the stack. 084 LDMFD LR!, {R0} MSR SPSR cxsf, R0 085 单步运行到如下位置: R3 0x00100519 vPortStartFirstTask 069 0x00000001 · R4 RUN NEXT TASK 070 0x0020223c · R5 071 ; make sure this function is called in SVC mode · R6 0x00000000 ; 1. Set the LR to the task stack 072 R7 LDR R0, =pxCurrentTCB - R8 073 R9 074 LDR R0, [R0] R10 LR, [R0] ; First Item is pxTopOfStack 075 LDR R11 076 R12 0x00100733 077 The critical nesting depth is the first item on the stack. R13 (SP) 0x00203fc8 ; 2. Load it into the ulCriticalNesting variable. 078 R14 (LR) R15 (PC) 0x002004a8 0x002000d0 LDR RO, =ulCriticalNesting 079 ±---- CPSR LDMFD LR!, {R1} 0x400000d3 080 ±---- SPSR 0x00000000 081 STR R1, [R0]

注意此时的处理器模式为 SVC,而 LR 的值,正是前面构建的任务的 stack 值。

单步运行到函数末,可以看到寄存器的恢复情况:



此时的处理模式为 system,所以的寄存器均为任务创建时的状态。