STR730X 在 IAR 环境中的存储代码分析

这是本人花一天的时间分析所画的,分析的有些乱,不为别的,主要是为了整理自己的思路。同时也希望对这部分内容有疑惑的朋友有所帮助,本人也是个初学者。此文难免有许多的不足和错误.望各位高手指教!! konglong_722@163.com

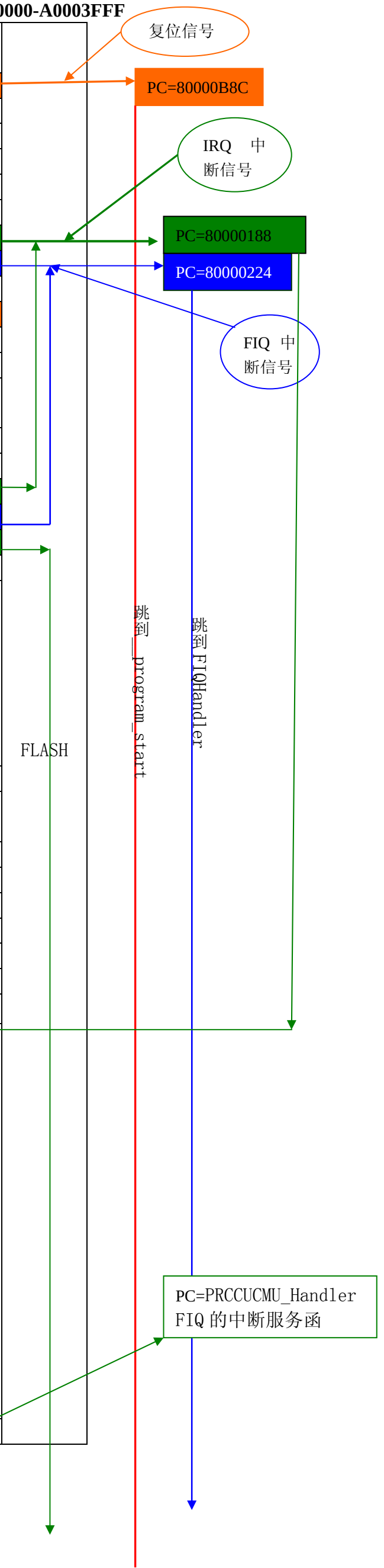
注: 1 表中"[ ]"中的数据表示为地址," …… "表示有后续的指令,表中的数据全部为 16 进制的,CODE 表示为指令,DATA 表示数据.

2 此表示根据 STR730X 控制器,经过反汇编而得出的,此器件的 ROM=80000000-8003FFFF, RAM=A0000000-A0003FFF

地址	存储内容	数据属性	汇编语言	执行结果
0x80000000	e59ff018	CODE	LDR PC,Reset_Addr	PC=[80000020]
0x80000004	e59ff018	CODE	LDR PC,Undefined_Addr	PC=[80000024]
0x80000008	e59ff018	CODE	LDR PC,SWI_Addr	PC=[80000028]
0x8000000C	e59ff018	CODE	LDR PC,Prefetch_Addr	PC=[8000002C]
0x80000010	e59ff018	CODE	LDR PC,Abort_Addr`	PC=[80000030]
0x80000014	e1a00000	CODE	NOP	PC=[80000034]
0x80000018	e59ff018	CODE	LDR PC,IRQ_Addr	PC=[80000038]
0x8000001C	e59ff018	CODE	LDR PC,FIQ_Addr	PC=[8000003C]
0x80000020	80000b8c	DATA	Reset_Addr DCD program_start	[80000020]=80000b8c
0x80000024	80000140	DATA	Undefined_Addr DCD UndefinedHandler	[80000024]=80000140
0x80000028	80000164	DATA	SWI_Addr DCD SWIHandler	[80000028]=80000164
0x8000002C	800001d4	DATA	Prefetch_Addr DCD PrefetchAbortHandler	[8000002C]=800001d4
0x80000030	800001fc	DATA	Abort_Addr DCD DataAbortHandler	[80000030]=800001fc
0x80000034	0	DATA	0	[80000034]=0
0x80000038	80000188	DATA	IRQ_Addr IRQHandler	[80000038]=80000188
0x8000003c	80000224	DATA	FIQ_Addr FIQHandler	[8000003C]=80000224
0x80000040	8000024c	DATA	PRCCUCMU_Addr DCD PRCCUCMUIRQHandler	[80000040]=8000024c
0x80000044	8000026c	DATA	EXTIT01_Addr DCD EXTIT01IRQHandler	[80000044]=8000026c
……	……	……	通过 IRQHandler 的执行,PC 会自动指向所对应的中断向量,假设 PRCCUCMU 产生了中断.则 PC=[80000040]=8000024C	
0x8000013C	8000096C	DATA	RTC_Addr DCD RTCIRQHandler	[8000013C]=8000096C
	e92d5fff	CODE	UndefinedHandler	Undefined 中断装配程序
0x80000140		CODE	SaveContext r0,r12	宏函数
0x8000014C	e59f0838	CODE	ldr r0, =Undefined_Handler	R0=中断服务程序地址
0x80000150	e59fe838	CODE	ldr lr,=Undefined_Handler_end	
0x80000154	e12fff10	CODE	bx r0	PC=中断服务程序地址
		CODE	Undefined_Handler_end:	
0x80000158		CODE	RestoreContext r0,r12	宏函数
0x80000164	e92d5fff	CODE	SWIHandler ……	和 Undefined 功能一样

0x80000188	……	CODE	IRQHandler
			SaveContext r0,r12
			LDR lr, =ReturnAddress;
			LDR r0, =EIC_base_addr
			LDR r1, =IVR_off_addr
			ADD pc,r0,r1
			ReturnAddress
			LDR r0, =EIC_base_addr
			LDR r2, [r0, #CICR_off_addr]
			CMP r2,#31
			SUBHI r2, r2, #32
			MOV r3,#1
			MOV r3,r3,LSL r2
			STRHI r3,[r0,#IPR1_off_addr]
			STRLS r3,[r0, #IPR0_off_addr]
			RestoreContext r0,r12
				执行 PC=IVR 处的指令
				=>此处的结果是根据 IPRn 中的内容把 XXX_Addr 里的数据赋给 PC,即:
				PC=[XXX_Addr].这样程序就跳到后面的中断装配程序了.`

800001d4	……	CODE	PrefetchAbortHandler ……	Prefetch 中断装配程序



800001fc	……	CODE	DataAbortHandler ……	DataAbort 中断装配程序	
80000224	……	CODE	FIQHandler ……	FIQ 中断装配程序	
8000024c	……	CODE	PRCCUCMUIRQHandler ……	PRCCUCMU 中断装配程序	
……	……	……	……	XXXX 中断装配程序	
0x8000096c	……	CODE	RTCIRQHandler……	RTC 中断装配程序	
0x80000988: 向量地址结束地址					
0X8000098C	XXXXXXXX				
……					
0x80000B8B	vect.s 文件最后的地址				
0x80000B8C	e59ff144	CODE	__program_start	复位程序开始	
	……	CODE	……	初始化 stack; FLASH_RAM_REMAP; EIC 初始化,全部关闭; IPRn=XXX`_Addr;	
0X80000C94	ea000037	CODE	B ?main	跳到 MAIN 函数	
……					
0x80000DE3	init.s 的最后的地址				
0x80000DE4	mian.c 的开始地址				
……					
0x8000294B	main.c 的最后地址				
0x8000294C - 0x80002963: 保存 INITTAB(分别指示 DATA_I 和 DATA_Z 等数据的信息(SIZE,ADDREES))					
0x80002964: DATA_ID(FLASH 中的数据,运行是要拷到 RAM 中的)的开始地址					
0x8003FFFF: FLASH_END					

保留

A0000000 - A0000000:存放 DATA_I 数据					
A0000000 - A00000A7:存放 DATA_Z 数据					
CSTACK	A00000A8 - A0000287	1E0	rel	2	RAM
SVC_STACK	A0000288 - A00002A7	20	rel	2	
UND_STACK	A00002A8		rel	2	
ABT_STACK	A00002A8		rel	2	
FIQ_STACK	A00002A8 - A00002E7	40	rel	2	
IRQ_STACK	A00002E8 - A00003E7	100	rel	2	
0xA0003FFF: RAMEND					

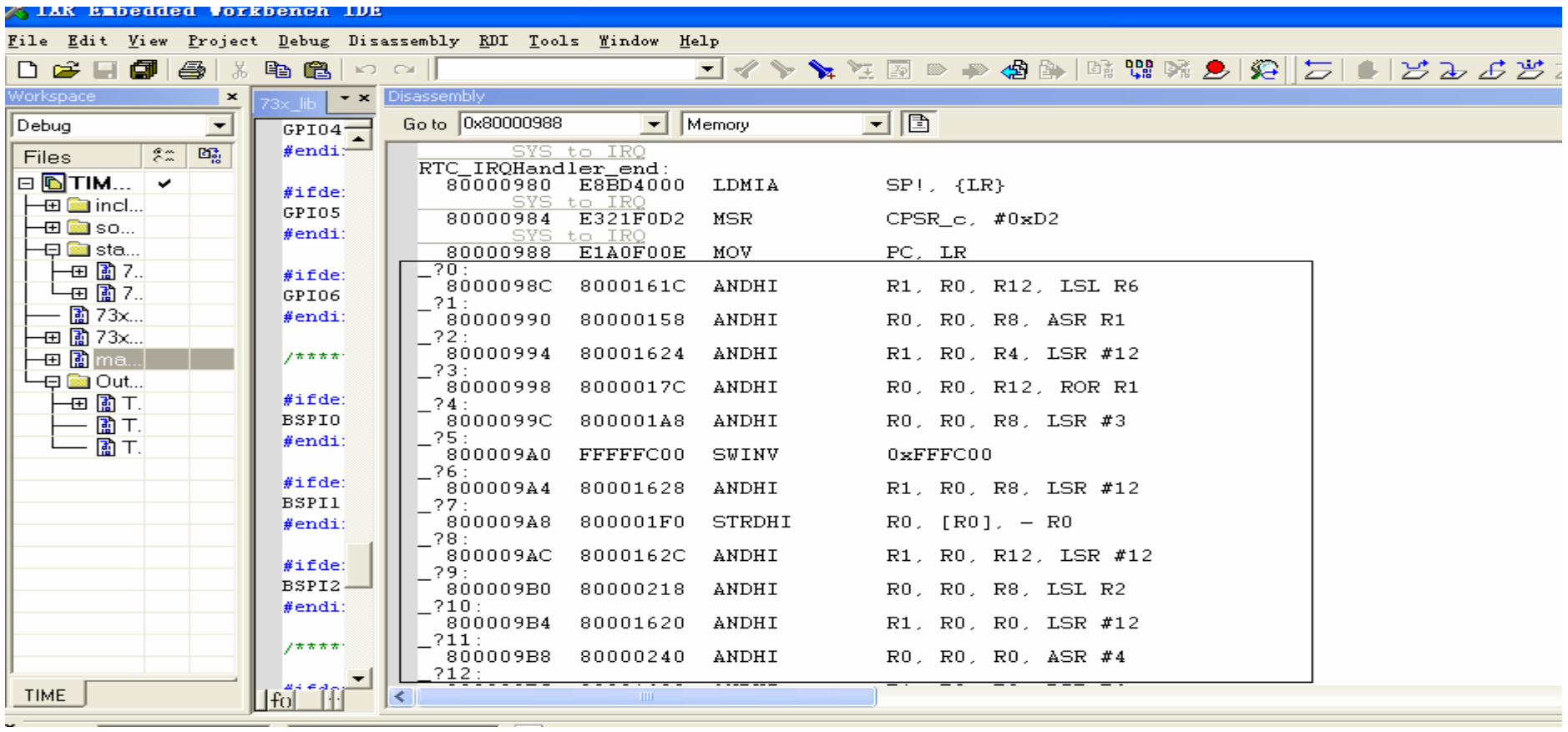
注:

:复位路线

FIQ 执行路线

IRQ 执行路线`

在输出的可执行文件中发现,存放在 FLASH 中的代码并不示连续存放的，在 vect.s，init.s 和 main.c 之间都有一段未知代码,不知道是不是编译器添加的有用代码,还是一些无用的随机数。


IAR Embedded Workbench IDE
File Edit View Project Debug Disassembly RDI Tools Window Help
Workspace
Debug
Files
TIM...
incl...
so...
sta...
73x...
ma...
Out...
TIME
73x_lib
GPIO4
GPIO5
GPIO6
BSPI0
BSPI1
BSPI2
Disassembly
Go to 0x80000988
Memory
SYS to IRQ
RTC_IRQHandler_end:
80000980 E8BD4000 LDMIA SP!, {LR}
SYS to IRQ
80000984 E321F0D2 MSR CPSR_c, #0xD2
SYS to IRQ
80000988 E1A0F00E MOV PC, LR
_?0:
8000098C 8000161C ANDHI R1, R0, R12, LSL R6
_?1:
80000990 80000158 ANDHI R0, R0, R8, ASR R1
_?2:
80000994 80001624 ANDHI R1, R0, R4, LSR #12
_?3:
80000998 8000017C ANDHI R0, R0, R12, ROR R1
_?4:
8000099C 800001A8 ANDHI R0, R0, R8, LSR #3
_?5:
800009A0 FFFFFC00 SWINV 0xFFFC00
_?6:
800009A4 80001628 ANDHI R1, R0, R8, LSR #12
_?7:
800009A8 800001F0 STRDHI R0, [R0], - R0
_?8:
800009AC 8000162C ANDHI R1, R0, R12, LSR #12
_?9:
800009B0 80000218 ANDHI R0, R0, R8, LSL R2
_?10:
800009B4 80001620 ANDHI R1, R0, R0, LSR #12
_?11:
800009B8 80000240 ANDHI R0, R0, R0, ASR #4
_?12: