# 专题5-核心初始化

一、异常向量表

1.1、异常定义

异常由内部和外部来源产生，以使处理器处理事件，如外部产生的中断或尝试执行未定义的指令。 在处理异常之前的处理器状态通常被保留，以便在异常例程完成时可以恢复原始程序。 同时可以出现不止一个异常。

ARM架构支持七种类型的异常。 表A2-4列出了用于处理每种类型的异常类型和处理器模式。 当发生异常时，强制执行从与异常类型相对应的固定内存地址。 这些固定地址称为异常向量。

**Table A2-4 Exception processing modes**

| Exception type | Mode | VE[a] | Normal address | High vector address |
|---|---|---|---|---|
| Reset | Supervisor | | 0x00000000 | 0xFFFF0000 |
| Undefined instructions | Undefined | | 0x00000004 | 0xFFFF0004 |
| Software interrupt (SWI) | Supervisor | | 0x00000008 | 0xFFFF0008 |
| Prefetch Abort (instruction fetch memory abort) | Abort | | 0x0000000C | 0xFFFF000C |
| Data Abort (data access memory abort) | Abort | | 0x00000010 | 0xFFFF0010 |
| IRQ (interrupt) | IRQ | 0 | 0x00000018 | 0xFFFF0018 |
| | | 1 | IMPLEMENTATION DEFINED | |
| FIQ (fast interrupt) | FIQ | 0 | 0x0000001C | 0xFFFF001C |
| | | 1 | IMPLEMENTATION DEFINED | |

a. VE = vectored interrupt enable (CP15 control); RAZ when not implemented.

1.2、代码编写

1.2.1、start.S

```
4       b reset
5       ldr pc, _undifined_instrution
6
7
8 _undifined_instrution: .word undifined_intruction
9
10
11
12 undifined_intruction:
13       nop
```

使用一个内存存储单元异常处理程序的地址，使用LDR装载而不是伪指令来跳转到异常处理程序。

1.2.2、gboot.lds

```
1 OUTPUT_ARCH(arm)
2 ENTRY(_start)
3 SECTIONS{
4       . = 0x50008000;
5
6       . = ALIGN(4);
7       .text :
8       {
9              start.o(.text)
10              *(.text)
11       }
12
13       . = ALIGN(4);
14       .data :
```

```
15      {
16              *(.data)
17      }
18
19      . = ALIGN(4);
20      bss_start = .;
21      .bss :
22      {
23              *(.bss)
24      }
25      bss_end = .;
26
27 }
```

1.2.3、Makefile

```
1 all : start.o
2       arm-linux-ld -Tgboot.lds -o gboot.elf $^
3       arm-linux-objcopy -O binary gboot.elf gboot.bin
4
5 %.o : %.S
6       arm-linux-gcc -g -c $^
7
8 $.o : %.c
9       arm-linux-gcc -g -c $^
```

1.2.4、210头文件的添加

V210会对BL1进行校验（根据头文件里的BL1大小信息和实际信息进行校验）

```
./mkv210_image led.bin 210.bin
```

二、设置SVC32模式

通过汇编操作CPSR改变工作模式。在设置SVC32模式的时候也禁止中断和快速中断。

```
42 reset:
43      bl set_svc
44
45 set_svc:
46      mrs r0, cpsr
47      bic r0, r0, #0x1f
48      orr r0, r0, #0xd3
49      msr cpsr, r0
```

三、关闭看门狗

通过看门狗寄存器关闭看门狗复位功能。

## 34.4 SPECIAL FUNCTION REGISTER

### 34.4.1 MEMORY MAP

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| WTCON | 0x7E004000 | R/W | Watchdog timer control register | 0x8021 |
| WTDAT | 0x7E004004 | R/W | Watchdog timer data register | 0x8000 |
| WTCNT | 0x7E004008 | R/W | Watchdog timer count register | 0x8000 |
| WTCLRINT | 0x7E00400C | W | Watchdog timer interrupt clear register | - |

| Register | Address | R/W | Description | Reset Value |
|---|---|---|---|---|
| WTCON | 0x7E004000 | R/W | Watchdog timer control register | 0x8021 |

| WTCON | Bit | Description | Initial State |
|---|---|---|---|
| Prescaler value | [15:8] | Prescaler value.<br>The valid range is from 0 to ($2^8$-1). | 0x80 |
| Reserved | [7:6] | Reserved.<br>These two bits must be 00 in normal operation. | 00 |
| Watchdog timer | [5] | Enable or disable bit of Watchdog timer.<br>0 = Disable<br>1 = Enable | 1 |
| Clock select | [4:3] | Determine the clock division factor.<br>00: 16 01 : 32<br>10: 64 11 : 128 | 00 |
| Interrupt generation | [2] | Enable or disable bit of the interrupt.<br>0 = Disable<br>1 = Enable | 0 |
| Reserved | [1] | Reserved.<br>This bit must be 0 in normal operation. | 0 |
| Reset enable/disable | [0] | Enable or disable bit of Watchdog timer output for reset signal.<br>1: Assert reset signal of the S3C6410 at watchdog time-out<br>0: Disable the reset function of the watchdog timer. | 1 |

```
53 #define WTCON 0x7E004000
54 disable_watchdog:
55      ldr r0, =WTCON
56      mov r1, #0x0
57      str r1, [r0]
58      mov pc, lr
```

四、关闭中断

因为前面在程序状态寄存器已经关闭了总的IRQ和FIQ，但对于独立的中断，也可以单独使能，所以通过 Interrupt Enable Clear Register 来禁止中断（操作掩码）。

**12.6.6 INTERRUPT ENABLE CLEAR, VICINTENCLEAR**

| REGISTER | ADDRESS | R/W | DESCRIPTION | RESET VALUE |
|---|---|---|---|---|
| VIC0INTENCLEAR | 0x7120_0014 | W | Interrupt Enable Clear Register (VIC0) | - |
| VIC1INTENCLEAR | 0x7130_0014 | W | Interrupt Enable Clear Register (VIC1) | - |

| Name | BIT | DESCRIPTION | RESET VALUE |
|---|---|---|---|
| IntEnable Clear | [31:0] | Clears corresponding bits in the VICINTENABLE Register:<br>0 = no effect<br>1 = interrupt disabled in VICINTENABLE Register.<br>There is one bit of the register for each interrupt source. | - |

```
61 #define ELFIN_VIC0_BASE_ADDR (0x71200000)
62 #define ELFIN_VIC1_BASE_ADDR (0x71300000)
63 disable_interrupt:
64      ldr r0, =ELFIN_VIC0_BASE_ADDR
65      ldr r1, =ELFIN_VIC1_BASE_ADDR
66      mvn r3, 0x0
67      str r3, [r0, #0x14]
68      str r3, [r1, #0x14]
69      mov pc, lr
```

五、关闭MMU与Cache

5.1、MMU定义与作用

MMU是Memory Management Unit的缩写，中文名是内存管理单元，它是中央处理器（CPU）中用来管理虚拟存储器、物理存储器的控制线路，同时也负责虚拟地址映射为物理地址，以及提供硬件机制的内存访问授权，多用户多进程操作系统。

5.2、Cache的定义与作用

Cache被用作CPU针对内存的缓存，利用程序的空间局部性和时间局部性原理，达到较高的命中率从而避免CPU每次都一定要与相对

慢速的内存交互数据来提高数据的访问速率。

### 5.3、关闭MMU和Cache操作

#### 5.3.1、刷新 I/D cache

```
74 flush_cache:
75      mov r0, #0
76      mcr p15, 0, r0, c7, c7, 0
77      mcr p15, 0, r0, c8, c7, 0
78      mov pc, lr
```

#### 5.3.2、关闭MMU和Cache

**Table 3-39 Control Register bit functions (continued)**

| Bits | Field name | Access | Function |
|---|---|---|---|
| [2] | C bit | Banked | Enables level one data cache.<br>0 = Data cache disabled, reset value.<br>1 = Data cache enabled. |
| [1] | A bit | Banked | Enables strict alignment of data to detect alignment faults in data accesses. The A bit setting takes priority over the U bit.<br>0 = Strict alignment fault checking disabled, reset value.<br>1 = Strict alignment fault checking enabled. |
| [0] | M bit | Banked | Enables the MMU.<br>0 = MMU disabled, reset value.<br>1 = MMU enabled. |

Attempts to read or write the Control Register from Secure or Non-secure User modes results in an Undefined exception.

Attempts to write to this register in Secure Privileged mode when **CP15SDISABLE** is HIGH result in an Undefined exception, see *TrustZone write access disable* on page 2-9.

Attempts to write Secure modify only bit in Non-secure privileged modes are ignored.

Attempts to read Secure modify only bits return the Secure bit value. Table 3-40 lists the actions that result from attempted access for each mode.

```
80 disable_mmu_cache:
81      mrc p15, 0, r0, c1, c0, 0
82      bic r0, r0, #0x00000007
83      mcr p15, 0, r0, c1, c0, 0
84      mov pc, lr
```