Hercules 入门知识

什么是 Hercules?

Hercules 安全微处理器是基于 TI 针对汽车电子市场的 20 多年安全关键型系统专业技术、行业协作而发展成熟的硬件。该平台包含三个基于 ARM® Cortex™ 的微处理器系列（RM48x、TMS570 和 TMS470M），可提供可扩展的性能、连接、内存和安全功能。与强烈依赖于软件以获得安全功能的某些微处理器不同，Hercules 微处理器在硬件内实施安全保护，从而使性能最佳化并减少软件使用量。

产品系列

RM4x 是 Hercules 安全微处理器系列中性能最高的产品。基于运行速度高达 220 MHz 的 ARM® Cortex™-R4F 浮点内核，它包括闪存和连接选项。为满足 IEC 61508 SIL-3 安全标准和支持要求，已在硬件中集成多个安全功能。

TMS470M 安全微处理器系列基于广泛采用的、运行速度为 80 MHz 的 ARM® Cortex™-M3 CPU。此系列提供多种闪存和 RAM 存储器选项，并提供 CAN 和 LIN 连接及其灵活的控制外设。包括 CPU 和 RAM 自检 (BIST) 引擎、ECC 和奇偶校验等内置安全功能。TMS470M 安全微处理器同样符合 AEC-Q100 标准。

TMS570LS 安全微处理器系列让客户能轻松开发安全关键型运输应用产品。这款基于 ARM® Cortex™-R4F 的系列微处理器可提供多种性能、存储器和连接选择，已开发出满足 ISO 26262 ASIL D 和 IEC 61508 SIL-3 安全标准和符合 AEC-Q100 汽车电子规范的产品。

硬件中的安全功能

RM48x 和 TMS570 双核 CPU 锁步构架在消除冗余系统要求的同时也使开发得以简化，从而降低了成本。CPU 硬件内置自检 (BIST) 无需复杂安全软件和代码大小费用即可检测到潜在缺陷。CPU 输出的硬件比较提供即时安全响应时间，而不会对其它性能产生影响。ECC 逻辑集成在 CPU 中，这会对存储器和总线起到保护作用。使用 HW BIST 可对所有 RAM 存储器进行测试，从而实现高诊断覆盖范围，且内存保护单元 (MPU) 对应用软件中的确定性错误起到帮助防范作用。

Hercules 之 GIO 学习入门

GIO 模块提供数字输入捕捉和数字输入/输出。在这个块中没有处理功能。GIO 通常用于静态的或者很少发生改变的输出，诸如收发器使能信 号、报警光等。GIO 也可被用于提供外部中断输入功能。GIO 的大体框 图如下：（以 TMS570 系列为例）

GIO 主要特点

GIOA 和 GIOB 的每个端口都包含 8 个双向可位操作的 IO 引脚

GIOA 具有外部中断触发功能

可编程的单边沿触发或者双边沿触发；可编程的边沿极性；可编程的中断优先级；管脚的配置问题；配置数据传输方向；数据输入或者输出配置；数据设置或者清零设置；漏极开路配置；上拉下拉配置。以上的配置都是通过相关的寄存器实现，配置寄存器如下

相关寄存器的介绍

GIOPSL 选择上拉或者下拉操作的寄存器

GIOPULDIS 失能管脚上下拉操作

GIOPDR 控制管脚的漏极开路配置

GIODOUT 配置输出管脚的电平

GIODIN 从外部管脚读到的数据信息

GIODIR 控制管脚的方向

Hercules 系列的 ADC 学习

Hercules 器件系列产品执行两个带有共享通道的用于快速转换的模块（乒乓操作方法）。使用双 ADC 转换器来执行两个通道的系统也许能够在应用中请求故障容错。 Hercules 系列的 ADC 不同于我们普通 CPU 里面的 ADC，称为 MibADC，ADC 的结构也是 SAR，分辨率是 12 位。有两个 ADC 核的 CPU 有 24 个通道的 ADC 例如 MS570LS20216 。24 个通道的组成是每个 ADC 核自己拥有单独的 16 个通道的 ADC，还有 8 个通道 是两个 ADC 内核共享的。

Hercules 的嵌套中断学习理解

基本上，Cortex R 内核只提供了 IRQ 和 FIQ 两种默认中断，而且进入 IR 的 ISR 后，CPU 会自动屏蔽其他所有 IRQ。也就是说，只要是 IRQ， 无论优先顺位，一律不能嵌套，直到当前的 IRQ 的 ISR 执行完成。 如果用户想实现 IRQ 的中断嵌套，那么可以在进入 ISR 之后，手动将 CPSR (current program status register)寄存器的"I" 位清零。这样后面的更高优先级的 IRQ 就可以得到响应，并嵌套进来了。 这个嵌套是没有层级限制的但是用户需要自行对被打断的 ISR 现场进行保护，这个工作也会着嵌套层级增多而变得复杂，稍有不慎，就可能引起内存溢出。所以建议大家谨慎使用中断嵌套。

在 IRQ 处理函数起始处通过如下所示汇编代码进行现场保护，处理完毕后再进行恢复。同时在保护好现场后需要将优先级等于或小于本身的 IRQ 中断禁掉，恢复现场后再将相应中断打开。中断关闭打开操作可以通过 VIM 进行。这样就可以实现高优先 IRQ 的
嵌套处理。

```
#pragma INTERRUPT(rtiCompare0Interrupt, IRQ)
void rtiCompare0Interrupt(void)
{
/* USER CODE BEGIN (39) */
/* USER CODE END */
```

```
    rtiREG1->INTFLAG = 1U;
    asm("     STMFD SP!, {R0-R12, LR}");/*Save R0- R12, LR_irq*
/
    asm("     mrs lr, spsr"); /* Copy SPSR_irq to LR */
    asm("     STMFD SP!, {LR}"); /* Save SPSR_irq */
    asm("     MSR CPSR_c, #0x1F"); /* Enable IRQ (Sys Mode) */
    asm("     STMFD SP!, {LR}"); /* Save LR */
    rtiDisableNotification(rtiNOTIFICATION_COMPARE0);/*Disable rtiCo
mpare1 Interrupt iteself */
    rtiNotification(rtiNOTIFICATION_COMPARE0);
    //asm(" LDMFD SP!, {R0-R12}");/*Restore R0-R12*/
        asm(" LDMFD SP!, {LR}"); /* Restore LR */
        asm(" MSR CPSR_c, #0x92"); /* Disable IRQ (IRQ Mode) */
        asm(" LDMFD SP!, {LR}"); /* Restore SPSR_irq to LR */
        asm(" MSR SPSR_cxsf, LR"); /* Copy LR to SPSR_irq */
        asm("     LDMFD SP!, {R0-R12, LR}");/* Restore LR */
        rtiEnableNotification(rtiNOTIFICATION_COMPARE0);/*Enable rti
Compare1 Interrupt    */
}
```

记得做好保护啥

Hercules 开发流程

1.使用 HALCoGen，根据设定的配置生成代码使用 HALCoGen 生成代码

2.将代码导入到 CCS5 中

3. 由用户根据需要完善代码功能

4. 然后进行编译

5. 编译无误后，利用下载工具下载到 CPU 中

6.重新上电 运行

以上就是开发 Hercules 系列的一般流程 。