

无限可能+简便易用
TI MCU > 其他 MCU



2012 TI Hercules Seminar



•Hercules™ 培训

www.ti.com/herculestraining

培训课程:

Hercules 安全研讨会



- 开班仪式
- 最新 Hercules/SafeTI 介绍
- 什么是功能性安全?
- 安全标准概述
- IEC 61508 安全标准
- ISO 26262 安全标准
- 随机失效管理
- 安全系统架构
- Hercules 安全概念

- 实验 2: 将 NHET 用作 GIO
- 通信接口: 异步收发器 (UART), 本地互连网络接口 (LIN), 控制器局域网 (CAN), FlexRay, 多缓冲串行外设接口 (MibSPI)

- 实验 3: PC 与串行通信接口 (SCI) 通信
- 外部存储器接口 (EMIF) / 参数覆盖
- 多缓冲模数转换器 (MibADC)
- 支持结构: 网络、论坛、维基网站 (WIKI)

- 实验 1: Hercules 安全微控制器 (MCU) 演示
- Hercules 架构
- 开发工具: 硬件 (HW) 套件, 软件 (SW) 工具
- 嵌入式闪存存储器工具
- 实时中断 (RTI)
- 矢量中断管理器 (VIM)
- 直接内存访问 (DMA)
- 通用输入输出接口 (GIO) & 高端定时器 (NHET)

- 实验 4: 安全电机控制演示
- Hercules controlCARD 概述
- 浏览 Hercules MotorWare™
- DSP 库介绍
- 电机控制的安全方面

•谁应参加此课程:

- 硬件和软件开发人员
- 项目经理
- 安全专家
- 任何一位对 Hercules MCU 和功能安全感兴趣的人员



Hercules™ 安全 MCU: 介绍



3

TI 嵌入式处理产品系列

微控制器				应用处理器 / 数字信号处理器 (DSP)		
16 位	32 位实时	32 位 M3/M4F ARM	32 位 R4F ARM	32 位 ARM+	32 位 ARM+DSP	DSP
ARM 内核提供以下器件						
MSP430 超低功耗 高达 25 MHz 闪存 1 KB 至 256 KB 模拟 I/O, ADC, LCD, USB, 射频(RF) 测量、感测、通用	C2000 定点 & 浮点 高达 300 MHz 闪存 32 KB 至 512 KB PWM, ADC, CAN, SPI, I ² C 电机控制、数字电源、照明	Stellaris-M3 工业标准低功耗 <100 MHz 闪存 高达 512 KB USB, ENET, ADC, PWM, CAN 主机控制	Hercules™ TMS570/RM4 浮点 超过 350 DMIPS 闪存 高达 3 MB 定时器协处理器 ENET, ADC, CAN 安全运输, 工业 & 医疗	ARM9 ARM Cortex-A8 工业标准内核, 高性能 GPP 加速器 MMU USB, LCD, MMC, EMAC Linux/WinCE 用户应用程序	ARM9/Cortex-A8 加上 C64x+ 工业标准内核 + 用于信号处理的 DSP 4800 MMACS/1.07 DMIPS/MHz MMU, 高速缓存 VPSS, USB, EMAC, MMC Lin/Win O/S + 视频、图像、多媒体	C55x, C64x+ C647x 领先的 DSP 性能 24,000 MMACS 高达 3 MB L2 高速缓存 1G EMAC, SRIO, DDR2, PCI-66 通信, WIMAX, 工业用、医疗成像



软件、工具和开发板



4

TI Hercules™ MCU 平台

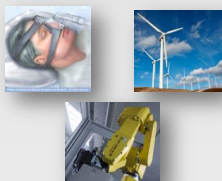
基于 ARM® Cortex™ 的微控制器

Hercules 平台

RM4x

高性能 工业和医疗用 安全 MCU

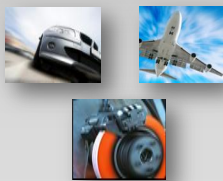
- 工业应用
- 医疗应用
- -40 至 105° C 的运行温度范围
- 以太网, USB, CAN & UART
- 针对以下安全标准开发
 - IEC 61508 SIL-3
- Cortex-R – 超过 350 DMIPs



TMS570

高性能 交通运输和 安全 MCU

- 交通运输应用
- 符合汽车级 Q100 规范
- -40 至 125° C 运行温度范围
- FlexRay, 以太网, CAN, LIN/UART
- 针对以下安全标准开发
 - ISO 26262 ASIL-D
 - IEC 61508 SIL-3
- Cortex-R – 超过 280 DMIPs



TMS470M

超值线 运输和 安全 MCUs

- 交通运输应用
- 符合汽车级 Q100 规范
- -40 至 125° C 运行温度范围
- CAN, LIN/UART 连通性
- 支持针对以下系统的安全性
 - IEC 61508 系统
- Cortex-M – 达到 100 DMIPs



5

Hercules™ ARM® Cortex™ 安全 MCU 应用

航空航天 & 铁路应用



工业应用



汽车应用

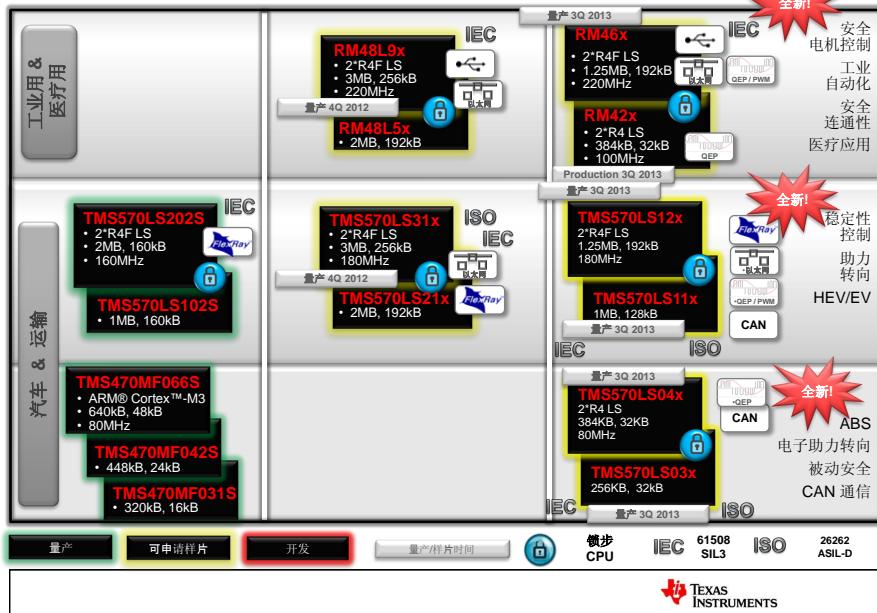


医疗应用



6

Hercules™ 安全 MCU 产品路线图



RM4x 系列 – 对比表

系列	RM48x		RM46x		RM42x
CPU	Cortex-R4 w/ FPU	Cortex-R4 w/ FPU	Cortex-R4 w/ FPU	Cortex-R4 w/ FPU	Cortex-R4
CPU 最高频率	220MHz	220MHz	220MHz	220MHz	100MHz
闪存	3MB / 2MB	3MB / 2MB	1.25MB / 1MB	1.25MB / 1MB	384KB / 256KB
RAM	最高 256KB	最高 256KB	最高 192KB	最高 192KB	24KB
数据闪存 (EE 模拟)	64KB	64KB	64KB	64KB	16KB
DMA	16 通道	16 通道	16 通道	16 通道	无
NHET (通道数)	2 (44)	2 (40)	2 (44)	2 (40)	1 (19)
eCAP/ePWM/eQEP	无	无	6 通道/14 通道/2 模块	6 通道/14 通道/2 模块	1 QEP
MbADC 12 位 (通道数)	2 (24 通道)	2 (24 通道)	2 (24 通道)	2 (24 通道)	1 (16 通道)
以太网	可选	可选	可选	可选	无
USB 主机/事件	可选	可选	可选	可选	无
CAN (mbox)	3 (每个 64mb)	3 (每个 64mb)	3 (每个 64mb)	3 (每个 64mb)	2 (32mb & 16mb)
MbSPI1 (已缓冲)	3	3	3	3	1
标准 SPI	2	2	2	2	2
I2C	1	1	1	1	None
UART	2	2	2	2	2
GPIO	16	10	16	10	8
EMIF	有	无	有	无	无
ETM/RTIP/DMM	有	无	无	无	无
温度范围	-40c 至 105c	-40c 至 105c	-40c 至 105c	-40c 至 105c	-40c 至 105c
封装	337BGA	144QFP	337BGA	144QFP	100QFP

请注意：上面的表格显示了每个模块的最大配置-某些功能是可复用的

TI 信息-有选择性公开



TMS570LS 系列 – 对比表

系列	TMS570LS31x/21x		TMS570LS12x/11x		TMS570LS04x/03x
CPU	Cortex-R4 w/ FPU	Cortex-R4 w/ FPU	Cortex-R4 w/ FPU	Cortex-R4 w/ FPU	Cortex-R4
CPU 最大频率	180MHz	160MHz	160MHz	160MHz	80MHz
闪存	3MB / 2MB	3MB / 2MB	1.25MB / 1MB	1.25MB / 1MB	384KB / 256KB
RAM	最高 256KB	最高 256KB	最高 192KB	最高 192KB	24KB
数据闪存 (EE 模拟)	64KB	64KB	64KB	64KB	16KB
DMA	16 通道	16 通道	16 通道	16 通道	无
NHET (通道数)	2 (44)	2 (40)	2 (44)	2 (40)	1 (19)
eCAP/ePWM/eQEP	无	无	6 通道/14 通道/2 模块	6 通道/14 通道/2 模块	1 QEP
MilADC 12 位 (通道数)	2 (24 通道)	2 (24 通道)	2 (24 通道)	2 (24 通道)	1 (16 通道)
以太网	可选	可选	可选	可选	无
FlexRay	可选	可选	可选	可选	无
CAN (mbox)	3 (每个 64mb)	3 (每个 64mb)	3 (每个 64mb)	3 (每个 64mb)	2 (32mb & 16mb)
MilSPI (巴拿马)	3	3	3	3	1
标准 SPI	2	2	2	2	2
I2C	1	1	1	1	None
LIN / SCI	1 / 1	1 / 1	1 / 1	1 / 1	1 / 0
GPIO (w/ Flexray)	16	10 (4)	16	10 (4)	8
EMIF	有	无	有	无	无
ETM/RTDMM	有	无	无	无	无
温度范围	-40c 至 125c	-40c 至 125c	-40c 至 125c	-40c 至 125c	-40c 至 125c
Package	337BGA	144QFP	337BGA	144QFP	100QFP

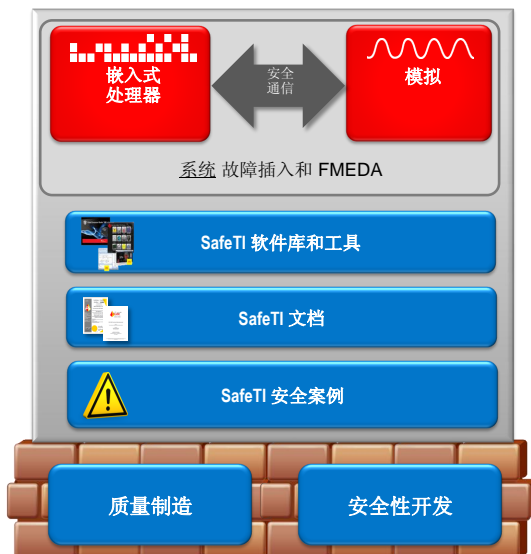
请注意，上面的表格显示了每个模块的最大配置-某些功能是可复用的

TI 信息-有选择性公开



使用 SafeTI™ 设计包加速功能安全认证

特定 SafeTI 一揽子解决方案



- SafeTI - 61508
工业、医疗、铁路和其它应用
- SafeTI - 26262
汽车类应用
- SafeTI - 60730
家用电器
- SafeTI - QM
质量管理

See www.ti.com/safeti



什么是功能性安全 & 安全性标准概述

世界需要变得更加安全

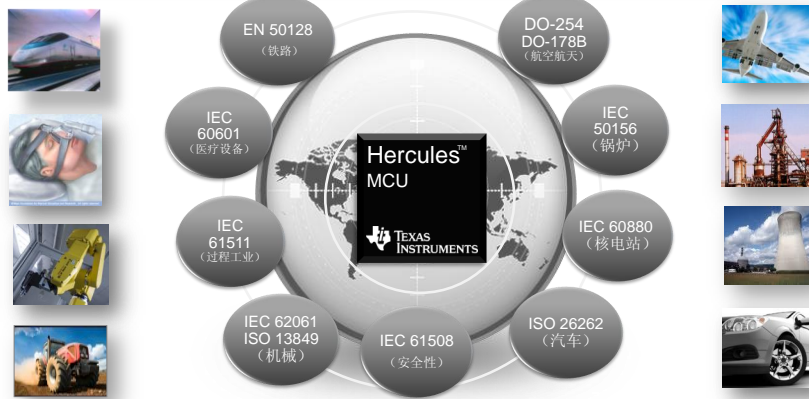
▶ 电子器件正被越来越多的应用于安全关键类应用中



- ▶ 系统中不断增加的电子元器件的数量
- ▶ 增加的电子硬件和软件复杂度
- ▶ 更多的系统将安全功能包括在内
- ▶ 不断增加的政府安全规定
- ▶ 过去十年中在铁路、能源、航空航天和汽车应用中被曝光的安全故障

国际安全标准

▶ 电子元器件正在被越来越多的应用于很多安全应用中



- IEC 61508 是被用作特定安全标准的基准，如果某些领域不存在针对性的安全标准，它也可直接使用IEC61508。
- Hercules™ TMS570LS20216 器件已经由 exida 认证用于 IEC 61508 SIL-3 系统。

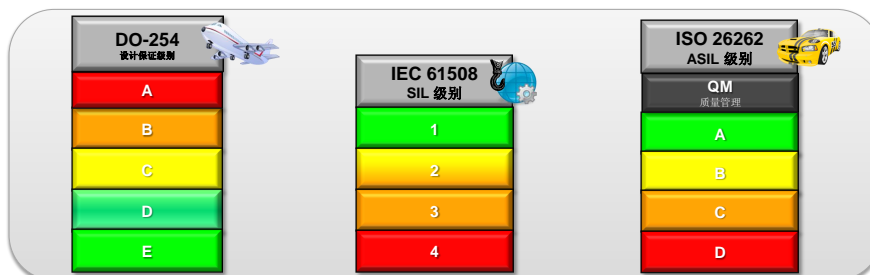
TEXAS INSTRUMENTS

13

什么是功能性安全？

• 基本功能性安全概念：

- 所有系统都将有一些固有的、可量化的故障率。开发一个零故障率的系统是不可能的。
- 对于每个应用，有一些可容忍的不会导致不可接受风险的故障率。
- 每个应用所接受的故障率是不同的，其基于系统发生故障时，直接或间接的造成人身伤害的可能性
- 可将相同级别的风险进行量化分类。这些被称为安全完整性级别， **Safety Integrity Levels** 或者简称为 **SIL**。



TEXAS INSTRUMENTS

14

什么是功能性安全？



- 一个功能安全电子系统是能够对每一组输入做出预期响应的系统
- 功能性安全要求系统在运行状态时，必须能实时应用质量、可靠性和产品测试技术以确保产品对于提供的输入做出正确的响应。
- 源自 IEC 61508 标准的经典功能性安全案例：
 - 一个能检测过温故障并且在故障之前将电机断电的实时诊断是功能性安全的一个范例。
 - 一个能耐过温条件而又不会发生故障的特别设计的电机不是功能性安全的一个范例。



功能性安全定义



- **IEC 61508 定义：**
 - 安全是避免系统遭受对人身财产安全造成直接或间接的损害或者对环境造成不良影响等不可接受的风险
 - 功能性安全是总体安全性的一部分，取决于对系统或装置对输入正确的响应
- **ISO 26262 定义：**
 - 避免由于电子或电子系统失效行为造成不可接受的风险



16

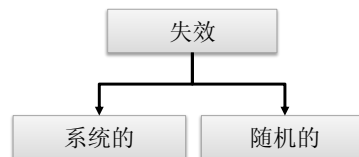
其他安全概念 & 定义



- **错误:**
 - 一个系统中有可能导致失效的运行事件。
- **故障:**
 - 错误导致系统无法执行安全性功能
- **容错:**
 - 错误发生后可继续安全运行的能力。
- **故障安全系统:**
 - 在一个有可能导致故障的错误被检测到的系统中，此系统被置于一个安全状态，这样错误就不会传播到其它系统
- **故障可用/可操作系统:**
 - 在一个有可能导致故障的错误被检测到的系统中，此系统可在不丢失安全性功能的情况下继续运行。
- **可靠性**
 - 系统无故障运行的性能（通常与对于一个安全功能的考虑无关）
- **可用性**
 - 安全性功能可用的时间除以总体系统运行时间。具有高可靠性和故障可操作系统往往比故障安全系统具有更高的可用性
- **安全**
 - 检测、抑制或防止损害产品功能性的能力。

系统失效和随机失效

- 功能性安全系统中的失效可大体被分成两类：系统失效和随机失效



- **系统失效**
 - 由设计、制造或使用中的可重复事件造成
 - 往往是不遵照最佳做法的结果
 - 可使用传统的质量和可靠性方法来管理和减少系统故障
- **随机失效**
 - 由不可预见性的原因导致
 - 随机失效的几率不能被降低; 应用中重点应放在随机失效的检测及处理上

功能性安全与质量/可靠性之间的关系

- 高质量和可靠性并不确保安全性。
- 确保质量和可靠性的方法与管理系统安全故障的方法很大部分相同。
- 管理随机硬件失效的要求通常与质量、可靠性要求不同。

当面对一个可能因过热故障引起线路融化从而导致系统故障的系统时：

- 一个可靠性工程师将此系统设计成具有耐高温的系统来解决这个问题
- 一个功能安全工程师把这个系统设计成能检测过温并在线路融化前将系统置于一个安全状态来解决这个问题。

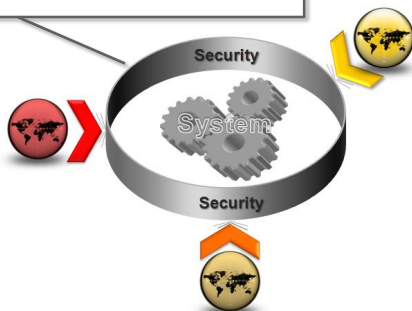


功能性安全与安全之间的关系

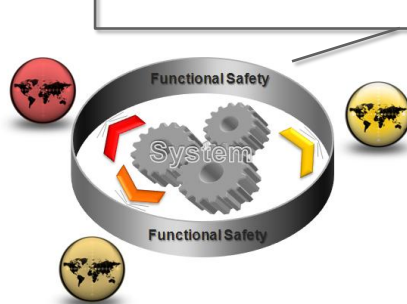
- 一个安全系统能够检测、抑制或防止来自外部环境的损害
- 一个功能安全系统能够检测故障并防止对外部环境的损害



安全系统：
避免外部环境对系统的操作



功能安全系统：
不会产生对人体健康、系统或环境有威胁的不可接受的风险



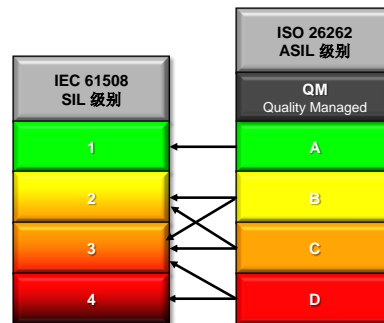
安全性目标



- 必须为一个安全系统定义安全性目标
 - 例如：汽车安全气囊系统：
 - 事故发生时安全气囊必须展开
 - 当没有事故发生时，安全气囊一定不能被意外打开。
- 安全目标被用来帮助确定系统中所需的功能安全级别。
- 这些安全级别通常是指安全完整性级别或 SIL。
- 不同的安全标准对 SIL 的命名和分类会不同。

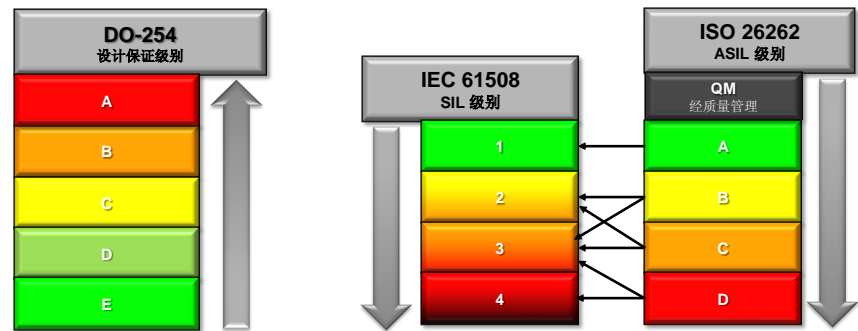
ISO 26262 与 IEC 61508 安全完整性级别对比

- 作为 IEC 61508 的替代，ISO 26262 是专门针对汽车应用。
- IEC 61508 定义了 4 个安全完整性级别 (SIL-1,2,3,4)
- ISO26262 除了定义 4 个安全完整性级别 (ASIL-A,B,C,D) 之外还定义了一个质量管理
- IEC61508 SIL 和 ISO 26262 ASIL 级别之间没有直接关联

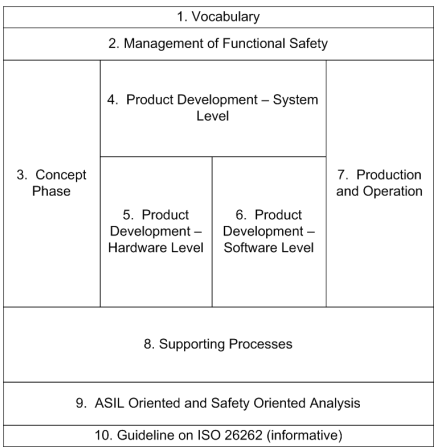


航空航天 DO-254 设计保障级别

- DO-254 包含 5 个 “设计保障级别” (DAL)
 - DAL-A 是最严格的，DAL-E 是最宽松的
 - 必须经 FAA “指定工程代表” (DER) 审核并批准



ISO 26262 标准

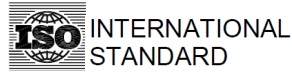


- ISO 26262 是针对乘用车应用的功能安全标准。
- 相较于 IEC 61508，ISO26262 更适合于汽车应用开发：
 - 安全性必须贯穿整个开发流程并且在产品定义时即作出安全保证，而不是在现有终端设备上来应用和验证
 - 周期内的运行时间小于 1 小时，而工业系统的此类时间为几天/几个月/几年
 - 所有功能被认为具有 “高需求” -必须在车辆上电时可用
 - 针对多点潜在故障的功能和诊断

ISO 26262 标准介绍

• 标准的当前状态是什么？

- 在 2006 年时开始开发
- 章节 1-9 作为国际标准 (IS) 发布于 2011 年 11 月
- 章节 10 于 2012 年 3 月作为最终草案国际标准 (FDIS) 被发布
- 章节 10 IS 预计将于 2012 年年底发布



• ISO 26262 何时生效？

- 目前，没有国家立法要求执行 ISO 26262
- 在某些国家和地区，OEM 有责任参照最新的技术标准来进行开发。而 ISO 26262 是目前最新的安全标准
- 到 2012 年底，很多 OEM 要求新产品开发必须符合 ISO 26262



对系统设计和组件选择的影响

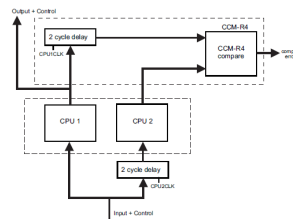
• 必须对现有产品进行重新设计以符合 ISO 26262 吗？

- 现有的已经验证过的量产产品可以继续使用。
- 如果硬件、软件或系统设计发生任何变更，升级后的产品必须按照 ISO 26262 进行开发

• 所有组件必须按照 ISO 26262 进行开发吗？

- 不是。简单组件（电阻，电容等）可由一个资质审查过程进行检查。
- 合格的组件仍必须在系统安全分析中进行考虑。

- 由开发人员依据相应的失效模式来采用适当的诊断手段以实现目标 ASIL 等级。对于失效模式和架构没有特定要求。
- 双核锁步 CPU 器件，例如 TI 的 Hercules TMS570 MCU，能够大大简化系统开发、缩短安全认证时间并帮助客户更快的将产品推向终端市场。



产品评估/认证

• 要求评估?

- 是的。对于汽车应用，必须是来自外部评估。
对于系统组件级应用，可以是内部或外部评估。
- 对于内部评估，开发和评估人员必须是独立的
(不同团队、不同管理、不同部门等)



• 谁能够评估并且/或者认证我的产品?

- 需要公认可信的评估人员提供认证服务
- 很多公司可提供这些服务，例如 TÜV SÜD, TÜV NORD, 和 exida.

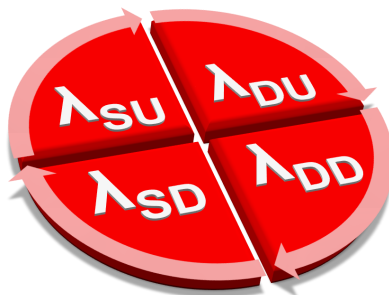
• 针对是否符合规范，需要评估哪些内容?

- 清晰列出由标准定义的产品的详尽列表
- 出具的相关证明不会因内部或者外部评估而发生变化



IEC 61508 失效率

失效率 λ



- λ_S - 安全失效率
 - 对安全功能**无**影响
 - λ_{SD} - 检测到的安全失效率
 - λ_{SU} - 未检测到的安全失效率
- λ_D - 危险失效率
 - 对安全功能**有**影响
 - λ_{DD} - 检测到的危险失效率
 - λ_{DU} - 未检测到的危险失效率

$$\lambda = \lambda_S + \lambda_D = (\lambda_{SD} + \lambda_{SU}) + (\lambda_{DD} + \lambda_{DU})$$

FIT = 特定时间内的失效数 = 10^9 器件小时内 1 次失效



28

IEC 61508 安全失效划分 & SIL 确定



$$\text{安全失效划分 (SFF)} = 1 - \frac{\lambda_{DU}}{\lambda}$$

高需求系统

硬件错误容差 = 0 (单通道)

- 1 个错误有可能导致安全功能的损失。
- EX: 1oo1, 1oo1D, 2oo2...

硬件错误容差 = 1 (冗余)

- 2 个或更多错误会导致安全功能的损失。
- 2oo3, 4oo5...

安全失效划分 (高需求系统)	硬件错误容差	
	HFT = 0	HFT = 1
0 ... < 60	-	SIL1
60 ... < 90	SIL1	SIL2
90 ... < 99	SIL2	SIL3
≥ 99	SIL3	SIL4



29

ISO 26262 - IEC 61508 主要差异



30

ISO 26262 与 IEC 61508 的主要差异

- 与汽车工业使用实例和可接受风险的定义保持一致
- IEC 61508 安全功能的概念被安全目标所取代。
 - 安全功能基于对受控系统的定义并随后“采取”措施减少风险
 - 安全目标要求在系统的研发初始阶段，其控制系统设计就必须考虑降低风险
- 典型 IEC 61508 系统可以在组装完成后才在现场验证。ISO 26262 系统必须在上市前进行验证。
- ISO 26262 针对产品的要求定义很清晰。这使得开发者更容易设计符合 ISO26262 标准的产品，但同时也限制了系统定义开发的灵活性。
- ISO 26262 具有适合于汽车安全实例的危险和风险分析、失效率。

安全系统架构

安全系统架构

- 功能性安全的系统架构有很多种。下面会简要介绍最常见的架构。
- 常用术语为 “XooY”
 - Y 中 X 的数量
 - 总共 Y 个组件中，必须有 X 个组件发生失效才会导致整个系统失效

1oo1 系统架构



- 一个子系统失效将导致整个系统失效
- 最基本的系统配置
- 无内部诊断
- 无错误冗余

1oo2系统架构



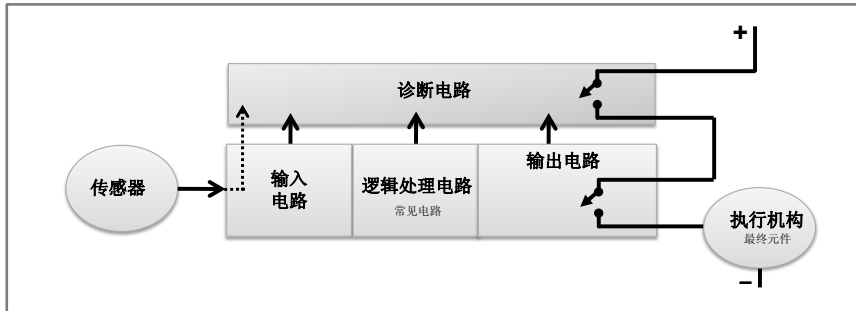
- “两个中的一个” 子系统发生失效才会导致系统失效
- 两个具有独立 I/O 的控制器
 - 每个控制器都必须控制输出
 - 任何一个失效将导致系统失效
- 在安全气囊系统中，通常由一个32位主CPU和一个8位CPU组成来控制爆管

2oo2系统架构



- 两个子系统都失效才会导致系统失效
- 每个子系统都可以控制输出
 - 用于当系统是闭合安全情况
 - 子系统开路有故障冗余

1oo1D系统架构



- 1oo1 系统的扩展版本，带有诊断通道
- 诊断通道可在检测到故障时禁止系统输出
- 诊断电路可能会导致额外的失效概率（通告故障）
- TI Hercules™ “双核锁步” 处理器是一个 1oo1D 系统

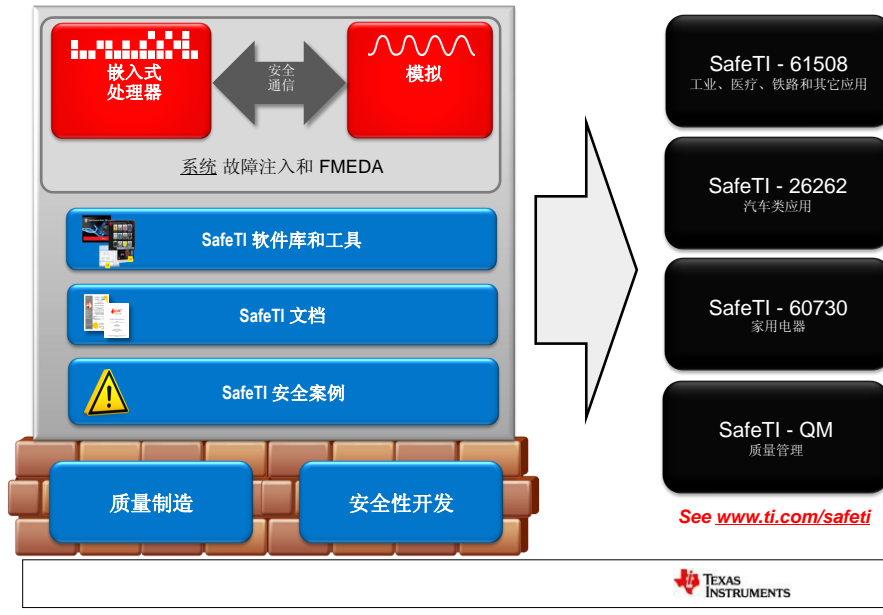
2oo3系统架构



- 3 个子系统中必须有 2 个子系统发生失效才会导致系统失效
- 容错
 - 开路故障时减为 1oo2 系统
 - 短路故障时减为 2oo2 系统

使用 SafeTI™ 设计包加速功能安全认证

SafeTI 一揽子解决方案



安全标准评估



- 所有 Hercules 产品将由独立的审核人员进行内部评估。
- 选择的 Hercules TMS570 产品将由外部审核人员进行评估。
- 对 Hercules MCU 的评估包括以下部分：
 - 开发过程
 - 产品安全性架构/概念
 - 基于开发过程和安全架构/概念的器件执行
- 针对 IEC 61508:2010 和 ISO 26262:2011, TI 已经选择与 TÜV-SÜD (经 DAKKS 授权) 和 Exida (经 ANSI 授权) 合作

TMS570 安全文档

- TI 在保密条款 (NDA) 下提供的相关文档有助于安全认证过程:



Hercules™ Cortex™-R4F 安全概念

Hercules™ 安全概念的合理性

- “安全岛”方法
- 所有安全功能共用的器件区域受到硬件诊断措施的重点保护
 - CPU
 - CPU 中断
 - 电源、复位和时钟的系统控制
 - OS 关键 IP: DMA, OS 定时器
- 一旦可确保一个已知安全区域, 这个区域的逻辑可被用于提供对于其它区域的诊断覆盖
- 这个分区已显示在大大减少安全对于系统物料清单 (BOM) 成本影响的同时提供强大的安全标准

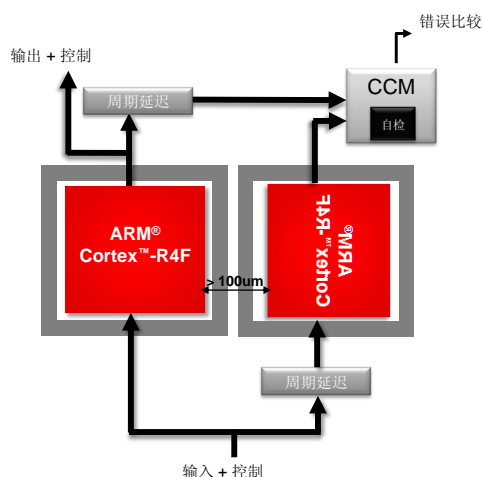
Hercules – 随机错误管理

- TI 的方法是重点保护带有硬件诊断的关键逻辑区域。这种方法提供一个已知正常逻辑的“安全岛”。
- 随后, 此“安全岛”可通过一个硬件、软件和系统诊断的成本有效性组合来检测器件其余部分的故障。
- **安全岛硬件诊断 (红)**
 - CPU 包含一个针对逐周期错误检测的锁步校验器。
 - 用于闪存和 SRAM 的 SECCDED ECC 检测每一个内存访问时的错误
 - 硬件 BIST 引擎可实现最终应用的工厂级测试
 - 电源、时钟和复位有内部和外部硬件诊断接线
- **混合硬件诊断 (蓝)**
 - 对外设存储器的奇偶校验
 - 模拟和数字 I/O 回路测试
 - ADC 自检和校准
 - 双 ADC 上的共用通道
- **非安全关键功能 (黑)**
 - 开发和调试模块
- 所有硬件诊断包括在校验器机制中启用延迟错误测试的机制



1001D 双核安全概念

- 独特设计以减少常见故障 (βIC)
 - 第二个 CPU 被镜像和旋转
 - CPU 之间最小距离100μm
 - 周期延迟锁步
 - 每个 CPU 的保护环
 - 每个 CPU 复制的时钟树
- CPU 比较模块 (CCM)
 - 自检功能
 - 自检错误注入/强制错误
 - 输出错误注入

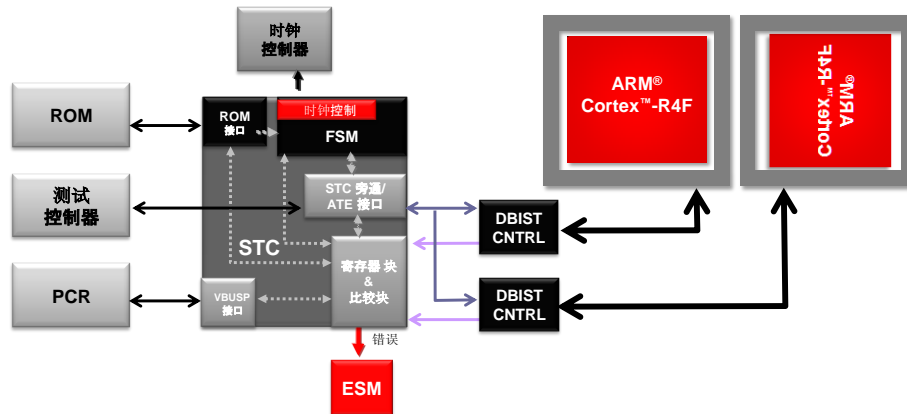


1001D 双核锁步优势



- 锁步 CPU 相对于基于软件和多核的解决方案间的优势
 - 更快速的错误检测。在使用紧密控制环时序的系统中启用安全性。
 - 更佳错误覆盖。可检测确实的、瞬态的和 AC 错误类型。
 - 对性能影响很小。全部 CPU 性能用于应用任务而非 CPU 诊断。
 - 最小内才能影响。闪存和 SRAM 用于应用而非 CPU 诊断。
 - 简便集成。减少上市时间，而无需集成复杂软件。
 - 已证明，可轻松调整诊断覆盖。花费更少的时间，这向审核人员证明您的 CPU 安全解决方案留出了更多的时间用于开发您的应用。

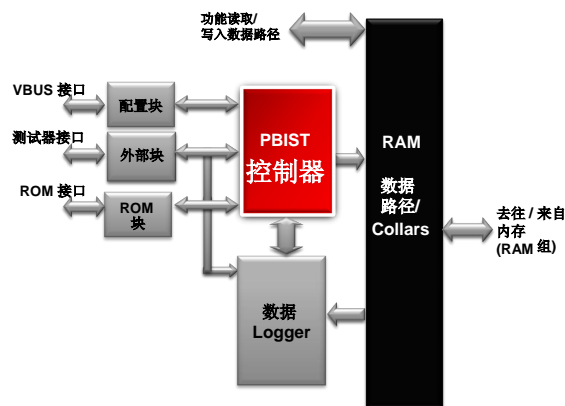
CPU 自检控制器 (STC/LBIST)



- 提供高诊断覆盖
- 大大降低了软件和运行时间开销
- 闪存内无软件 BIST (内置自检) 代码开销
- 可通过寄存器简便配置和启动 BIST

可编程内存 BIST (PBIST)

- 可测试所有片载 RAM
- 简单寄存器设置和配置
- 通常在启动时运行，但是在应用期间执行
- 多个内存测试算法
- 检测多个故障模式



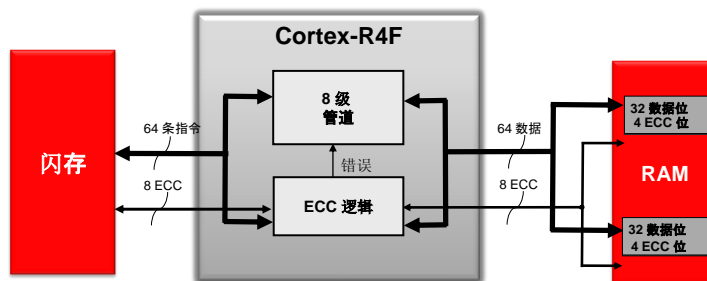
- 提供一个机制以确定运行时间错误是否由硬件或软件错误误差引起。这个功能可被用来通过从软件误差中的联机恢复来改进可用性。

硬件 BIST 优势



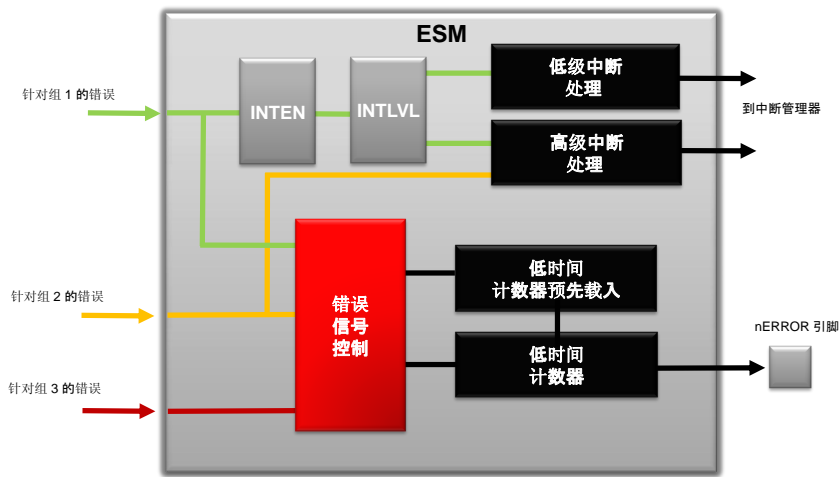
- 硬件 BIST 相对于软件测试的优势
 - 更加快速的测试执行。要达到同等的测试覆盖，软件测试要求10 倍到100 倍的运行时间。
 - 更佳错误覆盖。比只支持软件的解决方案有可能对多个错误模块进行寻址并实现更高的覆盖。
 - 最小内存影响。使您的闪存和 SRAM 用于应用而非内存和 CPU 测试。
 - 轻松集成。缩短上市时候而又无需集成复杂软件。
 - 已证明，可轻松调整诊断覆盖。花更多的时间在上应用上，花费更少的时间向审核人员证明您的 CPU 安全解决方案。

闪存 / RAM ECC 保护



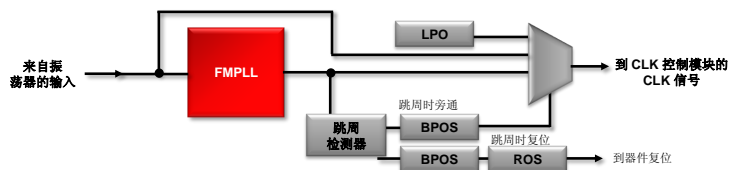
- 在 Cortex R4F CPU 内进行评估的 ECC
 - 单一位纠错和双位纠错 (SECEDED)
 - 与数据/指令处理并行进行评估的 ECC
 - 通常不影响性能
 - 保护 CPU 与闪存和 RAM 间的总线

错误信号模块 (ESM)



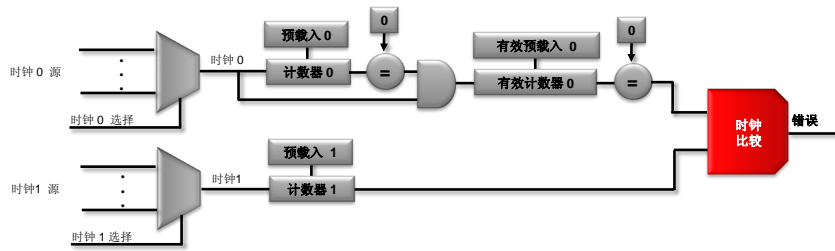
时钟监视

- 外部时钟与分频器 (ECLK)
 - 可实现 CPU 时钟频率的外部监视
 - 配置引脚 (GIO 或 ECLK)
- 振荡器监视器
 - 如果振荡器频率超过定义的最小/最大阈值则检测到故障
 - 振荡器故障时的可选硬件相应
 - 复位器件
 - 切换到内部“低功耗振荡器” (LPO) 时钟源
- FMPLL 跳周检测器
 - 如果相位锁丢失则表示 PLL 跳周
 - PLL 跳周时可选的硬件相应
 - 复位器件
 - 切换到内部“低功耗振荡器” (LPO) 时钟源
 - 切换到外部振荡器时钟源



双时钟比较器 (DCC)

- 使用一个第二时钟信号作为基准，DCC 模块被用于测量一个时钟信号的频率。
 - 使得应用能够确保在两个时钟信号间保持一个固定频率
 - 在基准时钟周期数量方面，支持一个可编程容差窗口的定义
 - 支持连续监视而无需应用介入
 - 或者可被用在针对现场测量的单序模式中
 - 导致几个特定使用实例的针对计数器 0 和计数器 1 的灵活时钟源选择

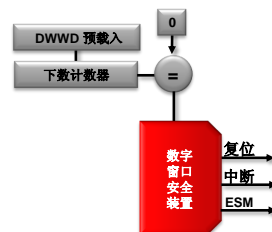
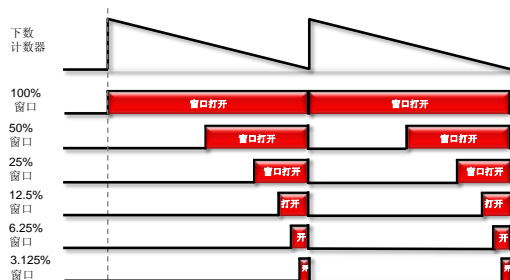


TEXAS
INSTRUMENTS

53

数字窗口安全装置 (DWWD)

- 如果应用在适当的时间窗口内没有成功处理安全装置，那么DWWD 模块将复位 MCU 或者生成一个到 CPU 的不可屏蔽中断。
 - 可检测一个 CPU 的可选安全诊断
 - 包括一个 25 位下数计数器
 - 当生成一个失控 CPU 中断时，警告错误信令模块
 - 支持多个处理窗口：100%, 50%, 25%, 12.5%, 3.125%
 - 处理要求一个特定的两部分密钥序列
 - 一旦被启用则只能由一个系统或加电复位禁用



TEXAS
INSTRUMENTS

54

内存保护单元 (MPU)

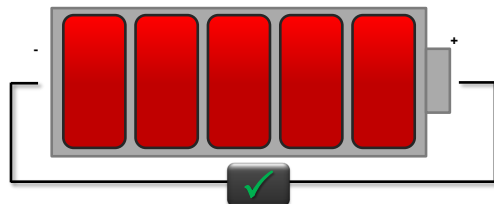
- 针对关键总线主控，执行一个专用内存保护单元 (MPU)



- 总线主控包括 CPU, DMA, HTU 和 FTU
- 一个被定义的内存区域可实现对总线主控的读取和写入访问
- 定义区域之外的访问可为一下任一模式
 - 只读:** 允许对区域外内存的读取访问。写入访问被阻止
 - 无访问: 读取和写入被阻止。
- 在内存保护违反错误指令模块时 (ESM) 通知

电压监视器

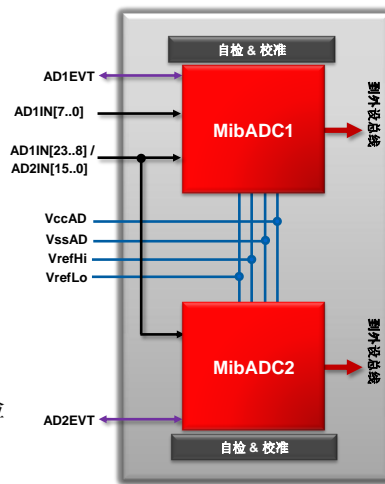
- 电源电压监视器 (VMON)
 - 保持复位直到内核和 I/O 电源轨在预期的范围内 (免除了对电源排序的要求)
 - 如果内核或 I/O 电源超过定义的最小/最大阈值的话讲复位置为有效
 - 当内核电源低于额定最小电压时将复位置为有效并且异步将所有 I/O 引脚置位为高阻抗模式



双模数转换器

• 双 12 位 ADC 内核:

- 内核 1 支持 24 个模拟输入 & 内核 2 支持 16 个模拟输入
- 2 个内核间可共用多达 16 个模拟通道以实现安全转换/比较 (1002 安全冗余)
- 内部 ADC 基准电压可被用来检查转换器功能性。
- 应用在 ADC 输入上检测到开路/短路时启用自检模式
- ADC 校准逻辑可改进准确度或被用来检测多个测试结果间的漂移。



TEXAS
INSTRUMENTS

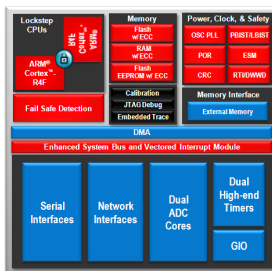
57

Hercules™ Safety MCU 为开发人员提供

Hercules MCU
为开发人员提供
关键安全应用:

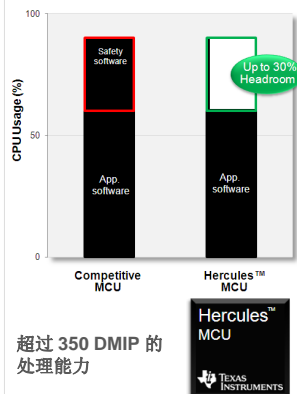
- ▶ 防止随机和系统故障
- ▶ 针对应用区分的净空余量
- ▶ 简化的开发和子系统认证

硬件安全架构

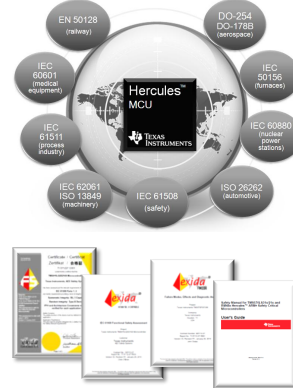


- 锁步 CPU
- CPU & RAM 内置自检
- 闪存 & RAM ECC
- 时钟监视
- 电压监视

应用净空余量



安全认证 & 文档



TI 机密-受 NDA 限制

TEXAS
INSTRUMENTS

58

Hercules™ MCU 开发工具



59

Hercules™ 开发套件

评估

TMDXRM48USB – RM48 USB 棒套件
TMDX570LS31USB – TMS570 USB 棒套件
TMDX470MF066USB – TMS470M USB 棒套件

- USB 供电
- 板载 USB XDS100v2 JTAG 调试
- 板载 SCI 至 PC 串行通信
- 到选择信号引脚测试点的访问
- LED, 温度传感器 & 光传感器
- 加速计
- CAN 收发器



开发

TMDXRM48HDK – RM48 开发套件
TMDXRM46HDK – RM46 开发套件
TMDXRM42HDK – RM42 开发套件
TMDX570LS31HDK – TMS570 开发套件
TMDX470MF066HDK – TMS470M 开发套件

- 板载 USB XDS100v2 JTAG 调试
- 借助 JTAG 的外部高速仿真
- CAN 收发器
- LED, 温度传感器 & 光传感器
- 针对 ETM/RTP/DMM 的跟踪器 (TMS570 & RM48)
- RJ45 10/100 以太网 (TMS570 & RM48 & RM46)
- USB-A 主机接口 (RM48 & RM46)
- USB-B 器件接口 (RM48 & RM46)



每个套件中包括的软件:

- CCS studio IDE: C/C++ 编译器/连接器/调试器
- HALCoGen 外设驱动程序生成工具
- CCS 和 nowFlash 闪存编程工具
- HET IDE: 仿真器 & 汇编程序
- 带有项目/代码示例的 GUI 演示



60

Hercules™ 电机控制开发套件

DRV8301-RM48-KIT
DRV8301-RM46-KIT
DRV8301-LS31KIT
DRV8301-LS12KIT

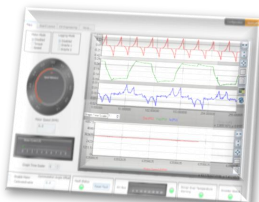
Hercules 安全 MCU
控制卡

具有编码器和霍尔传感器的
Teknic 7 Amp, 6000 RPM 电机

具有自我保护和可编程增益放大
器的 DRV 8301 60V, 60A EVM

起售价
\$499

演示软件



包括在套件中的软件

- Code Composer Studio v5
- 电机控制演示
- 具有 CMSIS DSP 库的 MotorWare™ 软件示例
- HALCoGen 驱动程序配置器
- 定时器协处理器工具 (HET IDE)
- 用户指南



61

Hercules™ 软件工具

IDE (编译器 & 调试器)



使用以下 IDE 的编程/调试代码:

- Code Composer Studio
- IAR Workbench
- KEIL μVision
- Green Hills MULTI IDE

RTOS 支持



Real Time Operating System Support:

- SAFERTOS: High Integrity Systems
- μC/OS: Micrium
- ThreadX: Express Logic
- SCIOPTA RTOS
- AUTOSAR: Vector Microsar and EB tresos

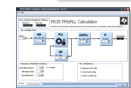


基于 GUI 的代码生成工具和其它软件工具



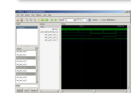
安全 MCU 演示

- 安全特性亮点
- 周围照明 & 温度演示
- LED 灯显示
- Maze 游戏
- 借助 CCS 的可视源代码



PLL 计算器

轻松配置 Hercules 平台锁相环模块内的
FMzPLL 和 FPLL。



HET IDE

- 图形化编程环境
- 输出仿真工具
- 生成 CCS 可用的软件模块
- 包括 TI 的功能示例



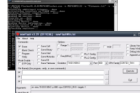
HALCoGen

- 高抽象级别上的用户输入
- 基于图形的代码生成
- 轻松配置
- 针对新项目的快速启动
- 支持 CCS, IAR & KEIL IDEs



now ECC ECC 生成工具

生成用于 Hercules 器件纠错码的命令行
程序。可与 CCS 一同使用



nowFlash 闪存编程工具

在不使用 IDE 的情况下, 将代码载入到器
件内的 GUI 和命令行编辑器。



62

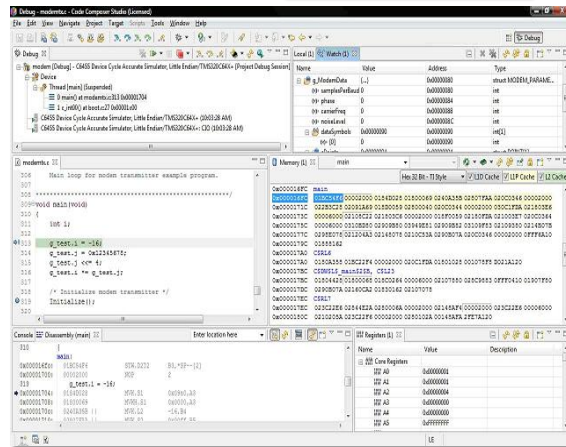
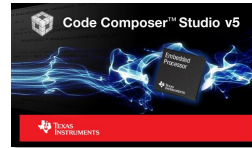
Code Composer Studio

• 基于针对嵌入式调试工具的 Eclipse 业界标准

- 现代窗口环境
- 先进的源代码编辑器
- 可扩展多内核/处理器环境
- 通过 JTAG 的编程和调试应用
- 借助脚本处理的测试自动化

• Hercules™ 调试特性

- 6 硬件断点
- 无限制的软件断点
- 集成闪存编程



63

Code Composer Studio 组件:

目标连接

- 源 & 目标文件
- 文件依赖关系
- 编译器、汇编程序 & 连接器搭建选项

源代码查看



反汇编窗口

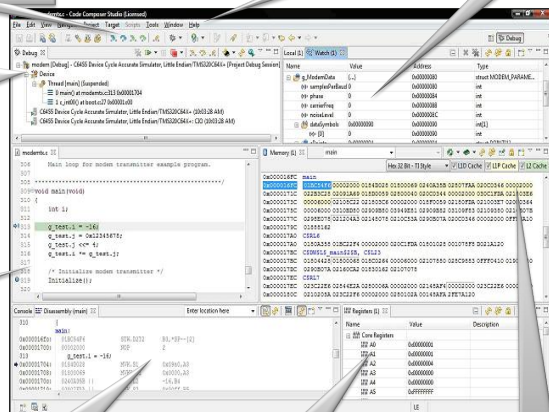
CPU 窗口

内存窗口

菜单和图标

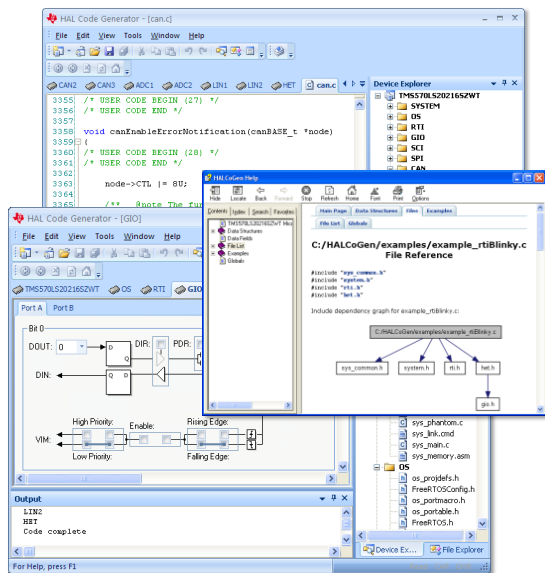
帮助

观察窗口



64

HALCoGen: 硬件抽象层代码生成器



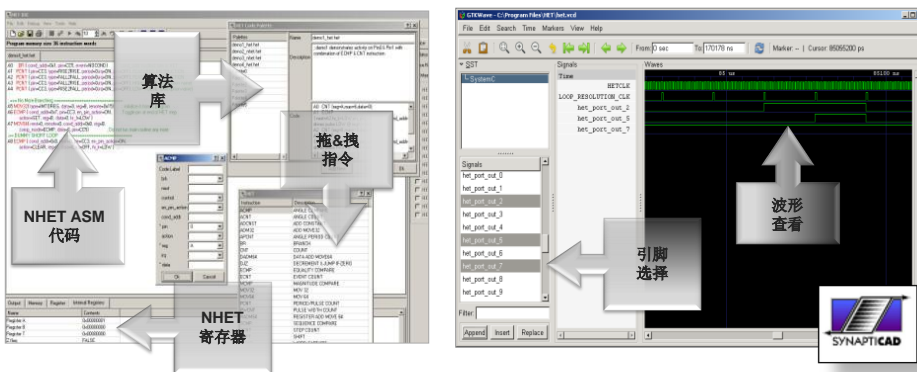
特性

- 高抽象级别上的用户中断
 - 基于图形的代码生成
 - 轻松配置
 - 针对新项目的快速启动
- 生成 C 源代码
 - 符合 ANSI
 - 清晰、结构化的编码风格
 - 用于用户维护的可定制代码
- 支持的外设
 - 系统模块
 - RTI
 - GIO
 - SCI/LIN
 - CAN
 - SPI
 - ADC
 - 定时器协处理器 (nHET)
- 交互式帮助系统
 - 描述了工具特性和功能
 - 提供了详细的依赖关系图
 - 提供有用的示例代码
 - 提供工具提示帮助
- 对 CCS, KEIL 和 IAR IDE 的本地支持



65

NHET 定时器协处理器开发工具



- 图形化编程环境
- 输出仿真工具
- 生成 CCS 可用软件模块
- 包括 TI 的功能示例

- 图形化波形查看器
- 输入生成工具
- 到编码工具的无缝接口
- 可升级至完全 SynapticAD



66

练习：Hercules 安全 MCU 演示



67

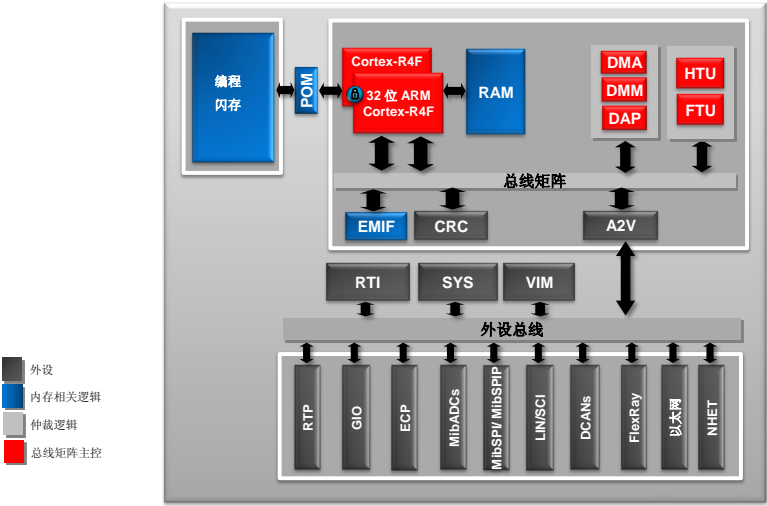
联系 1： Hercules™ 安全 MCU 演示



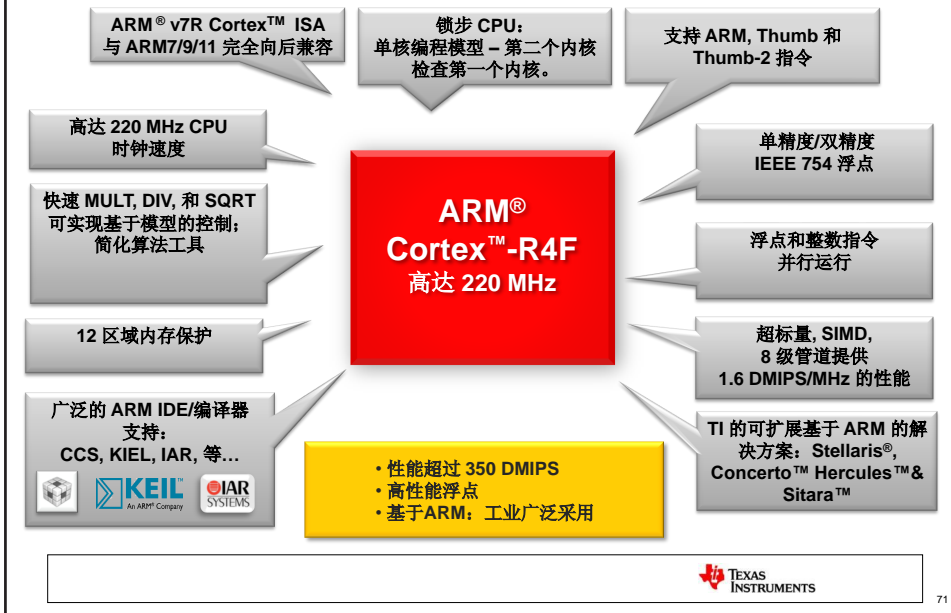
68

Hercules Cortex™-R4F 架构预览：
内存映射、计时和异常

Hercules™ 架构预览



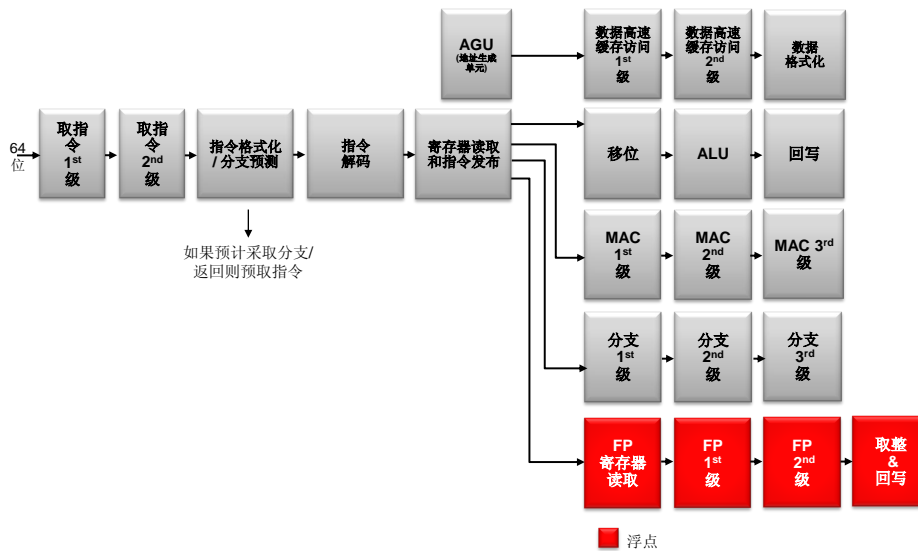
高性能 Cortex-R4F 浮点 CPU



Cortex-R4F 特性

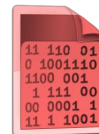
- 32 位 ARM 和 16/32 位 Thumb2 指令集
- 带有整体嵌入式 ICE-RT 逻辑的整数单元
- 使用全局历史缓冲器和返回堆栈的动态分支预测
- 浮点单元
- 低中断延迟
- 不可屏蔽中断
- Harvard 一级内存系统具有:
 - 紧密耦合内存 (TCM) 接口, 此类接口支持纠错或奇偶校验内存
 - 内存保护单元 (MPU)
- 二级内存接口:
 - 单一 64 位主控接口
 - 64 位受控接口, TCM RAM 块和高速缓存 RAM 块。
- 到 CoreSight® ETM-R4® 或 CoreSight DAP 的调试接口

Cortex-R4F 管道



数据类型

- 此处理器支持下列数据类型
 - 双字 (64 位)
 - 字 (32 位)
 - 半字 (16 位)
 - 字节 (8 位)
- 虽然处理器支持非对齐访问，TI 不推荐为了提高总线性能而使用非对齐访问
 - 上面的数据类型应该在它们各自的尺寸边界上对齐
 - 大多数非对齐访问被转换为多个对齐访问
- 由于内存接口的修改，TMS570 器件用变字大端序格式 (BE32) 存储它们的数据
- RM4 器件用小端序格式存储它们的数据 (LE)



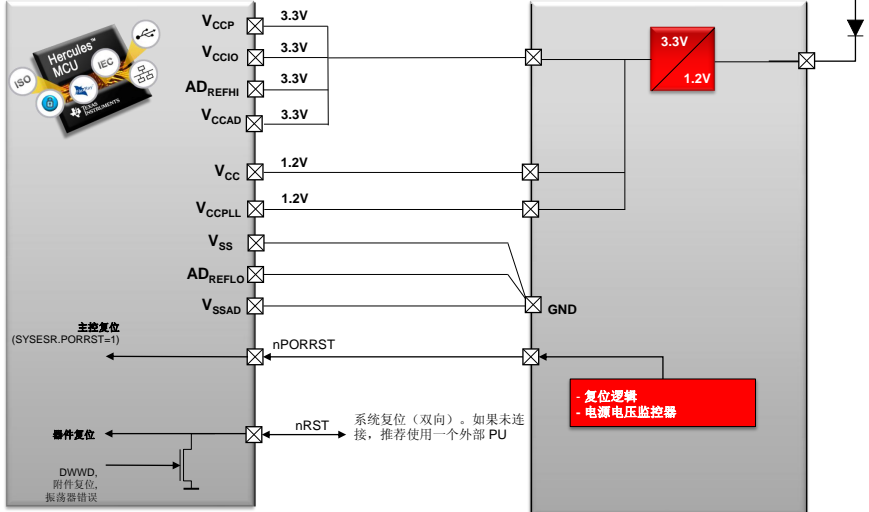
内存映射

- 缺省情况下，闪存开始地址为 0x00000000，CPU RAM 开始地址为 0x00000000
- 闪存被映射到 CPU TCM 的外部用于 ECC 诊断
- 缺省情况下，只允许从闪存、RAM 和外部异步内存中执行代码
- 缺省情况下，系统和外设控制寄存器的空间被定义为“严格排序”
- CPU 访问“被保留”区域会导致一个中止异常
- 在数据表中规定了已定义数据帧内到保留位置访问时

0xFFFFFFF	SYSTEM Modules	0xFFFF8000
0xFF000000	Peripherals - Frame 1	
0xFE000000	CRC	
0xFC000000	RESERVED	
0xFC000000	Peripherals - Frame 2	
0xF07FFFFF	RESERVED	
0xF0000000	Flash Wrapper Bus2 Interface (Flash ECC, OTP and EEPROM accesses)	
0x87FFFFF	RESERVED	
0x80000000	EMIF (128MB) SDRAM	
0x6FFFFF	RESERVED	
0x60000000	EMIF (16MB * 3) Async RAM	0x607FFFFF
0x202FFFFF	Flash (3MB) (Mirrored Image)	0x60000000
0x20000000	RESERVED	
0x0843FFFF	RAM - ECC	
0x08400000	RESERVED	
0x0803FFFF	RAM (256KB)	
0x08000000	RESERVED	
0x002FFFFF	Flash (3MB)	
0x00000000		

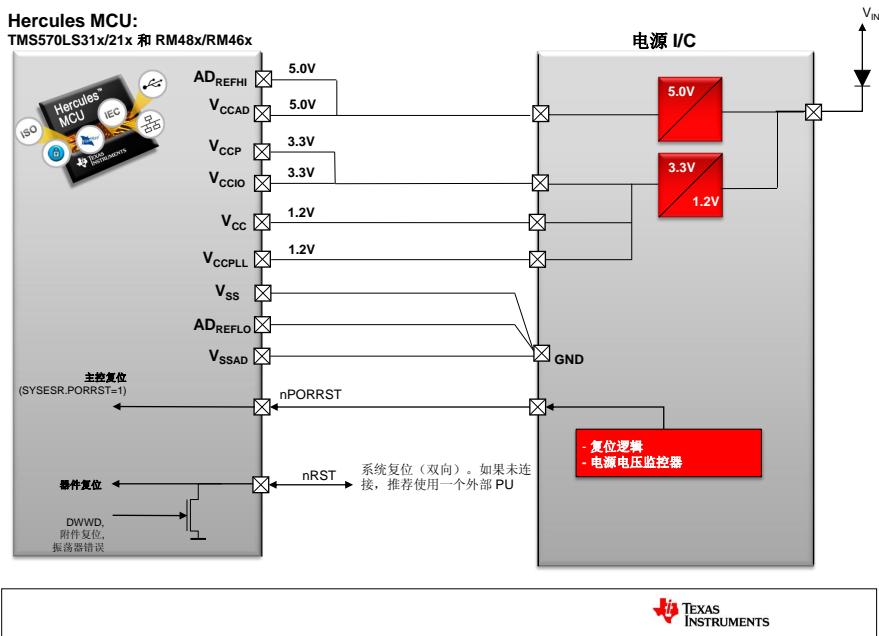
电源要求

Hercules MCU:
TMS570LS31x/21x 和 RM48x/46x



电源要求

Hercules MCU:
TMS570LS31x/21x 和 RM48x/RM46x

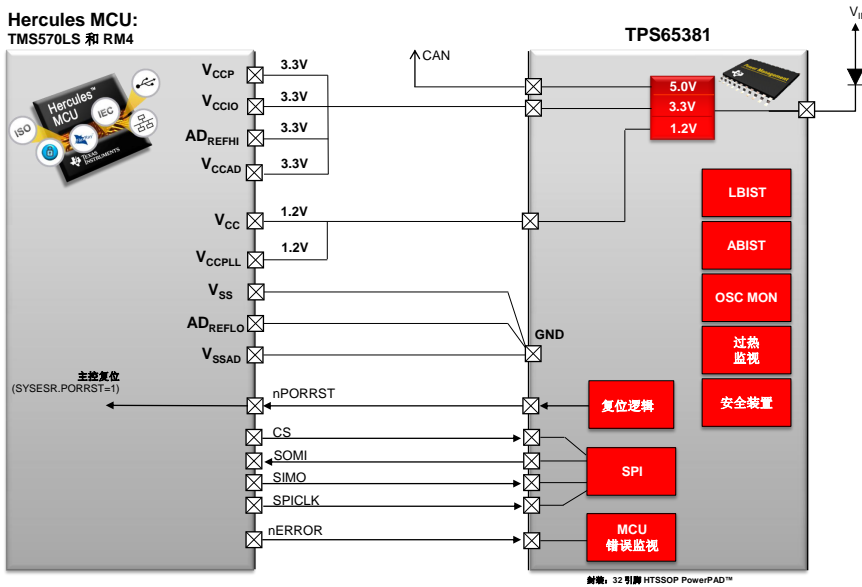


TEXAS
INSTRUMENTS

77

Hercules & TPS65381 安全伴侣 IC

Hercules MCU:
TMS570LS 和 RM4



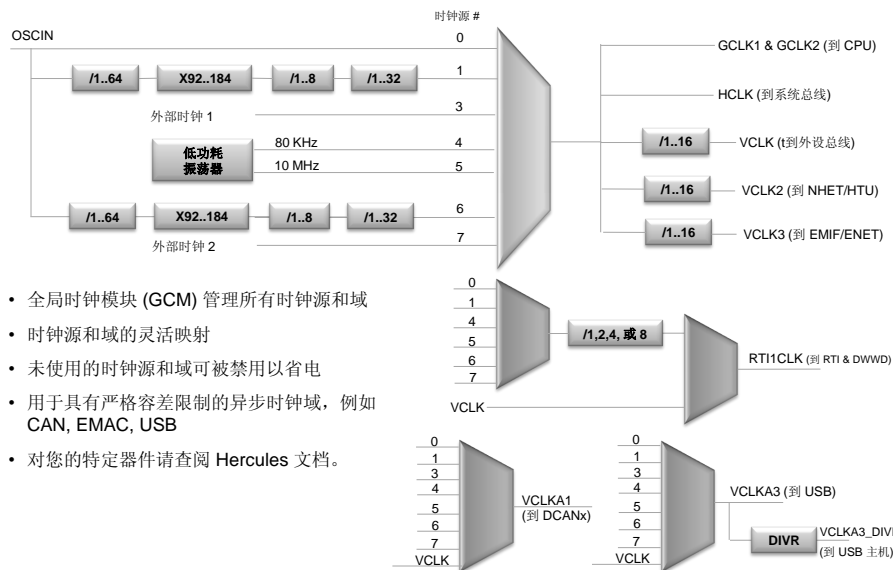
封装: 32 引脚 HTSSOP PowerPAD™

更多与 TPS65381 有关的信息请访问: <http://www.ti.com/product/tps65381-q1>

TEXAS
INSTRUMENTS

78

示例 Hercules 时钟源和域

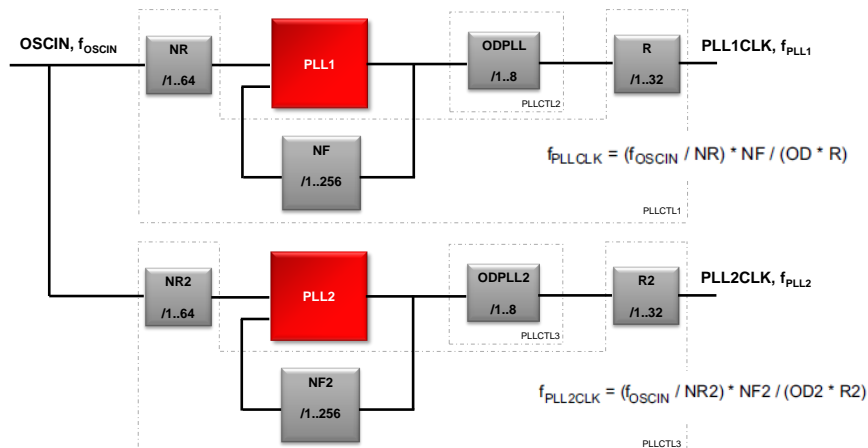


- 全局时钟模块 (GCM) 管理所有时钟源和域
- 时钟源和域的灵活映射
- 未使用的时钟源和域可被禁用以省电
- 用于具有严格容差限制的异步时钟域，例如 CAN, EMAC, USB
- 对您的特定器件请查阅 Hercules 文档。

TEXAS
INSTRUMENTS

79

锁相环 (PLL1 和 PLL2)



- PLLCTL1 和 PLLCTL2 寄存器来配置 PLL1
- 使用 PLLCTL3 寄存器来配置 PLL2
- TRM 描述了针对 PLL1 配置频率调制设置的过程
- 频率调制对 PLL2 不可用

TEXAS
INSTRUMENTS

80

复位资源

- 加电复位
 - 由外部电压监视器或者内部电压监视器置为有效
- 振荡器故障
 - 当由软件启用时由内部时钟监视器置为有效
- CPU 复位
 - 在 LBIST 操作完成后由 CPU 自检控制器置为有效
- 软件复位
 - 通过软件写入异常控制寄存器来置为有效
- 外部复位
 - 由外部电路通过驱动热启动 (nRST) 信号为低电平来置为有效
- 调试复位
 - 由 ICEPICK JTAG 模块置为有效



81

Hercules™: 闪存工具



82

nowECC

```
<return_value> nowECC [options] -i <input_file> [-o <output_file>]
```

- 为编辑闪存生成 ECC 数据
- 可执行命令行
- 返回值 = 0 表示运行期间无错误
 - 独立的错误代码以区分每一类型的错误
- 输入文件是唯一需要的参数
 - 可被扩展 Tektronix, Intel Hex, Motorola-S, COFF 或 ELF 格式
- 输出文件指定将被生成的输出文件的名称
 - 如果未指定名称, ECC 被加在指定输入文件的末尾



83

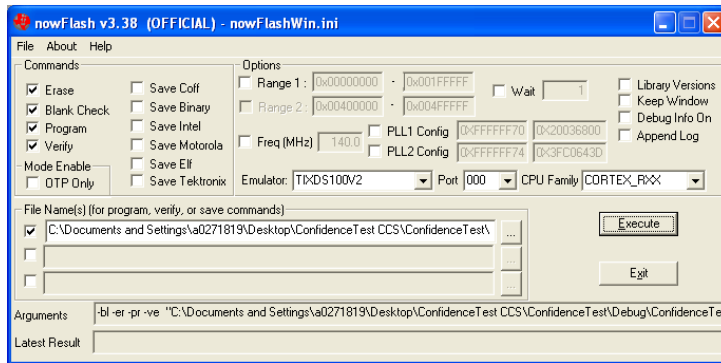
闪存编程选项

- 使用 nowFlash/Code Composer Studio v4.x 的片载编程
 - 要求 JTAG 连接
 - 仿真器支持:
 - Blackhawk BHUSB560M
 - Spectrum Digital XDS510PP, XDS510PP+, XDS510USB, XDS560RUSB
 - Signum JTAGjet
 - 德州仪器 (TI) SPI525, XDS100v2, XDS560
- 借助用户引导加载程序代码的板载编程
 - 必须使用德州仪器 (TI) 发布的 API 例程
 - 可使用多个通信接口
 - 有必要验证编程和擦除例程
- 板外编程
 - 单一器件或并发编程
 - 支持高度自动化



84

nowFlash 闪存编程工具

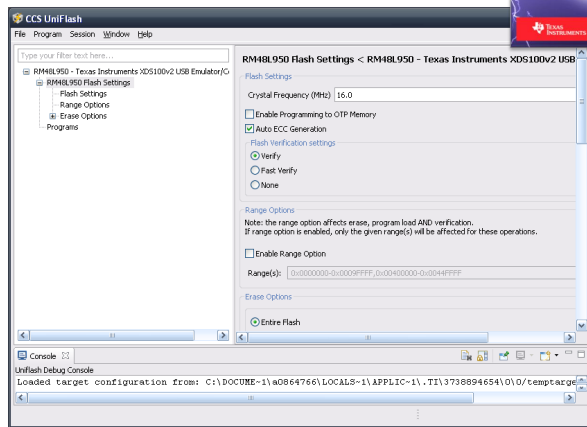


- 基于 PC 的软件工具：可执行的 GUI + 命令行
- 通过 JTAG 与微控制器通信
- 可被用来编辑、擦除、读取或验证闪存存储器
- 也支持 RAM 的定制代码的执行（只来自命令行）



85

UniFlash 闪存编程工具



- 基于 PC 的软件工具
- 通过 JTAG 与微控制器通信
- 可被用来编辑和擦除闪存存储器
- 基于 Eclipse – 支持 Windows 和 Linux

[UniFlash WIKI](#)



86

闪存应用编程接口 (API)

- 只作为目标库文件发布
- 支持片载 RAM 发出闪存操作
- 支持最大额定时钟频率上的运行
- 针对以下操作的库例程
 - 检查地址空间
 - 紧凑法
 - 擦除
 - 编程原点
 - 编程数据
 - 计算校验和
 - 验证
- 例程也管理 ECC



87

实时中断模块 (RTI)

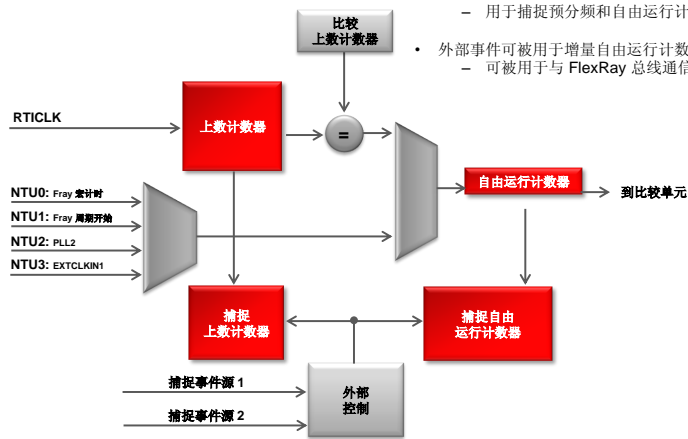


88

RTI: 计数器方框图

RTI 计数器特性:

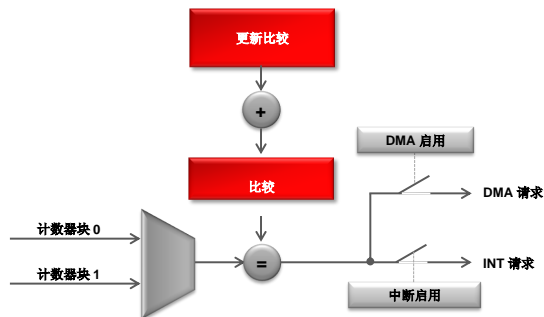
- 用于生成不同时基的两个独立计数器块
 - 每个块包括
 - 一个 32 位预分频计数器
 - 一个 32 位自由运行计数器
 - 用于捕捉预分频和自由运行计数器的捕捉寄存器
- 外部事件可被用于增量自由运行计数器 0
 - 可被用于与 FlexRay 总线通信周期或者外部系统时钟输入同步



RTI: 比较块方框图

RTI 比较特性:

- 四个比较中断和 DMA 请求
 - 每个可使用两个可用自由运行计数器中的任一个
 - 比较值自动更新以大大减少 CPU 介入
 - 可选择生成 DMA 请求以及比较中断
- 两个计数器-溢出中断
 - 当一个自由运行计数器溢出并且归零时生成

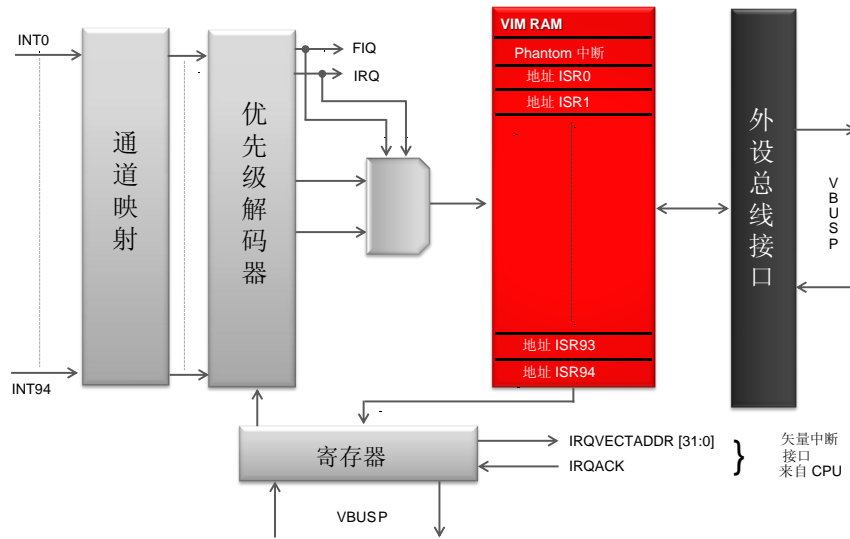


矢量中断管理器 (VIM)

VIM：主要特性

- **VIM 硬件**
 - 到 ARM CPU 的专用矢量中断接口
 - IRQ 矢量地址的硬件重定向
 - 针对优先级排序和控制中断源的硬件辅助
- **VIM 功能**
 - 96 个中断请求
 - 通过编程将中断请求映射到中断通道。
 - 通过中断请求映射来提供可编程优先级
 - 对连接至 CPU 的中断通道进行优先级排序
 - 将中断处理例程 (ISR) 的地址提供给 CPU
- **VIM 模式**
 - 传统 ARM7 模式 (FIQ/IRQ)
 - 矢量中断 (FIQ/IRQ)
 - 硬件矢量中断 (只适用于 IRQ)

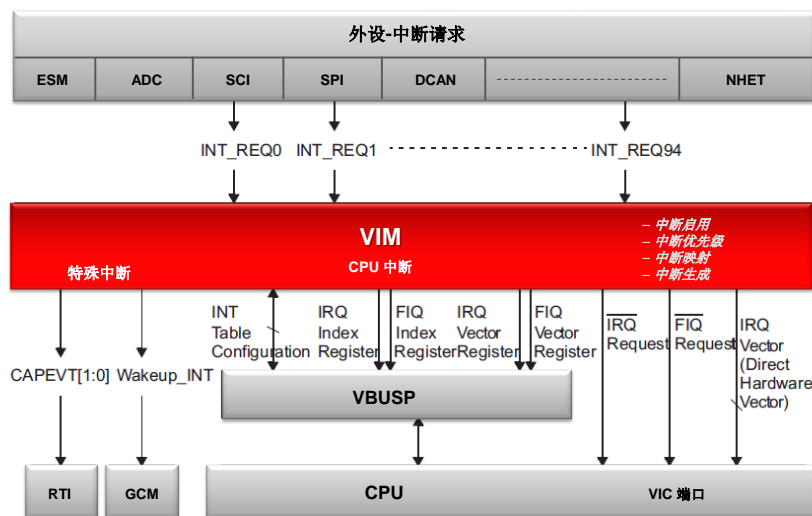
VIM:



TEXAS
INSTRUMENTS

93

VIM: 连接方框图



TEXAS
INSTRUMENTS

94

直接内存访问 (DMA)

DMA: 主要特性

- 可单独启用的 32 个通道
- 64 个 DMA 请求
 - 软件和硬件 DMA 请求 (事件同步)
- 支持 8, 16, 32 或 64 位处理
- 针对源/目的的多寻址模式
 - 固定的、增量的、加索引的
- 自动启动
- 通道链接功能
- 1 个 FIFO (先进先出)
- 一个与总线矩阵对接的 AHB 到主控端口 (64 位宽)
- 一个与针对寄存器接口的 VBUS 对接的受控端口
- 针对 DMA 可访问的地址范围的内存保护

DMA: 如何开始一个传输?

- 软件请求
 - 通过设置 SWCHENAS [31:0] 寄存器内的位 x, 传输 (通道 x) 将被触发。
- 硬件请求
 - 一个有效 DMA 请求信号将触发一个 DMA 事务。
 - 可处理多达 64 条 DMAREQ 线路。
 - 由于 DMA 控制器被 HCLK 锁定, 所有 DMA 请求信号的持续时间必须至少达到 HCLK 的长度。
- 由其它控制数据包触发
 - 当一个控制数据包完成已设定数量的传输时, 它能够触发另外一个通道来启动传输。



97

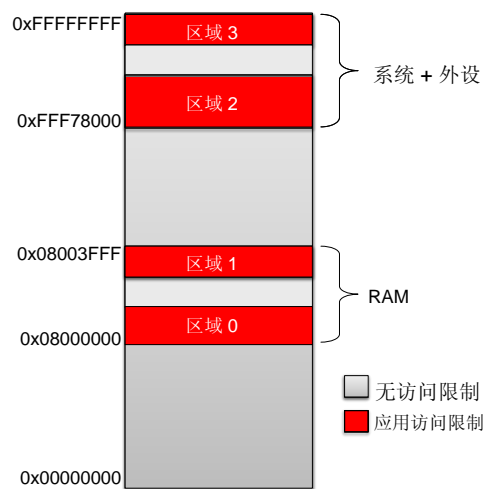
DMA: 通道中断

- 每个通道可被配置成在几个传输条件时生成中断:
 - FTC (数据帧传输完成) 中断
 - LFS (最后一个数据帧传输开始) 中断
 - HBC (头半个块完成) 中断
 - BTC (块传输完成) 中断
 - BER (总线错误) 中断



98

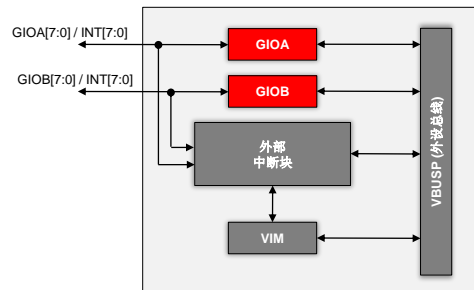
内存保护



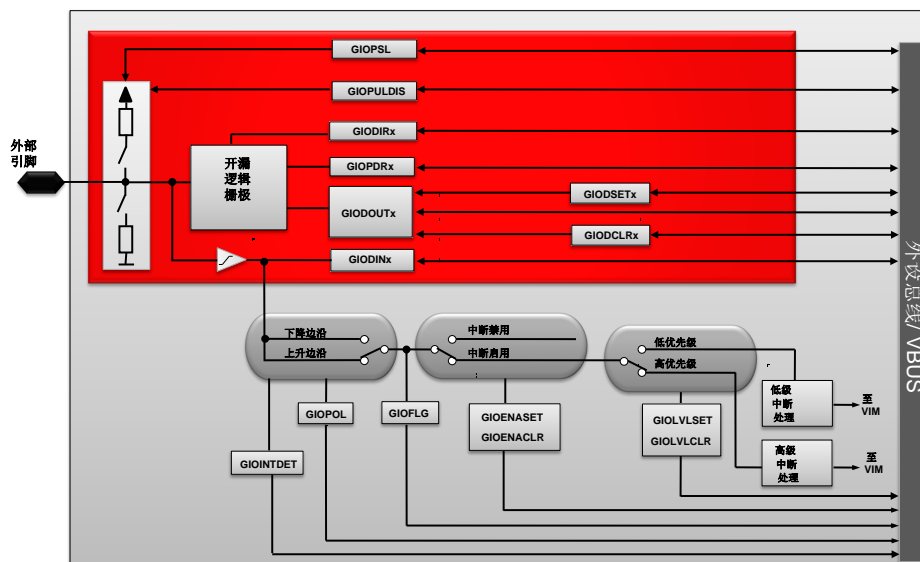
通用 I/O (GPIO)

GIO 主要特性

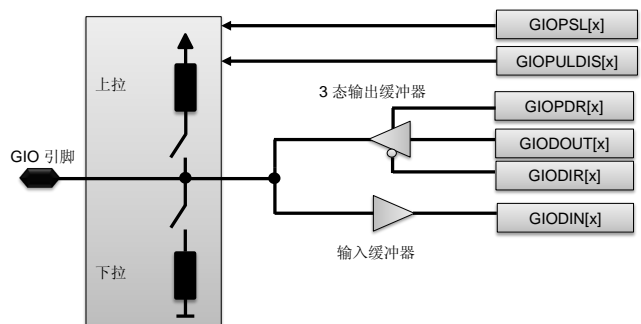
- 两个端口 (GIOA/B), 每个端口有 8 个双向和位可编程 I/O 引脚
- 外部中断功能
 - 单边沿或双边沿上的可编程中断检测
 - 可编程边沿检测极性
 - 可编程中断优先级
- 可能的引脚配置:
 - 数据方向
 - 数据输入/输出
 - 数据设置/清零
 - 开漏
 - 上拉/下拉



GIO 方框图



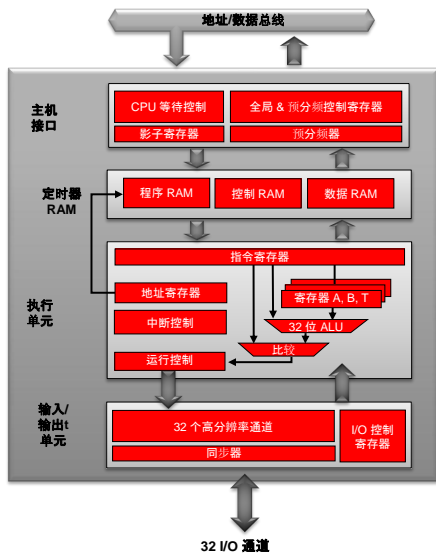
GIO 单元配置



寄存器	说明
GIOPSL	在引脚上选择拉动类型 (上拉/下拉)
GIOPULDIS	禁用引脚上的拉动控制功能
GIOPDR	控制引脚的开漏配置
GIODOUT	当配置位输出时, 控制发送到外部引脚的信息
GIODIN	从外部引脚接收信息
GIODIR	控制引脚的方向 (输入/ 输出)

高端定时器 (N2HET)

高端定时器 (NHET)



- 用户可编程定时协处理器
- 使用低 CPU 开销来提供高级和复杂定时功能
- 具有奇偶校验保护的 128 字指令
- 将数据从 NHET 传输到数据 RAM (写入或输出 CPU) 的专用 DMA 功能性 (HTU)
- 基于引脚条件和比较的有条件程序执行
- 针对复杂和传统定时功能，诸如比较、捕捉，PWM，GPIO 等的 32 个输入/输出 (I/O) 通道 (引脚)
- 抑制滤波器消除了有害的输入频率
- 多个用于定时器的 25 位虚拟计数器、事件计数器和模拟计数器
- 由于环路针对多重分辨率功能执行的高分辨率 I/O 和粗糙分辨率

NHET: 应用示例

脉宽调制

- 单通道/多通道 PWM
- 支持同步/异步占空比更新的 PWM
- 支持同步定期更新的 PWM
- 使用 RADM64 指令的相移 PWM

其它特性

- 调频输出
- 脉宽计数 (使用 PWCNT)
- 时间戳 (使用 WCAP)
- 事件计数器 (使用 ECNT)
- 脉冲累加器示例 (使用 ECNT)
- 多分辨率机制

频率和脉冲测量

- 脉宽和周期测量 (使用 PCNT)
- 使用处于 HR 模式中的 PCNT，Hrshare 特性和“自动读取/清零”位被置位的 64 位读取访问进行的定期测量

NHET: 命令行汇编程序

- 调用 NHET 汇编程序 (hetp.exe): `hetp [options] input file`
- 选项:
 - c32 为 TMS570 代码生成工具产生一个包含汇编程序指令的输出文件
 - hc32 生成一个 C 语言文件和一个头文件。(与 -nx 选项一起使用)
 - nx 在器件上指定 x-th HET 模块 (与 -hc32 选项一起使用)
 - l (小写 L) 生成一个与扩展名为 .lst 的文件具有同样文件名的列表文件。
 - x 生成一个对照表并将其放在列表文件的末尾。
- 示例: `hetp -hc32 -n0 pwm.het`
- 输入: `pwm.het` 包含 HET 程序的汇编源代码
- 输出: `pwm.c` 提供一个 C 阵列, 此阵列包含 HET 程序操作代码
- `pwm.h` 提供一个 C 语言结构, 这样可实现其它 C 语言代码到 NHET 字段的简单访问



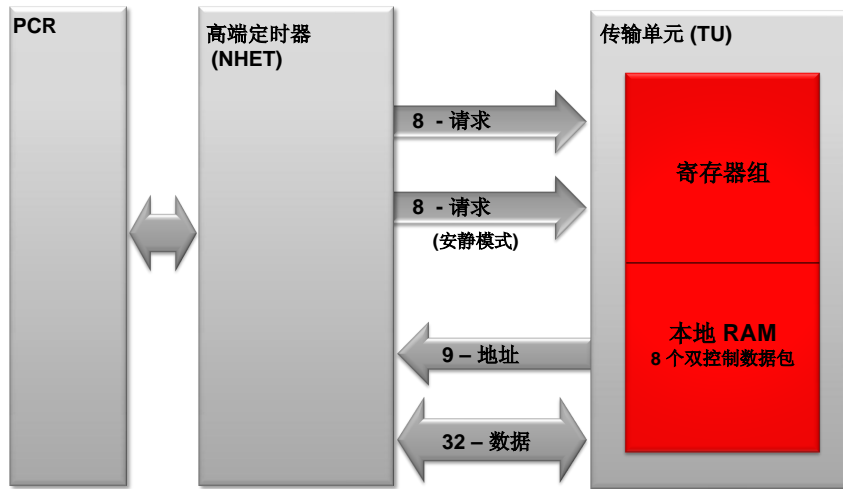
107

高端定时器传输单元 (HTU)



108

HTU: 方框图



HTU: 主要特性

- CPU 和 DMA 独立
- 主控端口直接访问系统内存
- HTU 主控访问受到专用内存保护单元的保护
- 一个与针对寄存器接口的 VBUS 进行对接的受控端口
- 最多 8 个支持双缓冲配置的双控制数据包
- 控制数据包信息被存储在受奇偶校验保护的 RAM 中
- 事件同步 (HET 传输请求)
- 支持 32 或 64 位事务处理
- 针对 HET 地址 (8 字节或 16 字节) 以及系统内存地址 (固定, 32 字节或 64 字节)
- 每种类型的中断可被路由至两个不同主机 CPU 中的任何一个
- 单次、循环和自动缓冲器传输模式
- 请求丢失检测

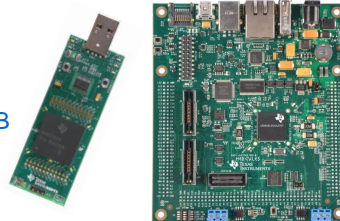
练习 2: 使用 N2HET 的 PWM 生成



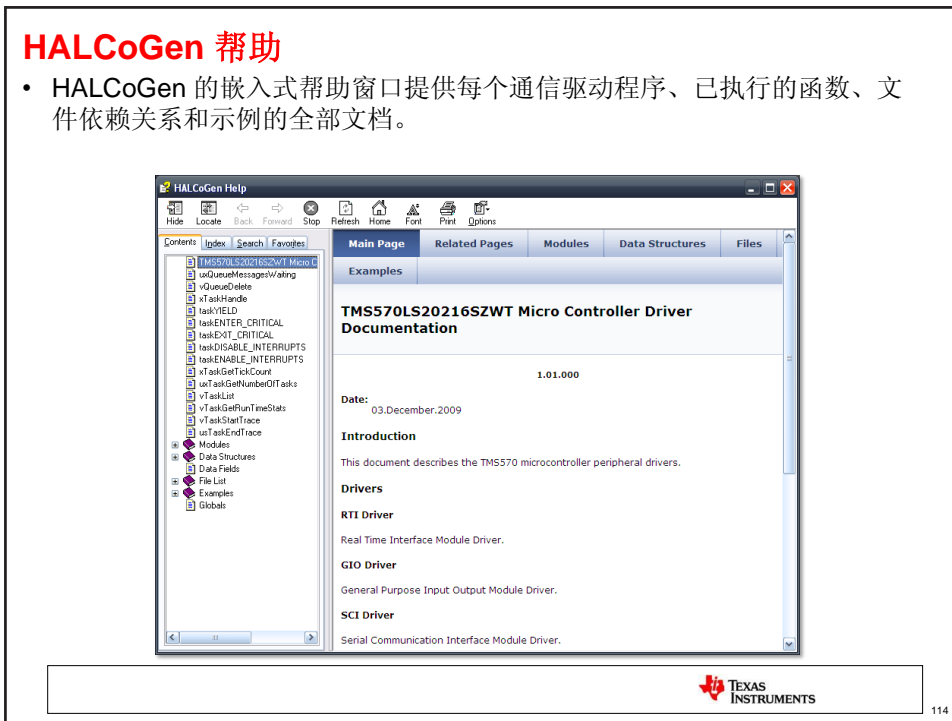
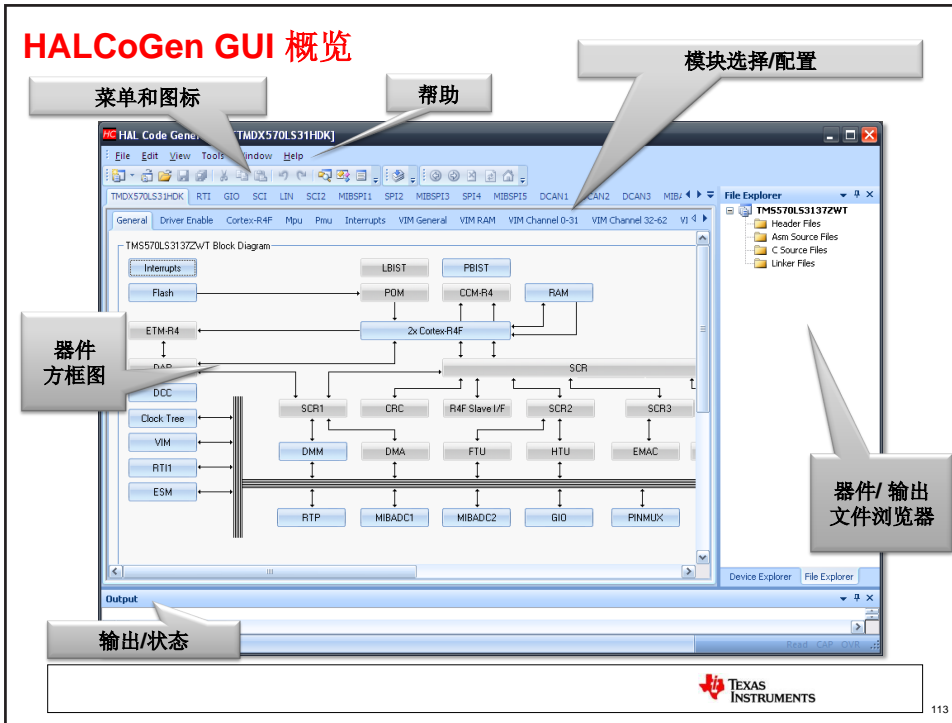
111

概述

- 在这个练习中我们将:
 - 创建一个新的 HALCoGen 项目
 - 配置 HALCoGen 以生成
 - 具有 1 秒周期和 75% 占空比的基本 PWM
 - 具有 0.5 秒周期和 50% 占空比的基本 PWM
 - 使用 PWM 来触发板上的 NHET1[0] 和 NHET1[17] LED
 - 生成并将代码导出至 Code Composer Studio
 - 搭建并将我们的代码部署至微控制器
- 所需硬件:
 - 基于Windows 的 PC (WinXP, Vista, 7)
 - TMS570LS31HDK/USB 或 RM48HDK/USB
- 所需软件:
 - [HALCoGen](#)
 - [Code Composer Studio](#)



112



HALCoGen 文件依赖关系和函数列表

文件浏览器

文件信息控制

文件依赖关系图

函数列表

TEXAS INSTRUMENTS

建立一个新的 HALCoGen 项目

- 启动 HALCoGen:



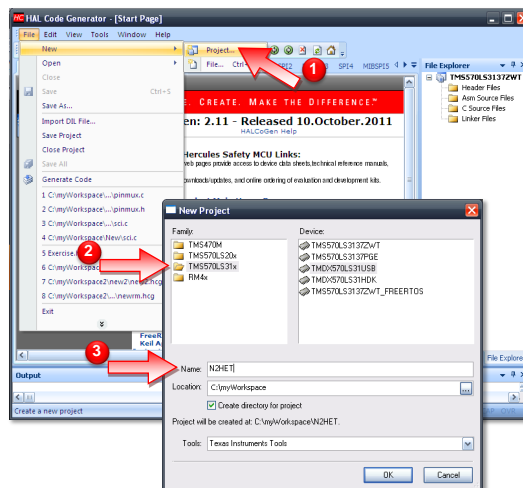
→ Programs → Texas Instruments → Hercules → HALCoGen

- 创建一个新项目:
 - File → New → Project

- 对于 TMS570 套件:
 - 选择系列: TMDX570LS31x
 - 器件: TMDX570LS31USB

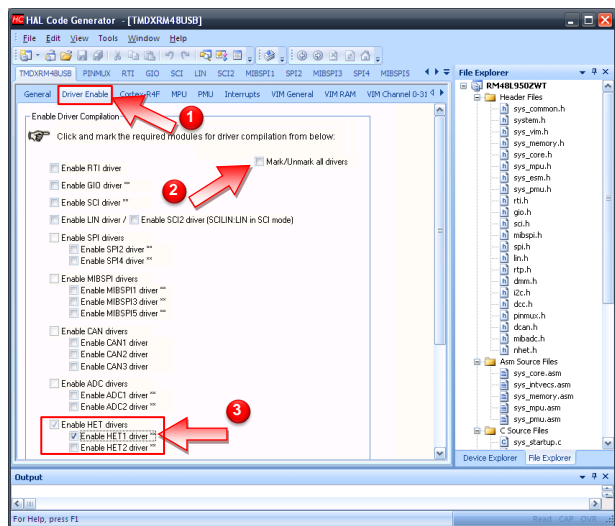
- 对于 RM48 套件:
 - 选择系列: TMDXRM48
 - 器件: TMDXRM48USB

- 然后定义一个名称: 'N2HET'
- 位置: "C:\myWorkspace"



驱动程序启用

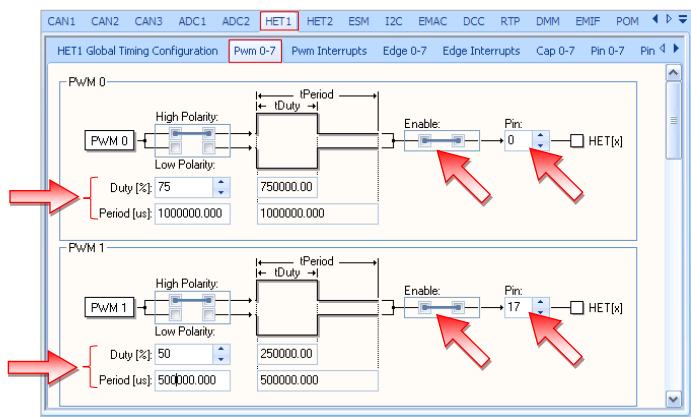
- 在 'Driver Enable' 标签页中启用 HET1 驱动程序。



117

N2HET PWM 配置

- 在 'Driver Enable' 标签页中启用 NHET1 驱动程序。
- 在 'HET1' → 'PWM 0-7' 标签页中:
 - 将 PWM 0 配置为 75% 占空比, 引脚 0 上的周期为 1000000.00uS
 - 将 PWM 1 配置为 50% 占空比, 引脚 17 上的周期为 500000.00uS

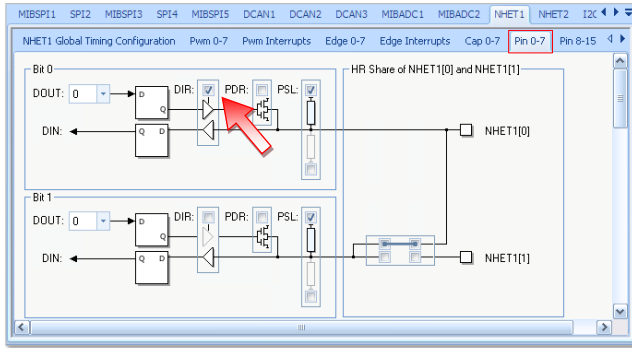


118

N2HET 输出配置



- 在 'HET1' → 'Pin 0-7' 标签页中:
 - 确保输出引脚 0 上的输出被启用
- 在 'HET1' → 'Pin 16-23' 标签页中:
 - 确保输出引脚 17 上的输出被启用



- 生成代码: File → Generate Code

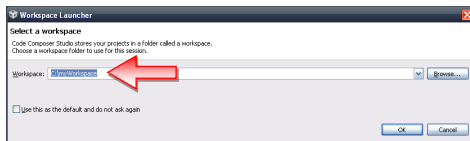


119

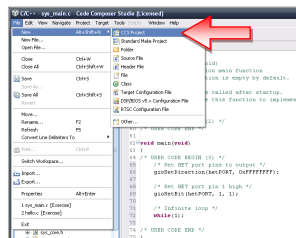
设置 Code Composer Studio



- 启动 Code Composer Studio (CCS)
 - start → Programs → Texas Instruments → Code Composer Studio v5
 - Code Composer Studio v5
- 启动时, CCS 将请您选择一个工作站, 我们选择 "C:\myWorkspace"



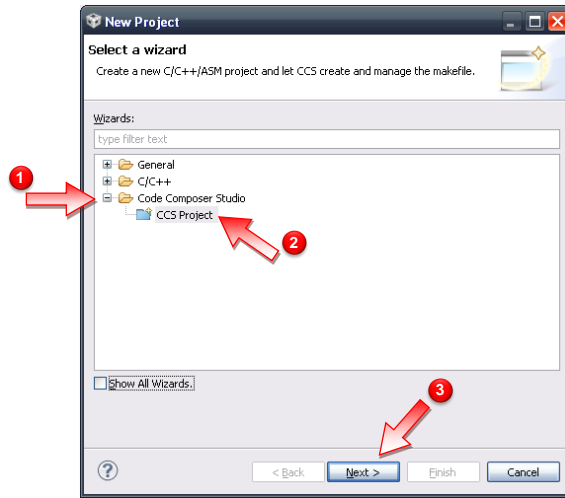
- 一旦 CCS 加载, 则指向
File → New → CCS Project



120

建立我们的项目

- 从新项目窗口，展开 ‘Code Composer Studio’ 文件夹。
- 选择 ‘CCS Project’
- 点击 ‘Next’



121

建立我们的项目

- 我们项目的名称需要与我们的 HALCoGen 项目名称，‘Exercise’，相匹配
- 确保您的项目 ‘Family’ 被置入 ARM

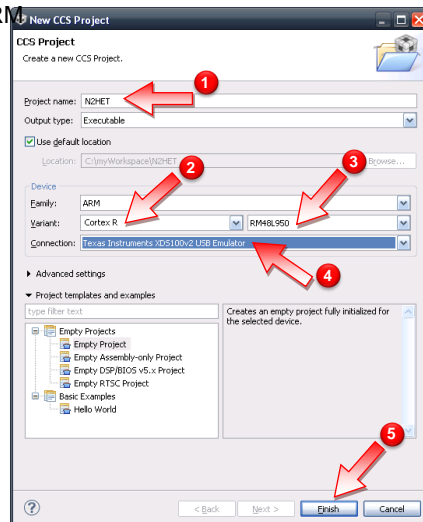
- 下一步，将变量设置为 “Cortex R”

- 对于 TMS570 套件：
 - 选择: TMS570LS3137

- 对于 RM48 套件：
 - 选择: RM48L950

- 然后设定到德州仪器 (TI) XDS100v2 的 ‘Connection’

- 然后点击 ‘Finish’



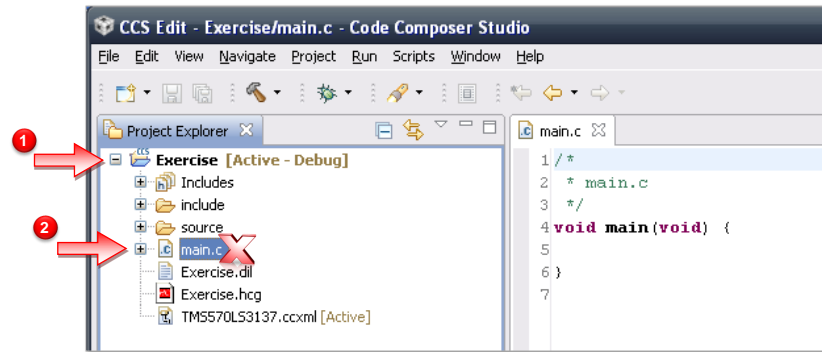
122

建立我们的项目



- 首先展开项目并从 CCS “Project Explorer” 中删除 ‘main.c’ 文件

注意： 当一个新项目被创建时，CCS 将自动创建一个名称为 ‘main.c’ 的文件。由于我们在使用 HALCoGen 为我们创建的代码，我们将不需要这个文件。



123

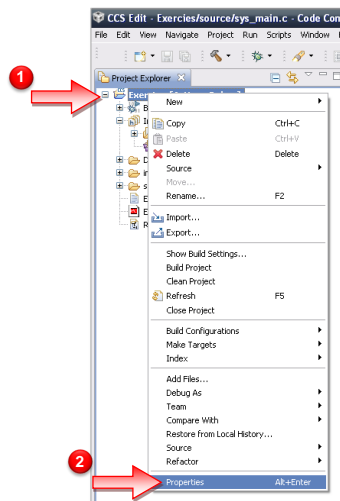
建立我们的项目



- 下一步需要将我们的 ‘include’ 文件夹从 CCS ‘Project Explorer’ 添加到项目

- 右键点击项目浏览器中的 ‘Exercise’ 项目

- 然后选择 ‘Properties’



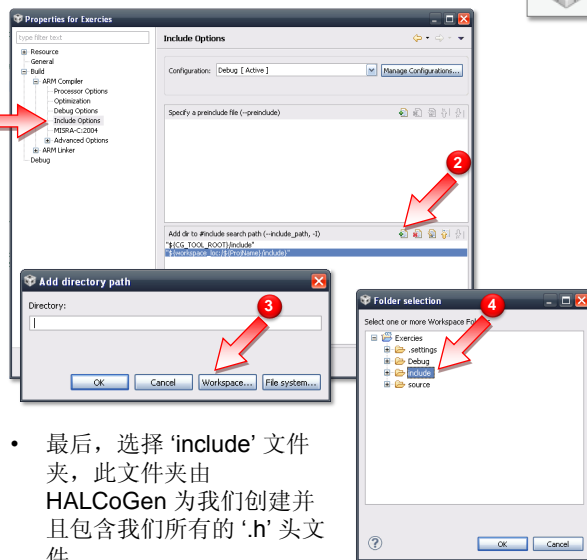
124

建立我们的项目

- 然后在 'Properties' 窗口中展开 'Build -> ARM Compiler' 类别并选择 'Include Options'

- 然后选择 '+' 按钮来将我们的 '.h' 头文件添加到目录

- 在 'Add directory path' 窗口中, 点击 'Workspace...' 按钮



- 最后, 选择 'include' 文件夹, 此文件夹由 HALCoGen 为我们创建并且包含我们所有的 '.h' 头文件

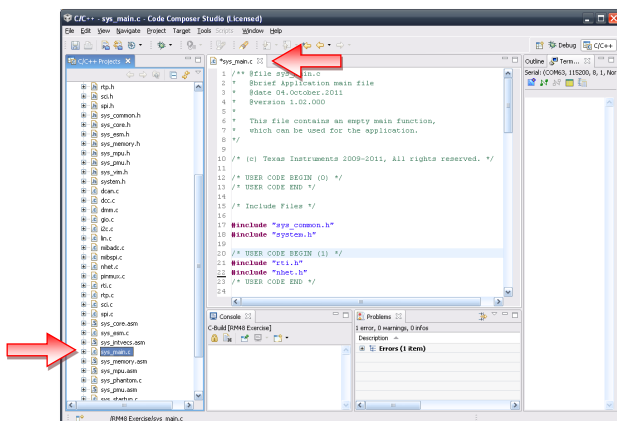


125

将代码输入到 CCS 项目

- 展开项目并在 CCS "Project Explorer" 中打开 "sys_main.c" 文件

注意: 由于 HALCoGen 和 CCS 项目的名称一样, 所有由 HALCoGen 生成的启动和驱动程序文件将被自动包含在 CCS 项目中。使用 "sys_main.c" 文件作为您的主程序代码。



126

Code Composer Studio



- 在 Code Composer Project 中并输入以下代码：
 - 在用户代码 1 内，插入以下代码。

```
/* USER CODE BEGIN (1) */  
#include "het.h"  
/* USER CODE END */
```

- 然后在用户代码 3 内，插入以下代码。

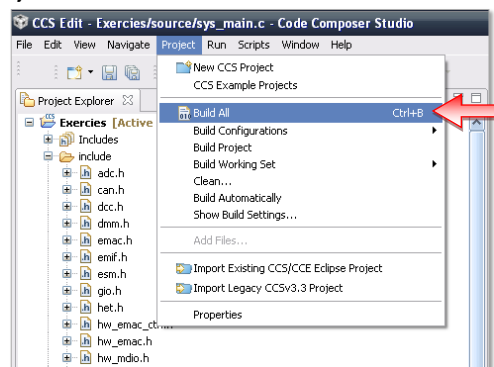
```
/* USER CODE BEGIN (3) */  
hetInit();  
while(1);  
/* USER CODE END */
```



127

编译项目

- 现在代码已完成，我们着手建立我们的项目。
 - 指向 Project → Build All



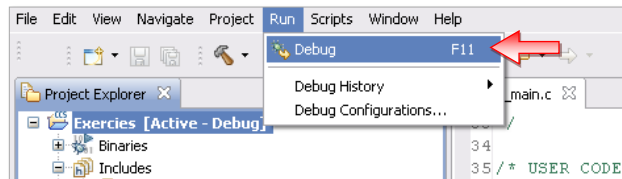
- 现在我们有 .out 文件，我们需要编辑微控制器闪存存储器。



128

编辑闪存

- 我们现在准备编辑闪存。
 - 指向 **Run → Debug**
 - 当它编辑闪存存储器时，应出现一个新窗口。
 - 这将花费一些时间。

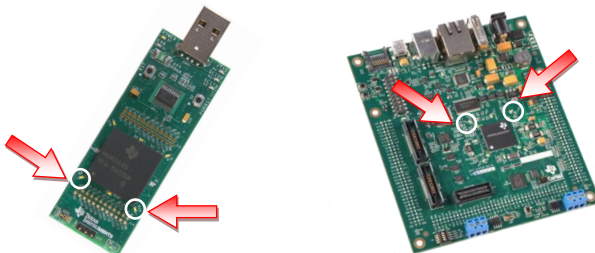


测试我们的程序

- 点击调试标签页上的绿色箭头来运行我们的程序



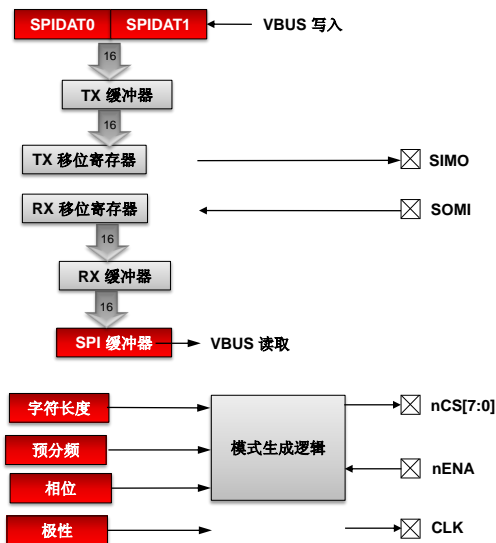
- 程序也可在调试器未连接时运行
- 点击调试标签页上的红色方框来中止调试器连接
- 按下板上的复位按钮并观察 N2HET LED 的状态



- 恭喜您！您已经完成了这个练习。

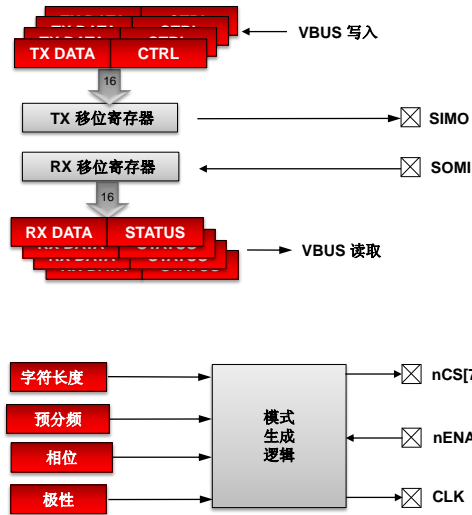
多缓冲串行外设接口 (MibSPI)

SPI – 方框图 & 特性



- 16 位移位寄存器
- 双缓冲 TX 和 RX
- 主控或受控模式
- 多达 4 个并行 SIMO / SOMI
- 可选 MSbit 或 LSbit 第一次传输
- 未使用引脚可用作 GP I/O
- CLK 频率 VCLK/2 至 VCLK/256
- 2 至 16 位字符长度
- 可选 CLK 相位和极性
- 中断 / DMA 请求当以下情况发生时
 - TX 缓冲器空
 - RX 缓冲器满

MibSPI – 方框图，特性

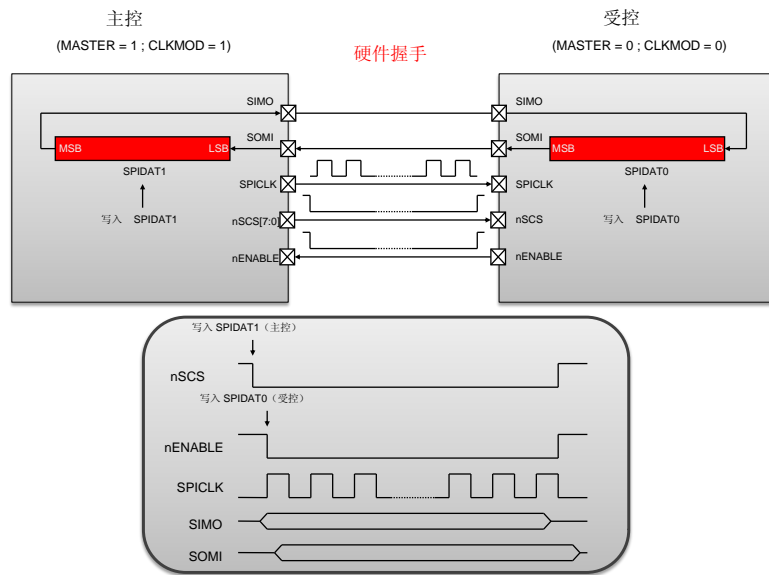


- 16 位移位寄存器
- 多达 128 个用于 TX 和 RX 的寄存器
- 多达 8 个传输组
- 15 个源以触发传输
- 受奇偶校验保护的内存
- 主控或受控模式
- 多达 4 个并行 SIMO / SOMI
- 可选 MSbit 或 LSbit 第一次传输
- 未使用引脚可用作 GP I/O
- CLK 频率 VCLK/2 至 VCLK/256
- 2 至 16 位字符长度
- 可选 CLK 相位和极性
- 可编程中断和 DMA 请求生成条件
- 多达 16 个 DMA 请求

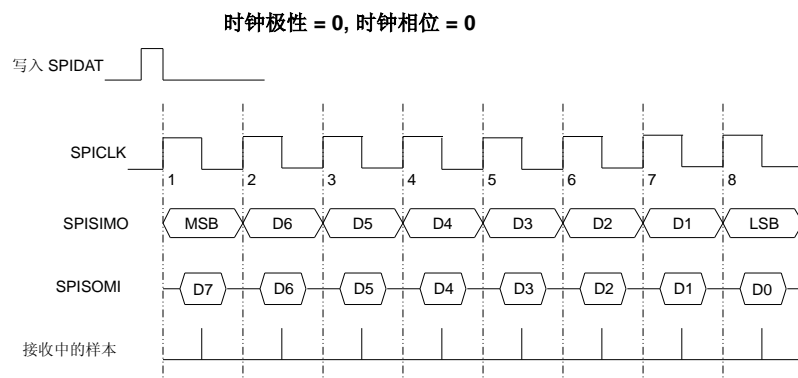
SPI / MibSPI 安全特性

- 对所有读取自 MibSPI RAM 的数据进行奇偶校验错误检测
- 持续监视主控和受控模式中已发送的数据进行
- 受控去同步检测 (只适用于主控模式)
- 针对非响应受控的超时 (只适用于主控模式)
- 接收器溢出中断条件以防止数据丢失
- 数据长度不匹配的检测

传输模式 – 5 引脚选项



时钟选项

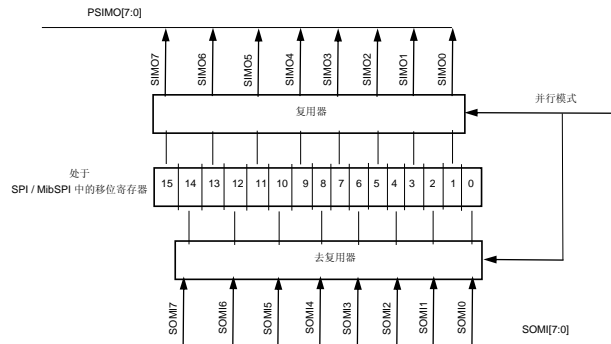


时钟相位 = 0 (SPICLK 无延迟)

- 数据是 SPICLK 上升边沿的输出
- 输入数据被存储在 SPICLK 的下降边沿上
- 一个到 SPIDAT 寄存器的写入会启动 SPICLK

SPI / MibSPI 并行模式

- 为了实现更高数据流，SPI / MibSPI 的并行模式使得模块能够在多于一条数据线路上发送数据 (并行 2, 或 4)。
- 支持移位寄存器最高有效位 (MSB) 优先的并行模式：

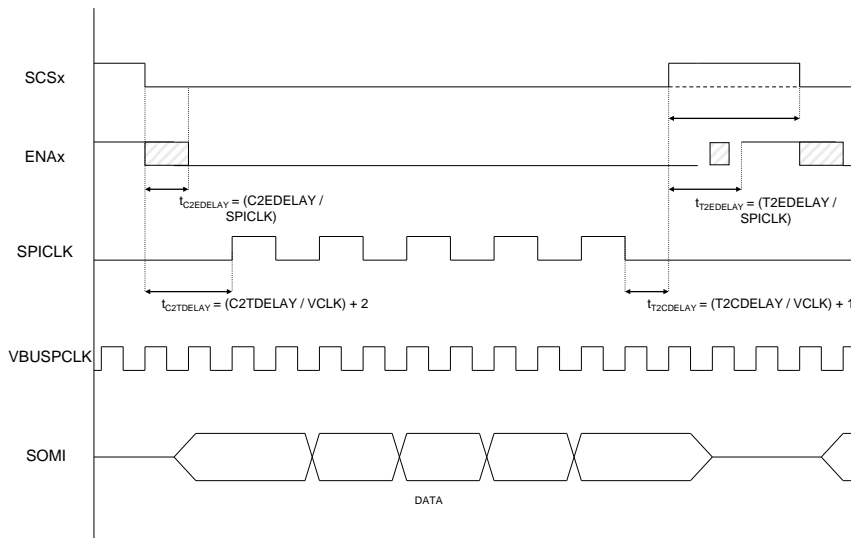


- 请注意：
 - 使用并行模式时，必须将数据长度设定为 16 位
 - 如果奇偶校验被启用，一个额外的 SPICLK 将触发奇偶校验位传输



137

时序设置 – 延迟寄存器 (SPIDELAY)



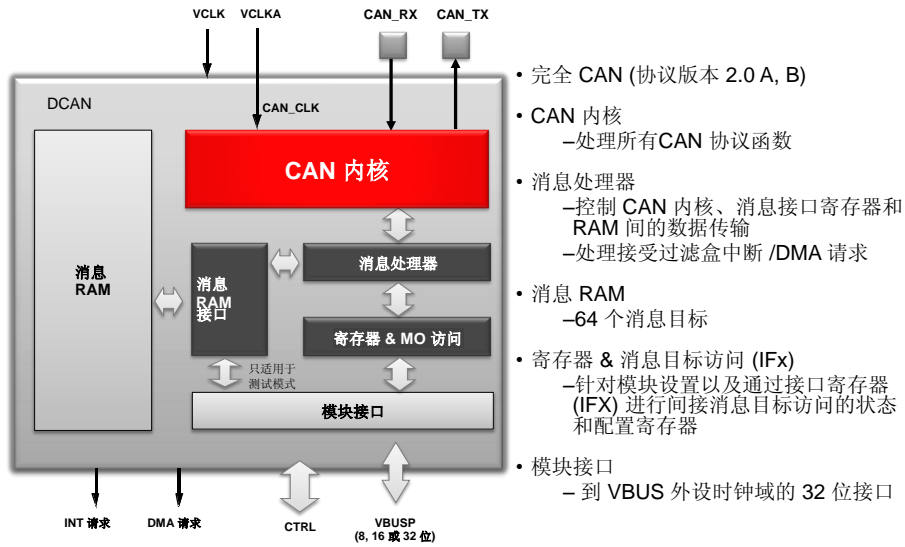
138

控制器局域网 (DCAN)

DCAN 特性概述

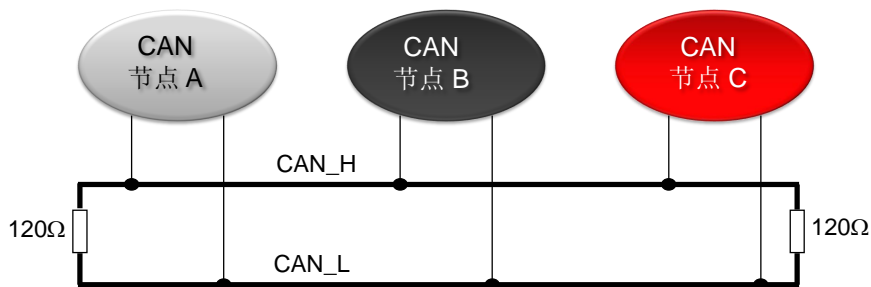
- 遵照协议版本 2.0 部分 A, B 的完全 CAN
- 标准和扩展标识符
- 可编程位时序, 位速率高达 1Mb/s
- 多达 128 消息目标 (MO)
- 针对每个消息目标的标识符屏蔽
- 针对消息目标的可编程 FIFO 模式
- 双时钟功能
- 在发生仲裁丢失或错误时可自动重传数据帧
- 总线诊断: 总线关闭、总线错误认可、总线错误警告、总线 胶着支配
- 数据帧错误报告: CRC, 堵塞, 结构, 位和确认错误
- 针对自检操作的可编程回路模式
- 针对调试支持的挂起模式
- 针对所有 RAM 模块的奇偶校验机制

DCAN 方框图 & 特性



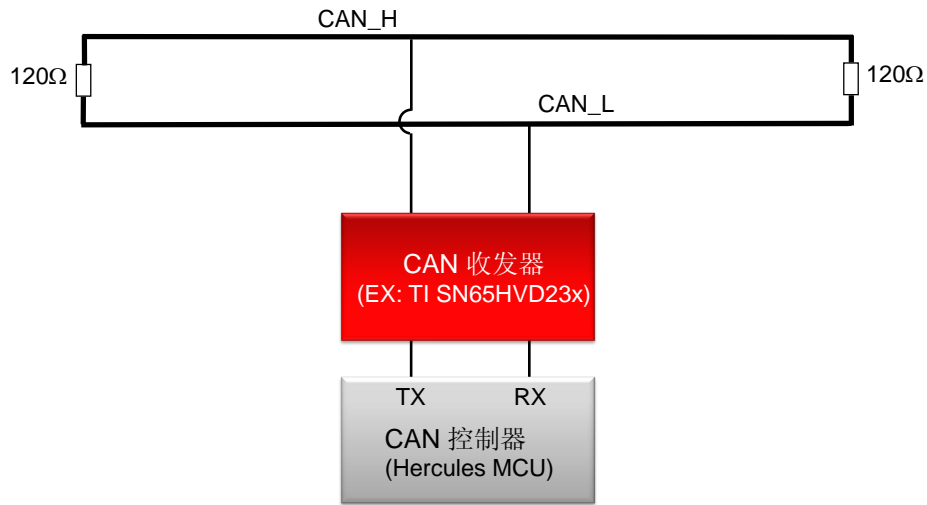
CAN 总线

- 两线制差分总线（通常为双绞线）
- 最大总线长度取决于传输速率
 - 40 米 @ 1 Mbps



CAN 节点

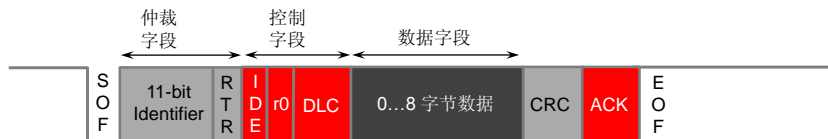
接线和总线连接



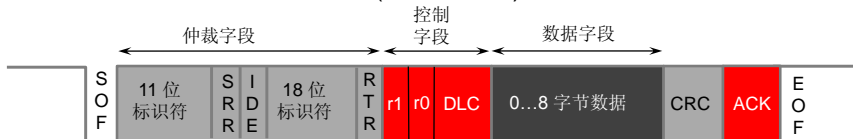
CAN 消息格式

- 使用消息帧来发送和接收数据
- 每条消息 8 字节数据有效载荷
- 标准和扩展标识符格式

标准帧：11 位标识符 (CAN v2.0A)



扩展帧：29 位标识符 (CAN v2.0B)



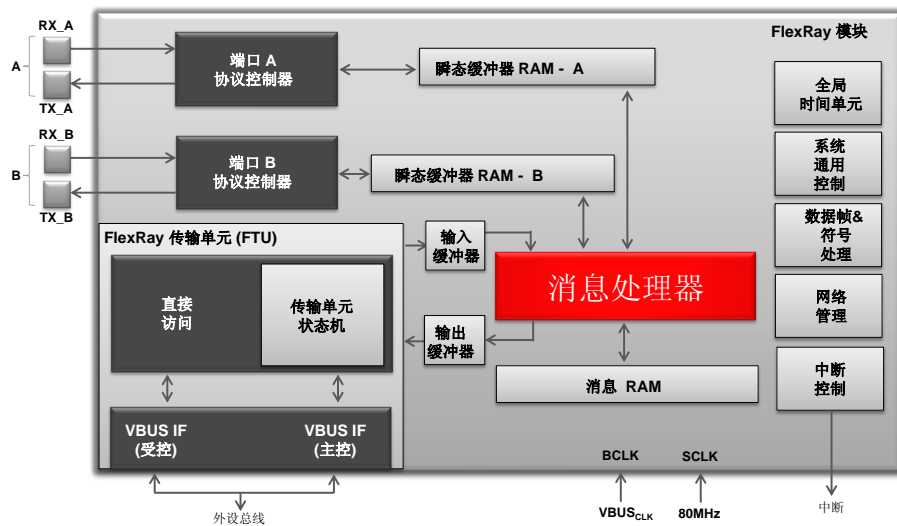
FlexRay / 传输单元 (只适用于 TMS570)

FlexRay 特性概述



- 开总线系统
- 支持冗余传输通道
- 数据速率 20Mb/s (每通道 10Mb/s)
- 支持同步容错全局时基同步
- 静态和动态数据传输 (可扩展)
 - 确定的数据传输
 - 无仲裁传输
- 硬件中执行的容错和时间触发的处理
- 支持光纤和电气物理层

FlexRay 方框图



TMS570LS 上的 FlexRay (1/2)

- Bosch FlexRay 内核 (E-Ray)
- 符合 FlexRay 协议规范 v2.1
- 2 个通道的每个通道上高达 10 Mb/s 的数据传输速率
- 8 字节消息 RAM，用于存储
 - 128 个消息缓冲器，数据部分最大 48 字节或者
 - 30 个消息缓冲器，数据部分 254 字节
 - 可支持不同的有效载荷
- 消息 RAM 的奇偶校验保护
- 消息处理器控制
 - 消息 RAM 访问仲裁
 - 接受过滤
 - 保持传输进度安排
 - 提供状态信息

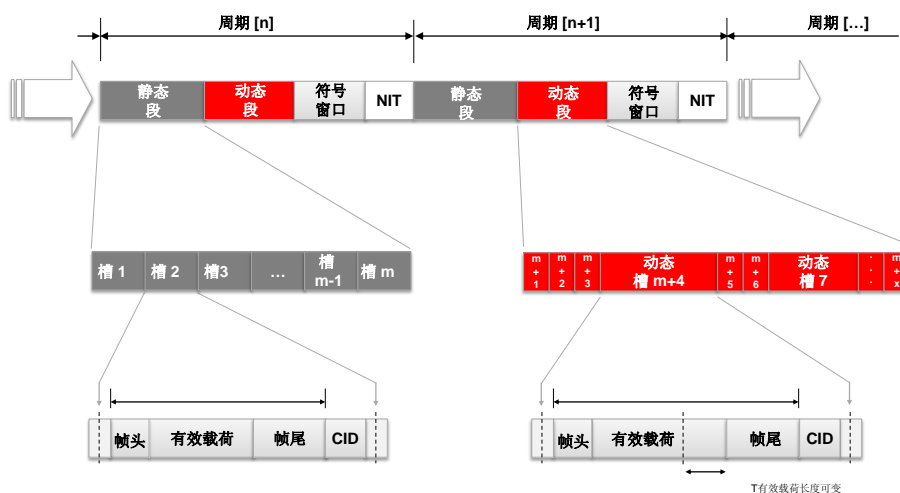
TMS570LS 上的 FlexRay (2/2)

- 每个消息缓冲器可被配置为
 - 接收缓冲器
 - 发送缓冲器
- 每个消息缓冲器可被分配为
 - 通信周期的静态分段
 - 通信周期的动态分段
 - 一个接收 FIFO 的部分
- 通过输入和输出缓冲器到消息缓冲器的直接 CPU 访问
- 针对与消息缓冲器进行自动数据交换的专用传输单元 (类似于 DMA)，而无需 CPU 干预
- 针对数据帧 ID，通道 ID 和周期计数器的过滤
- 可屏蔽模块中断
- 支持网络管理



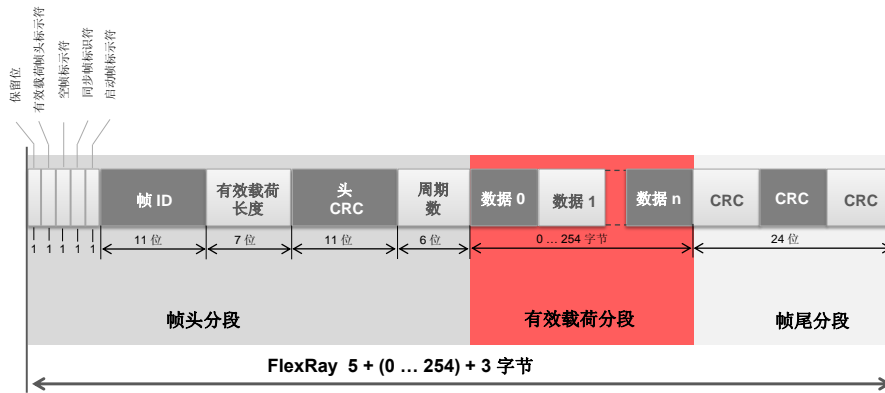
149

FlexRay 通信结构

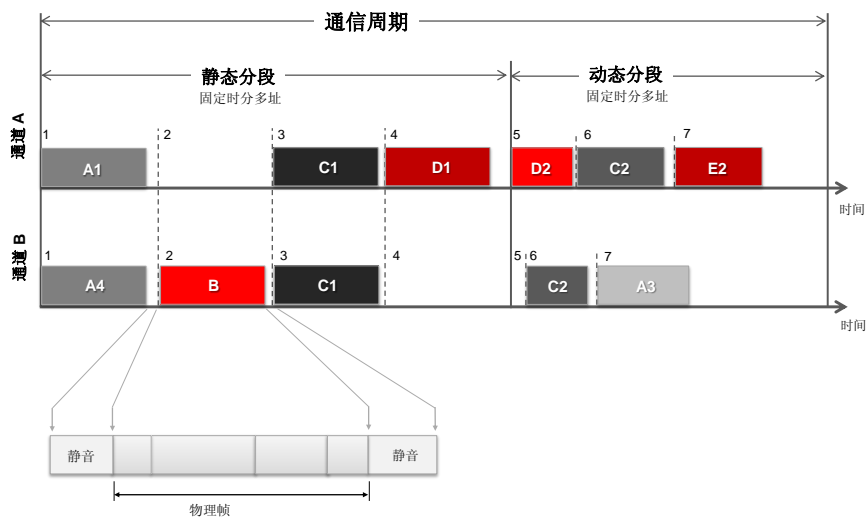


150

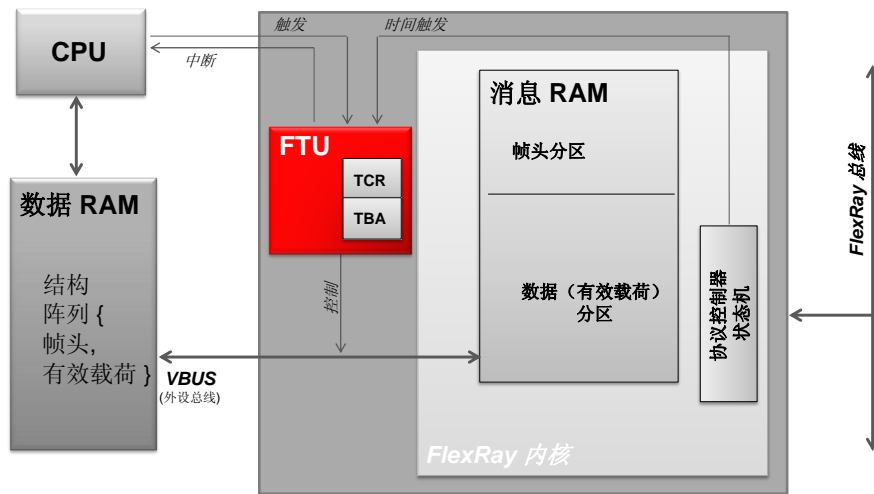
FlexRay 消息帧格式



FlexRay 通信周期



FTU 数据传输机制



FlexRay 传输单元关键特性

- 无需 CPU 干预的数据传输
 - 从 FlexRay 消息 RAM 到数据 RAM (读取)
 - 从数据 RAM 到 FlexRay 消息 RAM (写入)
- 传输类型
 - 数据和帧头部分
 - 只传帧头部分
 - 只传数据部分
- 传输配置 RAM (支持奇偶校验)
 - 配置传输序列
 - 奇偶校验保护
- 触发启动一个传输
 - 由 CPU 驱动 (单一传输序列)
 - 由事件驱动 (单一或连续传输序列)

FlexRay 传输单元关键特性...

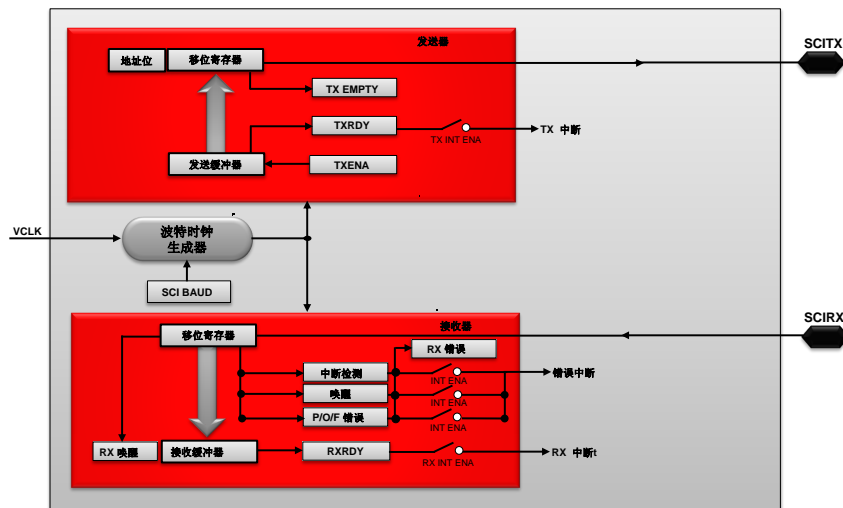
- 不同传输条件
 - 如果消息缓冲器的状态标志 (帧头部分) 已经被分别更新
 - 如果消息缓冲器的数据部分已经被分别更新
 - 保持不变
- 当消息缓冲器传输完成时, 可屏蔽中断生成
- 内存保护单元
 - 可定义一个内存段 (起始和终止地址)
 - 复位后, 不设置内存段

串行通信接口 (SCI/UART/LIN)

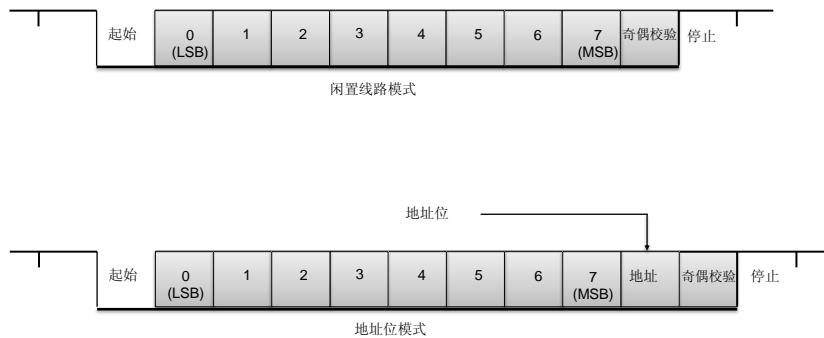
SCI 特性

- 可编程数据帧格式
 - 1 起始位
 - 1 至 8 数据位
 - 0 或 1 地址位
 - 0 或 1 奇偶校验位
 - 1 或 2 停止位
- 异步通信格式
- 支持唤醒功能的 2 个多处理器模式
 - 闲置线路模式; 地址位模式
- 可编程波特率
 - 多于 16700000 个不同的波特率
 - 100MHz VCLK 时的最大值为 3.125Mbps
- 错误检测
 - 奇偶校验、撤销和组帧错误
 - 中断检测
- 噪声保护功能
- 双缓冲接收和发送功能

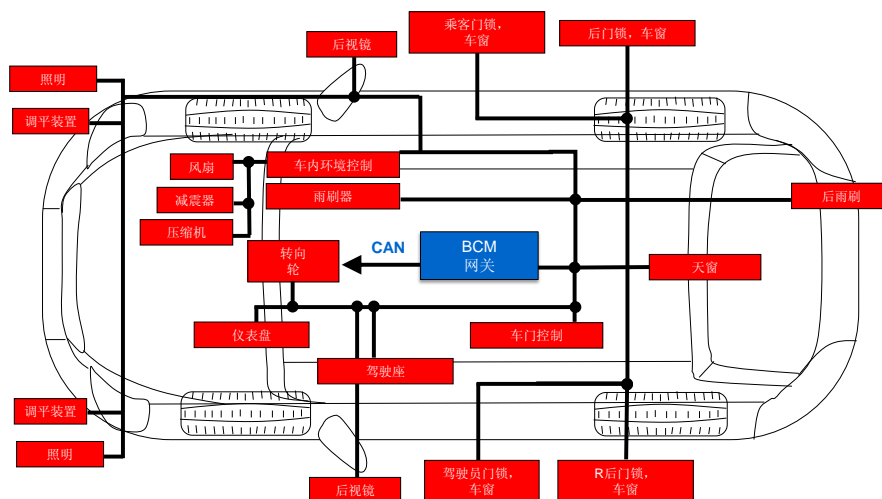
SCI 方框图



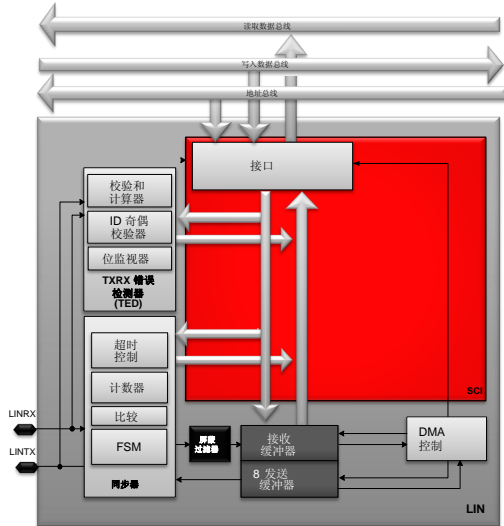
SCI 帧格式



典型 LIN 应用 (TMS570)



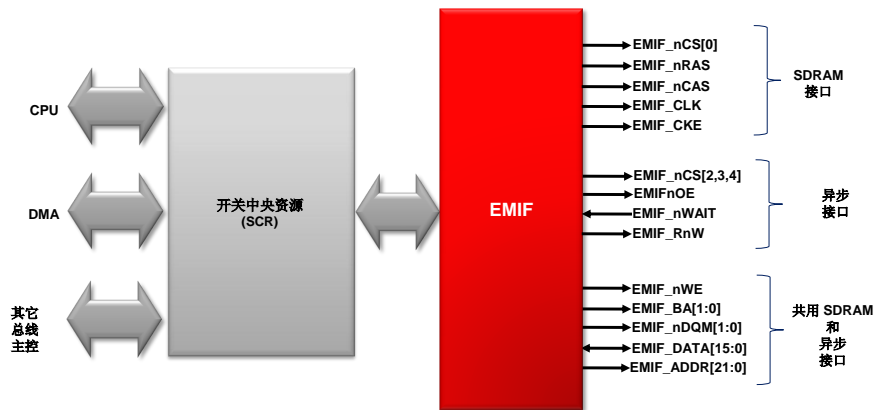
LIN 关键特性



- 与 LIN 1.3 或 2.0 兼容
- LIN 2.0 主控兼容
- 硬件 LIN 协议处理器
 - 多缓冲接收和发送单元
 - 自动校验和生成和验证
 - 针对消息过滤的 ID 屏蔽
 - DMA 功能
- 同步中断检测
- 受控自动同步
- 可选波特率更新
- 同步验证
- 自动位监视
- 自动错误检测
- SCI (UART) 模式

外部内存接口 (EMIF) / 参数覆盖模块 (POM)

EMIF: 方框图



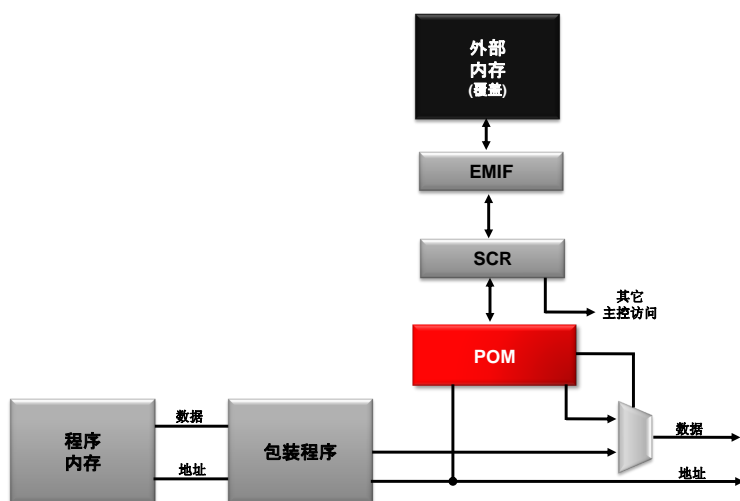
EMIF: 主要特性

- 异步内存支持
 - 到 SRAM 内存以及 NOR 闪存存储器的接口
 - 22 条地址线路, 3 个芯片选择, 每个高达 16MB
 - 16 位数据总线宽度
 - 可编程周期时序
 - 选择选通模式选项
 - 具有可编程超时周期的扩展等待模式
 - 数据总线驻留
- 同步 DRAM 内存支持
 - 一组、两组和四组 SDRAM 器件
 - 22 条地址线路, 1 个芯片选择
 - 具有 8 列, 9 列, 10 列和 11 列地址的器件
 - 两个或三个时钟周期的 CAS 延迟
 - 16 位数据总线宽度
 - 3.3V LVCMOS 接口
 - 支持 SDRAM 自动刷新和省电模式

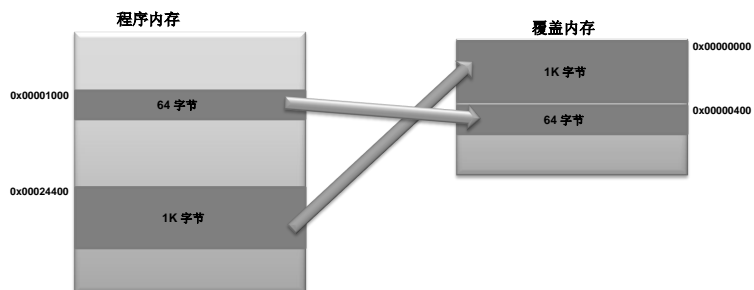
参数覆盖模块：主要特性

- 到内部/外部内存接口的重定向程序内存访问 (覆盖)
- 提供高达 32 个可编程内存区域以替代非易失性内存
 - 可编程区域起始地址
 - 可编程区域尺寸 (64 字节到 256k 字节 (以 2 的幂为步长))
- 高达 4M 字节外部覆盖内存
 - 22 位起始地址尺寸
- 覆盖内存为被映射的内存
 - 可由任一主控写入 (例如 CPU, DMA, DMM 等)

POM: 方框图

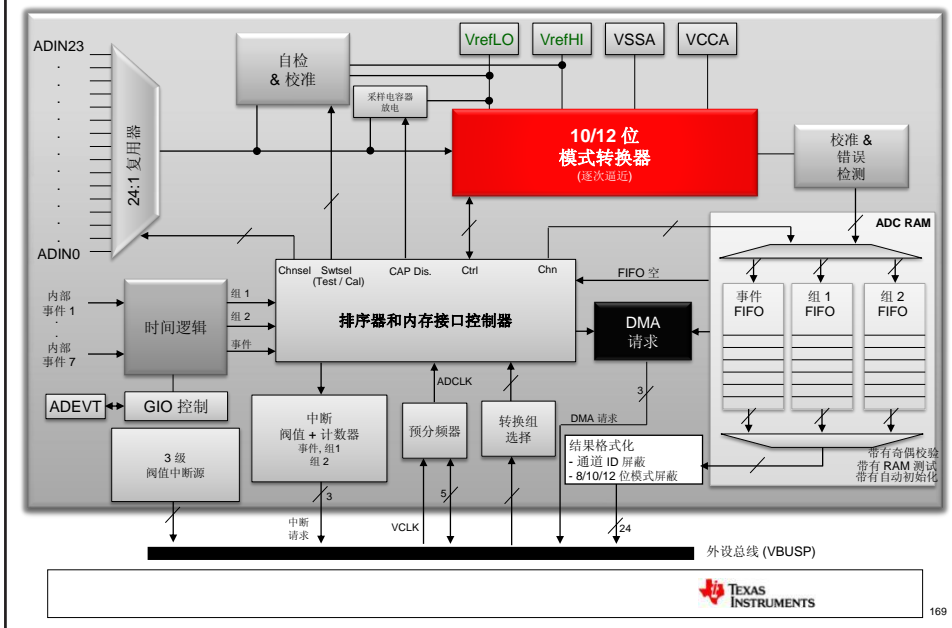


覆盖区域示例



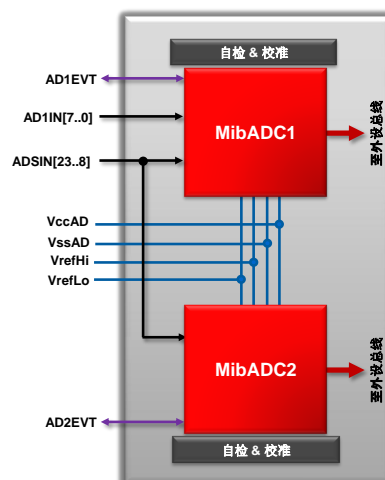
多缓冲模数转换器 (MibADC)

MibADC 方框图



MibADC ADC 工具

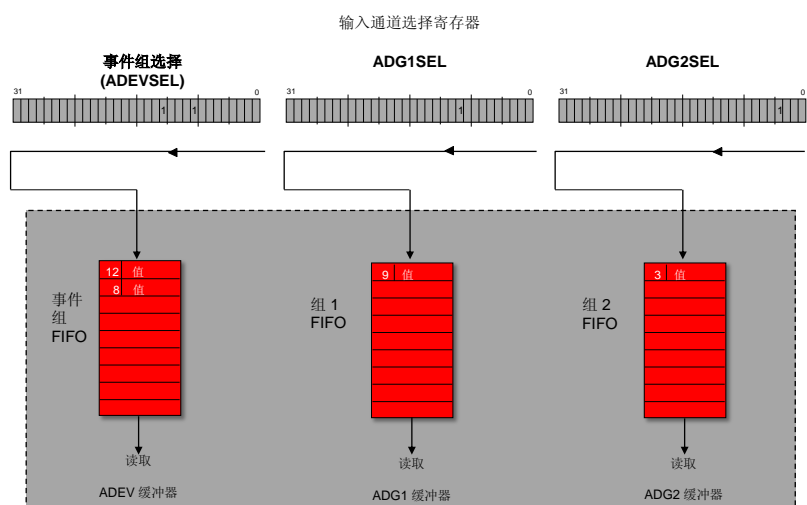
- 双 12 位 ADC 内核:
 - MibADC1 (AD1IN + ADSIN = 24 通道)
MibADC2 (ADSIN = 16 通道)
 - 2 个内核间共用 16 个模拟通道来进行关键安全转换/比较
→ 1002 安全冗余
 - 内部 ADC 基准电压可被用于检查转换器功能性
 - 自检模式使得应用能够检测 ADC 输入的开路/短路
 - ADC 校准逻辑可改进准确度或被用于检测多个测试结果间的漂移
→ 偏移纠错



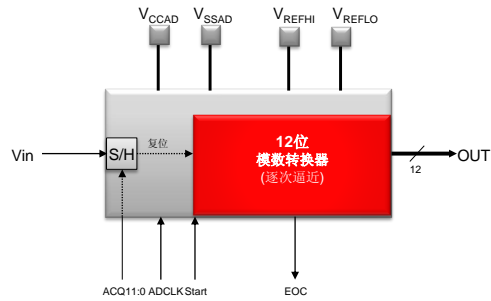
MibADC 运行模式

转换模式	• 用于转换所选外部输入电压的正常激活模式
采样电容器放电模式	• ADC 采样电容器接地的激活模式
校准模式	• 用于使用内部基准电压进行校准的特别激活模式
自检模式	• 用于使用内部基准电压进行故障检测的激活模式
省电模式	• ADC 内部时钟停止的无效模式

MibADC 转换组



MibADC 转换结果



12 bit ADC:
$$\text{DIGITAL_RESULT} = 4096 * (V_{in} - V_{REFLO}) / (V_{REFHI} - V_{REFLO})$$

MibADC 中断

- | | |
|-----------|--|
| 组
转换终止 | • 所有被分配给一个特定组的通道被转换 |
| 组
内存阈值 | • 转换结果的数量超过阈值寄存器值 |
| 组内存
溢出 | • ADC 转换的数量超过分配给那个转换组的缓冲器数量 |
| 量级
阈值 | • 最多三个通道上转换结果的量级比较。两个通道转换结果或一个通道转换结果与一个门限值的可编程比较 |
| 奇偶校验错误 | • 奇偶校验错误发生时，ADC 模块发送一个奇偶校验错误信号到系统模块 |

练习 3:

使用 MibADC 来采集环境光照度传感器数据



175

ADC 练习概述

- 我们将在这个练习中：
 - 使用 ADC 模块从环境光照度传感器获得数据
 - 将经转换的 ADC 值通过 SCI/UART 模块发回 PC。



176

建立一个新的 HALCoGen 项目



- 启动 HALCoGen:

→ Programs → Texas Instruments → Hercules → HALCoGen

- 创建一个新项目:

File → New → Project

- 对于 TMS570 套件:

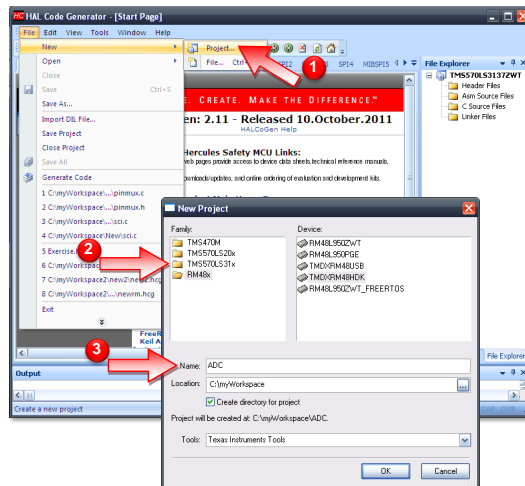
- 选择系列: TMDX570LS31x
- 器件: TMDX570LS31HDK
- 或
- 器件: TMDX570LS31USB

- 对于 RM48 套件:

- 选择系列: TMDXRM48
- 器件: TMDXRM48HDK
- 或
- 器件: TMDXRM48USB

- 然后定义一个名称: 'ADC'

- 位置: "C:\myWorkspace"

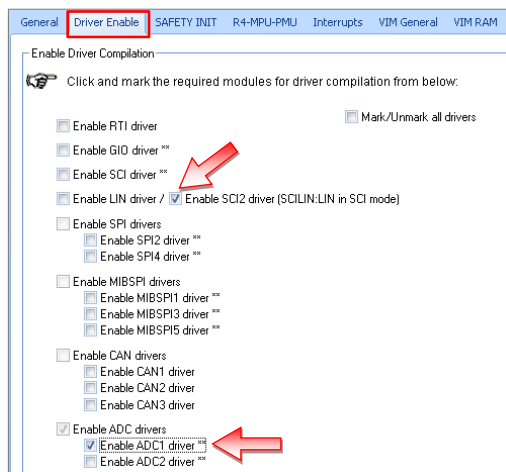


177

驱动程序启用



- 启用器件 'Driver Enable' 子标签页中的 SCI2 和 ADC1 驱动程序

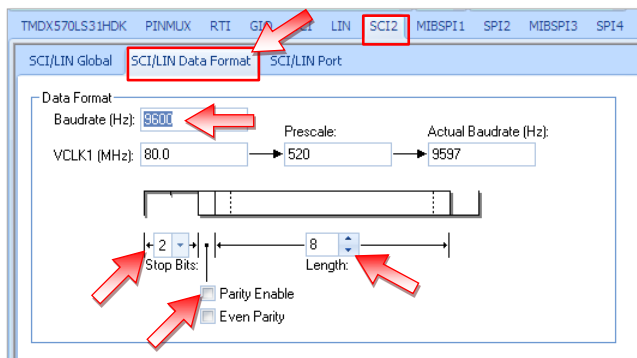


178

SCI 配置

HAL
CoGen

- 选择 SCI2 标签页，然后选择 “SCI/LIN Data Format” 子标签页
- 确保用以下参数设置 SCI 模块：
 - 波特率：9600
 - 数据位：8
 - 无奇偶校验，2 个停止位



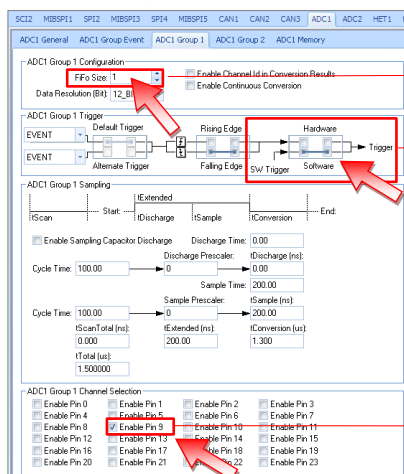
TEXAS
INSTRUMENTS

179

ADC 组配置

HAL
CoGen

- 在 ‘ADC1’ 标签页和 ‘ADC1 Group 1’ 子标签页中，配置 FIFO 大小，触发源和 ADC 通道，入下图所示



每软件触发中，只有一个转换被完成

软件触发作为触发源

环境光照度传感器被连接至 MibADC1 引脚 9

- 生成代码：File → Generate Code 或点击

TEXAS
INSTRUMENTS

180

MibADC 练习



- 将以下代码插入 'sys_main.c' 文件内的相应部分

- 用户代码开始 (0) - #include header section

```
/* USER CODE BEGIN (0) */
#include "sci.h"
#include "adc.h"
#include "stdlib.h"
unsigned char command[8];
/* USER CODE END */
```



181

MibADC 练习



- 用户代码开始 (3) - Main() 部分

```
void main(void)
{
    /* USER CODE BEGIN (3) */
    adcData_t adc_data; //ADC Data Structure
    adcData_t *adc_data_ptr = &adc_data; //ADC Data Pointer
    unsigned int NumberOfChars, value; //Declare variables

    sciInit(); //Initializes the SCI (UART) module
    adcInit(); //Initializes the ADC module

    while(1) // Loop to acquire and send ADC sample data via the SCI (UART)
    {
        adcStartConversion(adcREG1, 1U); //Start ADC conversion
        while(!adcIsConversionComplete(adcREG1, 1U)); //Wait for ADC conversion
        adcGetData(adcREG1, 1U, adc_data_ptr); //Store conversion into ADC pointer
        value = (unsigned int)adc_data_ptr->value;
        NumberOfChars = ltoa(value, (char *)command);
        sciSend(sciInREG, 2, (unsigned char *)"0x"); //Sends '0x' hex designation chars
        sciSend(sciInREG, NumberOfChars, command); //Sends the ambient light sensor data
        sciSend(sciInREG, 2, (unsigned char *)"\r\n"); //Sends new line character
    }
    /* USER CODE END */
}
```



182

测试您的代码



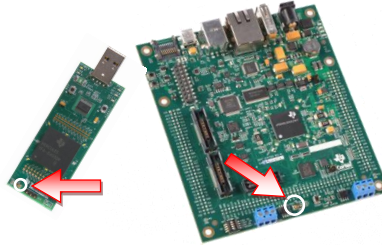
- 完成时，打开您想要的中断程序。
 - 注意：CCS 中包含一个中断终端程序。为了启用它，指向 'View' -> Other 并从 'Show View' 菜单中选择 'Terminal'。如果 'Terminal' 选项不可用，可通过以下指令将它添加为一个 Eclipse 插件：
http://processors.wiki.ti.com/index.php/How_to_install_the_terminal_plugin_in_CCSv5

- 用以下属性设置中断程序：
 - 波特率：9600
 - 数据位：8
 - 无奇偶校验，2 个停止位

- 点击 'Run' 按钮来运行程序



现在您应该在中断程序中看到 ADC 结果



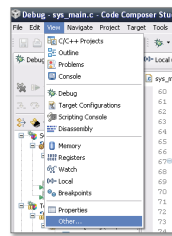
- 使用包含在开发套件中的 LED 闪光灯来改变提供给板载环境光照度传感器的光照级别并且注意中断程序中输出值的改变。



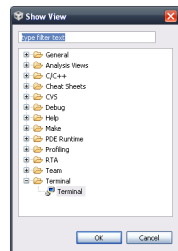
183

启用 CCS 中断

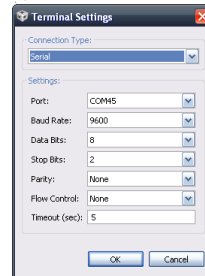
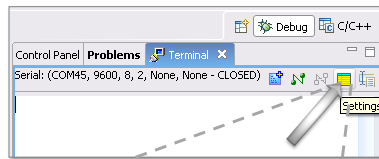
- 1) 选择 View -> Other



- 2) 然后从 'Show View' 中选择 'Terminal'



- 3) 为 'Terminal Tab' 输入合适的通信设置



184

Hercules™ 支持结构



185

Hercules™ TMS570 软件和支持

TMS570 软件资源:

- [TMS570 软件产品主页](#)
 - RTOS 合作伙伴
 - 闪存 API
- [TMS570 MotorWare 软件](#)
 - InstaSPIN-BLDC
 - 具有 SMO 的 FOC 编码器
- [DSP 库主页](#)
 - 60+ 定点和浮点函数
 - CMSIS 兼容
- [TMS570 示例代码库](#)
 - 初始化
 - 引导载入程序
 - CAN, 以太网, FlexRay 以及更多
- [HALCoGen 外设驱动程序](#)
 - GUI 配置

TMS570 网页:

www.ti.com/TMS570

- 数据表
- 技术参考手册
- 应用说明
- 订购评估和开发套件

TMS570 工程师交流论坛:

www.ti.com/hercules-support

- 新闻和公告
- 有用链接
- 询问技术问题
- 搜索技术内容

TMS570 WIKI:

www.ti.com/hercules-tms570-wiki

- 如果指导
- 介绍视频
- 通用信息



186

还想参加其它培训吗？

Hercules™ 微控制器安全设计和编程：

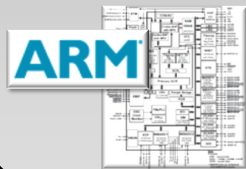
第一天

- 欢迎和介绍
- Hercules™ 产品概述 / MCU 路线图
- 安全标准和 Hercules 安全特性 / 练习
- HALCoGen / 练习
- Code Composer Studio / 演示
- 编译器
- 闪存概述
- 闪存工具：nowFlash, nowECC / 练习



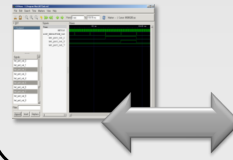
第二天

- 总结 / 提问
- ARM® Cortex™-R4F CPU
- 系统模块概述
- 器件设置/启动、实时中断模块、矢量中断管理器
- CRC 控制器, CPU 比较模块, 错误信令模块, 双时钟比较, JTAG 安全模块
- 通用 I/Os / 练习
- 直接内存访问控制器 (DMA)
- 串行通信接口 (SCI/UART) / 练习



第三天

- 总结 / 提问
- 多缓冲 ADC (MIBADC) / 练习
- 多缓冲串行外设接口 (SPI / MIBSPI-P)
- DCAN
- FlexRay
- 外部内存接口 (EMIF) / 参数覆盖模块 (POM)
- 以太网
- USB 主机 / 器件
- HET (高端定时器) IDE
- N2HET & 传输单元 / 练习
- 总结 / 提问 / 调查



187



谢谢！

请填写培训课程调查问卷



188

