Application Note

MSPM0Gx51x 至 MSPM33C32xx 软件迁移指南

Eric Rentschler

摘要

MSPM0Gx51x 和 MSPM33C32xx 具有许多封装、特性和引脚排列。因此，可以使用相同的布局来容纳 MSPM0Gx51x 和 MSPM33C32xx。本应用手册讨论了在器件之间迁移时的软件注意事项。也可以使用硬件迁移指南 (SLAAES2)。

内容

商标

1 简介

本文档介绍了在 MSPM0Gx51x 与 MSPM33C32xx 系列器件之间进行软件迁移时的相关注意事项。

现代微控制器集成了日益丰富的功能与外设。因此，在器件之间迁移可能是一项挑战。为了简化该流程，本文档重点介绍了 MSPM0Gx51x 和 MSPM33C32xx 系列器件之间的差异，但同样的原理也可应用于任何 M0 和 M33 器件

本指南在迁移或设计过程之前使用最为有效。请按照以下步骤处理 MSPM0Gx51x 与 MSPM33C32xx 之间兼容性考虑的各个方面。提供这两个数据表即可支持迁移过程。

1. 选择目标 MSPM0Gx51x 器件和目标 MSPM33C32xx 器件
2. 查看特性级别差异
3. 查看引脚排列差异

2 查看 MSPM0Gx51x 和 MSPM33C32xx 之间的软件差异

MSPM0Gx51x 系列和 MSPM33C32x 系列具有一些相同的特性和规格，但以下各节描述了影响软件迁移的差异。

2.1 开发环境

软件开发工具

建议使用以下工具为 MSP 产品开发软件：

MSPM33 软件开发套件：MSPM33SDK
Code Composer Studio
示例代码
驱动程序库
SysConfig
CCS 集成 SysConfig
调试接口

2.2 软件移植流程

下图展示了从 MSPM0Gx51x 系列器件迁移至 MSPM33C 系列器件的软件迁移步骤。以下各节介绍了部署环境之间的差异详细信息，并简要概述了外设之间的差异，同时还结合了 SLAAES2。

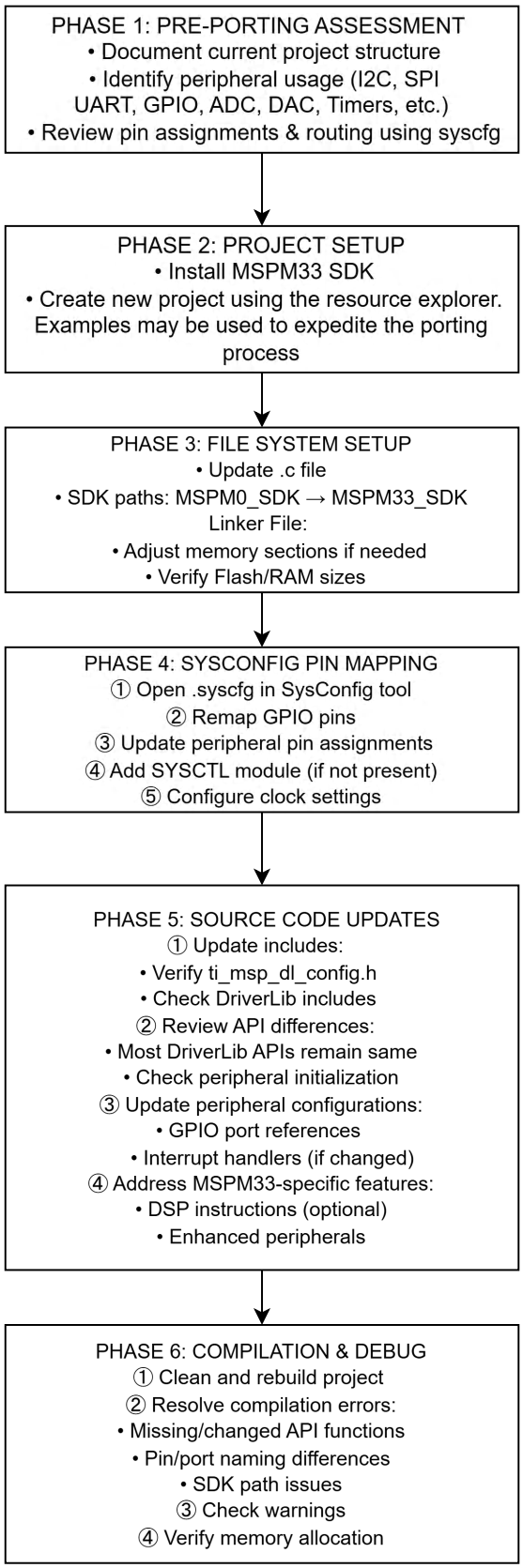

图 2-1. 软件移植流程图

此流程图介绍了迁移过程。尽管 SDK 之间有相似之处，但建议工程师一次实现一个子系统，以便在每个步骤都可以验证性能。

2.3 外设差异

以下各节介绍了不同外设的迁移注意事项。

2.3.1 中断差异

MSPM0SDK 和 MSPM33SDK 之间最显著的区别之一是中断的格式化。MSPM0G 系列限制为 32 个器件中断，而 M33 内核支持多达 496 个中断，每个外设都具有专用中断矢量。这意味着之前由 GROUP_1_IRQHandler 处理的 GPIOA 中断由 GPIOA_IRQHandler 处理。此外，与组关联的任何状态 API 不再适用，必须传输到特定于外设的 API。

在大多数应用中，中断可以是必须进行的最大更改之一。但是，由于中断请求 (IRQ) 被移动到外设本身，而不是利用一个大的组来处理其中包含的所有外设，因此更改非常简单。

2.3.2 ADC 差异

与 MSPM0x51x 中的 ADC12 相比，MSPM33C 系列中的 ADC 外设具有一系列显著的变化。虽然两者都是 12 位 SAR ADC 外设，但 MSPM33C 的高速模数转换器 (HSADC) 具有额外的时序控制，过采样和后处理功能，转换到器件时必须考虑这些功能。

除命名约定之外，以下各项的更改极少：

- 基本通道选择
- 电源管理 (启用/禁用/复位)
- 基本中断
- 时钟源选择

以下情况可能需要更显著的重新设计：

- 转换序列
- 硬件均值计算
- 结果处理
- 中断配置
- 触发

下图显示了两个芯片之间每个功能的不同之处。如果要移植的代码利用这些功能，则必须使用 SysConfig 或应用程序代码中的定义更改许多常量。

外设规格	ADC12 (MSPM0Gx51x)	HSADC (MSPM33Cx)
转换模式	单个采样，重复单个采样，序列，重复序列	4 个具有优先级和抢占支持的独立序列发生器 (SEQ1-4)
触发	使用 DL_ADC12_TRIG_SRC_SOFTWARE、DL_ADC12_TRIG_SRC_EVENT 触发	使用 DL_HSADC_Trigger_TieLow_SW_Trig、DL_HSADC_Trigger_GEN_SUB_X 按序列发生器触发
后处理	基本硬件均值计算 (2 至 128 个样本) ，通过 SysConfig 提供简单的分频选项	4 个后处理块 (PPB)，具有偏移消减、二进制补码、可配置过采样、右移功能、跳闸比较器和误差计算
结果存储	12 个 MEMRES 寄存器 (MEMRES0-11)，简单 FIFO 模式	16 个结果寄存器 (ADCRESULT0-15)，4 个 PPB 结果寄存器，每个序列发生器具有状态的 FIFO，过采样和/计数寄存器
中断	DL_ADC12_INTERRUPT_MEM0_RESULT_LOADED，DL_ADC12_INTERRUPT_OVERFLOW，DL_ADC12_INTERRUPT_WINDOW_COMP_HI/LO	4 个可配置中断，4 个 dma 中断，每个块 PPB 跳闸中断，预先生成占先中断
采样定时器	2 个采样计时器 (SCOMP0/1)，采样时间= SCOMP +1 * CLK_DIV	每个序列发生器的采样窗口介于 1 到 1472 个周期之间，使用 VrefLo 和 VrefHi 提供简单的电容复位选项

外设规格	ADC12 (MSPM0Gx51x)	HSADC (MSPM33Cx)
时钟配置	DL_ADC12_CLOCK_DIVIDE_1 … DL_ADC12_CLOCK_DIVIDE 48，总共 48 个分频选项	DL_HSADC_CLOCK_DIVIDE_1_0、DL_HSADC_CLOCK_DIVIDE_2_0、DL_HSADC_CLOCK_DIVIDE_2_5...DL_HSADC_CLOCK_DIVIDE_8_5，从 1 到 8.5
通道	DL_ADC12_INPUT_CHAN_0 至 DL_ADC12_INPUT_CHAN_31	DL_HSADC_IN_0 至 DL_HSADC_IN_31

2.3.3 安全差异

MSPM0Gx51x 和 MSPM33Cx 之间的许多安全特性几乎相同，并具有兼容的包装器。下面重点介绍了每个安全外设中必要的包装器更改。

AESADV 功能	MSPM0SDK	MSPM33SDK
命名规则	DL_AESADV_* AESADV_REGS	所有函数和结构都是相同的，但在“AESADV”末尾添加 HP 除外 DL_AESADVHP_* AESADVHP_Regs
中断寄存器结构	• aesadv->CPU_INT.IMASK • aesadv->CPU_INT.RIS • aesadv->CPU_INT.MIS • aesadv->CPU_INT.ICLR • aesadv->CPU_INT.IIDX • aesadv->DMA_TRIG_DATAIN.IMASK • aesadv->DMA_TRIG_DATAOUT.IMASK	• aesadv->INT_EVENT0.IMASK • aesadv->INT_EVENT0.RIS • aesadv->INT_EVENT0.MIS • aesadv->INT_EVENT0.ICLR • aesadv->INT_EVENT0.IIDX • aesadv->INT_EVENT1.IMASK • aesadv->INT_EVENT2.IMASK

2.4 通信外设

I2C、UART 和 SPI 是通过 MSPM33Cx 上的 UNICOMM 实现的，而不是 MSPM0Gx51x 上的各个外设。虽然运行每个外设的方式相似，但包装器实现方式不同，在传输到 MSPM33Cx 时，必须对任何启用 UNICOMM 的外设的实现进行更改。下表列举了 MSPM0SDK 和 MSPM33SDK 之间包装的差异。

2.4.1 SPI 差异

SPI 外设特性	SPI (MSPM0SDK)	UNICOMM SPI (MSPM33SDK)
包装器	通过 SPI_Regs 直接访问寄存器	使用 using UNICOMM_Inst_Regs->spi 访问 UNICOMM 中包装的 SPI
电源管理	DL_SPI_enablePower(SPI0)	DL_SPI_enablePower(UNICOMM0)
时钟源	• DL_SPI_CLOCK_BUSCLK • DL_SPI_CLOCK_MFCLK • DL_SPI_CLOCK_LFCLK	• DL_SPI_CLOCK_BUSCLK • DL_SPI_CLOCK_MFCLK • DL_SPI_CLOCK_LFCLK • DL_SPI_CLOCK_ASYNC_LFCLK • DL_SPI_CLOCK_ASYNC_SYSCLK • DL_SPI_CLOCK_ASYNC_HFCLK • DL_SPI_CLOCK_ASYNC_PLL
FIFO 阈值级别	• DL_SPI_yX_FIFO_LEVEL_3_4_EMPTY • DL_SPI_yX_FIFO_LEVEL_1_2_EMPTY • DL_SPI_yX_FIFO_LEVEL_1_4_EMPTY • DL_SPI_yX_FIFO_LEVEL_ONE_ENTRY • DL_SPI_yX_FIFO_LEVEL_EMPTY	• DL_SPI_yX_FIFO_LEVEL_3_4_EMPTY • DL_SPI_yX_FIFO_LEVEL_1_2_EMPTY • DL_SPI_yX_FIFO_LEVEL_1_4_EMPTY • DL_SPI_yX_FIFO_LEVEL_ONE_ENTRY • DL_SPI_yX_FIFO_LEVEL_EMPTY • DL_SPI_yX_FIFO_LEVEL_ALMOST_EMPTY • DL_SPI_yX_FIFO_LEVEL_ALMOST_FULL
抢占添加项	MSPM0 中无抢占	• DL_SPI_INTERRUPT_DMA_PREIRQ_RX • DL_SPI_INTERRUPT_DMA_PREIRQ_TX • DL_SPI_INTERRUPT_LTOUT • DL_SPI_IIDX_PREIRQ_RX • DL_SPI_IIDX_PREIRQ_TX • DL_SPI_IIDX_LTOUT
状态寄存器添加		• DL_SPI_getCurrentCDMODEValue(unicomm) • DL_SPI_isTXFIFOCleared(unicomm) • DL_SPI_isRXFIFOCleared(unicomm)
暂停/恢复		• DL_SPI_enabledSuspend(unicomm) • DL_SPI_disableSuspend(unicomm)
包装特性	• DL_SPI_enablePacking(spi) • DL_SPI_disablePacking(spi) • DL_SPI_isPackingEnabled(spi)	删除了封装特性，所有 32 位运算都不使用封装使能即可运行

2.4.2 UART 差异

UART 外设特性	UART (MSPM0SDK)	UNICOMM UART (MSPM33SDK)
包装器	通过 UART_REGS 访问的独立 UART 外设	UNICOMM_Inst_regs->uart
时钟源增强功能	• DL_UART_CLOCK_BUSCLK • DL_UART_CLOCK_MFCLK • DL_UART_CLOCK_LFCLK	• DL_UART_CLOCK_BUSCLK • DL_UART_CLOCK_MFCLK • DL_UART_CLOCK_LFCLK • DL_UART_CLOCK_ASYNC_LFCLK • DL_UART_CLOCK_ASYNC_SYSCLK • DL_UART_CLOCK_ASYNC_HFCLK • DL_UART_CLOCK_ASYNC_PLL
API 兼容性宏		向后兼容性高，所有 DL_UART_MAIN_* 和 DL_UART_EXTEND_* API 均映射到 Unicomm
FIFO 管理	• DL_UART_enableFIFOs() • DL_UART_disableFIFOs()	始终启用 FIFO，但仍提供 API 以确保向后兼容性。例如，调用 DL_UART_isFIFOsENABLED() 总是返回 true
FIFO 阈值级别	• DL_UART_yX_FIFO_LEVEL_3_4_EMPTY • DL_UART_yX_FIFO_LEVEL_1_2_EMPTY • DL_UART_yX_FIFO_LEVEL_1_4_EMPTY • DL_UART_yX_FIFO_LEVEL_ONE_ENTRY • DL_UART_yX_FIFO_LEVEL_EMPTY	• DL_UART_yX_FIFO_LEVEL_3_4_EMPTY • DL_UART_yX_FIFO_LEVEL_1_2_EMPTY • DL_UART_yX_FIFO_LEVEL_1_4_EMPTY • DL_UART_yX_FIFO_LEVEL_ONE_ENTRY • DL_UART_yX_FIFO_LEVEL_EMPTY • DL_UART_yX_FIFO_LEVEL_ALMOST_EMPTY • DL_UART_yX_FIFO_LEVEL_ALMOST_FULL
抢占添加项	MSPM0Gx51x 上没有兼容特性	• DL_UART_IIDX_PREEMPT_INT_TX • DL_UART_IIDX_PREEMPT_INT_RX • DL_UART_INTERRUPT_LINE_TIMEOUT • DL_UART_IIDX_LINE_TIMEOUT
寄存器命名	• 控制： CTLx • 线路控制：LCRH • 状态：STAT • TX 数据：TXDATA • RX 数据：RXDATA • 波特率：IBRD/FBRD	• 控制：CTLx • 线路控制：LCRH • 状态：STAT • TX 数据：TXDATA • RX 数据：RXDATA • 波特率：IBRD/FBRD
电源管理	• DL_UART_enablePower(UARTx)	• DL_UART_enablePower(UNICOMM0) • DL_UNICOMM_enablePower(unicomm)

UART 外设特性	UART (MSPM0SDK)	UNICOMM UART (MSPM33SDK)
备份/恢复配置	• DL_UART_Main_backupConfig • DL_UART_Extend_backupConfig • DL_UART_Main_saveConfiguration() • DL_UART_Extend_saveConfiguration()	• DL_UART_backupConfig • DL_UART_saveConfiguration() • DL_UART_restoreConfiguration()
模拟干扰滤波器		MSPM33C 中没有模拟滤波器。
未提及的核心功能	使用 UART_Regs_*	使用 UNICOMM_Inst_Regs_*
API 兼容性宏		向后兼容性高，所有 DL_UART_MAIN_* 和 DL_UART_EXTEND_* API 均映射到 Unicomm

2.4.3 I2C 差异

I2C 外设特性	I2C (MSPM0SDK)	UNICOMM I2C (MSPM33SDK)
包装器	通过 I2C_Regs 直接访问寄存器，在同一外设中支持控制器和目标模式	I2C 控制器使用 UNICOMM 模块包装，通过 UNICOMM_Inst_Regs->i2cc/i2ct 访问寄存器
模式选择	同一外设中的 I2C 控制器和目标 DL_I2C_enableController(I2Cx)、DL_I2C_enableTarget(I2Cx)	• 控制器使能 DL_I2CC_ENABLE(UNICOMMx) • 目标使能 DL_I2CT_ENABLE(UNICOMMx)
寄存器命名更改	• 控制寄存器：MCTR • 目标地址：MSA • 状态寄存器：MSR • 计时器周期：MTPR • Configuration: MCR	• 控制寄存器：CTR • 目标地址：SA • 状态寄存器：SR • 计时器周期：TPR • Configuration: CR
时钟源	• DL_I2C_CLOCK_BUSCLK • DL_I2C_CLOCK_MFCLK	• DL_I2Cx_CLOCK_BUSCLK • DL_I2Cx_CLOCK_MFCLK • DL_I2Cx_CLOCK_ASYNC_SYSCLK • DL_I2Cx_CLOCK_ASYNC_HFCLK • DL_I2Cx_CLOCK_ASYNC_PLL “I2Cx”指控制器的 I2CC 或目标的 I2CT

I2C 外设特性	I2C (MSPM0SDK)	UNICOMM I2C (MSPM33SDK)
FIFO 阈值级别	• DL_I2C_TX_FIFO_LEVEL_EMPTY • DL_I2C_TX_FIFO_LEVEL_BYTES_x • DL_I2C_RX_FIFO_LEVEL_BYTES_x LEVEL_BYTES_x 从 1 到 7	• DL_I2Cx_yX_FIFO_LEVEL_3_4_EMPTY • DL_I2Cx_yX_FIFO_LEVEL_1_2_EMPTY • DL_I2Cx_yX_FIFO_LEVEL_1_4_EMPTY • DL_I2Cx_yX_FIFO_LEVEL_ONE_ENTRY • DL_I2Cx_yX_FIFO_LEVEL_EMPTY • DL_I2Cx_yX_FIFO_LEVEL_ALMOST_EMPTY • DL_I2Cx_yX_FIFO_LEVEL_ALMOST_FULL 对于目标配置，I2CC 使用 TX，I2CT 使用 RX
干扰滤波	• DL_I2C_ANALOG_GLITCH_FILTER_WIDTH_X • DL_I2C_DIGITAL_GLITCH_FILTER_WIDTH_X	MSPM33C 没有模拟干扰滤波器。 • DL_I2CC_DIGITAL_GLITCH_FILTER_WIDTH_X
API 函数名称更改	• DL_I2C_enableController() • DL_I2C_startControllerTransfer() • DL_I2C_transmitControllerData() • DL_I2C_receiveControllerData() • DL_I2C_getControllerStatus() • DL_I2C_fillControllerTXFIFO()	• DL_I2Cx_enable() • DL_I2Cx_startTransfer() • DL_I2Cx_transmitData() • DL_I2Cx_receiveData() • DL_I2Cx_getStatus() • DL_I2Cx_fillTXFIFO()
中断命名	• DL_I2C_INTERRUPT_CONTROLLER_RX_DONE • DL_I2C_INTERRUPT_CONTROLLER_TX_DONE • DL_I2C_INTERRUPT_CONTROLLER_RXFIFO_TRIGGER • DL_I2C_INTERRUPT_CONTROLLER_TXFIFO_TRIGGER • DL_I2C_INTERRUPT_CONTROLLER_NACK	• DL_I2Cx_INTERRUPT_RX_DONE • DL_I2Cx_INTERRUPT_TX_DONE • DL_I2Cx_INTERRUPT_RXFIFO_TRIGGER • DL_I2Cx_INTERRUPT_TXFIFO_TRIGGER • DL_I2Cx_INTERRUPT_NACK
抢先式 DMA 中断	MSPM0 不支持抢先式 DMA 中断。	• DL_I2Cx_IIDX_PREEMPT_INT_TX • DL_I2Cx_IIDX_PREEMPT_INT_RX

3 总结

总之，本应用手册提供了将软件从 MSPM0Gx51x 迁移到 MSPM33C32xx 的指南。它将与硬件迁移指南 (SLAAES2) 一起使用，以便在两个设备之间执行完全迁移。

4 参考资料

1. 德州仪器 (TI)，MSPM33C321x 混合信号微控制器数据表
2. 德州仪器 (TI)，MSPM33 C 系列 160MHz 微控制器，技术参考手册
3. 德州仪器 (TI)，MSPM0Gx51x 混合信号微控制器数据表
4. 德州仪器 (TI)，MSPM0 G 系列 80MHz 微控制器，技术参考手册
5. 德州仪器 (TI)，MSPM0Gx51x 和 MSPM33C32xx 硬件迁移指南，硬件迁移指南

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月