life.augmented

AN2945
应用笔记

STM8S 和 STM32™ MCU：一致的 8/32 位产品线实现轻松移植

前言

在屡获殊荣的 STM32™ 微控制器上市后，意法半导体推出了 STM8S 系列，借此完成了其微控制器产品线的更新。为了使 MCU 产品组合合理化，意法半导体付出了很大努力，特别是对旨在简化产品移植的通用外设和软件工具方面的投入。

如采用新 MCU 系列进行设计，维持开发团队所需的时间成本和费用成本是选择微控制器供应商时的主要标准。因此，如果这种非重复性投资适用于范围广泛的 MCU，这将是一个优势。8 位 STM8S 和 32 位 STM32 系列 MCU 产品采用 20 到 144 引脚的封装，内存大小从 2 KB 到 512 KB，这为产品组合的构建提供了极大的灵活性。如果 8 位应用耗尽 MIPS，可以升级到 STM32 系列。相反，如果希望削减 32 位平台的成本，切换到 STM8 系列也相当简单。

本文档介绍了 STM8S 和 STM32 产品线的相似之处和共同特性，旨在帮助用户从一个系列移植到另一个系列。

目录

表格索引

图片索引

1 内核

STM8™ CPU 是一种专有架构，它保持了以前的 ST7 内核的传统，同时在 8 位 CPU 效率和代码密度方面实现了突破。STM32 围绕行业标准 ARM® Cortex™-M3 32 位内核构建，并受益于与 ARM 处理器有关的开发工具和软件解决方案的完整生态产业环境。尽管它们被认为是两种完全不同的处理器，但它们在架构方面实际有许多相似之处，*表 1* 对此进行了汇总。

表 1. STM8 和 STM32：内核比较

	STM8	Cortex-M3
数据路径	8 位	32 位
Drhystone MIPS (0WS)	0.29 DMIPS	1.22 DMIPS
架构	哈佛	哈佛
流水线	有，三级	有，三级
指令集	CISC	RISC
程序总线数据宽度	32 位	32 位
预取缓冲器	有，2 × 32 位，内部	有，2 × 64 位，在存储器接口中
平均指令大小	2 字节	2 字节
中断类型 延迟	向量化 9 个周期，支持中断咬尾功能	向量化 12 个周期，支持中断咬尾功能
低功耗模式	慢速、等待事件或中断、暂停、退出时暂停	慢速、睡眠（等待事件或中断）、退出时睡眠、深度睡眠
调试接口	单线 (SWIM)	双线或传统 JTAG

两种内核均基于哈佛架构。它们采用 3 级流水线执行，可将执行时间降至最低，对于 STM8S，时钟速度高达 24 MHz，对于 STM32 系列，时钟速度高达 72 MHz。

它们具有多种低功耗模式，高效节能，并且受益于比平均指令长度更宽的存储器接口（分别为 32 位和 64 位宽总线）。这可以大幅减少对存储器总线的访问次数，从而降低与地址总线切换和非易失性存储器读访问有关的功耗。中断咬尾功能和退出时暂停/睡眠模式还有助于避免不必要的堆栈存取。

最后，在代码密度方面，它们均有优异的表现，这归功于 STM8S 系列的 8 位 CISC 指令集以及 STM32 系列的 Cortex 内核引入的 16 位 Thumb-2 模式。

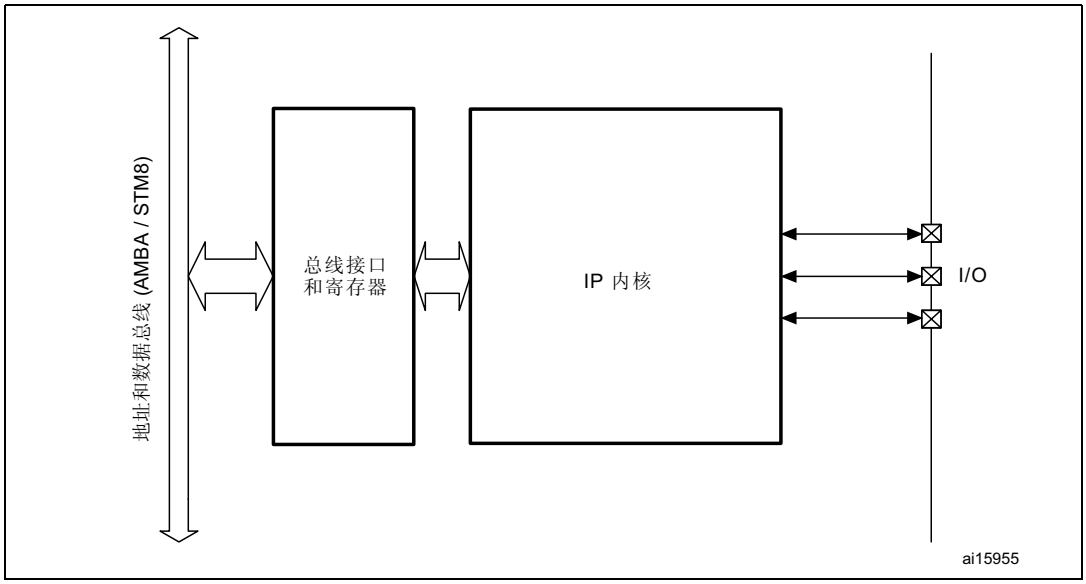
上面的简单比较表明这两种处理器在微架构特性方面都具有领先的技术水平。STM8 处于传统 16 位处理器的级别，而 Cortex-M3 则满足当前使用 32 位 MCU 至中高端 16 位 MCU 的应用的要求。因此，STM8 和 STM32 的组合确保了性能的连续性，而且现在有第三方为这两条产品线提供了统一的开发平台，使这种连续性在工具层面上也得到支持。

2 外设

MCU 外设（也称为 IP）是 8 位和 32 位产品线之间保持 ST MCU 一致性的另一个示例：大多数基本 IP 均定义和构建为可从一个产品系列移植到另一个产品系列。这通过将 8 位外设简单但有效地修改为 32 位字来实现。这样做的好处是可节约成本和功耗，并且资源易于了解，如果需要更高性能，可在系统层面通过更宽的总线和 DMA 控制器对资源加以补充。在了解了外设的工作原理后，可以将外设应用到 STM8S 和 STM32 系列，从而加速两种器件之间的转换。

图 1 给出了数字外设的简化表示。

图 1. 数字外设的内部结构

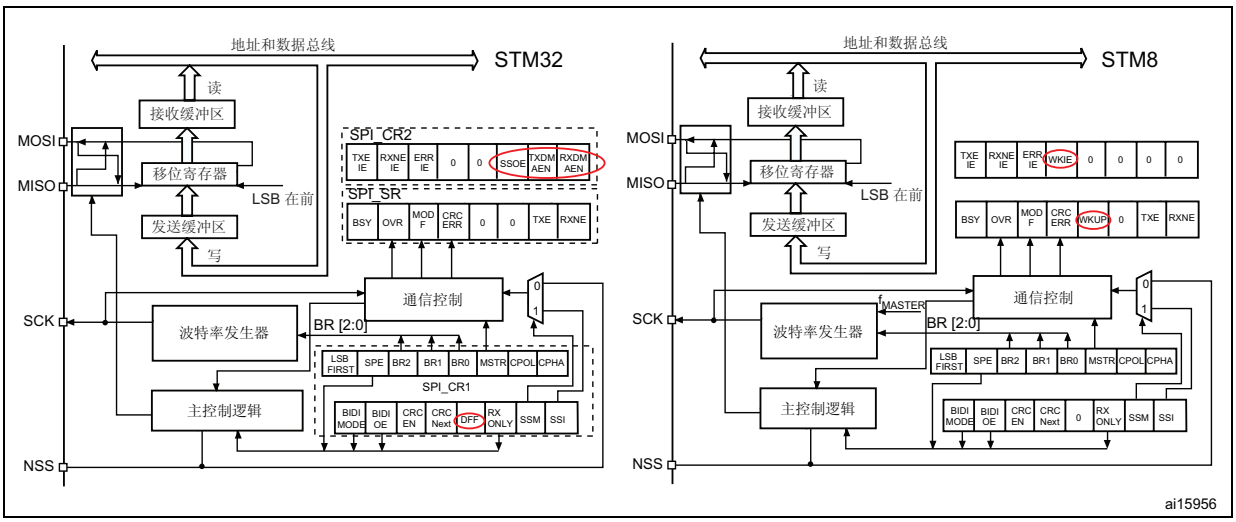


外设可分成两个主要部分。首先，内核包含状态机、计数器以及任意种类的组合逻辑或顺序逻辑，它们是执行不需要处理器的任务（如低级通信层、模拟后端管理或时序驱动的功能）时所必需的。如有必要，可通过 MCU 端口将内核与外界连接。外部连接可能包含一些 I/O 或复杂总线。其次，应用程序通过与内部总线连接的寄存器来初始化和控制外设，该内部总线与其它 MCU 资源共用。在 8 位微控制器中，处理器直接写入和读取寄存器，而在 32 位产品中，寄存器读写操作通常通过总线桥来完成。然而，两个系列的主要区别在于外设必须遵循的内部总线规范。这解释了 STM8S 和 STM32 器件能够共用外设的原因：它们基于相同的内核，而且仅针对两种不同的总线接口定制。ARM 处理器和外设符合 AMBA 总线规范，采用 32 位数据总线，而 STM8S 器件使用更为简单但有效的 8 位总线标准。从功能角度看，它们仅在以下方面存在差异：

- 寄存器大小：8 位与 16 或 32 位
- 直接取决于 CPU 运行速度的最大时钟频率
- DMA，可通过简单数据管理减轻 CPU 的负荷并提高最大数据吞吐量
- 一些产品特定功能，如 I/O 端口管理

我们来看一下*图 2* 所示的 STM8S 和 STM32 SPI 框图。初看上去，它们的框图是相同的，只是有几个位不同（*图 2* 中以红色突出显示），例如，在 DMA 级别。

图 2. SPI框图



现在看一下*表 2*和*表 3*所示的寄存器映射，它们明显是基于相同的设计：除了几个有区别的位和寄存器大小外，寄存器和位的名称以及在寄存器中的位置都是相似的。

表 2. STM32 SPI 寄存器映射和复位值

偏移	寄存器	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**SPI_CR1**	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**SPI_CR2**	Reserved																								TXEIE	RXNEIE	ERRIE	Reserved		SSOE	TXDMAEN	RXDMAEN
	Reset Value																									0	0	0			0	0	0
0x08	**SPI_SR**	Reserved																								BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset Value																									0	0	0	0	0	0	1	0
0x0C	**SPI_DR**	Reserved																DR[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**SPI_CRCPR**	Reserved																CRCPOLY[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	**SPI_RXCRCR**	Reserved																RxCRC[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**SPI_TXCRCR**	Reserved																TxCRC[15:0]															
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**SPI_I2SCFGR**	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset Value																					0	0	0	0	0		0	0	0	0	0	0
0x20	**SPI_I2SPR**	Reserved																						MCKOE	ODD	I2SDIV							
	Reset Value																							0	0	0	0	0	0	0	0	1	0

表 3. STM8 SPI 寄存器映射和复位值

地址偏移	寄存器名称	7	6	5	4	3	2	1	0
0x00	SPI_CR1 Reset value	LSBFirst 0	SPE 0	BR2 0	BR1 0	BR1 0	MSTR 0	CPOL 0	CPHA 0
0x01	SPI_CR2 Reset value	BDM 0	BDOE 0	CRCEN 0	CRCNEXT 0	Reserved 0	RXONLY 0	SSM 0	SSI 0
0x02	SPI_ICR Reset value	TXIE 0	RXIE 0	ERRIE 0	WKIE 0	Reserved 0	Reserved 0	Reserved 0	Reserved 0
0x03	SPI_SR Reset value	BSY 0	OVR 0	MODF 0	CRCERR 0	WKUP 0	Reserved 0	TXE 1	RXNE 0
0x04	SPI_DR Reset value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
0x05	SPI_CRCPR Reset value	MSB 0	- 0	- 0	- 0	- 0	- 1	- 1	LSB 1
0x06	SPI_RXCRCR Reset value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0
0x07	SPI_TXCRCR Reset value	MSB 0	- 0	- 0	- 0	- 0	- 0	- 0	LSB 0

表 4 列出了通用外设，强调了两种产品在寄存器、位和特性层面上的一致性。

表 4. STM8 和 STM32 器件共有的外设

外设名称	
STM32	**STM8**
独立看门狗 (IWDG)	
窗口看门狗 (WWDG)	
串行外设接口 (SPI)	
内部集成电路 (I^2C) 接口	
通用同步/异步收发器 (USART)	
高级控制定时器	16 位高级控制定时器
通用定时器	16 位通用定时器
基本定时器	8 位基本定时器

尽管定时器看起来与许多配置不同，但它们在各产品系列之间和内部的架构是相同。只存在一种定时器架构的变体。从集合角度看，可以选择性地去除子集，以减少捕获/比较通道数，或仅删除一些特定应用（如电机控制）所需的选项。

3 系统特性

当今的 MCU 是复杂的 SoC（片上系统），其中不仅包含许多外设，还包含一些高级系统特性，旨在缩减物料清单或增强系统的安全性和稳定性。8 位和 32 位平台均是如此。

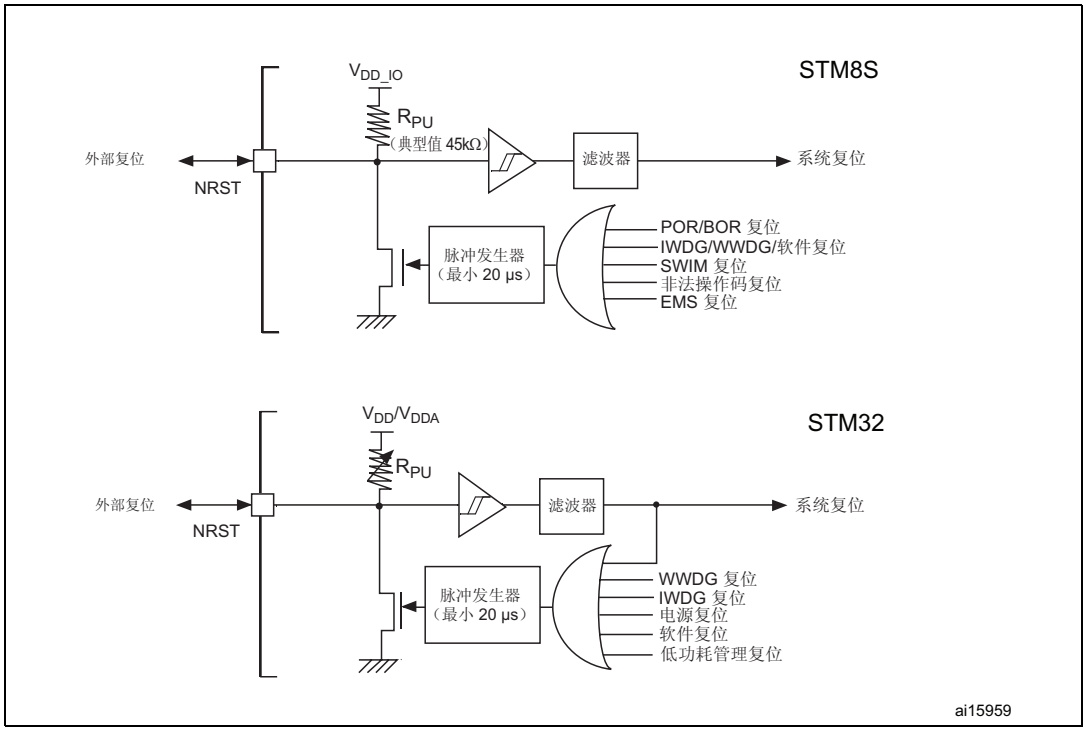
3.1 复位

如*图 3* 所示，STM8S 和 STM32 器件具有相同的复位电路，仅有细微的差异。

NRST 引脚既是输入也是具有内置上拉电阻的开漏输出。为实现 EMS（电磁敏感度）鲁棒性，插入了一个滤波器以避免毛刺传播到数字电路中。双向复位有三个优点：

- 对于多 MCU 系统，双向复位可确保在启动或热复位时所有子处理器均正确同步
- 对于其它 IC，在系统层面还可以使用 MCU 内嵌的电压监控器（上电复位和欠压复位）
- 当生成假内部复位时，在调试过程中会有很大帮助

图 3. STM8S 和 STM32 复位电路



3.2 时钟

从时钟系统的角度看，这两种产品具有三个相同的主要时钟源，它们共有相似的电气特性。详细信息，请参见*表 5*。

振荡器负责管理晶振和谐振器，它被称为 HSE（高速外部时钟源）。也可以旁路此振荡器以向 MCU 馈送外部时钟。这适用于对精度和稳定性具有严格要求的应用，例如通信用途。

使用 HSI（高速内部）时钟，应用可以在高频率下运行，而无需外部晶振。此时钟源的电流消耗比 HSE 低 10 倍，而且精度误差百分比非常低。它还可用作 STM32 上的 PLL 输入，将内部频率提高至 64 MHz。

最后，低速内部时钟 (LSI) 是一个超低内部电源（几 µA），它可以永久使能，以在暂停或停止模式下为自动唤醒的外设提供时钟。它还可以为辅助的板上看门狗提供时钟（更多详细信息，请参见 *第 3.4 节：安全性*），并可用作 STM8 产品的 CPU 时钟。此时钟源并不准确（误差为几十个百分点），但可以定期使用精确的 HSI 时钟对其进行测量，以补偿芯片制造差异或因温度等原因引起的漂移。

表 5. STM8S/STM32 时钟源特性（指示性数据(1)）

系统时钟源	频率		精度误差	电流消耗
	STM8S	STM32		
高速外部 (HSE)	1-24 MHz	4-16 MHz	取决于晶振，低至几十 ppm	1 到 2 mA
高速内部 (HSI)	16 MHz	8 MHz	典型值 1%	100 到 250 µA
低速内部	110-146 kHz	30-60 kHz	20 到 50%	1 到 5 µA

1. 有关详细的电气特性，请参见产品数据手册。

3.3 存储器

两种产品线均基于非易失性存储器并具有一个选项字节加载器。此机制取代了用于 MCU 上电配置的传统熔丝位：用户可以在编程时选择多种选项，这些选项会随程序二进制映像一起写入。

所有新型微控制器都具有以下几个特性：

- 暂停、停止或待机模式下的复位：可在 MCU 意外进入低功耗模式时避免发生死锁情况，适用于不能处理此类配置的应用
- 硬件/软件看门狗，可以在复位之后立即通过硬件启动看门狗
- 存储器读保护，用于防止对程序内容的窃取
- 存储器写保护，用于保护存储器中包含关键代码的部分。通常，这适用于自举代码或 IAP（应用内编程）驱动程序

这些选项可自动使能安全性和可靠性特性，这样即使在 CPU 获取第一个指令之前出现干扰或攻击，应用也可以恢复。

STM8S 和 STM32 器件具有嵌入式自举加载器，通过它可以使用板上串行接口（例如 UART）重新烧写内部 Flash。随后可以将任何具有串行通讯接口的 PC 用作编程工具，来烧写或更新 Flash 以及数据 EEPROM 存储器的内容。ST 提供了一个软件实用程序来执行自举加载器支持的所有操作。

3.4 安全性

汽车行业首先要求提高基于 MCU 的电子控制系统的可靠性。随后各个工业领域也提出了类似要求，现在，家用电器必须都符合特定标准 IEC60335-1。STM32 和 STM8S 器件均为符合此标准的 B 类器件。使用独立测试机构认证的专用自测库并借助某种特定硬件电路以达到此标准要求。对于具有严格功能安全要求的应用，软件和硬件均有助于大幅减少其开发时间和认证时间。

- 看门狗

  MCU 内置两个看门狗：

  – 窗口看门狗用于监视主循环并检查循环时间是否在给定时间范围内。它基于系统时钟运行。

  – 还可以并行激活一个独立看门狗以提高系统的稳健性。即使主时钟出现故障（例如，由于晶振损坏），此看门狗仍将继续工作。

- 时钟监视

  标准还要求在谐波/分谐波中检测晶振故障或振荡。这通过*第 3.2 节：时钟*中所述的时钟系统实现，以使用内部时钟源定期测量外部晶振或谐振器频率。最后，时钟安全系统 (CSS) 还会监视 HSE 时钟源，并在发生故障时自动将系统时钟切换回内部 HSI 时钟。

3.5 低功耗

除了内核固有的低功耗模式，STM8S 和 STM32 器件都能够在片上系统级别降低总体功耗。

可通过下列方法之一降低运行模式和等待模式下的功耗：

- 降低系统时钟速度：可以根据应用所需的性能调整功耗。这通过时钟控制器中包含的预分频器来实现。
- 当外设时钟未用于降低与时钟树切换活动有关的动态功耗时，对外设时钟进行门控。

两种产品均内置有调压器，可为内部逻辑提供 1.8 V 的电源。这些调压器在运行模式下具有大工作电流（几十 µA），能够提供几 mA 或几十 mA 的电流。为了进一步降低功耗，当为逻辑供电所需的电流在几 µA 时（通常在暂停或停止模式期间），可以将调压器配置为低功耗模式，并尽可能降低其静态功耗。此模式可以提供最低功耗，其唤醒时间稍长于调压器处于运行模式的配置。

4 软件库

ST 的 STM8 和 STM32 MCU 系列的外设兼容性可促进平台设计，非常有助于从一条产品线切换到另一条产品线。但是，当进入开发阶段时，软件支持将至关重要。STM8S 和 STM32 器件均可使用大量软件库，这些软件库为用户提供了适用于所有 MCU 资源的硬件抽象层 (HAL)。此外，C 函数或 API 覆盖了每个控制/状态位。

软件库具有三个抽象层，包括：

1. 一个完整的寄存器地址映射，其中的所有位、位域和寄存器均以 C 语言声明。通过提供此映射，软件库可以使设计人员的任务轻松许多，更为重要的是，它可以提供无错参考映射文件的所有好处，从而缩短早期项目阶段的时间。
2. API 形式的例程和数据结构的集合，它覆盖了所有外设功能。可以直接将此集合用作参考框架，因为它还包含了多个宏，这些宏支持内核相关的固有特性以及常见的常量和数据类型定义。此外，该集合与编译器无关，因此可与任何现有或未来的工具链配合使用。该集合使用 MISRA C 汽车行业标准开发。
3. 一组涵盖所有可用 IP 的示例（目前，STM32 系列有 85 个示例，STM8S 系列有 57 个示例），以及最常见开发工具链的模板项目。通过合适的硬件评估板，入门全新的微控制器仅需几个小时。

如何使用软件库取决于用户。可以选取对设计有用的文件，通过示例来获得培训或快速评估产品。还可以使用 API 来节省开发时间。

我们来看一下几个关键文件和概念。两个独立的库支持 STM8S 和 STM32F 器件。在下面的文件名中，只需根据所选产品将前缀“*stmxxx_*”替换为“*stm32f10x*”或“*stm8s*”。

- *stmxxx_.h*

  此文件是唯一必须包含在 C 源代码（通常为 *main.c*）中的头文件。此文件包含：
  - 数据结构和所有外设的地址映射
  - 用于访问外设寄存器硬件的宏（例如，用于位操作），以及 STM8S 内核内部函数
  - 用于选择目标应用中实施的器件的配置部分。还可以选择是否在应用程序代码（即，基于对寄存器的直接访问的代码，而不是通过 API 驱动程序访问的代码）中使用外设驱动程序
- *stmxxx_conf.h*

  这是外设驱动程序配置文件，其中指定了要在应用中使用的外设，以及一些应用程序特定的参数，如晶振频率。
- *stmxxx_it.c*

  此文件包含要填充的中断服务程序的模板 (IRQ)，但这已经是第一个开发步骤！

了解上述操作原理和文件组织之后，对于简单的应用，用户无需参阅参考手册即可从一个产品虚拟切换到另一个产品。
我们来看一个实例：一个配置为主模式的 SPI 外设，用于对外部 EEPROM 执行读取/写入操作。

下面的*图 4* 和*图 5* 分别显示了 STM8S 和 STM32 产品的初始化代码（使用软件库）。

图 4. STM8S 代码示例

```
/* --------------- 在主模式下初始化 SPI -------------- */
SPI_Init(SPI_FIRSTBIT_MSB,
SPI_BAUDRATEPRESCALER_4,
SPI_MODE_MASTER,
SPI_CLOCKPOLARITY_LOW,
SPI_CLOCKPHASE_2EDGE,
SPI_DATADIRECTION_1LINE_TX,
SPI_NSS_SOFT,
0x07); /* CRC Polynomial */
/* ------------------------- 使能 SPI ----------------------- */
SPI_Cmd(ENABLE);
```

图 5. STM32 代码示例

```
  /* 私有变量 ------------------------------------- */
  SPI_InitTypeDef SPI_InitStructure;

/* ---------------------- SPI1 主模式 ------------------------- */
SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_4;
SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
SPI_InitStructure.SPI_CPOL = SPI_CPOL_Low;
SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;
SPI_InitStructure.SPI_Direction = SPI_Direction_1Line_Tx;
SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
SPI_InitStructure.SPI_CRCPolynomial = 7;
SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
SPI_Init(SPI1, &SPI_InitStructure);
/* ----------------------- 使能 SPI1 ------------------------ */
SPI_Cmd(SPI1, ENABLE);
```

所有参数均相同，对于配置和启动，两个函数调用的过程相似。主要差别在于将参数传递给函数的方式。STM32 器件使用通过地址传递的结构，而对于 STM8S 器件，则是直接传递参数，这样可以在初始化阶段将所需的 RAM 量降至最低（这对于 RAM 低至 1 KB 的器件而言很有必要）。

另一个差别是在使用 STM32 器件时可以为 SPI 指定数据大小（8 位或 16 位）。在上面的示例（*图 5*）中，显式定义了数据大小，但是，库已完善到可以将其忽略：如果此字段在结构中未初始化，默认情况下将使用 8 位数据大小，以保持与 STM8S 器件的兼容性。

5 结论

本应用笔记讨论了简化 8 位 STM8S 与 32 位 STM32 相互转换的要点。在 8 位到 32 位内核性能连续性的基础上，全新的 STM32 和 STM8S MCU 系列具有许多共同的特性。在外设级别，它们共用标准 IP，如定时器和通信接口。在系统级别，它们具有相同的特性，从而减少了外部元件数量（时钟和复位系统、安全特性等）。

还提供了一组软件库对这些共同特性加以补充，这可为新开发的启动带来很大帮助。由于软件库提供抽象层，这些库还可以用作支持 8 位和 32 位 MCU 的统一开发平台的基础。

最后，STM8S 和 STM32 器件的共同特性及其软件库的好处使设计重复使用最大化，并缩短了上市时间，尤其是应用在处理能力、连接性或控制功能复杂度等方面派生出各种要求时。

6 版本历史

表 6. 文档版本历史

日期	版本	变更
2009 年 07 月 28 日	1	初始版本。

请仔细阅读：

中文翻译仅为方便阅读之目的。该翻译也许不是对本文档最新版本的翻译，如有任何不同，以最新版本的英文原版文档为准。

本文档中信息的提供仅与ST产品有关。意法半导体公司及其子公司（“ST”）保留随时对本文档及本文所述产品与服务进行变更、更正、修改或改进的权利，恕不另行通知。

所有ST产品均根据ST的销售条款出售。

买方自行负责对本文所述ST产品和服务的选择和使用，ST概不承担与选择或使用本文所述ST产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为 ST 授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在ST的销售条款中另有说明，否则，ST对ST产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

意法半导体的产品不得应用于武器。此外，意法半导体产品也不是为下列用途而设计并不得应用于下列用途：（A）对安全性有特别要求的应用，例如，生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）汽车应用或汽车环境，且/或（D）航天应用或航天环境。如果意法半导体产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向意法半导体发出了书面通知，采购商仍将独自承担因此而导致的任何风险，意法半导体的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML或JAN正式认证产品适用于航天应用。

经销的ST产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致ST针对本文所述ST产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大ST的任何责任。

意法半导体集团公司

澳大利亚 - 比利时 - 巴西 - 加拿大 - 中国 - 捷克共和国 - 芬兰 - 法国 - 德国 - 中国香港 - 印度 - 以色列 - 意大利 - 日本 - 马来西亚 - 马耳他 - 摩洛哥 – 菲律宾 – 新加坡 - 西班牙 - 瑞典 - 瑞士 - 英国 - 美国

www.st.com