AN4062
应用笔记

STM32F0DISCOVERY 外设固件示例

简介

本应用笔记介绍了为 STM32F0DISCOVERY 套件提供的一些外设固件示例。

这些示例可立即使用，可帮助用户快速了解 STM32F0xx 外设和 STM32F0DISCOVERY 板硬件。每个示例都配备了一些预配置项目，涵盖到 EWARM (IAR Embedded Workbench® for ARM®)、MDK-ARM™、Atollic TrueSTUDIO® 和 Altium TASKING® 工具。

可以从 www.st.com/stm32f0discovery 下载固件应用程序软件包，其中就包含了这些示例。

建议用户首先阅读 STM32F0DISCOVERY *套件*的*软件和固件环境入门* (UM1523) 以熟悉 STM32F0DISCOVERY 套件。

表 1 列出了本应用笔记所涉及的微控制器和开发工具。

表 1. 适用的产品和工具

类型	料号和产品类别
微控制器	STM32F0 系列
开发工具	STM32F0DISCOVERY
软件	STSW-STM32049

目录

1 外设固件示例结构概述 3

2 时钟配置 4

2.1 PLL_SOURCE_HSI 4

2.2 PLL_SOURCE_HSE 4

2.3 PLL_SOURCE_HSE_BYPASS 4

3 外设固件示例说明 5

3.1 GPIO 翻转示例 5

3.2 EXTI 示例 5

3.3 SysTick 示例 5

3.4 待机模式示例 6

3.5 停机模式示例 6

3.6 电流消耗示例 7

3.7 Flash 程序示例 8

3.8 IWDG（独立看门狗）示例 8

3.9 ADC DMA 示例 9

3.10 ADC 低功耗示例 9

3.11 DMA Flash RAM 示例 9

3.12 DMA RAM DAC 示例 10

3.13 DAC 信号生成示例 10

3.14 TIM 互补信号示例 10

3.15 TIM 时基示例 11

3.16 TIM PWM 输入示例 12

3.17 TIM PWM 输出示例 12

3.18 定时器 ADC 触发器示例 13

3.19 NVIC WFI 模式示例 13

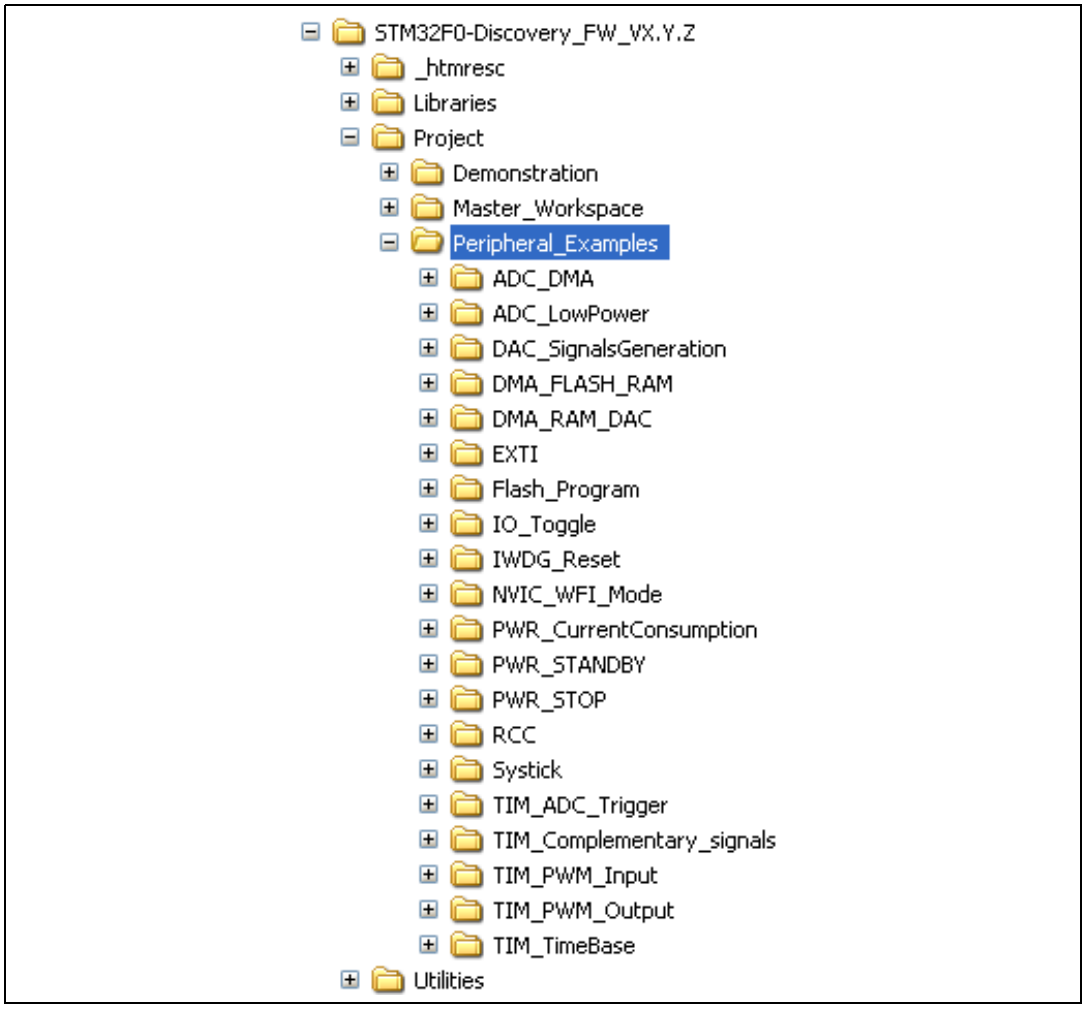
3.20 RCC（复位和时钟控制）示例 14

4 版本历史 15

1 外设固件示例结构概述

在 STM32F0DISCOVERY 固件应用程序软件包中提供了一些外设固件示例，这些示例位于 \Project 文件夹下，如*图 1* 中所示。

图 1. 硬件环境



1. VX.Y.Z 表示软件包版本，例如 V1.0.0。

要运行示例，请使用首选工具打开项目，然后编译，再加载和运行项目。某些示例可能需要额外硬件，如示波器（以进行观测）。有关必需硬件的更多详细信息，请参见每个示例中提供的自述文件。

2 时钟配置

STM32F0-Discovery 套件固件软件包中提供的外设示例以 48 MHz 运行，且使用 HSI 作为时钟源。

用户可以修改配置以使用 HSE（晶振模式或旁路模式）作为时钟源，但是这需要改变 Discovery 套件硬件。

每个示例中提供的“system_stm32f0xx.c”文件都进行了定制，用户可以选择以下三个配置之一（通过取消注释 adequate define）来配合 Discovery 使用。

2.1 PLL_SOURCE_HSI

HSI 时钟信号通过内部 8 MHz RC 振荡器来生成，可直接用作系统时钟，或者经 2 分频用作 PLL 输入。

HSI RC 振荡器的优点是时钟源成本较低（无外部元件），而且启动时间也比 HSE 晶振的启动时间更快。但是，即使是经过了校准，其频率的精度仍然低于外部晶振或陶瓷谐振器。

注： *此配置为默认配置。*

2.2 PLL_SOURCE_HSE

可以通过两种时钟源来生成高速外部时钟信号 (HSE)：

- HSE 外部晶振/陶瓷谐振器
- HSE 用户外部时钟

Discovery 套件未随附 HSE 晶振。要连接到该晶振需要进行一些硬件改造。

有关更多详细信息，请参见“STM32F0 Discovery 套件用户手册 (UM1525)”中的“4.7 OSC 时钟”一节。

2.3 PLL_SOURCE_HSE_BYPASS

在此模式中，将通过外部时钟（频率固定为 8 MHz，来自于 ST-Link 电路）来旁路 HSE。它用于对 PLL 进行时钟控制，并且 PLL 用作系统时钟源。

要通过来自 ST-Link 电路的时钟来旁路 HSE，必须进行某些硬件改造。

有关更多详细信息，请参见“STM32F0 Discovery 套件用户手册 (UM1525)”中的“4.7 OSC 时钟”一节。

3 外设固件示例说明

3.1 GPIO 翻转示例

目的

此示例说明如何使用 BSRR 和 BRR（端口位设置 / 复位寄存器高电平与低电平）以最快速度翻转 I/O。

GPIO 端口连接到 AHB 总线；使用 BSRR 和 BRR 寄存器时，需要两个周期来拉高一个引脚，并需要另外两个周期来拉低该引脚。因此，GPIO 引脚能够以 AHB 时钟的 1/4 进行切换。

说明

在此示例中，PC8 和 PC9（以推挽输出模式配置）不停地翻转：

- 通过在 BSRR 寄存器中设置对应的位来拉高 PC8 和 PC9。
- 通过在 BRR 寄存器中设置对应的位来拉低 PC8 和 PC9。

在本示例中，HCLK 配置为 48 MHz，因此 PC8 和 PC9 以 12 MHz 翻转。

要实现 I/O 最大翻转频率，必须配置编译器选项进行高速优化。

3.2 EXTI 示例

目的

本示例表明如何配置外部中断线。

说明

在本示例中：

- PA0 引脚在悬空输入状态下配置。
- PA0 配置为外部中断源线 0 (EXTI0)。
- EXTI 线 0 配置为在 PA0 引脚上检测到的每个上升沿上生成一个中断。每次按 User（用户）按钮时都会触发中断。
- 在 NVIC（嵌套向量中断控制器）中，配置了 EXIT 线 0 中断优先级并且使能了中断。

当执行程序并且用户按 User（用户）按钮（EXTI0 中断程序）时，将切换 LED3（与 PC9 连接）和 LED4（与 PC8 连接）。

3.3 SysTick 示例

目的

本示例说明如何配置系统节拍定时器以及如何将其用于生成 1 ms 时基。

说明

在本示例中：

- 系统节拍定时器初始化。
- 系统节拍定时器中断在 NVIC 中使能。
- 系统节拍定时器/计数器以自由运行模式启动以产生周期中断。
- 每隔 1 ms 触发一次系统节拍定时器中断。
- 根据系统节拍定时器计数结束事件，实现延迟函数。

两个 LED（LED3 和 LED4）通过延迟函数定义的计时来切换。

3.4 待机模式示例

目的

本示例说明如何将系统置于待机模式以及如何使用外部复位（RTC 闹钟 A）将系统从该模式中唤醒。

说明

在本示例中：

- 系统节拍定时器初始化。
- 系统节拍定时器中断在 NVIC 中使能。
- 系统节拍定时器 / 计数器以自由运行模式启动以产生周期中断。系统节拍计时器中断每隔 250 ms 触发一次。在 SysTick 中断处理器中，将闪烁 LED3；这用于指示 MCU 是处于待机模式还是运行模式。
- EXTI 线 0 配置为在 PA0 引脚上检测到的每个上升/下降沿上产生一个中断。每次 PA0 更改电平（GND 或 VDD）时，都将产生外部中断。
  - 在 EXTI 行上检测到下降或上升沿时，将产生一个中断。在 EXTI 处理器程序中，RTC 配置为在 3 秒内产生闹钟事件，3 秒后，系统将进入待机模式并且 LED3 熄灭。

从待机模式中唤醒后，程序执行将按照复位后的方式来重新启动，RTC 配置（时钟源和预分频器）得到保留且 LED3 再次切换。因此，不需要重新配置 RTC。

LED3 用于监视系统状态，如下所示：

- LED3 闪亮：系统处于运行模式
- LED3 熄灭：系统处于待机模式
- LED3 闪亮：系统从待机模式中恢复

这些步骤以无限循环方式重复。

3.5 停机模式示例

目的

本示例表明如何使用 EXTI 线中断使系统进入停机模式并唤醒。EXTI 线源是 PA.0 和 RTC 闹钟。

说明

在本示例中：

- EXTI Line0 配置为在下降沿上产生中断。
- EXTI line17（RTC 闹钟）配置为在上升沿上产生中断。
- SysTick 编程为每 250 ms 产生一个中断。在 SysTick 中断处理器中，将闪烁 LED3，这表明 MCU 是处于停机模式还是处于运行模式。

系统进入停机模式，等待 RTC 告警（每 5 秒产生一次）或者等待按下用户按键。

- 如果是 RTC 告警 (EXTI_Line17) 将系统从停机中唤醒，则将闪亮 LED3。
- 如果是 User（用户）按钮将系统从停机中唤醒，则 LED4 开启并闪亮 LED3。

LED 用于监视系统状态：

- LED3 闪烁：系统处于运行模式；系统被 RTC 告警从停机中唤醒。
- LED4 开启：系统被 EXTI Line0（User（用户）按钮）从停机中唤醒。

3.6 电流消耗示例

目的

本示例表明如何配置 STM32F0xx 系统以测量不同低功耗模式电流。低功耗模式为：

- 睡眠模式
- 带有 RTC 的停机模式
- 带有唤醒引脚的待机模式（无 RTC）
- 带有 RTC 的待机模式

要选择将要测量的低功耗模式，请取消注释 *stm32f0xx_lp_modes.h* 文件中对应的行。

注： *通过移除跳线 JP2（标记为 IDD）并连接一个电流表，便可以在 STM32F0DISCOVERY 板上测量 STM32F0xx 功耗。*

说明

在复位后，程序等待连接到 PA.00 的 User（用户）按钮被按下，从而进入所选的低功耗模式。

- 使用 RTC 时，将由 RTC 自动产生从低功耗模式的唤醒（5 秒后）信号。
- 在睡眠模式和待机模式中，再次按 User（用户）按钮可退出低功耗模式。

不同低功耗模式配置为：

睡眠模式

- 系统运行在 PLL (48 MHz) 下
- 3 个 Flash 等待周期
- 代码从内部 Flash 运行
- 所有外设关闭
- 使用 EXTI 线唤醒（User（用户）按钮 PA.00）

停机模式
– RTC 由 LSI 提供时钟信号
– 调压器工作在 LP 模式下
– HSI、HSE 和 LSI（如果未用作 RTC 时钟源）关闭
– 无 IWDG
– 处于深度掉电模式的 Flash
– 使用 LSI 提供时钟信号的 RTC 自动唤醒

待机模式
– RTC OFF
– IWDG 和 LSI OFF
– 使用唤醒引脚 (PA.00) 唤醒

包含 RTC（由 LSI 提供时钟信号）的待机模式
– 由 LSI 提供时钟信号的 RTC
– IWDG OFF 和 LSI OFF（如果未用作 RTC 时钟源）
– 使用 LSI 提供时钟信号的 RTC 自动唤醒

3.7 Flash 程序示例

目的

本示例说明如何对 STM32F0xx 内部 Flash 进行编程。

说明

在本示例中：
- 在复位后，将锁定 Flash 程序/擦除控制器。FLASH_Unlock 函数用于将其解锁。
- 在对所需地址进行编程之前，将使用 Flash 擦除扇区功能来执行擦除操作。擦除过程首先计算要使用的扇区数量。将通过调用 FLASH_EraseSector 函数来逐一擦除这些扇区。
- 将使用 FLASH_ProgramWord 函数来执行编程操作。随后将检查写入的数据并且编程操作的结果将存储在 MemoryProgramStatus 变量中。

3.8 IWDG（独立看门狗）示例

目的

本示例说明如何定期更新 IWDG 重载计数器，以及如何模拟软件故障，使编程好的时间段到期时生成 MCU IWDG 复位。

说明

在本示例中：
- 独立看门狗超时设置为 250 ms。
- 系统节拍配置为每 250 ms 产生一个中断。

- 在系统节拍中断服务例程中，将重载独立看门狗计数器以防止独立看门狗复位，并且将闪烁 LED4。
- 连接到 PA0 引脚的 EXTI 线 0 配置为在其下降沿上产生中断。
- 在 NVIC 中，将使能 EXTI 线 0 对应中断向量且优先级等于 0，并且 SysTick 中断向量优先级设置为 1（EXTI 中断优先于 SysTick 中断）。
- EXTI 线用于模拟固件故障：当触发 EXTI 线事件时（在 STM32F0DISCOVERY 板上按下 User（用户）按钮后），将提供对应的中断。在 ISR 中，LED3 熄灭并且 EXTI 线挂起位不会清零。CPU 无限期执行 EXTI 线 ISR 并且永远不会进入系统节拍中断例程，因此不会重载独立看门狗计数器。因此，当独立看门狗计数器达到 00 时，独立看门狗将产生一个复位。

当程序在运行并且产生了独立看门狗复位后，LED4 将在系统恢复操作之后点亮。

3.9 ADC DMA 示例

目的

本示例描述如何使用 ADC1 和 DMA 以将连续转换数据从 ADC1 传输到存储器。

说明

在本示例中：

- ADC1 配置为连续转换电压参考和温度传感器。
- 每次发生转换时，DMA 将以循环方式将转换的数据从 ADC1 DR 寄存器传输到 RegularConvData_Tab[2] 表。

3.10 ADC 低功耗示例

目的

本示例简要介绍了如何将 ADC 外设与自动延迟转换模式和自动关闭电源模式配合使用。

说明

ADC 由连接到 TIM3_Update 事件的 TIM3_TRGO 来触发。每次触发 ADC 时，ADC 会转换连接到 PC.1 的输入电压（对应于 ADC 通道 11），然后 ADC 进入延迟模式（非溢出检测），直至已通过按 USER（用户）按钮读取了 ADC 数据寄存器。

注： *将外部信号（范围是 0V 到 3.3V）连接到要转换的 ADC1 引脚 (PC.01)。*

注： *将电流计连接到 JP2 以测量 I_{DD} 电流。*

3.11 DMA Flash RAM 示例

目的

本示例说明如何使用 DMA 通道将字数据缓冲区从 Flash 传输到嵌入式 SRAM 存储器。

说明

DMA1 通道 1 配置为将 32 字数据缓冲区的内容（存储在 Flash 中）传输到 RAM 中指定的接收缓冲器。

- 传输的启动由软件触发。将使能 DMA1 通道 1 存储器到存储器传输。还将使能源和目标地址递增。
- 通过将 DMA1 通道 1 的通道使能位置 1 来启动传输。
- 在传输结束时，由于使能了中断，因此将产生传输完成中断。一旦产生了中断，便将读取要传输的其余数据，这必须等于 0。然后会将传输完成中断挂起位清零。还将比较源缓冲区与目标缓冲区，以检查是否所有数据都已正确传输。

3.12 DMA RAM DAC 示例

目的

本示例介绍了如何使用 DMA 通道将数据缓冲区从 RAM 存储器传输到 DAC。

说明

DMA1 通道 3 配置为以连续方式将半字缓冲区逐字传输到 DAC 寄存器 DAC_DHR12R。DAC 通道转换配置为由 TIM2 TRGO 触发器触发，并且不生成噪声 / 三角波。由于我们选择访问 DAC_DHR12R 寄存器，因此选择 12 位数据右对齐。

注： *将 PA.04 引脚连接到示波器。*

3.13 DAC 信号生成示例

目的

本示例简要介绍了如何使用 DAC 外设通过 DMA 控制器来生成若干信号。

说明

当用户按下按钮时，DMA 会将所选波形传输到 DAC。

- 每按一次按钮，便已选择了一个信号并且在 DAC 通道 1 中监视。
- 阶梯波形（通道 1）。
- 正弦波形（通道 1）。

注： *使用连接到 PA0 的按钮。*

注： *将 PA4（DAC 通道 1）引脚连接到示波器以监视 DAC 输出波。*

3.14 TIM 互补信号示例

目的

本示例说明如何配置 TIM1 外设以生成三个互补的 TIM1 信号，并分别用于插入定义的死区值，使用刹车功能以及锁定所需参数。

说明

TIM1CLK 固定为 SystemCoreClock，TIM1 预分频器等于 0，因此使用的 TIM1 计数器时钟为 SystemCoreClock (48 MHz)。

目标是以 17.57 KHz 生成 PWM 信号：

- TIM1_Period = (SystemCoreClock / 17570) - 1

三个占空比的计算方式如下：

  – 通道 1 占空比设置为 50%，因此通道 1N 设置为 50%。
  – 通道 2 占空比设置为 25%，因此通道 2N 设置为 75%。
  – 通道 3 占空比设置为 12.5%，因此通道 3N 设置为 87.5%。

定时器脉冲的计算方式如下：

- ChannelxPulse = DutyCycle * (TIM1_Period - 1) / 100

将在两个不同互补信号之间插入一个等于 11/SystemCoreClock 的死区，并且选择锁定电平 1。中断极性在高电平使用。

可以使用示波器来显示 TIM1 波形。

- 将 TIM1 引脚连接到示波器以监视不同波形：
  – TIM1_CH1 引脚 (PA.08)
  – TIM1_CH1N 引脚 (PB.13)
  – TIM1_CH2 引脚 (PA.9)
  – TIM1_CH2N 引脚 (PB.14)
  – TIM1_CH3 引脚 (PA.10)
  – TIM1_CH3N 引脚 (PB.15)
- 将 TIM1 刹车引脚 TIM1_BKIN 引脚 (PB.12) 连接到 GND。要产生刹车事件，请将此引脚电平从 0V 切换到 3.3V。

3.15 TIM 时基示例

目的

本示例说明如何在输出比较定时模式中配置 TIM 外设（包含每个通道的对应中断请求），以生成两个不同时基。

说明

TIM3CLK 频率设置为 SystemCoreClock (Hz) 以使 TIM3 计数器时钟为 6 MHz，因此预分频器的计算方式如下：

- 预分频器 = (TIM3CLK / TIM3 计数器时钟) - 1。
- SystemCoreClock 设置为 48 MHz。
- TIM3 CC3 寄存器等于 13654，CC3 更新率 = TIM3 计数器时钟 / CCR3_Val = 439.4 Hz。因此，TIM3 通道 3 每隔 2.27 ms 产生一个中断。
- TIM3 CC4 寄存器等于 6826，CC4 更新率 = TIM3 计数器时钟 / CCR4_Val = 878.9 Hz。因此，TIM3 通道 4 每隔 1.13 ms 产生一个中断。

当计数器值达到输出比较寄存器值时，将产生输出比较中断，并且在处理器例程中，分别连接到 PC08 和 PC09 的 2 个 LED（即 LED3 和 LED4）将按照以下频率闪亮：

– LED3(PC09)：219.7 Hz (CC3)
– LED4(PC08)：439.4 Hz (CC4)

3.16 TIM PWM 输入示例

目的

本示例说明如何使用 TIM 外设来测量外部信号的频率和占空比。

说明

TIMxCLK 频率设置为 SystemCoreClock，预分频器为 0，因此计数器时钟等于 SystemCoreClock。STM32F0xx 器件的 SystemCoreClock 设置为 48 MHz。

TIM2 在 PWM 输入模式下配置：外部信号连接到用作输入引脚的 TIM2 通道 2。

为了测量频率和占空比，我们使用 TIM2 CC2 中断请求，以便在 TIM2_IRQHandler 例程中计算外部信号的频率和占空比。

变量“Frequency”包含外部信号频率：

- TIM2 计数器时钟 = SystemCoreClock。
- Frequency = TIM2 计数器时钟 / TIM2_CCR2（以 Hz 为单位）。

变量“DutyCycle”包含外部信号占空比：

– DutyCycle = (TIM2_CCR1*100) / (TIM2_CCR2)（百分比形式）。

要测量的最小频率值是 732 Hz（TIM2 计数器时钟/CCR 最大值）。

注： *连接外部信号以测量 TIM2 CH2 引脚 (PA.01)。*

3.17 TIM PWM 输出示例

目的

本示例说明如何配置 TIM1 外设以生成具有四个不同占空比（50%、37.5%、25% 和 12.5%）的 PWM 信号。

说明

TIMxCLK 频率设置为 SystemCoreClock，预分频器为 0，因此计数器时钟等于 SystemCoreClock。STM32F0xx 器件的 SystemCoreClock 设置为 48 MHz。

目标是以 17.57 KHz 生成 PWM 信号：

– TIM1_Period = (SystemCoreClock / 17570) - 1
– TIM1 Frequency = TIM1 计数器时钟 / (ARR + 1) = 48 MHz / 2730 = 17.57 kHz

- TIM1 CCR1 寄存器值等于 1364，因此 TIM1 通道 1 生成一个频率等于 17.57 kHz 且占空比等于 50% 的 PWM 信号：

  TIM1 通道 1 占空比 = (TIM1_CCR1 / TIM1_ARR + 1)* 100 = 50%

- TIM1 CCR2 寄存器值等于 1023，因此 TIM1 通道 2 生成一个频率等于 17.57 kHz 且占空比等于 37.5% 的 PWM 信号：

  TIM1 通道 2 占空比 = (TIM1_CCR2 / TIM1_ARR + 1)* 100 = 37.5%
- TIM1 CCR3 寄存器值等于 682，因此 TIM1 通道 3 生成一个频率等于 17.57 kHz 且占空比等于 25% 的 PWM 信号：

  TIM1 通道 3 占空比 = (TIM1_CCR3 / TIM1_ARR + 1)* 100 = 25%
- TIM1 CCR4 寄存器值等于 431，因此 TIM1 通道 4 生成一个频率等于 17.57 kHz 且占空比等于 12.5% 的 PWM 信号：

  TIM1 通道 4 占空比 = (TIM1_CCR4 / TIM1_ARR + 1)* 100 = 12.5%

注： *可以使用示波器来显示 PWM 波形。*

将 TIM1 引脚连接到示波器以监视不同波形：

- TIM1_CH1 引脚 (PA.08)
- TIM1_CH2 引脚 (PA.09)
- TIM1_CH3 引脚 (PA.10)
- TIM1_CH4 引脚 (PA.11)

3.18 定时器 ADC 触发器示例

目的

本示例说明如何配置 TIM 以触发 ADC 转换。

说明

在本示例中，TIM1 在 PWM 模式下配置，TIM1 CC4 事件用于触发 ADC。

ADC 配置为连续转换 ADC_Channel_11（连接到外部电压）。

每次发生 TIM1 CC4 事件时，ADC 都将转换可变电压。

注： *将范围在 0V 到 3.3V 之间的外部信号连接到 ADC 引脚 (PC.01)。*

3.19 NVIC WFI 模式示例

目的

本示例说明如何进入 WFI 模式以及如何通过用户键中断从该模式中唤醒。

说明

在关联的软件中，系统时钟设置为 48 MHz。用户按 User Key（用户键）按钮后，MCU 将进入 WFI 模式。如果用户再次按 User Key（用户键），将通过某个频率（取决于系统时钟）来闪烁 LED3。这用于指示 MCU 处于 WFI 模式还是运行模式。

- 按 key（键）按钮（在 EXTI Line0 上生成上升沿）会将内核置于 WFI 模式，从而导致 LED3 停止闪烁。
- 要从 WFI 模式中唤醒，必须再次按该按钮，这将产生一个中断，从而使系统退出 WFI 模式。LED3 重新启动闪烁。

注： *按 User（用户）按钮可进入和退出 WFI 模式。*

3.20 RCC（复位和时钟控制）示例

目的

本示例说明如何执行以下操作：

- 将 HSE（高速时钟）配置为 RCC 时钟
- 使用时钟安全性系统 (CSS) 功能来生成 NMI 中断
- 在 MCO 上输出系统时钟

说明

出于调试目的，RCC_GetClocksFreq() 函数用于得到不同片上时钟的当前状态和频率。

您可以使用工具调试器来查看 RCC_ClockFreq 结构内容（其中保存了不同片上时钟的频率）。

本示例还处理高速外部时钟 (HSE) 故障检测：当 HSE 时钟消失时（断路或者与外部晶体振荡器断开连接），将禁止 HSE 和 PLL（但是不会对 PLL 配置进行任何更改），HSI 选为系统时钟源并生成中断 (NMI)。在 NMI ISR 中，将使能 HSE 和支持 HSE 的中断。HSE 时钟恢复后，将生成 HSERDY 中断，并且在 RCC ISR 例程中，会将系统时钟重新配置为 HSE 时钟故障之前的状态。可以在 MCO 引脚 (PA.8) 上监视系统时钟。

两个 LED 按照延迟函数定义的定时来闪烁。

4 版本历史

表 2. 文档版本历史

日期	版本	变更
2012 年 03 月 23 日	1	初始版本。
2012 年 05 月 14 日	2	增加了*第 2 节：时钟配置*。
2013 年 03 月 28 日	3	更新了*表 1：适用的产品和工具*和*第 3.1 节：GPIO 翻转示例*。

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。

ST和ST徽标是ST在各个国家或地区的商标或注册商标。

本文档中的信息取代之前提供的所有信息。

ST徽标是意法半导体公司的注册商标。其他所有名称是其各自所有者的财产。

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