AN12313

KE16Z PDB 在 back-to-back 模式下触发 ADC 转换

版本 0 — 2018 年 12 月

应用笔记

目录

1 简介

可编程延迟块（PDB）利用内部、外部触发或可编程的计时器实现可控延迟，从而可以实现 ADC 转换的精确时序。PDB 模块支持一个可配置通道用于 ADC 硬件触发，该通道与一个 ADC 模块关联。对于每个 PDB 通道，都有一个触发输出信号作为 ADC 模块的硬件触发信号，以启动 ADC 的模数转换。并且，每个 PDB 通道有多达四个预触发信号输出到 ADC 模块，可以实现四个不同的 ADC 通道转换。PDB back-to-back 操作模式使得 ADC 转换完成信号（COCO 标志）能够触发下一个 PDB 通道的预触发并自动输出触发信号，无需任何额外的外部触发输入到 PDB 模块。在某些实际应用中，例如电机控制，必须在几乎同一时间采样两个 ADC 通道，以获取正确的模拟值。对于某些具有多个 ADC 模块的 MCU 来说，这不是问题，因为每个 ADC 模块都可以采样一个 ADC 输入，所以两个 ADC 模块可以并行采样两个模拟输入。而 KE16Z 只有一个 ADC 模块，但它必须一个接一个地实现模数转换。所以，两次 ADC 转换之间的间隔变得很关键。在这种情况下，PDB 的 back-to-back 模式是一个很好的解决方案，因为它可以在上一个 ADC 通道转换完成后立即开始另一个 ADC 通道的转换。

本应用笔记描述了 PDB 和 ADC 的内部连接，并提供了一个应用案例来帮助用户了解 back-to-back 模式下怎么配置 PDB 和 ADC。该应用案例的实现基于 IAR 嵌入式 Workbench 8.30.1 开发环境，KE16Z SDK 2.4.0 软件，FRDM-KE16Z 开发板。

2 功能介绍

2.1 概述

PDB 模块是一个为 ADC 的硬件触发输入提供可控制延迟的计时器，可以在 ADC 各通道转换之间提供精确的时序。PDB 使用内部或外部触发输入来开始计数。在本文的应用案例中，通过 TRGMUX 模块实现 PDB 触发源选择。FTM 通道的匹配触发信号或初始化触发信号都可以作为 PDB 的触发输入源，并且 PDB 的一个通道可以和一个 ADC 模块相连。一个 PDB 通道支持四个预触发，可用于选择不同的 ADC 通道。模块概述如 图 1 所示。

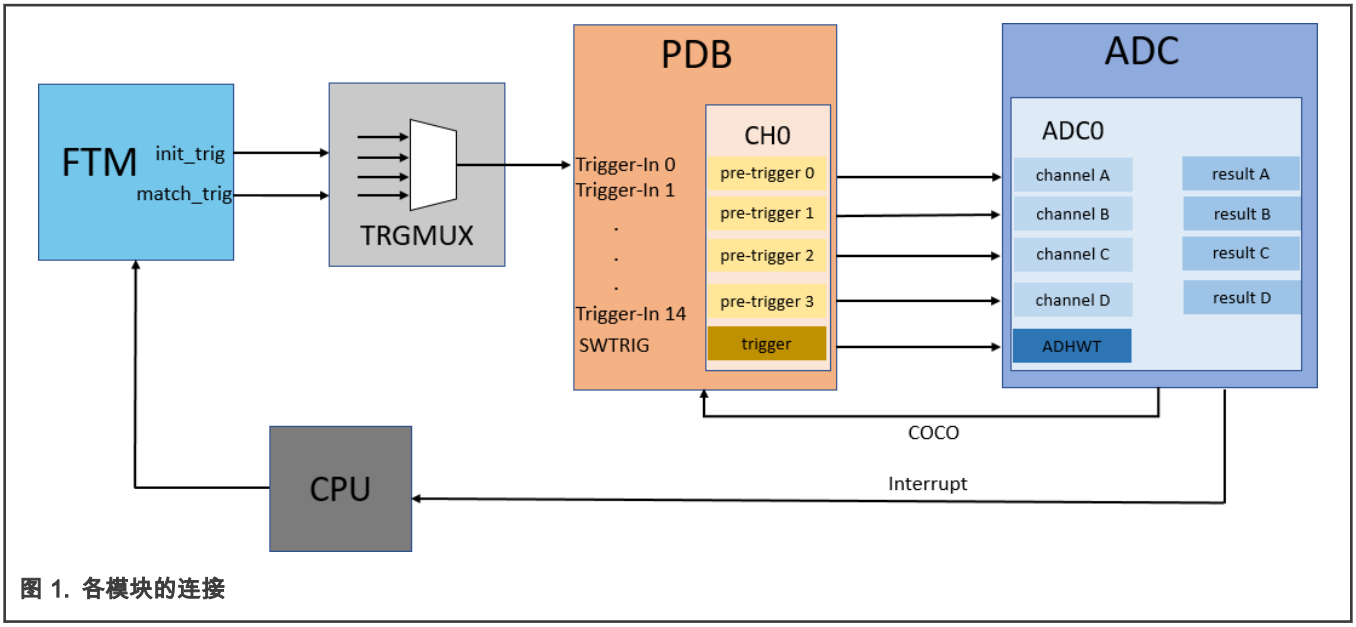

图 1. 各模块的连接

2.2 back-to-back 模式的连接

在 PDB 单元内部以环的形式来实现 PDB 的 back-to-back 模式的连接。back-to-back 模式下，需要先接收到一个 ADC 转换完成的标志，以触发下一个 PDB 预触发输出。PDB 一旦接收到触发输入信号，便会在预定义的延迟时间后生成第一个预触发输出。下文所示的 PDB back-to-back 模式是基于配置 PDB0_CH0C1_BB 位实现的。

- 当 BB [0] = 1 时，PDB0 通道 0 的预触发 0 的输入是：ADC0SC1D_COCO;
- 当 BB [1] = 1 时，PDB0 通道 0 的预触发 1 的输入是：ADC0SC1A_COCO；
- 当 BB [2] = 1 时，PDB0 通道 0 的预触发 2 的输入是：ADC0SC1B_COCO；
- 当 BB [3] = 1 时，PDB0 通道 0 的预触发 3 的输入是：ADC0SC1C_COCO。

在本文的应用案例中，FTM0 产生一个周期性的硬件触发信号以启动 PDB0，写 PDB0_CH0C1_BB 等于写入 0xE。back-to-back模式的使能位和预触发之间的映射如 表 1 所示。back-to-back 模式下的内部连接图如 图 2 所示。

表 1. 使能位和预触发之间的映射

back-to-back 使能位	PDB 通道预触发
BB[0]	pre-trigger 0
BB[1]	pre-trigger 1
BB[2]	pre-trigger 2
BB[3]	pre-trigger 3

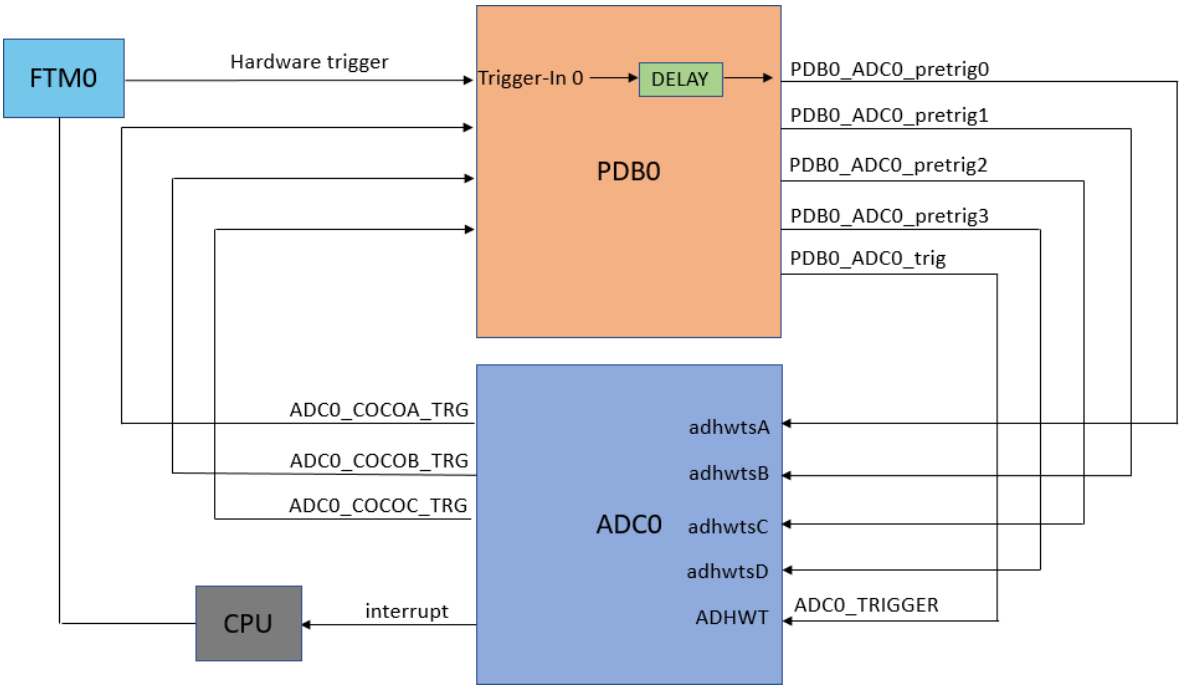

图 2. back-to-back 模式下的内部连接图

2.3 PDB 预触发和触发输出的时序

KE16Z MCU 仅具有一个 PDB 模块和一个 ADC 模块。PDB0 通道 0 与 ADC0 模块关联，并且 PDB0 通道 0 具有四个预触发输出。每个预触发输出和触发输出都连接到 ADC 硬件输入。 预触发用于在实际触发发生之前对 ADC 模块进行预处理。当 ADC 接收到触发信号的上升沿时，ADC 将根据预触发条件确定的前提条件开始转换。

ADC 模块有四组配置和结果寄存器，例如 ADC_SC1A 和 ADC_RA。 因此，ADC 可以在四个不同的模拟输入源之间交替转换。PDB 预触发输出用于指定接下来哪个信号会被 ADC 采样。当预触发 m 置位时，根据第 m 组配置寄存器的配置来触发 ADC 转换，转换结果会存储在 m 组的结果寄存器中。将 PDB0_CH0C1_BB 设置为 0xE，PDB0 至 ADC0 触发的预触发和触发输出的时序图，如 图 3 所示。

在 back-to-back 模式下，PDB0 触发 ADC0 转换的过程分为以下步骤：

1. FTM0 产生周期性的通道匹配触发信号以启动 PDB0。
2. 当外部触发信号输入到 PDB0 时，经过预定义的延迟时间后生成预触发 0，通过配置 PDB0_CH0DLY0_DLY 来设置预定义延迟时间。此预触发 0 的信号输出后可以触发一个 ADC0 通道转换。
3. BB [1] 设置为 1，ADC 转换完成的标志位 COCOA，会触发 PDB0 的预触发 1。此预触发 1 的信号输出后可以触发一个 ADC0 通道转换。
4. BB [2] 设置为 1，ADC 转换完成的标志位 COCOB，会触发 PDB0 的预触发 2。此预触发 2 的信号输出后可以触发一个 ADC0 通道转换。
5. BB [3] 设置为 1，ADC 转换完成的标志位 COCOC，会触发 PDB0 的预触发 3。此预触发 3 的信号输出后可以触发一个 ADC0 通道转换。
6. 在一个 FTM0 周期之后，FTM0 会向再产生一个触发信号去触发 PDB0，以启动下一轮的 ADC0 转换。

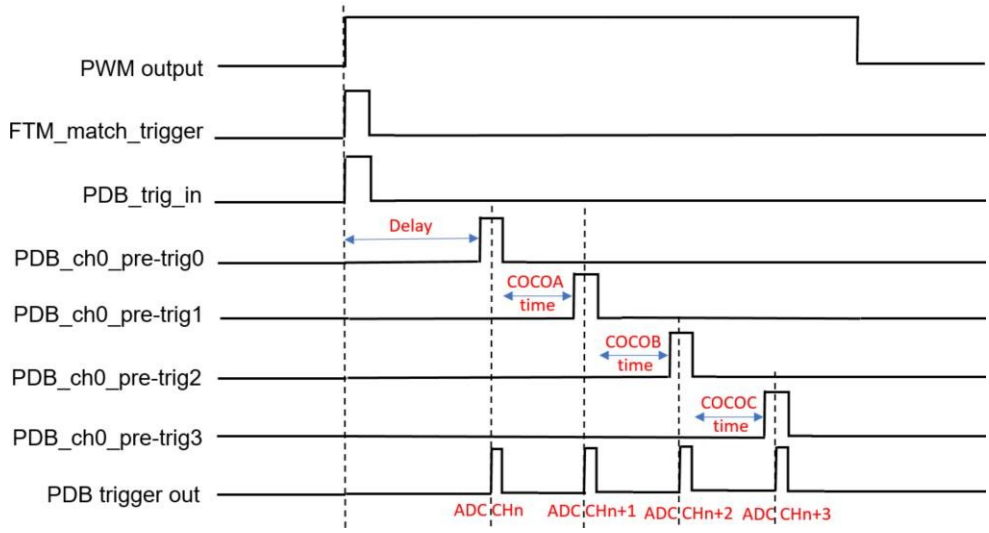

图 3. PDB 的触发、预触发输出信号触发 ADC 转换的时序图

3 实现细节

下文介绍的是一个 PDB 在 back-to-back 模式下触发多个 ADC 通道转换的应用案例的实现细节。在本次的应用案例中，必须配置 FTM0，PDB0，ADC0 三个外设来实现。FTM0 产生一个周期性的外部触发信号，以启动 PDB0 通道 0 触发信号。然后，PDB0 在 back-to-back 模式下触发四个 ADC 通道交替转换。

3.1 FTM 配置

配置 FTM0 模块产生通道匹配触发信号。KE16Z 的 TRGMUX 模块可以实现各模块之间内连。在本次的应用案例中，FTM 的触发信号会作为一个外部的输入信号连接到 TRGMUX 的输入，TRGMUX 的输出作为 PDB0 触发的输入。必须通过配置 TRGMUX，来实现 FTM0 和 PTB0 触发信号的连接。

```
/* select FTM0 hardware trigger as PDB0 trigger input */
TRGMUX_SetTriggerSource(TRGMUX0, kTRGMUX_Pdb0, kTRGMUX_TriggerInput0, kTRGMUX_SourceFtm0);
```

通过配置 TRGMUX，将 FTM0 硬件触发输出作为 PDB0 的触发输入。FTM0 的触发输出信号连接到 PDB0 的 Trigger-In 0。

初始化 FTM0 以输出边沿对齐的 PWM 信号，PWM 的频率配置为 100 Hz。通过将 FTM0_EXTTRIG_CH0TRIG 设置为 1，选择 FTM0 通道 0 以产生外部触发。

供参考的初始化代码如下：

```
/* Initialize FTM module.
*/ FTM_GetDefaultConfig(&ftmConfigStruct);
ftmConfigStruct.extTriggers = kFTM_Chnl0Trigger; /* Enable to output the trigger. */
FTM_Init(DEMO_FTM_BASE, &ftmConfigStruct);

/* Configure ftm params with frequency 100Hz */
pwmParam.chnlNumber = kFTM_Chnl_0;
pwmParam.level = pwmLevel;
pwmParam.dutyCyclePercent = 50U; /* Percent: 0 - 100. */
pwmParam.firstEdgeDelayPercent = 0U;
FTM_SetupPwm(DEMO_FTM_BASE, &pwmParam, 1U, kFTM_EdgeAlignedPwm, DEMO_FTM_PWM_HZ,
DEMO_FTM_COUNTER_CLOCK_HZ);
```

3.2 ADC 配置

PDB0 在 back-to-back 模式下触发四个 ADC0 通道交替转换。除了 ADC0 时钟选择，判断和一些基本的初始化配置外，在本次应用案例中还需要注意几点。

将 ADC0_SC2_ADTRG 设置为 1 可以使能 ADC0 的 PDB0 硬件触发。ADC0 包含四组配置和结果寄存器：A，B，C，D，将 ADC0_SC1m_ADCH 设置为通道号可以选择外部通道作为 ADC0 输入，将 ADC0_SC1D_AIEN 设置为 1 可以启用转换完成中断。

```
/* Enable hardware trigger mode. */
ADC12_EnableHardwareTrigger(DEMO_ADC_BASE, true);

/* Configure the ADC12 conversion channels and interrupt. */
adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = false;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP, &adc12ChannelConfigStruct);

adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = false;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL+26;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP+27, &adc12ChannelConfigStruct);

adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = false;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL+29;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP+2, &adc12ChannelConfigStruct);

adc12ChannelConfigStruct.enableInterruptOnConversionCompleted = true;
adc12ChannelConfigStruct.channelNumber = DEMO_ADC_USER_CHANNEL+30;
ADC12_SetChannelConfig(DEMO_ADC_BASE, DEMO_ADC_CHANNEL_GROUP+3, &adc12ChannelConfigStruct);
```

3.3 PDB 配置

PDB0 产生预触发，触发信号输出到 ADC0。为了使 PDB0 在双向模式下正常工作，在配置 PDB0 时需要注意几点。在以下内容中将提供有关 PDB0 的配置详细信息：

1. **时钟选择**

   PDB 模块的唯一时钟源是系统时钟。通过将 PCC_PDB0_CGC 设置为 1，确保 PDB0 开启了系统时钟。

   ```
   /* Enable PDB0 clock. */
   CLOCK_EnableClock(kCLOCK_Pdb0);
   ```

2. **启用并选择 PDB0 的触发源**

   PDB0 的某些功能由 PDB0_SC 寄存器配置。当 PDB0_SC_LDOK 设置为 1 后，在检测到触发输入事件时， PDB0 会将缓冲区的值加载到内部寄存器中。因此，该位可以最后写入以更新 MOD，IDLY 和其它之前已经写入寄存器内部缓冲区的值。

   将 PDB0_SC_PDBEN 设置为 1 以启用 PDB 功能。PDB0 包含一个计数器，该计数器用于实现各个 PDB 通道的延时，当启用 PDB0 时，该计数器将复位并启动。

   将 PDB0_SC_LDMOD 设置为 0，可以在将 1 写入 PDB0_SC_LDOK 之后立即将缓冲区中的值加载到内部寄存器中。

   将 PDB0 _SC_PRESCALER 和 PDB0_SC_MUTL 设置成适当的值，对计数器的时钟进行分频，它们用于计算延迟时间。

   PDB0 通道 0 有高达 15 个触发输入源，将 PDB0_SC_TRGSEL 设置为 0，以选择 Trigger-In 0 作为触发输入源。

   ```
    /* Initialize PDB module. */
   PDB_GetDefaultConfig(&pdbConfigStruct);
   /* The trigger would be selected by TRGMUX. */
   pdbConfigStruct.triggerInputSource = kPDB_TriggerInput0;
   PDB_Init(DEMO_PDB_BASE, &pdbConfigStruct);
   ```

3. **PDB0 模数和通道延迟时间配置**

不使用 PDB 的 mod，但必须将其设置为适当的值。它应该大于延迟值。延迟值确定预触发的延迟时间。在本应用案例中，将 mod（PDB0_MOD_MOD）设置为 2000，并将通道 0 的预触发 0 延迟值（PDB0_CH0DLY0_DLY）设置为 500。在 back-to-back 模式下，无需为其他预触发配置延迟值。预触发的通道延迟时间由时钟，预分频比，乘法因子和延迟值确定。

```
/* Configure the PDB mod. */
PDB_SetModulusValue(DEMO_PDB_BASE, DEMO_PDB_MODULO_VALUE);
/* Configure the PDB pre-Trigger 0 delay time. */
PDB_SetADCPreTriggerDelayValue(DEMO_PDB_BASE, DEMO_PDB_TRIGGER_CHANNEL,
DEMO_PDB_PRETRIGGER_CHANNEL, DEMO_PDB_PRETRIGGER_DELAY_VALUE);
```

4. **back-to-back 模式配置**

   将 PDB0_CH0C1_EN 设置为 0xF，以使能 PDB0 通道 0 的四个预触发。

   将 PDB0_CH0C1_TOS [0] 设置为 1 ，当检测到选定的触发输入信号的上升沿之后，经过计数器计数延迟的时间加上一个外设时钟周期的时间后，预触发 0 就会置位。其他三个预触发处于旁路模式。

   将 PDB0_CH0C1_BB 设置为 0xE，在 back-to-back 模式下启用预触发 1，预触发 2 和预触发 3。当第一个 ADC 转换完成标志 COCOA 时置位时，它可以触发下一个 PDB0 的预触发，并且此预触发输出可以触发下一个 ADC 转换。

```
/* Configure the ADC Pre-Trigger. */
  PDB_SetADCPreTriggerDelayValue(DEMO_PDB_BASE, DEMO_PDB_TRIGGER_CHANNEL,
DEMO_PDB_PRETRIGGER_CHANNEL, DEMO_PDB_PRETRIGGER_DELAY_VALUE);

  /* PDB Channel Back-to-back Enable / PDB Channel Pre-Trigger Enable / PDB Channel Pre-Trigger
Output Select */
  pdbAdcPreTriggerConfigStruct.enablePreTriggerMask = (1U << DEMO_PDB_PRETRIGGER_CHANNEL)|(1U <<
DEMO_PDB_PRETRIGGER_CHANNEL+1)|(1U << DEMO_PDB_PRETRIGGER_CHANNEL+2)|(1U <<
DEMO_PDB_PRETRIGGER_CHANNEL+3);
  pdbAdcPreTriggerConfigStruct.enableOutputMask = (1U << DEMO_PDB_PRETRIGGER_CHANNEL);
  pdbAdcPreTriggerConfigStruct.enableBackToBackOperationMask = (1U <<
DEMO_PDB_PRETRIGGER_CHANNEL+1)
|(1U << DEMO_PDB_PRETRIGGER_CHANNEL+2)|(1U << DEMO_PDB_PRETRIGGER_CHANNEL+3);
  PDB_SetADCPreTriggerConfig(DEMO_PDB_BASE, DEMO_PDB_TRIGGER_CHANNEL,
&pdbAdcPreTriggerConfigStruct);
```

5. **加载完成**

   模数（Modulus）寄存器，通道 0 延迟 0 寄存器和其它寄存器都是有内部缓冲区的，写入该寄存器的任何值都将写入其内部缓冲区。仅在将 1 写入 PDB0_SC_LDOK 位之后，才将该寄存器的内部缓冲区中的值加载到该寄存器中。

```
 /* Load PDB counter register. */
PDB_DoLoadValues(DEMO_PDB_BASE);
```

4 结论

本应用笔记介绍了 FTM，PDB 和 ADC 的互连，并介绍了如何使用 PDB 的 back-to-back 模式来实现周期性触发的四个 ADC 通道转换。用户可以通过本文提供的参考配置细节轻松地实现该功能。

5 参考资料

恩智浦网站上提供以下参考资料：

1. *KE16Z Reference Manual*（文档 KE1xZP48M48SF0RM）
2. 《在 KE1xF 上使用 FTM、PDB 和 ADC 驱动双 PMSM FOC 和 PFC》（文档 AN5380）
3. *Synchronize Analog Modules and Timers with PDB Modules in MC9S08MP16*（文档 AN4424）
4. *Tips and Tricks Using PDB in Motor Control Applications on Kinetis*（文档 AN4822）

5. *PDB Driver for the MC9S08GW64* (AN4163)

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2018 年 12 月

Document identifier: AN12313

arm