Document Number: AN4419
Rev 0, 01/2012

在 Kinetis 系列器件上使用 DMA 和 GPIO 模拟定时器功能

作者: Donnie Garcia， Hiroki Maruoka
微控制器解决方案部

内容

1 简介 1
1.1 特性 1
2 实现细节 2
2.1 时钟 2
2.1.1 初始化 2
2.1.2 过程 2
2.2 PWM 3
2.2.1 初始化 3
2.2.2 过程 4
3 限制 5
4 示例 6
4.1 环境 6
4.2 示例代码 6
4.2.1 时钟 6
4.2.2 PWM 7
4.3 结果 9
5 结论 10

1 简介

当今的嵌入式应用需要不断提高功能级别。MCU 执行的其中一项关键功能是定时器功能。Freescale Kinetis 器件现已包含多个定时器外设。例如，MK 器件包含 3 个 FTM 模块（8ch、2ch、2ch）、1 个 RTC、PIT 通道、CMT 和 LPTMR。即便已经具备以上这些丰富的定时器资源，而在实际的应用中还需要更多额外的定时器。

本应用笔记介绍“如何在 Kinetis 上使用 PIT、DMA 和 GPIO 生成 PWM 或时钟”。一般来说，这些功能是通过定时器外设来实现的。Kinetis 内部集成了强大的 FlexTimer（FTM）定时器模块。

但是，用户可以使用本应用笔记中的方法控制更多定时器。

1.1 特性

- 时钟
  - 时钟可由 1ch PIT、1ch DMA 和 GPIO 引脚生成
  - 时钟占空比为 50%。
  - 时钟最大频率为 PIT 计数器频率的二分之一。PIT 最大频率为总线时钟频率。
- PWM

- PWM 可由 1ch PIT、2ch DMA 和 GPIO 引脚生成
- PWM 占空比可在 0%至 100%之间变化。当占空比为 0%时，GPIO 输出为 L；当占空比为 100%时，GPIO 输出为 H

2 实现细节

本章节介绍各功能的实现细节。

2.1 时钟

2.1.1 初始化

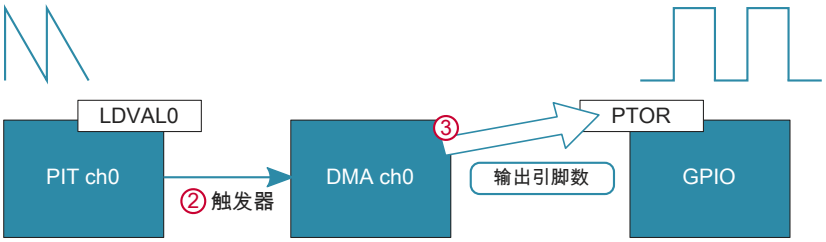
用户需要按照下述步骤初始化各模块:

1. GPIO
   - 在引脚控制寄存器(PCR)中打开 PORTx CLOCK，并使能引脚的中断功能。将引脚 Mux 设置使能为 GPIO 功能。将中断配置设置为触发上升或下降沿上的 DMA 请求。
   - 设置端口数据方向，PDDR 为输出。
2. PIT
   - 启动 PIT 时钟。置位 PIT 模块控制寄存器 MDIS 位。MDIS 为禁用模块。用于禁用模块时钟。必须在执行任何其他设置之前使能该位。
   - 设置 LDVAL 寄存器初始值。载入该寄存器中的 PIT 计数器值。
3. DMA
   - 打开 DMA Mux 时钟。
   - 设置 DMAMUX 通道配置寄存器。将 SOURCE 位段配置为要访问的 GPIO 的端口号。
   - 按照如下所示设置传输控制描述符:
   - saddr：GPIO 输出端口位配置信息存放地址。
   - daddr：GPIO 端口值翻转寄存器地址。
   - nbyte：4
   - tcdAtter：ssize 和 dsize 均为 2
   - channelno：0
   - 置位 DMA 使能请求寄存器

2.1.2 过程

1. 启动 PIT 定时器。
2. PIT 计数器变为 0 时，PIT 触发 DMA。PIT 计数器加载 LDVALn 寄存器。
3. DMA 翻转 GPIO 输出信号。可通过访问 PTOR 寄存器翻转 GPIO 输出信号。
4. 自动重复步骤 2 和 3。

下表为过程概述。PIT 为已使用的通道 0（ch0），DMA 为已使用的 ch0。PIT chn 将触发 DMA chn。如果 PIT 计数器值变为 0，则 PIT 将触发 DMA 并加载 LDVAL 的计数器值。DMA 将翻转 GPIO 输出信号。



LDVAL0：载入此寄存器值时，PIT ch0将向下计数。
PTOR：如果在该寄存器位写入1，则会切换相关输出引脚。

图 1. 时钟过程概述

2.2 PWM

2.2.1 初始化

用户需要按照下述步骤初始化各模块:

1. GPIO
   - 设置引脚控制寄存器（PCR）的引脚复用控制和中断配置。将引脚复用设置为 GPIO 功能。将中断配置设置为任一边沿上的 DMA 请求。
   - 设置端口数据方向为输出，PDDR。
2. PIT
   - 启动 PIT 时钟
   - 置位 PIT 模块控制寄存器 MDIS 位。MDIS 为禁用模块，用于禁用模块时钟。必须在对 PIT 模块执行任何其他设置之前使能该位。
   - 设置初始 LDVAL 寄存器值。载入该寄存器中的 PIT 计数器值。
3. DMA
   - 启动 DMA 多路复用器时钟。
   - 设置 DMA 多路复用器通道配置寄存器。将 SOURCE 位段配置为要访问的 GPIO 的端口号。
   - 按照下述步骤设置传输控制描述符:
     - tcd0
       - saddr: GPIO 输出端口位配置信息存放地址。
       - daddr: GPIO 端口值翻转寄存器地址。
       - nbyte: 4
       - tcdAtter: ssize 和 dsize 均为 2
       - csr: 设置链路通道为 tcd1
       - channelno: 0~4
     - tcd1
       - saddr: 下一个 PIT 计数器值地址。
       - daddr: PIT LDVAL 地址的地址。
       - nbyte: 4
       - tcdAtter: ssize 和 dsize 均为 2
       - soff: 4
       - slast: -8
       - loopcount: 2
       - channelno: 5~
     - 置位 DMA 使能请求寄存器

2.2.2 过程

1. 启动 PIT 定时器。
2. PIT 计数器变为 0 时，PIT 触发 DMA。PIT 计数器加载 LDVALn 寄存器。
3. DMA 翻转 GPIO 输出信号。可通过访问 PTOR 寄存器翻转 GPIO 输出信号。该 DMA 与其他 DMA 链接在一起，因此，将会触发下一个 DMA，以载入新的 LDVALn 寄存器值。
4. 下一个 DMA 将新的值存储到 LDVALn 寄存器中。然后，用户将 2 个 32bit 的字依次传递到 PIT 的 LDVAL 寄存器中作为向下计数的值。在奇数循环中，第一个字存储在 LDVAL 寄存器中，在偶数循环中，第二个字存储在 LDVAL 寄存器中。
5. 重复步骤 2 至 4。

注

占空比和频率由 LDVAL 寄存器中存储的 2 个字阵列值决定。

下图为 PWM 过程概述。demo 中使用到了 PIT 的通道 0 (ch0)，DMA 的通道 0 (ch0) 和通道 5 (ch5)。PIT chn 将触发 DMA chn。Kinetis 具有 4 条 PIT 通道。在本示例中，DMA ch5 用于存储新的 PIT 计数器值。

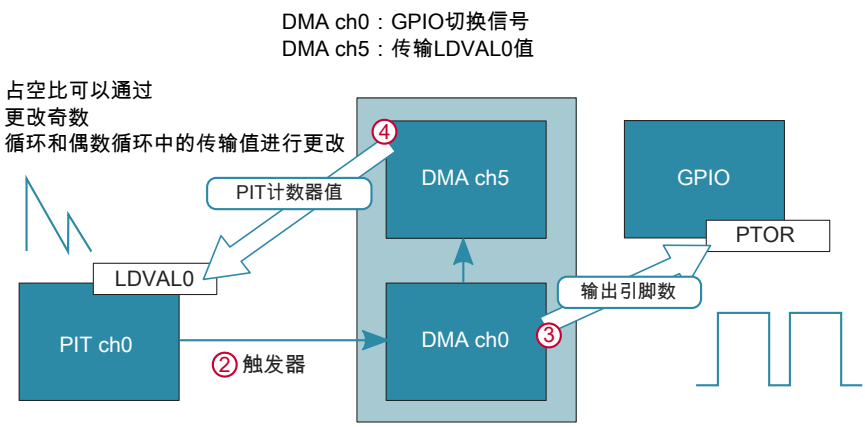
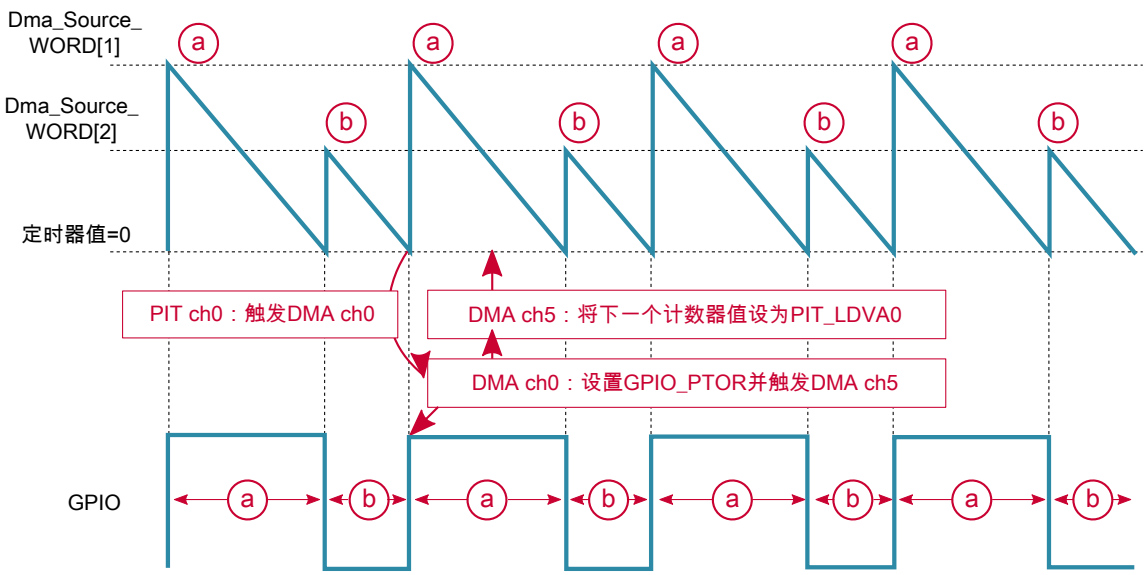


图 2. PWM 过程概述

计数器值与 GPIO 输出信号之间的关系如下图所示。上面的波形表示 PIT 计数过程，下面的波形为 GPIO 的输出波形。当 PIT 计数器值变为 0 时，翻转 GPIO 信号。另外，PIT 将加载 LDVAL 中的新计数器值并启动向下计数。DMA 触发下一个 DMA ch5，而 DMA ch5 将更新新的计数器值至 LDVAL。



另外，PIT计数器值阵列Dma_Source_WORD将被存储为LDVALn，
且GPIO输出时序随后将发生改变。

图 3. 计数器值与 GPIO 输出信号之间的关系。

3 限制

这些设置具有一些限制。

1. PIT 生成 DMA 多路复用器的周期性触发事件的通道分配，如下表所示。

表 1. 实现 DMA 周期性触发的 PIT 通道分配

DMA 通道编号	PIT 通道
DMA 通道 0	PIT 通道 0
DMA 通道 1	PIT 通道 1
DMA 通道 2	PIT 通道 2
DMA 通道 3	PIT 通道 3

2. 如果 PWM 占空比设为 0 或 100%，则存在一些尖峰信号。
3. 每个 GPIO 端口的 GPIO 输出引脚数为 1。不允许 1 个端口存在 2 个输出引脚。因为 GPIO 只接受 1 个 DMA 触发器。
4. 最大频率为 PIT 频率的一半。但是，当 CPU 频率为 100MHz、总线频率为 50MHz、PIT 频率为 50MHz 时，频率可高达 1.5MHz。
5. 抖动：如果将 LDVAL 设置为小于 30，则约为 300nsec。

使用这些功能时，请注意以下建议：

1. 为了获得最佳效果，不允许有其他的 DMA 传输，或者对 GPIO/Peripheral Bridge 1 进行写入操作。此外，如果正在传输 SRAMU 的 DMA，则任何 CPU 或其他主机 RAM 访问将限制为 SRAML，反之亦然。
2. 如果无法满足以上条件，则在执行诸如以上之类的访问之前，软件应检查 PIT 计数寄存器，以确保不会在执行所需访问过程中挂起的 DMA 传输未被处理。

4 示例

本节介绍生成 PWM 或时钟所使用的测试环境和示例代码。

4.1 环境

本应用笔记使用下列器件作为测试环境。

- TWR-K40X256
- 内核/系统频率：100MHz
- 总线频率：50MHz
- PIT 频率：50MHz
- IAR Embedded Workbench v6.10.1

在示例代码中，GPIOD 7 引脚用作输出引脚。

4.2 示例代码

4.2.1 时钟

- void main()

  设置频率配置并调用部分函数以初始化 GPIO、PIT 和 DMA。最后启动 PIT。

- init_gpio()

  初始化 GPIO 并设置部分配置。

- void Set_Pit0(void)

  初始化 PIT0。

- void dma_32bit()

  初始化 DMA 并设置部分配置以切换 GPIO 输出和更新 LDVAL0。

全局阵列:

- Dma_Source_WORD[2]

  元素 0 的值代表 GPIO 引脚引脚电平控制配置值。

  元素 1 的值代表 GPIO 引脚引脚电平控制配置值。

```
void main (void)
{
volatile uint32 *tempptr;
printf("TWR-K40X256 GPIO Example!¥n");
/* Turn on all port clocks */
SIM_SCGC5 = SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTC_MASK |
SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;
/* Initialize GPIO on TWR-K40X256 */
```

```
init_gpio();
//PTD
Dma_Source_WORD= 0x00000080; //7pin toggle
// 1kHz PIT_LDVAL0
Dma_Source_WORD[1]= 0x00000957;
GPIOD_PTOR = 0x00000080; //PTD7 toggle
dma_32bit(); // DMA setting
Set_Pit0(); // PIT module enbale
PIT_LDVAL0 = 0x00000957;
PIT_TFLG0 = PIT_TFLG_TIF_MASK;
GPIOD_PSOR = 0x00000080; // PTD7 output
PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK;
while(1){
run_cmd();
}//While(1)
}//Main
/*
* Initialize GPIO
*/
void init_gpio()
{
//DMA outputs
PORTD_PCR7|=(0|PORT_PCR_MUX(1)|PORT_PCR_IRQC(0x1));
//Change PTD7 to outputs
GPIOD_PDDR|=GPIO_PDDR_PDD(GPIO_PIN(7));
}
void Set_Pit0(void)
{
SIM_SCGC6 |= SIM_SCGC6_PIT_MASK; // turn on PIT cloccks
PIT_MCR = 1; // reset MDIS -> enable the module
}
void dma_32bit (void)
{
volatile uint32 *tempptr;
SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;
//////////////////////////////////////////
//Set up DMA from PIT0
//////////////////////////////////////////
DMAMUX_CHCFG0|=DMAMUX_CHCFG_SOURCE(52); //PORTD
DMAMUX_CHCFG0 |= DMAMUX_CHCFG_ENBL_MASK|DMAMUX_CHCFG_TRIG_MASK;
tempptr = &Dma_Source_WORD[0];
tcd.saddr = (uint32_t)tempptr;
tcd.daddr = 0x400Ff0CC; //ADDRESS of PTD_PTOR
tcd.nbytes = 4;
tcd.tcdAttr = DMA_ATTR_SSIZE(2) | DMA_ATTR_DSIZE(2); //EDMA_TCD_ATTR_SSIZE_32BIT|
EDMA_TCD_ATTR_DSIZE_32BIT ;
tcd.soff = 0x0;
tcd.doff = 0x0;
tcd.slast = 0x0;
tcd.loopcount = 0x1;
tcd.dlast_sga = 0x0;
tcd.csr = 0x0;
tcd.channelno = 0;
dma_config(CONFIG_BASIC_XFR, &tcd);
DMA_ERQ|= 1;
}
```

4.2.2 PWM

函数:

- void main()

  设置频率配置并调用部分函数以初始化 GPIO、PIT 和 DMA。最后启动 PIT。

- init_gpio()

初始化 GPIO 并设置部分配置。

- void Set_Pit0(void)

  初始化 PIT0。

- void dma_32bit()

  初始化 DMA 并设置部分配置以切换 GPIO 输出和更新 LDVAL0。

全局阵列:

- Dma_Source_WORD[3]

元素 0 的值代表 GPIO 引脚引脚电平控制配置值。元素 1 和 2 的值代表 LDVAL0 的更新值。元素 1 的值为高电平时间，元素 2 的值为低电平时间。

```
void main (void)
{
volatile uint32 *tempptr;
printf("TWR-K40X256 GPIO Example!¥n");
/* Turn on all port clocks */
SIM_SCGC5 = SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTC_MASK |
SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;
/* Initialize GPIO on TWR-K40X256 */
init_gpio();
//PTD
Dma_Source_WORD[0]= 0x00000080; //7pin toggle
// 1kHz duty 50% PIT_LDVAL0
Dma_Source_WORD[1]= 0x00000957;
Dma_Source_WORD[2]= 0x00000957;
GPIOD_PTOR = 0x00000080; //PTD7 toggle
dma_32bit(); // DMA setting
Set_Pit0(); // PIT module enbale
PIT_LDVAL0 = 0x00000957;
PIT_TFLG0 = PIT_TFLG_TIF_MASK;
GPIOD_PSOR = 0x00000080; // PTD7 output
PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK;
while(1){
run_cmd();
}//While(1)
}//Main
/*
* Initialize GPIO
*/
void init_gpio()
{
//DMA outputs
PORTD_PCR7|=(0|PORT_PCR_MUX(1)|PORT_PCR_IRQC(0x1));
//Change PTD7 to outputs
GPIOD_PDDR|=GPIO_PDDR_PDD(GPIO_PIN(7));
}
void Set_Pit0(void)
{
SIM_SCGC6 |= SIM_SCGC6_PIT_MASK; // turn on PIT cloccks
PIT_MCR = 1; // reset MDIS -> enable the module
}
void dma_32bit (void)
{
volatile uint32 *tempptr;
SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;
//////////////////////////////////////////
//Set up DMA from PIT0
//////////////////////////////////////////
DMAMUX_CHCFG0|=DMAMUX_CHCFG_SOURCE(52); //PORTD
DMAMUX_CHCFG0 |= DMAMUX_CHCFG_ENBL_MASK|DMAMUX_CHCFG_TRIG_MASK;
tempptr = &Dma_Source_WORD[0];
```

```
tcd.saddr = (uint32_t)tempptr;
tcd.daddr = 0x400Ff0CC; //ADDRESS of PTD_PTOR
tcd.nbytes = 4;
tcd.tcdAttr = DMA_ATTR_SSIZE(2) | DMA_ATTR_DSIZE(2); //EDMA_TCD_ATTR_SSIZE_32BIT|
EDMA_TCD_ATTR_DSIZE_32BIT ;
tcd.soff = 0x0;
tcd.doff = 0x0;
tcd.slast = 0x0;
tcd.loopcount = 0x1;
tcd.dlast_sga = 0x0;
tcd.csr = 0x0520;
tcd.channelno = 0;
dma_config(CONFIG_BASIC_XFR, &tcd);
DMA_ERQ|= 1;
tempptr = &Dma_Source_WORD[1];
tcd.saddr = (uint32_t)tempptr;
tcd.daddr = 0x40037100; //ADDRESS of PIT_LDVAL0
tcd.nbytes = 4;
tcd.tcdAttr = DMA_ATTR_SSIZE(2) | DMA_ATTR_DSIZE(2);
//EDMA_TCD_ATTR_SSIZE_32BIT|EDMA_TCD_ATTR_DSIZE_32BIT ;
tcd.soff = 0x04;
tcd.doff = 0x0;
tcd.slast = -8;
tcd.loopcount = 0x2;
tcd.dlast_sga = 0x0;
tcd.csr = 0x0;
tcd.channelno = 5;
dma_config(CONFIG_BASIC_XFR, &tcd);
DMA_ERQ|= 0x1 << 1;
}
```

4.3 结果

我们已观察到 PTD7 和 PTA16 的波形。

如下图所示，每个引脚输出 1kHz 时钟。

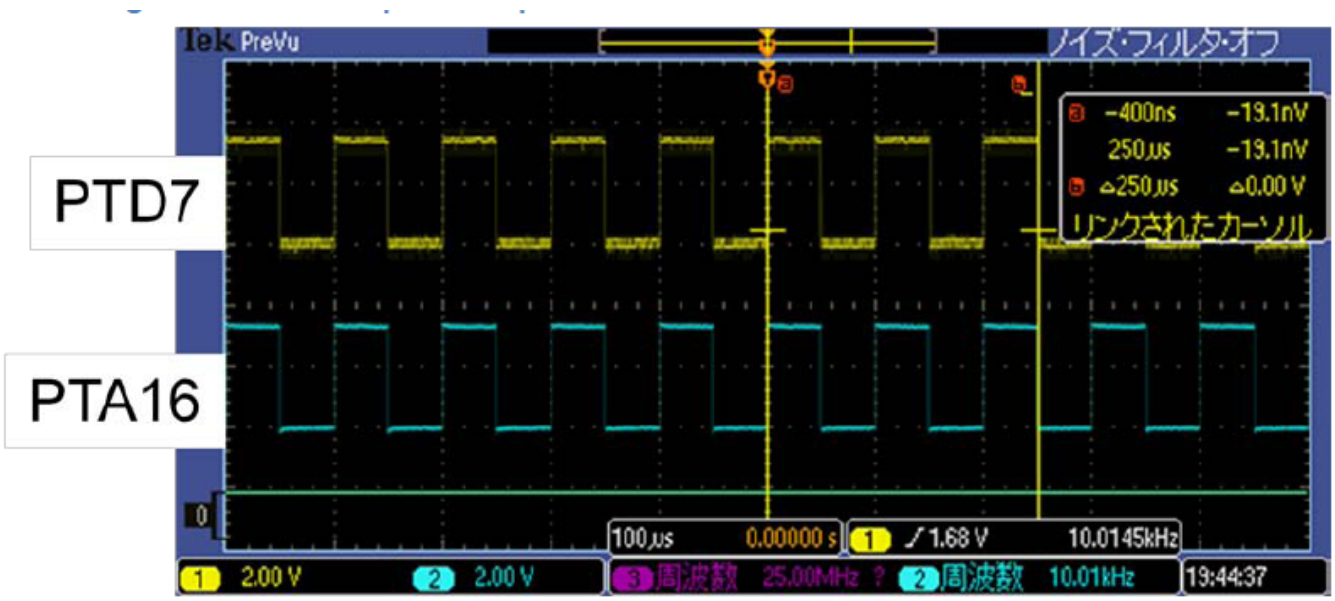


图 4. 输出 1kHz 时钟

如下图所示，PTD7 输出频率为 500Hz，占空比为 75%。PTA16 输出频率为 2kHz，占空比为 50%。

图 5. PWM 输出

5 结论

本应用笔记提供有关如何在 Kinetis 系列器件上使用 DMA 和 GPIO 模拟定时器功能的信息。所提供的源代码以及本应用笔记（AN4419SW）可用作您配置的基础。有关 Freescale Kinetis 系列的更多信息，请参阅

http://www.freescale.com/webapp/sps/site/homepage.jsp?code=KINETIS&tid=VANKINETIS

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

本文档中的信息仅供系统和软件实施方使用 Freescale 产品。 本文并未明示或者暗示授予利用本文档信息进行设计或者加工集成电路的版权许可。 Freescale 保留对此处任何产品进行更改的权利，恕不另行通知。

Freescale 对其产品在任何特定用途方面的适用性不做任何担保、表示或保证，也不承担因为应用程序或者使用产品或电路所产生的任何责任，明确拒绝承担包括但不局限于后果性的或附带性的损害在内的所有责任。 Freescale 的数据表和/或规格中所提供的“典型”参数在不同应用中可能并且确实不同，实际性能会随时间而有所变化。 所有运行参数，包括“经典值”在内，必须经由客户的技术专家对每个客户的应用程序进行验证。 Freescale 未转让与其专利权及其他权利相关的许可。 Freescale 销售产品时遵循以下网址中包含的标准销售条款和条件：freescale.com/SalesTermsandConditions.

Document Number AN4419
Revision 0, 01/2012