华中科技大学瑞萨实验室

技术报告

K60的底层驱动开发总结

Ver0.1

胡春旭

曹申

程鹏

张越

华中科技大学控制科学与工程系

2012年5月7日

目录

1 MCG 模块

1.1 MCG 模块简介

MCG 成为多用途时钟信号发生器，该模块为 MCU 和芯片内部各个模块提供时钟信号。MCG 模块通过对输入的参考时钟进行分频和倍频处理，可以输出各种频率的时钟信号，并达到超频运行的目的。输入的参考时钟可以来自内部也可以来自外部时钟或外部晶振。输入的参考时钟一般经过分频后进入 FLL（锁频环）或 PLL（锁相环）进行倍频处理，当然也可以不通过 FLL 或 PLL，而直接输出给各个模块。一般情况下，我们会选择外部晶振的参考时钟，分频后由 PLL 倍频的方法来达到超频运行的目的。

MCG 模块框图如图所示：

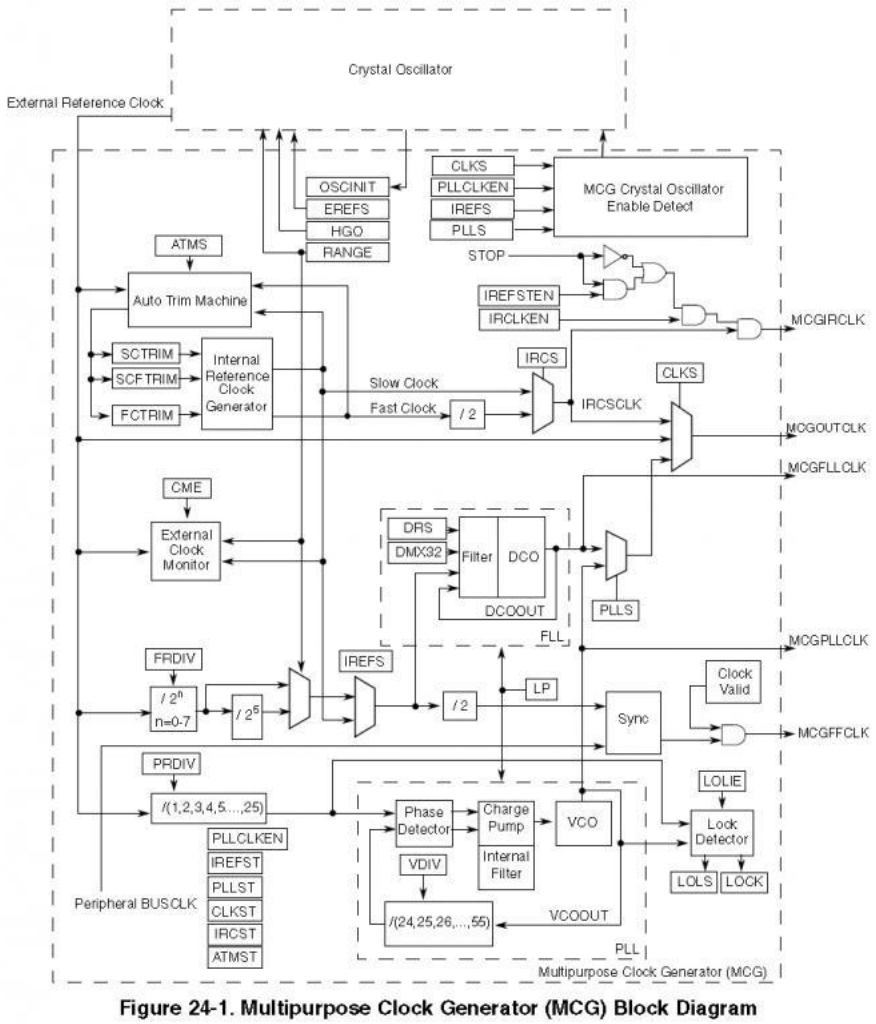
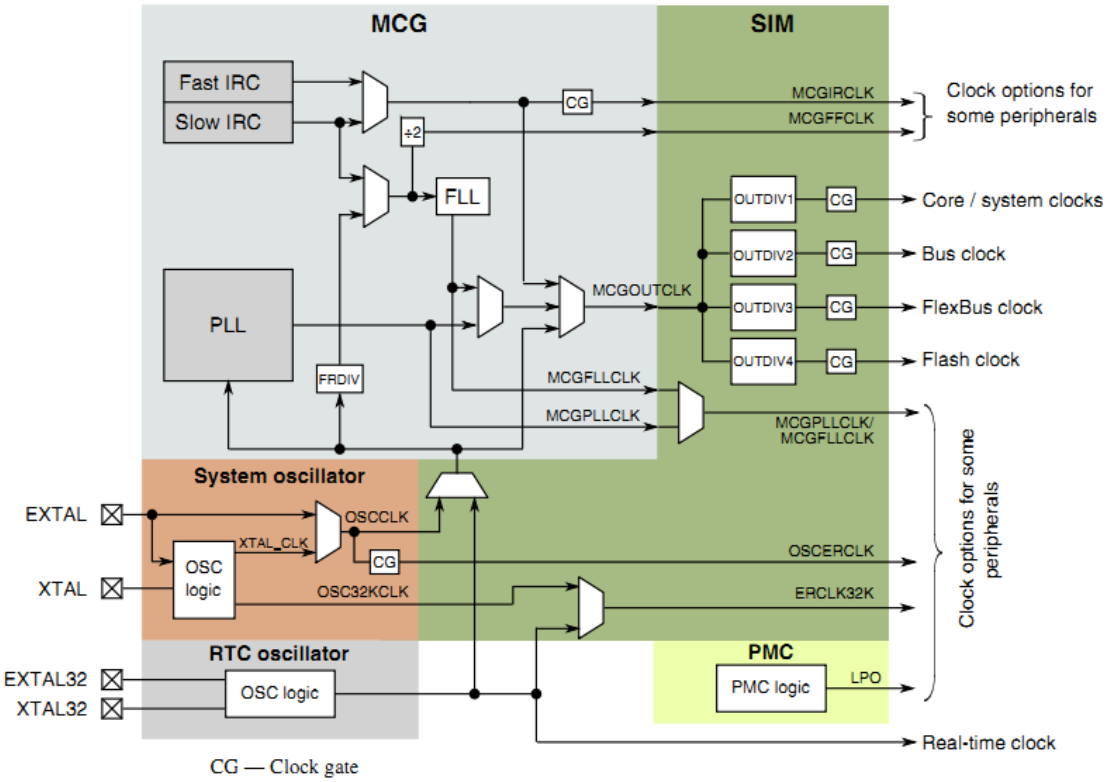


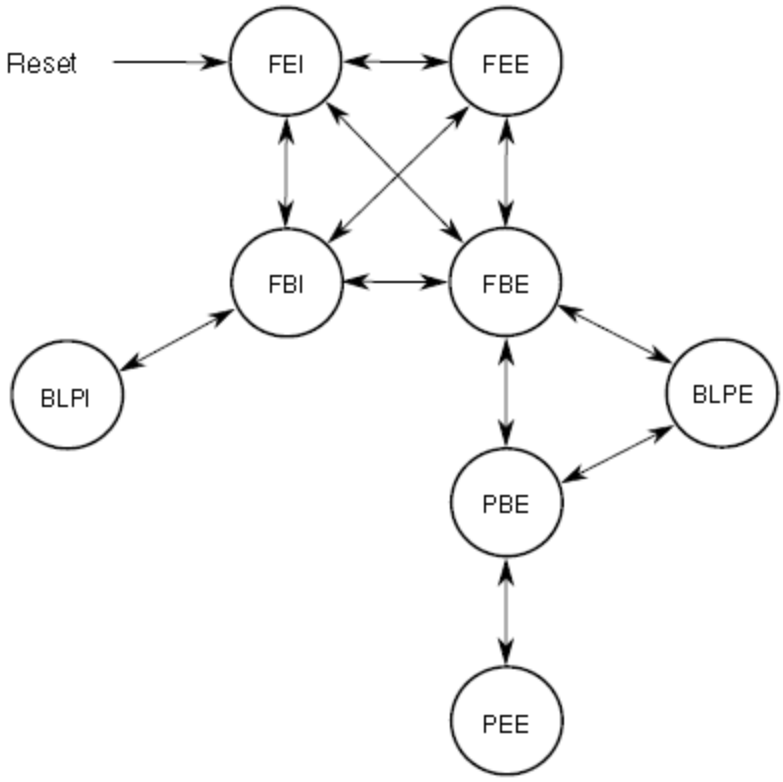
Figure 24-1. Multipurpose Clock Generator (MCG) Block Diagram

MCG 的核心是一个 FLL（锁频环）和一个 PLL（锁相环）。FLL 和 PLL 都可以对输入的参考时钟倍频并锁定后输出，输出的时钟在 SIM 模块的控制下提供给 CPU 和各个模块。锁频环 FLL 的核心是一个 DCO（数字控制振荡器），锁相环 PLL 的核心是一个 VCO（电压控制振荡器）。如图所示。

注意：MCG 模块可输出多个时钟，比如 MCGFFCLK、MCGPLLCLK、MCGOUTCLK 等，详见图所示。这些时钟可以为其它模块提供更多的时钟选择。在这些时钟中，最主要的是 MCGOUTCLK。这个时钟是核心时钟、总线时钟、FLASH 时钟的时钟来源。我们要实现 PLL 超频最终得到的也是 MCGOUTCLK。



不同的参考时钟，以及不同的对参考时钟的处理方式的组合，使得 MCG 模块有 9 种不同的工作模式。9 种模式及相互之间的转换如图所示。



这 9 种模式我们未必都用到，一般情况下，我们都会使用 PEE 模式，PEE 模式是选择外部晶振经过 PLL 倍频后输出时钟，可以达到较高的时钟频率。而上电复位后，MCG 工作在 FEI 模式下。我们必须通过设置相关的寄存器，实现 FEI 到 FBE 的转换，再由 FBE 转换到 PBE，最后再转换到 PEE。下面我们就介绍一下我们用到的几种模式及在转换过程中用到的寄存器。

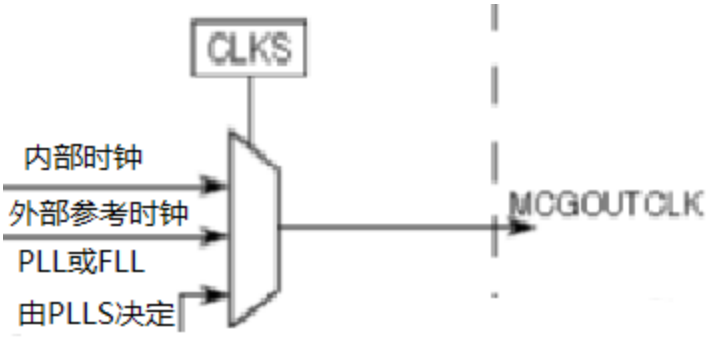
1.2 相关寄存器

(1) MCG 控制寄存器 1（MCG_C1）

Bit	7	6	5	4	3	2	1	0
Read Write	CLKS		FRDIV			IREFS	IRCLKEN	IREFSTEN
Reset	0	0	0	0	0	1	0	0

- CLKS：时钟源选择。该位用于选择 MCG 的输出时钟 MCGOUTCLK 从何而来。在 MCG 的模式转换过程中，每一步都牵涉到 CLKS 的设置。注意，每次改变 CLKS，输出的时钟并不是马上发生变化，而是有一个过程，必需查询相关的标志位以确保转换完成才能进行下一步设置，需查询的相关标

志位在 MCG_S 寄存器中，有关该寄存器的内容稍后介绍。



CLKS 选择时钟如表所示：

CLKS	描述
00	选择 PLL 或 FLL 输出
01	选择内部参考时钟
10	选择外部参考时钟
11	保留，缺省为 00

- FRDIV：锁频环 FLL 外部参考分频。当 FLL 的输入参考时钟来自外部晶振时，可通过设置 FRDIV 对这个来自晶振的参考时钟分频。分频后的时钟可以提供给 FLL 再倍频，并且这个时钟也可以提供给 MCGFFCLK 输出。对于有些模块，比如 FTM，MCGFFCLK 也是可供选择的一个时钟选项。FRDIV 的分频系数和外部晶振的频率范围有关，频率范围在 MCG_C2 寄存器中的 RANGE 设置。RANGE=0，分频比=2^FRDIV（FRDIV<=7），如果外部晶振 RANGE!=0，则分频比=2^（FRDIV+5）（FRDIV<=5）。

  注意：分频后提供给 FLL 倍频的参考时钟不得超过 32K。

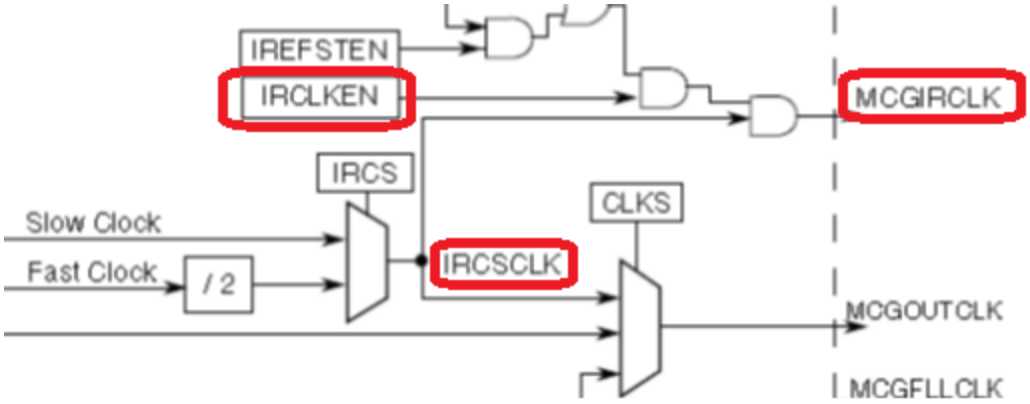
- IREFS：FLL 参考时钟源选择。FLL 对输入的参考时钟倍频，而这个参考时钟可以来自芯片自带的内部慢速参考时钟，速率为 32k，也可以来自外部晶振经由 FRDIV 设置的分频比分频后的时钟。选择哪一个输入到 FLL 由 IREFS 设置。

IREFS=0	选择外部时钟
IREFS=1	选择内部慢速参考时钟源

注意，上电复位时，IREFS 默认为 1，即 FLL 对 32K 内部慢速时钟倍频后输

出。

- IRCLKEN、IREFSTEN：前面提到，MCG 模块有多个时钟输出，其中一个就是 MCGIRCLK，这个时钟来自于芯片内部自带的参考时钟，这个时钟也可以提供给某些模块作为时钟选项。IRCLKEN 和 IREFSTEN 就是和这个时钟有关。如图 1.6 所示。如需要这个时钟，可设置 IRCLKEN=1，如希望在停止模式下，该时钟也可以使用，则可设置 IREFSTEN=1。在超频过程中，这两位可暂时忽略。



(2) MCG 控制寄存器 2（MCG_C2）

Bit	7	6	5	4	3	2	1	0
Read	0	0	RANGE		HGO	EREFS	LP	IRCS
Write								
Reset	0	0	0	0	0	0	0	0

- RANGE：晶振频率范围选择，用于为使用的外部晶振选择频率范围，具体如表所示：

RANGE	描述
00	为晶振选择低频范围 1K~32K
01	为晶振选择高频范 3M~8M
1X	为晶振选择甚高频范围 8M~32M

注意：一般情况下，如果使用 32K 晶振，则可选择 RANGE=00，如晶振频率为 3~32MHz，则可选择 RANGE=01，如果使用更高频率的晶振，则可选择 RANGE=1X。

- HGO：高增益振荡器选择

HGO=0	设置晶振为低功耗操作
HGO=1	设置晶振为高增益操作

- EREFS：外部参考选择。来自外部的时钟信号可以是外部直接输入的时钟脉冲，或来自晶体振荡器，可通过 EREFS 选择。

EREFS=0	外部参考时钟
EREFS=1	外部振荡器

注意：一般情况下，我们使用都是外部晶振，该位置 1 即可。

- LP：低功耗选择，用于选择在 bypass 等模式下，FLL 和 PLL 是否工作，主要出于省电考虑，超频过程中，该位可暂不考虑，默认即可。
- IRCS：内部参考时钟选择。如果我们选择把芯片自带的内部时钟作为 MCGOUTCLK 输出，即设置 MCG_C1 寄存器中的 CLKS=01，则有两个选择，一个是 32K 的慢速时钟，一个是 2M 的快速时钟，由 IRCS 选择。在 PLL 超频过程中，该位可忽略。

IRCS=0	选择慢速内部参考时钟；
IRCS=1	选择快速内部参考时钟。

(3) MCG 控制寄存器 5（MCG_C5）

Bit	7	6	5	4	3	2	1	0
Read	0	PLLCLKEN	PLLSTEN	PRDIV				
Write								
Reset	0	0	0	0	0	0	0	0

- PLLCLKEN：PLL 时钟使能。MCGPLLCLK 也是 MCG 模块输出的一个时钟选项。如果要使能这个时钟输出，则设置 PLLCKEN=1 即可。PLLCLKEN 和我们要超频输出的 MCGOUTCLK 无关，在 PLL 超频过程中可忽略。如果使用 MCGPLLCLK 的话，需要注意，置位 PLLCLKEN 之前，PRDIV 需要设置合适的分频系数以产生 2~4M 的 PLL 参考时钟，置位 PLLCLKEN 将使能外部振荡器。每次置位 PLLCLKEN 使能 PLL 时钟，并且外部振荡器作为参考时钟，都需要检查 OSCINIT 位。

- PLLSTEN：设置 stop 模式下，PLL 是否使能。PLL 超频过程中，该位可忽略。
- PRDIV：PLL 外部参考时钟分频。我们选择外部晶振作为参考时钟输入到 PLL 时，需要先分频到 2~4MHz 的范围内才可以由 PLL 倍频后输出。PRDIV 就是设置对外部时钟的分频系数。

  分频系数=PRDIV+1，注意 PRDIV<=24。

  注意：通过设置 PRDIV，结果频率应在 2M~4M 范围内，当 PLL 使能，即将 PLLCLKEN 置 1 后，在锁相环 PLL 未锁定频率之前，不可改变 PRDIV 的值。

(4) MCG 控制寄存器 6（MCG_C6）

Bit	7	6	5	4	3	2	1	0
Read Write	LOLIE	PLLS	CME	VDIV				
Reset	0	0	0	0	0	0	0	0

- LOLIE：当时钟失去锁定时，是否发生中断请求。

LOLIE=0	禁止中断请求
LOLIE=1	允许中断请求

- PLLS：PLL 选择，选择 PLL 还是 FLL 作为 MCG 的时钟源。当 CLKS[1:0]=00 时，我们可以选择 MCGOUTCLK 来自 PLL 或者 FLL。到底是 PLL 还是 FLL 则由 PLLS 指定。

PLLS=0	选择 FLL
PLLS=1	选择 PLL，设置 PLLS 之前，需要设置合适的 PRDIV 以产生 2~4M 的参考时钟

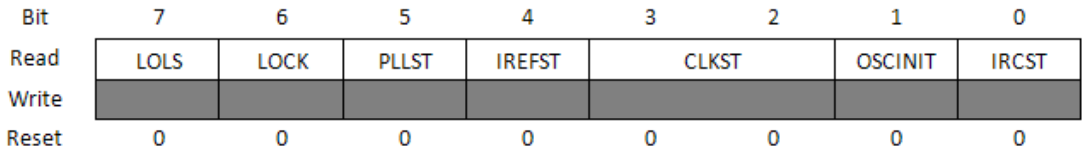
- CME：时钟监控使能，设置当失去外部时钟时，是否产生中断。
- VDIV：PLL 压控振荡器倍频系数，VDIV 决定参考时钟的倍数。当我们最终选择 PLL 倍频后输出到 MCGOUTCLK 时，需要设置 VDIV 得到我们需要的时钟频率。

倍数=VDIV+24，且 VDIV<=31。

注意：PLL 最大 55 倍频，参考时钟最大 4MHz，也就是 PLL 超频最高到 220MHz。

(5) MCG 状态寄存器（MCG_S）

这个寄存器中包含反映 MCG 状态的标志位，供程序查询。在 MCG 的模式转换过程中，我们必须查询该寄存器中的某些标志位以确保转换完成。



Bit	7	6	5	4	3	2	1	0
Read	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	IRCST
Write								
Reset	0	0	0	0	0	0	0	0

- LOLS：该位反映锁相环的锁定状态，失锁时置 1，写 1 清除，该位置 1 时，由 MCG_CR6 中的 LOLIE 决定是否产生中断。
- LOCK：该位置 1 表示锁相环已经锁定。在锁定过程中，MGC 的 PLL 时钟被禁止输出，直到 LOCK 置位。LOCK 置位后，任何改变 PRDIV 或 VDIV 的操作都会清除 LOCK 位直到再次锁定频率。
- PLLST：该位指示，PLLS 的时钟源是来自 FLL 时钟还是 PLL 时钟。当我们改变 PLLS 在 PLL 和 FLL 时钟之间切换时，时钟输出的变化需要查询该位以确保切换完成。

PLLST=0	**FLL 时钟**
PLLST=1	PLL 时钟

- IREFST：FLL 内部参考时钟状态。当我们改变 IREF，对 FLL 的参考时钟的来源切换时，需要查询该位以确保切换完成才可进行下一步。

IREFST=0	**FLL 参考时钟来自外部**
IREFST=1	FLL 参考时钟来自内部

- CLKST：指示当前的时钟模式。当我们改变 CLKS，改变 MCGOUTCLK 的时钟来源时，需要查询该位以确保切换完成才可进行下一步。

CLKST=00	**选择 FLL 输出**
CLKST=01	选择内部参考时钟

CLKST=10	选择外部参考时钟
CLKST=11	选择 PLL 输出

- OSCINT：晶振初始化状态，该位置 1 表示晶振完成初始化。
- IRCST：内部参考时钟状态。如改变 IRCS，则须查询该位，该位和 PLL 超频无关。

IRCST=0	**选择慢速内部时钟，32K**
IRCST=1	选择快速内部时钟，2M

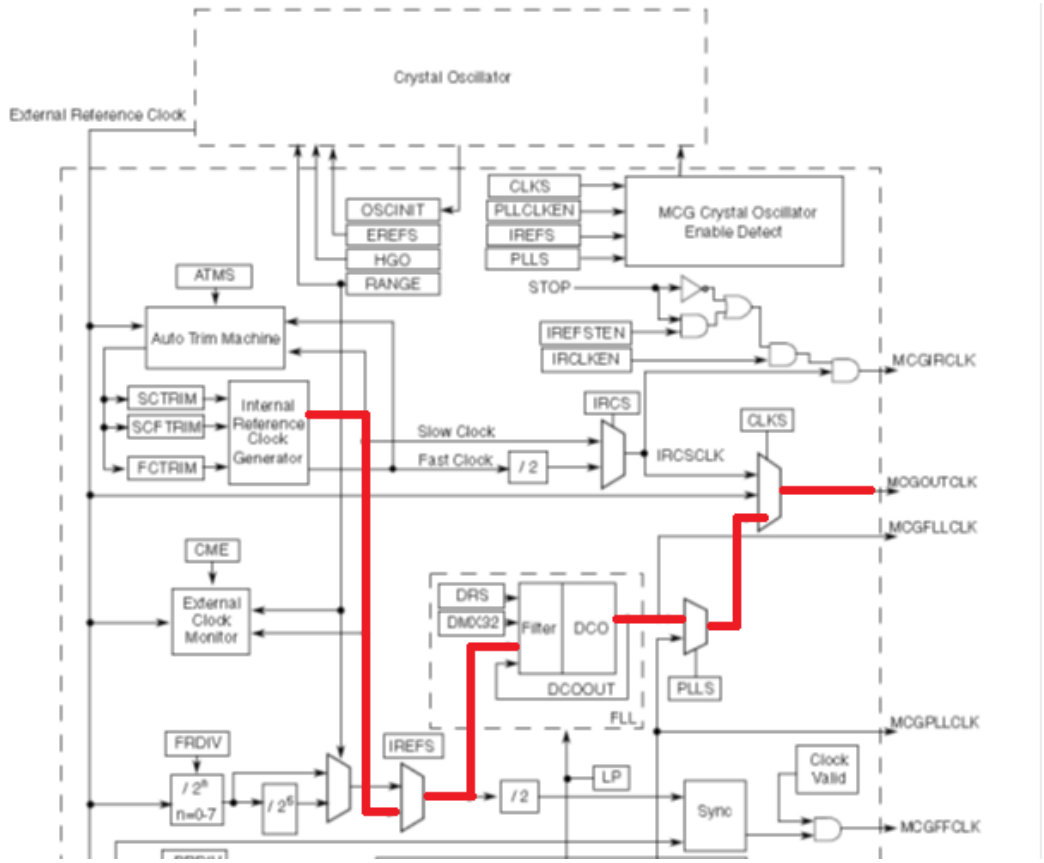
MCG 模块和 PLL 超频有关的寄存器都已经列出来，还有些寄存器和 PLL 超频无关，不做详细介绍，如想进一步了解可参考数据手册。

1.3 模式转换及代码解析

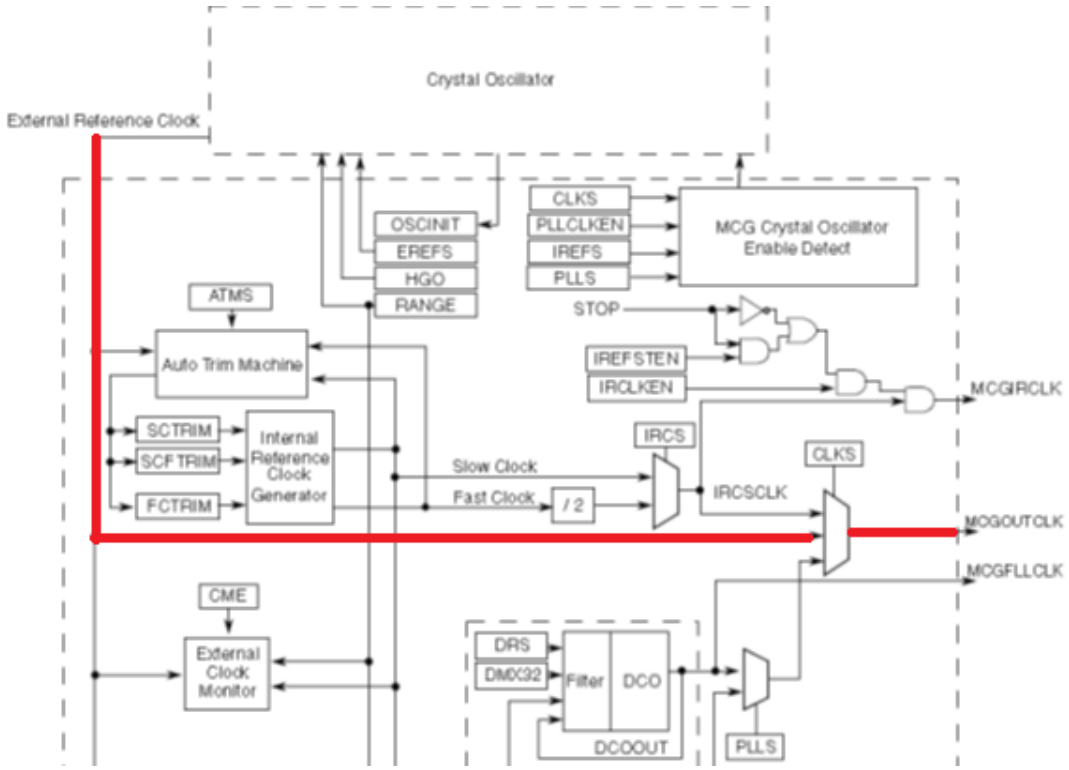
如 1.1 图所示，我们必须通过设置相关的寄存器，实现 FEI 到 FBE 的转换，再由 FBE 转换到 PBE，最后再转换到 PEE。

(1) FEI 转换到 FBE

FEI 模式下，时钟的产生如图所示。FEI 是系统复位后 MCG 默认的工作模式，输出的时钟 MCGOUTCLK 来自 FLL。FLL 的参考时钟来自内部 32K 慢速时钟，FLL 倍频后大约 20MHz。



FBE 模式下，时钟的产生如下图所示。在 FBE 模式下，MCGOUTCLK 来自外部参考时钟，此时，FLL 仍正常工作，但是 FLL 的参考时钟输入从内部慢速时钟切换到外部时钟。此时 FLL 产生的时钟并不输出。



从 FEI 到 FBE 的转换，主要改变的是 MCGOUTCLK 的来源从 FLL 切换到外部晶振，即 MCG_C1 寄存器的 CLKS 从 00 切换到 10。同时 FLL 的参考时钟也从内部 32K 慢速时钟切换到晶振分频后的时钟，即 MCG_C2 寄存器中 RANGE、HGO 和 EREFS 都需要设置，而且 MCG_C1 寄存器中的 FRDIV 也需要合理设置以保证外部晶振分频后得到不大于 32KHz 的参考时钟提供给 FLL。

以 4MHz 外部晶振为例，首先设置 MCG_C2 寄存器，代码如下：

```
MCG_C2=MCG_C2_RANGE(1) | MCG_C2_HGO_MASK | MCG_C2_EREFS_MASK;
```

使用 4MHz 晶振，RANGE 设置为 1；HGO 置 1，选择高增益；EREFS 置 1，选择外部晶振。在这段代码中，大量采用系统自带的宏定义，具体和查询头文件 MK60X256VMD100.h。使用系统自带的宏定义可防止自己计算二进制出错，当然，也可以不使用，宏定义，而直接定义：

```
MCG_C2=0X1C;
```

接下来设置，MCG_C1 寄存器，代码如下：

```
MCG_C1 = MCG_C1_CLKS(2) | MCG_C1_FRDIV(3);
```

CLKS 设为 10，选择外部参考时钟输出到 MCGOUTCLK；FRDIV 置为 3，对 4MHz 晶振 256 分频。注意分频后的时钟只需要小于 32KHz 即可，由于 FBE 只是过渡状态，所以不必详细计算。同样，上面的代码我们也使用了系统自带的宏定义。

当设置完毕后，需查询 MCG_S 中的几个状态位以确保状态切换完成，才能进行后面的操作，查询状态位的代码如下：

```
while (!(MCG_S & MCG_S_OSCINIT_MASK)){}     //等待锁相环初始化结束
while (MCG_S & MCG_S_IREFST_MASK){}     //等待时钟切换到外部参考时钟
while (((MCG_S & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT)!= 0x2){}
//等待 MCGOUTCLK 切换到外部时钟输出
```

(2) FBE 切换到 PBE

PBE 模式下，MCGOUTCLK 输出时钟不变，仍然是外部晶振直接输出，所以 MCG_C1 中的 CLKS 不必改变。但是，在这一步，我们开始启用 PLL 工作，并计算好我们需要超频的倍率。在 PBE 模式下，PLL 虽然已正常工作，PLL 的时钟并不输出。在这一步，我们只要设置 MCG_C5 和 MCG_C6 两个寄存器。

```
MCG_C5 = MCG_C5_PRDIV(1);//分频在 2~4MHz 之间，分频后频率 2MHz
```

```
MCG_C6 = MCG_C6_PLLS_MASK |MCG_C6_VDIV(26);  //选择 PLL，倍频 50 倍
```

这里我们需要设置好 MCGOUTCLK 提供给系统核心、总线和 FLASH 的时钟分频，通过 SIM 模块的 SIM_CLKDIV1 来设置，由 MCGOUTCLK 提供给几个主要模块的时钟分频比，分别是 CORE、BUS、FLEXBUS、FLASH。注意 FLASH 模块不可超过 25MHz，否则出错。

```
SIM_CLKDIV1 = SIM_CLKDIV1_OUTDIV1(0)| SIM_CLKDIV1_OUTDIV2(1)
            | SIM_CLKDIV1_OUTDIV3(1) |SIM_CLKDIV1_OUTDIV4(3);
//MCG=PLL, core = MCG, bus = MCG/3,FlexBus = MCG/3, Flash clock= MCG/8
```

如按以上分频设置，当最终转换到 FEE 模式后，则系统核心频率为 100MHz，BUS 和 FLEXBUS 都为 50MHz，FLASH 为 25MHz。

设置完毕后，需查询标志位以确保转换完成。代码如下：

```
while (!(MCG_S &MCG_S_PLLST_MASK)){}; // 等待切换到 PLL
while (!(MCG_S &MCG_S_LOCK_MASK)){};  //等待 PLL 锁定频率
```

(3) PBE 切换到 PEE

在上面完成后，PLL 输出的时钟已经准备完毕，只需要最后通过设置 MCG_C1 中的 CLKS，把 MCGOUTCLK 的来源从外部晶振切换到 PLL 时钟即可。代码如下：

```
MCG_C1 &= ~MCG_C1_CLKS_MASK;      //CLKS=00，PLL 输出
while (((MCG_S & MCG_S_CLKST_MASK)>> MCG_S_CLKST_SHIFT) != 0x3){};//
等待切换完毕。
```

2 FTM 模块

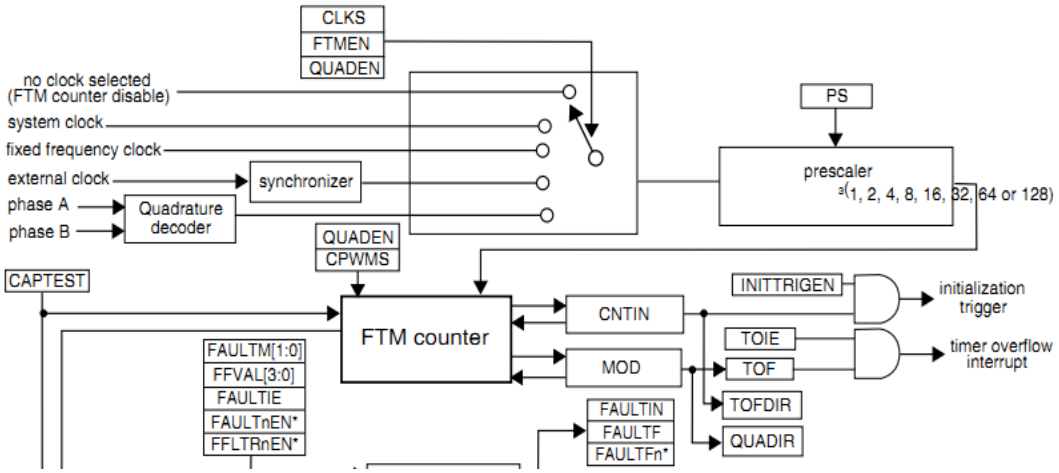
2.1 定时器功能（FTM_TIMER）

2.1.1 简介

FTM_TIMER 模块共有三个模块，分别为 FTM0、FTM1、FTM2。每个模块可以通过 CLKS[4:3]选择时钟，我们选择的为 system clock；选择时钟后经过预分频 prescaler 分频，预分频实际上为一个 7 位计数器，通过 PS[2:0]来选择分频系数；

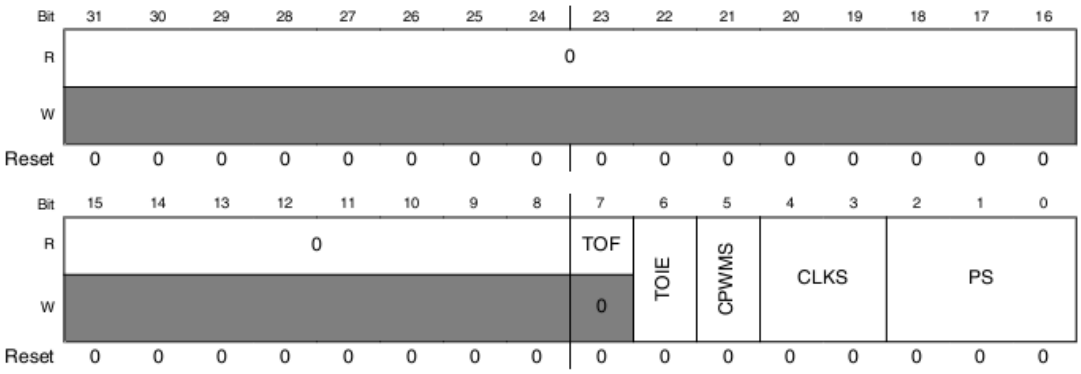
每个模块都有一个 16 位计数器，这个计数器可以向上累加，也可以先向上再向下累加，我们选择的为 Up Counting；计数器初值在 Counter Initial Value 寄存器中设置，默认为 0；终值在 Modulo 寄存器中设置。最终通过 FTM_TIMER 可以产生 3 个基准定时。

配置流程可以参考下图：



2.1.2 相关寄存器介绍

(1) 状态和控制寄存器（FTMx_SC）



- TOF：定时器溢出标志，当 FTM 模块计数器到达 MOD 寄存器中设置的结束值时，无论是递增计数还是先加后减计数，在计数值从结束值变化到下一个值时，该位置 1。当读取该寄存器，且该位置 1 时，写 0 可清除该标志，写 1 则没有效果。

- TOIE：定时器溢出中断使能。当 TOF 置 1 时是否触发中断。对外界固定时钟计数配合 TOF 和 TOIE 则可以实现定时中断的功能。

TOIE=0	定时器溢出中断禁止
TOIE=1	定时器溢出中断使能

- CPWMS：中心对齐 PWM 选择。这一位实际是设置计数器加减计数，所谓 PWM 中心对齐模式就是指的计数器先加后减。该位平时出于写保护状态，只有在 MODE[WPDIS] = 1 时才可被写入。

CPWMS=0	计数器加法计数
CPWMS=1	计数器先加后减计数

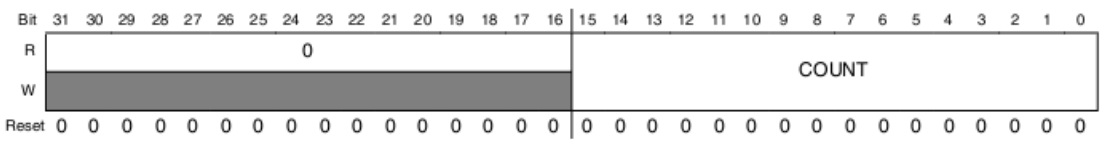
- CLKS：时钟源选择。选择 FTM 计数器的时钟来源。该位平时写保护，只有在 MODE[WPDIS] = 1 时才可写入。

CLKS=00	未选择时钟
CLKS=01	系统时钟（推荐，即 Bus Clock）
CLKS=10	定频时钟
CLKS=11	外部时钟

- PS：预分频设置。设置对 CLK 选中的时钟预分频。该位一般写保护，只有在 MODE[WPDIS] = 1 时才可写入。

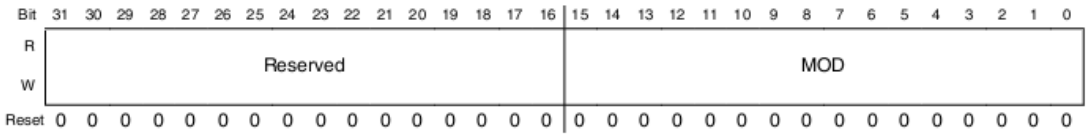
  预分频比=2^PS，最大 128 分频。

(2) 计数器 FTMx_CNT



该寄存器包含 FTM 计数器的值。复位时该寄存器清 0，向该寄存器写入任何值将会使该寄存器回到初始设定值。CNTIN 中保存的是初始设定值。

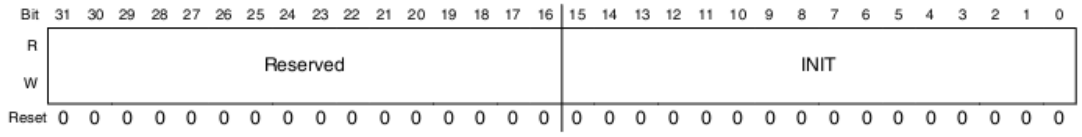
(3) 模数寄存器 FTMx_MOD



该寄存器保存 FTM 计数器的模数，即计数器计数终止值，当计数器到达对应的模数值时，TOF 将在下一个时钟到来时置 1。此时计数器的值取决于选择的计数器计数方案，默认回到初始值。

写该寄存器的值会先锁存到一个缓冲器里，不会立刻更新，而是和寄存器更新设置有关。

(4) 计数器初始值寄存器（FTMx_CNTIN）



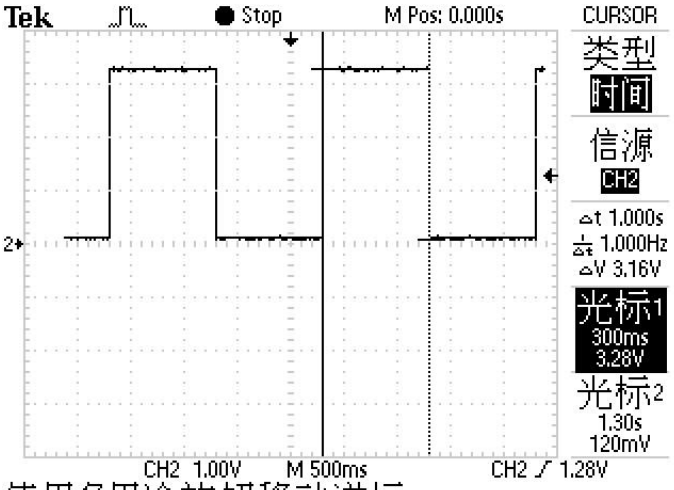
该寄存器保存 FTM 计数器的初始值。写入该寄存器的值会预先锁存在缓冲器内。在选择时钟前，先设置该寄存器以初始化 FTM 计数器，否则，计数器会默认从 0 开始计数。

2.1.3 部分代码解析

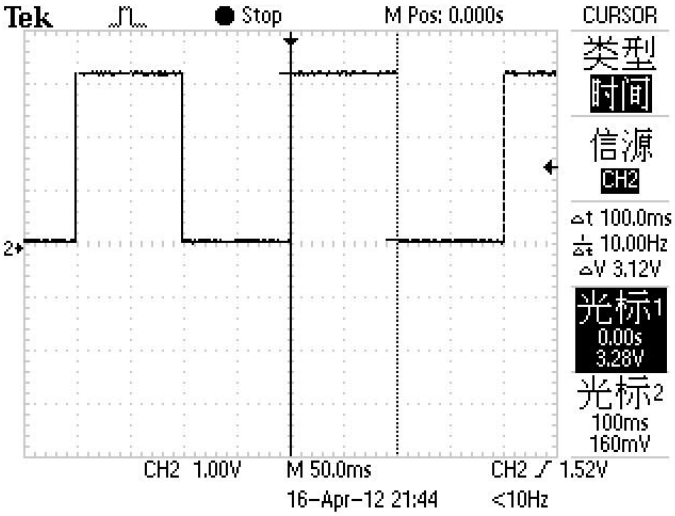
```
    /*FTM 计数器的终值寄存器，根据加载时钟调节，如现在分频后的时钟为
3MHZ，那么 30000 次就是 1 毫秒。
    如果系统时钟变为 50MHZ，那么 8 分频后是 6.25MHZ，62500 次就是 1 毫秒
*/
    FTM_MOD_REG(FTM0_BASE_PTR) = FTM_TIMER0_XMS;
    /*FTM 计数器初始值，一般设为 0*/
    FTM_CNTIN_REG(FTM0_BASE_PTR)=FTM0_CNTIN_REG;
    /*写入该寄存器时会立刻装入 CNTIN 寄存器中的初试值*/
    FTM_CNT_REG(FTM0_BASE_PTR)=FTM_CNT;
    /*使能 TOF 溢出中断,选择时钟,选择预分频系数*/
    FTM_SC_REG(FTM0_BASE_PTR)
|=((FTM_SC_TOIE_MASK)|(FTM_SC_CLKS(FTM_SC_CLKS_SYS)|(FTM_SC_PS(FTM_SC
_PS_16))));
```

以上是 FTM_TIMER0 部分的配置程序，FTM_TIMER1 和 FTM_TIMER2 的配置与其一样。

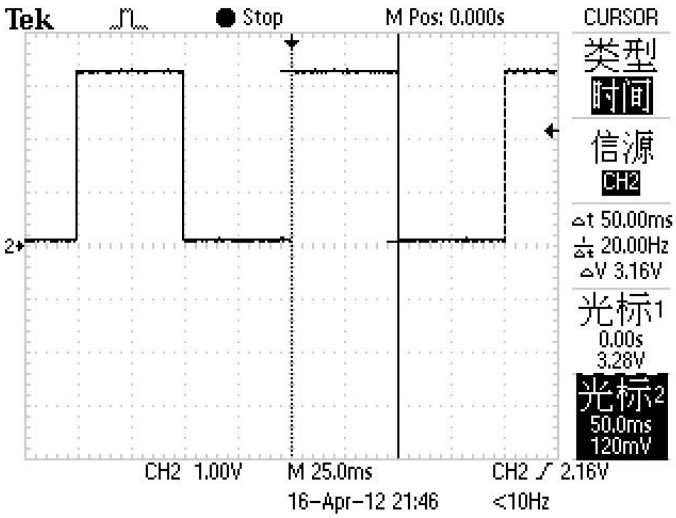
2.1.4 测试结果记录



上图为 FTM 定时器 0，定时 1s，与符合要求。



上图为 FTM 定时器 1 定时 100ms，符合要求。



上图为 FTM 定时器 2 定时 50ms，符合要求。

2.1.5 特别说明

FTM 模块在时钟确定后，计数器计数，默认情况下，对分频时钟进行无符号计数。计数器初值由 FTMx_CNTIN 指定，然后一直计数到达 FTMx_MOD 寄存器的设定值，TOF 置位，计数器回到初试值，循环往复。

计数器初始值也可是负数，例如 CNTIN=0xFFFC（补码-4），MOD=4，则计数器从-4 计数到 4，循环计数。当 CNTIN 的最高位，即 CNTIN[15]=1 时，初始值作为负数对待，否则视为正数。

FTM 计数时，建议 MOD 值必须大于 CNTIN 值，编程时需注意。如果 MOD==CNTIN，则计数值一直保持 MOD 值，且 TOF 位一直置 1。

无论何时对 FTM 计数器的写入操作，都会复位计数器，计数器的值重新回到 CNTIN 指定值。也可以使用 FTM 的同步功能，让计数器重新回到初值，各通道输出也回到初始值。

FTMx_CONF 寄存器中的 NUMOF[4:0]可设置 TOF 置位的频率。默认情况下，即 NUMOF[4:0]=0，每个计数周期，TOF 置位一次，如果 NUMOF=n，（n<32），则每 n+1 个周期，TOF 置位一次。

2.2 正交解码功能（FTM_DRCODER）

2.2.1 引脚说明

芯片资料中说明可以用来配置成正交编码功能的引脚有：

Pin（引脚号）	Function（引脚功能）	Alt（复用功能）
PTA8	FTM1_QD_PHA	Alt6
PTA9	FTM1_QD_PHB	Alt6
PTA10	FTM2_QD_PHA	Alt6
PTA11	FTM2_QD_PHB	Alt6
PTA12	FTM1_QD_PHA	Alt7
PTA13	FTM1_QD_PHB	Alt7
PTB0	FTM1_QD_PHA	Alt6
PTB1	FTM1_QD_PHB	Alt6
PTB18	FTM2_QD_PHA	Alt6
PTB19	FTM2_QD_PHB	Alt6

*开发板上用到的是 PORTB18 和 PORTB19 两个引脚。

2.2.2 功能简介

正交解码功能（Quadrature Decoder Mode）只是 FTM（FlexTimer）模块中的一个小功能，当寄存器位 FTMEN=1，QUADEN=1 时，选择为正交解码模式。

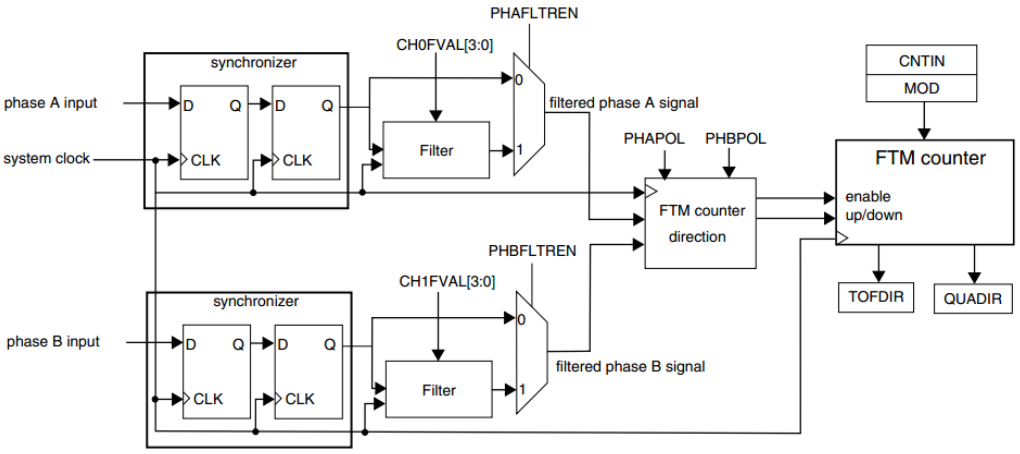


图 1. 功能框图

寄存器位 QUADMODE=1，解码模式选为 count and direction encoding mode。则 A 相输入值（phase A input）决定了计数频率，B 相输入值（phase B input）决定了计数的方向。通过图 2 可以更好的理解其工作方式。

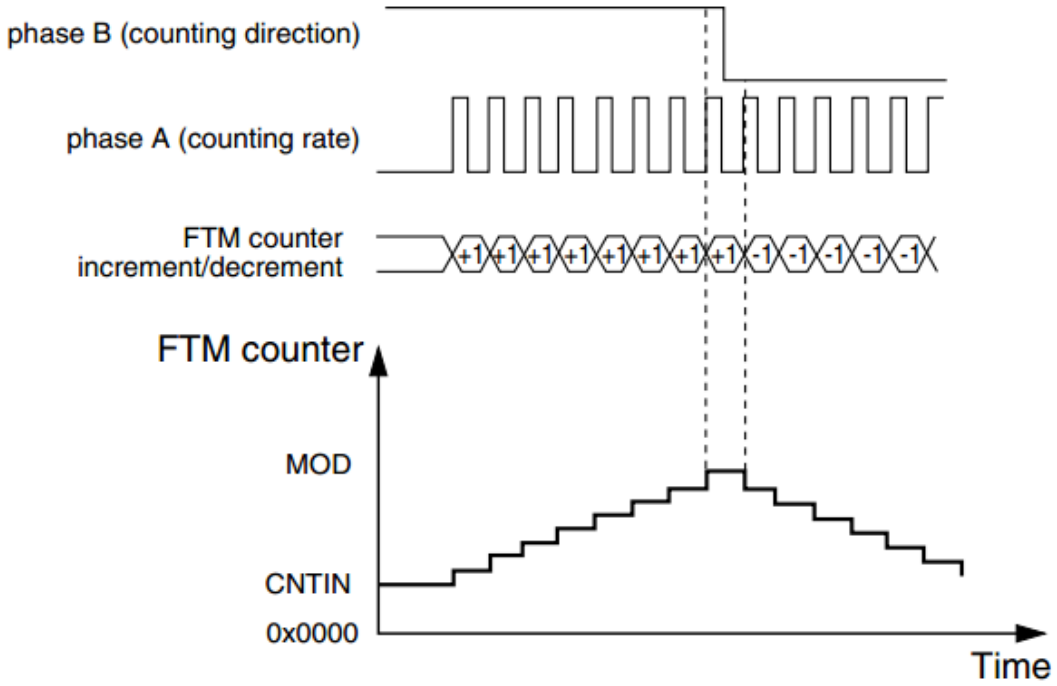
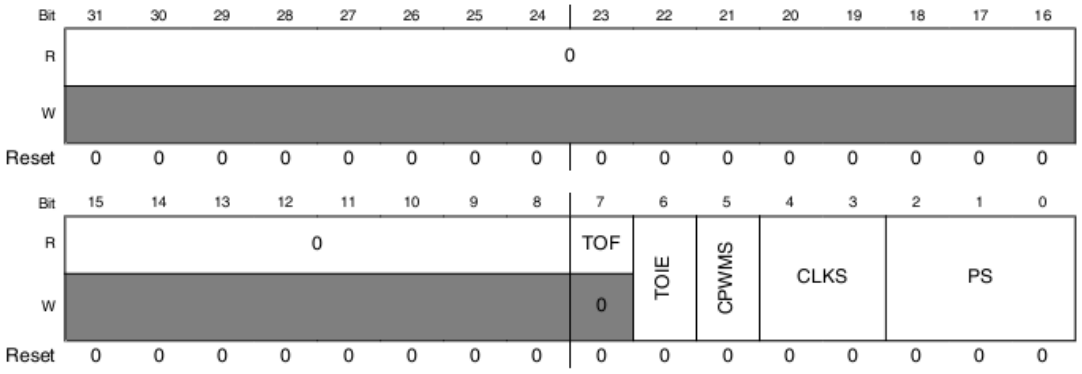


图 2. Quadrature Decoder – Count and Direction Encoding Mode

2.2.3 相关寄存器介绍

(1) 状态和控制寄存器（FTMx_SC）



- TOF：定时器溢出标志，当 FTM 模块计数器到达 MOD 寄存器中设置的结束值时，无论是递增计数还是先加后减计数，在计数值从结束值变化到下一个值时，该位置 1。当读取该寄存器，且该位置 1 时，写 0 可清除该标志，写 1 则没有效果。

- TOIE：定时器溢出中断使能。当 TOF 置 1 时是否触发中断。对外界固定时钟计数配合 TOF 和 TOIE 则可以实现定时中断的功能。

TOIE=0	**定时器溢出中断禁止**
TOIE=1	定时器溢出中断使能

- CPWMS：中心对齐 PWM 选择。这一位实际是设置计数器加减计数，所谓 PWM 中心对齐模式就是指的计数器先加后减。该位平时出于写保护状态，只有在 MODE[WPDIS] = 1 时才可被写入。

CPWMS=0	**计数器加法计数**
CPWMS=1	计数器先加后减计数

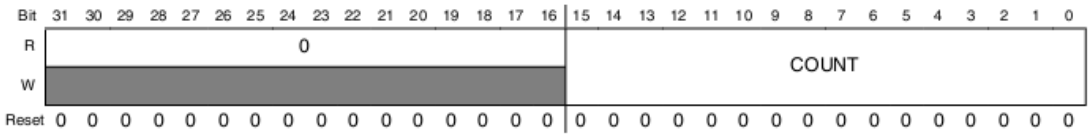
- CLKS：时钟源选择。选择 FTM 计数器的时钟来源。该位平时写保护，只有在 MODE[WPDIS] = 1 时才可写入。

CLKS=00	**未选择时钟**
CLKS=01	系统时钟（推荐，即 Bus Clock）
CLKS=10	定频时钟
CLKS=11	外部时钟

*我们在此功能中配成 CLKS=11，为外部时钟。

- PS：预分频设置。设置对 CLK 选中的时钟预分频。该位平时写保护，只有在 MODE[WPDIS] = 1 时才可写入。
  预分频比=2^PS，最大 128 分频。

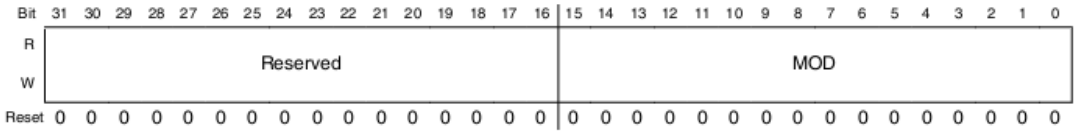
(2) 计数器 FTMx_CNT



该寄存器包含 FTM 计数器的值。复位时该寄存器清 0，向该寄存器写入任何值将会使该寄存器回到初始设定值。CNTIN 中保存的是初始设定值。

*在此功能中，它对脉冲进行计数，32 位的寄存器最多可计 2^32-1 个脉冲，对于我们来说是足够的。

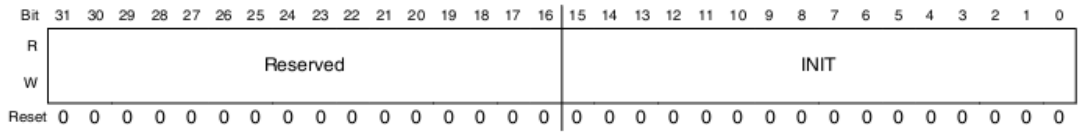
(3) 模数寄存器 FTMx_MOD



该寄存器保存 FTM 计数器的模数，即计数器计数终止值，当计数器到达对应的模数值时，TOF 将在下一个时钟到来时置 1。此时计数器的值取决于选择的计数器计数方案，默认回到初始值。

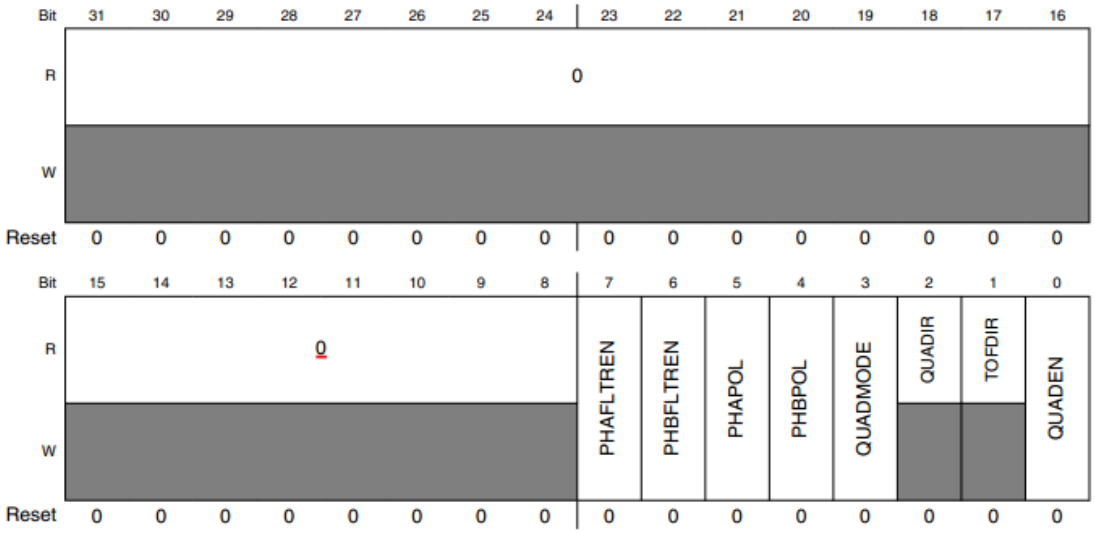
写该寄存器的值会先锁存到一个缓冲器里，不会立刻更新，而是和寄存器更新设置有关。

(4) 计数器初始值寄存器（FTMx_CNTIN）



该寄存器保存 FTM 计数器的初始值。写入该寄存器的值会预先锁存在缓冲器内。在选择时钟前，先设置该寄存器以初始化 FTM 计数器，否则，计数器会默认从 0 开始计数。

(5) 正交解码控制和状态寄存器 (FTMx_QDCTRL)



- PHAFLTREN：A 相输入过滤使能，0 不使能，1 使能。
- PHBFLTREN：B 相输入过滤使能，0 不使能，1 使能。
- PHAPOL：A 相输入极性，0 正常极性，1 翻转极性。
- PHAPOL：B 相输入极性，0 正常极性，1 翻转极性。
- QUADEMODE：正交解码模式选择，0 是 Phase A and phase B encoding mode，1 是 Count and direction encoding mode。
- QUADEN：正交解码模式使能，0 不使能，1 使能。

*寄存器中的第 3 位配成 QUADMODE=1：Count and direction encoding mode.

第 0 位配成 QUADEN= 1： Quadrature decoder mode is enabled

2.2.4 部分代码解析

(1) 程序 ftm_decoder.h 中定义的引脚

从 2.1 的引脚说明可以看出同一 FTM 模块 A、B 引脚的组合有较多的情况，我在程序中只定义了以下几种;

```
#define PTA8_9      0           //PTA8   FTM1_QD_PHA    ALT6
                                //PTA9   FTM1_QD_PHB    ALT6
#define PTA10_11    1           //PTA10  FTM2_QD_PHA    ALT6
                                //PTA11  FTM2_QD_PHB    ALT6
#define PTA12_13    2           //PTA12  FTM1_QD_PHA    ALT7
                                //PTA13  FTM1_QD_PHA    ALT7
#define PTB0_1      3           //PTB0   FTM1_QD_PHA    ALT6
                                //PTB1   FTM1_QD_PHA    ALT6
#define PTB18_19    4           //PTB18  FTM2_QD_PHA    ALT6
                                //PTB19  FTM2_QD_PHB    ALT6
```

(2) 程序 ftm_decoder.c 中的主要代码

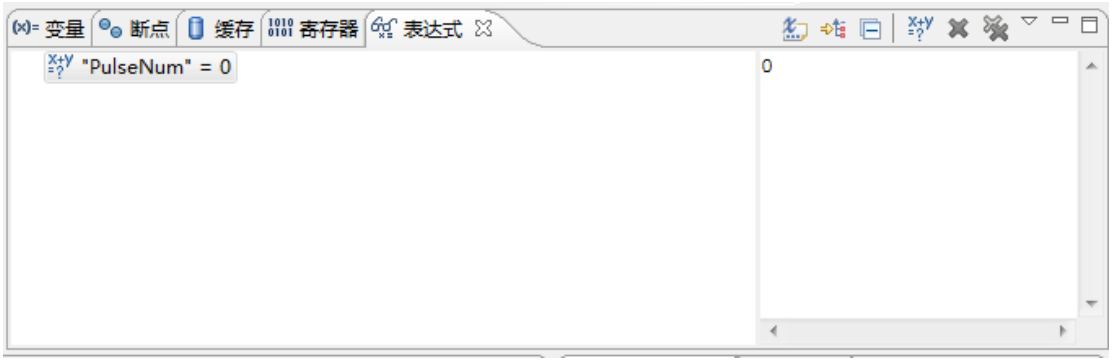
```
    case 4: //PORTB18_19
    //使能 FTM 时钟
        IM_SCGC3 |= SIM_SCGC3_FTM2_MASK;
        //选择引脚复用功能，选择引脚第 6 个功能
        PORTB_PCR18 = PORT_PCR_MUX(6);
        PORTB_PCR19 = PORT_PCR_MUX(6);
        //配置成正交编码功能
        FTM_CNTIN_REG(FTM2_BASE_PTR) = 0x0000; //初始值
        FTM_MOD_REG(FTM2_BASE_PTR) = 0xffff; //累加结束值，可根据需要设定
        //以下两句可不写，都为缺省值。因为要对寄存器写操作，所以不使能写保护。
        FTM_FMS_REG(FTM2_BASE_PTR) = 0x00; //write protect disable
        FTM_MODE_REG(FTM2_BASE_PTR) = 0x05; //不使能 write protect 功能，缺省值;FTMEN=1;
    //对时钟分频
```

```
        FTM_SC_REG(FTM2_BASE_PTR) = 0x18 | DIVIDE; //0001,1000
CLKS=11: extern Clock;PS=000|DIVIDE: system clock divided by
1or2or4or8or16or32or64or128;

        FTM_QDCTRL_REG(FTM1_BASE_PTR) = 0x0d; //0000,1101 不使能过滤,
QUADMODE=1: count and direction mode; QUADIR=1:increasing;QUADEN=1
```

2.2.5 程序的调试与测试

通过信号发生器产生脉冲信号，脉冲频率为 F=1Khz（即 1ms 产生一个脉冲），幅度 V=4v。程序中定义了一个变量 PluseNum 去读取寄存器 FTMx_CNT 的值，方便在调试时显示脉冲个数,测试引脚为 PTA8,PTA9。在 CodeWarriorIDE 10.1 下进行调试。

脉冲个数初值为 PluseNum=0,引脚 PTA8 接信号发生器的脉冲输入，PTA9 未接高电平，如前所述，说明是递减计数。调试界面如下图：



在调试过程中的某个时刻测得 PulseNum=27716，如下图：

用手机定时 10s，理论上精确的值为 PulseNum=17716，由于操作反应时间等误差，结果为 Pulsenum=17610，还算比较准确，如下图：

2.3 PWM 功能（FTM_PWM）

2.3.1 简介

FTM_PWM 模块共有三个模块，分别为 FTM0、FTM1、FTM2。FTM0 有 8 个通道，FTM1 和 FTM2 均有两个通道。每个通道可能有不只一个引脚可以选择作为输出。每个模块可以独立选择时钟源，配置预分频因子，选择 PWM 的工作模式。

2.3.2 相关寄存器介绍

(1) 状态和控制寄存器（FTMx_SC）

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TOF	TOIE	CPWMS	CLKS		PS		
W									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- TOF：计数器溢出标志，当 FTM 模块计数器到达 MOD 寄存器中设置的结束值时，无论是递增计数还是先加后减计数，在计数值从结束值变化到下一个值时，该位置 1。当读取该寄存器，且该位置 1 时，写 0 可清除该标志，写 1 则没有效果。

- TOIE：计数器溢出中断使能。当 TOF 置 1 时是否触发中断。对外界固定时钟计数配合 TOF 和 TOIE 则可以实现定时中断的功能。

TOIE=0	计数器溢出中断禁止
TOIE=1	计数器溢出中断使能

- CPWMS：中心对齐 PWM 选择。这一位实际是设置计数器加减计数，所谓 PWM 中心对齐模式就是指的计数器先加后减。该位平时出于写保护状态，只有在 MODE[WPDIS] = 1 时才可被写入。

CPWMS=0	计数器加法计数
CPWMS=1	计数器先加后减计数

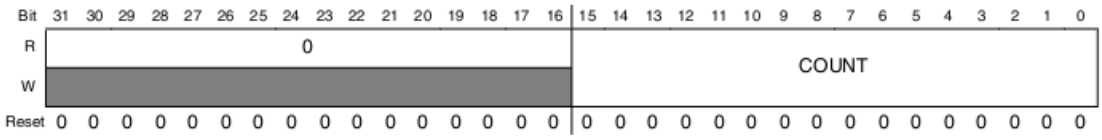
- CLKS：时钟源选择。选择 FTM 计数器的时钟来源。该位平时写保护，只有在 MODE[WPDIS] = 1 时才可写入。

CLKS=00	未选择时钟
CLKS=01	系统时钟（推荐，即 Bus Clock）
CLKS=10	定频时钟
CLKS=11	外部时钟

- PS：预分频设置。设置对 CLK 选中的时钟预分频。该位平时写保护，只有在 MODE[WPDIS] = 1 时才可写入。

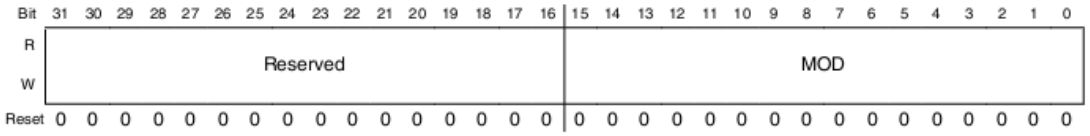
  预分频比=2^PS，最大 128 分频。

(2) 计数器 FTMx_CNT



该寄存器包含 FTM 计数器的值。复位时该寄存器清 0，向该寄存器写入任何值将会使该寄存器回到初始设定值。CNTIN 中保存的是初始设定值。

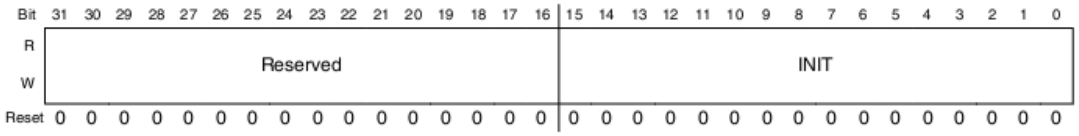
(3) 模数寄存器 FTMx_MOD



该寄存器保存 FTM 计数器的模数，即计数器计数终止值，当计数器到达对应的模数值时，TOF 将在下一个时钟到来时置 1。此时计数器的值取决于选择的计数器计数方案，默认回到初始值。

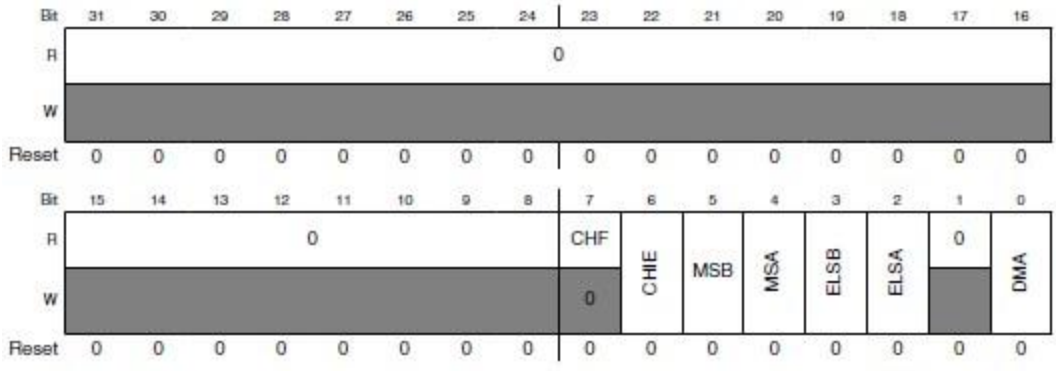
写该寄存器的值会先锁存到一个缓冲器里，不会立刻更新，而是和寄存器更新设置有关。

(4) 计数器初始值寄存器（FTMx_CNTIN）



该寄存器保存 FTM 计数器的初始值。写入该寄存器的值会预先锁存在缓冲器内。在选择时钟前，先设置该寄存器以初始化 FTM 计数器，否则，计数器会默认从 0 开始计数。

(5) 通道状态和控制寄存器（FTMx_CNTIN）



- CHF 通道标志位

  当一个事件在通道上发生时，硬件将其置位。

CHF=0	通道上没有事件发生
CHF=1	通道上有事件发生

- CHIE 通道中断使能

使能通道中断

CPWMS=0	计数器加法计数
CPWMS=1	计数器先加后减计数

- MSB 通道模式选择
- MSA 通道模式选择
- ELSB 边缘或水平选择
- ELSA 边缘或水平选择

以上四位合作选择 FTM 模块的工作模式,选择方法参考下表：

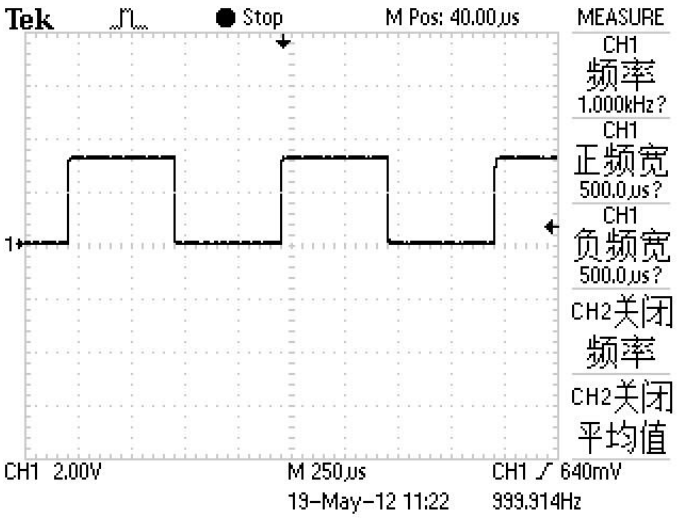
CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	configuration
0	1X	10	Edge-aligned PWM	High-true pulses (clear Output on match)
		X1		Low-true pulses (set Output on match)
1	XX	10	Center-aligned PWM	High-true pulses (clear Output on match-up)
		X1		Low-true pulses (set Output on match-up)

2.3.3 部分代码解析

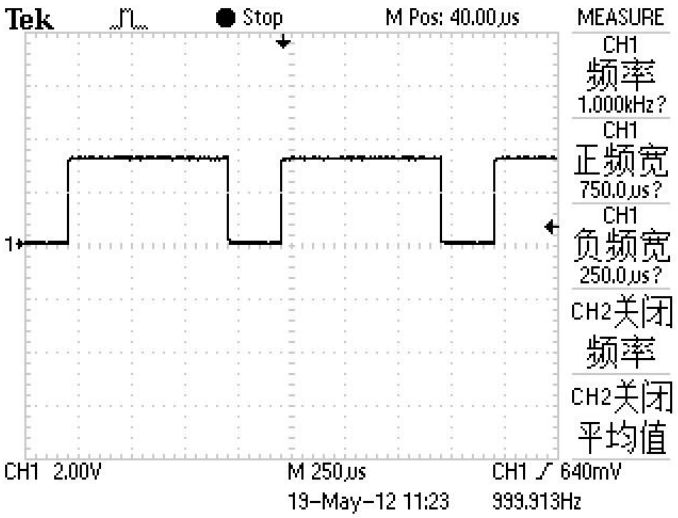
```
/*使能ftm0模块时钟*/
SIM_SCGC6 |= SIM_SCGC6_FTM0_MASK;

/*选择时钟源，分频系数，以及计数器计数方式*/
FTM_SC_REG(FTM0_BASE_PTR) = FTM_SC_CLKS(PWM0_CLOCK_SOURCE)
| FTM_SC_PS(PWM0_PRESCALE)
|((PWM0_WORK_MODE&0x10)<<1);

/*设置计数器计数初值*/
FTM_CNTIN_REG(FTM0_BASE_PTR) = CNTIN_VALUE;

/*初始化计数器*/
FTM_CNT_REG(FTM0_BASE_PTR) = ZERO;

/*设置计数器最大模值*/
FTM_MOD_REG(FTM0_BASE_PTR) = PWM0_CYCLE;

/*设置FTM模块的工作方式*/
FTM_CnSC_REG(FTM0_BASE_PTR,channel) =((PWM0_WORK_MODE&0x0f)<<2);

/*设置初始化占空比*/
FTM CnV REG(FTM0 BASE PTR,channel) = PWM INITDUTY VALUE;
```

以上是将 FTM0 模块的 PWM 功能，FTM1 与 FTM2 的配置与其类似。

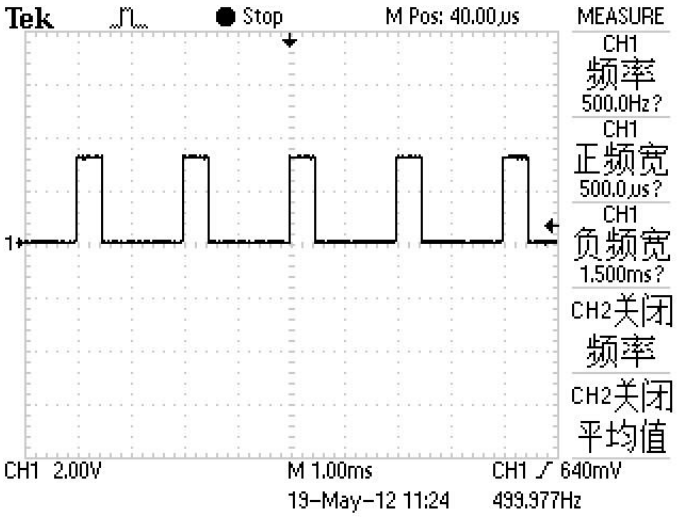
2.3.4 测试结果记录



FTM0CH1 选择 PTC2 输出，周期为 1ms，占空比为 50%，与设定符合。



FTM0CH2 选择 PTC3 作为输出，周期为 1ms，占空比为 75%,与设定符合。



FTM1CH1 选择 PTA9 作为输出，周期为 2ms，占空比为 25%，与设定符合。

2.3.5 特别说明

1.不同的的 PWM 工作模式下 PWM 的周期与占空比的计算方法是不同的，对于 PWM 来说有四种工作模式

1 边缘对齐型，初始为低电平

2 边缘对齐型，初始为高电平

3 中间对齐型，初始为低电平

4 中间对齐型，初始为高电平

2.我们选择的是第 2 种，边缘对齐型，初始为高电平。

这种模式周期计算方式为（FTMx_MODE-FTMx_CNTIN）*（时钟周期）/（分频因子）

FTMx_MODE 在 pwm.h 中宏定义为 PWMx_CYCLE

FTMx_CNTIN 取 0

我们的时钟频率为 50Mhz

3 占空比的计算方式为（duty_count）/（FTMx_MODE-FTMx_CNTIN）

duty_count 为 ftm_pwm_output()函数的传入值

FTMx_MODE 在 pwm.h 中宏定义为 PWMx_CYCLE

FTMx_CNTIN 取 0

4 每个通道可能有不只一个引脚作为输出，可在 pwm.h 的宏定义中选择其输出引脚

如 ： `#define FTM0CH0_PIN  PORT_C1`

将 FTM0CH0 选择在 PORT_C1 上进行输出。

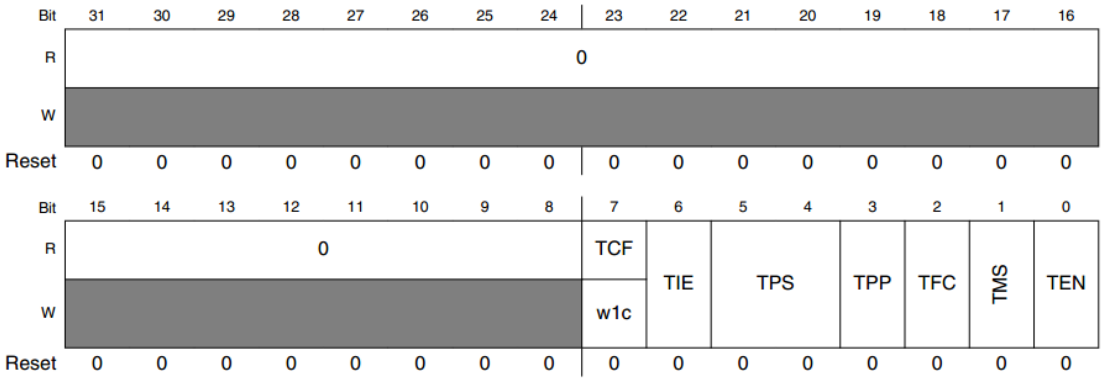
3 LPTMR 模块

3.1 LPTMR 简介

LPTMR 模块可以配置成定时器模式或者输入脉冲捕捉模式。

3.2 LPTMR 寄存器

（1）LPTMR 状态控制寄存器



- TCF[7]:当计数器与要比较的值相同时置 1；写 1 清除该位。
- TIE[6]：等于 1 时中断使能，TCF 置 1 时产生中断。
- TPS[5-4]：脉冲累加时选择输入引脚。
- TPP[3] :脉冲累加模块的输入信号极性设置。注意只能在 LPTMR 禁止的时候改变改值。

TPP[3]=0	在输入信号上升时计数器增加；
TPP[3]=1	在输入信号下降时计数器增加；

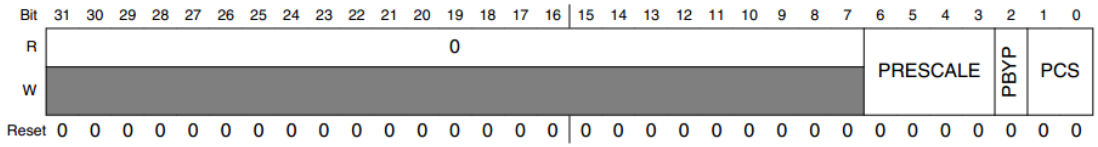
- TMS[1]:

TMS[1]=0	定时模块
TMS[1]=1	脉冲累加

- TEN[0]:

TEN[0]=0	禁止 LPTMR 内部逻辑清零
TEN[0]=1	使能 LPTMR

(2)LPTMR 预分频寄存器



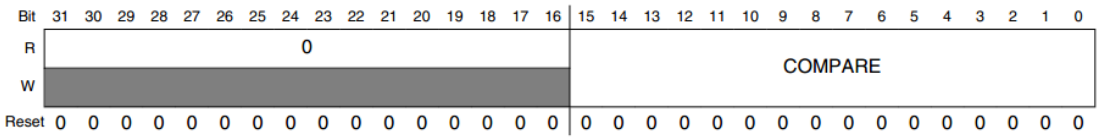
- PRESCALE[6-3]:在定时模块计算预分频系数；在脉冲模块计算脉冲宽度。
- BPYP[2]:

BPYP[2]=0	可以以时钟频率的 2-65536 倍为单位累加。
BPYP[2]=1	计数器只能以时钟频率的一倍为单位累加。

- PCS[1-0]:预分频时钟选择。

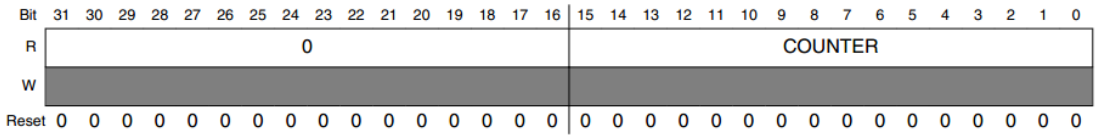
LPTMR0_PSR[PCS]	Prescaler/glitch filter clock number	Chip clock
00	0	MCGIRCLK — internal reference clock (not available in VLPS/LLS/VLLS modes)
01	1	LPO — 1 kHz clock
10	2	ERCLK32K — secondary external reference clock
11	3	OSCERCLK — external reference clock

(3)LPTMR 比较寄存器



COMPARE[0-15]:设置 LPT 计数器的累加的终值。

(4)LPTMR 计数寄存器



COUNTER[0-15]:返回当前计数器中的值。

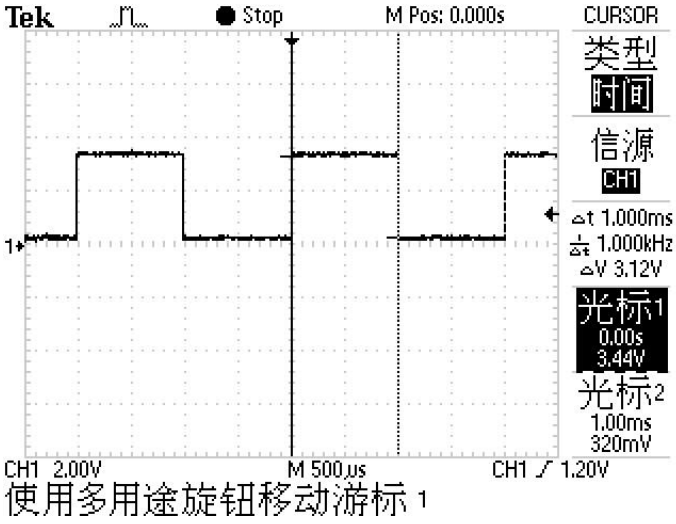
3.3 LPTMR 定时器

在定时器模块外部时钟没有使用，定时时只用内部高频的 2MHZ 和 LPO 的 1KHZ 来做时钟，时钟可经过 LPTMRx_PSR 寄存器预分频；然后该模块的一个 16 位计数器开始计数，计数器初值不能设定，只能读取，默认为 0；计数器终值由 LPTMRx_CMR 寄存器设置。当计数器累加到终值后 TCF 标志位置位，在进入下一轮定时。

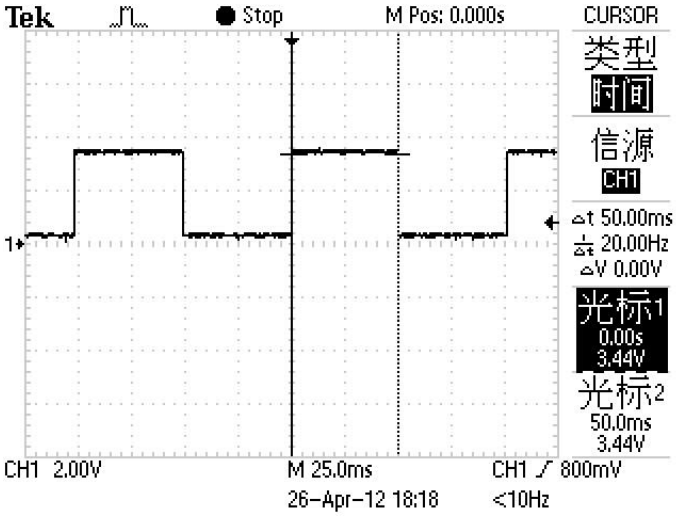
3.3.1 部分代码解析

```
switch(lptmr_timer_clock)
{
    case LPTMR_INTERNAL_CLOCK:
        SIM_SCGC5|=SIM_SCGC5_LPTIMER_MASK;    /* 加载LPT模块时钟    */
        lptmr_registers_clear();
        MCG_C1|=MCG_C1_IRCLKEN_MASK;          /* 使能内部参考时钟    */
        MCG_C2|=MCG_C2_IRCS_MASK;              /* MCG_C2[IRCS]=1,使能快速内部参考时钟（2MHz） */
        /* 配置 LPTMR参数 */
        LPTMR0_CMR=LPTMR_CMR_COMPARE(LPTMR_TIMER_COMPARE_INTERNAL);
        /* 设置比较寄存器值 */
    LPTMR0_PSR=LPTMR_PSR_PCS(LPTMR_INTERNAL_CLOCK)|LPTMR_PSR_PRESCALE(LPTMR_TIMER_PRESCALE8); /* 使用内部时钟，系数预分频为8*/
        LPTMR0_CSR|=LPTMR_CSR_TEN_MASK;/* 开启LPT模块设置*/
        break;
    case LPTMR_LPO_CLOCK:
        SIM_SCGC5|=SIM_SCGC5_LPTIMER_MASK;      /*加载LPT模块时钟*/
         lptmr_registers_clear();               /* 把LPTMR寄存器清零 */
        /* 配置 LPTMR */
        LPTMR0_CMR=LPTMR_CMR_COMPARE(LPTMR_TIMER_COMPARE_LPO);
/*设置比较寄存器值 */
       LPTMR0_PSR=LPTMR_PSR_PCS(LPTMR_LPO_CLOCK)|LPTMR_PSR_PBYP_MASK;
/*设置PBYP为1，计数器一次一次累加*/
        LPTMR0_CSR|=LPTMR_CSR_TEN_MASK;     /*开启LPT模块设置*/
          break;
    }
```

3.3.2 测量结果



上图为 LPTMR_INTERNAL 定时器，定时 1ms，结果正确。



上图为 LPTMR_LPO 定时器，定时 50ms，结果正确。

3.4 LPTMR 输入捕捉

输入捕捉可以捕捉 PTC5 和 PTA19 两个引脚的脉冲，通过设置 LPTMRx_CSR 寄存器中的 TMS=1 来进入该模式，捕捉次数通过 LPTMRx_CMR 寄存器来设定，不过该值最大只能为 65536；当 LPTMRx_CSR 中 TPP=0 时捕捉上升沿，TPP=1 时捕捉下降沿，默认捕捉的是下降沿。当捕捉次数达到 LPTMRx_CMR 值后 TCF 标志位置位，同时计数器清零，进入下一轮捕捉。

3.4.1 捕捉引脚设定

捕捉引脚通过 LPTMRx_CSR 中[TPS]来设定：

LPTMR_CSR[TPS]	Pulse counter input number	Chip input
00	0	CMP0 output
01	1	LPTMR_ALT1 pin
10	2	LPTMR_ALT2 pin
11	3	Reserved

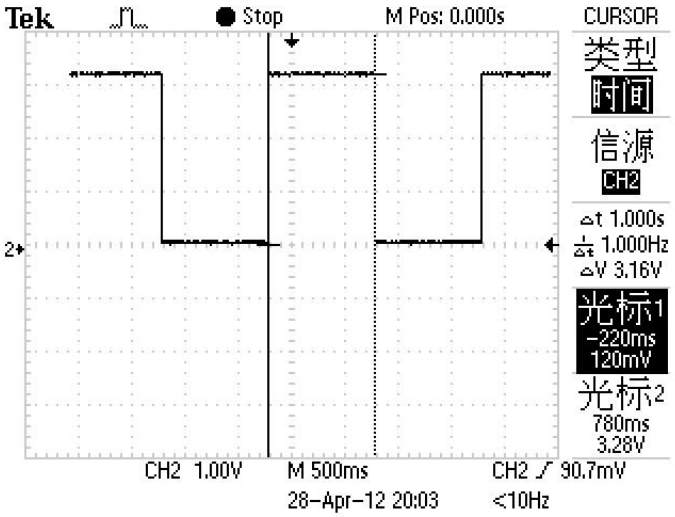
110	D8	PTC5/ LLWU_P9			PTC5/ LLWU_P9	SPI0_SCK		LPT0_ALT2	FB_AD10	CMP0_OUT
73	M11	PTA19	XTAL	XTAL	PTA19		FTM1_FLT0	FTM_CLKIN 1		LPT0_ALT1

3.4.2 代码解析

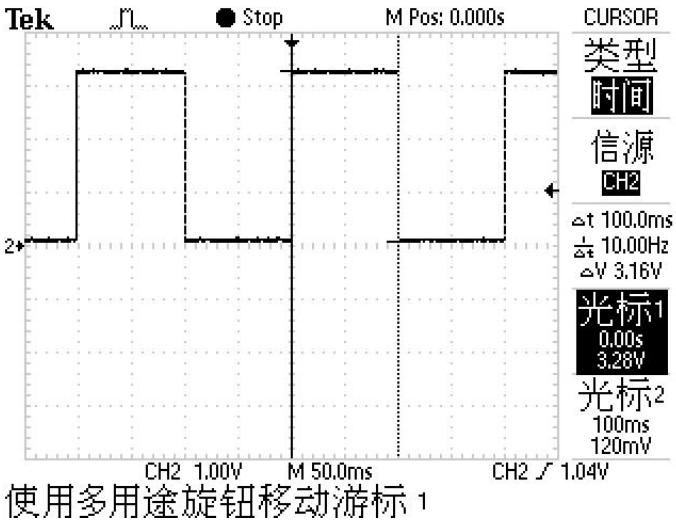
```
switch(lptmr_pulse_pin)
{
   case LPTMR_CAPTURE_PTA19:
   SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;     /*打开 PORTA 时钟*/
   PORTA_PCR19=PORT_PCR_MUX(0x6);     /*在PTA19上使用 ALT6*/
   break;
   case LPTMR_CAPTURE_PTC5 :
      SIM_SCGC5 |= SIM_SCGC5_PORTC_MASK; /*打开 PORTC 时钟*/
      PORTC_PCR5=PORT_PCR_MUX(0x4);  /*在PTC5上使用 ALT4*/
   break;

   default:
      return 1;
```

```
    }
    LPTMR0_PSR|=LPTMR_PSR_PBYP_MASK;
/*设定PBYP为1即每捕捉到一次上升沿计数器累加一次*/
    LPTMR0_CMR=LPTMR_CMR_COMPARE(LPTMR_PULSE_COMPARE);
/*设置比较值*/
    LPTMR0_CSR=LPTMR_CSR_TPS(lptmr_pulse_pin)|LPTMR_CSR_TM
S_MASK;   /*设置lptmr使用选择的引脚,进入脉冲累加模式,上升沿捕捉*/
    LPTMR0_CSR|=LPTMR_CSR_TEN_MASK;      /*开启 LPT 模块*/
```

3.4.3 测量结果



上图为 LPTMR 捕捉 PTA19 引脚，捕捉次数为 1000 次，脉冲频率 1KHz，结果正确。



上图为 LPTMR 捕捉 PTC5 引脚，脉冲频率 1KHz，捕捉次数 100 次，结果正确。

3.4.4 特别说明

LPTMR 模块只有一个 16 位的计数器，所以只能使用它做定时器或者脉冲捕捉。

在 LPTMRx_PSR 寄存器中有一个 PBYP[2]位，当该位置于 1 时，定时器中计数器只能每一个时钟计数一次，等价与预分频没用；此时输入脉冲捕捉计数器每捕捉一次上升沿或下降沿计数一次。当该位为 0 时，定时器预分频生效；此时输入脉冲捕捉将捕捉预分频个脉冲后计数器才计数一次。

4 PIT 模块

4.1 功能描述

周期性中断定时器模块是一组 32 位的定时器，有 4 个定时器，该模块一般用来触发外围模块或唤醒周期性中断。PIT 是一个递减计数器，首先给计数寄存器设定一个初值，每经过一个总线时钟，32 位定时器做一次减 1 操作，当减到 0 时，触发对应中断。

PIT 模块没有外围引脚。功能框图见图 3。

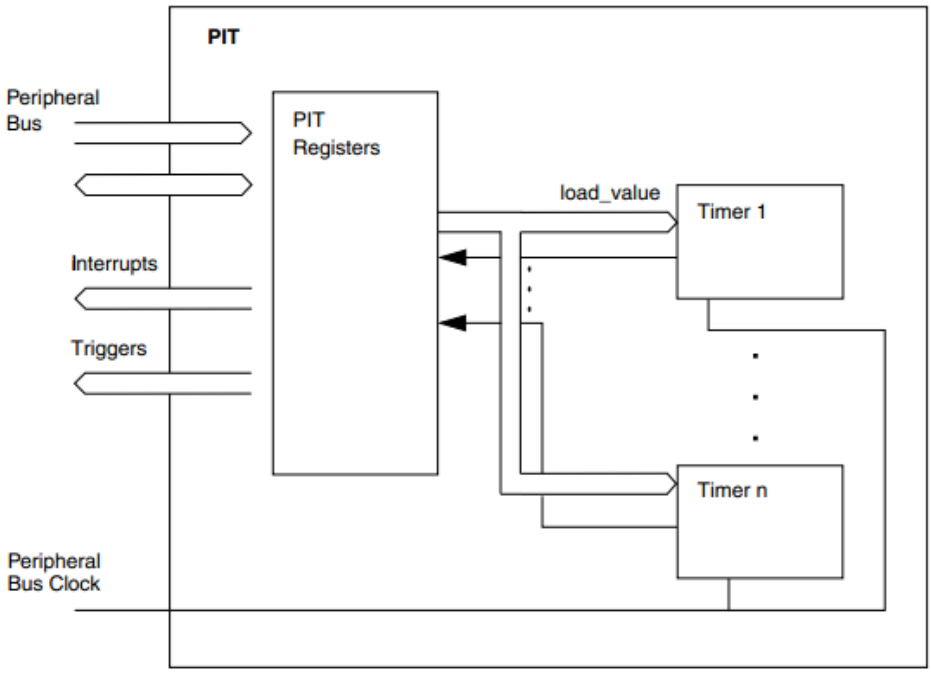
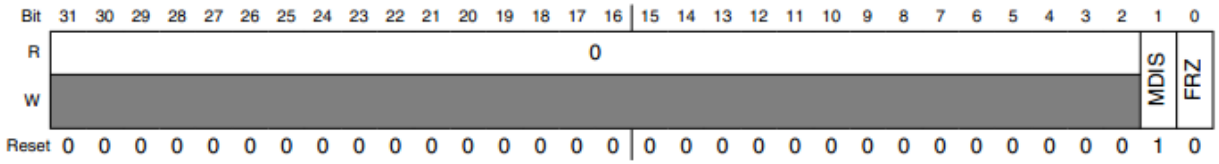


图 3.PIT 框图

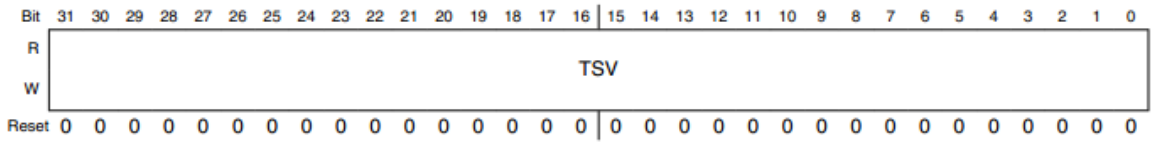
4.2 相关寄存器介绍

(1) PIT 模块控制寄存器（PIT_MCR）



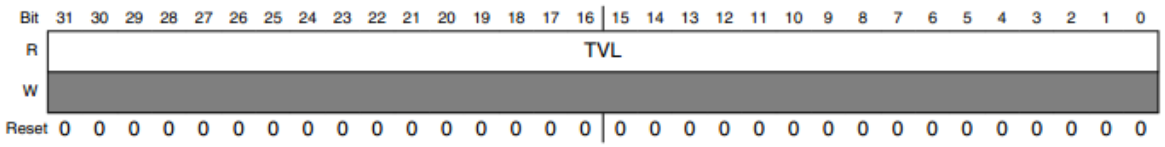
- MDIS：Module Disable,0 PIT 定时器的时钟使能；1 PIT 定时器的时钟不使能。
- FRZ：Freeze，0 定时器在调试模式下继续运行；1 定时器在调试模式下不运行。

(2) 定时器装载值寄存器（PIT_LDVALn）



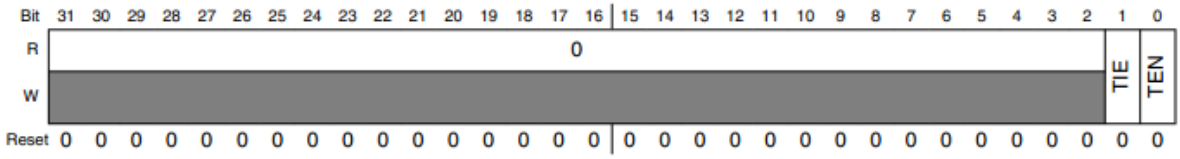
这个寄存器存放计数的初始值，定时器会在总线时钟下递减计数，直到减到 0 产生中断，并重新装载初始值。

(3) 当前定时器值寄存器（PIT_CVALn）



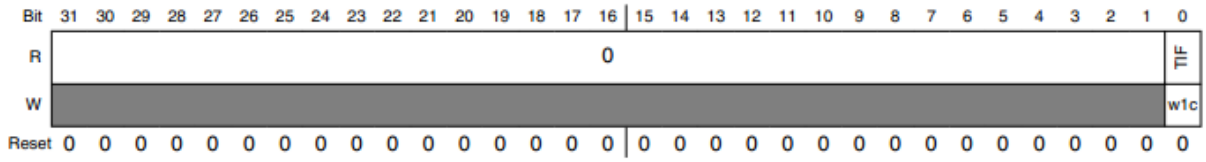
如果定时器已使能，寄存器显示当前定时器的值，如果定时器未使能，不要用此寄存器，因为此时它的值不可靠。

(4) 定时器控制寄存器（PIT_TCTRLn）



- TIE：定时器中断使能，0 不使能来自定时器 n 的中断请求；1 中断被请求不管 TIF 是否置位。
- TEN：使能时钟位，0 不使能定时器 n；1 使能定时器 n。

(5) 定时器标志寄存器（PIT_TFLGn）



- TIF：定时器中断标志位，0 没超时（time-out）;1 超时发生（time-out）。

4.3 部分代码解析

PIT 模块的代码由胡春旭学长修改优化后，主要代码如下：

```
/**   @brief  PIT 初始化函数
 *    @param  pitno     定时器模块号
 *    @param  timeout   设定时间的值
*/
void pit_int(ID pitno, uint32_t timeout)
```

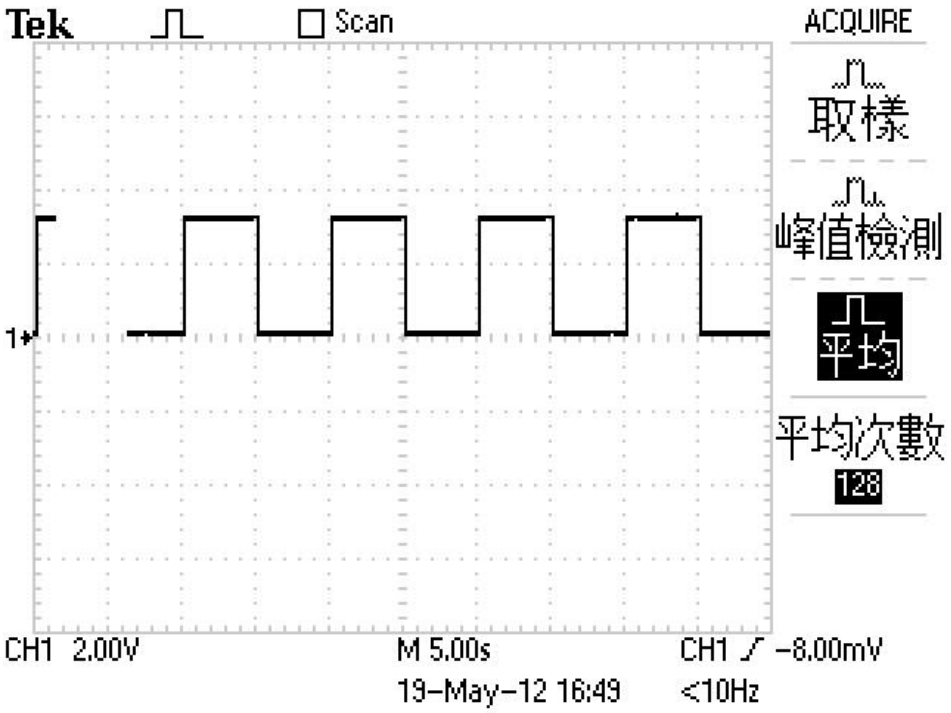
```
    {
        SIM_SCGC6|=SIM_SCGC6_PIT_MASK;                  /* 使能 PIT 时钟 */
        PIT_MCR&=~(PIT_MCR_MDIS_MASK);                  /* 使能 PIT 模块时
钟 */
        PIT_MCR|=PIT_MCR_FRZ_MASK;                      /* 调试模式下禁止
*/
        PIT_LDVAL(pitno)=timeout;                       /* 设置周期 */
        PIT_TCTRL(pitno)|=PIT_TCTRL_TEN_MASK;           /* 使能 pit 模块运
行 */
        PIT_TCTRL(pitno)&=~(PIT_TCTRL_TIE_MASK);        /* 关 pit 中断 */

         /* 中断向量表注册，中断优先级设置 */
        switch(PIT_ID_SET(pitno))
        {
             case PIT_TIMER0:
                 exc_install(INT_PIT0, pit0_timer_handler);
                 exc_set_pri(INT_PIT0, 1);
```

4.4 程序调试与测试

PIT 模块没有外围引脚，但可以触发外围模块，调试的方法是在程序中定时 5 秒，没 5s 产生一次中断，中断程序让 LED 灯闪烁。

现象是 LED 灯每 5 秒闪烁一次。

测得 LED 灯引脚电平波形如下：



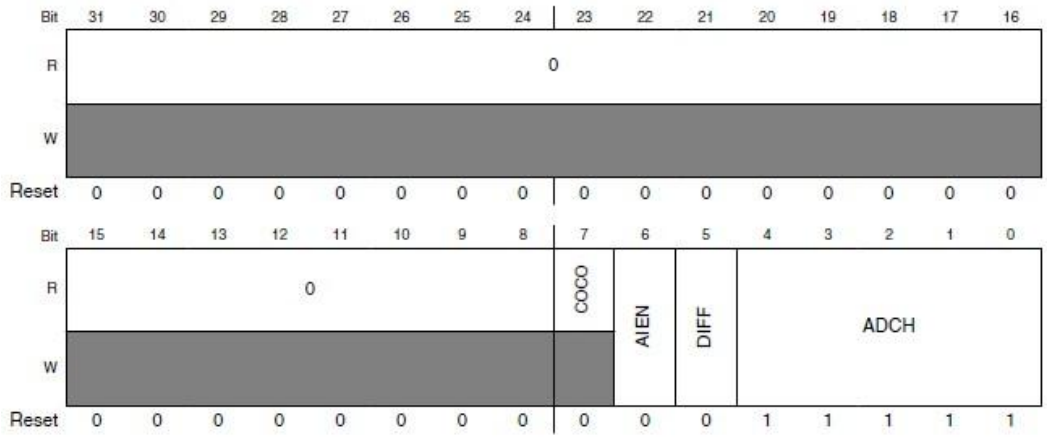
5 ADC 模块

5.1 简介

ADC有两个模块，每个模块都有4个双工和24个单工输入。输出模式有16位、13位、11位、9位不同的双工模式，或者16位、12位、10位、8位的单工模式。 具有硬件平均的功能。

5.2 寄存器说明

(1) ADC 状态控制寄存器 1（ADCx_SC1n）



- COCO：转换完成标志

COCO 标志位是只读的，当比较功能取消（ACFE=0），硬件均值功能也取消（AVGE=0）时，每当转换完成之后就会置位该位。当允许比较功能（ACFE=1）时只有当比较结果是正确的时候，转换完成时才会置位 COCO 标记位。当硬件均值是有效的（AVGE=1），只有当转换号选择完成时才会置位 COCO 标记位。当校准次序完成的时候，也会置位在寄存器 SC1A 中的标记为 COCO。当对寄存器 SC1A 进行写操作或者对寄存器 Rn 进行读操作都会清除标记位 COCO。

0	**转换没有完成**
1	转换完成

- ALEN：中断使能

AIEN使能转换完成中断。当AIEN为高时置位COCO就　会引发一个中断。

0	**转换完成中断取消**
1	转换完成时中断有效

- DIFF：双工模式使能

DIFF 配置 ADC 在双工模式下进行操作。当配置有效时，该模式会自动选择一个双工通道，改变转换算法和周期号来完成转换。

0	选择单工转换和输入通道
1	选择双工转换和输入通道

- ADCH：双工模式使能

输入通道选择：

ADCH 是由 5 位组成可以用于选择输入通道。输入通道解码依赖于这 5 位的

值。当选择位全部设置为 1111 时连续近似值转换器子系统会关闭。该特征可以明确地结束 ADC，同时可以将输入通道与所有其他的资源隔离开来。

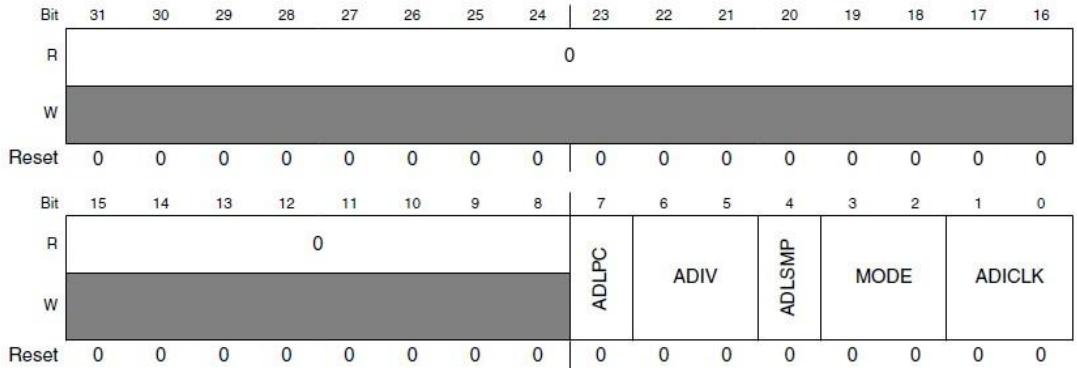
结束正在执行的转换可以防止新的转换发生。当正在执行的转换无效时，没有必要将通道选择位全部设置为 1 来将 ADC 置于低功耗状态，因为转换完成之后模块会自动进入低功耗状态。

00000	**当 DIFF=0，DADP0 选择为输入；当 DIFF=1，DAD0 选择为输入。**
00001	当 DIFF=0，DADP1 选择为输入；当 DIFF=1，DAD1 选择为输入。
00010	当 DIFF=0，DADP2 选择为输入；当 DIFF=1，DAD2 选择为输入。
00011	当 DIFF=0，DADP3 选择为输入；当 DIFF=1，DAD3 选择为输入。DAD0-DAD3 与输入引脚对 DADPx 和 DADMx 有关。
00100	当 DIFF=0，AD4 选择为输入；当 DIFF=1，该位保留。
00101	当 DIFF=0，AD5 选择为输入；当 DIFF=1，该为保留。
00110	当 DIFF=0，AD6 选择为输入；当 DIFF=1，该为保留。
00111	当 DIFF=0，AD7 选择为输入；当 DIFF=1，该为保留。
01000	当 DIFF=0，AD8 选择为输入；当 DIFF=1，该位保留。
01001	当 DIFF=0，AD9 选择为输入；当 DIFF=1，该位保留。
01010	当 DIFF=0，AD10 选择为输入；当 DIFF=1，该位保留。

01011	当 DIFF=0，AD11 选择为输入；当 DIFF=1，该位保留。
01100	当 DIFF=0，AD12 选择为输入；当 DIFF=1，该位保留。
01101	当 DIFF=0，AD13 选择为输入；当 DIFF=1，该位保留。
01110	当 DIFF=0，AD14 选择为输入；当 DIFF=1，该位保留。
01111	当 DIFF=0，AD15 选择为输入；当 DIFF=1，该位保留。
10000	当 DIFF=0，AD16 选择为输入；当 DIFF=1，该位保留。
10001	当 DIFF=0，AD17 选择为输入当 DIFF=1，该位保留。
10010	当 DIFF=0，AD18 选择为输入；当 DIFF=1，该位保留。
10011	当 DIFF=0，AD19 选择为输入；当 DIFF=1，该位保留。
10100	当 DIFF=0，AD20 选择为输入；当 DIFF=1，该位保留。
10101	当 DIFF=0，AD21 选择为输入；当 DIFF=1，该位保留。
10110	当 DIFF=0，AD22 选择为输入；当 DIFF=1，该位保留。
10111	当 DIFF=0，AD23 选择为输入；当 DIFF=1，该位保留。
11000	保留。

11001	保留。
11010	当 DIFF=0，温度传感器选择为输入；当 DIFF=1，温度传感器选择为输入。
11011	当 DIFF=0，Bandgap 选择为输入；当 DIFF=1，Bandga 选择为输入。
11100	保留。
11101	当 DIFF=0，VEREFSH 选择为输入；当 DIFF=1，-VREFSH 选择为输入。在 SC2 寄存器中电压接口由 REFSL 位决定。
11110	当 DIFF=0，VREFSL 选择作为输入；当 DIFF =1,该位保留。在寄存器 SC2 中电压接口选择由 REFSEL 决定。
11111	模块停止工作。

(2) ADC 配置寄存器 1（ADCx_CFG1）



- ADLPC 低电压配置：

ADLPC 控制连续近似值转换器的电压配置。当不要求高采样率的时候供电充足。

ADLPC=0	正常供电配置
ADLPC=1	低电压配置。在要求最小时钟速度时电压减小。

- ADIV 时钟分频选择：

ADIV 选择分频系数处理时钟 ADCK

ADIV=00	分频系数为 1
ADIV=01	分频系数为 2
ADIV=10	分频系数为 4
ADIV=11	分频系数为 8

- ADLSMP 采样时间配置：

ADLSMP 会根据选择的转换模式选择不同的采样次数。该位能够在采样期适应高阻抗输入以达到精确采样或者为低 输入加快转换速度。如果正在执行的转换被取消同时不要求高转换率，则长时间采样也可以用在更低的电压状态下。当 ADLSMP=1，长时间采样选择位置位。

ADLSMP =0	短采样时间
ADLSMP =1	长时间采样

- MODE 转换精度选择：

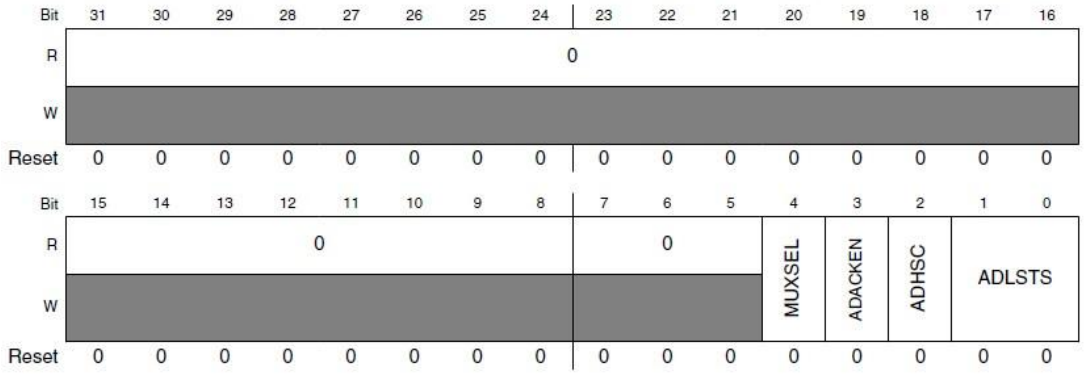
MODE=00	当DIFF=0：为单工8为转换；当DIFF=1，为带有 的9位双工转换。
MODE=01	当DIFF=0：为单工的12位转换；当DIFF=1，为 的13位双工转换。
MODE=10	当DIFF=0：为单工12位转换；当DIFF=1，为带 的13位双工转换。
MODE=11	当DIFF=0：为16位的单工转换；当DIFF=1，为 出的16为双工转换

- ADICLK 输入时钟选择：

ADICLK 位选择输入时钟源来处理内部时钟，ADCK。注意当选择 ADACK 时钟源，不要求提前开始转换。选择该位同事又不需要提前开始转换，异步时钟在转换开始时有效，在转换结束时关闭。在这种情况下，每次时钟源再次有效时，都有一个相关的时钟开始时间被推迟。

ADICLK=00	总线时钟
ADICLK=01	总线时钟/2
ADICLK=10	交替时钟（ALTCLK）
ADICLK=11	异步时钟（ADACK）

(3) ADC 配置寄存器 1（ADCx_CFG1）



- MUXSEL ： ADC 复用选择

ADC 复用选择位用于改变 ADC 复用用于 ADC 通道设置

MUXSEL =0	选择 ADxxa
MUXSEL =1	选择 ADxxb

- ADACKEN :异步时钟输出使能

ADACKEN=0	取消异步时钟输出；只有 ADICLK 选择异步时钟才有效同时转换也有效。
ADACKEN=1	不管 ADC 的状态是什么，异步时钟和

	时钟输出都有效。

- ADHSC ： 高速配置

ADHSC 配置 ADC 高速操作。转换持续会触发允许高速转换时钟。

ADHSC=0	正常转换次序
ADHSC=1	高速转换次序

- ADLSTS ： 选择长采样时间

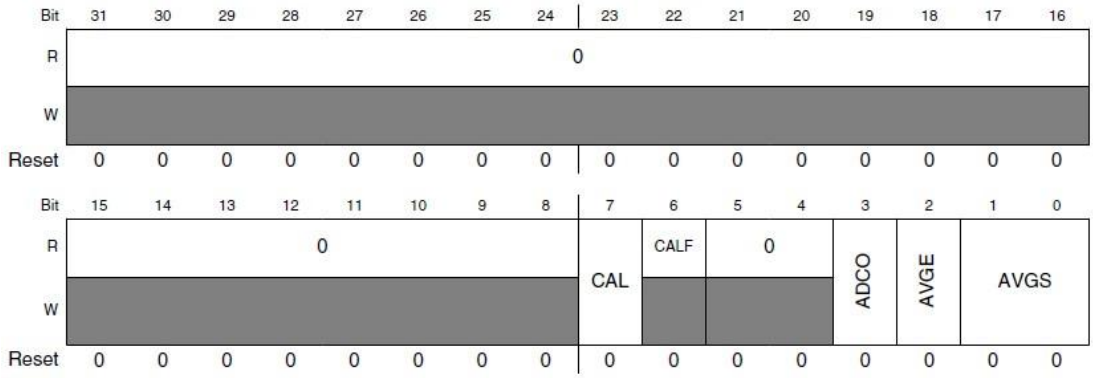
当选择了长采样时间（ADLSMP=1）时， ADLSTS 选择扩展采样时间中的一个作为采样时间。该特点允许高阻抗输入，可以达到精确采样或在低阻抗输入时可以将转换速度最大化。只要持续转换有效并且不要求高转换率，更长的采样时间也可以用在电压相对较低的情况。

ADLSTS=00	默认的最长采样时间，另外附加 20 个 ADCK 周期；总共有 24 个 ADCK 周期的采样时间。
ADLSTS=01	另外附加 12 个 ADCK 周期；总共有 16 个 ADCK 周期的采样时间。
ADLSTS=10	另外附加 6 个 ADCK 周期；总共有 10 个 ADCK 周期的采样时间。
ADLSTS=11	另外附加 2 个 ADCK 周期；总共有 6 个 ADCK 周期的采样时间。

(4) ADC 数据结果寄存器 （ADCx_Rn）

保存 ad 转换结果

(5) ADC 状态控制寄存器 3（ADCx_SC3）



我们所用的只有 AVGE 与 AVGS

(5)AVGE 硬件平均功能使能：

AVGE=0	禁止硬件平均功能
AVGE=1	使能硬件平均功能

(6)AVGS 硬件均值选择：

AVGS 选择硬件平均取样的次数

AVGS=00	4 个采样的均值
AVGS=01	8 个采样的均值
AVGS=10	16 个采样的均值
AVGS=11	32 个采样的均值。

5.3 部分代码解析

```
/*选择正常供电配置，选择分频系数，选择长采样时间，选择转换精度，选择输入时钟*/
ADC0_CFG1 = ADLPC_NORMAL
          | ADC_CFG1_ADIV(ADC_PRESCALE)
          | ADLSMP_LONG
          | ADC_CFG1_MODE(ADC_ACCURACY)
          | ADC_CFG1_ADICLK(ADC_CLOCK);

/*复用选择ADCxa，取消异步时钟输出，高速转换模式，选择20个额外时钟*/
ADC0_CFG2 =  MUXSEL_ADCA
          | ADACKEN_DISABLED
          | ADHSC_HISPEED
          | ADC_CFG2_ADLSTS(ADLSTS_20);

/*选择硬件平均的方式*/
ADC0_SC3  = ADC_HARDWARE_AVG_MODE;

/*使能中断，单端模式，选择通道*/
ADC0_SC1A = AIEN_ON | DIFF_SINGLE | ADC_SC1_ADCH(Channel);
```

5.4 测试结果记录

测试方法为用一滑动变阻器调节模拟电压输入，将 AD 转换结果通过串口发送到电脑上观察，结果如下：

通道	精度	输入电平	五次 AD 转换结果					理论值
ADC0_DM0	16	1.90	38819	38811	38824	38815	38817	38707
ADC0_DP0	16	1.11	22785	22826	22841	22817	22739	22732
ADC1_DM0	16	0.39	8206	8199	8202	8190	8185	7987
ADC1_DP0	16	2.50	51160	51143	51153	51160	51137	51200

考虑到万用表测量误差，结果基本与理论值相符

参考文献

1. Freescale Semiconductor, Inc. 《K60 Sub-Family Reference Manual》 Rev. 6, Nov 2011.

2. 苏州大学 苏州大学最小系统板资料