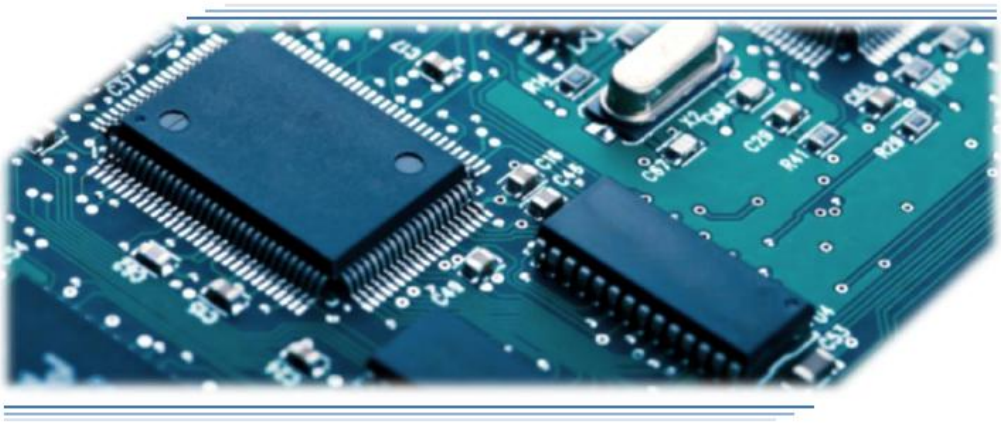
FM33LG0XX 开发注意事项



上海复旦微电子集团股份有限公司

Shanghai Fudan Microelectronics Group Company Limited

开发者论坛：http://www.fmdevelopers.com.cn

本资料是为了让用户根据用途选择合适的上海复旦微电子集团股份有限公司（以下简称复旦微电子）的产品而提供的参考资料，不转让属于复旦微电子或者第三者所有的知识产权以及其他权利的许可。
在使用本资料所记载的信息最终做出有关信息和产品是否适用的判断前，请您务必将所有信息作为一个整体系统来进行评价。
采购方对于选择与使用本文描述的复旦微电子的产品和服务全权负责，复旦微电子不承担采购方选择与使用本文描述的产品和服务的责任。除非以书面形式明确地认可，复旦微电子的产品不推荐、不授权、不担保用于包括军事、航空、航天、救生及生命维持系统在内的，由于失效或故障可能导致人身伤亡、严重的财产或环境损失的产品或系统中。
未经复旦微电子的许可，不得翻印或者复制全部或部分本资料的内容。
今后日常的产品更新会在适当的时候发布，恕不另行通知。 在购买本资料所记载的产品时，请预先向复旦微电子在当地的销售办事处确认最新信息，并请您通过各种方式关注复旦微电子公布的信息，包括复旦微电子的网站(http://www.fmsh.com/)。
如果您需要了解有关本资料所记载的信息或产品的详情，请与上海复旦微电子集团股份有限公司在当地的销售办事处联系。

商 标

上海复旦微电子集团股份有限公司的公司名称、徽标以及“复旦”徽标均为上海复旦微电子集团股份有限公司及其分公司在中国的商标或注册商标。

FM33LG0XX 系列芯片是一款 M0 内核的 ARM 芯片，在客户开发中通常会遇到一些普遍的问题，本文针对这些问题进行了详细的描述，以加快客户的开发过程。具体细节请参考相关手册、例程。

改版记录

2021.1	初版	
2021.1	V02	增加 LPUART 的注意事项
2021.6	V03	芯片改版，部分 V02 的注意事项中的问题在新版芯片中已经修复。市面上仍有少部分旧版本芯片，请查看 V02 版本的开发注意事项。
2021.8	V04	VBAT 独立供电注意事项
2022.2	V05	1 VBAT 独立供电注意事项修改 2 新增 FL 库使用注意事项章节，介绍 FL 库新增延时函数及其初始化流程。
2022.3	V06	新增 ADC 和 DAC 使用注意事项
2022.6	V07	1 增加 13 节 LSCLK 的说明 2 增加 FL 库使用中关于 LSCLK 的修改说明

1． GPIO

当 XTLF 的振荡强度≤250nA 时，PH15 输出高频信号会对 XOUT 产生影响。

2． ADC

1 使用 ADC 时，无论使用什么基准电压， VREFN 必须接地。

2 ADC 使用 VREFP_VREG 电路产生的电压做基准源时 ,需要在 VREFP 管脚外接 1uf 电容。在 VREFP_VREG 关闭时，电容上会被芯片内部微小的漏电慢慢充电，几小时后电容可能会被充到 VDD。这时打开 VREFP_VREG 进行 ADC 采样，当前的 ADC 基准电压是高于设定值导致采样结果异常。

解决方法：

（1） 功耗允许情况，VREF1P2(~1.5μA)，VREFP(~50μA)常开

（2） 硬件上在 VREFP 引脚对地接 M 级电阻放电

（3）使用 ADC 采样前开启 DAC，为 VREFP 累积电荷提供泄放通路。 DAC 开启 30ms 左右可从 5V 放电到 1V 左右。

（4） 使用 VDDA 作为 ADC 基准源

3 ADC 校准完成后会多产生一个 EOC 信号，因此建议等待校准完成后，关闭 ADC，消除 EOC 信号，关闭 ADC 校准值仍然保存。

```
while (FL_ADC_IsActiveFlag_EndOfCalibration(ADCx) == 0); //等待校准完成
FL_ADC_ClearFlag_EndOfCalibration(ADC);
FL_ADC_Disable(ADCx);
```

采样时再清下标志

```
FL_ADC_ClearFlag_EndOfConversion(ADC);
```

4 ADC 在进行校准时不能设置过采样。

3． 内部基准

ulpbg_vdd trim 值没有 autoload，因此在初始化时建议添加：

```
#define ULPBG_LDT_TRIM          (*(uint32_t *)0x1FFFFA98)
PMU->ULPB_TR= ULPBG_LDT_TRIM;
```

例程中已添加。

4． CMU（时钟管理单元）

systick 建议选择内部时钟。

5． IWDT（独立看门狗）

IWDT 的读结果会保留在总线上，建议读取 IWDT 寄存器后必须读 IWDT 一个为 0 的寄存器，SEVR 寄存器读一直为 0。

如：READ_REG(IWDTx->CNT);读 IWDT 寄存器 CNT，结果留在总线上；

READ_REG(IWDTx->SERV);读 IWDT 寄存器 SERV，总线上留存的数据清 0；

驱动函数中已做处理，可以直接调用 IWDT 相关函数。

6． SPI

SPI 判断发送完成，TXBUF 标志不等于移位寄存器发送完成，BUSY 标志置位有个同步过程，有从 0 到 1 的一个过程，写完 TXBUFF，BUSY 没有立刻置 1 还是 0，在 SPI 的工作时钟和系统时钟相差特别大时，这个问题会特别明显。因此建议在开发过程中采用以下方式：

1）全双工模式：SPI 原理是主从移位寄存器交换，可以判断 RXBUF 标志来判断接收完成。

2）半双工模式：发送完成不能判断 RXBUF，使用 BUSY，先判断 BUSY 为 1，再判断 BUSY 为 0，发送完成。

```
FL_WriteTXBuff(SPI1,data);//写 TXBUF
while（SPI1->ISR &(1U<<8)）==(1U<<8));//等待 BUSY 置 1，使用寄存器操作可以确保主时钟在64MHZ 也可以保证程序逻辑没有问题
while(FL_SPI_IsActiveFlag_Busy(SPI1));//BUSY 变 0，发送完成
```

7． LPTIM

ARR 不能写 0、1。

8． 比较器

窗口功能不建议使用。

9． DMA

1）size 不会自动更新，memory 地址会自动更新，因此查询发送的进度可以看 memory 地址。

2）DMA 的传输完成标志只是代表外设和 RAM 之前数据搬移完成，不代表外设完成相应动作。如 UART 发送的 DMA 应用，DMA 通道传输完成标志置起，但 UART 的发送还没有完成。拿 UART 举例，建议在 DMA 传输完成标志置起后再查询 UART 的发送完成标志 TXSE。

10． DAC

（1）DAC 外设复位和 EN 后，DAC 使能后输出的是数据是上一次 DAC 输出的电平。DAC 有两个数据寄存器，软件可操作的是 DHR，数据写入 DHR，写入的动作会将 DHR 的数据搬入另外一个寄存器 DAC_DOR(程序不可见)，DAC 才输出对应的电平，DAC 输出的是 DOR 中电平。程序复位 DAC 模块或重新使能 DAC 模块和写寄存器的动作不同，初始化的是 DHR 寄存器，DOR 还是保存之前的数据。因此在使能 DAC 之前，把需要输出的数据先写入数据保持寄存器 DHR。

（2）DMA 触发 DAC 只是把 RAM 中数据搬入 DHR，需要第二个数据搬入 DHR，才把第一个数据搬入影子寄存器从 DAC 输出。原理如上一条描述,另外 DMA 模式的 DAC 是触发模式，需要触发才能把 DHR 数据搬入 DOR。

举例：

RAM 中待搬入 DAC 数据寄存器的数据（1000，2000）

DMA 数据长度设置为 3

DMA 搬运第一次数据 DHR 写入 1000

DMA 搬运第二次 DOR 写入 1000，DHR 写入 2000

DMA 搬运第三次 DOR 写入 2000

此时 DACDHR 中为一个未知的数据，所以下次再使能 DAC 前需要把 DHR 初始化一下。

（3）DAC 当使用内部 VREFP_VREG 电路产生的电压做基准源时，当 VREFP_VREG 长时间关闭再打开会有和 ADC 一样的问题（参考 ADC 中的相关注意事项），开始的几十 ms,输出波形可能有上冲。

11． LPUART

LPUART 使用 RCLF 做工作时钟时，波特率为 9600 时容错非常低不建议使用。

12． VBAT

VABT 独立供电需要电源切换功能时注意以下两点：

1 关闭 BOR 下电复位，将 PDR 下电复位的阈值设置为最高 1.5V。

2 VBAT 接入电压不能高于 4.2V。

13． LSCLK

LSCLK 有两个源：外部低速晶体 XTLF 和内部 RC 低速环振 RCLP。

控制 LSCLK 输出的是 3 个寄存器:

1 CMU_SYSCLKCR.LSCATS:LSCLK 自动切换控制位，默认自动切换。

自动切换是指 XTLF 和 RCLP 同时有效时，LSCLK 默认使用 XTLF。当 XTLF 停振，LSCLK 自动切换到 RCLP，当 XTLF 恢复振荡时 LSCLK 自动切回 RCLP。

2 CMU_LSCLKSEL:这个寄存器的值只有在自动切换失效时才有效，不同的 LSCLKSEL 的配置可以决定 LSCLK 输出 XTLF 还是 RCLP。

3 CDIF_CR_INTF_IEN:这个寄存器是 VAO 电源域下的信号传输到 CPU 电源域的开关。

需要注意当 VAO 电源域到 CPU 电源域的通道断开时，CMU_SYSCLKCR.LSCATS 失效，即使 LSCATS 为 1 使能了自动切换，但 LSCLK 的输出还是取决于 LSCLKSEL 的值。

14． FL 库使用注意事项

14．1 延时

在 FM33LG0xx FL 库 v2.3.0 及之后的版本中，FL 库添加了一系列延时函数：

FL_DelayMs 毫秒延时；

FL_DelayUs 微秒延时；

FL_DelayMsStart/FL_DelayUsStart + FL_DelayEnd 用于执行带超时的循环；

这些函数都使用了__WEAK 弱定义，方便用户进行修改。FL 库使用 FL_Init()函数对如上功能进行初始化，请务必在使用 FL 库前先调用该函数！

FL 库使用了 CMSIS 标准的全局系统时钟频率值变量 SystemCoreClock 进行对 Systick 延时的计算（位于 system_fm33lc0xx.c/.h）。为了确保延时初始化正确，请在 调用 FL_Init 函数初始化前，按如下流程进行配置：

1 按照正常流程配置芯片时钟树；

2 如果主时钟源自内部来源（例如 RCHF），那么仅需要调用 SystemCoreClockUpdate()函数更新 SystemCoreClock 变量；如果主时钟源自外部（XTHF），那么需要配置全局变量 XTHFClock 为当前使用的外部晶体频率值（单位 Hz，默认值为 8M），再调用 SystemCoreClockUpdate()函数更新 SystemCoreClock 变量；

14. 2 LSCLK

例程 0.30，驱动 V2.2.0 版本

SystemInit()函数中默认宏定义 USE_LSCLK_CLOCK_SRC_XTLF 与 USE_LSCLK_AUTO_SWITCH，程序会执行以下代码：

```
CMU->SYSCLKCR |= 0x8000000U;
CMU->LSCLKSEL = 0xAAU;
```

但如第 13 节 LSCLK 中第三点描述，即使使能了自动切换，但如果关闭 VAO 到 CPU 的通道，自动切换功能失效。这时因为通道关闭 XTLF 信号也不能输出到 CPU 域，而 LSCLK 是 0xAA，LSCLK 被固定在 XTLF 上，等于 LSCLK 没有信号。

这点在写程序时需要注意。从例程 0.40 驱动 V2.3.0 开始，工程中 LSCLKSEL 选择 RCLP，同时使能自动切换且关闭 VAO 到 CPU 的通道。这样的配置下 LSCLK 使用 RCLP，在需要使用 XTLF 时如 RTCA 走时，打开 VAO 到 CPU 的通道这时 LSCLK 自动切换到 XTLF。

旧版本：

```
void SystemInit(void)
{

    #if defined(USE_IWDT_ON_STARTUP)
    CMU->PCLKCR1 |= 0x20U;              /* Enable IWDT Operation Clock */
    IWDT->CR = IWDT_OVERFLOW_PERIOD;    /* Configure IWDT overflow period */
    IWDT->SERV = 0x12345A5AU;           /* Enable IWDT */
    #endif

    /* Enable VREF Operation Clock */
    CMU->PCLKCR1 |= 0x1U << 12;

    /* Enable PAD Operation Clock */
    CMU->PCLKCR1 |= 0x1U << 7;

    /* CDIF: VAO->CPU Enable */
    CDIF->CR |= 0x1U << 1;

    #ifndef MFANG   /* MFANG handles clock configurations by itself */
    #ifdef USE_LSCLK_CLOCK_SRC_XTLF
    #ifdef USE_LSCLK_AUTO_SWITCH

        /* Enable LSCLK auto switch */
        CMU->SYSCLKCR |= 0x80000000U;

        /* LSCLK from XTLF */
        CMU->LSCLKSEL = 0xAAU;

    #else

        /* Disable LSCLK auto switch */
        CMU->SYSCLKCR &= 0x7FFFFFFFU;

        /* LSCLK from XTLF */
        CMU->LSCLKSEL = 0xAAU;

    #endif  /* USE_LSCLK_AUTO_SWITCH */
    #else

        /* Disable LSCLK auto switch */
        CMU->SYSCLKCR &= 0x7FFFFFFFU;

        /* LSCLK from RCLP */
        CMU->LSCLKSEL = 0x55U;

    #endif  /* USE_LSCLK_CLOCK_SRC_XTLF */
    #endif  /* MFANG */

    /* Keep timers running and disable IWDT && WWDT under debug mode */
    DBG->CR = 0x3U;
```

新版本：

```
void SystemInit(void)
{

    #if defined(USE_IWDT_ON_STARTUP)
    CMU->PCLKCR1 |= 0x20U;                /* Enable IWDT Operation Cl
    IWDT->CR = IWDT_OVERFLOW_PERIOD;      /* Configure IWDT overflow p
    IWDT->SERV = 0x12345A5AU;             /* Enable IWDT */
    #endif

    /* Enable VREF Operation Clock */
    CMU->PCLKCR1 |= 0x1U << 12;

    /* Enable PAD Operation Clock */
    CMU->PCLKCR1 |= 0x1U << 7;

    #ifdef USE_LSCLK_AUTO_SWITCH

        /* Enable LSCLK auto switch */
        CMU->SYSCLKCR |= 0x8000000U;
        CMU->LSCLKSEL = 0x55U;

    #else

        /* Disable LSCLK auto switch */
        CMU->SYSCLKCR &= 0x7FFFFFFU;
        CMU->LSCLKSEL = 0x55U;

    #endif  /* USE_LSCLK_AUTO_SWITCH */

    /* Keep timers running and disable IWDT && WWDT under debug mode
    DBG->CR = 0x3U;

    #ifdef USE_DEBUG_UNDER_SLEEP
        /* Keep debug connnection under sleep mode */
    DBG->CR |= 0x1U << 16;
    #endif
```