LGT8FX8P

FLASH Based 8bit Microcontroller

应用笔记

V1.0.0

2017/5/26

从 LGT8F8XD 移植到 LGT8FX8P

文档主要介绍 LGT8FX8D 与 LGT8FX8P 芯片的区别以及设计从 LG8F8XD 移植到 LGT8FX8P 时的注意事项。

关于 LGT8X8P 新增功能部分，请参考 LGT8FX8P 编程手册。

概述

LGT8FX8P 为 LGT8FX8D 的增强版本。LGT8FX8P 重点增强了 LGT8FX8D 的模拟性能，并增加了更多功能和更多的引脚。主要更新和增加的项目如下：

- 增强内部 32MHz RC 的精度和温度稳定性
- 12 位 ADC 增加了校准功能，保证在整个量程范围内的精度
- 内部参考改为 1.024V/2.048V/4.096V 三挡，并大幅提供参考电压的精度
- 增加差分放大器，增益 x1/8/16/32 可调
- 比较器增加了失调校准和数字滤波
- 更新 EEPROM 控制器，增加了更高效的连续数据存储模式
- 增加 1uA 低功耗模式(DPS2)
- 增加 16 位数字运算加速器(uDSC)
- 增加一个 16 位定时器 3，支持三路 PWM 输出
- 增加互补 PWM 的输出和死区控制
- SPI 控制器更新为支持 4 字节数据 FIFO
- 增加更多引脚，支持 QFP48 封装
- DAC 更改为独立模块；与 LGT8FX8D 相比减少了一路 DAC

以下章节主要是描述项目从 LGT8FX8D 移植到 LGT8FX8P 时，软件设计需要注意的事项。

ADC 相关注意事项

ADC 参考电压：

LGT8FX8P 内部参考电压为 1.024V/2.048V 以及 4.096V 三档。此部分与 LGT8FX8D 不同，在软件移植时，需要根据应用做适当调整。

在选择参考电压时，需要同时注意参考电压对系统电源的要求。为保证参考电压输出稳定，需要保证芯片的工作电压比所选择的参考电压大 0.7V。

内部参考出厂校准。校准参数存储在系统配置信息区域，并在上电过程中加载到寄存器中。

LGT8FX8P 上电后，默认加载 1.024V 的校准参数。如果换用其他参考，需要软件更新校准参数。

校准系数	地址	功能说明
VCAL1	0xCD	1.024V 校准参数
VCAL2	0xCE	2.048V 校准参数
VCAL3	0xCC	4.096V 校准参数
VCAL	0xC8	内部参考校准寄存器。将以上校准参数写入 VCAL 寄存器，完成对所选参考电压的校准。 比如 VCAL = VCAL2；校准 2.048V 参考电压。

参考电压的选择配置，请参考 LGT8FX8P 编程手册 ADC 章节中 ADMUX 寄存器的描述。

ADC 转换与校准：

LGT8FX8P 的 ADC 支持失调校准。通过在 ADC 转换时校准转换结果，可以有效增强 ADC 在整个量程范围的测量精度。ADC 的校准需要通过 SPN 位的控制进行连续两次采样，校准的同时也有滤波作用。

下面为 LGT8F8XP 转换的校准算法：

ADC 转换与校准示例

```
#include "lgt8f328p_spec.h"

// ADC 单次转换(无校准)
static u16 _adcReadRaw()
{
    u16 wtmp;

    ADCSRA |= 0x40;
    while((ADCSRA & 0x40) != 0x00);
    wtmp = ADCL;
    return (ADCH << 8) | wtmp;
}

// ADC 失调补偿与增益补偿
u16 adcRead()
{
   u16 pVal, nVal;

   ADCSRC |= (1 << SPN);
   nVal = _adcReadRaw() >> 1;

   ADCSRC &= ~(1 << SPN);
   pVal = _adcReadRaw() >> 1;

   pVal += nVal;          //失调误差补偿
   pVal -= (pVal >> 7); //增益误差补偿

   return pVal;
}
```

EEPROM 相关注意事项

LGT8FX8P 的 E2PCTL 控制与 LGT8FX8D 基本相同。LGT8FX8P 内部的 FLASH 为 32 位接口宽度。LGT8F328D 内部 FLASH 为 16 位接口宽度。LGT8FX8P 支持 8/16/32 位宽度数据读写，LGT8FX8D 只支持 8 位/16 位数据的读写。

当移植到 LGT8F328P 时，需要注意 E2PCTL 模块初始化稍微不同，下面是使用 LGT8F328P 的初始化过程：

LGT8F328P E2PCTL 模块初始化

```
#include "lgt8f328p_spec.h"


#define EEPROM_SZ_1KB  0x00
#define EEPROM_SZ_2KB  0x01
#define EEPROM_SZ_4KB  0x02
#define EEPROM_SZ_8KB  0x03

// 1KB EEPROM 控制器初始化
void e2pInit()
{
    u8 btmp = 0x4C | EEPROM_SZ_1KB;
    ECCR = 0x80;
    ECCR = btmp;
}
```

由于 LGT8FX8P 的 FLASH 为 32 位宽度，因此通过 E2PCTL 写 FLASH 空间只支持 32 位访问模式。而 LGT8FX8D 的 FLASH 位 16 位宽度，通过 E2PCTL 写 FLASH 空间只支持 16 位访问模式。这部分在代码移植时需要特别注意。LGT8FX8P 的 32 位访问模式，请参考 LGT8FX8P 编程手册。

LGT8FX8P 增加了连续写模式(SWM)，用于优化连续数据的读操作，减少中间页交换的频率。此部分为 LGT8FX8P 新增的功能， 具体细节请参考 LGT8FX8P 编程手册存储相关章节。

SPI 相关注意事项

LGT8F8F328P 与 LGT8F328D 的 SPI 控制器在工作模式上完全一致。但 LGT8F328P 的 SPI 数据收发器具有 4 字节 FIFO。因此在数据收发过程与 LGT8FX8D 有比较明显的不同。下面是 LGT8F328P 在 SPI 查询模式下的传输函数实现：

LGT8F328P SPI 模块收发函数

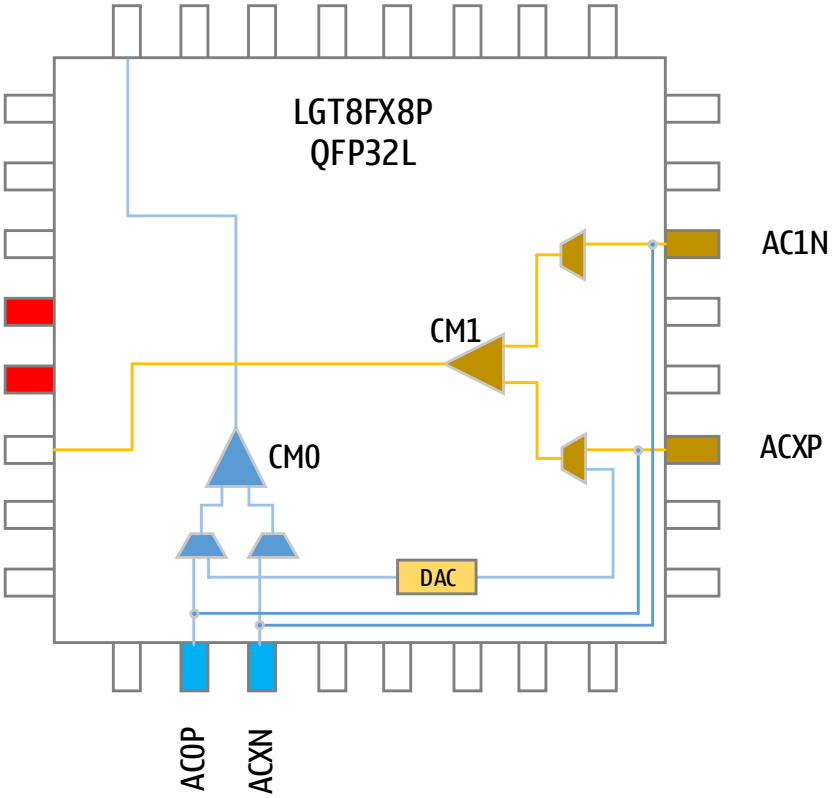
```
#include "lgt8f328p_spec.h"

// spi 字节收发
u8 spiTransferByte(u8 data)
{
    u8 rcvdat;
    SPDR = data;                          // 更新发送数据
    while(SPFR & _BV(RDEMPT));            // 等待数据传输完成
    rcvdat = SPDR;                        // 读取接收数据
    SPFR = _BV(RDEMPT) | _BV(WREMPT); //清 FIFO
    return rcvdat;
}

// spi 数据缓冲收发
void spiTransferBuffer(u8 *src, u8 length)
{
    u8 cnt = 0;
    u8 *ptx = src, *prx = src;
    SPFR |= _BV(RDEMPT) | _BV(WREMPT);
    while(cnt++ < 4) {    //预填充 FIFO
        SPDR = *ptx++;
        if(cnt == length) break;
     }

     while(length > 4) {
        if(!(SPFR & _BV(RDEMPT))) { *prx++ = SPDR; --length; } // 读数据
        if(!(SPFR & _BV(WRFULL))) { SPDR = *ptx++; } //填充 FIFO
     }

     while(length > 0) { // 读完剩余数据
         if(!(SPFR & _BV(RDEMPT))) { *prx++ = SPDR; --length; }
     }

     SPFR |= _BV(RDEMPT) | (_BV(WREMPT);
}
```

模拟比较器/8 位 DAC 相关注意事项

LGT8F328P 的比较器与 LGT8F328D 相比在输入结构上有很大的变化，具体改变如下：

1. LGT8FX8P 比较器输入结构独立，内部不再关联其他模块
2. LGT8FX8P 比较器输入可以选择为差分放大器模拟输出
3. LGT8FX8P 比较器输入可以选择为内部 8 位 DAC 输出
4. LGT8FX8P 比较器 0/1 的外部输入与 LGT8FX8D 兼容，但配置更加灵活
5. LGT8FX8P 的 8 位 DAC 为独立 IP，DAC 参考输入可配置(DACON)
6. LGT8FX8P 比较器支持数字滤波和比较器输出到端口(C0XR/C1XR)

下图为 LGT8FX8P 的内部比较器 0/1 输入/输出结构：



LGT8FX8P 比较器 0/1 的控制寄存器命名有所改变。控制寄存器分别为 C0SR/C1SR。控制比较器输出以及输出数字滤波的寄存器为新增寄存器 C0XR/C1XR。

由于 LGT8FX8P 比较器输入源相比 LGT8FX8D 有所增加，所以比较器输入选择寄存器也有所改变：

- 比较器 CM0 的正端选择由 C0SR 的 C0BG 位与 C0XR 寄存器的 C0PS0 位共同确定；
- 比较器 CM0 的负端选择由 ADCSRB 寄存器的 CME00/CME01 位确定；
- 比较器 CM1 的正端选择由 C1SR 的 C1BG 位与 C1XR 寄存器的 C1PS0 位共同确定；
- 比较器 CM1 的负端选择由 ADCSRB 寄存器的 CME10/CME11 位确定；

LGT8FX8P 比较器的具体配置，请参考 LGT8FX8P 编程手册相关章节。

功耗管理相关注意事项

LGT8FX8P 的低功耗管理兼容 LGT8FX8D。在 LGT8FX8D 下的功耗管理程序可以正常工作在 LGT8FX8P 上。但由于 LGT8FX8P 的端口没有默认上拉电阻， 因此需要软件在休眠前将可能处于浮空输入状态端口的内部上拉打开， 以避免浮空端口带来的漏电。

与 LGT8FX8D 相比， LGT8FX8P 在功耗管理方面有以下改进：

- 增加了端口电平变化唤醒（仅对于非 DPS2 休眠模式有效）
- 增加了定时器 Timer1 异步模式唤醒（仅对于非 DPS2 模式）
- 增加了更低的休眠模式 DPS2（仅支持 PD 组 IO 的电平变化唤醒）

DPS2 休眠模式：

LGT8FX8P 新增超低功耗模式 DPS2。DPS2 模式可在 3.3V 供电时，达到 1uA 左右的休眠功耗。DPS2 模式将会关闭内部 LDO 输出，因此所有的寄存器，以及 RAM 信息都将丢失。从 DPS2 模式唤醒首先会唤醒内部 LDO，然后执行一个完整的上电复位过程。

DPS2 模式下，所有的 IO 都有自动打开一个微弱的内部上拉电阻(约 20K~40K)，系统设计时需要注意！！！此上拉电阻仅在 DPS2 模式下有效，在其他休眠模式下无效。

DPS2 模式需要单独的寄存器控制：

- DPSR2 寄存器用于控制 DPS2 休眠模式
- IOCWK 寄存器用于配置 DPS2 的外部唤醒源

DPS2 模式的详细信息，请参考 LGT8FX8P 编程手册功耗管理章节。

以下为 LGT8FX8P 休眠管理编程代码示例：

LGT8F328P 功耗管理函数

```
#include "lgt8f328p_spec.h"

// 使能 PD7/6 作为 DPS2 的电平变化唤醒源
#define PMU_IOCD   (1 << 7) | (1 << 6)

// 休眠模式下选择关闭 SWD 接口以便开启 SWD/SWC 的内部上拉
#define PMU_SWDD   1

// 使能 DPS2 模式控制
#define PMU_DPS2   1

// 使能 LDO 待机模式控制
#define PMU_LDOPD  1

// 休眠前的系统时钟控制
// 如果使能了 LDO 待机模式，必须使能 CLKPR 控制
#define PMU_CLKPR  1
```

```
#define PMU_SLEEP_IDLE  0      // 待机模式
#define PMU_SLEEP_ANRM  1      // ADC 噪声抑制休眠模式
#define PMU_SLEEP_SAVE  2      // 省电模式
#define PMU_SLEEP_DPS1  3      // 掉电 DPS1 模式
#define PMU_SLEEP_DPS1  6      // 掉电 DPS0 模式
#define PMU_SLEEP_DPS1  7      // 掉电 DPS2 模式

// DPS2 寄存器控制
#define pmuIOCD(pid) IOCWK = pid
#define pmuEnableDPS2() DPS2R |= ( 1 << DPS2E)

// PMU 休眠管理函数
void pmuSleep(u8 mode)
{
    u8 CLKPR_reg = CLKPR;
    u8 LDOCR_reg = LDOCR;
    u8 MCUSR_reg = MCUSR | 0x7f;
    u8 SREG_reg = SREG;

    // global interrupt disable
    CLI();

#if PMU_SWDD == 1
    // disable SWD interface to release PE0/2
    MCUSR = 0xff;
    MCUSR = 0xff;
#endif

#if PMU_DPS2EN == 1
    pmuIOCD(PMU_IOCD);
    pmuEnableDPS2();
#endif

#if (PMU_CLKPR == 1) || (PMU_LDOPD == 1)
    // minimize system power before LDO power/down
    CLKPR = 0x80;
    CLKPR = 0x08;
#endif

    //NOP(); NOP(); NOP(); NOP();
#if PMU_LDOPD == 1
    LDOCR = 0x80;
    LDOCR = 0x02; // LDO power/down
```

```
#endif

    //NOP(); NOP(); NOP(); NOP();
    // bring system to sleep
    SMCR = ((mode & 0x7) << 1) | 0x1;
    SLEEP();

    NOP(); NOP(); // NOP(); NOP();

#if PMU_LDOPD == 1
    // restore LDO settings
    LDOCR = 0x80;
    LDOCR = LDOCR_reg;
#endif
    NOP(); NOP(); NOP(); NOP();

#if PMU_CLKPR == 1
    // restore system clock prescaler
    CLKPR = 0x80;
    CLKPR = CLKPR_reg;
#endif
    // restore SWD/SWC interface
    MCUSR = 0x80;
    MCUSR = MCUSR_reg;

    // restore interrupt settings
    SREG = SREG_reg;
}
```

版本历史

版本	作者	日期	版本日志
1.0.0	LGT	2017/5/26	The first edition