Application Report
ZHCA421 – March 2011

UCD3028 数字电源软件设计说明

Neil Li, Sundy Xu *China Telecom Application Team*

摘　要

UCD3028 数字电源功能丰富，可以完成两个独立功率支路的控制；软件配置灵活，可以实现多种功能包括多种故障的检测和处理。本文基于一款 1/8 砖模块产品对 UCD3028 的软件设计进行了详细说明，内容包括外界模拟信号的采集与转换，副边电流的采集与相关设置，输入电压的采集与相关设置，输出电压的采集与相关设置，输出电压预偏置(Prebias)功能的设计与实现，中央中断模块和中断功能实现，前馈功能的设计与实现，环路补偿器（CLA)的初始化，DPWM（数字脉宽调制）的初始化，通用寄存器的初始化，时钟初始化及状态机介绍，芯片内部存储设备介绍，最后给出了整个 UCD3028 软件的流程图。

目　录

图

1 模拟信号采集与转换

外界模拟量如输入电压、输出电压、输出电流和外界温度等全部是通过 UCD3028 的 AD00~AD08 引脚采集到芯片内部。在数据处理过程中使用了滑动平均滤波算法，详见下文介绍。

1.1 滑动平均滤波算法

滑动平均滤波算法的原理是，把连续采集的 N 个采样值看成 1 个队列，队列的长度固定为 N，每次采集到的 1 个新数据放入队尾，并扔掉原来队首的一个数据，然后把队列中的 N 个数据进行平均运算，即可得到新的滤波结果。

1.2 软件具体实现

1、U3028 软件实现中将采样样本值设置为 512，即 N=512。

2、U3028 软件实现中，在得到新采样样本后，扔掉的不是队首的第一个数据，而是之前的平均值。即，新平均值= (新采集数据样本+前 N 次数据总和-前 N 次数据平均值) /N。

3、如下是副边电流采样的实现细节。其中 adc_overall.average.isec 保存了最新的 512 次采样样本的总和。

```
adc_overall.raw.isec = AdcRegs.ADCRESULT2.all; //data acquired from AD02
adc_overall.average.isec = adc_overall.raw.isec + adc_overall.average.isec – (adc_overall.average.isec >> 9);
```

2 副边电流采样与相关设置

通过检测主功率支路一段走线的电压并放大实现了对副边电流值的检测。依靠该检测值，UCD3028 软件实现了平均过流保护和快速过流保护。同时，该检测值还上传到 GUI 软件中反算出实际值并显示。

2.1 硬件电路设计

如图 1 所示，R15 为主功率支路回路串接的电阻（实际电路设计中采用铜皮替代），U5 及其外围电路完成对 R15 电阻电压的采集、放大，放大倍数为 331。经放大后的电压通过 AD02 传输到 UCD3028 芯片中。由此可得出 AD02 引脚的电压为：V_{AD02}= (Io×R15) ×331

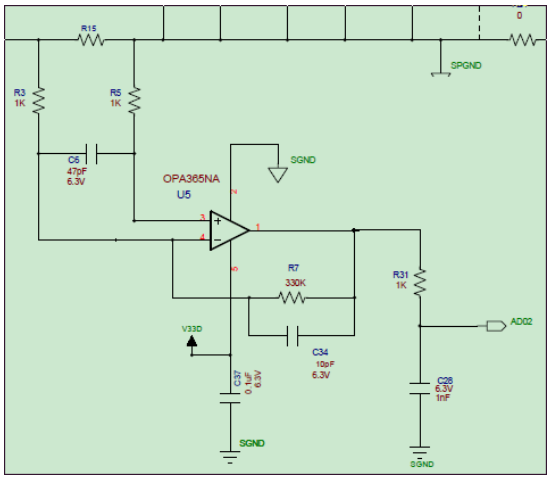

图 1：副边电流采样电路

2.2 平均过流保护功能的实现

在 1/8 砖项目中，检测电阻 R15 的阻值为 0.15 毫欧。

欲设置平均过流保护点为 23A，则可得出：(23×0.00012) ×331×(4096/2.5)=1497。其中 4096/2.5 是完成采样数据的量化，即采样的模拟信号电压为 2.5V 时，量化为 4096。

如下代码完成对过流保护标志位的设置：

```
#define OVER_CURRENT_FAULT (adc_overall.average.isec > (1497 << 9))
```

2.3 快速过流保护功能的实现

UCD3028 芯片内部设计有模拟比较器，可以快速响应 AD 引脚电压，如图 2 所示。

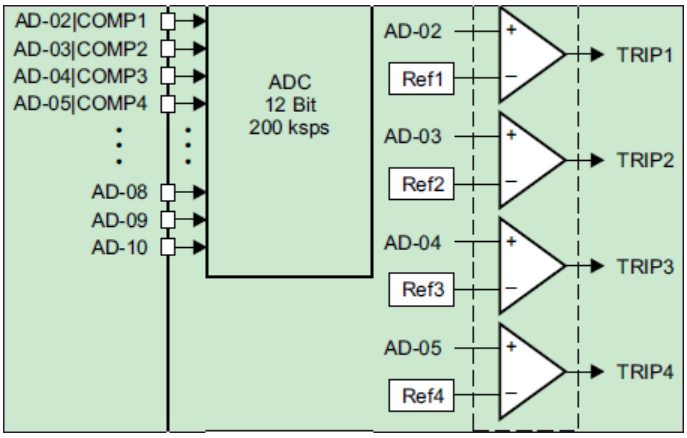

图 2：UCD3028 通用 ADC

该模拟比较器参考电压的设置在 COMPCTL 寄存器中完成。其第 0 位至第 5 位装载参考电压的值，分辨率为 31.25mv。

Bits 5-0: COMP_ADJ_A – Reference Select for Analog Comparator A
000000 = Comparator Reference of 31.250 mV (Default)
000001 = Comparator Reference of 62.5 mV
……..
111111 = Comparator Reference of 2.0 V

图 3：COMPCTL 寄存器

欲设置快速过流保护点为 30A，则有：(30×0.00012) ×331=1.192V。

而 1.192V/0.03125V=38。因此有如下代码：

```
MacRegs.COMPCTRL.bit.COMP_ADJ_A =38;
```

2.4 GUI 软件显示值的转换与实现

1、显示值算法如下：Io=Io×R15×331×(4096/2.5) ×(0.25×K×1/216) ×0.25，由此可计算出 K=15711，因此有如下代码：

```
#define IOUT_ADC_TO_LITERAL_SCALER_NOMINAL (15711)
```

2、函数的具体实现

在函数 pmbus_read_iout_handler()中调用函数 unsigned_short_q_multiply()，并完成数据格式的转换，如下所示：

```
temporary = unsigned_short_q_multiply(iout >> 2, K);
pmbus_number_of_bytes = 2;
pmbus_buffer[1] = ((temporary & 0x700) >> 8) + 0xF0;
pmbus_buffer[0] = temporary & 0xff;
```

而函数 unsigned_short_q_multiply()完成两个输入值的乘积计算并对结果进行转换，如下所示：

```
product = ((unsigned int) multiplier) * ((unsigned int) multiplicand);
return (signed short)((product + 0x8000) >> 16);   //8000 is for rounding
```

3、PMBus 通信：Literal 数据格式

输出电流上传到 GUI 显示采用 PMBus 通信方式，数据格式采用 Literal。该数据格式如下图 4 所示。

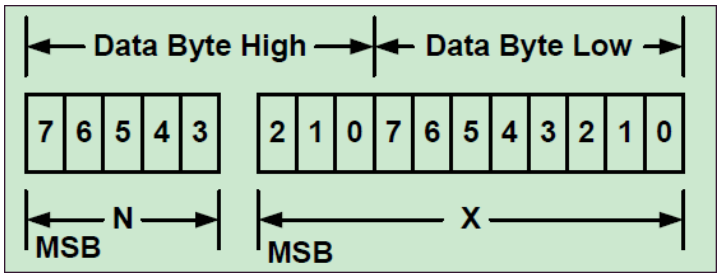

图 4：Literal 数据格式

高 5 位，是数据的缩放系数，采用补码存放；低 11 位存放数据。欲传输的数据 $Y=X*2^N$

4、举例如下

设有输出电流为 10A，则 temporary= 10×0.00012×331×(4096/2.5) ×(0.25×15711×1/216) ×0.25=39，其二进制为 0000 0000 0010 0111。

pmbus_buffer[1]=(0000 0000 0010 0111&0x700) >> 8) + 0xF0=1111 0000

pmbus_buffer[0]=0010 0111

因此，待传输数据为 1111 0000 0010 0111。即，N=-2，X=39。

最后有，$Y= X\times 2^{N}=39\times 2^{-2} =39\times 0.25$.

3 输入电压的采集与相关设置

3.1 硬件电路设计

图 5 为输入电压线性转换电路。通过该电路的转换，V_A 点的电压会与输入电压成线性比例关系。将 UCD3028 的 AD01 引脚连接到 V_A 点并采集到 V_A 点电压，再经过软件计算即可以得出输入电压的值。

对该电路的分析见下文。

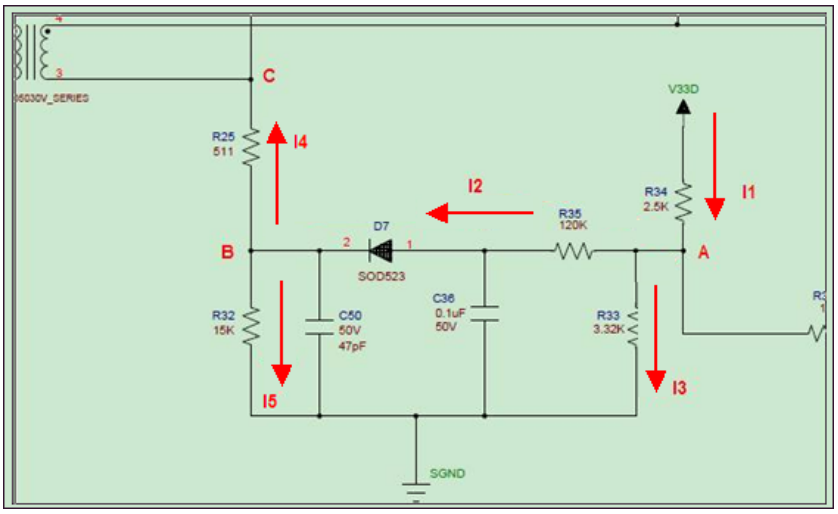

图 5：输入电压转换电路

将该电路主干支路的电流分别标识为 I1、I2、I3、I4、I5，表达式分别见式(1)至(5)：

$$I_1 = \frac{V33D - V_A}{R_{34}} \quad \text{.....(1)}$$

$$I_2 = \frac{V_A - 0.7 - V_B}{R_{35}} \quad \text{.....(2)}$$

$$I_3 = \frac{V_A}{R_{33}} \quad \text{.....(3)}$$

$$I_4 = \frac{V_B - V_C}{R_{25}} \quad \text{……..……..(4)}$$

$$I_5 = \frac{V_B}{R_{32}} \quad \text{…….…..(5)}$$

根据基尔霍夫电流定理，流入一个节点的总电流与流出一个节点的总电流相等，故有式(6)和式(7)：

$$I_1 = I_2 + I_3 \text{…….……..(6)}$$

$$I_2 = I_4 + I_5 \text{…….……..(7)}$$

而 Vc 点是反激变压器的异铭端，该点电压与输入电压的关系如式(8)所示：

$$V_C = 12 - Vin \quad \text{……..……..(8)}$$

经过计算，最后有： $V_A = 2.14 - 0.023Vin$ ……....(9)

通过多次测试，该公式最终修正为： $V_A = 2.18 - 0.02Vin$ 。

3.2 欠压保护功能设计

设有欠压保护点为 28V，恢复点 30V，则有 V_A 分别对应 1.62V 和 1.58V，量化之后分别是 2654 和 2589，因此有如下代码：

```
#define VIN_ON                  (2589) //30v
#define VIN_OFF                 (2654) //28v
#define SUFFICIENT_INPUT_VOLTAGE
(adc_overall.average.vin < (((int)(VIN_ON)) << 9))
//#define INSUFFICIENT_INPUT_VOLTAGE
(adc_overall.average.vin > (((int)(VIN_OFF)) << 9))
```

说明，在实际应用中，输入电压的采集次数不能设置为 512 次，因为此时采样转换的时间过长，电源系统将不能及时响应输入电压的变化，因此欠压保护功能会面临失效。实际调试发现，设置 8 次采样可以保证采集到的输入电压的精度和欠压保护功能的有效性。

3.3 硬件电路设计 GUI 软件显示值的转换和实现

下面函数完成了对采集到的输入电压的显示。

```
Uint8 pmbus_read_vin(void)
{
 #if (DCDC_PAGE_NUMS >= 1)
  if (page < DCDC_PAGE_NUMS)
    {pmbus_read_vin_handler((Uint32)(0xDF3 - (adc_overall.average.vin >> 9))); }
  else
```

```
        #endif
          {pmbus_read_two_byte_handler(pfc_reading.vin_reading; }
        return PMBUS_SUCCESS;
}
```

上述函数实现如下功能：调用函数 pmbus_read_vin_handler()，同时传递的参数是 $\left[3571-\left(\frac{4096}{2.5}\right)\times\left(2.18-0.02Vin\right)\right]$。下面代码是函数 pmbus_read_vin_handler()的具体实现。

```
int pmbus_read_vin_handler(Uint32 vin)
{
            Uint32 temporary;
            temporary = (vin * VIN_ADC_TO_LITERAL_SCALER)>> 10 ;
            pmbus_number_of_bytes = 2;
            pmbus_buffer[1] = ((temporary & 0x700) >> 8) + 0xE0;
            pmbus_buffer[0] = temporary & 0xff;
            return 0;
 }
#define VIN_ADC_TO_LITERAL_SCALER  (500)
```

上述代码实现如下计算：

$$\left[3571-\left(\frac{4096}{2.5}\right)\times\left(2.18-0.02Vin\right)\right]\times 500\times\frac{1}{2^{14}}\times\frac{1}{2^{4}}=Vin \quad \text{.....(10)}$$

即，经过上述转换，最终显示的值即为实际输入电压。

4 输出电压的采集与相关设置

4.1 输出电压的软件设置

固定外部分压电阻后，输出电压的调整可以通过软件设置，即写入预设值到 EADCDAC 寄存器中。该寄存器有 10 位，最大值对应的输出电压为 1.6V，因此分辨率为 1.5625mV。

设有输出电压预设值为 12V，外部分压电阻的比例为 10:1，因此参考电压应设置为 1.1V，需要进行如下赋值：Dpwm1Regs.EADCDAC.bit.DAC_VALUE=710;

4.2 慢速过压保护

慢速过压保护是基于 ADC 多次采集到的外部输出电压而做出动作，因此响应相对较慢但精度相对较高，一般需要设置较低的过压点。

设有过压保护点为 14.5V，则到达 AD 引脚的电压为 1.32V，量化之后的值为 0x86F。代码实现如下：

```
#define OVER_VOLTAGE_FAULT      (adc_overall.raw.VO > 0x86F)
#define OVER_VOLTAGE_FAULT_NOT  (adc_overall.raw.VO < 0x860)
```

系统触发慢速过压后，打嗝四次，如果故障不消除，系统锁死。

4.3 快速过压保护

UCD3028 芯片中设计有快速模拟比较器，可以快速响应 AD 引脚电压的变化。

欲设置快速过压保护点为 15.5V，则有：15.5/11/0.03125=45。

```
MacRegs.COMPCTRL.bit.COMP_ADJ_B = 45;
```

5 输出电压预偏置功能实现

5.1 UCD3028 系统预偏置实现概述

系统关机后在输出电压还未下降到 0V 时再次开机，如果没有预偏置功能，则在同步整流拓扑中会有很大的倒灌电流，该倒灌电流极有可能损坏整流管。

为防止该倒灌电流的产生，可以在有残压开机时，直接将占空比展开到当前适合值，同时将误差放大器的参考电压直接设置到当前适合值。

上述两个适合值的计算方式如下：

1）合适的占空比，需要根据当前输出电压和当前输入电压计算得出；

2）合适的参考电压，需要根据输出电压和分压电阻计算得出。

5.2 预偏置功能软件实现及相关系数计算

软件系统中将 Prebias 设计为一个独立的状态机，包含了当前占空比计算、当前参考电压计算、数字补偿器（CLA）赋值并启动等三个主要部分。

1、当前占空比计算

在 1/8 砖项目中，采用传统全桥拓扑，变压器匝比为 3:1:1，因此半周期占空比为：$D = \frac{3 \times Vo}{2 \times Vi}$。

下面代码给出当前占空比的具体实现。

```
vout = adc_overall.raw.VO;
vin = adc_overall.average.vin >>9;
vin = 0xfff - vin;
tempor = (vin * 2.7);   //2.7 vin has same scaling as vout
duty_prebias = (vout << 15) / tempor;
if(duty_prebias > 0x3d80)
{duty_prebias = 0x3d80; }
```

上述代码中的 2.7 用 K 表示，则有 tempor=vin×K。因此，占空比计算如下：

$$D_{prebias}=\frac{\left(\frac{Vo}{11}\right)\times\left(\frac{4096}{2.5}\right)}{\left[4096-\left(\frac{4096}{2.5}\right)\times(2.18-0.02Vin)\right]\times K}=\frac{\left(\frac{Vo}{11}\right)\times\left(\frac{4096}{2.5}\right)\times 382\times\frac{1}{2^{14}}}{Vin\times K}=D\times\left(\frac{2\times 382}{330\times K}\right)$$

当有 $\frac{2\times 382}{330\times K}=1$ 时，Dprebias=D，此时要求 K=2.3。

上述理论计算值为依据，K 的取值还需要在实际应用中进行微调以确定最终的值。实际应用中，略微增大 K 值可以使残压在开机时有轻微跌落，但可以有效降低应力。在 1/8 砖项目中，该值选最终选取 3.3。

2、当前参考电压计算

由之前的介绍已知，EADCDAC 寄存器中的值与预设输出电压的值的关系如下：$Vref=\left(\frac{Vo}{11}\right)\times\left(\frac{2^{10}}{1.6}\right)$

下述代码实现了有残压开机时，参考电压应合理赋予的值。

```
dac_prebias = (vout * 806);
timer_interrupt_temporary_1 =dac_prebias;
Dpwm1Regs.EADCDAC.bit.DAC_VALUE = (timer_interrupt_temporary_1 >> 11);
```

上述代码中 806 的计算过程如下：

$$Vout\times K\times\frac{1}{2^{11}}=\frac{Vo}{11}\times\frac{4096}{2.5}\times K\times\frac{1}{2^{11}}=\frac{Vo}{11}\times\frac{2}{2.5}\times K=Vref$$

因此可以得出 K=800。

1/8 砖项目中，该值选取为 806。

6 中央中断模块与中断实现

6.1 中央中断模块简述

为满足 Firmware 的时序要求，中央中断模块（Central Interrupt Module，CIM）设计有 32 个中断请求。由于 ARM 处理器自身只支持两级中断，即快中断（Fast Interrupt Request，FIQ）和标准中断（Normal Interrupt Request，IRQ），于是 CIM 通过 FIQ/IRQ 向量寄存器提供向量表，在向量表中对各个中断做索引，以此实现了中断的扩展。

每个中断在向量表中的数字索引表明了其优先级，0 号中断是最低优先级的中断，31 号中断则是最高优先级的中断。CIM 对中断请求的执行严格遵循优先级顺序，在发出中断请求之后，在执行该中断之前，需要一直保持该中断的状态为高。

6.2 中断在 UCD3028 软件中的具体实现

UCD3028 软件中，使用函数 write_reqmask()完成了对中断的使用声明，即只有对中断向量表中的某个中断索引号赋值为 1 才能使用该中断，默认为标准中断。在下面语句中，对 10^{th} 中断、17^{th} 中断和 28^{th} 中断进行了声明：

write_reqmask(0x10020400);

其中，0x10020400 换算为二进制为 0001 0000 0000 0010 0000 0100 0000 0000，即第 10、17 和 28 位分别被赋值为 1，则这三个索引号对应的中断被声明可以使用。

下面语句完成了对快中断的声明使用。同样是在中断向量表中对欲声明为快中断的索引号赋值为 1。

write_firqpr(0x10000400);

其中，0x10000400 换算为二进制为 0001 0000 0000 0000 0000 0100 0000 0000，即第 10 位和 28 位分别被赋值为 1，这两个索引号对应的中断被声明为快中断使用。

需要注意的是，某个中断在声明为快中断前需要先进行可使用的声明，即函数 write_firqpr()需要与函数 write_reqmask()配合使用。

7 前馈功能的设计与实现

7.1 前馈功能设计实现概述

UCD3028 系统中前馈功能的实现思路如下：采集原边剧烈上升的电压到模拟 ADC-->触发快速中断而关闭主功率支路驱动-->输出电压不再上升，并被负载逐渐拉低-->UCD3028 重新输出驱动，系统重新正常工作（在输出电压重新建立过程中使用 Prebias 起机）。该前馈功能设计的特点为：

1)使用快速模拟比较器触发快速中断，反应速度快；

2）触发快速中断后，驱动关闭，系统时时检测输出电压。直到输出电压降低到正常电压值后，系统才重新启动；

3）重新启动后，系统使用 Prebias 功能实现输出电压的重新上升，无倒灌风险。

下面详细阐述前馈功能的设计。

7.2 带有快速触发信号的输入电压采集电路

输入电压采集电路如图 6 所示，RC 电路连接到变压器副边中心抽头处，该处电压与输入电压成正比，该处电压表达式为：$Vin \times D \times \frac{1}{N}$（N 为变压器匝比）。

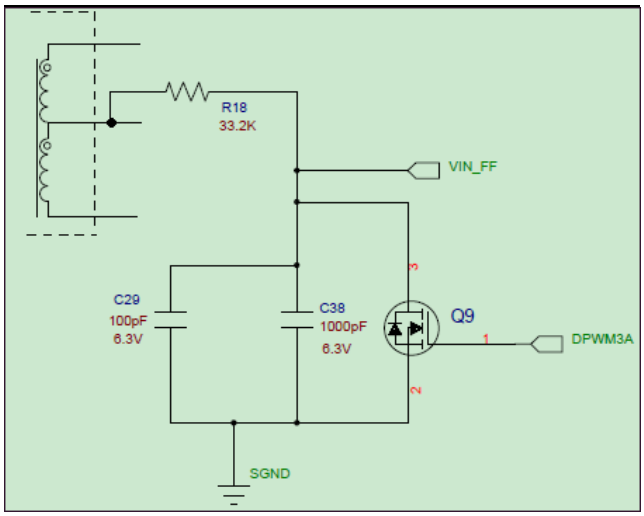

图 6：带触发信号的输入电压采集电路

当变压器处于传递能量阶段时，Q9 设置为关断，输入电压的剧烈上升会造成电容电压（VIN_FF 网络电压）的快速上升。而 VIN_FF 网络连接到了 UCD3028 的 ADC 接口 AD-04（参见图 2），因此输入电压的剧烈上升可以触发模拟比较器的翻转，从而进入快中断。

在全桥进入副边续流阶段时，Q9 设置为导通，电容放电，为下一次充电做好准备。

该电路的详细分析见文献《fast vin sensing and control algorithm for surge protection》的 P3~P5。其中，一个重要的定量分析结论为：$V_{DS\_Q9} = \frac{Vo}{RC} \times \frac{Ts}{2}$

由该式可知，当系统处于稳态时，输入电压的缓慢变化会引起环路对占空比的调节，从而保持输出电压不变，V_{DS_Q9} 亦保持不变。而当输入电压剧烈上升时，环路受带宽限制无法立即完成调整，输出电压会有上冲，此时 V_{DS_Q9} 亦有上冲，当上冲电压达到模拟比较的阈值电压后便会触发快中断。

7.3 模拟比较器触发的快速中断子程序

当进入模拟比较器快速中断（10th FIQ）后，软件首先关闭所有驱动输出，以下语句完成该动作：

```
Dpwm1Regs.DPWMCTRL1.bit.GPIO_A_ENA = 1; //shut down
Dpwm1Regs.DPWMCTRL1.bit.GPIO_B_ENA = 1; //shut down
Dpwm2Regs.DPWMCTRL1.bit.GPIO_A_ENA = 1; //shut down
Dpwm2Regs.DPWMCTRL1.bit.GPIO_B_ENA = 1; //shut down
Dpwm4Regs.DPWMCTRL1.bit.GPIO_A_ENA = 1; //shut down
```

由于快速过流（输出电压信号连接到了 AD-02）和快速过压（副边电流信号接入到了 AD-03）同样会进入该 FIQ，因此快速中断子程序对入口条件进行判别并分别进行相关处理。随后，软件依次完成如下动作：

1．重新设置快速过压和快速过流的阈值电压，而修改该值的原因见下文分析；

```
MacRegs.COMPCTRL.bit.COMP_ADJ_B = 32; //lower the OV threshold
MacRegs.COMPCTRL.bit.COMP_ADJ_A = (OC_SETTING -7);
```

2．重新初始化 12 位通用 ADC（只打开采集输入电压和输出电压的通道）；

```
AdcRegs.ADCCTRL1.half.HALF0 =ADCCTRL1_HALF0_SINGLE_SWEEP +(1 << 4) +1;
AdcRegs.ADCCTRL1.bit.INT_ENA = 1; //enable interrupt at this level to avoid clear on read problems
AdcRegs.ADCSEQSEL1.half.HALF0 = 3 + (1 << 5); //5 bits per channel selector
```

3．失效模拟比较器快速中断（10th FIQ），使能 DPWM 的周期快速中断（28th FIQ）；

```
MacRegs.COMPCTRL.bit.COMP_INT_ENA =0;
Dpwm1Regs.DPWMINT.bit.PRD_INT_ENA = 1;
```

4．对状态传递信号 Vin_state 赋值；

```
vin_state =1;
```

5．在退出该子程序时清空中断标示位；

```
fir_flag_clear = MacRegs.COMPREAD.all; //clear interrupt flag
```

7.4 DPWM 快速中断子程序

当系统退出模拟比较器快速中断子程序后立即开始一个新的开关周期，但由于使能了 DPWM 的每周期快中断，系统会进入 DPWM 快中断子程序。该程序中首先调用函数 vff_handler()，在该函数中根据 vin_state 的值分别进行处理。详细分析见下：

第一次进入该中断子程序，此时 vin_state 的值为 1，程序首先进行条件判断，判断是否当前有快速过压或快速过流的异常。当二者皆不存在时，vin_state 的状态被赋值为 2，并退出该函数；否则，直接退出该函数。退出函数后，清空快中断标志位，随后退出中断子程序。

```
if(vin_state ==1)
 {
  if((MacRegs.COMPREAD.bit.COMP_B == 0) && (MacRegs.COMPREAD.bit.COMP_A ==0))
  { vin_state =2;// delay one period
    AdcRegs.ADCCTRL1.bit.SW_START = 1 ; //start a conversion
    AdcRegs.ADCCTRL1.bit.SW_START = 0 ; //clear trigger, so we don't restart
  }
```

```
}
```

当退出该中断后，系统开始一个新的周期但此时会再次触发 DPWM 快中断，并进入中断子程序。如果 vin_state 的值依然为 1，说明输出电压或输出电流还存在异常，系统将在该中断子程序中不做任何动作，而是清空标示位后退出，循环往复，直到输出电压和输出电流都正常，vin_state 的状态被赋值为 2，程序开始进入一个新的处理阶段。

vin_state 被赋值为 2 后，在中断子程序中 vin_state 被重新赋值为 3，随后退出。

vin_state 被赋值为 3 后，在中断子程序中采集当前输入电压和输出电压，完成当前占空比的计算，同时将占空比数据赋值给环路。在将 vin_state 被赋值为 4 并清空标示位后退出中断子程序。

vin_state 被赋值为 4 后，在中断子程序中依次完成如下动作：

1）根据当前输出电压对误差放大器的参考电压进行重新写入；

```
timer_interrupt_temporary_1 = (vout * 806);
Dpwm1Regs.EADCDAC.bit.DAC_VALUE = (timer_interrupt_temporary_1 >> EADC_DAC_SHIFT);
```

2）打开补偿器，开始启动环路采样和调节，同时设置各路驱动为 DPWM 模式；

```
Filter1Regs.FLTRCTRL.bit.CLA_EN = 1;
Dpwm2Regs.DPWMCTRL1.bit.GPIO_A_ENA = 0;
Dpwm1Regs.DPWMCTRL1.bit.GPIO_A_ENA = 0;
Dpwm2Regs.DPWMCTRL1.bit.GPIO_B_ENA = 0;
Dpwm1Regs.DPWMCTRL1.bit.GPIO_B_ENA = 0;
Dpwm4Regs.DPWMCTRL1.bit.GPIO_A_ENA = 0;
Filter1Regs.FLTRCTRL.bit.USE_CPU_SAMPLE =0;
```

3）将快速过压点和快速过流点重新设置为正常值。

```
MacRegs.COMPCTRL.bit.COMP_ADJ_B =41;
MacRegs.COMPCTRL.bit.COMP_ADJ_A = OC_SETTING;
```

4）赋值 vin_state 为 5。

```
vin_state =5;
```

在系统退出中断子程序并开始一个新的周期时，系统再次触发中断并进入中断子程序。此时 vin_state 的值为 5，函数 vff_handler()首先进行是否存在快速过压或过流故障的判断，只有二者皆不存在时，执行如下操作。否则，退出中断子程序。两种故障皆不存在时执行的操作为：

1）使能模拟比较器快速中断；

```
MacRegs.COMPCTRL.bit.COMP_INT_ENA =1;
```

2）失效 DPWM 快速中断；

```
Dpwm1Regs.DPWMINT.bit.PRD_INT_ENA = 0;//fast ADC FIQ begin,fast period FIQ close
```

3) vin_stated 值修改为 6；

vin_state =6;

在执行完上述操作并清空中断标示位后退出中断子程序，此时系统开始正常工作，输出电压从当前值开始重新软起。

7.5 软件设计框图

从输入电压的剧烈上升而第一次触发快中断开始，到系统再次正常输出的整个软件操作流程如图 7 所示。其中，第一个条件执行操作决定了系统关闭驱动的时间，而该段时间避开了输入电压急剧上升及稍后的震荡，确保了系统不会出现掉电复位。

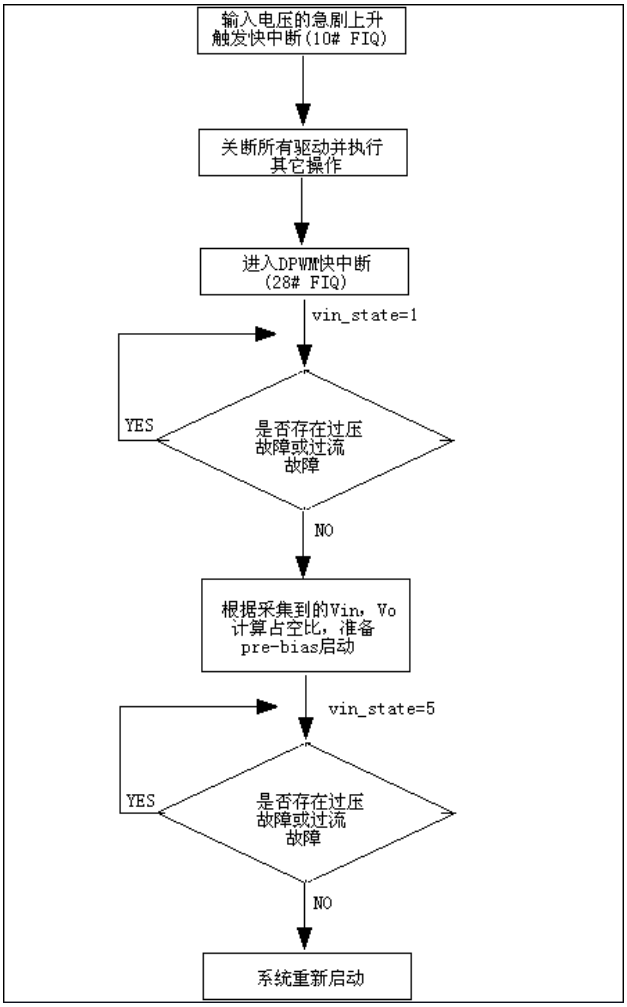

图 7：软件操作流程

8 CLA 的初始化及设置

8.1 CLA 简述

数字补偿器（CLA，compensator or control law accelerator）完成对量化之后的输出电压误差的运算补偿，最终计算出相应的占空比信息传递给 DPWM Engine。

如下图 8 所示，数字补偿器包含了输入误差信号的缩放，非线性增益、数字补偿、输出信号的缩放等。

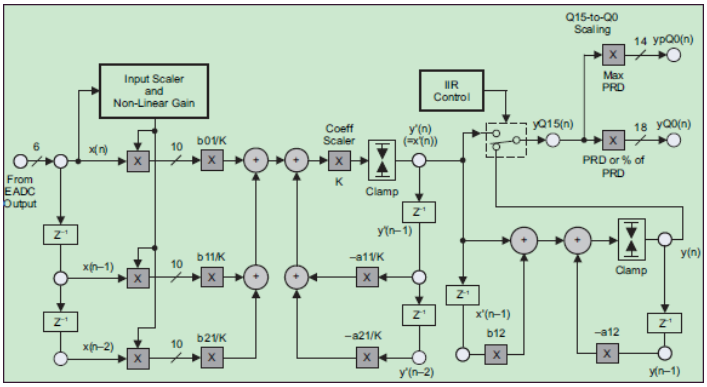

图 8：CLA 硬件框图

8.2 非线性增益的应用

非线性增益（nonlinear）对输出电压误差进行了划分，并对每个划分范围的误差提供了不同的增益值。这将有益于减小系统的输出误差，尤其是负载处于大动态时。

如图 9 所示。（测试条件：50% load step, 20A design, Fsw= 500KHz）

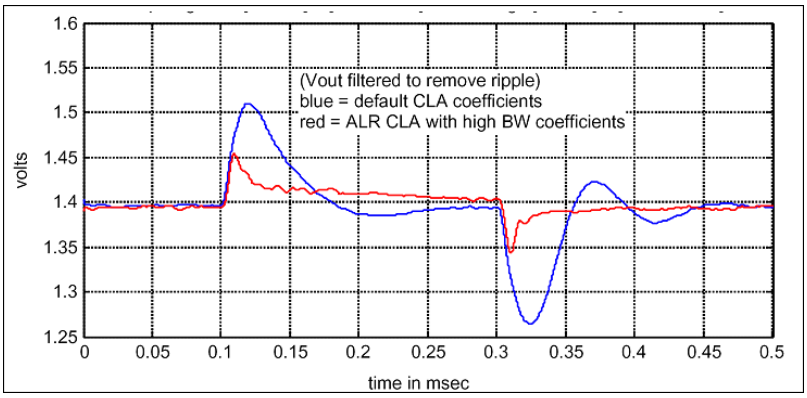

图 9：使用非线性增益的实测动态波形

寄存器 Filter Nonlinear Response Register 2(FLTRNLR2)提供了对误差范围的设定。共包含有四个值，每个值由 6 位组成，以补码方式存放，范围是-32~32。

Bit Number	29:24	21:16	13:8	5:0
Bit Name	LIMIT3	LIMIT2	LIMIT1	LIMIT0
Access	R/W	R/W	R/W	R/W
Default	00_0000	00_0000	00_0000	00_0000

图 10：FLTRNLR2 寄存器

寄存器 Filter Nonlinear Response Register 1(FLTRNLR1)则提供了每个范围对应的增益值，该增益值的范围是 0.25~15.75，步进值是 0.25.

Bit Number	31:30	29:24	23:18
Bit Name	AFE_GAIN	NOM_GAIN_MULT	POS_MID_GAIN_MULT
Access	R/W	R/W	R/W
Default	11 – Bank A 01 – Bank B	00_0100 – Bank A 01_0000 – Bank B	00_0100 – Bank A 01_0000 – Bank B

Bit Number	17:12	11:6	5:0
Bit Name	POS_LRG_GAIN_MULT	NEG_MID_GAIN_MULT	NEG_LRG_GAIN_MULT
Access	R/W	R/W	R/W
Default	00_0100 – Bank A 01_0000 – Bank B	00_0100 – Bank A 01_0000 – Bank B	00_0100 – Bank A 01_0000 – Bank B

图 11：FLTRNLR1 寄存器

下图给出了误差在不同范围内时的增益值。

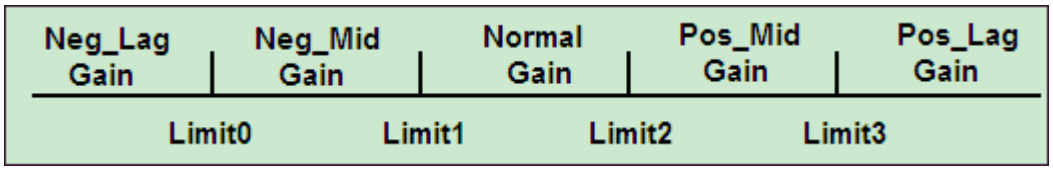

图 12：线性增益区间及对应值

8.3 UCD3028 软件实现

在 CLA 初始化函数 init_cla1_fast_filter()中，通过调用函数 init_cla_download()和 load_filter_from_flash()完成了对 CLA 线性增益、非线性增益和补偿器系数的装载，其中缺省值设置见下文代码。CLA 初始化函数中还包含了最大占空比和最小占空的设置。

1、缺省值的设置

下面代码是对 CLA 的线性增益、非线性增益和补偿器系数进行了缺省值的设置。对应的系数依次是 B01、B11、B21、SCALER、-A11、-A21、 B12、-A12、AFE、Normal Gain、Pos_Mid Gain、Pos_Lag Gain、Neg_Mid Gain、Neg_Lag Gain、Limit3、Limit2、Limit1、Limit0。

```
#define  DEFAULT_LOOP_0_ACTIVE_BANK_CLA
{0x038E, 0x09AA, 0x02CF, 3, 0x01AF, 0x0F51, 0, 0, 3, 2, 3, 4, 3, 4, 0x0C, 0x07,   0xF9, 0xF4 }
```

#define DEFAULT_LOOP_0_INACTIVE_BANK_CLA

{0x00B8, 0x0EC5, 0x0086, 1, 0x0570, 0x0E90, 0, 0, 1, 4, 4, 4, 4, 4, 0x00, 0x00, 0x00, 0x00 }

如果调用 DEFAULT_LOOP_0_ACTIVE_BANK_CLA，则使用了非线性增益，且对误差范围进行了划分，如下图所示。例如，当误差在 7~12 范围内，增益是 3×0.25=0.75。由于线性增益设置为 8，因此总增益为 8×0.75=6。

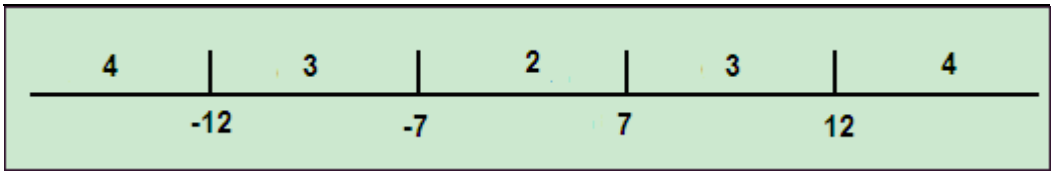

图13：实例中的线性增益区间划分

而如果调用 DEFAULT_LOOP_0_INACTIVE_BANK_CLA，则将不使用非线性增益，因为此时误差的范围全部设置为 0。

2、最大占空比与最小占空比的设置

1）FLTRCLAMP 寄存器介绍

该寄存器是一个 32 位寄存器，完成了对最大和最小占空比的设置。其第 0 位至第 15 位存储的是最小占空比的值，包含最高位的符号位。第 16 位至第 31 位为最大占空比的值，包含最高位的符号位。

Bit Number	31:16	15:0
Bit Name	CLAMPH	CLAMPL
Access	R/W	R/W
Default	0111_1111_1111_1111	0000_0000_0000_0000

图 14：FLTRCLAMP 寄存器

2 ）UCD3028 软件的具体实现

(1) 如下代码实现了最大占空比为 48%的设置：

Filter1Regs.FLTRCLAMP.bit.CLAMPH = 0x3D70;

计算如下：0x3D70 转换为 10 进制为 15728；包含符号位的 16 位寄存器最大值为 2^{15}=32768，因此，最大占空比为 15728/32768=48.00%。

(2) 选择 CLAMPH 和 CLAMPL 的 Page 由如下代码实现：

Filter1Regs.FLTRCTRL.bit.CLAMP_PG_CONTROL = 1;

即选择 page A 激活。

(3)UCD3028 软件中的页（Page）

UCD3028 中的所有补偿器系数和绝大多数补偿器系数控制寄存器数据都复制两份，分别存放在页 A 和页 B 中。一个操作原则是，当前使用（读取）的页是不能够被写入数据的。如下代码完成对页的选择操作。

```
Filter1Regs.FLTRCTRL.bit.CLAMP_PG_CONTROL = 1; //激活页 B，准备写入数据到页 A 中。
```

9 DPWM 的初始化及设置

9.1 传统全桥原副边驱动简述

传统全桥原副边驱动波形（副边采用同步整流方式）的示意图如下所示。

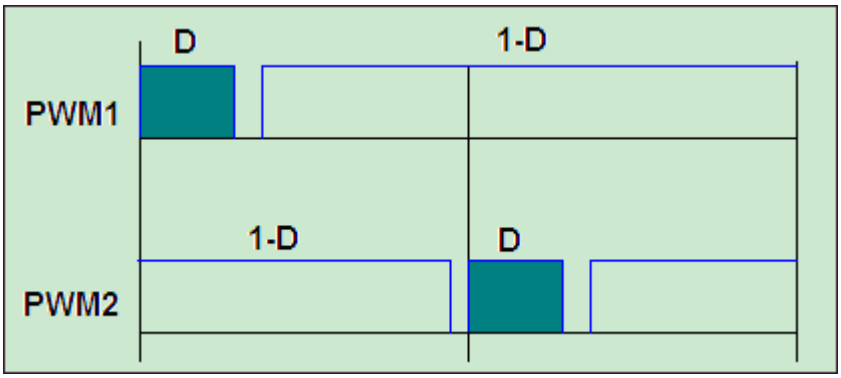

图 15：传统全桥驱动波形

驱动波形的特点如下：

1）可以将四个驱动分为两组，每组的两个驱动分别是 D 和(1-D)的关系；

2）两组驱动的相位相差 180° 。

3）D 的最大值限定在 50%以下；

9.2 DPWM1 的初始化设置

1、开关频率设置

UCD3028 软件中，对开关频率的设置是通过写入值到寄存器 DPWM Period Register（DPWMPRD，该寄存器包含有 14 位）中完成的。例如，写入 2500 到该寄存器，则系统的开关频率为 250000KHz/2500=100KHz。

2、死区设置

传统全桥驱动的死区主要指原副边驱动之间的死区。下面代码完成了对 Event3 和 Event4 的设置，其中变量 DELAY1 和 DELAY2 可以通过 GUI 设置。

```
Dpwm1Regs.DPWMPRD.all = period;//完成周期的设置。
Dpwm1Regs.DPWMEV1.all = 0;
Dpwm1Regs.DPWMEV2.all = 0;
Dpwm1Regs.DPWMEV3.all = DELAY_1 * HIGH_RESOLUTION_MULTIPLIER;
```

Dpwm1Regs.DPWMEV4.all = (period << 4) - DELAY_2 * HIGH_RESOLUTION_MULTIPLIER;

系统运行之后，Event2 会根据占空比的大小进行调节，Event3 则在 Event2 和死区的基础上进行调节。Event1 和 Event4 位置固定。

在对死区设置的同时也完成了对 PWM1A 和 PWM1B 的设置。

3、移相设置

多个 PWM 之间的同步，最主要的应用是移相全桥拓扑。同时，也应用于含多个功率支路的场景以及一个环路补偿器驱动多路驱动的场景，比如传统全桥拓扑。

同步时序的设置在寄存器 DPWMPHASETRIG 中实现，代码如下。

Dpwm1Regs.DPWMPHASETRIG.all = period >> 1;

该代码是实现两路驱动相位相差 180°的第一步操作，包含在对 DPWM1 的初始化函数 init_dpwm1()中。任何一个其它的 DPWM 都可以与 Dpwm1 同步（本例中是相位相差 180°）。实现代码如下。

Dpwm2Regs.DPWMCTRL1.bit.MSYNC_CH_SEL = 0;

Dpwm2Regs.DPWMCTRL1.bit.MSYNC_SLAVE_ENA = 1;

其中，MSYNC_SLAVE_ENA 标示位用来使能同步；MSYNC_CH_SEL 标示位用来选择要同步的 DPWM，该代码包含在对 DPWM2 的初始化函数 init_dpwm2()中。此时，DPWM1 和 DPWM2 相位相差 180°的设置已经完成。

4、其它重要设置

1）误差采样时刻

输出电压的误差采样时刻，取决于寄存器 DPWMSAMPTRIG。它决定了每周期中 EADCD 的误差采样时刻。实现代码如下：

Dpwm1Regs.DPWMSAMPTRIG.all = (period >>2) * 0.8;

该代码包含在对 DPWM1 的初始化函数 init_dpwm1()中，它表示在每个周期的 80%位置处采样误差，即接近整个周期的结束处。在传统全桥中，由于 DPWM1 和 DPWM2 相位相差 180°，因此，无需再在 DPWM2 的初始化中进行该设置。

2）CLA 的选择

以下代码实现了对 CLA1 的选择并使能该 CLA1。对于 UCD3028 而言，外部硬件电路设计时需要将输出电压以差分形式反馈到 EAP1 和 EAN1。

Dpwm1Regs.DPWMCTRL1.bit.CLA_CH_SEL = 0;

Dpwm1Regs.DPWMCTRL1.bit.CLA_ENABLE = 1; //use cla now

对于每周期根据采样误差计算的新占空比何时更新对应的 DPWM，下面代码给出了设置。

Dpwm1Regs.DPWMCTRL1.bit.UPDATE_END_PRD_ENA = 1;

即，必须在每个周期即将结束时去更新 DPWM，防止错误的发生。

3）DPWM 模式的选择

对 DPWM 是选择 Normal 模式还是其它模式，下面代码给出了实现。该代码设置 DPWM1 为 Normal 模式并使能该模式。

```
Dpwm1Regs.DPWMCTRL1.bit.PWM_ENA = 1; //and now enable the PWM.
```

10 12 位通用 ADC 初始化及相关设置

10.1 通用 ADC 简述

UCD3028 芯片包含了 10 个通用 ADC，通道号分别是 CH0~CH8，CH15。其中，CH15 用于内部温度检测，无外部引脚。

正常情况下，每个通道的采样时间是 2us，数据转换时间是 3us，共计 5us，因此，每个通道的采样率为 200ksps。

10.2 通用 ADC 的相关寄存器

1、ADC 控制寄存器

该寄存器共有 23 位，重点需要对低 9 位进行初始化设置，如下：

Bit Number	8	7:4	3:1	0
Bit Name	SINGLE_SWEEP	MAX_CONV	EXT_TRIG_SEL	ADC_ENA
Access	R/W	R/W	R/W	R/W
Default	0	0000	000	0

图 16：ADC 控制寄存器

对该寄存器的初始化由下面代码实现：

```
AdcRegs.ADCCTRL1.half.HALF0 =0x100+ (8 << 4) +1;
```

上述代码表示，通用 ADC 采用单次采样模式；共完成 8 个 ADC 的转换；无软件中断，使能 ADC。

2、采样时序控制寄存器

采样时序控制寄存器共有 6 个，每个包含 15 位，可以设置 3 个 ADC 的时序。如下所示。

Bit Number	14:10	9:5	4:0
Bit Name	SEQ2	SEQ1	SEQ0
Access	R/W	R/W	R/W
Default	00000	00000	00000

图 17：采样时序控制寄存器

具体采样的时序控制代码如下：

AdcRegs.ADCSEQSEL1.half.HALF0 = 3 + (1 << 5) + (2 << 10);

AdcRegs.ADCSEQSEL2.half.HALF0 = 3 + (6 << 5) + (1 << 10);

AdcRegs.ADCSEQSEL3.half.HALF0 = 7 + (8 << 5) + (15 << 10);

AdcRegs.ADCSEQSEL4.half.HALF0 = 4;

即，时序设置如下表所示。

采样时序	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
通道号	CH3	CH1	CH2	CH3	CH6	CH1	CH7	CH8	CH15	CH4

11 时钟中断初始化及状态机介绍

11.1 时钟中断简述

UCD3028 时钟可以实现中断和 PWM 的产生。下图是 T16PWMX 的框图。ICLK 的频率是 15.6MHz，应用中断模式时需要装载数据到寄存器的 Prescaler 中。

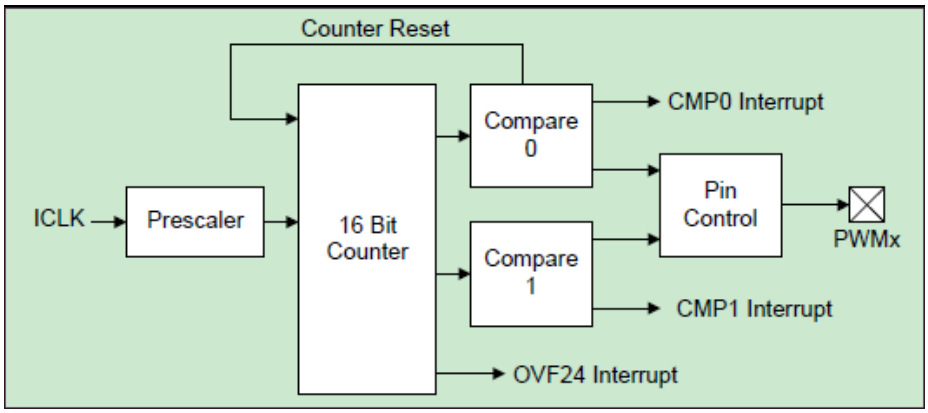

图 18：时钟硬件框图

11.2 时钟中断软件实现

软件实现的具体代码如下：

1、初始化函数中的实现

TimerRegs.T16PWM0CMP0DAT.bit.CMP_DAT = 1587;

//1587×(1/15.6MHz)=100us。即每 100us 便触发一次中断。

TimerRegs.T16PWM0CMPCTRL.bit.CMP0_INT_ENA = 1;

//计数器复位时触发 CMP0 中断。

TimerRegs.T16PWM0CNTCTRL.bit.CMP_RESET_ENA = 1;

//允许 compare0 的中断可以复位整个 counter

```
TimerRegs.T16PWM0CNTCTRL.bit.SW_RESET = 1;
```

//复位后计数器持续运行

2、中断函数体中的实现

```
TimerRegs.T16PWM0CMPCTRL.bit.CMP0_INT_ENA = 1;
```

//计数器复位时触发 CMP0 中断。

```
TimerRegs.T16PWM0CNTCTRL.bit.CMP0_INT_FLAG = 1;
```

//该标识位表示一个有效的 compare0 中断事件。当对该位赋值 1 时，即进行了清空操作。

11.3 时钟中断与状态机

由上节所介绍的时钟中断软件实现知，该 16 位时钟每 100us 产生一个中断。该时钟中断在 UCD3028 软件中设置为标准中断，用以进入状态机。

1.时钟中断进入状态机的软件设置

下面代码实现了时钟触发中断进入状态机的功能。函数 standard_interrupt()包含了各个状态机的具体实现。

```
#pragma INTERRUPT(standard_interrupt,IRQ)
void standard_interrupt(void)
```

2.状态机

UCD3028 软件中采用状态机管理电源，可以灵活快捷的实现各种功能和处理各种故障，共包含如下状态机：

1）POWER_ON_DELAY

系统上电后，主功率支路需要延时 1s 开始工作，以避开系统开机初始时刻输入母线上的扰动。上述操作在该状态机中实现。

2）STATE_IDLE

该状态机为主功率支路开始工作做好准备。进入该状态机后首先判断当前系统是否存在故障，如无各种故障，各个状态标志位被清空并使能模拟比较器快中断。

3）STATE_START_DELAY

该状态机实现延时 1ms 的功能。

4）STATE_PREBIAS

该状态机实现预偏置起机功能。在该状态机内，读取当前输入电压和输出电压并据此计算当前占空比和当前 EADC DAC 的值。

5）STATE_RAMP_UP

该状态机实现输出电压的缓启动。进入该状态机后，系统从 STATE_PREBIAS 状态机计算出的占空比开始缓起并逐渐增大占空比，实现输出电压的缓慢上升。

6）STATE_REGULATED

进入该状态机则表明系统已经开始稳定输出。该状态机主要完成故障检测，每 100us 进入一次，当前如无任何故障，则退出该状态机。

7）STATE_RESTART_DELAY

该状态机主要完成系统在触发过压后实现打嗝时间的设置，打嗝间隔时间为 1s。

下图给出了各个状态机在外界条件触发下的跳转示意。

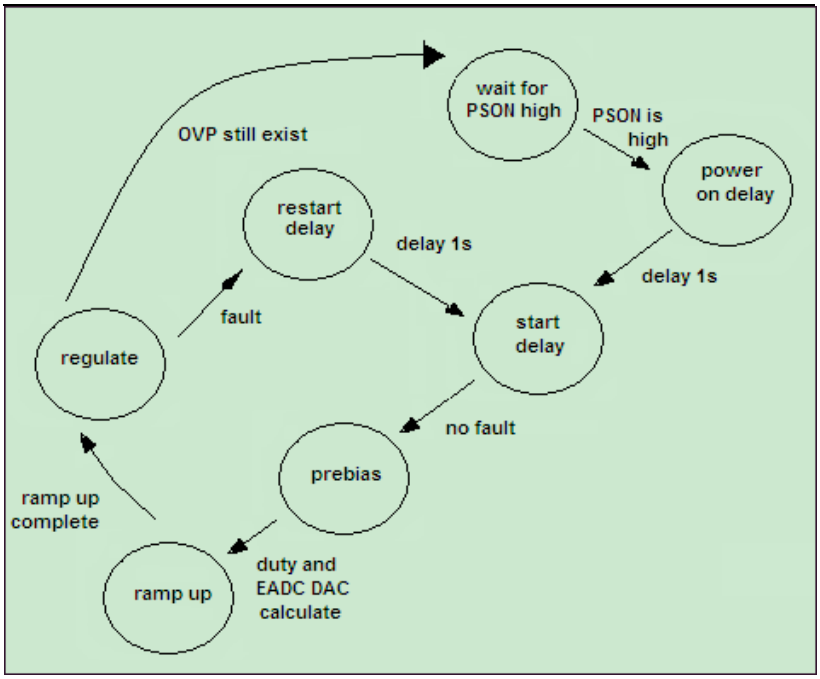

图 19：状态机跳转示意图

12 UCD3028 芯片的存储模块

12.1 存储模块简述

1、Boot ROM

Boot ROM 的空间为 4Kbyte，包含了用于 PMBus 通信和软件烧录的初始启动程序。芯片上电后，该程序便立即执行，同时判断芯片内部是否有一个有效的 Flash 程序（主程序），如果具备，主程序便开始执行。

2、Flash

包含了程序区（32Kbyte）和数据区（2Kbyte）。其中程序区主要用来存放 firmware，而数据区主要用来完成 firmware 数据的存储和数据的搬运。

3、RAM

RAM 空间大小为 4Kbyte，主要用来进行程序运行期间的数据存放。

12.2 芯片上电后的启动流程

1、执行 Flash 程序的手动选择操作方式

UCD3028 上电后首先运行 ROM 中的程序，此时可以通过发送命令或在 GUI 软件中手动操作，使芯片立即执行 Flash 程序。具体实现为通过主设备发送 0xF0 命令到 0xB。下图是在 GUI 软件中进行手动操作的示意图。

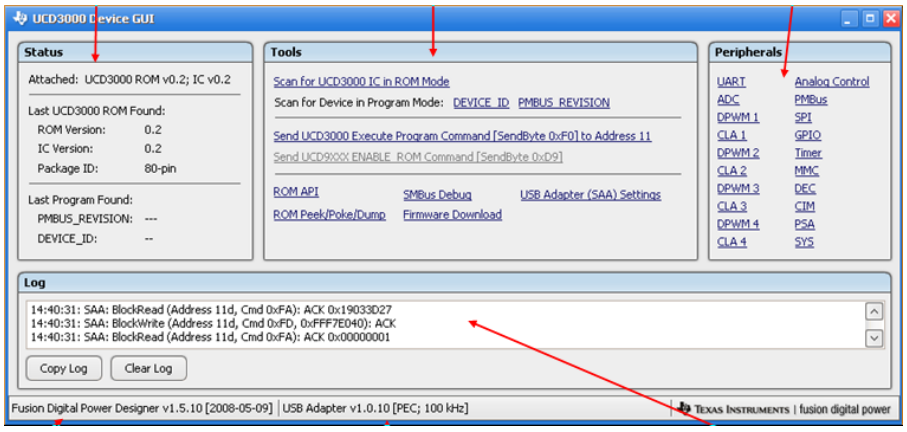

图 20：手动操作示意图

2、禁选校验和设置

上电后，Boot ROM 代码进行程序检验和操作，该校验和位于程序区的最后。如果该校验和与 ROM 计算结果匹配，Boot ROM 便自动从程序区开始运行，此时便不需要等待 0xF0 命令。

在调试期间，宜将芯片设置为手动操作模式，防止芯片因为故障而锁死。这可以通过在烧写 firmware 时勾选“不写入程序校验和”来实现。下图为其示意图。

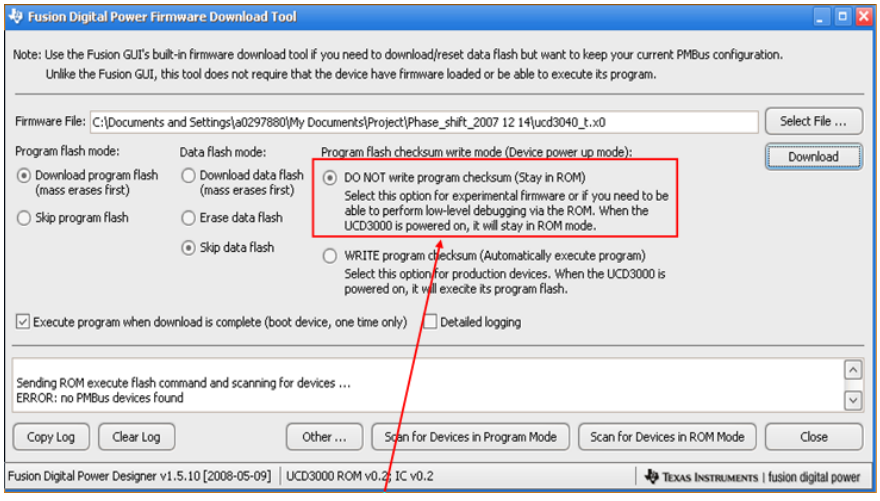

图 21：禁选校验和设置

13 UCD3028 软件流程图

UCD3028 软件程序主要包含了主程序，标准中断程序，快速中断程序和通信程序四个部分。各个部分的功能如下。

1.主程序，完成外部硬件的初始化，随后进入一个无条件的循环程序进行无限循环，等待中断触发。

2.标准中断程序，由时钟中断触发，每 100us 触发一次，主要完成状态机的跳转。在状态机中包含了电源的运行控制与管理及突发故障的处理。

3．快速中断程序，由模拟比较器触发，主要完成快速过压保护、快速过流保护和前馈功能的实现。快速过压保护时，第一次触发的处理和第二次触发的处理有所不同。快速过流保护和前馈功能的处理详见第七章。

4．通信程序，主要完成外界系统与 UCD3028 的通信，以进行参数的配置与更改、读取输出电压和输出电流等参数及其它交互需求。

软件流程图见图 22。

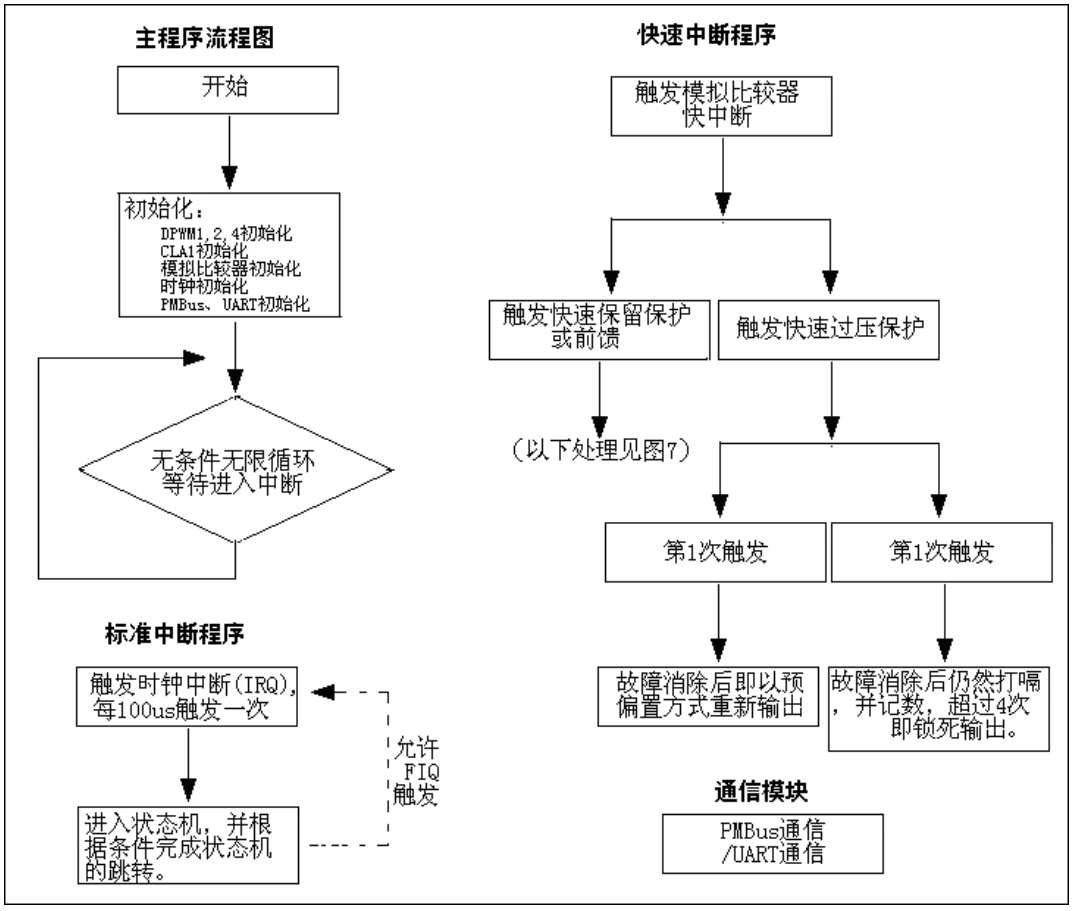

图 22：UCD3028 软件流程图

14 参考资料

1. UCD30xx datasheet (SLUS868B), Texas Instruments Inc., 2010

2. UCD30xx Central Interrupt Module (CIM) Programmer's Manual, Texas Instruments Inc., 2010

3. UCD30xx Fusion Digital Power Peripherals Programmer's Manual, Texas Instruments Inc., 2010

4. UCD30xx General Purpose 12-bit ADC (ADC12) Programmer's Manual, Texas Instruments Inc., 2010

5. UCD30xx Miscellaneous Analog Control _MAC_, Texas Instruments Inc., 2010

6. UCD30xx Timer Modules Programmer's Manual, Texas Instruments Inc., 2010

7. Fast Vin Sensing and Control Algorithm for Surge Protection, Texas Instruments Inc., 2010

重要声明

德州仪器(TI) 及其下属子公司有权在不事先通知的情况下, 随时对所提供的产品和服务进行更正、修改、增强、改进或其它更改，并有权随时中止提供任何产品和服务。客户在下订单前应获取最新的相关信息，并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的硬件产品的性能符合TI 标准保修的适用规范。仅在TI 保证的范围内，且TI 认为有必要时才会使用测试或其它质量控制技术。除非政府做出了硬性规定，否则没有必要对每种产品的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用TI 组件的产品和应用自行负责。为尽量减小与客户产品和应用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何TI 专利权、版权、屏蔽作品权或其它与使用了TI 产品或服务的组合设备、机器、流程相关的TI 知识产权中授予的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从TI 获得使用这些产品或服务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是TI 的专利权或其它知识产权方面的许可。

对于TI 的产品手册或数据表，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况下才允许进行复制。在复制信息的过程中对内容的篡改属于非法的、欺诈性商业行为。TI 对此类篡改过的文件不承担任何责任。

在转售TI 产品或服务时，如果存在对产品或服务参数的虚假陈述，则会失去相关TI 产品或服务的明示或暗示授权，且这是非法的、欺诈性商业行为。TI 对此类虚假陈述不承担任何责任。

TI 产品未获得用于关键的安全应用中的授权，例如生命支持应用（在该类应用中一旦TI 产品故障将预计造成重大的人员伤亡），除非各方官员已经达成了专门管控此类使用的协议。购买者的购买行为即表示，他们具备有关其应用安全以及规章衍生所需的所有专业技术和知识，并且认可和同意，尽管任何应用相关信息或支持仍可能由TI 提供，但他们将独力负责满足在关键安全应用中使用其产 品及TI 产品所需的所有法律、法规和安全相关要求。此外，购买者必须全额赔偿因在此类关键安全应用中使用TI 产品而对TI 及其 代表造成的损失。

TI 产品并非设计或专门用于军事/航空应用，以及环境方面的产品，除非TI 特别注明该产品属于“军用”或“增强型塑料”产品。只 有TI 指定的军用产品才满足军用规格。购买者认可并同意，对TI 未指定军用的产品进行军事方面的应用，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

TI 产品并非设计或专门用于汽车应用以及环境方面的产品，除非TI 特别注明该产品符合ISO/TS 16949 要求。购买者认可并同意，如果他们在汽车应用中使用任何未被指定的产品，TI 对未能满足应用所需要求不承担任何责任。

可访问以下URL 地址以获取有关其它TI 产品和应用解决方案的信息：

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP 机动性处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity		
	德州仪器在线技术支持社区	www.deyisupport.com	

邮寄地址： 上海市浦东新区世纪大道 1568 号，中建大厦 32 楼 邮政编码： 200122
Copyright © 2012 德州仪器 半导体技术（上海）有限公司