BMT Recharge
for new charging Algorithm

Version: 1.0
Release date: 2012-6-11

Document Revision History

Revision	Date	Author	Description
1.0	2012-8-1	Fengyun.Cao	Over Voltage Recharge
1.0	2012-8-1	Zixiang.Peng	Over Temp Recharge

Table of Contents

目录

1 过压回充

注意：此方法只限用于 SW OVP 恢复后 recharge，对于硬件 OVP 没有效果。

//充电器 OVP 后，恢复充电可以参考以下方法:
//定义相关的宏
#define VCHR_RECHARGE_SUPPORT

//(1) 在 **bmt_linear_li.c** 中修改状态机
static const CHR_FUNCBMT_CHRFUNC[][2] =
{
{BMT_CHRPRE_OFF, BMT_CHRPRE_ON} //CHR_PRE
,{BMT_CHRFAST_OFF, BMT_CHRFAST_ON} //CHR_FAST
,{BMT_CHRTOPOFF_OFF, BMT_CHRTOPOFF_ON} //CHR_TOPOFF
,{BMT_CHRBATFULL_OFF, BMT_CHRBATFULL_ON} //CHR_BATFULL
,{BMT_MEASURE_STOP,BMT_MEASURE_STOP} //CHR_ERROR
,{BMT_CHRHOLD,BMT_CHRHOLD} //CHR_HOLD
,{BMT_MAINTENANCE,BMT_MAINTENANCE}
#if defined(VCHR_RECHARGE_SUPPORT)
,{BMT_CheckChr, BMT_CheckChr }
#endif
};

//(2) 在 **bmt_linear_li.h** 中添加一个状态，
#define CHR_PRE 0
#define CHR_FAST 1
#define CHR_TOPOFF 2
#define CHR_BATFULL 3
#define CHR_ERROR 4
#define CHR_HOLD 5
#define CHR_POSTFULL 6
#define CHR_PRE_FULL_CHECK 7
#define CHR_MAINTENANCE 6
#define CHR_VCHRCHECK 7

//(3) 在 **bmt_timer_control.c** 中添加一个变量，用于标记电压过压了。
#if define (VCHR_RECHARGE_SUPPORT)
kal_bool voltage_error = KAL_FALSE; //全局变量
#endif
其他文件用到此变量，请 extern kal_bool voltage_error;

//(4) 在 **bmt_utility.c** 中修改 BMT_PhyCheck_VCharger
kal_bool bmt_PhyCheck_VCharger(BATPHYStruct *BATPHYS)

```
{
        static kal_uint8 over_charger_count = 0;
        static kal_uint8 low_charger_count = 0;

#if defined(DRV_BMT_HIGH_VCHG_ADAPTIVE_CHARGE_CURRENT_SUPPORT)
        if((BATPHYS->VCHARGER> bmt_charging_para.HIGH_VCHG_TABLE[VCHG_VOL_LEVEL
- 1][0]) )
#else
        if ( (BATPHYS->VCHARGER > bmt_charging_para.VCHARGER_HIGH) )
#endif // End of #if defined(DRV_BMT_HIGH_VCHG_ADAPTIVE_CHARGE_CURRENT_SUPPORT)
        {
                over_charger_count ++;
                drv_trace1(TRACE_ERROR, BMT_PHY_CHECK_OVER_CHARGER_COUNT_TRC,
over_charger_count);
        }
        else
        {
                over_charger_count = 0;
        }

        if ( (BATPHYS->VCHARGER < bmt_charging_para.VCHARGER_LOW) )
        {
                low_charger_count++;
        }
        else
        {
                low_charger_count = 0;
        }

        if ( (low_charger_count > VCHARGER_LOW_CHECK_COUNT) || (over_charger_count >
VCHARGER_OVER_CHECK_COUNT) )
        {
                low_charger_count = 0;

    #if defined (VCHR_RECHARGE_SUPPORT)
                        if( over_charger_count > VCHARGER_OVER_CHECK_COUNT )
                        {
                                voltage_error = KAL_TRUE;
                                over_charger_count = 0; //这个地方把它清 0，下次恢复充电后，
OVP 重新开始
                        }
    #endif
                return KAL_FALSE;
        }
        return KAL_TRUE;
}
```

//(5) 在 **bmt_timer_control.c** 中修改 BMT_PhyCheck

```
static void bmt_PhyCheck(BATPHYStruct *BATPHYS)
 {
        if (bmt_charging_para.bmt_check_charger)
        {
                if (bmt_PhyCheck_VCharger(BATPHYS) == KAL_FALSE)
                {
                        drv_trace0(TRACE_ERROR, BMT_INVALID_CHARGER_TRC);
                        bmt_sendMes2UEM(BMT_INVALID_CHARGER);

      #if defined (VCHR_RECHARGE_SUPPORT)      //修改状态机的状态
                                if( voltage_error )
                                        BMT.bat_state = CHR_VCHRCHECK;
                                else
            #endif
                        BMT.bat_state = CHR_ERROR;
                        return;
                }
        }
```

//6，在 **bmt_timer_control.c** 中实现 BMT_CheckChr (void)

```
#if defined (VCHR_RECHARGE_SUPPORT)
#define RECHARGEVOLT        5000000  //when the vcharge drop below the RECHARGEVOLT, then recharge again
#define CHECKCOUNT                  3
#define TCHECK              3       //每 3 秒检测一次

void BMT_CheckChr(void)
{
        BATPHYStruct BATPHYS;
        static kal_uint32 check_count = 0;
        bmt_CtrlCharge((kal_uint8)KAL_FALSE);      //Turn off Charge
        if (bmt_ObtainBMTPhystate(&BATPHYS)) //The measure is on
        {
                bmt_PhyCheck(&BATPHYS);                 //get the phy, if there is any other error, then stop all the process
                if( (BMT.bat_state == CHR_VCHRCHECK)&& voltage_error )         //if the bat_state is CHR_VOLTCHECK
                {
            BMT.pmictrl_state= PMIC_CHARGEOFF;              //OFF

            if((BATPHYS.VCHARGER <= RECHARGEVOLT))    //when the vcharger drop RECHARGEVOLT
            {
                check_count ++;
```

```
            }
            else
            {
                check_count = 0;
            }
            if( check_count >= CHECKCOUNT)          //when the chr volt is normal CHECKCOUNT
times, then return to charge again
            {
              voltage_error = KAL_FALSE;
            }
            kal_prompt_trace(MOD_BMT, "the check_count is %d!!!", check_count);
            kal_prompt_trace(MOD_BMT, "the voltage_error is %d!!!", voltage_error);
            bmt_timer_config(TCHECK * CHARGING_TIME_UNIT);//3                    // every 3
second to dectect one time
        }
        else
        {
            check_count = 0;
            kal_prompt_trace(MOD_BMT, "return to charge again!!!");
            bmt_charge_start();  // start to charge again
    // Here should send the MSG to MMI, so the icon in the screen could run again
        }
      }
}
#endif
```

2 过温回充

//温度检测出错后，当温度正常后，恢复充电可以参考以下方法:
//(1)定义温度回充的宏以及全局变量
#define TEMP_RECHARGE_SUPPORT
Kal_bool temp_error;

//由于新架构中已经将温度考虑到代码中，所以新架构中不需要跟老架构一样添加回充函数以及状态机
//(2) bmt_utility.c 中修改 BMT_PhyCheck_HighBatTemp 和 BMT_PhyCheck_LowBatTemp

```
kal_bool bmt_PhyCheck_LowBatTemp(BATPHYStruct *BATPHYS)
{
	static kal_uint8 low_temper_count = 0;

	if (BATPHYS->BATTMP <CHR_BATTMP_LOW_TEMP && BATPHYS->BATTMP > CHR_BATTMP_BAD_CONTACT)  /*battery temperature < 0C*/
	{
		low_temper_count ++;
		drv_trace1(TRACE_ERROR, BMT_PHY_CHECK_LOW_TRMPER_COUNT_TRC, low_temper_count);
	}
	else
	{
		low_temper_count = 0;
	}

	if (low_temper_count > LOW_BATTEMP_CHECK_COUNT)
	{
		#if define (TEMP_RECHARGE_SUPPORT)		//zixiang for tem_recharge
			low_temper_count = 0;
			kal_prompt_trace(MOD_BMT, "Low bat temp occurs!!");
		#endif

		return KAL_FALSE;
	}

	return KAL_TRUE;
}
```

//(3)bmt_utility.c 中修改

```
kal_bool bmt_PhyCheck_HighBatTemp(BATPHYStruct *BATPHYS)
{
	static kal_uint8 over_temper_count = 0;
```

```
        /*y=69.65-47.4x*/
        if (BATPHYS->BATTMP > CHR_BATTMP_HIGH_TEMP)/*battery temperature > 45C*/
        {
                over_temper_count++;
                drv_trace1(TRACE_ERROR,    BMT_PHY_CHECK_OVER_TEMPER_COUNT_TRC,
over_temper_count);
        }
        else
        {
                over_temper_count = 0;
        }

        if (over_temper_count > OVER_BATTEMP_CHECK_COUNT)
        {
                over_temper_count = 0;

                #if define (TEMP_RECHARGE_SUPPORT)//zixiang for tem_recharge
                        kal_prompt_trace(MOD_BMT, "High bat temp occurs!!");
                #endif

                return KAL_FALSE;
        }

        return KAL_TRUE;
}

//(4) bmt_timer_control.c 中修改 BMT_PhyCheck
static void bmt_PhyCheck(BATPHYStruct *BATPHYS)
 {
……..……..……..……..……..……
         if (bmt_charging_para.bmt_check_temp)
         {
                 if (bmt_PhyCheck_HighBatTemp(BATPHYS) == KAL_FALSE)
                 {
                         drv_trace0(TRACE_ERROR, BMT_VTEMP_OVER_TRC);
                        #if define (TEMP_RECHARGE_SUPPORT)          //zixiang                  for
tem_recharge
                         if(!temp_error)
                         bmt_sendMes2UEM(BMT_OVERBATTEMP);
                        #else
                         bmt_sendMes2UEM(BMT_OVERBATTEMP);
                        #endif
                        #if define (TEMP_RECHARGE_SUPPORT)          //zixiang                  for
tem_recharge
                                                BMT.bat_state = CHR_MAINTENANCE;
                        #else
```

```
                                                BMT.bat_state = CHR_ERROR;
                        #endif
                        return;
                }

                if (bmt_PhyCheck_LowBatTemp(BATPHYS) == KAL_FALSE)
                {
                        drv_trace0(TRACE_ERROR, BMT_VTEMP_LOW_TRC);
                        #if define (TEMP_RECHARGE_SUPPORT)        //zixiang                for
tem_recharge
                        if(!temp_error)
                        bmt_sendMes2UEM(BMT_LOWBATTEMP);
                        #else
                        bmt_sendMes2UEM(BMT_LOWBATTEMP);
                        #endif
                        #if define (TEMP_RECHARGE_SUPPORT)        //zixiang                for
tem_recharge
                                                BMT.bat_state = CHR_MAINTENANCE;
                        #else
                                                BMT.bat_state = CHR_ERROR;
                        #endif
                        return;
                }

                if (bmt_PhyCheck_BadBattContact(BATPHYS) == KAL_FALSE)
                {
                        bmt_sendMes2UEM(BMT_BATTERY_BAD_CONTACT);
                        BMT.bat_state = CHR_ERROR;
                        return;
                }
        }


//(5)修改状态机
#define CHR_MAINTENANCE          6

//(6)修改回充条件
kal_bool bmt_PhyCheck_ReChargeTemp(BATPHYStruct *BATPHYS)
{
        static kal_uint8 recharge_temper_count = 0;
        /*y=69.65-47.4x*/
        if (BATPHYS->BATTMP < CHR_BATTMP_RECHARGE_TEMP) /*battery    temperature    >
45C*/
        {
                recharge_temper_count++;
```

```
                kal_prompt_trace(MOD_BMT,"recharge_temper_count                    =
%d",recharge_temper_count);//ZIXIANG
                drv_trace1(TRACE_ERROR,
BMT_PHY_CHECK_RECHARGE_TEMPER_COUNT_TRC, recharge_temper_count);
        }
        else
        {
                recharge_temper_count = 0;
        }

        if (recharge_temper_count > RECHARGE_BATTEMP_CHECK_COUNT)
        {
                recharge_temper_count = 0;
                return KAL_TRUE;
        }

        return KAL_FALSE;
}

//(7) bmt_linear_li.c 中回充消息处理
static void BMT_MAINTENANCE(BATPHYStruct *BATPHYS)
{
        if (bmt_charging_para.bmt_check_temp)
        {
                kal_prompt_trace(MOD_BMT,"BMT_MAINTENANCE");//ZIXIANG
                if(KAL_TRUE == bmt_PhyCheck_ReChargeTemp(BATPHYS))
                {
                        temp_error = KAL_FALSE;
                        bmt_charge_start();
   // Here should send the MSG to MMI, so the icon in the screen could run
again
                }
                else
                {
                        kal_prompt_trace(MOD_BMT,"NON_RECHARGE");//ZIXIANG
                        temp_error = KAL_TRUE;
                        BMT.pmictrl_state = PMIC_CHARGEOFF;
                        bmt_CtrlCharge(KAL_FALSE);
                        bmt_timer_config(RECHARGE_TOFF_TIME*CHARGING_TIME_UNIT);
                }
        }

}
```

3 关于上层修改的一些建议

由于上述做法是 driver 层级的修改，已经起到了回充的功能，关于如何实现如下功能，

// Here should send the MSG to MMI, so the icon in the screen could run again

有如下思路，

1，背景

a> 对于如下写法是不建议的，

BMT_sendMes2UEM(BMT_USB_CHARGER_IN); or
BMT_sendMes2UEM(BMT_CHARGER_IN);

先不特别区分 BMT_USB_CHARGER_IN 和 BMT_CHARGER_IN，charger IN/OUT 是配对的，都有防止重复插入/拔出的机制，在 IN 之后如果没有 OUT, 如果再去 IN 的话，L4 就不会把相应的 MSG 上报给 MMI，

所以我们需要在 pmic_status_enum 中新增一个 pmic_status 表示这种 case，即 PMIC_CHARGER_RESUME.

b> 在执行 BMT_sendMes2UEM(xxx)时，会发送 BMT→L4C 的 MSG:MSG_ID_DRVUEM_PMIC_IND,其中的 handler 中这个会根据相应的 status 决定是否要发送 ind。

uemdrv_pmic_ind_hdlr—>uemdrv_pmic_ind,

```
case PMIC_USB_NO_CHARGER_IN:    /* pass */
{  …     }
```

2，修改步骤

2-1，修改 device.h 中的 pmic_status_enum ，在最后添加 PMIC_CHARGER_RESUME，

有两个 device.h （code\interface\ps\include，code\ps\l4\include ）

```
typedef enum
{
 …
 PMIC_BATTERY_IN,
 PMIC_BATTERY_OUT,
 PMIC_CHARGER_RESUME
}
```

2-2，修改 drvsignals.h 中 BMT_CHR_STAT，在 最后添加 BMT_CHARGER_RESUME

```
/*Communicated with MMI, charging status*/
typedef enum
{
   …
   BMT_BATT_IN,
   BMT_BATT_OUT,
   BMT_CHARGER_RESUME
}BMT_CHR_STAT;
```

2-3，在 bmt.c 中，如上述 code，在 BMT_CheckChr 中添加

```
else
{
    check_count = 0;
    kal_prompt_trace(MOD_BMT, "return to charge again!!!");
    bmt_charge_start();  // start to charge again
    // Here should send the MSG to MMI, so the icon in the screen could run again
    BMT_sendMes2UEM(BMT_CHARGER_RESUME);
 }
```

2-4，在 uem_battery_status_convert 中，增加一个 case：

```
        case BMT_CHARGER_RESUME :
            bs = PMIC_CHARGER_RESUME;
            break;
```

2-5，在 uem_proc_msg.c 中 uemdrv_pmic_ind 中加这个

```
    case PMIC_CHARGING_RESUME:
        l4cuem_battery_status_ind(PMIC_CHARGER_IN, vbat_level);
```

or

```
        l4cuem_battery_status_ind(PMIC_USB_CHARGER_IN, vbat_level);
```