**Code 说明**

流程图:

初始化（上次分压电压设为中间值，PWM设50%占空比,检测频率设为MAX\_PWM/2）

检测比较结果

低电平

高电平

提高占空比

降低占空比

判断检测电压是否翻转

是

否

保存检测电压,减小翻转次数n, 保存本次检测电压

n重设翻转次数

判断连续翻转次数n是否为0

否

是

n重设翻转次数,计算检测电压,与上次结果平均后推算电池电压,延时1000ms

延时10ms

当此循环结束

本code采用task方式来实现，请根据实际情况自行更改方式。

首先保证检测口（gpio）设置为输入模式(本code因为接在gpio按键上，所以不需设置)。

以下是参照代码做简要的使用介绍（红色字体部分）：

#define TASK\_STK\_SIZE 1024 /\* Size of each task's stacks (# of WORDs) \*/

OS\_STK TaskStartStkMyTask[TASK\_STK\_SIZE];

unsigned short my\_task\_id = 66;

**#define MAX\_PWM 260**

**#define RES\_COEFF ((float)3/5)**

**#define VOLTAGE\_DROP 0**

**#define TURN\_NUM 4**

**#define V\_CHK\_COND (\*(volatile unsigned \*)0x12000a8 & (1<<5))**

void MyTask(void \*arg)

{

unsigned char n = TURN\_NUM;

unsigned char last\_drct = 1;

float v\_val;

int high\_val = MAX\_PWM/2;

int low\_val = MAX\_PWM/2;

float last\_val = (3.3\*((float)high\_val/MAX\_PWM));

**set\_lcd\_pwm0(low\_val, high\_val);**

AVTimeDly(10);

printf("start checking voltage...\n");

while(1)

{

if(V\_CHK\_COND)

{

high\_val = (high\_val+1)>MAX\_PWM?MAX\_PWM:(high\_val+1);

low\_val = (low\_val-1)>0?(low\_val-1):0;

}

else

{

high\_val = (high\_val-1)>0?(high\_val-1):0;

low\_val = (low\_val+1)>MAX\_PWM?MAX\_PWM:(low\_val+1);

}

set\_lcd\_pwm0(low\_val, high\_val);

if(last\_drct == !V\_CHK\_COND)

{

n = TURN\_NUM;

}

else

{

last\_drct = !V\_CHK\_COND;

--n;

last\_val = (3.3\*((float)high\_val/MAX\_PWM));

}

if(!n)

{

n = TURN\_NUM;

v\_val = (3.3\*((float)high\_val/MAX\_PWM));

v\_val = (((v\_val + last\_val) / 2) / (RES\_COEFF)) + VOLTAGE\_DROP;

printf("voltage is %.3fV\n",v\_val);

AVTimeDly(1000);

}

else

AVTimeDly(10);

}

}

1, **MAX\_PWM**。

时钟源/PWM频率。时钟源若是系统频率建议600k,时钟源若是晶振建议200k（若设置600k,会使基值较小，每阶幅度较大，从而造成精度降低）。

2, **RES\_COEFF**

电阻系数，例:R12/(R9+R12)。

3, **VOLTAGE\_DROP**

压降。指电池电源到测试点之间的压降。

4, **TURN\_NUM**

翻转次数。获取电压值条件。须为大于0的整数。

5, **V\_CHK\_COND**

需要增加占空比的判断条件

6, **set\_lcd\_pwm0(low\_val, high\_val);**

设置PWM，请根据实际应用进行修改。