英创 ESM335x 嵌入式主板 Linux 系统输入脉冲计数

1、综述

在工业控制中，经常需要获取脉冲信号计数值、频率、周期、占空比等参数。英创嵌入式主板 ESM335X 系列 Linux 系统现已实现外部输入脉冲信号的计数、频率、周期、占空比测量功能。

主要功能及技术指标如下：

1. 读取一段时间内的外部输入脉冲信号计数值。
2. 外部输入脉冲信号周期、有效脉宽测量。
3. 根据测得周期计算外部输入脉冲信号瞬时频率。
4. 根据测得计数值和测量时间间隔计算两次有效信号读取时间内外部输入脉冲信号重复频率（平均频率）。
5. 测得误差 200KHz 左右时最大，瞬时频率误差不超过 0.1%,重复频率误差不超过 0.005%，占空比误差不超过 0.05%，计数值测量准确无误差。

2、硬件连接

ESM335X 系列嵌入式主板引出了 3 路 PWM 输出，其中两路（PWM1 和 PWM2）可以用来进行外界输入的脉冲信号计数、频率、占空比测量，相应的 GPIO 复用脚为 GPIO6 和 GPIO7, 对应的引脚请参考光盘资料《ESMARC 335x 工控主板数据手册》，用户使用脉冲波输入计数功能时可将外来信号接到上述两个 GPIO 管脚中的任意一个，并且地线与开发板接地引脚相连，然后在应用程序中获得计数值、频率、占空比。若用户启用了脉冲输入计数功能，则相应管脚不能再作为 PWM 脉冲输出或 GPIO 使用。

使用注意事项:

1. GPIO 管脚最大只允许输入 3.3V 电压，超过将会导致开发板损坏！以开发板接地脚为基准，负值电压无效，不符合电压要求时需要设计缓冲放大电路。
2. 输入脉冲波最大频率不应超过 200KHz！超过之后由于硬件限制测量误差将急剧变大。

3、应用程序

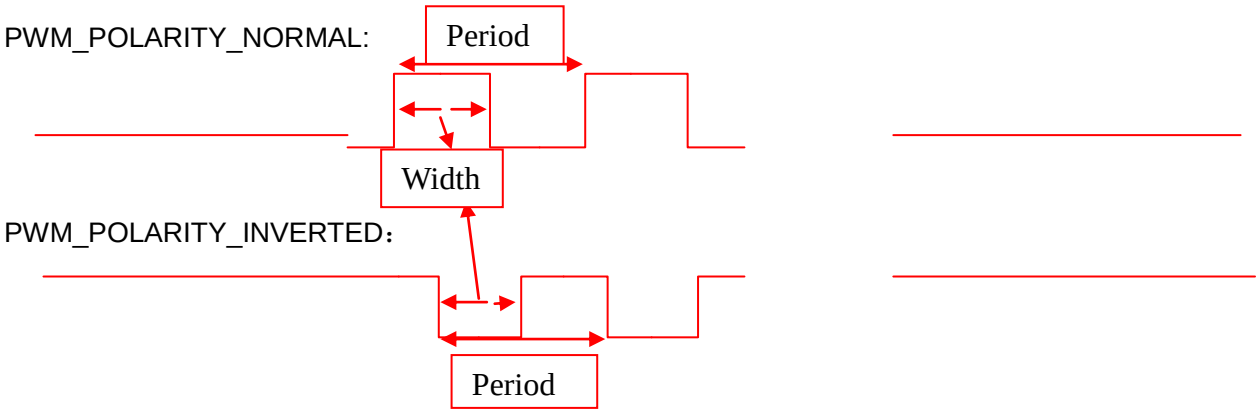
为了简化用户使用脉冲计数功能，内部使用了原 pwm 输出模块，Linux 系统内部没有增加新的设备节点，使用设备的计数功能（capture 模式，后文简称 cap），仍然需要使用 pwm 模式的设备节点进行模块功能设置。

（1）程序中使用到的结构体及常量定义

与内核直接交换数据读取设备原始测量值的结构体为 cap_config_info,应用于 read、write 函数：

```
typedef struct cap_config_info
{
    //以下为输入参数
    unsigned int    dwPolarity;     //设置输入信号极性
    unsigned int    dwMaxFreq;      //设置输入信号最大频率
    //以下为输出参数，输入无效
    unsigned long   dwTimeUs;       //两次读取之间的时间差
    unsigned int    dwCount;        //两次读取之间的计数值
    unsigned int    dwPeriodNs;     //读取时刻脉冲周期
    unsigned int    dwWidthNs;      //读取时刻的脉冲有效脉宽
}CAP_INFO, *PCAP_INFO;
```

1. dwPolarity 用于设置输入脉冲信号极性,可设为 PWM_POLARITY_NORMAL 和 PWM_POLARITY_INVERTED，定义在 pwm_api.h 头文件中



2. dwMaxFreq 等于 0 时用于停止计数功能，dwMaxFreq 不为零时用于设置输入信号最大频率，最大频率不应超过 200KHz，输入单位为 Hz。
3. dwTimeUs 为输出参数，读取获得上次读操作（或使能操作）到本次读操作之间的时间差，单位为 us。
4. dwCount 为输出参数，读取获得上次读操作（或使能操作）到本次读操作之间的计数值，单位为 个。
5. dwPeriodNs 为输出参数，读取获得本次读操作时脉冲信号周期， 单位为 ns。
6. dwWidthNs 为输出参数，读取获得本次读操作时脉冲信号有效脉冲宽度，单位为 ns。
7. 用户可根据 dwPeriodNs 和 dwWidthNs 计算占空比。
8. 用户可根据 dwPeriodNs 计算读操作时的脉冲信号频率。
9. 用户可根据 dwTimeUs 和 dwCount 计算两次读操作之间的输入信号的平均频率。

（2）函数及系统调用

在进行计数操作时，首先打开相应的设备节点/dev/em335x_pwmX,X 为编号（1 或者 2）,使能设备开始计数相关代码：

a) 打开设备节点：

```
int                                 npwm = 1;
sprintf( device, "/dev/em335x_pwm%d", npwm );
fd = open(device, O_RDWR);
if ( fd < 0)
{
        printf("can not open /dev/em335x_pwm%d device file!\n", npwm);
        return -1;
}
printf( "Open %s\n", device );
```

b) 使能 cap 模式：

```
int CAP_Start(int fd, unsigned int polarity, unsigned int maxfreq )
{
        int                                 rc;
        struct cap_config_infoconf;
        memset(&conf, 0 ,sizeof(struct cap_config_info))


        conf.dwPolarity = polarity;
        conf.dwMaxFreq = maxfreq;


        rc = write(fd, &conf, sizeof(struct cap_config_info));
        if ( rc == 0 )
                return rc;
        else
        {
                printf( " config for cap model failed!\n");
                exit(1);
        }
   }
```

用户只需要调用此函数即可使能脉冲信号计数功能，并且计数功能开始计数。也可自己设置参数调用 write 使能 cap 模式。如：

```
#include "pwm_api.h"

unsigned int   polarity = PWM_POLARITY_NORMAL;
unsigned int   maxfreq = 200000000;
CAP_Start( fd, polarity, maxfreq );
```

c) 在计数过程中可以调用 read 函数或者我们提供的 CAP_Read 函数读取测量数值！

```
int CAP_Read(int fd, struct cap_config_info* conf )
{
        printf ( "reading^^^^^^^\n" );
        int                                     rc;


        rc = read(fd, conf, sizeof(struct cap_config_info));
        return rc;
}
```

用户可根据自己需要在任意时刻选择调用此函数或者直接调运 read 获取测量值。

根据测量值可以计算脉冲频率并转换单位，注意测得 dwCount 小于 2 时其他参数均无效：

```
        double PeriodUs ;               //单位us
        double WidthUs ;                //单位us
        double Duty ;           //单位%
        int     Count;                  //个数
        double Freq;                    //单位Hz
        double AVGFreq;                 //单位Hz
        unsigned int TimeUs;            //单位 us
        CAP_Read ( fd, &conf );

        Count = conf.dwCount;
        if(Count>1)
        {
                PeriodUs = (double)conf.dwPeriodNs/1000.0;          //单位转换
                WidthUs = (double)conf.dwWidthNs/1000.0;
                Duty = (double)conf.dwWidthNs*100.0/conf.dwPeriodNs;
                Freq = 1000000000.0/(double)conf.dwPeriodNs;
                AVGFreq = (double)conf.dwCount*1000000/(double)conf.dwTimeUs;
        }
        else
        {
```

```
            PeriodUs = 0;
            WidthUs = 0;
            Duty = 0;
            Freq = 0;
            AVGFreq = 0;
        }
```

d) 使用完成后需要关闭计数功能，同样可以自行设置参数调用 write 或者使用 CAP_Stop，也可以直接使用 close 关闭设备节点同事停止使用计数功能：

```
int CAP_Stop(int fd )
{
        printf ( "stopping^^^^^^^\n" );
        int                                     rc;
        struct cap_config_infoconf;

        memset( &conf, 0, sizeof(struct pwm_config_info));
        conf.dwMaxFreq = 0;
        rc = write(fd, &conf, sizeof(struct cap_config_info));

        return rc;

}
```

调用上述函数：

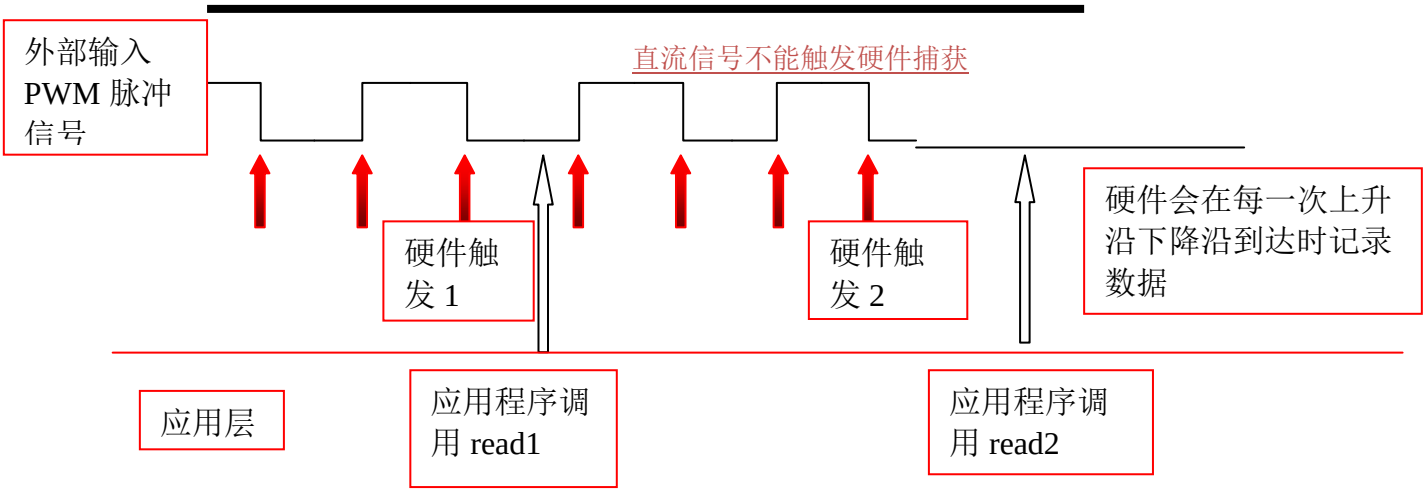
```
CAP_Stop( fd);
```

使用完成后需要关闭设备节点。

```
close(fd);
```

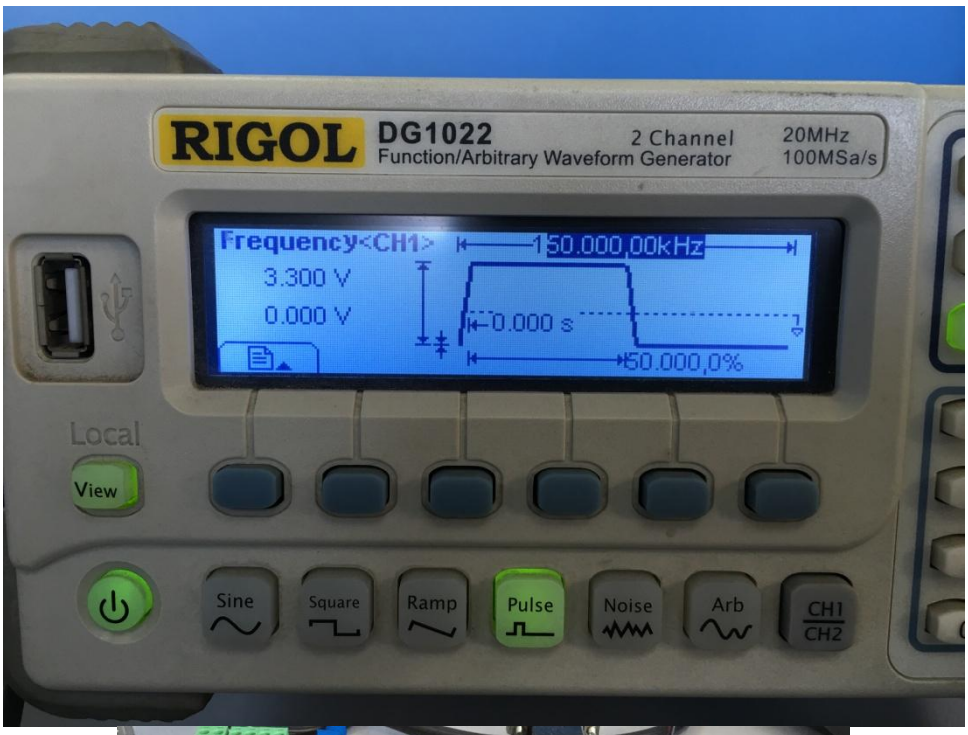
e) 触发机制（选读）

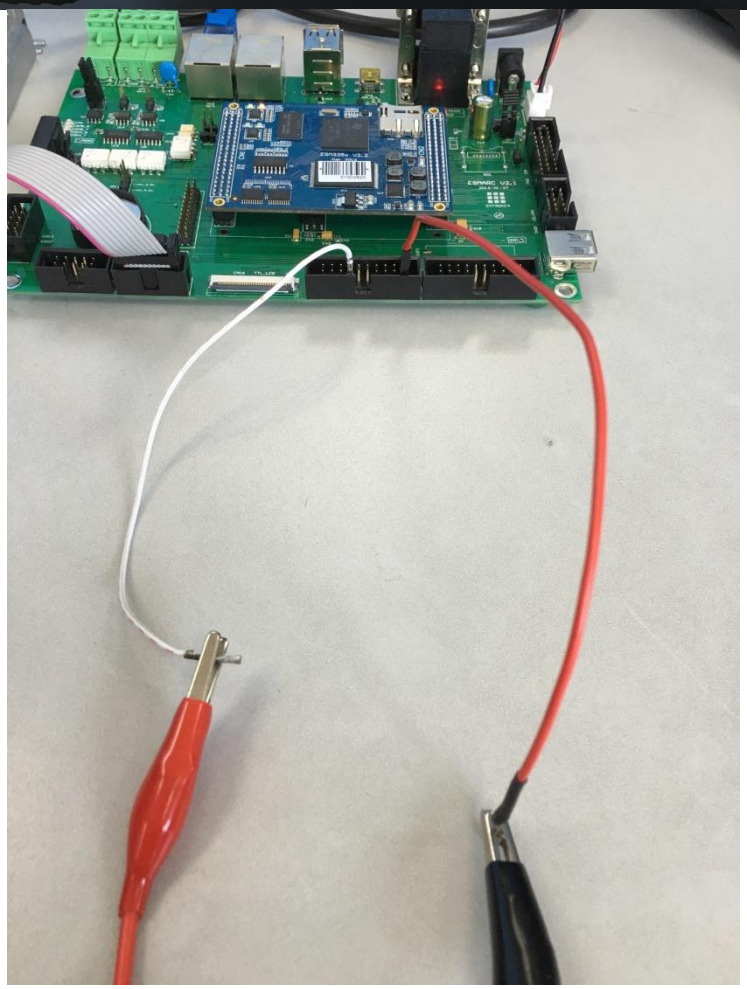
如下图所示，在每一次上升沿或者下降沿处会触发硬件捕获功能，获得输入脉冲信号的参数并保存，应用程序中在任意时刻调用读操作得到的数据是最近一次上升沿或下降沿处的数据。此图中调用 read1 一之后调用 read2，得到的计数值为 2，时间差为（硬件触发 2-硬件触发 1）的时间差。用户使用时请注意输入信号有效时间段。所得时间不能作为两次读操作之间的时间差使用。如果要将读取的时间值作为两次读操作的时间值需要在前一次读操作后调用 CAP_Start 或直接用 write 函数传递相同参数清零硬件保存值。



4、实验测量

由于硬件的限制，测得瞬时频率、周期、占空比精度有限。硬件操作使用的时钟信号为 100MHz，即周期、有效脉宽时间只能得到高于 10ns 的数值。平均频率的测量需要保证整个测量时间段内输入脉冲信号一直有脉冲输入！平均频率的误差整个测量范围内不超过 10Hz。实验使用 RIGOL DG1022 信号发生器作为外部脉冲信号源，可以调整信号周期占空比，设置一定时间内的脉冲个数。


RIGOL
DG1022
Function/Arbitrary Waveform Generator
2 Channel
20MHz
100MSa/s
Frequency<CH1>
150.000,00kHz
3.300 V
0.000 V
0.000 s
50.000,0%
Local
View
Sine
Square
Ramp
Pulse
Noise
Arb
CH1
CH2



频率测量（kHz）：

输入	200.000	197.000	97.000	1.020
瞬间频率	200.000	196.850	96.993	1.020
平均频率	199.992	196.992	96.998	1.020

测得瞬时数据误差随频率增加而变大，具体数据可根据测量时间最小值 10NS 进行计算

在 150KHz 时测量的不同占空比值如下表：

输入%	90.00	50.00	20.00	10.00
测得%	89.96	50.00	19.97	9.91

在 100KHz 时测量的不同占空比值如下表：

输入%	90.00	50.00	20.00	10.00
测得%	90.00	50.00	20.00	10.00

在 100KHz 时连续测量的 1S 内脉冲个数如下表（信号源输入个数设置模式上限 50K 个，通过设置信号源取不同输入值）：

输入	39 999	23 338	8 766	432
测得	39 999	23 338	8 766	432

注：当频率升高时，系统高负荷运转，实际读取时间差变化加大，如果连续读取，读到的计数值要以读到的时间差为准进行数值判断，实际读取个数并无误差

计数值在允许输入频率内无误差！

如果需要使用此功能或有任何疑问，请和我们联系。