智能车模拟摄像头

图像采集方法详解

--By LastRitter

superyongzhe@163.com

www.znczz.com

2009 年 11 月 16 日

目录

一、 智能车程序的组成部分

基于CCD或者CMOS模拟摄像头的智能车程序主要包含以下部分：

--图像，速度，加速度等数据采集；

--数字图象处理，从图像中获取赛道信息；

--以赛道信息，速度，加速度等传感器数据为输入参数进行自动控制，有PID，棒棒，模糊控制等算法。

--调试模块，要能将车的行驶状态等数据显示出来，主要有串口，液晶。最好还要有设置参数的按键，设计良好的调试模块能大大的加速我们的调试过程，不要觉得是浪费时间。

二、 采集方案选择

对于摄像头图像采集，也可以用OV6620数字摄像头模块，或者使用高速外部AD进行采集。而在这篇文章中我主要要讲的是使用PAL制式黑白摄像头和单片机片内部AD来进行图像采集。另外根据摄像头的安装方式不同，也有旋转90度进行采集的。对于整个程序的流程也有很多不同，有采完一幅图像后进行处理的，也有采集一行就进行处理的。

这都是大同小异的地方，我在这里只讲旋转了90度，同时是采集完整一副图像后再处理的方案。之所以旋转90度是因为单片机速度的限制，纵向采集的点数比较多，横向的点数少的原因。旋转后可以解决这个问题，但是付出的代价是只能一整幅图像采集完后才能进行图像处理，增加了时间延迟。这是我当初选用的方案，但我现在并不建议使用这种方案，当车的速度很快时，这个时间延迟会很难以忍受的。

三、 PAL信号格式

在采集图像之前，我们首先要知道摄像头输出信号的特性。目前的模拟摄像头一般都是PAL制式的，输出的信号由复合同步信号，复合消隐信号和视频信号组成。

视频信号：真正的图像信号，对于黑白摄像头，图像越黑，电压越低，图像越白，电压越高。在这里我们通过AD采集来得到亮度信号。

复合同步信号：用于控制电视机的电子枪对电子的偏转。当电子枪收到行同步信号时，电子束就从上一行的最右端移动到下一行的最左端。当电子枪收到场同步信号时就从屏幕的最右下角移到最左上角。在这里我们需要用这个信号来控制采集像素的时序。

复合消隐信号：在图像换行和换场时电子枪回扫时不发射电子。即收到复合同步信号后，电子枪要

换位置时是不能发射电子束的，这时候就由这个信号来消隐。在这里我们完全不用理会这个信号。

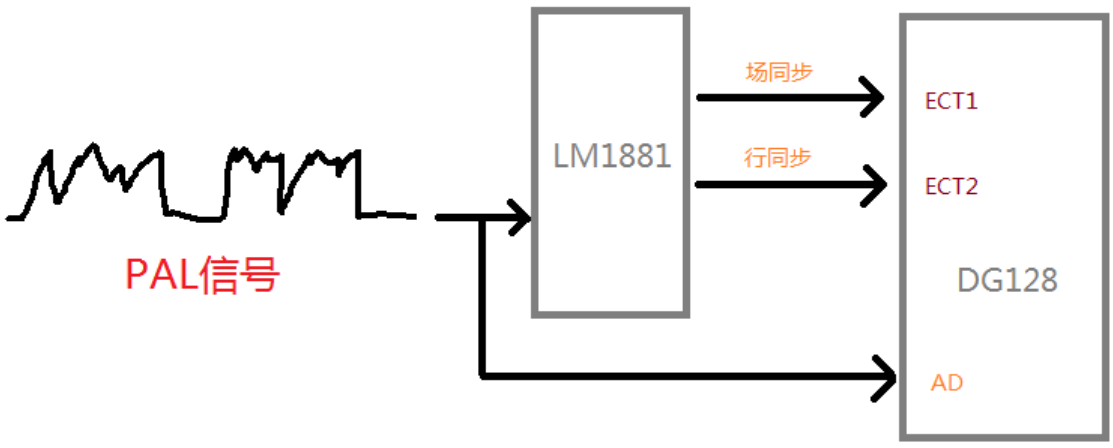
由于人眼看到的图像大于等于 24Hz 时人才不会觉得图像闪烁，所以 PAL 制式输出的图像是 25Hz，即每秒钟有 25 幅画面，说的专业点就是每秒 25 帧，其中每一帧有 625 行。但由于在早期电子技术还不发达时，电源不稳定，容易对电视信号进行干扰，而交流电源是 50Hz 所以，为了和电网兼容，同时由于 25Hz 时图像不稳定，所以后来工程师们把一副图像分成两场显示，对于一幅画面，一共有 625 行，但是电子枪先扫描奇数场 1，3，5.....，然后再扫描 2,4,6.....，所以这样的话，一副图像就变成了隔行扫描，每秒钟就有 50 场了。其中具体的细节请参考这个网站：电视原理与系统

http://courseware.ecnudec.com/zsb/zjx/zjx09/zjx090000.htm

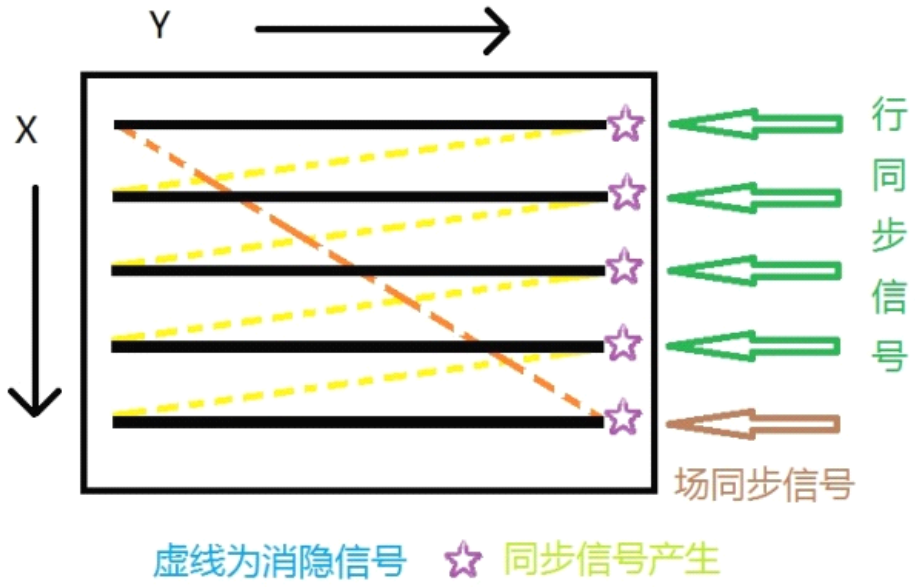
只用看前面的黑白全电视信号和 PAL 制式就可以了（当然如果感兴趣可以全部看完）。

四、 采集流程详解

通过上面的内容如果你对PAL制式信号了然于心，那么就可以开始图像的采集了，PAL输出的信号有复合同步信号，复合消隐信号和视频信号。那么我们首先就是要从这三种信号中分理出复合同步信号，复合消隐信号和视频信号，以便我们对AD采样到的值进行存储，从而形成一幅画面。具体如何分离，我们使用的是LM1881视频同步分离器件，具体的硬件连接请参看论坛内相关文章(www.znczz.com论坛里有介绍LM1881的文章，自己搜吧，我不重复了)，这里只给一个原理图。



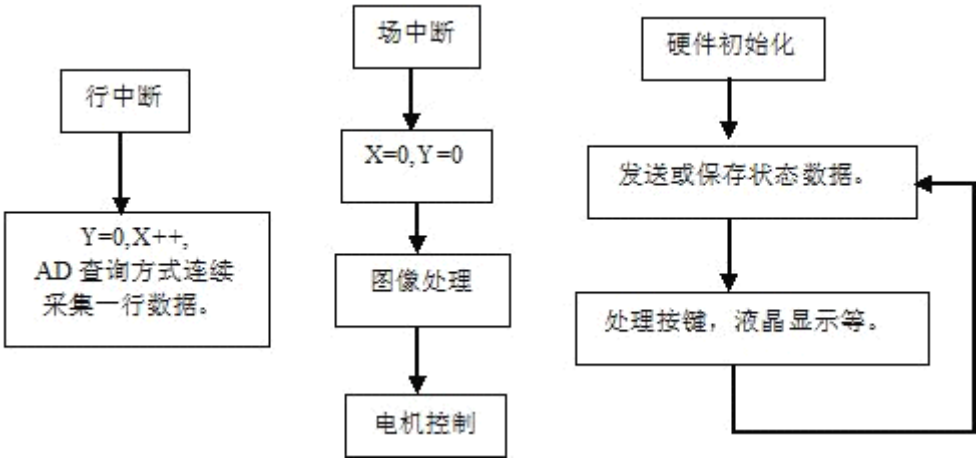
分离出行场同步，奇偶场信号后，就把他们接到单片机的外部中断口，产生中断，在中断服务程序中对 AD 采集到的数据进行图像存储，从而形成一个二维数组的数字图像。行场信号用来改变存储图像的二维数组的下标，当场信号到来时，就表明一幅图像存储完毕，要开始新一副图像了。



下面就说说图像采集编程方案，我使用的方案是在行中断中读取 AD 采样的灰度值，在场同步中交换图像采集和处理缓存指针，并对图像进行处理，然后控制小车，在主函数中只有初始化和键盘扫描和串

口输出函数。这样做效率比较高，而且可以把调试和图像采集处理分开，编程起来比较方便。

假设存储图想的数组是 image[x][y]，采集处理流程如下：



在行中断中的示意代码如下（当然可以使用定时器定时 AD 采集）：

```
wait();          //等待消隐信号结束。
y=0;
for(i=0;i<一行采样点数；i++)
{
```

```
    wait();//这个时间根据一行中要采样的点数确定。
    image[x][y++]=ADResult;
}
    x++;
```

在场中断中的示意代码如下：

```
ImageProcess();          //图像处理
AutoCntrol();            //控制算法及电机控制。
x=0;
y=0;
```

以上代码并不代表是最优的写法，仅供参考。

大家遇到的还有一个很棘手的问题可能是AD采样频率该设置多大呢？建议大家先通过PLL超频，然后把AD时钟频率设置的高点才行，板子做得好的话可以超到48M。

五、 部分源代码讲解

```
void vPLLInit(void)//锁相环初始化
{                                  //BUS-CLOCK=PLL-CLOCK/2=32M
   REFDV = 1;  // set the REFDV register 16M*2*(3+1)/(1+1)=64M
   SYNR =3;     // set the SYNR register to give us a 64 MHz PLL-clock.
    asm nop     // nops required for PLL stability.
    asm nop
    asm nop
    asm nop
   while ((CRGFLG&0x08)==0); // wait here till the PLL is locked.
   CLKSEL|=0x80;               // switch the bus clock to the PLL.
}
```

设置总线时钟为 32M

```
void vECTInit(void)//定时器初始化
{
 TIOS =0x00;     //设为输入捕捉
 TSCR1=0x80;     //定时器使能
 TSCR2=0x83;     //允许定时器溢出中断，定时器时钟 32M/(2^3)=4M
 TCTL4=0xAA;     //触发电平:下降沿
 TIE  =0x07;     //开中断
 TFLG1=0xFF;     //清除中断标志
}
```

输入捕捉的 1，2 通道接行场中断。

```
void vADInit(void)//AD 转换初始化程序
{
//ATD1 设置
//上电，标志位快速清零，忽略外部触发，执行一次停止，中断禁止。
 ATD1CTL2  = (ATD1CTL2_AFFC_MASK | ATD1CTL2_ADPU_MASK);

//转换序列长度为 1，FIFO 模式，Freeze 模式下继续转换。|ATD0CTL3_FIFO_MASK
 ATD1CTL3  = (ATD1CTL3_S1C_MASK);

//8 位精度，2AD 采样周期，采样长度 8。
//ATDClock=[BusClock*0.5]/[PRS+1]  ; PRS=15, divider=32
 ATD1CTL4  =(ATD1CTL4_SRES8_MASK|ATD1CTL4_PRS0_MASK);

// 右对齐无符号，扫描模式连续采样，单通道采样 // 多通道采样
|ATD0CTL5_MULT_MASK。
 ATD1CTL5  = (ATD1CTL5_DJM_MASK|ATD1CTL5_SCAN_MASK);

//禁止数字输入缓冲
ATD1DIEN=0x00;
}
ATD1 的 0 通道用于 AD 转换
```

```
//当前采样图像的行和列。
unsigned int ui_SampleRow=0,ui_SampleColumn=0;

//图像数据缓存
unsigned char uca_Buffer1[IMAGE_ROW][IMAGE_COLUMN];
unsigned char uca_Buffer2[IMAGE_ROW][IMAGE_COLUMN];

//指向当前采集数据采样缓存首地址的指针
unsigned char *puca_BufferSample=&uca_Buffer1[0][0];
//指向当前处理数据采样缓存首地址的指针
unsigned char *puca_BufferProcess=&uca_Buffer2[0][0];

//用于图像采集和处理交换缓存。（注意：在每次交换指针后保证 puca_BufferTemp 与
puca_BufferSample 相同）
unsigned char *puca_BufferTemp=&uca_Buffer1[0][0];
```

```
#pragma CODE_SEG NON_BANKED
//输入捕捉 2 通道中断函数，行同步 ,用于数据采集。
void interrupt 10 vIC2ISR(void)
{
 unsigned char ucTemp;
 unsigned char *pucTemp;
 TFLG1_C2F=1;

 if(ui_SampleRow>=SAMP_ROW_START&&ui_SampleRow<SAMP_ROW_MAX)
 {
   if(ui_SampleRow%SAMP_ROW_SEP==0)
   {
    for(ui_SampleColumn=0;ui_SampleColumn<SAMP_COL_MAX;ui_SampleColumn++)
    {
     while(!ATD1STAT1_CCF0);
     if(ui_SampleColumn>=SAMP_COL_START)
     {
       if(ui_SampleColumn%SAMP_COL_SEP==0)
       {
        pucTemp=puca_BufferSample
                                                          +((ui_SampleRow-
SAMP_ROW_START)/SAMP_ROW_SEP)*IMAGE_COLUMN
          +(ui_SampleColumn-SAMP_COL_START)/SAMP_COL_SEP;
        *pucTemp=ATD1DR0L;
       }
     }
    }
   }
 }
 ucTemp=ATD1DR0L;
 ui_SampleRow++;                //采样行坐标加一。
}
```

```
//输入捕捉 1 通道中断函数，场同步，交换缓存以及图像处理和模型车控制。
void interrupt 9 vIC1ISR(void)
{
 TFLG1_C1F=1;
 ui_SampleRow=0;                //把采样行坐标清零。
 ui_SampleColumn=0;

//交换图像采集和处理缓存
 puca_BufferSample=puca_BufferProcess;
 puca_BufferProcess=puca_BufferTemp;
 puca_BufferTemp=puca_BufferSample;

//系统时间加一。
 ul_SystemTime+=1;

//开中断，允许行信号中断进行采样。
 EnableInterrupts;

 if(uc_CarState==STATE_START)
 {
//    PORTB_BIT1=1;
  //分析图像，获取路径参数，根据路径参数控制模型车。。
    vImageProcess();

    //根据路径参数控制模型车。
    vAutoControl();
//    PORTB_BIT1=0;
 }
}
```