# mini2440裸机试炼之--ADC\_W1\_LED

## 实现功能

ADC项目需要实现的功能：通过开发板上的滑动变阻器控制的模拟信号

经过A/D转换器转换为数字信号用uart\_printf()打印在串口上。并且

滑动变阻器同时控制指示灯led的闪烁速度(时长)。

S3C2440ADC的主要特性如下：

●分辨率：10位

●最大转换速率：500KSPS

●微分线性度误差：±1.0 LSB

●积分线性度误差：±2.0 LSB

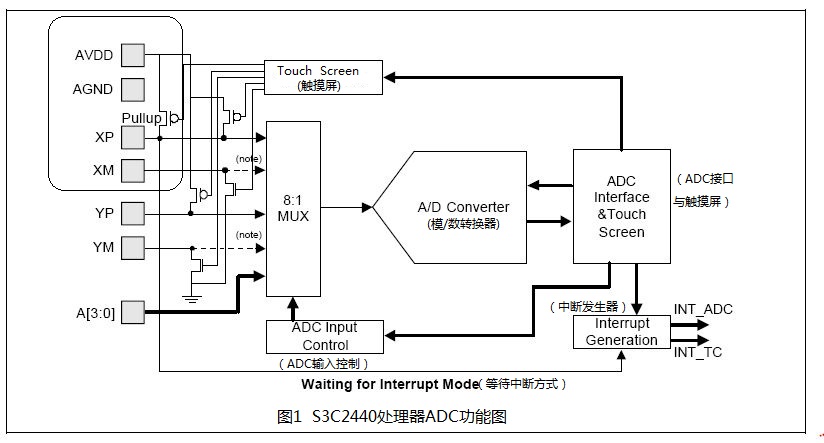
●供电电压：3.3V

●模拟输入电压范围：0~3.3V

●模拟输出数值范围：0~1023

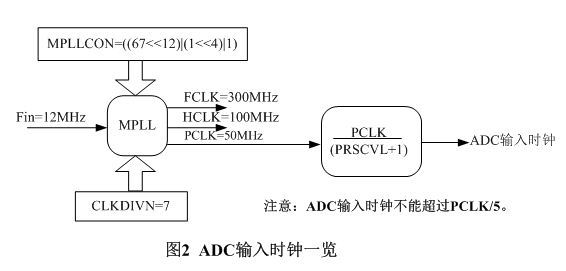
## ADC原理

ADC是一种将模拟信号转化为数字信号的方法，一般要经过采样、保持、量化、编码4个步骤。在实际电路中，有些过程是合并进行的，如采样和保持，量化和编码在转换过程中时同时实现的。由奈奎特采样定理可知，当采样频率大于模拟信号中最高频率的2倍时，采样值才能不失真地反映原来模拟信号。



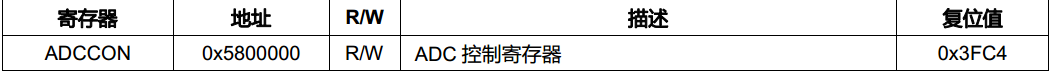
从图1可以看出，ADC共有8路模拟输入，其中XP、XM、YP和YM是触摸屏使用的4路，剩下的4路模拟输入A[3:0]可以用于一般的ADC输入通道（AIN0、AIN1、AIN2和AIN3）。当触摸屏引脚（YM、YP、XM和XP）为禁止时，这些端口可以被用于ADC的模拟输入端口（AIN4、AIN5、AIN6和AIN7）

对于S3C2440处理器，ADC输入时钟是由PCLK分频得到的，如图2所示



## ADC寄存器

### ADCCON部分



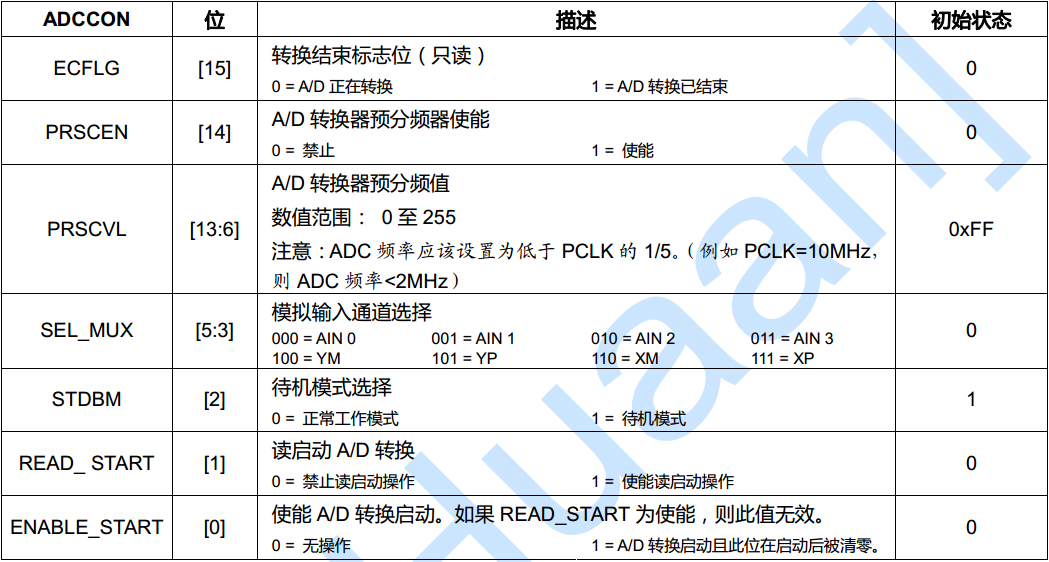


图3

根据 图3，ADCCON控制寄存器

ECFLG 是转换结束标志位，当转换结束时，ECFLG值为1，进入转换状态为0；

PRSCEN是使能预分频控制位 ；

PRSCVL 是预分频值放置位；0~255

■当PRSCEN为1时，设置PRSCVL的值将会决定AD转换的时间。ADC使用的时钟是系统的PCLK，由于main函数设置mpll为400MHZ,所以PCLK = 50MHz，设置PRSCVL的值为49，那么：

A/D 转换的频率 = 50MHz / （49+1） = 1 MHz

A/D 转换的时间 = 1 /（1MHz / 5cycles）=5 uS

由此可知当PRSCVL的值越小，A/D 转换的时间越短，A/D 转换的频率也就越大。

注意：当从待机模式中变换到正常工作模式时，ADC的预分频器必须在最后的3个ADC时钟前使能？？？？

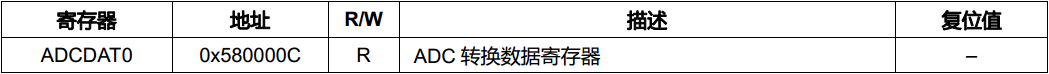
SEL\_MUX是模拟输入通道选择控制器

READ\_ START要使能A/D转换启动，就必须禁止READ\_ START

ENABLE\_START 使能A/D转换控制位

### ADCDAT0

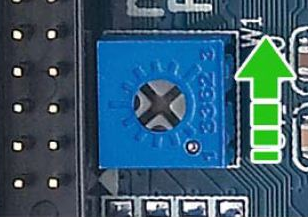
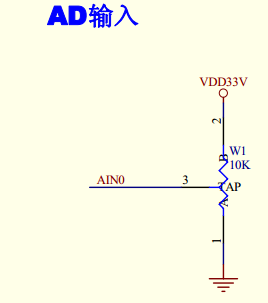
由于只是通过AIN0模拟信号输入。并没使用触摸屏功能，所以，ADCDAT0只考虑普通ADC转换数值（模拟信号转换的数字数值存于XPDATA）





## W1可调电位器

mini2440上有一个可调电位器，如下图所示。电位器的中间抽头部分接在ADC输入通道AIN0上，当电位器滑动头位于最下端时，AIN0引脚电压为0V；当电位器滑动头位于最上端时，AIN0引脚电压为3.3V；当电位器上、下滑动时，AIN0引脚的电压值会在0~3.3V之间变换。由此可知，W1是通过AIN0模拟输入通道间模拟信号传给ADC，再进行转换的。



## 代码部分

|  |
| --- |
| #include "2440addr.h" //引脚宏定义  #include "def.h" // U8 U32宏定义  #include "2440lib.h" //使用Uart\_Printf,Dalay声明，Uart\_Printf定义在2440lib.c文件  /\*  当PRSCEN为1时，设置PRSCVL的值将会决定AD转换的时间。ADC使用的时钟是系统的PCLK  ，由于main函数设置mpll为400MHZ,所以PCLK = 50MHz，设置PRSCVL的值为49，那么：  A/D 转换的频率 = 50MHz / （49+1） = 1 MHz  A/D 转换的时间 = 1 /（1MHz / 5cycles）=5 uS  \*/  /\*\*\*\*\*\*\*\*ADC\*\*\*\*\*\*\*\*/  #define PRSC\_EN 0x1 //允许预分频  #define PRSCVL 0x19 //预分频值  #define SEL\_MUX ~0x7 //模拟通道选择AIN0通道  #define STDBM 0x0 //正常工作模式  #define READ\_START 0x0 //读数时不进行A/D转换  #define ADC\_STAR 0x1 //ADC开启  /\*\*\*\*\*\*\*\*LED\*\*\*\*\*\*\*\*\*/  #define LED1\_out (1<<(5\*2))  #define LED2\_out (1<<(6\*2))  #define LED3\_out (1<<(7\*2))  #define LED4\_out (1<<(8\*2))  #define LED4\_ON (0x0E<<8) //LED1+LED2+LED3+LED4 ON  #define LED\_OFF (0x0F<<5)  void ADC\_display(){  Uart\_Printf("\n");  Uart\_Printf("\n");  Uart\_Printf("\n");  Uart\_Printf("---------HELLO ADC\n");  Uart\_Printf("PRSC\_EN: %5x 允许预分频\n",PRSC\_EN);  Uart\_Printf("PRSCVL: %5x 预分频值\n",PRSCVL);  Uart\_Printf("SEL\_MUX: ~0x7 模拟通道选择AIN0通道\n",SEL\_MUX);  Uart\_Printf("STDBM: %5x 正常工作模式\n",STDBM);  Uart\_Printf("READ\_START:%4x 读数时不进行A/D转换\n",READ\_START);  Uart\_Printf("ADC\_STAR: %4x ADC开启\n",ADC\_STAR);  Uart\_Printf("变阻器控制的模拟信号经过A/D转换器转换为\n");  Uart\_Printf("数字信号用串口显示,并且滑动变阻器同时控\n");  Uart\_Printf("制指示灯led的闪烁速度(时长)\n");  Uart\_Printf("#########################################\n\n");  }  //初始化AD控制寄存器  void ADC\_INIT(){  rADCCON&=(~((0x1<<14)|(0xff<<6)|(0x7<<3)|(0x1<<2)|(0x1<<1)|(0x1<<0)));  rADCCON=(PRSC\_EN<<14)|(PRSCVL<<6)|(SEL\_MUX<<3)|(STDBM<<2)|(READ\_START<<1);  rADCCON|=(ADC\_STAR<<0); //开启ADC（开启ADC后，此位会自动清零）  }  //初始化LED,四个LED输出  void LED\_INIT(){  rGPBCON=LED1\_out|LED2\_out|LED3\_out|LED4\_out;  }  //LED函数  void W1\_LED(int get\_data0){  rGPBDAT= LED4\_ON; //点亮四个LED  Delay(get\_data0); //ADC后十位时延函数  rGPBDAT= LED\_OFF; //熄灭LED  }  //串口打印变阻器模拟信号数值  void W1\_ADC(int \*read\_xpdata){  ADC\_INIT(); //初始化AD控制寄存器  /\*\*\*\*\*串口打印ADC数据函数\*\*\*\*/  while(rADCCON & 0x1); //因为成功启动A/D转换后，该位会  //自动清零，因此在这里检查ADC是否真正启动  while(!(rADCCON & (1<<15))); //使用查询方式等待ADC转换结束  \*read\_xpdata=(int)(rADCDAT0&0x3ff); //读取ADCDAT0后十位  Uart\_Printf("\n HELLO ADC# W1值为： %04d ",\*read\_xpdata);//串口输出ADCDAT0后十位值  Delay(500);  }  //子main函数  void ADC\_read\_W1(){    int read\_xpdata0;  ADC\_display();  LED\_INIT(); //初始化LED    while(1)  {  /\*串口打印变阻器模拟信号数值\*/  W1\_ADC( &read\_xpdata0 );//传递read\_xpdata0地址，改变read\_xpdata0的值来给W1\_LED函数使用    /\*\*\*\*\*LED点亮函数\*\*\*\*\*/  W1\_LED(read\_xpdata0); //LED函数    }  } |

◆串口输出截图

