关于Flash_UART的特别说明

最近我添加了往Teyvant_1.4.1中添加了存储两位的温度|电压数据到flash闪存中去,同时串口发送指令"55"可以获得存入flash闪存中的所有数据,串口发送"100"可以擦除flash闪存中的所有温度/电压数据后,因为采用了不同的引脚配置,所以在这个地方予以说明

USB-TTL转接器

交易成功

超优电子商行

STC单片机烧录器 下载器编... ¥6.30 x1

外壳颜色随机发货

7天无理由退货

现货，11月18日08:51前发货

卖了换钱 申请售后 加入购物车

实付款 ¥9.80

收货信息: 芙卡洛斯，86-15282149533，四川省 成都市 成华区 猛追湾街道 成华区建设北路电子科技大学沙河校区校内宿舍

订单编号: 2019326376585990297 | 复制

交易快照 发生交易争议时，可作为判断依据

查看更多

查看更多

订单服务保障中

7天无理由退货

查看服务

更多

评价

加入购物车

再买一单

晚上9:33

¥6.3

淘金币可抵0.18元 >

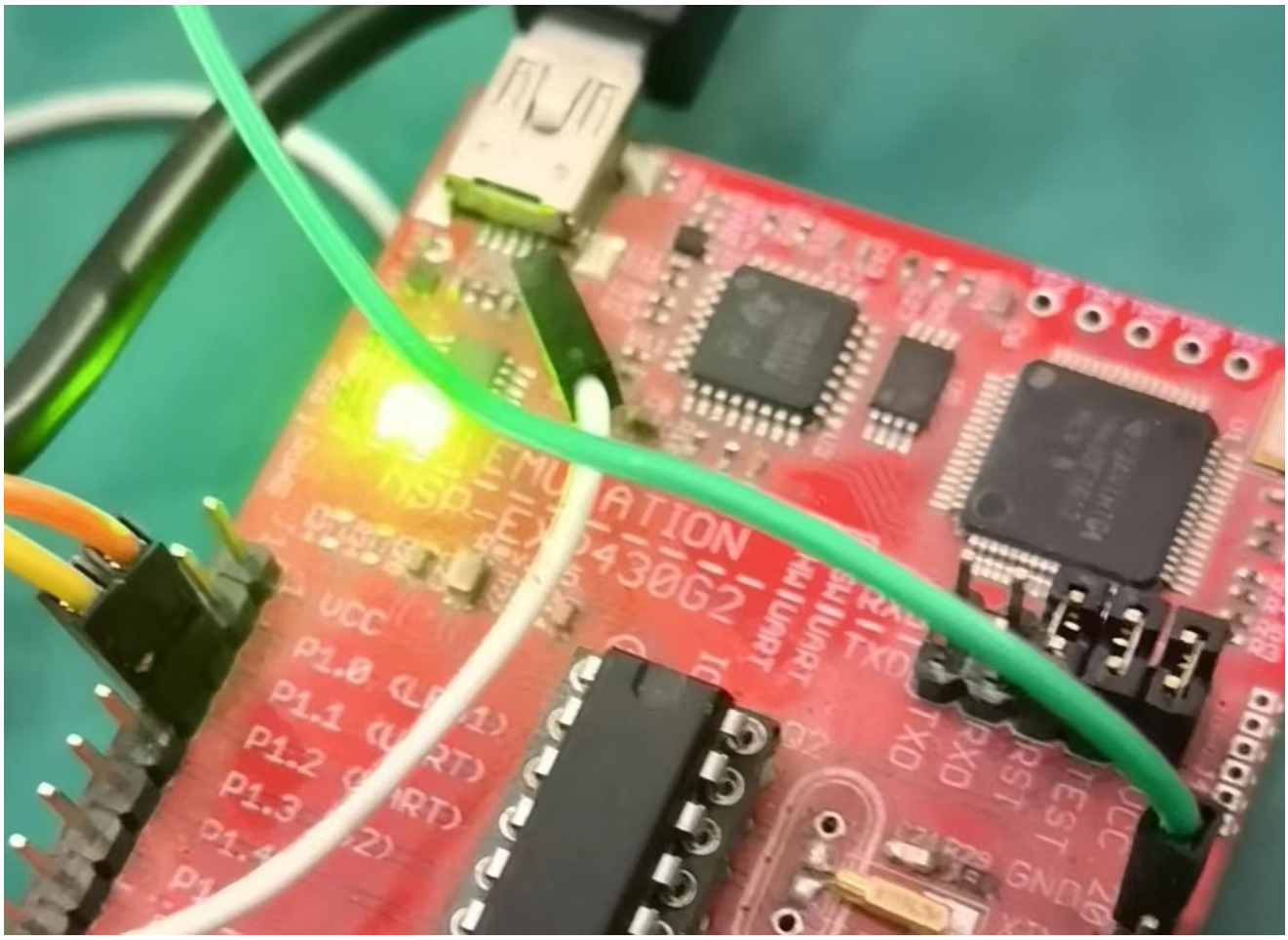
STC单片机烧录器 下载器编程器烧写器USB转TTL自动免冷启动全自动

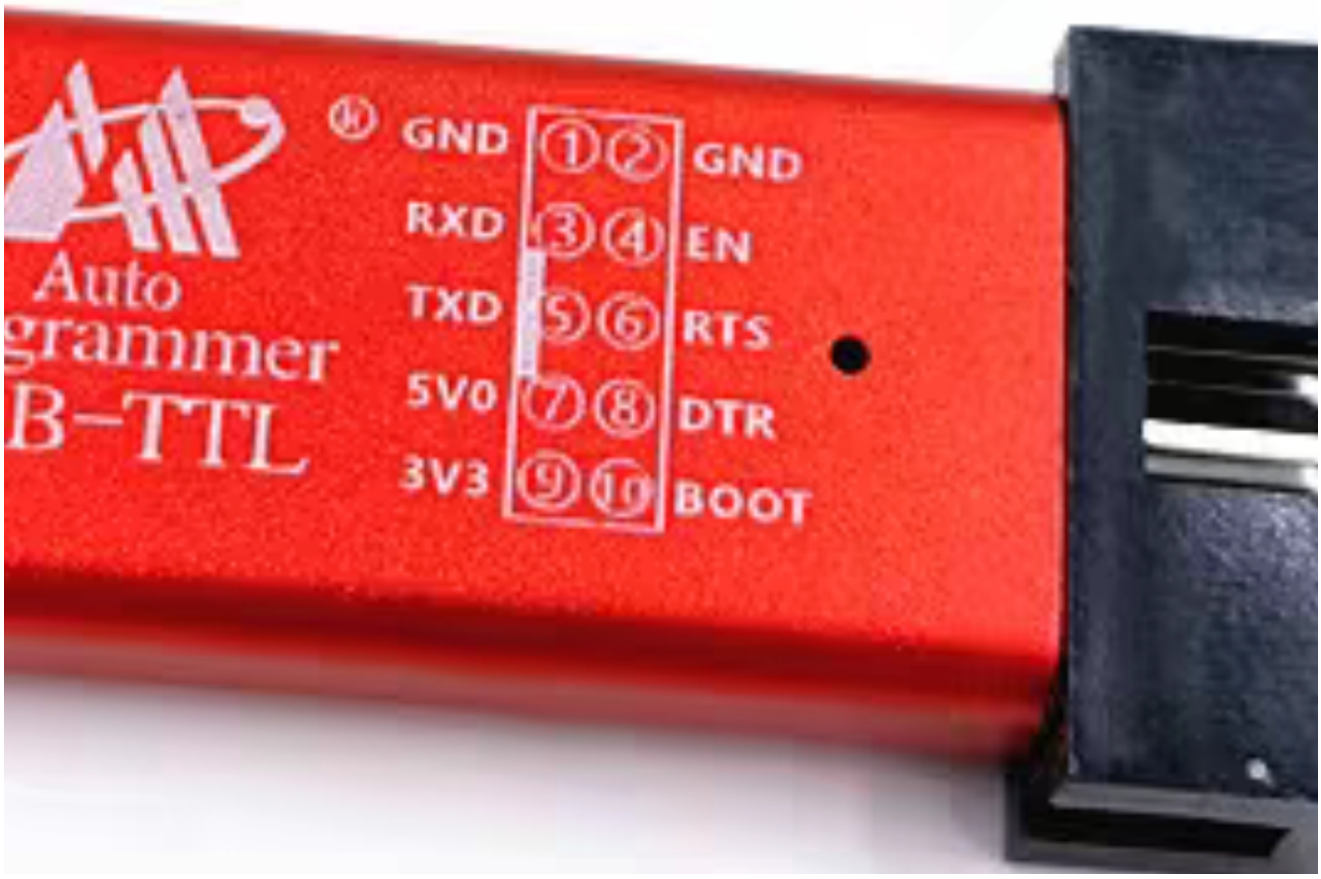
已售 100+

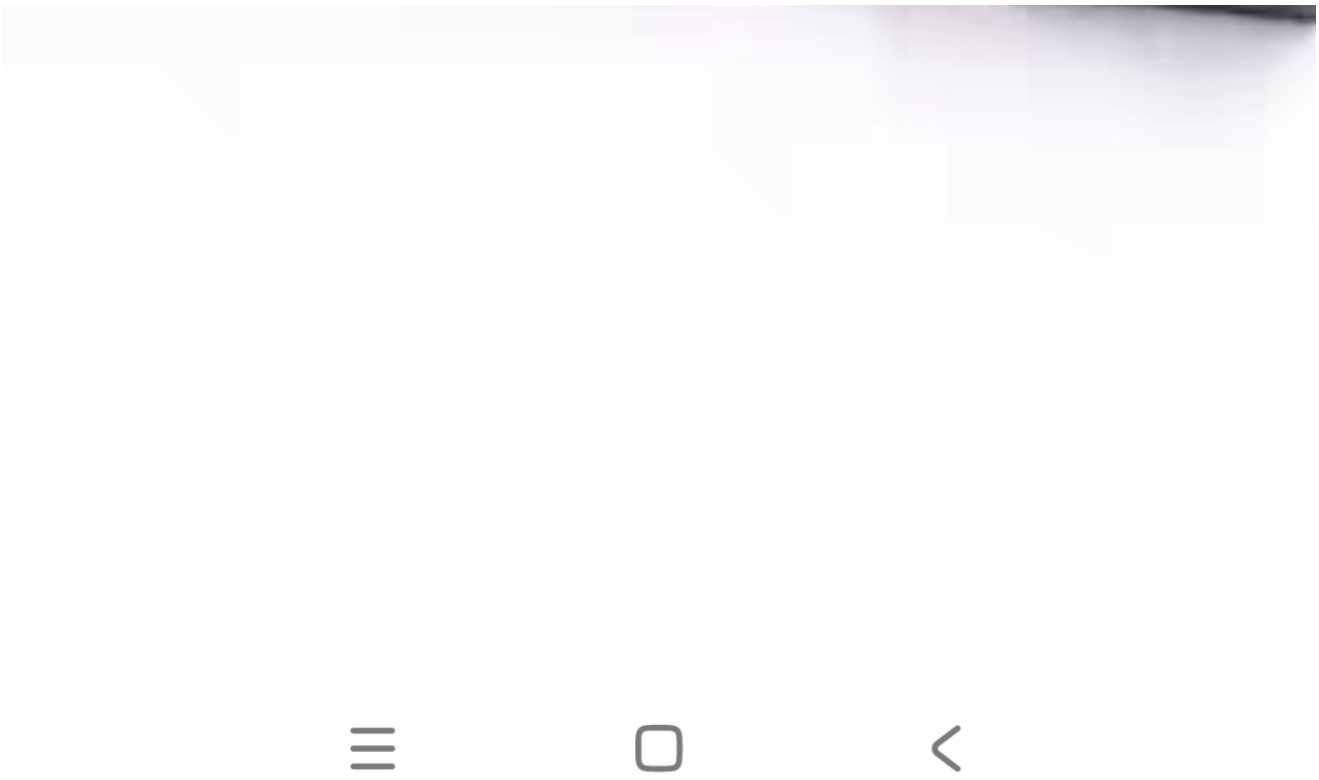
浙江绍兴 | 快递: 3.50

这个是我们要用的外设
在淘宝上差不多就是不到10块的价格,直接搜单片机烧写器,仿真器,下载器这样的字眼就好了,虽然我们看到上面写的都是什么51单片机,stm32这样的字眼,但其实我们用这个也没有问题的,我们只需要最基础的下载和调试功能就好了,
为什么要使用这个外设呢?
毕竟我搞了一个这样的外部的东西还要用杜邦线配置引脚,这么麻烦,是因为光用usb的那根线出现有问题,而且我解决不了,在不使用现在的解决方案的情况下 ,为了解决我们程序通信的问题,之前我想用,就直接拿usb的板子的那根线来弄,RX,TX的配置无法解决,所以才要用这个东西,这个东西的好处就是

引脚的配置模式


MSP-EXP430G2
UCC
P1.0
P1.1
P1.2
P1.3
TXD
RXD
RST
TEST
GND





如图所示
(为什么没有用自己的下载器的照片,是因为我在写这个文档的时候找不到我的下载器了,气死我了,)

GND接板子的GND
RX结板子的1.1
TX接板子的1.2
供电直接拿USB供电就好了

代码的解释和具体功能实现

```
//flash初始化
void Flash_Init(void)
{
    unsigned char i = 0;

    FCTL2 = FWKEY + FSSEL0 + FN1;             // MCLK/3 for Flash Timing
Generator
    Flash_ptr = (unsigned char*)0x1000;

    data_nums = 0;
    for(i = 0; i < 64; i++){
        flash_temp[i] = *(Flash_ptr + 1);
        flash_temp[i] <<= 8;
        flash_temp[i] += *Flash_ptr;
        Flash_ptr += 2;
        if(flash_temp[i] == 0xFFFF){//遇到0xFFFF说明没有数据，结束检索
            break;
```

```
        }else data_nums++;//有数据的话，数据条数加一
    }

}
```

```
# 初始化一个计数器i

"FCTL2寄存器配置模式"
"FWKEY--> 对FCTL1 FCTL2 FCTL3 写入许可"
"FSSEL0+FN1 --> Flash Timing Generator 时钟源MCLK 分频1/3"

"取值地址配置"
"初始化Flash_ptr指向Flash的0x1000"

"循环遍历"
"共64组数据,存储在flash存储器的后128位当中"
"每次读两位的数据,左移移位合并"
"地址向后移动两位"
"读到空数据后后跳出循环"
"没读到一组数据,数据条数计数器+1"
```

```
//存储数据

void Save_Data(unsigned int dat)
{
    unsigned char temp = 0;

    if(data_nums < 64){
        Flash_ptr = (unsigned char*)(0x1000 + data_nums*2);
        data_nums++;
        FCTL1 = FWKEY + WRT;                    // Set WRT bit for write
operation
        FCTL3 = FWKEY;                          // Clear Lock bit
        temp = dat;
        *Flash_ptr++ = temp;
        temp = dat >> 8;
        *Flash_ptr++ = temp;
        FCTL1 = FWKEY;                          // Clear WRT bit
        FCTL3 = FWKEY + LOCK;                   // Set LOCK bit
    }
}
```

```
//存储数据
"接受两位数据dat  "

"初始化数据拆分的暂存1位器变量 temp"
"写满判断 未使用的flash存储空间指针初始化"
"数据条数计数器+1 "


"Flash 控制寄存器FCTL1 FCTL3配置"
"FWKEY + WRT --> FCTL1 FCTL2 FCTL3 写入许可"
" FWKEY --> 清除锁定位"
"现在可以进行写入操作了"

"低八位存入临时变量"
"写入存储器指针后移"
"高8位存入临时变量"
"写入存储器指针后移"
""

"清除写入位"
"重置锁存"
```

```
//"55"输出数据

void Flash_Command(void)
{
    unsigned char i;
    unsigned int data_temp = 0;

    if(IFG2 & UCA0RXIFG){//收到串口发送来的命令
        data_rxed = UCA0RXBUF;
        if(data_rxed == 0xAA){//擦除数据
            data_nums = 0;
            for(i = 0; i < 64; i++){
                flash_temp[i] = 0xFFFF;
            }
            Flash_ptr = (unsigned char*)0x1000;
            FCTL1 = FWKEY + ERASE;// Set Erase bit
            FCTL3 = FWKEY;// Clear Lock bit
            *Flash_ptr = 0;// Dummy write to erase Flash segment
            FCTL1 = FWKEY;                          // Clear WRT bit
            FCTL3 = FWKEY + LOCK;                   // Set LOCK bit



                        Flash_ptr = (unsigned char*)0x1040;
```

```
            FCTL1 = FWKEY + ERASE;// Set Erase bit
            FCTL3 = FWKEY;// Clear Lock bit
            *Flash_ptr = 0;// Dummy write to erase Flash segment
            FCTL1 = FWKEY;                            // Clear WRT bit
            FCTL3 = FWKEY + LOCK;                     // Set LOCK bit
            UARTSendString("存储器已清空\r\n",16);
        }else if(data_rxed == 0x55){//读取存储器中的数据
            if(data_nums){
                Flash_ptr = (unsigned char*)0x1000;
                for(i = 0; i < data_nums; i++){
                    data_temp = *(Flash_ptr+1);
                    data_temp <<= 8;
                    data_temp += (*Flash_ptr);
                    Flash_ptr += 2;
                    PrintNumber(data_temp);
                }
            }else{
                UARTSendString("存储器中无数据\r\n",18);
            }
        }
    }
}
```

```
"初始化计数器,初始化2位数据暂存变量"

"中断是否有数据接受"
"读取缓冲区数据"

"读取到100,16进制AA,擦除全部数据"
"初始计数器,flash指针"
"擦除许可,清除锁存,虚拟擦除,清除写入,重制锁存"
"串口返回擦除完毕报告"


"读取到55,返回全部数据"
"初始计数器,flash指针"
"分位取得,合并数据,返回数据,直到取完"
"串口返回读取完毕报告"
```

```
/*主函数逻辑*/

int main(void)
{
```

```
    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer


        P1DIR |= BIT6;
        P1OUT &= ~BIT6;


        P1DIR &= ~BIT3;
        /* 使能P1.3口的上拉电阻 */
        P1REN |= BIT3;
        P1OUT |= BIT3;
          /* 打开P1.3口的中断 设定为下降沿触发 */ */
        P1IE |= BIT3;
        P1IES |= BIT3;
        /* 清除中断标志位 */
        P1IFG &= ~BIT3;
        /* 初始化Flash读写相关程序段 */
        /* 打开全局中断 */
        __bis_SR_register(GIE);

        InitSystemClock();
        InitUART();
        InitADC();

        while(1)
         {
             if (adcMode == 1)
             {
                 Mea_V(); // 启动ADC_V模式
             }
             else if (adcMode == 2)
             {
                 Mea_T();// 启动ADC_T模式
             }
             else if(adcMode == 0)//非AD转换模式下可以用串口读取和擦除Flash中的数据
             {
                 Flash_Command();
             }
             // 程序的其他部分
             __delay_cycles(300000);
         }
    return 0;
}
```

```
"初始化1.6 1.3-->按键控制.中断"
"初始化时钟,UART,ADC"

"模式选择"
```

```
"Mea_T"
"Mea_V"
"STOP--extends-->AA-->clear-->flash"
"             -->55-->output all dat in flash"
```