HMI 高级应用与特殊指令详解

本文档主要讲述一些高级应用，实现一些特殊功能，如果您的项目用不上，可以忽律此文档。

一.串口数据解析模式篇之主动解析模式应用详解

屏幕上电默认是被动解析模式，所有串口指令需按照指令集中的格式来对屏幕进行操作，假如你需要自定义协议，不按照指令集格式来发串口数据给屏幕，而使用你自定义的格式，那么就需要把屏幕配置为主动解析模式。要使用此功能，请务必确保你有以下 2 点基础：

1.明白什么叫 HEX,什么叫 String,什么叫 ASCII,分别什么关系，怎么转换。

2.明白单字节数值，双字节数值，四字节数值，分别有什么区别，它们在内存中是什么样的储存方式，明白什么叫小端模式，什么叫大端模式。明白低位在前是什么含义。

如果以上 2 点你都比较明白，那么请继续往下看，否则强烈建议不要再继续往下看了，因为大多数的项目是用不上这个功能的，使用默认的被动解析模式就可以了，没必要配置为下面的主动解析模式。

此篇幅涉及到以下几个内容:

1. 串口数据解析模式系统变量:recmod
2. 串口缓冲区数据大小系统变量:usize
3. 串口缓冲区数据组:u[index]
4. 串口缓冲区数据拷贝指令:ucopy
5. 串口数据解析模式退出密码

1.串口数据解析模式系统变量 recmod(0 为被动解析模式，**1 为主动解析模式)**

屏幕上电 recmod 为 0，即被动解析模式，在此模式下，外部设备按照标准指令集的指令格式发送串口指令给屏幕执行；如果你将 recmod 设置为 1(可以在上电默认页的初始化事件中写上 recmod=1 即可)，那么屏幕进入主动解析模式，然后所有的串口指令都不会被执行(注意：是串口指令不会被执行，上位软件编辑界面时写入事件中的固件指令是不会受影响的，依然正常执行)，所有的串口数据均存放在串口缓冲区中，等待您去主动读取。建议每次读完一个过程，清空一次串口缓冲区(清空指令为 code_c,标准指令集中有详细解释)，否则缓冲区溢出后就无法接收新数据。

2.串口缓冲区数据大小系统变量 usize(只能读取，不可设置)

读取此变量可以知道当前串口缓冲区已经缓存多少数据。

3.串口缓冲区数据组

串口缓冲区数据组的写法为 u[index]　(index 为序号)

例 1：从缓冲区中 0 位置开始获取一个 1 字节的数值，赋值给数字控件 n0，写法如下:

```
n0.val=u[0]
```

例 2：从缓冲区中 0 位置开始获取一个 2 字节的数值(小端模式，低位在前)，赋值给数字控件 n0，写法如下:

```
n0.val=u[1]
```

```
n0.val<<=8
n0.val+=u[0]
code_c          //读完以后清空缓冲区数据，防止溢出
```

例 3: 从缓冲区中 0 位置开始获取一个 4 字节的数值(小端模式，低位在前)，赋值给数字控件 n0，写法如下:

```
n0.val=u[3]
n0.val<<=8
n0.val+=u[2]
n0.val<<=8
n0.val+=u[1]
n0.val<<=8
n0.val+=u[0]
code_c          //读完以后清空缓冲区数据，防止溢出
```

难道对一个 4 字节的整型变量赋值缓冲区中的内容只能是分 4 次单字节赋值再加 3 次移位吗？ 当然不是！当然有更方便的做法，请继续往下看!

4.串口缓冲区数据拷贝指令 ucopy

格式: ucopy,att, srcstar, lenth, decstar

说明：将串口缓冲区中的数据拷贝到变量中(recmod=1 模式下有效)

att:目标变量名称

srcstar:串口缓冲区数据起始位

lenth:拷贝长度

decstar:目标变量数据起始位

此指令可以从串口缓冲区的指定位置连续拷贝指定数量的数据到目标变量(目标变量可以是字符串变量，可以是数值变量)。

例 1：从缓冲区中 0 位置开始获取一个 4 字节的数值(小端模式，低位在前)，赋值给数字控件 n0，写法如下:

```
ucopy n0.val,0,4,0
code_c          //读完以后清空缓冲区数据，防止溢出
```

温馨提示：每个数值变量系统都是按 4 字节分配内存，假如你使用 ucopy 从缓冲区获取小于 4 字节的数值，一定要注意剩余部分的字节数据处理，以免出现数据异常，操作方法请参考下面这个例子：

例 2: 从缓冲区中 0 位置开始获取一个 2 字节的数值(小端模式，低位在前)，赋值给数字控件 n0.val 的低 16 位，写法如下:

```
n0.val=0
ucopy n0.val,0,2,0          //如果要赋值给高 16 位，就写成 ucopy n0.val,0,2,2
code_c                      //读完以后清空缓冲区数据，防止溢出
```

解释：先将 n0.val 赋值为 0，目的在于确保 n0.val 的 4 个字节全置 0，然后再从缓冲区拷贝 2 个字节进来，否则会因为你只拷了 2 字节而导致 n0.val 原来剩下的 2 字节数据还在，然后导致 n0.val 最终的数值并不是你想要的数值。

例 3: 从缓冲区中 0 位置开始获取一个 10 字节的字符串，赋值给文本控件 t0，写法如下:

```
if(usize>=10)                   //确保缓冲区数据大小足够 20 个
```

```
{
    ucopy t0.txt,0,10,0
    code_c                    //读完以后清空缓冲区数据，防止溢出
}
```

重要提示：code_c 指令只是将 usize 置 0 和下一次串口数据写入缓冲区位置置 0，并不会把缓冲区里的所有内存数据设置为 0，所以在使用的时候你需要判断 usize 的大小后再使用缓冲区数据。

以上案例中使用的所有语句，指令，都是在上位编辑界面下，写入事件中的固件指令，串口一旦配置为主动解析模式，就不能再执行串口指令了，所以必须是由固件指令来操作读取串口数据，而不能是通过串口指令操作(否则就自相矛盾了)。

5.退出主动解析模式

常规的退出主动解析模式方法是在事件中写入 recmod=0 的固件指令，如果想通过串口数据来退出，串口发送 recmod=0 是肯定没有用的，可以通过发送一串退出密码来实现退出主动解析模式,退出密码为一串 24 字节的字符串+3 字节的结束符。

24 字节的字符串:

DRAKJHSUYDGBNCJHGJKSHBDN　(字符串数据，必须大写)

3 字节的结束符：

0xff 0xff 0xff　　(Hex 数据)

合计 27 字节

……本章节已完,谢谢收看!……

二.HMI 下载协议详解

本文讲述的 HMI 下载协议仅适用于希望自己制作下载程序或者希望单片机去控制 HMI 下载资源文件的用户，属于高级应用范畴，不属于 HMI 界面设计的范畴，因此需要有一定基础的用户才能操作。深圳市淘晶驰电子有限公司仅仅只对此协议做一个公布说明，不提供任何跟下载协议有关的技术支持，如果对串口操作不熟悉的朋友建议忽略此说明，请直接使用 USART HMI 软件进行下载即可，无需对此协议有任何了解。

下载步骤 1：联机操作

此步骤主要用来搜索 HMI 设备在哪个串口上，以及设备当前的波特率。如果这两个条件是已知的，那么可以不用做这个步骤，在你的程序中直接固定串口号和设备当前使用的波特率后直接跳到步骤 2 开始下载。

搜索方法：分别向电脑的每个串口分别用不同波特率发送一个联机指令:connect

设备收到联机指令后会返回联机数据，如果收到正确的联机数据，说明设备联机成功，至此，得到当前设备的串口号和当前使用的波特率.

联机指令发送说明：因为一直在循环发送指令，所以当屏幕在正确的波特率上收到数据时，数据的最前面肯定会有部分上一次错误的波特率下的错误数据，因此这个时候第一条指令肯定是会被当成错误指令的。所以每次发送的时候需要发两条指令，第一条发空指令（即单纯的三个 0XFF）,第二条才是 connect+3 个 0XFF

延时说明:每此尝试一次联机指令后需要等待数据返回的最短时间为(单位:ms): (1000000/尝试的波特率)+30

假如在 9600 波特率下尝试联机，需要等待返回的最短时间为:

1000000/9600+30=134ms

其他波特率以此类推

数据解释：

以 TJC4024T032_011R 设备为例，设备返回如下 8 组数据(每组数据逗号隔开)：

comok

1,101,TJC4024T032_011R,52,61488,D264B8204F0E1828,16777216

comok:握手回应

1:表示带触摸(0 是不带触摸)

101:设备内部预留数据

TJC4024T032_011R:设备型号

52:设备固件版本号

61488:设备主控芯片内部编码

D264B8204F0E1828:设备唯一序列号

16777216:设备 FLASH 大小(单位：字节)

下载步骤 2：开始下载

此时已经知道设备在哪个串口号上，也知道设备当前的波特率了，可以发送下载指令了。

第一步：发送指令 whmi-wri filesize,baud,res0

filesize:tft 文件的大小(单位：字节)

baud:强制下载使用的波特率

res0:预留数据，使用任意 ASCII 字符即可

假如需要下载的 tft 文件大小为 10000 字节,需要使用 115200 波特率下载，那么就发送指令：

whmi-wri 10000,115200,0

发送完此指令以后，需要修改电脑的波特率为刚才设置的强制波特率（如果当前波特率和强制下载波特率不一致的话）

第二步：下发 tft 文件的二进制数据

设备收到 whmi-wri 指令后在 250ms 左右后会返回一个 0x05 的数据（仅仅是一个字节，没有 3 个 0XFF 的结束符，波特率为刚才设置的强制下载波特率）,收到此数据后，可以开始下发 tft 文件的二进制数据,下发格式为每包下发 4096 字节,最后一包剩余多少就发多少，每包发送完成以后，需要等待屏幕返回响应信号，响应信号依然为一个单一字节的 0x05。

……本章节已完,谢谢收看!……