串口HMI指令集

注：

1．设备接收指令结束符为"0XFF 0XFF 0XFF"三个字节(HEX数据，不是字符串数据)。

2．所有指令名以及参数全部使用ASCII字符串数据，非HEX数据，便于阅读和调试。

3．所有指令名使用小写字母(此处仅仅指的是指令名称为小写，参数该大写的时候还是要大写)。

对象及系统操作指令汇总表(点击指令名称或参数会自动跳转到该指令的详解处)

序号	指令名	指令功能	参数
1	page	刷新页面	page pageid
2	ref	重绘控件	ref obj
3	click	激活控件按下/弹起事件	click obj,event
4	get	带格式获取变量值/常量值	get att
5	prints	从串口打印一个变量/常量	prints att,lenth
6	printh	从串口打印一个Hex	printh hex
7	vis	隐藏/显示控件	vis obj,state
8	tsw	控件触摸使能	tsw obj,state
9	randset	随机数范围设置	randset minval,maxval
10	add	往曲线控件添加数据	add objid,ch,val
11	cle	清除曲线控件中的数据	cle objid,ch
12	addt	曲线数据透传指令	addt objid,ch,qyt
13	doevents	转让系统控制权给屏幕刷新	无参数
14	sendme	发送当前页面ID号到串口	无参数
15	covx	变量类型转换	covx att1,att2,lenth,format
16	strlen	字符串变量字符长度测试	strlen att0,att1
17	btlen	字符串变量字节长度测试	btlen att0,att1
18	substr	字符串截取	substr att0,att1,star,lenth
19	spstr	字符串分割	spstr src,dec,key,index
20	touch_j	触摸校准	无参数
21	ref_stop	暂停屏幕刷新	无参数
22	ref_star	恢复屏幕刷新	无参数

序号	指令名	指令功能	参数
23	com_stop	暂停串口指令执行	无参数
24	com_star	恢复串口指令执行	无参数
25	code_c	清空串口指令缓冲区	无参数
26	rest	复位	无参数
27	wepo	写入一个变量到用户存储区	wepo att,add
28	repo	从用户存储区读数据到变量	repo att,add
29	wept	透传数据写入用户存储区	wept add,lenth
30	rept	从用户存储区透传数据到到串口	rept add,lenth
31	cfgpio	扩展IO模式配置	cfgpio id,state,obj
32	crcrest	复位crc初始值	crcrest crctype,initval
33	crcputs	crc校验一个变量/常量	crcputs att,length
34	crcputh	crc校验一组Hex	crcputh Hex
35	crcputu	crc校验一段串口缓冲区数据(recmod=1时有效)	crcputu star,length
36	setlayer	运行中改变控件图层顺序(仅X3,X5系列支持)	setlayer obj0,obj1
37	move	控件移动 (仅X3,X5系列支持)	move obj,starx,stary,endx,endy,first,time
38	play	音频播放 (仅X3,X5系列支持)	play ch,audio,loop
39	twfile	串口透传文件 (仅X3,X5系列支持)	twfile filepath,filesize
40	delfile	删除文件 (仅X3,X5系列支持)	delfile filepath
41	refile	重命名文件 (仅X3,X5系列支持)	refile srcfilepath,decfilepath
42	findfile	查找文件 (仅X3,X5系列支持)	findfile filepath,att
43	rdfile	透传读文件 (仅X3,X5系列支持)	rdfile filepath,addr,size,crc
44	newfile	创建文件 (仅X3,X5系列支持)	newfile filepath,size
45	newdir	创建文件夹 (仅X3,X5系列支持)	newdir dir
46	deldir	删除文件夹 (仅X3,X5系列支持)	deldir dir
47	redir	重命名文件夹 (仅X3,X5系列支持)	redir srcdir,decdir
48	finddir	查找文件夹 (仅X3,X5系列支持)	finddir dir,att

GUI 绘图指令汇总表(点击指令名称或参数会自动跳转到该指令的详解处)

注：GUI 绘图指令主要应用在如下场合：

当上位界面编辑软件无法实现您的某些特殊显示要求的时候，使用 GUI 指令自己绘图来实现自己想要的显示效

果。 大多数情况下其实是不需要使用这些绘图指令的， 大多数的应用都可以通过界面编辑软件的控件操作来实现。

序号	指令名	指令功能	参数
1	cls	清屏指令	cls color
2	pic	刷图指令	pic x,y,picid
3	picq	切图指令	picq x,y,w,h,picid
4	xpic	高级切图指令	xpic x,y,w,h,x0,y0,picid
5	xstr	写字指令	xstr x,y,w,h,fontid,pointcolor,backcolor,xcenter,ycenter,sta,string
6	fill	区域填充指令	fill x,y,w,h,color
7	line	画线指令	line x,y,x2,y2,color
8	draw	画矩形	draw x,y,x2,y2,color
9	cir	画空心圆	cir x,y,r,color
10	cirs	画实心圆	cirs x,y,r,color

对象及系统操作指令详解

1.page 刷新页面

```
page pageid
pageid:页面ID或页面名称
实例1:page 0    (刷新ID为0的页面)
实例2:page main  (刷新名称为main的页面)
```

备注：
1.设备上电自动刷新第0页。

2.也可以对系统变量dp赋值来实现跳转页面(如dp=0)，系统变量dp可以设置可以读取，具体请参看系统变量列表。
3.page 指令是跳转指令。写在page指令后面的代码将被忽略没机会执行。

2.ref 重绘控件

```
ref obj
obj:控件ID或控件名称
实例1:ref 1  (重绘ID为1的控件)
实例2:ref t0  (重绘名称为t0的控件)
```

备注：

如果一个控件被GUI指令画出来的内容遮挡或者被另外的控件遮挡之后需要再显示出来，就使用ref来重绘。

3.click 激活控件的按下/弹起事件

```
click obj,event
obj:控件ID或控件名称
event:事件序号:0为弹起，1为按下
click b0,1  (激活名称为b0的控件的按下事件)
click 2,0  (激活ID为2的控件的弹起事件)
```

备注：

控件的按下/弹起事件在屏幕上触摸的时候会自动激活，如果在没有触摸的情况下想要手动激活，就使用click指令即可。

4.get 带格式获取变量值/常量值

```
get att
att:变量名称
实例1:get t0.txt (返回控件t0的txt属性值)
实例2:get j0.val (返回控件j0的val属性值)
实例3:get “123”  (返回常量字符串”123”)
实例4:get 123  (返回常量数值：”123”)
```

备注：

1.使用get指令获取的变量为字符串类型时，返回的数据为0X70+字符串内码+结束符，如果是数值类型(如进度条的val属性)设备返回0X71+变量的4字节十六进制数据(int类型)+结束符。数值的存放模式为小端模式（即低位在前，高位在后）。

2.get指令可以由串口发送，也可以在上位软件编辑界写进用户代码中实现屏幕主动发送变量(主动发送的时候可以配合printh指令在前面加一段自定义标示来告诉单片机此变量是属于哪个控件的)。

3.get指令和print指令很类似,唯一的区别是get返回的数据带了起始标示符（0x70或0x71）和结束符(0xff 0xff 0xff)，而print没有。4..数据具体返回格式请查看本表格后面的”串口HMI设备返回数据格式"。

5.prints 从串口打印一个变量/常量

```
prints att,lenth
att:变量名称
lenth:发送长度(0为自动长度)
实例1: prints t0.txt,0 (发送控件t0的txt属性值,长度为实际长度)
实例2: prints j0.val,0 (发送控件j0的val属性值,默认长度为4字节整形数据，小端模式储存)
实例3: prints "123",0  (发送常量字符串"123"即：0x31 0x32 0x33)
实例4: prints 123,0    (发送常量数值：123 即: 0x7b 0x00 0x00 0x00)
实例5: prints 123,1    (发送常量数值：123的低1位数据 即: 0x7b)
```

备注：

1.使用prints发送的变量为字符串类型时，设备直接返回字符串内码，如果是数值类型(如进度条的val属性)设备直接返回变量的4字节整形数据(Hex数据，储存方式为小端模式，即低位在前)。

2.使用prints指令获取数据的时候，设备仅仅只发送数据内容，没有起始标示符，也没有结束符。

3.prints指令可以配合printh指令在前面加一段自定义标示来告诉单片机此变量是属于哪个控件的)。

4.prints指令和get指令很类似,区别是get发送的数据带了起始标示符（0x70或0x71）和结束符(0xff 0xff 0xff)，而prints没有,不过prints可以在后面继续用printh语 句来加任何自定义标识符。

6.printh 从串口打印一个Hex

```
printh hex
hex:需要发送的字符的16进制字符串表达式
实例:printh d0 a0(让设备发送0xd0 0xa0两个字节)
```

备注：

1.使用printh指令发送数据的时候，设备仅仅只发送指定的字符，不会发起始符，不会发空格,不会发结束符。

2.参数中每组字符间必须有且只能有一个空格隔开，16进制的字符串表达式大小写均支持。

3.printh只能发送常量,不能发送变量,变量需要使用prints指令。

7.vis 隐藏/显示控件

```
vis obj,state
obj:控件名称或控件ID
state:状态(0或1)
实例1:vis b0,0 (隐藏b0控件)
实例2:vis b0,1 (显示b0控件)
实例3: vis 1,0 (隐藏ID为1的控件)
实例4: vis 1,1 (显示ID为1的控件)
```

备注：

第一个参数 为255表示 当前页面所有控件，例:vis 255,0(隐藏当前页面所有控件) vis 255,1(显示当前页面所有控件)。

8.tsw 控件触摸使能

```
tsw obj,state
obj:控件名称或控件ID
state:状态(0或1)
实例1:tsw b0,0 (让名称为b0的控件触摸失效)
实例2:tsw b0,1 (让名称为b0的控件触摸有效)
实例3:tsw 1,0  (让ID为1的控件触摸失效)
实例4:tsw 1,1  (让ID为1的控件触摸有效)
```

备注：

第一个参数 为255表示 当前页面所有控件，例:tsw 255,0(当前页面所有控件触摸失效) tsw 255,1(当前页面所有控件触摸有效)。

9.randset 随机数范围设置

```
randset minval,maxval
minval:最小值
maxval:最大值
实例:ranset 1,100 (设置当前随机数产生范围为最小1，最大100)
```

备注：

1.使用随机数之前需要先使用randset指令设定一次随机数产生范围，如果不设置，默认是最小0，最大2147483647。设置完范围以后，每读取一次系统变量rand将会得到一个随机数。

2.使用randset指令每设定一次范围，将一直有效，直到重新上电或者设备复位才会恢复默认。

3.随机数设定范围的数据类型为int类型(即：最小-2147483648,最大2147483647)。

10.add 往曲线控件添加数据

```
add objid,ch,val
objid:曲线控件ID序号(此处必须是ID号，不支持使用控件名称)
ch:曲线控件通道号val:数据 (最大255，最小0)
实例1:add 1,0,30 (往ID为1的曲线控件的0通道添加数据30)
实例2:add 1,1,n0.val (往ID为1的曲线控件的1通道添加数据n0.val)
```

备注：

1.曲线数据只支持8位数据，最小0，最大255。

2.每个page页面最多支持4个曲线控件,每个曲线控件最多支持4个通道。可以连续发送数据，控件会自动平推显示数据.在发送数据的过程中也可以随时修改控件属性，比如随时修改各个通道的前景色或背景色。

11.cle 清除曲线控件中的数据

```
cle objid,ch
objid:曲线控件ID序号(此处必须是ID号，不支持使用控件名称)
ch:曲线控件通道号(255表示所有通道)
实例1:cle 1,0  (清除ID为1的曲线控件的0通道数据)
实例2:cle 1,255 (清除ID为1的曲线控件的所有通道数据)
```

备注：

1.通道号为255时表示清除此曲线控件内的所有通道数据。

12.addt 曲线数据透传指令

```
addt  objid,ch,qyt
objid: 曲线控件ID序号(此处必须是ID号，不支持使用控件名称)
ch:曲线控件中的通道号
qyt:本次透传数据的点数量
实例:addt 1,0,100  (ID为1的曲线控件进入数据透传模式，透传点数为100点)
```

备注：

1.曲线数据只支持8位数据，最小0，最大255。单次透传数据量最大1024字节

2.发完透传指令后，用户需要等待设备响应才能开始透传数据，设备收到透传指令后，准备透传初始化数据大概需要5ms左右(如果在透传指令执行前串口缓冲区还有很多别的指令，那时间会更长)，设备透传初始化准备好以后会发送一个透传就绪的数据给用户（0XFE+结束符），表示设备已经准备好，此时可以开始发送透传数据。透传数据为纯16进制数据，不再使用字符串，也不再需要结束符,设备收完指定的数据量以后，才会恢复指令接收状态。否则一直处于数据透传状态，透传数据完成以后,设备会发送结束标记给用户（0XFD+结束符）。

3.在指定的透传数量传输完成以前，曲线不会刷新，透传完毕之后会立即自动刷新。

13.doevents 转让系统控制权给屏幕刷新

```
doevents
实例: doevents (此指令不需要参数)
```

备注：

1.在一个较多指令的过程执行中，或者在一个较长时间的循环语句中，系统所有控制权被此过程全部占用，在过程结束之前，尽管相应的内存数据可以任意正常读写，但是屏幕不会刷新显示，加入doevents后可以转让控制权给屏幕刷新，执行doevents之后，屏幕会刷新所有被改变过的控件，刷新完之后，控制权交回当前过程继续执行。防止屏幕呈现假死的显示状态。

2.doevents多数情况下是配合while或for语句使用，使用方法请参看while或for语句的实例。

14.sendme 发送当前页面ID号到串口

```
sendme
实例1：sendme（此指令不需要参数）
```

备注:

设备收到此指令会立刻把当前页面的ID号发送到串口，如果想要每次刷新页面自动发送页面ID,请在页面的初始化事件里写上sendme语句即可。发送格式请参看本表格后面的"串口HMI设备返回数据格式"表格。

15.covx 变量类型转换

```
covx att1,att2,lenth,format
att1:源变量
att2:目标变量
lenth:字符串的长度(0为自动长度，非0为固定长度)
format:申明数值类型(0-数字;1-货币;2-Hex)
实例1:covx h0.val,t0.txt,0,0 (把滑块h0的val数值变量转换成10进制数字字符串并赋值给文本t0的txt变量,长度为自动)
实例2:covx t0.txt,h0.val,0,0 (把文本t0的txt十进制数字字符串变量转换为数值并赋值给滑块h0的val数值变量,长度为自动)
```

备注:

1.lenth始终表示的是字符串长度，数值转字符串的时候是目标变量的长度，字符串转数值的时候是源变量长度。

2.如果目标变量和源变量类型相同，转换失败。

16.strlen 字符串变量字符长度测试

```
strlen att0,att1
att0:需要测试的字符串变量
att1:把测试结果赋值给此变量
实例:strlen t0.txt,n0.val (把字符串变量t0.txt的实际字符长度赋值给n0.val)
```

备注:

1.strlen测试的是以字符为单位的长度，而btlen测试的是以字节为单位的长度,比如一个汉字用btlen测试出来的长度是2字节,用strlen测试出来的长度是1字符。

2.被测试的变量必须是字符串类型，写入的变量必须是数值类型，否则会报错。

17.btlen 字符串变量字节长度测试

```
btlen att0,att1
att0:需要测试的字符串变量
att1:把测试结果赋值给此变量
实例:btlen t0.txt,n0.val (把字符串变量t0.txt的实际字节长度赋值给n0.val)
```

备注:

1.btlen测试的是以字节为单位的长度，而strlen测试的是以字符为单位的长度，比如一个汉字用btlen测试出来的长度是2字节,用strlen测试出来的长度是1字符。

2.被测试的变量必须是字符串类型，写入的变量必须是数值类型，否则会报错。

18.substr 字符串截取

```
substr att0,att1,star,lenth
att0:源变量(必须是字符串变量)
att1:目标变量(必须是字符串变量)
star:在源变量中的字符起始位置lenth:截取字符串长度
```

实例:substr t0.txt,t1.txt,0,2（从t0.txt中的0位置开始截取2个字符赋值给t1.txt)

19.spstr 字符串分割

```
spstr src,dec,key,index
src:源变量(必须是字符串变量)
dec:目标变量(必须是字符串变量)
key:分隔符字符串(必须是字符串变量)
index:取第几份(在src字符串中用key字符串做分割后，取第index份字符内容赋值给dec变量)
实例：data0.txt的d字符内容为:aaaa^bbbb^cccc^dddd
执行命令：spstr data0.txt,t0.txt,"^",2
运行结果：t0.txt内容为：cccc
```

20.touch_j 触摸校准

```
touch_j
实例1：touch_j (进入触摸校准功能,此指令不需要参数)
```

备注：

所有设备出厂时已经校准过，一般情况下不需要使用此功能。

21.ref_stop 暂停屏幕刷新

```
ref_stop
实例:ref_stop (此指令不需要参数)
```

备注：

1.暂停屏幕刷新之后，所有语句会继续解析并执行，相应的属性赋值操作也会正常运行，但是屏幕上的控件不会刷新，修改任何控件的任何属性都不会自动刷新显示（但是属性已经被正常修改了）。直到设备收到恢复刷新指令(ref_star)后，被修改过的控件将会立刻刷新显示。

2.暂停刷新之后，即便使用ref指令也不会立刻刷新，直到执行ref_star指令的时候统才会统一刷新，但是所有的gui绘图指令(比如画点，划线，画圆等)是不受影响的，会立即显示。

22.ref_star 恢复屏幕刷新

```
touch_j
实例1：touch_j (进入触摸校准功能,此指令不需要参数)
```

备注：

此指令和ref_stop配合使用。

23.com_stop 暂停串口指令执行

```
com_stop
实例1:com_stop（此指令不需要参数)
```

备注：

1.暂停串口指令执行之后设备会继续接受指令，但是都不会执行，全部放在指令缓存区，直到收到”com_star”指令后，设备会从暂停时的指令开始到当前为止的所有指令全部执行。

2.使用指令暂停与恢复功能的时候，请评估您的设备的串口缓存区大小和指令缓存队列的最大数量是否足够支持你需要缓存的指令数目。这两项参数在你购买的设备规格书中的参数表中可以查询到。

24.com_star 恢复串口指令执行

```
com_star
实例1:com_star（此指令不需要参数)
```

备注：

1.设备收到此指令之后，将从暂停时的指令开始到当前为止的所有指令全部执行。

2.使用指令暂停与恢复功能的时候，请评估您的设备的串口缓存区大小和指令缓存队列的最大数量是否足够支持你需要缓存的指令数目。这两项参数在你购买的设备规格书中的参数表中可以查询到。

25.code_c 清空串口指令缓冲区中还没有执行的所有指令

```
code_c
实例1: code_c（此指令不需要参数)
```

立即清空串口指令缓冲区还没有执行的所有指令。

26.rest 复位

```
rest
实例:rest (此指令不需要参数)
```

27.wepo写入一个变量到用户存储区(EEPROM)

(带用户存储的硬件才支持)

```
wepo att,add
att:变量/常量
add: 用户存储区位置(从0开始)
实例1:wepo t0.txt,10（将t0.txt的内容写入用户存储区的第10位置,在储存区中的占用空间为t0.txt的最大设置值+1，即t0的txt-maxl属性表示的大小+1)
实例2:wepo “abcd”,10（将字符串“abcd”写入用户存储区的第10位置，在储存区中占用大小为5字节)
实例3:wepo 125,10（将数值125写入用户存储区的第10位置, 在储存区中占用大小为4字节)
```

备注:

1.写入内容为变量字符串的时候，在储存区中的占用空间为此变量的最大字符数+1；写入内容为常量字符串的时候，在储存区中的占用空间为此常量字符串的实际字符数+1。

2.写入内容为变量数值或常量数值的时候，在储存区中的占用空间统一为4字节。

3.使用用户存储区读写操作过程中请切记规划好数据区位置，以免位置交错引起数据覆盖错乱。

28.repo 从用户存储区(EEPRO)读数据到一个变量

(带用户存储的硬件才支持)

```
repo att,add
att:目标变量
add: 用户存储区位置(从0开始)
实例1:repo t0.txt,10（从用户存储区的10位置读数据到t0.txt变量中,在储存区中的读取数据量为t0.txt的最大设置值+1，即t0的txt-maxl属性表示的大小+1)
实例2.repo n0.val,10（从用户存储区的10位置读数据到n0.val,在存储区中的读取数据量为4字节)
```

备注:

1.读入内容为变量字符串的时候，在储存区中的读取数据量为此变量的最大字符数+1。

2.读入内容为变量数值时候，在储存区中的读取数据量统一为4字节。

3.使用用户存储区读写操作过程中请切记规划好数据区位置，以免位置交错引起数据覆盖错乱。

29.wept 透传数据写入用户存储区(EEPROM)

(带用户存储的硬件才支持)

```
wept add,lenth
add: 用户存储区位置(从0开始)
lenth:透传长度
实例:wept 10,30（透传30个字节的数据存到EEPROM的10位置，占用空间为10-39)
```

备注：

1.发完透传指令后，用户需要等待设备响应才能开始透传数据，设备收到透传指令后，准备透传初始化数据大概需要5ms左右(如果在透传指令执行前串口缓冲区还有很多别的指令，那时间会更长)，设备透传初始化准备好以后会发送一个透传就绪的数据给用户（0XFE+结束符），表示设备已经准备好，此时可以开始发送透传数据。透传数据为纯16进制数据，不再使用字符串，也不再需要结束符,设备收完指定的数据量以后，才会恢复指令接收状态。否则一直处于数据透传状态，透传数据完成以后,设备会发送结束标记给用户（0XFD+结束符）。

30.rept 从用户存储区读取数据并透传发送到串口

(带用户存储的硬件才支持)

```
rept add,lenth
add: 用户存储区位置(从0开始)
lenth:读取并透传发送的长度
实例:rept 10,30（从用户存储区的10位置读取30个字节并透传发送到串口)
```

备注：

不管存储区中的数据是字符串还是数值，设备都按16进制来读取和发送指定的字节数量到串口，并且不会发结束符。

31.cfgpio 扩展IO模式配置

(带扩展IO的硬件才支持)

```
cfgpio id,state,obj
id:扩展IO的序号
state:配置模式(0-上拉输入模式,1-控件事件邦定输入模式,2-推挽输出模式,3-PWM输出模式,4-开漏模式)
obj:绑定控件名称或ID(此参数仅在配置为控件事件邦定输入模式下有效，其他模式下无效)
实例1:cfgpio 0,0,0（将io0配置为上拉输入，配置为此模式后，任意时刻可以使用系统变量pio0读取当前输入电平,如:n0.val=pio0)
实例2:cfgpio 1,2,0（将io1配置为推挽输出,配置为此模式后，任意时刻可以使用系统变量pio1控制当前输出电平，如:pio1=1)
实例3:cfgpio 2,1,b0 (将io2配置为控件事件邦定输入，邦定控件为b0,配置为此模式后，io2产生下降沿的时候将触发b0控件的按下事件，产生上升沿的时候将触发b0控件的弹起事件)
实例4:cfgpio 4,3,0 (将io4配置为PWM输出模式，配置之前需要先设置占空比，即系统变量变量中的pwm4)
```

备注：

1.K0系列只有io4-io7才支持PWM输出，X5系列只有io6-io7才支持PWM输出 其他IO不支持。配置其他IO为PWM模式会报错。

2.使用控件事件邦定输入模式时，必须是在当前配置时刻的当前页面的控件才能邦定，不可以邦定其他页面的控件（即使是全局内存占用的控件也不可以），邦定当前页面控件以后，当重新刷新页面或者切换到别的页面后，邦定事件将不会继续触发，因此每次刷新页面需要重新邦定，建议将邦定代码写在页面的前初始化事件中最为合适。

32.crcrest 复位crc初始值

```
crcrest crctype,initval
crctype:crc校验类型(必须为1,Modbus crc16校验方式)
initval:初始值(一般为0xffff)
实例1:crcrest 1,0xffff (复位CRC初始值为0xffff，以便后续检验数据)
```

备注：

1.复位之后，可使用crcputs或crcputh或crcputu校验指定数据,检验完毕读取系统变量crcval获得校验结果。

2.完整的CRC校验实例代码请参考:程序中使用CRC校验数据

33.crcputs crc校验一个变量/常量

```
crcputs att,length
att:变量名称
length:需要校验的数据长度(0为自动长度)
实例1:crcputs t0.txt,0  (CRC校验字符串变量t0.txt)
实例2:crcputs "abc",0  (CRC校验字符串常量"abc")
```

备注：

1.使用crcputs或crcputh或crcputu校验指定数据,检验完毕读取系统变量crcval获得校验结果。

2.完整的CRC校验实例代码请参考:程序中使用CRC校验数据

34.crcputh crc校验一组Hex

```
crcputh Hex
Hex:需要校验的数据的16进制字符串表达式(每个字节之间用空格隔开)
实例1:crcputh 03 25   (CRC校验hex:0x03,0x25)
```

备注：

1.使用crcputs或crcputh或crcputu校验指定数据,检验完毕读取系统变量crcval获得校验结果。

2.完整的CRC校验实例代码请参考:程序中使用CRC校验数据

35.crcputu crc校验一段串口缓冲区数据(recmod=1时有效)

crcputu star,length
star:串口缓冲区数据起始位
length:需要校验的数据长度
实例1:crcputu 0,22　　(校验串口缓冲区的前面22个字节，recmod=1时有效)

备注：

1.使用crcputs或crcputh或crcputu校验指定数据,检验完毕读取系统变量crcval获得校验结果。

2.完整的CRC校验实例代码请参考:程序中使用CRC校验数据

36.setlayer 运行中改变控件图层顺序

(仅X3,X5系列支持)

setlayer obj0,obj1
obj0:需要改变图层的控件ID或控件名称
obj1:控件ID或控件名称(将obj0控件置于此控件之上,此参数为0表示将obj0控件放置到所有控件的下面,此参数为255表示将obj0控件放置到所有控件的上面)
实例1:setlayer n0,b0 (将n0控件置于b0图层之上)
实例2:setlayer n0,255 (将n0控件置于最顶层)

37.move 控件移动

(仅X3,X5系列支持)

move obj,starx,stary,endx,endy,first,time
obj:控件名称或控件ID
starx:起始坐标X
stary:起始坐标Y
endx:结束坐标X
endy:结束坐标Y
first:优先级(0-100,数字越大优先级越大)
time:移动时间(单位ms)
实例:move t0,0,0,200,200,0,300　(控件t0从(0,0)坐标移动到(200,200)坐标，优先级为0，时间为300ms)

备注：

可以同时写入多条不同优先级的move指令，系统会按照优先级顺序来移动。优先级高的会先移动，移动完成以后，才会启动下一个优先级的指令。

38.play 音频播放

(仅X3,X5系列支持)

```
play ch,audio,loop
ch:音频通道序号
audio:音频ID
loop：是否循环
实例:play 1,3,0（在音频通道1上播放ID为3的音频文件，不循环播放)
```

备注：

1.play指令仅用于配置和启动音频播放，暂停和停止操作请使用系统变量audio0,audio1。

2.play指令控制的是独立于视频之外的音频通道，与视频中使用的音频通道没有关系，也不会产生冲突。

3.音频播放功能是全局的，不属于某个页面，因此play指令启动播放后，即便是跳转页面，音频依然会继续播放，如果希望离开页面后停止播放，可以在页面的离开事件中使用audio0/audio1系统变量来关闭或暂停指定通道的音频播放状态。

39.twfile 串口透传文件

(仅X3,X5系列支持)

```
twfile filepath,filesize
filepath:目标文件路径(例:"ram/0.jpg"或"sd0/1.jpg")
filesize:文件大小实例:twfile "ram/0.jpg",10345 (透传一个大小为10345的文件到内存文件系统中，文件名为"0.jpg")
实例:twfile "sd0/a.jpg",10345 (透传一个大小为10345的文件到SD卡根目录，文件名为"a.jpg")
```

备注：

1.要使用内存文件系统必须先在工程配置选项中配置内存文件系统的大小，新建工程默认内存文件系统大小为0，即不可能使用。

2.详细的文件透传逻辑请参看"高级应用与特殊指令详解"文档,**点击此处跳转到详细说明页面**。

40.delfile 删除文件

(仅X3,X5系列支持)

```
delfile filepath
filepath:文件路径(例:"ram/0.jpg"或"sd0/1.jpg")
实例1:delfile "ram/0.jpg" (删除内存文件系统中的"0.jpg"文件)
实例2:delfile "sd0/a.jpg" (删除SD卡根目录中的"a.jpg"文件)
```

备注：

要使用内存文件系统必须先在工程配置选项中配置内存文件系统的大小，新建工程默认内存文件系统大小为0，即不可能使用。

41.refile 重命名文件

(仅X3,X5系列支持)

```
refile srcfilepath,decfilepat
srcfilepath:源文件路径
decfilepath:目标文件路径
实例1:refile "ram/0.jpg","ram/1.jpg" (将内存文件系统中的0.jpg更名为1.jpg)
实例2:refile "sd0/a.jpg","sd0/b.jpg" (将SD卡根目录中的a.jpg更名为b.jpg)
```

备注：

要使用内存文件系统必须先在工程配置选项中配置内存文件系统的大小，新建工程默认内存文件系统大小为0，即不可能使用。

42.findfile 查找文件

(仅X3,X5系列支持)

```
findfile filepath,att
filepath:文件路径
att:查找结果写入变量,必须是数值型变量(0-查找失败;1-查找成功)
实例1:findfile "ram/0.jpg",va0.val (查找内存文件系统中的0.jpg，将结果返回到va0.val中)
实例2:findfile "sd0/a.jpg",sys0  (查找SD卡根目录中的a.jpg，将结果返回到sys0中)
```

备注：

要使用内存文件系统必须先在工程配置选项中配置内存文件系统的大小，新建工程默认内存文件系统大小为0，即不可能使用。

43.rdfile 透传读文件

(仅X3,X5系列支持)

```
rdfile filepath,addr,size,crc
filepath:文件路径
addr:件数据起始地址
size:透传读数据大小 **(如果为0,则返回小端模式的4字节整形数据表示文件大小)**
crc:数据尾部添加CRC检验码设置(0-无crc;1-crc16;10-crc32)
实例1:rdfile "ram/0.jpg",0,10,1  (从内存文件系统中的"0.jpg"文件的数据0位置开始，读取10个字节透传到串口,并在数据尾部加入crc16校验码。合计是10+2=12字节)
实例2:rdfile "sd0/a.jpg",0,10,10  (从SD卡根目录中的"a.jpg"文件的数据0位置开始，读取10个字节透传到串口,并在数据尾部加入crc32校验码。合计是10+4=14字节)
```

备注：

1.要使用内存文件系统必须先在工程配置选项中配置内存文件系统的大小，新建工程默认内存文件系统大小为0，即不可能使用。

2.CRC16校验算法为MODBUS CRC16,**点击此处查看参考函数代码** CRC32校验算法为标准CRC32。

44.newfile 创建文件

(仅X3,X5系列支持)

```
newfile filepath,size
filepath:文件路径(如:"sd0/1.txt")
size:文件大小(以字节为单位,最大2147483647)
实例1:newfile "ram/0.txt",4096 (在内存文件系统中创建一个名为"0.txt"的文件，大小为4096字节)
实例2:newfile "sd0/1.txt",4096 (在SD卡根目录中创建一个名为"1.jpg"的文件，大小为4096字节)
```

备注：

1.要使用内存文件系统必须先在工程配置选项中配置内存文件系统的大小，新建工程默认内存文件系统大小为0，即不可能使用。

45.newdir 创建文件夹

(仅X3,X5系列支持)

```
newdir dir
dir:文件夹目录(如:"sd0/newfolder/")
实例:newdir "sd0/a/" (在SD卡根目录中创建一个名称为a的子目录)
```

备注：

1.内存文件系统不支持子目录，SD卡支持子目录，目录路径必须以"/"结尾。

46.deldir 删除文件夹

(仅X3,X5系列支持)

```
deldir dir
dir:文件夹目录(如:"sd0/newfolder/")
实例:deldir "sd0/a/" (在SD卡根目录中删除名称为a的子目录)
```

备注：

1.内存文件系统不支持子目录，SD卡支持子目录，目录路径必须以"/"结尾。

47.redir 重命名文件夹

(仅X3,X5系列支持)

```
redir srcdir,decdir
srcdir:源文件夹目录(如:"sd0/newfoldersrc/")
decdir:目标文件夹目录(如:"sd0/newfolderdec/")
实例:redir "sd0/a/", "sd0/b/" (在SD卡根目录中将名称为a的子目录改为b)
```

备注：

1.内存文件系统不支持子目录，SD卡支持子目录，目录路径必须以"/"结尾。

48.finddir 查找文件夹

(仅X3,X5系列支持)

```
finddir dir,att
dir:文件夹目录(如:"sd0/newfolder/")
att:查找结果写入变量(0-查找失败;1-查找成功。此变量必须是数值类型变量)
实例:finddir "sd0/a/",sys0 （查找SD卡根目录中是否有a的子目录，将结果返回到sys0中)
```

备注：

1.内存文件系统不支持子目录，SD卡支持子目录，目录路径必须以"/"结尾。

GUI绘图指令详解

注：GUI绘图指令主要应用在如下场合：

当上位界面编辑软件无法实现您的某些特殊显示要求的时候，使用GUI指令自己绘图来实现自己想要的显示效果。大多数情况下其实是不需要使用这些绘图指令的，大多数的应用都可以通过界面编辑软件的控件操作来实现。

1.cls 清屏指令

```
cls color
color:十进制颜色值或颜色代号
实例1:cls 1024（用十进制1024的颜色值刷屏)
实例2：cls RED（用代号为RED的颜色（ RED代表红色）刷屏)
```

备注：

1.想了解设备支持的颜色代号表请参看本表格后面的”串口HMI颜色代号表”。

2.本指令表中所有指令中的颜色参数，全部都可以使用设备支持的颜色代号，也可以使用10进制的颜色值，请知晓。

2.pic 刷图指令

```
pic x,y,picid
x:起始点x坐标；
y:起始点y坐标；
picid:图片ID；
实例1:pic 10,20,0（在坐标(10,20)位置显示资源文件中图片ID为0的图片)
实例2:pic 40,50,1（在坐标(40,50)位置显示资源文件中图片ID为1的图片）
```

3.picq 切图指令

格式:picq x,y,w,h,pici
dx:屏幕起始点x坐标；
y:屏幕起始点y坐标；
w:区域宽度；
h:区域高度；
picid:图片ID；
实例1:picq 20,50,30,20,0 （将图片0起始坐标(0,0)宽度30高度20这个区域切到屏幕上显示，屏幕上的显示起始坐标为(20,50)）

备注：

此指令要求图片必须是全屏图片，否则切出来的图像不是你想要的。图片上的切图区域和屏幕上的显示区是重叠的。

4.xpic 高级切图指令

格式:xpic x,y,w,h,x0,y0,picid
x:屏幕起始点x坐标；
y:屏幕起始点y坐标；
w:区域宽度 ；h:区域高度；
x0:图片起始点x坐标；
y0:图片起始点y坐标；
picid:图片ID；
实例1:xpic 20,50,30,20,40,15,0 （将图片0起始坐标(40,15)宽度30高度20这个区域切到屏幕上显示，屏幕上的显示起始坐标为(20,50)）

5.xstr 写字指令

xstr x,y,w,h,fontid,pointcolor,backcolor,xcenter,ycenter,sta,string
x:起始点坐标x；
y:起始点坐标y；
w:区域宽度；
h:区域高度；
fontid:字库ID；
pointcolor:字体颜色；
backcolor:背景色(sta设置为切图或图片时，backcolor表示图片ID);
xcenter:水平对齐方式(0为左对齐，1为居中，2为右对齐)；
ycenter: 垂直对齐方式(0为上对齐，1为居中，2为下对齐)；
sta:背景填充方式(0为切图，1为单色，2为图片，3为无背景,sta设置为切图或图片时，backcolor表示图片ID)
string:字符内容；
实例1:xstr 0,0,100,30,1,RED,BLACK,1,1,1,”中国”
实例解释：使用字库1在起始坐标(0,0)，宽度100,高度30这个区域写出”中国” ，字体色为RED，背景色为BLACK(如果不想写背景色(即无背景)可以设置sta参数为3),水平对齐方式为居中，垂直对齐方式也为居中。

备注：

1.字符写到超过设定的w以后将自动换行，如果换行到h之后还有剩下的字符没写完，将会被忽略。

2.关于颜色值的说明请参看cls指令的备注。

6.fill 区域填充指令

```
fill x,y,w,h,color
x:起始点坐标x ；
y:起始点坐标y；
w:区域宽度；
h:区域高度；
color:填充颜色；
实例1:fill 0,0,100,30,RED  (在起始坐标(0,0)宽度100，高度30这个区域填充RED颜色)
```

备注：

关于颜色值的说明请参看cls指令的备注。

7．line 画线指令

```
line x,y,x2,y2,color
x:起始点坐标x；
y:起始点坐标y；
x2:结束点坐标x ；
y2:结束点坐标y；
color:画线颜色；
实例1:line 0,0,100,100,RED  (在坐标(0,0)和坐标(100,100)之间画出一条RED颜色的线)
```

备注：

关于颜色值的说明请参看cls指令的备注。

8.draw 画矩形

```
draw x,y,x2,y2,color
x:起始点坐标x ；
y:起始点坐标y；
x2:结束点坐标x ；
y2:结束点坐标y；
color:画线颜色；
实例1:draw 0,0,100,100,RED  (画一个矩形，左上角为(0,0),右下角为(100,100), 颜色为RED)
```

备注：

1.draw画出来的是空心矩形，需要填充实心矩形的话请直接使用fill区域填充指令。

2.关于颜色值的说明请参看cls指令的备注。

9.cir 画空心圆

cir x,y,r,color
x:圆心坐标x
y:圆心坐标y
r:半径
color:画线颜色；
实例1:cir 100,100,30,RED 以坐标(100,100)为圆心画一个半径为30的空心圆，颜色为RED

备注：

关于颜色值的说明请参看cls指令的备注。

10.cirs 画实心圆

cirs x,y,r,color
x:圆心坐标x
y:圆心坐标y
r:半径
color:填充颜色；
实例1:cirs 100,100,30,RED 以坐标(100,100)为圆心画一个半径为30的实心圆，填充颜色为RED

备注：

关于颜色值的说明请参看cls指令的备注。

提示：本指令表中所有指令中的颜色参数，全部都可以使用设备支持的颜色代号，也可以使用10进制的颜色值。

HMI书写语法

【目录】

一、【赋值操作】

●所有的赋值操作可以在上位编辑状态下写入控件事件中，也可以串口传输过来(串口传输记得加三个0xff的结束符)

►字符串属性赋值:

```
t0.txt="123"    给t0控件的txt属性赋值"123"
t0.txt=t1.txt   把t1控件的txt属性赋值给t0控件的txt属性
```

以下为错误的赋值写法:

t0.txt=123　　错误原因：t0控件的txt属性为字符串类型，字符串类型的属性赋值常量必须加双引号

t0.txt=h0.val　　错误原因：h0控件的val属性是数值类型，不能赋值给字符串类型的属性

►数值属性赋值

```
n0.val=123     给n0控件的val属性赋值123
n0.val=h0.val  把h0控件的val属性赋值给n0控件的val属性
dim=80         给系统变量dim赋值80（背光亮度立即变为80亮度）
baud=115200    给系统变量baud赋值115200(屏幕波特率立即变为115200)
n0.val=baud    把屏幕当前的波特率系统变量赋值给n0控件的val属性
```

以下为错误的赋值写法:

n0.val="123"　错误原因：n0控件的val属性为数值类型，数值类型的属性赋值常量不应该有双引号

n0.val=t0.txt　错误原因：t0控件的txt属性是字符串类型，不能赋值给数值类型的属性

温馨提示:

1.到目前为止，只有txt属性是字符串类型，其他属性都是数值类型。

2.字符串类型和数值类型可以通过covx指令来实现相互转换赋值，具体请参看指令集中的covx指令说明。

二、【运算操作】

1所有运算不支持乘除法优先，也不支持括号优先级，统一从左到右的顺序，请特别注意。

●所有的运算操作可以在上位编辑状态下写入控件事件中，也可以串口传输过来(串口传输记得加三个0xff的结束符)

►数值类型变量运算操作

支持的运算符：+ - * / % & | ^ << >>

实例：

```
n0.val=n0.val+n1.val+2
n0.val++
n0.val+=2
n0.val=n1.val%3
n0.val=h0.val*10
n0.val*=10
n0.val=n1.val&3
n0.val=n1.val^5
.....
```

1以下为错误的运算写法：

n0.val=t0.txt+1　错误原因：数值类型的变量必须跟数值类型的变量做运算，并赋值给数值类型的变量

n0.val=1+"2"　错误原因：数值类型的变量必须跟数值类型的变量做运算，并赋值给数值类型的变量

►字符串类型变量运算操作

运算符 "+"

实例：

```
t0.txt="1"+"2"
t0.txt=t0.txt+t1.txt
t0.txt+="abc"+"xy"
```

以下为错误的运算写法：

t0.txt=1+2　错误原因：1和2都是数值常量 字符串类型的变量只能跟字符串常量/变量相加，不能跟一个数值常量/变量相加

t0.txt=t0.txt+h0.val 错误原因：h0.val是数值变量,不能跟字符串变量相加

运算符 "-"

实例：

```
t0.txt=t0.txt-1    删除t0.txt最后1个字符
t0.txt=t0.txt-3    删除t0.txt最后3个字符
t0.txt-=n0.val     删除t0.txt最后n0.val个字符
```

在字符串变量运算中，"-"代表删除的意思,所以用"-"的时候，字符串变量必须"-"一个数值常量/变量来表示删除多少个字符，这里跟用"+"是不一样的。用"+"的时候必须是字符串+字符串；用"-"的时候必须是字符串-数值。

三、【跨页面操作控件属性】

多数情况下，我们都是在操作当面页面的控件属性，如果需要操作其他页面的控件属性请按如下书写方式:

[页面].[控件].[属性]=XXX

实例:

```
main.t0.txt="123"          给main页面的t0.txt属性赋值"123"
main.t0.txt=set.t3.txt     把set页面的t3.txt赋值给main页面的t0.txt
set.t4.txt="abc"           给set页面的t4.txt赋值"abc"
```

特别注意：跨页面操作控件属性的时候，不管是读取还是赋值，被操作控件的vscope属性必须设置为全局（默认是私有），否则操作会失败。

四、【HMI逻辑语句】

●注：所有的逻辑语句只能在上位编辑状态下写入控件的事件中，不支持串口传输逻辑语句。

1.if语句

实例1:（如果t0.txt等于"123456"那么就切换到页面1）

```
if(t0.txt=="123456")
{
  page 1
}
```

实例2:（如果t0.txt不等于"a"也不等于b"那么就切换到页面1）

```
if(t0.txt!="a"&&t0.txt!="b")
{
  page 1
}
```

实例3:（如果t0.txt等于"a"或者等于b"那么就切换到页面1）

```
if(t0.txt=="a"||t0.txt=="b")
{
  page 1
}
```

实例4:(以下语句写在b0按钮的按下事件中将实现b0的txt内容在开始和停止之间来回切换)

```
if(b0.txt=="开始")
{
  b0.txt="停止"
}else
{
  b0.txt="开始"
}
```

实例5:(以下语句写在b0按钮的按下事件中将实现b0的txt内容在1,2,3之间来回切换)

```
if(b0.txt=="1")
{
 b0.txt="2"
}else if(b0.txt=="2")
{
 b0.txt="3"
}else
{
 b0.txt="1"
}
```

备注:

1.数值类型变量支持: >; <; ==; !=; >=; <=

2.字符串类型仅支持: ==; !=

3.if判断的时候不支持括号优先级，比如:if((t0.txt=="a"||t0.txt=="b")&&t1.txt=="1")这样是不支持的！还有就是不要出现多余的空格。

4.if判断的时候不允许出现运算，比如:if(n0.val+2==3)这样是不支持的。

2.while语句

实例1: (n0.val一直自加到100为止，在自加过程中屏幕不会刷新显示，直到整个过程所有语句结束)

```
while(n0.val<100)
{
  n0.val++
}
```

实例2: (n0.val一直自加到100为止，在自加过程中屏幕会一直不断的刷新n0控件的显示)

```
while(n0.val<100)
{
  n0.val++
  doevents
}
```

备注:

1.在一个较多指令的过程执行中，或者在一个较长时间的循环语句中，系统所有控制权被此过程全部占用，在过程结束之前，尽管相应的内存数据可以任意正常读写，但是屏幕不会刷新显示，加入doevents后可以转让控制权给屏幕刷新，执行doevents之后，屏幕会刷新所有被改变过的控件，刷新完之后，控制权交回当前过程继续执行。防止屏幕呈现假死的显示状态。

2.while语句循环过程中，设备不会响应触摸事件，串口指令会接收到缓冲区，但不会执行，直到当前过程所有语句执行完毕为止，请慎重使用，以防进入死循环。

3.for语句

实例1: (n0.val一直自加到100为止，在自加过程中屏幕不会刷新显示，直到整个过程所有语句结束)

```
for(n0.val=0;n0.val<100;n0.val++)
{
}
```

实例2: (n0.val一直自加到100为止，在自加过程中屏幕会一直不断的刷新n0控件的显示)

```
for(n0.val=0;n0.val<100;n0.val++)
{
  doevents
}
```

备注:

1.在一个较多指令的过程执行中，或者在一个较长时间的循环语句中，系统所有控制权被此过程全部占用，在过程结束之前，尽管相应的内存数据可以任意正常读写，但是屏幕不会刷新显示，加入doevents后可以转让控制权给屏幕刷新，执行doevents之后，屏幕会刷新所有被改变过的控件，刷新完之后，控制权交回当前过程继续执行。防止屏幕呈现假死的显示状态。

2.for语句循环过程中，设备不会响应触摸事件，串口指令会接收到缓冲区，但不会执行，直到当前过程所有语句执行完毕为止，请慎重使用，以防进入死循环。

串口HMI系统变量列表

注：所有变量名称使用小写字符

序号	名称	含义	示例/备注
1	dp	当前页面ID	1.dp=1 (设置当前页面为1,等同于page 1) 2.prints dp,0(发送当面页面ID到串口)3.n0.val=dp (当前页面ID赋值给n0.val)
2	dim	当前背光亮度值(0-100)	1.dim=50 2.dim=dim+10 3.dim=dim-10
3	dims	上电默认背光亮度值(0-100)	1.dims=50 2.dims=dims+10 3.dims=dims-10
4	baud	当前波特率值(本次修改，掉电后丢失)	baud=9600备注:设备支持的波特率有:2400 4800 9600 19200 38400 57600 115200230400 256000 512000 921600
5	bauds	上电默认波特率值（掉电后保存，下次上电后继续有效）	bauds=9600
6	spax	字符显示横向间距(上电默认为0)	spax=2备注：仅对xstr指令写出来的字符有效，控件带的字符显示间距由控件内部的属性决定。
7	spay	字符显示纵向间距(上电默认为0)	spay=2备注：仅对xstr指令写出来的字符有效，控件带的字符显示间距由控件内部的属性决定。
8	thc	触摸绘图时的画笔色	1.thc=RED 2.thc=1024
9	thdra	触摸绘图功能开关	thdra=0 (关闭) thdra=1(打开)
10	ussp	无串口数据自动睡眠时间(单位:秒,最小3，最大65535，上电默认0[0为关闭串口数据超时自动睡眠])	ussp=30(30秒无串口数据自动进入睡眠模式)
11	thsp	无触摸操作自动睡眠时间(单位:秒,最小3，最大65535，上电默认0[0为关闭触摸超时自动睡眠])	thsp=30(30秒无触摸操作自动进入睡眠模式)
12	thup	睡眠模式下触摸自动唤醒开关(上电默认0)	thup=0(睡眠后触摸不会自动唤醒)thup=1(睡眠后触摸自动唤醒)备注:不管thup为0还是1，睡眠模式下有触摸操作的时候设备均会发送触摸坐标到串口。

序号	名称	含义	示例/备注
13	usup	睡眠模式下串口数据自动唤醒开关(上电默认0)	usup=0(睡眠后串口不会自动唤醒)usup=1(睡眠后串口自动唤醒)备注:上电默认为0，不会自定唤醒，需要发送sleep=0才能唤醒屏幕，如果设置为1，串口收到任何数据都会立刻自动唤醒。
14	wup	睡眠唤醒后刷新页面设置	wup=255(上电默认，睡眠唤醒后刷新睡眠前页面)wup=2 (睡眠唤醒后刷新页面指定页面:2)备注：设备已经在睡眠状态下，也可以执行串口传过来的wup=X赋值。
15	sleep	睡眠	sleep=0 (退出睡眠)sleep=1 (进入睡眠)备注：睡眠状态下可以执行如下指令：get,print, printh。 也可以 执行sleep=1，wup=X 的赋值语句，并且支持上位软件联机，其他指令不会执行。如果是带扩展IO的硬件,IO配置为绑定控件事件时，睡眠模式下也不会产生中断事件。
16	bkcmd	设置串口指令执行成功或者失败的数据返回(上电默认为2)	bkcmd=0(不返回结果)bkcmd=1(只返回成功的结果)bkcmd=2(只返回失败的结果)bkcmd=3(成功或者失败都返回结果)备注:此设置只影响串口指令执行成功或者失败的结果返回，上位软件编辑界面时写入的指令执行错误的时候一定会返回错误结果，成功的时候一定不会返回执行结果。此设置也不会影响获取设备控件数据时的数据返回。
17	sendxy	实时发送触摸坐标功能开关	sendxy=0(关闭) sendxy=1(打开)备注:1打开发送功能以后，有触摸按下的时候设备会通过串口发送触摸坐标。2.发送坐标的格式请参看本表格后面的"串口HMI设备返回数据格式"表格。
18	delay	延时	delay=100 (让设备停顿100ms)备注：执行延时指令后，设备CPU不会执行任何指令，但是会继续接受串口指令保存到串口指令缓存区。

序号	名称	含义	示例/备注
19	rand	随机数	dim=rand (把一个随机数赋值给背光亮度)n0.val=rand (把一个随机数赋值给n0.val变量)备注：1.使用随机数之前需要先使用randset指令设定一次随机数产生范围，如果不设置，默认是最小0，最大2147483647。设置完范围以后，每读取一次系统变量rand将会得到一个随机数。2.使用randset指令每设定一次范围，将一直有效，直到重新上电或者设备复位才会恢复默认。
20	tch0-tch3	实时触摸坐标	tch0:当前触摸坐标Xtch1:当前触摸坐标Ytch2:上一次按下时的坐标Xtch3:上一次按下时的坐标y备注：触摸坐标只能读取，不能赋值，没有按下时，实时坐标数据为0。
21	addr	设备地址	字符串写法:addr=256HEX写法:addr=0x0100以上两条写法是同一个意思，配置的是同一个地址，配置之后有断电保存功能。备注：1. 有效地址范围为256-2815(即0x0100-0x0aff),0为无地址,65535为广播地址，广播地址只能用于广播数据，不能配置某个设备为广播地址，出厂默认地址为0,即没有地址。2.向一个有地址的设备发送指令时，需要在指令前加上2字节的地址数据，以hex方式发送,2字节小端模式,比如设备配置的地址为addr=256,那么发送给他指令时需要在指令前面增加两个字节:0x00 0x01(注意，配置的时候是0x0100,发送指令的时候是低位在前，所以是0x00 0x01跟配置的写法是相反的)。
22	crcval	crc校验结果(只可获取不可设置,使用前请先用crcrest指令复位初始值)	n0.val=crcval (当前CRC校验结果赋值给n0.val)prints crcval,2 (当前CRC校验结果的低2位打印到串口)备注：1.先使用crcrest复位CRC值，复位之后，可使用crcputs或crcputh或crcputu校验指定数据,检验完毕读取系统变量crcval获得校验结果)2.完整的CRC校验实例代码请参考:程序中使用CRC校验数据

序号	名称	含义	示例/备注
23	rtc0-rtc6	RTC时钟变量(带RTC的硬件才支持)	n0.val=rtc5 (当前RTC的秒数值赋值给n0.val)rtc0=2016(RTC的年设置为2016)cov rtc5,t0.txt,0(当前RTC的秒数值转换给t0.txt)备注：1.rtc0-rtc6分别表示年，月，日，时，分，秒,星期。2.rtc6(星期)为只读。根据当前的年月日自动计算生成。
24	pio0-pio7	扩展IO端口(带扩展IO的硬件才支持)	pio4=1 (IO4置为1)n0.val=pio2 (io2的电平状态赋值给n0.val)cov pio3,t0.txt,0 (io3的电平状态转换给t0.txt)备注：1.使用pio端口之前一定要先使用cfgpio指令配置好IO模式。2.上电默认所有扩展IO模式为上拉输入（内部上拉电阻为50K）。
25	pwm4-pwm7	扩展IO占空比(带扩展IO的硬件才支持)K0系列仅io4-io7支持PWMX5系列仅io6-io7支持PWM	pwm6=30 (设置pwm6占空比为30)pwm7=90 (设置pwm7占空比为90)备注：1. 占空比最小值0最大值100,上电默认50。2.pwm4-pwm7依次对应扩展IO中的io4-io7,3.设置好PWM占空比以后,需要使用cfgpio指令配置此IO的模式为PWM输出模式，相应IO才会开始输出PWM,配置完PWM模式后在PWM输出的过程中可以随时修改占空比，不用重新配置。4. 上电默认所有扩展IO模式为上拉输入（内部上拉电阻为50K）。>
26	pwmf	PWM输出的频率(带扩展IO的硬件才支持)	pwmf=1024 (设置pwm的频率为1024HZ)n0.val=pwmf (将PWM频率赋值给n0.val)cov pwmf,t0.txt(将PWM频率转换给t0.txt)备注：1.频率单位为:HZ,范围为最小1，最大65535HZ，上电默认1000HZ。2.所有PWM输出统一为一个频率，不可单独设置。
27	eqleqmeqh	eql-低音衰减(31HZ-125HZ)eqm-中音衰减(250HZ-2000HZ)eqh-高音衰减(4000HZ-1600HZ)(带音频的硬件才支持)上位模拟器不支持	设置范围0-150-6为衰减，数字越小衰减越大8-15为提升，数字越大提升越大7为平衡，无衰减，无提升备注：系统底层是按eq0-eq9的设置来操作的，如果分别修改eql,eqm,eqh等同于分别修改eq0-eq2, eq3-eq6, eq7-eq9；但是修改eq0-eq9并不会影响到eql,eqm,eqh的值。

序号	名称	含义	示例/备注
28	eq0-eq9	独立频点衰减(带音频的硬件才支持)上位模拟器不支持 eq0@31HZ eq1@62HZ eq2@125HZeq3@250HZ eq4@500HZ eq5@1000HZeq6@2000HZ eq7@4000HZ eq8@8000HZeq9@16000HZ	设置范围0-150-6为衰减，数字越小衰减越大8-15为提升，数字越大提升越大7为平衡，无衰减，无提升备注：系统底层是按eq0-eq9的设置来操作的，如果分别修改eql,eqm,eqh等同于分别修改eq0-eq2, eq3-eq6, eq7-eq9；但是修改eq0-eq9并不会影响到eql,eqm,eqh的值。
29	volume	系统音量 (最小0，最大100)(带音频的硬件才支持)	volume=60(设置音量60)备注：音量设置范围为0-100，每次设置会自动保存，断电后再开机依然有效。
30	audio0audio1	音频通道控制0-停止;1-播放;2-暂停(带音频的硬件才支持)	audio0=2 (暂停0通道的音频播放)audio0=0 (停止0通道的音频播放)audio1=1 (继续1通道的音频播放)备注：1.play指令用于配置和启动音频播放系统变量audio0,audio1用于控制通道状态。2.只有当通道状态在暂停的时候，才能配置为继续播放。如果通道状态为停止，将不能配置为继续播放，需要使用play指令来配置并启动播放。3.音频播放功能是全局的，不属于某个页面，因此play指令启动播放后，即便是跳转页面，音频依然会继续播放，如果希望离开页面后停止播放，可以在页面的离开事件中使用audio0/audio1系统变量来关闭或暂停指定通道的音频播放状态。

【串口HMI颜色代号表】

注：所有代号的书写均为大写>

代号	10进制	所表示的颜色
RED	63488	红色
BLUE	31	蓝色
GRAY	33840	灰色
BLACK	0	黑色
WHITE	65535	白色
GREEN	2016	绿色
BROWN	48192	橙色
YELLOW	65504	黄色

串口HMI设备返回数据格式

表格一：串口指令执行成功或失败的通知格式

1. 只有当系统变量bkcmd为非0的时候才会返回指令执行成功或者失败数据，每次上电后bmcmd默认为2，即只返回指令执行出错的结果。

2. 上位软件编辑时写进资源文件的代码不受bkcmd影响，执行有错误时一定会返回错误数据，成功时不返回数据。

3：设备返回数据的结束符为"0XFF 0XFF 0XFF"三个字节。

<<<<<<<<<<<<< **返回数据第一位** >>>>>>>>>>	<<<<<<< **含义** >>>>>>>	**格式**
0X00	无效指令	0X00+结束符(当收到用户发来的无效指令时返回此数据)
0X01	指令成功执行	0X01+结束符(用户发来的指令被成功执行完毕时返回此数据)
0X02	控件ID无效	0X02+结束符(用户发来的指令中包含无效控件ID或者无效控件名称时返回此数据)
0X03	页面ID无效	0X03+结束符(用户发来的指令中包含无效页面ID或者无效页面名称时返回此数据)
0X04	图片ID无效	0X04+结束符(用户发来的指令中包含无效图片ID时返回此数据)
0X05	字库ID无效	0X05+结束符(用户发来的指令中包含无效字库ID时返回此数据)
0x06	文件操作失败	0X06+结束符(操作文件失败的时候返回)
0x09	CRC校验失败	0X09+结束符(带校验的指令通不过的时候返回此数据,指令校验说明
0X11	波特率设置无效	0X11+结束符(用户发来的波特率配置指令中包含无效波特率参数)设备支持的波特率有:2400 4800 9600 19200 38400 57600 115200

0X12	**曲线控件ID号或通道号无效**	**0X12+结束符(用户使用add指令往曲线控件添加数据的时候，曲线控件ID号或通道号无效时返回此数据)**
0X1A	变量名称无效	0X1A+结束符当串口收到的变量名称为无效名称时返回此数据注：控件属性也称为变量，比如您设置一个控件的属性的时候，输入的是一个它没有的属性名称，也会返回此数据。
0X1B	变量运算无效	0X1B+结束符比如文本控件t0的txt属性赋值时应该写成t0.txt=”abc”如果你写成t0.txt=abc就出错了，再比如进度条j0的val属性应该是数值，所以要写成j0.val=50,如果写成j0.val=”50”或者j0.val=abc也会出错
0X1C	赋值操作失败	0X1C+结束符属性赋值失败的时候返回此数据
0X1D	EEPROM操作失败	0X1D+结束符操作EEPROM失败时返回此数据
0X1E	参数数量无效	0X1E+结束符用户输入的指令中参数数量错误的时候返回此数据
0X1F	IO操作失败	0X1F+结束符操作IO失败时返回此数据
0X20	转义字符使用错误	0X20+结束符转义字符使用错误时返回此数据
0X23	变量名称太长	0X23+结束符变量名称长度最大29个字符，超出就会返回此数据
0X24	串口缓冲区溢出	0X24+结束符当串口缓冲区被占满以后会返回此数据(缓冲区溢出以后，缓冲队列里的指令执行完成后会为缓冲区腾出空间以继续接收指令，在此之前，串口收到的数据将会丢弃)

表格二：其他数据返回格式

1.设备返回数据的结束符为"0XFF 0XFF 0XFF"三个字节。2.以下数据的返回不受bkcmd影响。

返回数据第一位	<<<<<<< 含义 >>>>>>>	格式
0X65	触摸热区事件返回	0X65+页面ID+按键ID+触摸事件+结束符(用户创建的控件被按下或弹起时返回此数据，前提是您勾选了控件的”发送键值”选框)(触摸事件的定义：按下事件0x01 弹起事件0X00) 举例:0X65 0X00 0X02 0X01 0XFF 0XFF 0XFF含义:页面0 按钮2 按下
0X66	当前页面的ID号返回	0X66+页面ID+结束符(设备收到”sendme”指令时会返回此数据) 举例：0X66 0X02 0XFF 0XFF 0XFF含义：当前页面ID为2
0X67	触摸坐标数据返回	0X67+坐标X高位+坐标X低位+坐标Y高位+坐标Y低位+触摸事件状态+结束符(当系统变量”sendxy”为1之后，有触摸事件时返回此数据)(触摸事件的定义：按下事件0x01 弹起事件0X00) 举例:0X67 0X00 0X7A 0X00 0X1E 0X01 0XFF 0XFF 0XFF含义:坐标(122,30) 事件: 按下
0X68	睡眠模式触摸事件	0X68++坐标X高位+坐标X低位+坐标Y高位+坐标Y低位+触摸事件状态+结束符(当设备进入睡眠模式后，有触摸事件时返回此数据)(触摸事件的定义：按下事件0x01 弹起事件0X00) 举例:0X68 0X00 0X7A 0X00 0X1E 0X01 0XFF 0XFF 0XFF含义:坐标(122,30) 事件: 按下
0X70	字符串变量数据返回	0X70+变量内容ASCII码+结束符使用get指令获取的变量为字符串类型时，返回此数据. 举例:0X70 0X61 0X62 0X63 0XFF 0XFF 0XFF含义：返回字符串数据:”abc”
0X71	数值变量数据返回	0X71+变量二进制数据（4字节小端模式,低位在前）+结束符使用get指令获取的变量为数值时，返回此数据. 举例:0X71 0X66 0X00 0X00 0X00 0XFF 0XFF 0XFF含义：返回数值数据:102
0X86	设备自动进入睡眠模式	0x86+结束符只有设备自动进入睡眠模式的时候会返回此数据，如果是执行串口指令sleep=1进入的睡眠不会返回此数据
0X87	设备自动唤醒	0x87+结束符只有设备自动唤醒的时候会返回此数据，如果是执行串口指令sleep=0唤醒的睡眠不会返回此数据
0X88	系统启动成功	0x88+结束符设备上电初始化成功之后发送此数据
0X89	开始SD卡升级	0x89+结束符设备上电检测到SD卡之后将发送此数据,然后进入升级界面
0XFD	透传数据完成	0xFD+结束符透传数据结束并处理数据完成后发送此数据
0XFE	数据透传就绪	设备收到数据透传指令后，进入透传数据初始化，初始化完成以后发送此数据，表示此时已经进入数据透传模式，可以开始数据透传

名称组使用说明

大多数情况下，我们操作控件属性是这样的(这也是我们推荐的控件操作方式)：

```
t0.txt="123"    //给t0控件的txt属性赋值"123"
n0.val=15       //给n0控件的val属性赋值15
z0.val=185      //给z0控件的val属性赋值185
```

假如我们不知道控件名字，只知道控件ID怎么操作呢？这就需要名称组来实现

►控件名称组格式:b[控件ID].属性

例：

```
b[5].txt="123"       //当前页面ID为5的控件的txt属性赋值为"123"
b[n0.val].txt="123"  //当前页面ID为n0.val的控件的txt属性赋值为"123"
```

注:必须确保当前页面的此控件具有txt属性,否则赋值会失败

►页面名称组格式:p[页面ID].b[控件].属性

例：

```
p[2].b[4].txt="456"            //ID为2的页面中，ID为4的控件的txt属性赋值为"456"
p[n0.val].b[n1.val].txt="456"  //ID为n0.val的页面中，ID为n1.val的控件的txt属性赋值为"456"
```

注:必须确保此页面中的此控件具有txt属性,否则赋值会失败

►**配合for语句批量改变控件属性**(将eeprom地址100开始连续的值加载到数值控件n0～n9(ID:5~14)):

```
sys1=100                    //设置eeprom读取位置
for(sys0=5;sys0<=14;sys0++)    //for循环读取eeprom并赋值给数值控件
{
  repo b[sys0].val,sys1        //读取指定位置的eeprom数据
  sys1+=4                   //eeprom读取地址加4，因为一个数值类型占用4byte空间。
}
```

注:必须确保ID从5-14的控件都具有val属性,否则赋值会失败

●所有控件的ID号软件自动分配，不可手动设置，用户在编辑UI界面时，按顺序依次放置的控件，软件将连续分配ID。

●使用快捷栏的“置顶”、“置底”功能会使控件ID发生变化，因为图层的前后关系是跟控件ID关联的，ID最小的在最底层(所以页面ID是0),ID最大的在最上层，每个控件都有自己的图层，全部是通过ID来区别前后关系的。

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，大多数项目是用不上这些功能的，如果您的项目用不上，可以忽略此文档。

【章节索引】

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。每次读完数据后，使用udelete指令删除缓冲区中已经读取的字节数，否则缓冲区溢出后就无法接收新数据。

注：在屏幕设置主动解析模式，用上位机软件模拟器联机屏幕，屏幕将会强制退出主动解析模式，此时需要将屏重新上电才可以正常模拟器联机调试屏幕。

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]
```

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]
```

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
udelete 4      //删除缓冲区中前面4个字节，如果后面还有其他数据，会自动顶上来
```

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
udelete 2      //删除缓冲区中前面2个字节，如果后面还有其他数据，会自动顶上来
```

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10个
{
ucopy t0.txt,0,10,0
udelete 10               //删除缓冲区中前面10个字节，如果后面还有其他数据，会自动顶上来
}
```

重要提示：code_c指令会将缓冲区数据清空，有可能会导致还没有读取的数据就被提前删除，所以在主动解析模式下建议使用udelete指令来删除已经读取的内容，而不要使用code_c指令。

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(Hex数据)：
0xff 0xff 0xff

合计27字节

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。

温馨提示：官方有按照本协议开发好的专用下载工具(TFTFileDownload)，并且开放源代码，欢迎下载使用：点击此处下载

【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+结束符
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4个字节的HEX空指令（00 ff ff ff）,第二条才是connect+结束符
延时说明:每次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。

三.串口指令增加CRC校验(0.56及其以上上位机版本支持)>

正常情况下，指令直接发送即可，不需要校验，如果您的项目对指令传输要求很严格必须开启校验请按照下列说明发送指令

切记注意：上位软件版本0.56开始才支持指令CRC，之前的版本是不支持的。

带校验或不带校验无需做任何配置，只需修改指令即可，您可以上一条指令带校验，下一条指令不带校验也是可以的。两种指令的区别如下：

1.变更结束符

普通指令结束符为0xff 0xff 0xff,带校验的指令结束符为0xfe 0xfe 0xfe。

2.CRC16校验算法

指令CRC16校验算法使用MODBUS的CRC16校验算法,计算函数如下:

需要校验的数据为所有指令数据，如果是带地址的指令，从地址开始算，不带地址的指令，就从指令第一个字节开始计算，结束符不计算在内。

```
static U8 InvertUint8(U8 data)
{
  int i;
  U8 newtemp8 = 0;
  for (i = 0; i < 8; i++)
  {
    if ( (data & (1 << i) ) != 0) newtemp8 |= (U8)(1 << (7 - i));
```

```
    }
    return newtemp8;
}
static U16 InvertUint16(U16 data)
{
    int i;
    U16 newtemp16 = 0;
    for (i = 0; i < 16; i++)
    {
        if ( (data & (1 << i) ) != 0) newtemp16 |= (U16)(1 << (15 - i));
     }
     return newtemp16;
}
U16 CRC16_MODBUS(U8* data, int lenth)
{
    int i;
    U16 wCRCin = 0xFFFF;
    U16 wCPoly = 0x8005;
    U16 wChar = 0;
    while (lenth > 0)
    {
       wChar = *data;
       data++;
       wChar = InvertUint8( (U8)wChar);
       wCRCin ^= (U16)(wChar << 8);
       for (i = 0; i < 8; i++)
       {
          if ((wCRCin & 0x8000) != 0)
          {
             wCRCin = (U16)( (wCRCin << 1) ^ wCPoly);
          }else
          {
             wCRCin = (U16)(wCRCin << 1);
           }
        }
        lenth=lenth-1;
     }
     wCRCin = InvertUint16(wCRCin);
     return (wCRCin);
}
```

3.CRC16校验码写入方式

指令后面，结束符前面，加上2字节的CRC16校验码(HEX)+1字节的常数:0x01(HEX),相当于在指令和结束符中间插入了3个字节，CRC校验码的储存方式是小端模式,低位在前。

如果屏幕收到带校验的指令后发现校验失败，会返回错误：0x09 0xff 0xff 0xff

四.程序中使用CRC校验数据(1.60.1及其以上上位机版本支持)

1.以下代码将实现屏幕给外部设备发送2组常量数据，并在结尾带上Modbus的CRC16校验结果：

```
crcrest 1,0xffff            复位CRC
crcputs "hex date is:",0     CRC校验字符串:hex date is:
crcputh 03 25                CRC校验hex:0x03,0x25
prints "hex date is:",0      串口发送字符串:hex date is:
printh 03 25                 串口发送hex:0x03,0x25
prints crcval,2              串口发送校验结果
```

2.以下代码将实现屏幕给外部设备发送2组变量数据，并在结尾带上Modbus的CRC16校验结果：

```
crcrest 1,0xffff            复位CRC
crcputs t0.txt,0             CRC校验字符串变量t0.txt
crcputs n0.val,0             CRC校验数值变量n0.val
prints t0.txt,0             串口发送字符串变量t0.txt
prints n0.val,4              串口发送数值变量n0.val
prints crcval,2              串口发送校验结果
```

3.以下代码将实现在主动解析模式下对串口缓冲区的数据进行CRC校验,并将成功与否的结果显示到t0.txt

```
if(usize>=24)
{
  crcrest 1,0xffff        复位CRC
  crcputu 0,22             校验串口缓冲区的前面22个字节
  sys0=0                  sys0是4字节的整形数据,CRC结果只有2字节，所以先给多余字节数据清零
  ucopy sys0,22,2,0        将接收缓冲区中接收到的CRC结果赋值给sys0
  if(sys0==crcval)校验通过
  {
    t0.txt="校验通过"
  }else
  {
    t0.txt="校验不通过"
  }
  udelete 24              删除已经处理的24字节
}
```

五.运行中串口传输图片到内存或SD卡(0.56及其以上上位机版本支持)

如果要传输图片文件到内存，请先在工程设置里面配置内存文件系统空间大小，默认是0，即没有空间用来储存内存文件。

如果是传输图片文件到SD卡，请确保SD卡磁盘格式为FAT32,最大支持32G容量的卡

温馨提示：官方有按照本协议开发好的专用串口透传文件工具(SerialFileUp)，并且开放源代码，欢迎下载使用：点击此处下载

1.如何使用内存中的图片文件

使用“外部图片“控件，设置path属性为文件路径，如:ram/0.jpg

2.如何使用SD卡中的图片文件

使用“外部图片“控件，设置path属性为文件路径，如:sd0/0.jpg

3.支持什么类型的图片文件

jpg格式:如果使用jpg格式的图片，务必保证图片编码为:baseline DCT,否则将无法显示,如果屏幕方向设置的是0度，直接使用jpg原图即可，如果屏幕方向设置的是90度，需要将图片顺时钟旋转90度再给屏幕使用，否则显示出来的图片方向是错的。其他方向180度，270度以此类推，屏幕方向是多少度，就需要您提前把图片顺时钟旋转多少度。

xi格式:此格式为HMI专用图片格式，支持透明背景，推荐使用这种格式，xi格式的图片获取方式可以使用软件工具菜单中的图片转换工具:PictureBox转换

目前仅支持以上两种格式的图片文件。

4.串口传输协议

第一步:发送串口传输文件指令:twfile filepath,filesize

> filepath:文件存放路径 如:ram/a.jpg 或 sd0/a.jpg
> filesize:文件实际大小
> 假如要传一个文件到内存中，名字为a.jpg,大小为3282,指令为：twfile "ram/a.jpg",3282
> 假如要传一个文件到SD卡中，名字为a.jpg,大小为3282,指令为：twfile "sd0/a.jpg",3282
> 屏幕收到此指令后，会立即创建一个指定大小的文件在目标路径上，如果创建成功将返回:0xfe+结束符,表示已经进入透传状态，可以开始分包透传数据。如果创建文件失败会返回:0x06+结束符,并继续工作在指令接收状态。

第二步：分包透传文件数据

> 收到0xfe+结束符后，可以开始分包透传数据;一个完整的数据包由2部分组成：包头+数据

包头:3a a1 bb 44 7f ff fe +校验类型(1字节整形数据) +包ID(2字节整形数据)+数据大小(2字节整形数据)，合计12字节

> 校验类型:0x00为无校验,0x01为CRC16(MODBUS的CRC16校验算法,指令校验篇幅中有计算函数参考),0x0A为标准CRC32
> 包ID:文件透传开始第一次包ID为0，每成功透传一包，ID加1.
> 数据大小:数据大小用户随意指定，最小1字节，最大4096字节(此数据大小不包含12字节的包头，但是包含CRC校验码)

数据:文件数据+CRC校验码数据(小端模式，低位在前),如果是无校验的包，就没有CRC校验码。如果是CRC16就是2字节的校验码，如果是CRC32就是4字节的校验码，切记注意校验码数据要记入包头的数据大小参数中。

CRC16校验算法为MODBUS CRC16,**点击此处查看参考函数代码**) CRC32校验算法为标准CRC32。

屏幕收到一个完整的包数据后，如果处理成功会返回0x05(单字节，没有结束符),此时可进入下一个包的发送。

如果包ID不按规则累加或者包数据出错，屏幕将会返回本包处理失败的错误:0x04（单字节，无结束符）。

所有包发送完成后，屏幕会返回0xfd+结束符。并自动退出透传模式转为指令模式。

如果本包透传失败（超过500ms没有收到屏幕回应或者收到屏幕返回本包校验失败的错误信息），重新发本包数据，重发本包数据时包ID无需加1。

如果数据包发了一半后悔了，停顿20ms以上重新发本包数据即可。

如果透传文件中途想停止透传，请发一个包ID为65535，无校验，数据大小为0的数据包,即:3a a1 bb 44 7f ff fe 00 ff ff 00 00 屏幕收到这样的包数据后会立刻强制结束透传，并返回透传结束的数据：0xfd+结束符。

如果你的文件数据中包含退出包数据是不用担心的，因为这个数据的前面满足不了20ms以上的停顿，屏幕只会把他当数据储存，不会当结束包来处理。

设备与串口屏通信协议

【章节索引】

一、串口数据解析模式之被动解析模式

在默认情况下屏接收设备发送数据完整格式为字符串指令加上3个16进制ff，如果屏接收到不完整或者错误指令将会返回数据。例如①1a ff ff ff

②1c ff ff ff等；可通过ckcmd指令进行开启关闭返回数据。（在正常情况下建议先将屏幕报错原因找到解决了，再指令关闭返回数据）

1.1 **以文本控件显示为例**

单片机如何控制屏幕(文本控件)

1、在上位机工程新建一个文本控件，假设为t0，将程序下载到串口屏上，
2、串口屏串口与单片机串口连接，两者波特率应一致，单片机RX接串口屏TX，单片机TX接串口屏RX。
3、发送指令：单片机串口通过字符串模式发送t0.txt="六六六"
4、发送结束符：单片机通过HEX模式发送0xff 0xff 0xff
5、此时屏幕上的t0控件内的文字变为“六六六”

1.2 **以数字控件显示为例**

单片机如何控制屏幕(数字控件)

1、在上位机工程新建一个数字控件，假设为n0，将程序下载到串口屏上，

2、串口屏串口与单片机串口连接，两者波特率应一致，单片机RX接串口屏TX，单片机TX接串口屏RX。

3、发送指令：单片机串口通过字符串模式发送n0.val=666

4、发送结束符：单片机通过HEX模式发送0xff 0xff 0xff

5、此时屏幕上的n0控件内的文字变为“666”

1.3 **单片机发送变量到屏幕**

在通常情况下单片机是很少发送一个常量给屏赋值的，大多数情况都是单片机赋值一个变量到屏幕上的。下面代码以C语言为例

```
printf("n0.val=666"); 发送命令
printf("\xff\xff\xff"); 发送结束符
printf("n0.val=666\xff\xff\xff");
printf("n0.val=%d\xff\xff\xff",MyData); 一次性发完命令和结束符
printf("t0.txt="%d"\xff\xff\xff",MyTxt); 一次性发完命令和结束符
```

注：这里发送16进制是用\xff，若不明白"\"使用法，自行百度"c语言转义字符"。

二、串口数据解析模式之主动解析模式

在默认情况下屏接收设备发送数据完整格式为字符串指令加上3个16进制ff，如果将屏设置为recmod=1，那么屏将进入主动解析模式，即自定义通信协议。

具体使用详情，点击查看。

注：1.在主动解析模式下，只有定时器控件能够解析数据，定时器最快解析数据时间为50ms。

2.在主动解析模式下，所有串口指令都不会执行，都会存放串口缓冲区中，等待您主动去读取。

3.常规情况下，建议使用的是屏默认通信协议，如果设备实在是没有按照屏通信协议格式，再进行主动解析模式。

三、串口屏串口发送数据

串口屏串口发送数据是使用get prints printh指令。

注：1.同一个事件里分行写发送指令，实质是同一行发送的。

2.发送变量只能使用prints,get指令，发送16进制数据只能使用printh指令。

3.屏幕可通过printh指令,配合prints指令实现任何通信协议。