ZHD300 手持盒手册

Version 1.2

版 权 说 明

Zmotion®

本手册中的信息资料仅供参考。由于改进设计和功能等原因，正运动公司保留对本资料的最终解释权！内容如有更改，恕不另行通知！

调试机器要注意安全！请务必在机器中设计有效的安全保护装置，并在软件中加入出错处理程序，否则所造成的损失，正运动公司没有义务或责任对此负责。

目 录

第一章 硬件介绍

ZHD300 是一款开放式的可编程示教盒，通过 ZDevelop 软件和 Basic 语法来开发界面程序，可以在线跟踪调试，开发简单。

示教盒带有一个串口和一个 USB 口，一个 U 盘口，24V 电源(可 USB 供电)，带有 480*272 分辨率的真彩显示屏，47 个按键，配急停开关。

ZHD300 支持触摸屏，可以按键和触摸配合使用。

1.1 ZHD300 图

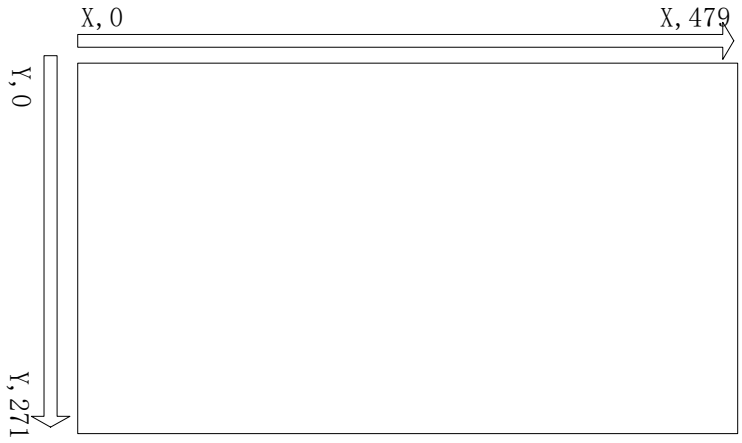
1.2 按键编码

按键的编码按行列组合而成， 键值 = 行号(1-10)×10+ 列号(1-5)；启动按键编码为 1，急停按键编码为 5.

 对定制或其他的按键面膜，按键的位置和键值是不一样的，具体请联系厂家获取。

1.3 显示屏点位坐标。

显示屏的点阵为 480*272. 坐标原点在左上角。



1.3.1 字体

需要在工程文件中加入字体文件。
有多个字体文件时， 可以通过函数切换当前字体，可以使用下面的工具制作字体文件。

1.3.2 图片文件

用到的图片文件请加入工程。

1.4 开发过程

1.4.1 驱动安装

与电脑连接，需要安装驱动程序 VCP_V1.3.1_Setup， 64 操作系统选用另外一个。
部分 WIN7 精简系统缺少文件，安装步骤：

先安装 VCP_V1.3.1_Setup ，可能会提示出错

然后把 usbser.sys 拷贝到 C:\WINDOWS\system32\drivers\

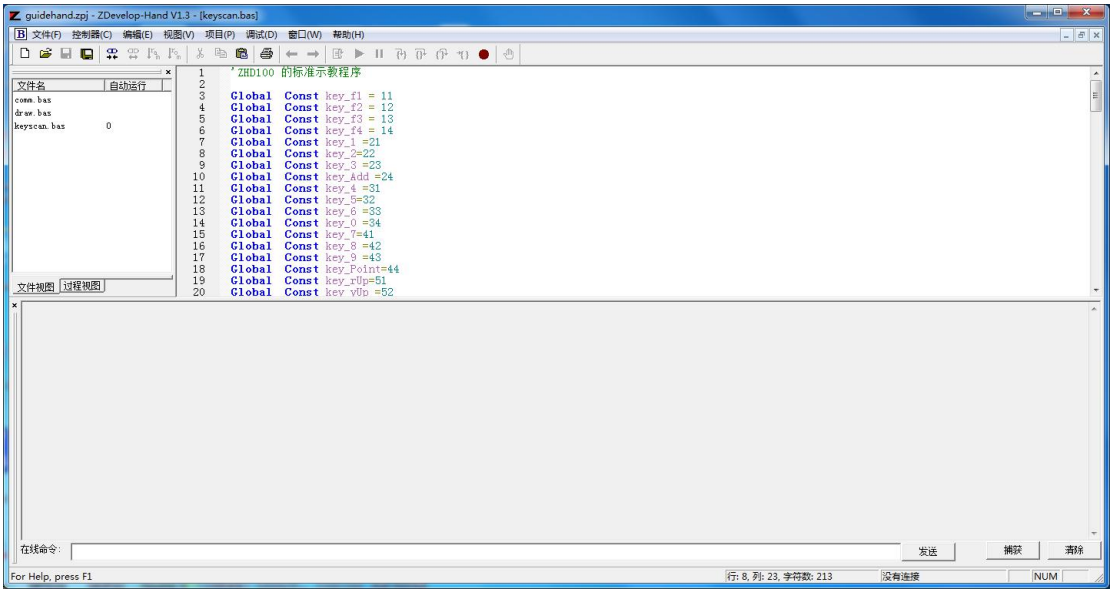
然后把 mdmcpq.inf 拷贝到 C:\WINDOWS\system32\

1.4.2 调试接口

底部的 miniUSB 接口可以用来调试。
此接口可以直接供电，可以用来升级下载程序，固件升级，或是调试运行。

1.4.3 软件开发

请采用专用版本的 ZDevelopHD 工具。



1.4.4 仿真器

开发软件带仿真器，通过“菜单”-“连接仿真器”来打开。

第二章 通用指令

2.1 语法

2.1.1’

类型： 特殊字符
描述： 后面全部是注释，直到下一行开始。

2.1.2_

类型： 特殊字符
描述： 后面换行继续。

在条件判断语句、内存存储、打印输出时尽量不要使用.

2.1.3 CONST

类型： 语法指令
语法： CONST CVARNAME = value
别名：无
描述：定义符号表示的常数，从而避免直接用数值表示。
参数： CVARNAME 常量名
value 常量值
举例： CONST MAX_VALUE = 100000’定义常量

2.1.4 DIM

类型： 语法指令
语法： DIM VarName, ArrayName(space)
描述： 定义文件模块变量，数组；当变量不定义就赋值时，会自动定义为文件模块变量。
文件模块变量只能在本程序文件内部使用。
数组可以作为字符串使用。
参数： VarName 变量名称
ArrayName 数组名称
space 数组长度
参见: MODBUS_STRING, VRSTRING 等字符串函数
举例：

```
DIM ARRAY1(100)’定义数组 ARRAY1
DIM VAR1 ’定义变量 VAR1
VAR2 = 100 ‘赋值语句会自动定义文件模块变量。
ARRAY1 = “asdf”
ARRAY1(0,100,200,300) ‘对数组进行连续赋值.
```

2.1.5 LOCAL

类型： 语法指令
语法： LOCAL localname
描述： 定义局部变量，局部变量。
参数： localname 局部变量名
举例：

```
LOCAL v1’定义变量 V1
```

同一个 SUB 的局部变量有个数限制，SUB 过程的输入参数自动转化为局部变量。

 不同任务的同一个 SUB 过程调用生成不同的局部变量，同一个任务的 SUB 过程递归调用也生成不同的局部变量。

2.1.6 GLOBAL

类型： 语法指令
语法 1： GLOBAL VAR1
语法 2： GLOBAL SUB SUB1()
语法 3： GLOBAL CONST CVARNAME = value
描述： 定义全局变量，数组；定义全局 SUB 过程。
参数：
VAR1 变量名称
SUB1 过程名称
CVARNAME 常量名称
value 常量值
举例：

```
GLOBAL SUB g_sub2() ’ 定义全局过程 g_sub2，可以在任意文件中使用。
GLOBAL CONST g_convar = 100 ‘定义全局使用的常量
GLOBAL g_var2 ‘定义全局变量 g_var2
```

2.1.7 SUB

类型：语法指令
语法：SUB label([para1] [,para2]…)
END SUB
描述：用户自定义 SUB 过程，可以前面增加 GLOBAL 描述以定义全局使用的 SUB 过程。
参见：GOSUB, GLOBAL
参数：
label: 过程名称，不能与现有的关键词冲突。
para1: 过程调用时传入的参数，自动作为 LOCAL 局部变量。
para2: 过程调用时传入的参数，自动作为 LOCAL 局部变量。
…
举例：

```
SUB sub1() ’ 定义过程 SUB1，只能在当前文件中使用。
End sub
GLOBAL SUB g_sub2() ’ 定义全局过程 g_sub2，可以在任意文件中使用。
End sub
```

2.1.8 GSUB

类型：语法指令
语法：GSUB label([char1] [,char2]…)
End sub
描述：用户自定义 GSUB 过程，可以前面增加 GLOBAL 描述以定义全局使用的 GSUB 过程，GSUB 过程调用的时候采用 G 代码语法，不要加括号。
参见：GOSUB, GLOBAL
参数：
label: 过程名称，不能与现有的关键词冲突。
char1: 过程调用时传入的字母参数，自动作为 LOCAL 局部变量。
char2: 过程调用时传入的字母参数，自动作为 LOCAL 局部变量。
…
举例：

```
G01 X100 Y100 Z100 U100 ’调用 G01
GLOBAL GSUB G01(X, Y, Z, U)’ 定义 GSUB 过程 G01
...
End sub
```

2.1.9 :

类型：语法结构
语法：label:
描述：用户过程标号，标号可以作为无参数的 SUB 过程来使用。
参见：SUB, GOTO, GOSUB
参数：
label: 标号名称，不能与现有的关键词冲突。

举例：

```
goto label1

label1: ’ 加：定义标号
end
```

2.1.10 DMINS

类型：语法指令
语法：DIMINS ArrayName(pos)
别名：无

描述：数组的链表操作，插入后当前元素以及后面的所有元素往后移动一个位置。
参数： ArrayName 数组名称
pos 数组索引
举例：
DMINS ARRAY1(0) ‘插入元素 0，原有的所有元素都想后移动一个位置

 谨慎对超长数组进行操作，特别是数组 TABLE。

2.1.11 DMDEL

类型：语法指令
语法：DMDEL ArrayName(pos)
别名：无
描述：数组的链表操作；删除数组元素，删除后当前元素后面的所有元素向前移动。
参数： ArrayName 数组名称
pos 数组索引
举例：
DMDEL a(0)’ 删除数组 a 的第 1 个元素

 谨慎对超长数组进行操作，特别是数组 TABLE。

2.1.12 DMCPY

类型：语法指令
语法：DMCPY ArrayDesName(startpos) , ArraySrcName(startpos)[, size]
别名：无
描述：数组拷贝，从 Src 拷贝到 Des。
参数： ArrayName 数组名称
startpos 数组起始索引
size 拷贝的个数，超过最大值会自动缩减
举例：
dmcpy aaa(0), bbb(0),100

 谨慎对超长数组进行操作，特别是数组 TABLE。

2.1.13 IF

类型：程序结构
语法： IF <condition1> THEN
commands
ELSEIF <condition2> THEN
commands
ELSE
commands
ENDIF
描述：条件判断，同标准 BASIC 结构。
参数： condition1 条件
condition2 条件
举例 1：

```
dim a ’ 定义变量
a=12 ’ 赋值
if a>11 then ‘条件判断
trace "the val a is bigger then 11"
elseif a<11 then
trace "the val a is less then 11"
end if
```

举例 2：

```
if in(0) then out(0,on)  ’当单行时可以不用 endif
```

2.1.14 THEN

类型：程序结构
参见：IF

2.1.15 ENDIF

类型：程序结构
参见：IF

2.1.16 ELSEIF

类型：程序结构
参见：IF

2.1.17 FOR

类型：程序结构
语法： FOR variable=start TO end [STEP increment]
...
'block of commands
...
NEXT variable

描述：循环语句，采用标准 BASIC 语法。
参数： variable 变量名称
start 起始循环值
end 结束循环值
increment 循环步进增量，可选
举例：

```
local a
for a=1 to 100' 1 循环至 100
  print a, '打印 a
  a=a+1
next
```

2.1.18 TO

类型：程序结构
参见：FOR

2.1.19 STEP

类型：程序结构
参见：FOR

2.1.20 NEXT

类型：程序结构
参见：FOR

2.1.21 GOSUB

类型：程序结构
语法：GOSUB label
别名：CALL
描述：SUB 过程调用，只能调用本文件的 SUB 过程或全局 SUB 过程。
直接调用 SUB 过程时，可以省掉 GOSUB 语句。
SUB 过程没有参数时，可以省掉括号()
GOSUB 后当前的内容会压栈，不能在调用的 SUB 程序中访问当前的局部变量，RETURN 返回时出栈。
举例：

```
GOSUB sub1()
sub2(1,2) ‘传入 1 给 para1，2 给 para2
call sub3
END

SUB sub1()
  a=100
  print “sub1”
return

SUB sub2(para1,para2)
  a=200
  print “sub2”,para1,para2
return

GLOBAL SUB sub3() ’ 可以在另外一个程序文件中
  a=300
  print “sub3”
return
```

2.1.22 CALL

类型：程序结构
参见：GOSUB

2.1.23 GOTO

类型：程序结构

语法：GOTO label
描述：强制跳转，与 GOSUB 的区别是 GOTO 不会压栈。
举例：

```
a=100
goto label1    ’强制跳转至 label1
a=1000
label1:
print a     ‘结果是 a=100
end
```

2.1.24 ON GOSUB

类型：程序结构
语法：ON expression GOSUB label
描述：当 expression 条件为真时，调用过程 label。
举例：

```
  a=100
  on a>10 gosub label1     ’ a>10 时调用 label 过程
  a=1000
  print a
  end

  label1:
  print a
  return   ‘SUB 过程要返回
```

2.1.25 ON GOTO

类型：程序结构
语法：ON expression GOTO label
描述：条件跳转，当 expression 条件为真时，跳转。
举例：

```
a=100
on a>10 goto label1
a=1000

label1:
print a
end     ‘goto 跳转无法 return 返回。
```

2.1.26 REPEAT

类型：程序结构
语法：REPEAT commands UNTIL condition
描述：循环语句，循环执行 commands，condition 为真时退出循环。
举例：

```
a=0
repeat
print a
a=a+1
delay(1000)
until IN(4)=ON  ‘直到输入 4 有效
```

2.1.27 WHILE

类型：程序结构
语法： WHILE condition
…
WEND
别名：无
描述：循环，condition 条件满足时执行循环
参数： condition 条件
例子：

```
a=0
while NOT IN(4)=ON  ’  直到输入 4 有效
a=a+1
print a
delay(1000)
wend
```

2.1.28 WEND

类型：程序结构
参见：WHILE

2.1.29 RETURN

类型：程序结构

语法：RETURN
别名：END SUB
参见：SUB
描述：用户自定义 SUB 过程返回。
举例：

```
sub sub1()
  …
return
```

2.1.30 UNTIL

类型：程序结构
参见：REPEAT WAIT

2.1.31 DELAY

类型： 语法指令
语法：DELAY(delay time)
别名：WA
描述：延时 delay time，单位毫秒。
参数： delay time 毫秒数
举例： DELAY(100) ‘延时 100ms

2.1.32 WAIT UNTIL

类型：程序结构
语法：WAIT UNTIL condition
描述：等待直到条件满足。
举例：

```
WAIT UNTIL i>5
```

2.2 数学函数

ZBASIC 支持标准 BASIC 的所有运算符，也采用标准 BASIC 的优先级。

优先级顺序：算术运算符 > 比较运算符 > 计算逻辑运算符。相同优先级的运算符采用从左到右的顺序计算。

算术运算符		比较运算符		逻辑运算符	
描述	符号	描述	符号	描述	符号
求幂	^	等于	=	逻辑非	Not
负号	-	不等于	<>	逻辑与	And
乘	*	小于	<	逻辑或	Or 或 |
除	/	大于	>	逻辑异或	Xor
整除	\	小于等于	<=	逻辑等价	Eqv
求余	Mod 或 %	大于等于	>=		
加	+				
减	-				

2.2.1 +

类型： 运算符
语法：expression1 + expression2
描述： 将两个表达式相加
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：

```
>>print 1+2
3
```

2.2.2 -

类型： 运算符
语法：expression1 - expression2
描述： 将表达式 1 减去表达式 2
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：

```
>>print 2-(2-1)
1
```

2.2.3*

类型： 运算符
语法：expression1 * expression2
描述： 将表达式 1 与表达式 2 相乘
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：

```
>>print 10*(1+2)
30
```

2.2.4/

类型： 运算符
语法：expression1 / expression2
描述： 表达式 1 除以表达式 2
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：

```
>>print 10/5
2
```

2.2.5=

类型： 运算符
语法：expression1 = expression2
描述： 比较运算符：如果表达式 1 等于表达式 2，返回 TRUE，否则返回 FALSE
赋值运算符：将表达式 2 赋值给前面的变量或参数等。
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例 1：

```
ON IN(0)=ON GOTO label1
```

如果输入通道 0 为 ON，程序跳转到标识"label1:"开始的行开始执行。

举例 2：

VAR1 = 100

修改 VAR1 为 100.

2.2.6 <>

类型： 运算符

语法：expression1 <> expression2

描述： 如果表达式 1 不等于表达式 2，返回 TRUE，否则返回 FALSE。

参数：

expression1: 任意有效的表达式

expression2: 任意有效的表达式

举例：

ON MODBUS_BIT(0)<>0 GOTO label1

如果 MODBUS 位寄存器 0 非零值，程序跳转到标识"label1:"开始的行开始执行。

2.2.7 >

类型： 运算符

语法：expression1 > expression2

描述： 如果表达式 1 大于表达式 2，返回 TRUE，否则返回 FALSE。

参数：

expression1: 任意有效的表达式

expression2: 任意有效的表达式

举例：

WAIT UNTIL MPOS>100

该条语句将使程序循环等待，直到测量反馈位置大于 100 为止。

2.2.8 >=

类型： 运算符

语法：expression1 > expression2

描述： 如果表达式 1 大于或等于表达式 2，返回 TRUE，否则返回 FALSE。

参数：

expression1: 任意有效的表达式

expression2: 任意有效的表达式

举例：

VAR1=1>=0

因为 1 大于 0，所以 VAR1 的值将等于 TRUE (-1)。

2.2.9 <

类型： 运算符
语法：expression1 < expression2
描述： 如果表达式 1 小于表达式 2，返回 TRUE，否则返回 FALSE。
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：
VAR1=1<0
因为 1 大于 0，所以 VAR1 的值将等于 FALSE(0)。

2.2.10 <=

类型： 运算符
语法：expression1 > expression2
描述： 如果表达式 1 小于或等于表达式 2，返回 TRUE，否则返回 FALSE。
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：
VAR1=1<=1
VAR1 的值将等于 TRUE (-1)。

2.2.11 \

类型： 运算符
语法：expression1 \ expression2
描述：整数除法。
参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：
>>print 10 \ (1+2)
3

2.2.12 $

类型： 特殊字符
语法：$hexnum
描述：表示紧接着的数据是 16 进制格式。
举例：
>>print $F
15

2.2.13 AND

类型： 运算符
语法：expression1 AND expression2
描述：按位与操作符，只操作整数部分。

AND		结果
0	0	0
0	1	0
1	0	0
1	1	1

参数：
expression1: 任意有效的表达式
expression2: 任意有效的表达式
举例：
>>print 1 and 2
0

2.2.14 OR

类型： 运算符
语法：expression1 OR expression2
描述：按位或操作符，只操作整数部分。

OR		结果
0	0	0
0	1	1

1	0	1
1	1	1

举例：

```
>>print 1 and 2
3
```

2.2.15 NOT

类型： 运算符

语法：NOT expression2

描述：按位非操作符，只操作整数部分。

举例：

举例：

```
>>print NOT 1
-2
```

2.2.16 XOR

类型： 运算符

语法：expression1 XOR expression2

描述：逻辑异或操作符，只操作整数部分。

XOR		结果
0	0	0
0	1	1
1	0	1
1	1	0

举例：

```
>>print 1 xor 1
0
```

2.2.17 EQV

类型： 运算符

语法：expression1 EQV expression2

描述：按位同或操作符，只操作整数部分。

EQV		结果
0	0	1
0	1	0
1	0	0
1	1	1

举例：

```
>>print 1 eqv 1
1
```

2.2.18 MOD

类型： 运算符
语法：expression1 MOD expression2
描述：求余数。
参数：
expression1: 任意有效的表达式，取整数部分。
expression2: 任意有效的表达式，取整数部分。
举例：

```
>>print 10 MOD (1+2)
1
```

2.2.19 ABS

类型： 数学函数
语法：ABS(expression)
描述：求绝对值。
参数： expression 任意有效的表达式
举例：

ABS(-11) ’ 结果是 11

2.2.20 TAN

类型： 数学函数
语法：TAN(expression)
描述：求正切三角函数，输入参数为弧度单位。。
参数： expression 任意有效的表达式

举例： tan(pi/3) ‘结果是 1.732

2.2.21 ATAN

类型： 数学函数
语法：ATAN(expression)
描述：求反正切三角函数，返回值为弧度单位。
参数： expression 任意有效的表达式
举例： atan(1) ’结果是 0.7854 = (45/180)*pi

2.2.22 ATAN2

类型： 数学函数
语法：ATAN2(y, x)
描述：反正切三角函数，返回值为弧度单位
参数： expression 任意有效的表达式
举例：

```
print atan2(1,0)   ‘结果是 1.5708
```

2.2.23 ASIN

类型： 数学函数
语法：ASIN(expression)
描述：反正弦三角函数，返回值为弧度单位
参数： expression 任意有效的表达式
举例：print asin(0.5) ’结果是 0.52360

2.2.24 SIN

类型： 数学函数
语法：SIN(expression)
描述：正弦三角函数，输入参数为弧度单位。
参数： expression 任意有效的表达式
举例：print sin(pi/6) ’ 结果是 0.5000

2.2.25 COS

类型： 数学函数
语法：COS(expression)
描述：余弦三角函数，输入参数为弧度单位。
参数： expression 任意有效的表达式
举例：print cos(pi/3) ’结果是 0.5000

2.2.26 ACOS

类型： 数学函数
语法：ACOS(expression)
描述：反余弦三角函数，返回值为弧度单位
参数： expression 任意有效的表达式
举例：print acos(0.5) ’结果是 1.04720

2.2.27 EXP

类型： 数学函数
语法：exp([base,] expvalue)
描述：指数函数
参数： base 底数，缺省为 e
expvalue 指数
举例 1：
print exp(2,4) ‘结果是 16 (2*2*2*2)
举例 2：
print exp(1) ‘结果是 2.7183

2.2.28 SQR

类型： 数学函数
语法：SQR(expression)
描述：平方根函数
参数： expression 任意有效的表达式
举例： a= sqr(4)
print a ‘结果是 2

2.2.29 LN

类型： 数学函数
语法：LN(expression)
描述：自然对数函数
参数： expression 任意有效的表达式
举例： a= ln(1)
print a ‘结果是 0

2.2.30 LOG

类型： 数学函数
语法：LOG(expression)
描述：对数，底数为 10
参数： expression 任意有效的表达式
举例： a= log(100)
print a ‘结果是 2

2.2.31 CLEAR_BIT

类型： 数学指令或函数
指令语法：CLEAR_BIT(bit#,vr#)
函数语法：CLEAR_BIT(bit#,int)
描述： 位操作，只对整数，修改对应位为 0；函数使用时返回操作后的结果。
参数： bit#, 位编号：0-31
vr# 要操作的 VR 变量编号
int 要操作的表达式，取整数部分
举例 1：

```
a=1
print clear_bit(0,a) ’ 返回清除 a 的第 0 位后的结果，a 的值不变。
0 ’ 结果为 0，a 为 1
Print a
1
```

举例 2：

```
clear_bit (0,23)
VR(23)的第 0 位将被清除(设置为 0)
```

2.2.32 READ_BIT

类型： 数学函数
语法：READ_BIT(bit#, vr#)
描述：位操作，只对整数，读取对应位状态
参数： bit#, 位编号：0-31
vr# 要操作的 VR 变量编号
举例：

```
dim a
a=read_bit(0,13)’读取 vr(13)的第 0 位,赋值给 a
print a‘输出 a 值
```

2.2.33 READ_BIT2

类型： 数学函数，20130813 以后的固件版本提供了支持。
语法：READ_BIT2(bit#, int)
描述：位操作，只对整数，读取对应位状态
参数： bit#, 位编号：0-31
int 要操作的表达式，取整数部分
举例：

```
dim a,b
b=1
a=read_bit2(0,b)’读取 b 的第 0 位,赋值给 a
print a‘输出 a 值
```

2.2.34 SET_BIT

类型： 数学指令或函数
指令语法：SET_BIT(bit#, vr#)
函数语法：SET_BIT(bit#,int)
描述：位操作，只对整数，修改对应位为 1
参数： bit#, 位编号：0-31
vr# 要操作的 VR 变量编号
int 要操作的表达式，取整数部分
举例 1：

```
a=1
print set_bit (1,a) ’ 返回设置 a 的第 0 位后的结果，a 的值不变。
3
Print a
1
```

举例 2：

```
set_bit (0,23)    ‘VR(23)的第 0 位将改为 1
```

2.2.35 FRAC

类型： 数学函数
语法：FRAC(expression)
描述：返回小数部分，总是大于 0 的部分。
参数： expression 要操作的数
举例：

```
a=frac(1.235)
print a ‘结果是 0.235
```

2.2.36 INT

类型： 数学函数
语法：INT(expression)
描述：返回整数部分。
参数： expression 任意有效的表达式
举例：

```
a=int(1.235)
print a ‘结果是 1
?int(-1.1)
-2’ 因为小数部分总处理为正数。
```

2.2.37 SGN

类型： 数学函数
语法：SGN(expression)
描述：返回符号。
1 大于 0
0 等于 0
-1 小于 0
参数： expression 任意有效的表达式
举例：

```
a=SGN(-1.235)
print a ‘结果是-1
```

2.2.38 B_SPLINE

类型： 数学函数
语法：B_SPLINE(type, data_start, points, data_out, ratio)
描述：将 TABLE 中的数据进行 B 样条平滑。
参数：

type	类型，目前只支持 1-B 样条。
data_start	图形数据在 TABLE 中的起始位置。
points	图形数据的个数。
data_out	平滑后的图形数据在 TABLE 中起始位置。
ratio	B_SPLINE 函数的平滑比率。平滑后的个数为 points * ratio.

举例：
举例： B_SPLINE(1,0,10,100,10) ‘平滑一个 10 点的图形数据，源图形点在 Table 的位置为从 0 到 9，平滑为 100 点的数据，并且平滑后的数据从 Table 地址 100 开始存放。

2.2.39 CHR

类型： 字符串函数
语法：CHR(expression)
描述：返回 ASCII 打印，只用于 PRINT。
参数： expression 任意有效的表达式
举例：

```
>>PRINT CHR(66);
   B
```

2.2.40 TOSTR

类型： 字符串函数
语法：TOSTR(VAR1, [N],[DOT])
描述：格式化输出函数。
参数：VAR1 任意有效的表达式

N	输出总位数，包括小数点和符号位。当 N 设置负数时，表示右对齐。
DOT	输出的小数个数，当 N 太小没有小数位置时，不输出小数。

举例：

```
>> print tostr(-100,6,2)
  -100.0
```

2.2.41 HEX

类型： 字符串函数
语法：HEX(expression)
描述：返回十六进制格式，只用于 PRINT。
参数： expression 任意有效的表达式，只取整数部分。
举例：

```
>>PRINT HEX (15);
   f              ‘十六进制
```

2.2.42 STRCOMP

类型： 字符串函数
语法：STRCOMP(str1, str2)
描述：字符串比较函数，根据两个字符串的情况返回 >0 =0 <0。
参数： str1 字符串 1
Str2 字符串 2
举例：

```
DIM  AAA(10)
AAA = “abc”
>>PRINT strcomp(AAA, “abc”)
   0
```

2.2.43 IEEE_IN

类型： 数学函数
语法： IEEE_IN(byte0,byte1,byte2,byte3)
描述： 把四个字节组合成一个单精度浮点数。
参数： byte0 - byte3 四个字节
举例：

```
VAR = IEEE_IN(VR(10),VR(11),VR(12),VR(13))
```

2.2.44 IEEE_OUT

类型： 数学函数
语法： byte_n = IEEE_OUT(VAR, n)
描述： 从一个单精度浮点数里面取一个字节。
参数： VAR 单精度浮点数

N 0-3， 取第几个字节

举例：

VAR = IEEE_OUT(VR(1),2)

2.2.45 PI

类型： 常数

：3.14159

2.2.46 TRUE

类型： 常数

：-1

2.2.47 FALSE

类型： 常数

：0

2.2.48 ON

类型： 常数

：1

2.2.49 OFF

类型： 常数

：0

2.3 通讯

2.3.1 SETCOM

类型：通讯指令

语 法 ： SETCOM (baudrate,databits,stopbits,parity,port[,mode] [,variable] [,timeout])

描述：串口的配置，对示教手柄，缺省为 MODBUS 主端。
参数：baudrate　　串口波特率：9600 19200 4800 115200 38400(缺省)
databits　　数据位数 8
stopbits　　停止位：只能设置 0/1/2
parity　　是否校验：

值	描述
0（缺省）	无校验
1	奇校验
2	偶校验

port　　串口 PORT 编号 0
mode　　协议：

值	描述
0	RAW 数据模式，无协议 此时可以使用 GET,PRITNT #
4（缺省）	MODBUS 从端（16 位整数）
14	MODBUS 主端（16 位整数）

variable　　寄存器选择，0-VR，1-TABLE，2-系统 MODBUS 寄存器。

当设置为 VR 或 TABLE 时，通用输出口会映射到 MODBUS_BIT(0)的位置，输入口会映射到 MODBUS_BIT(1000)的位置，因此此时不要使用 MODBUS_BIT 作为触摸屏的按钮等。

variable 参数是全局的设置，所有的端口共一个。

值	描述
0	VR
1	TABLE
2（缺省）	系统 MODBUS 寄存器

timeout　　mode=14 时为消息超时时间，毫秒单位，缺省值 1000
参见：ADDRESS，PROTOCAL，MODBUS_STRING，MODBUS_IEEE，　PORT
举例：

```
setcom(38400, 8,1,0,0,4,2)　　’设置串口 0 为 modbus 从端，波特率 38400
>>?*setcom ’打印出当前的串口配置信息

setcom(38400, 8,1,0,1,14,2,1000)　　‘设置串口 1 为 modbus 主端，波特率 38400
```

2.3.2 MODBUSM

2.3.2.1 MODBUSM_DES

类型：通讯指令
语法：MODBUSM_DES (address[, port]), ADDRESS1 = MODBUSM_DES([port])
描述：设置或读取 MODBUS 主端的对方.
参数：address　对端的 modbus 协议站号
port　　当前 modbus 主通讯的 port 号

参见：ADDRESS，PROTOCAL，PORT，SETCOM
举例：

```
MODBUSM_DES(1,1)     ’设置 485 端口，对方站号 1
```

2.3.2.2 MODBUSM_STATE

类型：通讯状态
语法：VAR1 = MODBUSM_STATE
描述：MODBUS 主通讯状态.

值	描述
0	状态正常
1	等待应答中
2	等待超时
3	应答错误

参数：port 当前 modbus 主通讯的 port 号
参见：PROTOCAL，PORT， SETCOM
举例：

```
Print  MODBUSM_STATE
0
```

2.3.2.3 MODBUSM_REGSET

类型：通讯指令
语法：MODBUSM_REGSET (startreg, num, local_reg)
描述：把本地 MODBUS 寄存器设置到对端.
参数：startreg 对端的寄存器起始编号，从 0 开始。
num 寄存器个数
local_reg 从本地系统 MODBUS 寄存器中取值，起始编号。
参见：ADDRESS，PROTOCAL，PORT，SETCOM
举例：

```
MODBUSM_REGSET (0,10,0) ’把本地寄存器 0-9 复制到通讯对端的寄存器 0-9
```

2.3.2.4 MODBUSM_REGGET

类型：通讯指令
语法：MODBUSM_REGGET (startreg, num, local_reg)
描述：把对端的 MODBUS 寄存器复制到本地
参数：startreg 对端的寄存器起始编号，从 0 开始。
num 寄存器个数

local_reg 从本地系统 MODBUS 寄存器中取值，起始编号。
参见：ADDRESS，PROTOCAL，PORT， SETCOM
举例：
MODBUSM_REGGET (0,10,0) ’把通讯对端的寄存器 0-9 复制到本地寄存器 0-9

2.3.2.5 MODBUSM_BITSET

类型：通讯指令
语法：MODBUSM_BITSET (startreg， num， local_reg)
描述：把本地 MODBUS 位寄存器设置到对端.
参数：startreg 对端的位寄存器起始编号，从 0 开始。
num 寄存器个数
local_reg 从本地系统 MODBUS 寄存器中取值，起始编号。
参见：ADDRESS，PROTOCAL，PORT， SETCOM
举例：
MODBUSM_BITSET (0,10,0) ’把本地位寄存器 0-9 复制到通讯对端的寄存器 0-9

2.3.2.6 MODBUSM_BITGET

类型：通讯指令
语法：MODBUSM_BITGET (startreg， num， local_reg)
描述：把对端的 MODBUS 位寄存器复制到本地
参数：startreg 对端的寄存器起始编号，从 0 开始。
num 寄存器个数
local_reg 从本地系统 MODBUS 位寄存器中取值，起始编号。
参见：ADDRESS，PROTOCAL，PORT， SETCOM
举例：
MODBUSM_BITGET (0,10,0) ’把通讯对端的位寄存器 0-9 复制到本地位寄存器 0-9

2.3.2.7 MODBUSM_BIT

类型：通讯指令
语法：MODBUSM_BIT (startreg,var1,var2,...)
描述：修改从端数据，自动发送消息，可以通过 MODBUSM_STATE 检测结果.
参数：startreg 对端的寄存器起始编号，从 0 开始编号。
Var1,var2 BIT 值 0/1。
参见：ADDRESS，PROTOCAL，PORT， SETCOM
举例：
MODBUSM_BIT(0,0,1) ’修改对端 2 个位数据（2 个 0x 寄存器）

2.3.2.8 MODBUSM_REG

类型：通讯指令
语法：MODBUSM_REG (startreg,var1,var2,...)
描述：修改从端数据，自动发送消息，可以通过 MODBUSM_STATE 检测结果.
参数：startreg　　对端的寄存器起始编号，从 0 开始编号。
　　Var1,var2 reg 值。
参见：ADDRESS，PROTOCAL，PORT，SETCOM
举例：
　　MODBUSM_REG (0,10,10)　’修改对端 2 个 16 位数据（2 个寄存器）

2.3.2.9 MODBUSM_LONG

类型：通讯指令
语法：MODBUSM_RONG (startreg, var1,var2,...)
描述：修改从端数据，自动发送消息，可以通过 MODBUSM_STATE 检测结果.
参数：startreg　　对端的寄存器起始编号，从 0 开始编号。
　　Var1,var2 32 位整数值。
参见：ADDRESS，PROTOCAL，PORT，SETCOM
举例：
　　MODBUSM_LONG (0,10,20)　’修改对端 2 个 32 位数据（4 个寄存器）

2.3.2.10　MODBUSM_IEEE

类型：通讯指令
语法：MODBUSM_BITGET (startreg, var1,var2,...)
描述：修改从端数据，自动发送消息，可以通过 MODBUSM_STATE 检测结果.
参数：startreg　　对端的寄存器起始编号，从 0 开始编号。
　　Var1,var2 32 位浮点数值。
参见：ADDRESS，PROTOCAL，PORT，SETCOM
举例：
　　MODBUSM_IEEE (0,10.0，20.0)’修改对端 2 个 IEEE 数据（4 个寄存器）

2.3.2.11　MODBUSM_STRING

类型：通讯指令
语法：MODBUSM_BITGET (startreg, chares, string)
描述：修改从端数据，自动发送消息，可以通过 MODBUSM_STATE 检测结果.
参数：startreg　　对端的寄存器起始编号，从 0 开始编号。
　　chares　　字符的总个数
　　string　　字符串，可以为表达式。
参见：ADDRESS，PROTOCAL，PORT，SETCOM
举例：
　　MODBUSM_STRING(0,10,　“ABC”)

2.3.3 MODBUS 寄存器

2.3.3.1 MODBUS_BIT

类型： MODBUS 位寄存器
命令语法：MODBUS_BIT(first,[last]) = value
指令语法：MODBUS_BIT(first,[last])
描述：修改或者读取 BIT 寄存器，触摸屏一般叫 0x 寄存器。
ZMOTION 运动控制器专门具备 2048 个 MODBUS 位寄存器。
另：特殊的 modbus 0x 寄存器，通过这些寄存器可以直接从触摸屏读取和设置 IO：

寄存器（zero based）	意义
10000-	输入 IN,每个 IO 占 1 个寄存器.
20000-	输出 OUT

参见：ADDRESS
参数： first 位寄存器编号，编号从 0 开始。
last 位寄存器编号，编号从 0 开始。
举例：

```
DIM VAR
MODBUS_BIT(100) = 1
VAR = MODBUS_BIT(10)
```

2.3.3.2 MODBUS_IEEE

类型： MODBUS 字寄存器
命令语法：MODBUS_IEEE (regnum) = value
函数语法：MODBUS_IEEE (regnum)
描述：修改或者读取字寄存器，32 位浮点数方式。触摸屏一般叫 4x 寄存器。
ZMOTION 运动控制器专门具备 MODBUS 字寄存器，会占用两个寄存器的地址。使用系统 MODBUS 字寄存器时，请设置 SETCOM 的参数为使用系统寄存器。
另：特殊的 modbus 4x 寄存器，通过这些寄存器可以直接从触摸屏读取一些状态：

寄存器（zero based）	类型	意义
10000-	IEEE	DPOS 读取,每个轴占两个寄存器.
11000-	IEEE	MPOS 读取
12000-	IEEE	VP_SPEED 读取

参数： regnum 寄存器编号，编号从 0 开始。
举例：

```
DIM VAR
MODBUS_IEEE(100) = 100.10
VAR = MODEBUS_IEEE(100)
```

2.3.3.3 MODBUS_LONG

类型： MODBUS 字寄存器
命令语法：MODBUS_LONG (regnum) = value
函数语法：MODBUS_LONG (regnum)
描述：修改或者读取字寄存器，32 位整数方式，会占用两个寄存器的地址。触摸屏一般叫 4x 寄存器。
ZMOTION 运动控制器专门具备 MODBUS 字寄存器，使用系统 MODBUS 字寄存器时，请设置 SETCOM 的参数为使用系统寄存器。
参数： regnum 寄存器编号，编号从 0 开始。
举例：

```
DIM VAR
MODBUS_LONG(100)  =  1000
VAR = MODBUS_LONG(100)
```

2.3.3.4 MODBUS_REG

类型： MODBUS 字寄存器
命令语法：MODBUS_REG(regnum) = value
函数语法：MODBUS_REG(regnum)
描述：修改或者读取字寄存器，触摸屏一般叫 4x 寄存器。
ZMOTION 运动控制器专门具备 MODBUS 字寄存器，使用系统 MODBUS 字寄存器时，请设置 SETCOM 的参数为使用系统寄存器。不同型号控制器的字寄存器个数有区别。
参数： regnum 寄存器编号，编号从 0 开始。
举例：

```
DIM VAR
MODBUS_REG(100)  =  100
VAR = MODBUS_REG(100)
```

2.3.3.5 MODBUS_STRING

类型： MODBUS 字寄存器，字符串函数
语法：MODBUS_STRING(index, chares)
描述：读取 MODBUS 寄存器区域的字符串，按字节来读取。触摸屏一般叫 4x 寄存器。
参数：index 起始的 MODBUS 寄存器编号，编号从 0 开始。不同型号控制器的字寄存器个数有区别。
chares 读取的字符总数。
参见：DIM
举例：

```
DIM ARR(8)
>> MODBUS_STRING (0, 8) = “abc”
>>print MODBUS_STRING(0, 8)
Abc
```

```
ARR = MODBUS_STRING(0,8)
```

2.3.4 U 盘更新

通过手持的 U 盘可以升级控制器程序，或是下载控制器工艺文件，此时接口必须为 MODBUS RTU 模式。

2.3.4.1 SEND_ZAR

类型： 通讯指令
语法：SEND_ZAR(“ufilename”)
描述：通过手持的 U 盘升级相连控制器的程序，结果通过 MODBUSM_STATE 查看。
参数：
ufilename U 盘文件名，支持字符串的数组和其它字符串类型.
参见： MODBUSM_STATE, SEND_PERCENT
举例：

```
SEND_ZAR(“1.ZAR”)
```

2.3.4.2 SEND_FLASH

类型： 通讯指令
语法：SEND_FLASH (dir, uid, flashid)
描述：手持的 U 盘和控制器的 FLASH 相互拷贝，结果通过 MODBUSM_STATE 查看。
参数：
Dir 1-U 盘拷贝到控制器 FLASH; 0- 控制器 FLASH 拷贝到 U 盘。
Uid U 盘的文件编号，规则同 U_WRITE。
Flashid 控制器的 FLASH 编号。
参见： MODBUSM_STATE, SEND_PERCENT
举例：

```
SEND_FLASH (1, 1, 1)
```

2.3.4.3 SEND_FILE

类型： 通讯指令
语法：SEND_FILE (dir, ufile, contrFile)
描述：手持的 U 盘和控制器的文件相互拷贝，只支持 BIN 文件和 Z3P 文件，不支持文件功能的控制器会返回失败。
参数：
Dir 1-U 盘拷贝到控制器 FLASH; 0- 控制器 FLASH 拷贝到 U 盘。
ufile U 盘的文件名。

contrFile 控制器的文件名。
参见： MODBUSM_STATE, SEND_PERCENT
举例：
SEND_FILE (1, “1.bin”, “1.bin’)

2.3.4.4 SEND_IFLASH

类型： 通讯指令
语法：SEND_IFLASH (dir, id, flashid)
描述：手持的 FLASH 和控制器的 FLASH 相互拷贝，结果通过 MODBUSM_STATE 查看。
参数：
Dir 1-手持 FLASH 拷贝到控制器 FLASH; 0- 控制器 FLASH 拷贝到手持 FLASH。
id 手持的 FLASH 编号。
Flashid 控制器的 FLASH 编号。
参见： MODBUSM_STATE, SEND_PERCENT
举例：
SEND_FLASH (1, 1, 1)

2.3.4.5 SEND_PERCENT

类型： 发送函数
语法：percent = SEND_PERCENT ()
描述：SEND 等需要长时间完成的指令返回百分比，可以用来显示进度条,范围 0-100。
参见： SEND_ZAR, SEND_FLASH, SEND_FILE

2.3.4.6 ZAR_CONTROL

类型： U 盘函数
语法：id = ZAR_CONTROL (“ufilename”)
描述：查看 U 盘的 ZAR 程序文件的对应控制器类型，此类型在 zdevelop 中设置，避免手持和控制器的文件混淆。
参数：
ufilename U 盘文件名，ZAR 文件.
参见： SEND_ZAR, CONTROL
举例：
Id = ZAR_CONTROL (“1.ZAR”)
If(id/3000 = 3) then ‘ZHD 系列
Print “zhd program.”

```
Endif
```

2.3.5 GET, PUTCHAR

2.3.5.1 GET

类型： 通讯指令
语法：GET #PORT, VARIABLE
描述：从 RAW 方式的通讯口里面读取一个字节，存入一个变量，没有读取到会阻塞，这个函数一般在多任务里面进行调用。
参见： SETCOM, PRINT #
举例：

```
DIM VAR1
SETCOM(38400,8,1,0,0,0)          ‘开启 RAW 方式
GET #0, VAR1
```

2.3.5.2 PUTCHAR

类型： 通讯指令
语法：PUTCHAR #PORT, 字符,
描述：从 RAW 方式的通讯口里面输出字符，可以多个字符。
参见：PORT, SETCOM, GET #
举例：

```
DIM VAR1, ARRAY1(10)
VAR1 = $FE
ARRAY1 = “ABCDEFGHIJ”
SETCOM(38400,8,1,0,0,0)          ‘开启 RAW 方式
PUTCHAR #0, VAR1
PUTCHAR #0, ARRAY1
```

2.3.5.3 PRINT

类型： 系统指令
语法：PRINT #PORT, “字符串”
描述：从 RAW 方式的通讯口里面输出字符串。
参见：PORT, SETCOM, GET #
举例：

```
DIM VAR
VAR = “AAAA”
SETCOM(38400,8,1,0,0,0)          ‘开启 RAW 方式
```

PRINT #0,VAR

2.3.6 其他

2.3.6.1 PRINT

类型： 打印输出函数
语法： PRINT expression, “string”
描述： 打印信息到 PC 端 ZDEVELOP 软件的输出窗口。
通过*特殊字符串可以打印一些特殊的信息。
参数： expression 有效表达式
*SET 打印所有参数值
*TASK 打印任务信息。
*MAX 打印所有规格参数
*FILE 打印程序文件信息。
*SETCOM 打印当前串口的配置信息。
*BASE 打印当前任务的 BASE 列表（140123 以后版本支持）。
*数组名 打印数组的所有元素，数组长度不能太长。
*参数名 打印一个所有轴的单个参数
举例 1：

```
>>print 1+2
3
```

举例 2：

```
>>print *task
Task:0 Running. file:"hmi.bas" line:280:
Task:1 Stopped.
Task:2 Stopped.
Task:3 Stopped.
Task:4 Stopped.
Task:5 Stopped.
Task:6 Stopped.
```

举例 3：

```
>> print *mpos
21872.400 0 0 0 0 0 0 0 0 0 0 0
```

2.3.6.2 TRACE

类型： 打印输出函数
语法： 同 PRINT
描述： 打印信息到 PC 端 ZDEVELOP 软件的输出窗口。

受 ERRSWITCH 开关的控制

举例：

```
ERRSWITCH = 3
Dpos(0) =0
Trace  “dpos(0) =” dpos(0)
结果：dpos(0) = 0
```

2.3.6.3 WARN

类型： 打印输出函数
语法： 同 PRINT
描述： 打印信息到 PC 端 ZDEVELOP 软件的输出窗口。
受 ERRSWITCH 开关的控制
举例：

2.3.6.4 ERROR

类型： 打印输出函数
语法： 同 PRINT
描述： 打印信息到 PC 端 ZDEVELOP 软件的输出窗口。
受 ERRSWITCH 开关的控制

2.3.6.5 ERRSWITCH

类型： 系统参数
语法： ERRSWITCH = switch
描述： 调试输出开关，控制 TRACE WARN ERROR 调试输出语句是否真的输出。
参数： switch 调试输出的开关，
0- TRACE WARN ERROR 指令全部不输出
1- 只输出 ERROR 指令
2- 输出 WARN ERROR 指令
3- TRACE WARN ERROR 指令全部输出

2.3.6.6 PORT

类型：通道修正辅助指令

语法：PORT (portnum)
描述：当访问通道参数时可以指定其他的通道。
参数：portnum 通道号
不同型号的控制器支持的通道数不同。
ZMC1xx 系列，支持同时两个以太网连接，端口号为 502，支持触摸屏的 MODBUS-TCP 协议连接。

通道号	协议
0	RS232 串口
1	RS485 串口
2	以太网接口，连接 1
3	以太网接口，连接 2

ZMC00x 系列

通道号	协议
0	串口 A
1	串口 B

参见：FILE_PORT

2.3.6.7 FILE_PORT

类型： 通道参数
语法：VAR1 = FILE_PORT，FILE_PORT = filenum
描述：返回或设置当前通道的缺省文件号，设定后可以通过在线命令来访问对应文件的文件模块变量或数组。
参数：filenum 程序文件的编号，可以通过 PRINT *FILE 来查看编号。
参见：PORT
举例：
>>FILE_PORT = 0 ’ 当前连接的通道的 FILE_PORT 参数


ZDevelop 集成开发环境会自动使用此参数，用不要在使用 ZDevelop 时修改此参数。

2.3.6.8 PROTOCAL

类型： 通道参数，只读
语法：VAR1 = PROTOCAL
描述：返回当前通道的通讯协议。

值	协议
0	RAW 数据格式，无协议。
3	MODBUS 协议，控制器为从端。（缺省）

10	MODBUS 协议，控制器为主端。
11	预留

参见：PORT，SETCOM，ADDRESS

2.4 存储相关

ZHD300 带有 U 盘接口。

2.4.1 FILE

类型： U 盘文件指令
语法：value = FILE “function” ,…
描述：加载 U 盘文件，或是搜索文件。
参数：function 功能选择

“LOAD_ZAR”	FILE “LOAD_ZAR”，“filename” 加载 U 盘里面的 ZAR 升级文件，升级失败返回 0，同时会 WARN 输出失败原因，升级成功后会自动启动 ZAR 文件，因此返回值 TRUE 没有意义。 升级后原来的调试状态的断点会被清除掉。
“FIND_FIRST”	FILE “FIND_FIRST”, type, vr 搜索 U 盘文件。 Type: 1- 文件 / 2-文件夹 / ".extend" 文件后缀名 vr: VRSTRING(vr)存储查找的结果，超过最大 VR 时，使用 MODBUS_STRING。
“FIND_NEXT”	FILE “FIND_NEXT”, vr vr: VRSTRING(vr)存储查找的结果，超过最大 VR 时，使用 MODBUS_STRING。
“FIND_PREV”	FILE “FIND_PREV”, vr vr: VRSTRING(vr)存储查找的结果，超过最大 VR 时，使用 MODBUS_STRING。
“FLASH_FIRST”	FILE “FLASH _FIRST”, type, vr 搜索 FLASH 文件，只支持 BIN 和 Z3P 文件。 Type: 1- 文件 / 2-文件夹 / ".extend" 文件后缀名 vr: VRSTRING(vr)存储查找的结果，超过最大 VR 时，使用 MODBUS_STRING。
“FLASH _NEXT”	FILE “FLASH _NEXT”, vr vr: VRSTRING(vr)存储查找的结果，超过最大 VR 时，使用

	MODBUS_STRING。
“FLASH _PREV”	FILE “FLASH _PREV”， vr vr: VRSTRING(vr)存储查找的结果，超过最大 VR 时，使用 MODBUS_STRING。
“COPY_FROM”	FILE “COPY_FROM”， “FLASH 文件名”[, “U 盘文件名”] FLASH 文件拷贝到 U 盘，只支持 BIN 和 Z3P 文件。
“COPY_TO”	FILE “COPY_TO”， “U 盘文件名”[, “FLASH 文件名”] U 盘文件拷贝到 FLASH，只支持 BIN 和 Z3P 文件。

举例：

```
>>?FILE "LOAD_zar", "main"   ‘U 盘里面已经存储了 main.zar 升级文件
-1
>>?FILE "FIND_FIRST", ".ZAR" ,0
-1
>>?VRSTRING(0)
AAA.ZAR
```

2.4.2 FLASHVR

类型： 存储指令

语法：FLASHVR (function)

描述：拷贝 RAM 的数据到 FLASH 里面。

参数：function 功能选择

-1	整个 TABLE 都存储到 FLASH，并且上电时自动读取到 TABLE。
-2	取消 FLASH 上电时读取到 TABLE

参见：FLASH_WRITE

举例：

```
FLASHVR (-1)
```

ZMC00x 控制器把 TABLE 数据存储到了 FLASH 的最后一个块，用 FLASH_WRITE, FLASH_READ 指令也可以操作到这个块，要避免冲突。

2.4.3 STICK_WRITE

类型： 存储指令

语法：value = STICK_WRITE(filenum, table_start [,length [,format]])

描述：拷贝 TABLE 的数据到外部存储器里面，value=TRUE 表示操作成功，否则操作失败。

参数：filenum 文件号，对应到 SD【filenum】.BIN

table_start 开始操作的起始 TABLE 号。

length 要操作的 TABLE 元素个数，缺省 128

format 写入文件的格式

值	描述
0（缺省）	float 格式存储
1	文本格式存储

参见：U_WRITE

举例：

STICK_WRITE(0,0,128,0) ‘拷贝 table 前 128 个元素以 float 格式到外部存储器

 不具备外部存储接口的控制器不支持此命令。

2.4.4 STICK_READ

类型： 存储指令

语法：value = STICK_READ (filenum, table_start [,format])

描述：拷贝外部存储器的数据到 TABLE 里面，value=TRUE 表示操作成功，否则操作失败。

参数：filenum 文件号，对应到 SD【filenum】.BIN

table_start 开始操作的起始 TABLE 号。

format 写入文件的格式

值	描述
0（缺省）	float 格式存储
1	文本格式存储

参见：U_READ

举例：

STICK_READ (0,0,128,0) ‘拷贝外部存储器 0.bin 文件数据到 TABLE 中

 不具备外部存储接口的控制器不支持此命令。

2.4.5 STICK_WRITEVR

类型： 存储指令

语法：value = STICK_WRITEVR (filenum, vr_start [,length [,format]])

描述：拷贝 VR 的数据到外部存储器里面，value=TRUE 表示操作成功，否则操作失败。

参数：filenum 文件号，对应到 SD【filenum】.BIN

vr_start 开始操作的起始 VR 号。

length 要操作的元素个数，缺省 128

format 写入文件的格式

值	描述
0（缺省）	float 格式存储
1	文本格式存储

举例：

STICK_WRITEVR (0,0,128,0) ‘拷贝 VR 前 128 个元素以 float 格式到外部存储器

 不具备外部存储接口的控制器不支持此命令。

2.4.6 STICK_READVR

类型： 存储指令

语法：value = STICK_READVR (filenum, vr_start [,format])

描述：拷贝外部存储器的数据到 VR 里面，value=TRUE 表示操作成功，否则操作失败。

参数：filenum 文件号，对应到 SD【filenum】.BIN

vr_start 开始操作的起始 VR 号。

format 文件的格式

值	描述
0（缺省）	float 格式存储
1	文本格式存储

举例：

STICK_READVR (0,0,0) ‘拷贝外部存储器 0.bin 文件数据到 VR

 不具备外部存储接口的控制器不支持此命令。

2.4.7 FLASH_WRITE

类型： 存储指令

语法： FLASH_WRITE sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)]

描述：存储变量或者数组,数组的单个或部分元素到内部 FLASH 里面。

参数： sect_num FLASH 块编号,不同类型不一样，FLASH_SECTES 表示块总数。

varname 变量名

arrayname 数组名，可以为 TABLE，VR。

a 操作的数组索引

length 操作的数组元素个数

参见： FLASHVR，FLASH_READ

例子： FLASH_WRITE 1, VAR, ARRAY1, ARRAY2(1)

 内部 FLASH 采用顺序存储的方式，读取的顺序必须与存储时的顺序一致。

 内部 FLASH 有存储次数限制，不要随意循环操作。

2.4.8 FLASH_READ

类型： 存储指令
语 法 ： FLASH_READ sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)]
描述：从内部 FLASH 读取数据到变量，或数组里面。
参数：sect_num FLASH 块编号,不同类型不一样，FLASH_SECTES 表示块总数。
varname 变量名
arrayname 数组名，可以为 TABLE，VR。
a 操作的数组索引
length 操作的数组元素个数
参见： FLASHVR，FLASH_WRITE
例子： FLASH_READ 1, VAR, ARRAY1, ARRAY2(1)

 内部 FLASH 采用顺序存储的方式，读取的顺序必须与存储时的顺序一致。

2.4.9 FLASH_SECTSIZE

类型： 系统状态函数
语法： value = FLASH_SECTSIZE
描述：读取内部 FLASH 一个块可以存储的变量个数。
参数： value 个数，不同控制器类型不一样。
参见：FLASH_SECTES
举例：

```
? FLASH_SECTSIZE
20480    ‘ZMC1xx 系列
1024     ‘ZMC00x 系列。
```

2.4.10 FLASH_SECTES

类型： 系统状态函数
语法： value = FLASH_SECTES
描述：读取内部 FLASH 的总块数。
参数： value 块数，不同控制器类型不一样。
参见：FLASH_SECTSIZE
举例：

```
?FLASH_SECTES
1000     ‘ZMC1xx 系列。
16       ‘ZMC00x 系列。
```

 对 ZMC00x 系列控制器，FLASHVR 会使用用最后一个块，要避免冲突。

2.4.11 U_WRITE

类型： 存储指令
语 法 ： U_WRITE sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)]
描述：存储变量或者数组,数组的单个或部分元素到外部存储器里面。
参数： sect_num 文件编号，对应到 SD【filenum】.BIN。
varname 变量名
arrayname 数组名，可以为 TABLE，VR。
a 操作的数组索引
length 操作的数组元素个数
参见： FLASHVR，FLASH_WRITE，STICK_WRITE，U_READ
例子： U_WRITE 1, VAR, ARRAY1, ARRAY2(1)

 不具备外部存储接口的控制器不支持此命令。

 文件格式为 32 位 ieee 浮点数顺序存储，一个变量或一个数组元素占用一个浮点数，可以通过 PC 事先做好文件，然后用 U_READ 指令读取。

2.4.12 U_READ

类型： 存储指令
语 法 ： U_READ sect_num [, varname] [, arrayname] [, arrayname(a)] [, arrayname(a,length)]
描述： 从外部存储器读取数据到变量或数组里面。
参数： sect_num FLASH 块编号,不同类型不一样，FLASH_SECTES 表示块总数。
varname 变量名
arrayname 数组名，可以为 TABLE，VR。
a 操作的数组索引
length 操作的数组元素个数
参见： FLASHVR，U_WRITE
例子： U_READ 1, VAR, ARRAY1, ARRAY2(1)

 不具备外部存储接口的控制器不支持此命令。

 文件格式为 32 位 ieee 浮点数顺序存储，一个变量或一个数组元素占用一个浮点数，可以通过 PC 事先做好文件，然后用 U_READ 指令读取。

2.4.13 U_STATE

类型：系统状态函数
语法：U_STATE

描述：检查 U 盘是否插入，有插入返回 TRUE，否则返回 FALSE。
举例：

```
If  U_STATE = TURE then
    U_READ  1, VAR, ARRAY1, ARRAY2(1)
End if
```

 不具备外部存储接口的控制器不支持此命令。

 U 盘里面的文件不要放太多。

2.5 任务相关

ZBASIC 支持实时多任务运行，一个文件上也可以同时运行多个任务。通过 RUN 可以从文件第一行开始运行任务，通过 RUNTASK 可以随意指定 SUB 过程开始运行

2.5.1 RUN

类型： 任务指令
语法：RUN “filename”[, tasknum]
描述：新建一个任务来执行控制器上的一个文件。多任务操作指令有：

END:	当前任务正常结束
STOP:	停止指定文件
STOPTASK	停止指定任务
HALT:	停止所有任务
RUN	启动文件执行
RUNTASK	启动任务在一个 SUB 上执行。

参数： filename 程序文件名，BAS 文件可不加扩展名。
tasknum 任务号，缺省查找第一个有效的。
举例： run “aaa”, 1 ’ 在任务 1 运行 aaa.bas

2.5.2 RUNTASK

类型： 任务指令
语法：RUNTASK tasknum, label
描述：把一个 SUB 过程或是标号作为一个新的任务执行。
参数： tasknum 任务号。
label 自定义 SUB 过程（不能带参数），或标号
举例：

```
runtask 1, taska     ‘启动任务 1 来跟踪打印位置
```

```
MOVE(1000,100)
MOVE(1000,100)
end

taska:   ‘循环打印位置
while 1
 print *mpos
 delay(1000)
wend
end
```

2.5.3 END

类型： 任务指令
描述：当前任务结束。
参见：RUN, RUNTASK

2.5.4 STOP

类型： 任务指令
语法：STOP program name, [tasknum]
描述：程序强制停止，当一个文件有多个任务时，建议用 STOPTASK 指令。
参数： program name 程序文件名，bas 文件可不用带扩展名.
tasknum 任务号,当程序文件有启动多个任务时,缺省任务号最小的任务。
举例：

```
RUN  aaa, 1  ‘执行 aaa.bas
RUN  aaa, 2
STOP  aaa ’ 停止任务 1
STOP  aaa, 2 ’ 停止任务 2
```

2.5.5 STOPTASK

类型： 任务指令
语法：STOPTASK [tasknum]
描述：任务强制停止。
参数： tasknum 任务号，缺省当前任务
举例： STOPTASK 2’ 停止任务 2

2.5.6 HALT

类型： 任务指令
描述：停止全部任务。
举例：

```
HALT                ‘停止所有任务
```

2.5.7 PAUSE

类型： 任务指令
描述：暂停全部任务，一般只有 PC 软件调用，断点断住后也会进入暂停状态。
举例：

```
PAUSE               ‘暂停所有任务
```

2.5.8 PROC

类型： 任务修正辅助指令
语法：PROC(tasknum)
描述：当访问任务参数，任务状态时可以指定其他的任务。
参数：tasknum 任务号
参见：任务参数，任务状态
举例：

```
Print ERROR_LINE PROC(1)          ‘打印任务 1 的错误行
```

2.5.9 PROCNUMBER

类型： 任务特殊状态，系统状态
语法：VAR1 = PROCNUMBER
描述：当前任务的任务号，当程序不知道当前任务号的时候，通过这个来访问。
这个状态不能通过 PROC 来修正。
参见：PROC
举例

```
Print PROCNUMBER          ‘打印当前任务号
```

2.5.10 ERROR_LINE

类型： 任务状态
语法：VAR1 = ERROR_LINE

描述：当前任务的错误行号，。
参见：PROC
举例：

```
Print  ERROR_LINE          ‘打印当前任务错误行
Print  ERROR_LINE(1)       ‘打印任务 1 错误行
```

2.5.11 PROC_LINE

类型： 任务状态
语法：VAR1 = PROC_LINE
描述：任务的当前行号，。
举例：

```
Print proc_line (0)        ‘打印任务 0 运行到那一行
>>print PROC_LINE
100
```

2.5.12 PROC_STATUS

类型： 任务状态
语法：VAR1 = PROC_STATUS
描述：当前任务的状态。

0 任务停止
1 任务正在运行
3 任务暂停中

举例：

```
Print PROC_STATUS(0)          ’ 打印任务 0 状态
>>print PROC_STATUS(0)
1                             ‘运行中
```

2.5.13 RUN_ERROR

类型： 任务状态
语法：VAR1 = RUN_ERROR
描述：任务错误号
举例：

```
Print run_error(1)            ‘打印任务 1 出错号
0
```

2.5.14 TICKS

类型： 任务参数
语法：VAR1 = TICKS，TICKS = value
描述：当前任务的计数周期数，每个周期减一。
每个任务都有独立的 TICKS 参数，ZMC00x 系列，ZMC1xx 系列的周期为 1 毫秒。
举例：

```
ticks = 1000
wait until  ticks  <  0      ‘等待 ticks  < 0 后
move(100)
```

2.6 系统

2.6.1 APP_PASS

类型： 系统指令
语法：APP_PASS(pass)
描述：用户可以给控制器设置一个应用密码，下载 ZAR 包时可以选择校验这个密码，校验错误将不能下载。LOCK 后不能修改 APP_PASS。
参数：pass 字母或数字，”_”等少数特殊符号，总共不要超过 16 个字符。
举例：

```
APP_PASS(Zmotion)
```

APP_PASS 采用不可逆算法加密，一旦忘记，将无法获知。

2.6.2 CONTROL

类型：系统只读参数
语法：CONTROL
描述：返回控制器型号。
举例：

```
>>print CONTROL
3100
```

2.6.3 LOCK

类型：系统指令
语法：LOCK (pass)
描述：锁定控制器，不让调试修改。
参数：pass 字母或数字，”_”等少数特殊符号，总共不要超过 16 个字符。
参见：UNLOCK
举例：

```
LOCK(“passwd”)
```

LOCK 密码采用不可逆算法加密，一旦忘记，将不可获知。

2.6.4 TSIZE

类型： 系统只读参数
语法：TSIZE
描述：TABLE 的所有元素个数。
参见：TABLE
举例：

```
Print TSIZE
```

2.6.5 UNLOCK

类型：系统指令
语法：UNLOCK (pass)
描述：解锁控制器。
参数：pass LOCK 时的密码。
参见：LOCK

LOCK 密码采用不可逆算法加密，一旦忘记，将不可获知。

2.6.6 VERSION

类型： 系统状态
语法：VAR1 = VERSION
描述：系统软件版本号。

第三章 显示与按键指令

3.1 KEY_SCAN

类型： 按键指令
语法：value=KEY_SCAN()
描述：读取当前按下的按键编码，按下返回按键编码，当松开的时候按键编码的负数，如果按键被输入框处理，则返回的按键编码增加 1000，返回 0 表示没有按键操作。
参数：
无
举例：

```
Dim Curkey
Curkey = KEY_SCAN()      ‘读取当前按键值
```

3.2 KEY_STATE

类型： 按键指令
语法：value=KEY_STATE(code)
描述：读取按键按下状态，按下 1.
参数：
Code 按键编号
举例：

```
?EY_SCAN(11) ‘F1 按键的状态
```

3.3 EMG_STATE

类型： 急停开关
语法：value=EMG_STATE
描述：读取按键按下状态，按下 1.

 只有带急停开关的手柄才有作用。

3.4 SPEAKOUT

类型： 喇叭指令
语法：SpeakOut(ms)
描述：响一定的时间，可以出现告警的时候响喇叭，按键按下的时候也可以响喇叭。

参数：
无
举例：
```
 Dim Curkey
Curkey = KEYSCAN()
If Curkey then
    SpeakOut(5)                ‘如果有按键按下喇叭响 5ms
End if
```

3.5 DRAWTEXT

类型： 显示指令
语法：DrawText(x,y,STRING)
描述：显示一个字符串，支持中文，STRING 可以为字符串表达式。
参数：
X, Y 显示区域左上角的坐标位置
STRING 显示的字符串
举例：
```
DrawText(10,10, "运动控制器")
LCD_UPDATE
```

3.6 DRAWTEXT2

类型： 显示指令
语法：DrawText(x1,y1, x2,y2, align, STRING[,drawrect])
描述：显示一个字符串，指定一个方框内显示。
参数：
X1, Y1 显示区域左上角的坐标位置
X2, Y2 显示区域右下角的坐标位置
Align 对齐选项， 0-居中，
>0，左边对齐，值表示距离左边的距离.
<0，右边对齐，绝对值表示距离右边的距离.
STRING 显示的字符串
Drawrect 1-画边框，0-不画边框.
举例：
```
DrawText2(10,10,200,100, 0, "运动控制器")
LCD_UPDATE
```

3.7 DRAWNUM

类型： 显示指令
语法：DrawNum(x,y,value,n,dot)
描述：显示一个数值。
参数：
X, Y 显示区域左上角的坐标位置
Value 缺省显示值
N 总长度位数，包括小数点和符号位。当 N 设置负数时，表示右对齐。
Dot 小数点后取几位
参见：TOSTR
举例：

```
DrawNum(10,10,0,4,2)
LCD_UPDATE
```

3.8 DRAWNUM2

类型： 显示指令
语法：DrawNum2(x1,y1, x2,y2, align,value,n,dot[,drawrect])
描述：显示一个数值，指定一个方框内显示。
参数：
X1, Y1 显示区域左上角的坐标位置
X2, Y2 显示区域右下角的坐标位置
Align 对齐选项， 0-居中，
>0，左边对齐，值表示距离左边的距离.
<0，右边对齐，绝对值表示距离右边的距离.
Value 缺省显示值
N 总长度位数，包括小数点和符号位。当 N 设置负数时，表示右对齐。
Dot 小数点后取几位
Drawrect 1-画边框，0-不画边框.
参见：TOSTR
举例：

```
DrawNum2(10,10,200,100, 0, 123,4,2)
LCD_UPDATE
```

3.9 DRAWRECT

类型： 显示指令
语法：DRAWRECT(X1,Y1,X2,Y2)
描述：画一个方框。
参数：

X1,Y1 显示区域左上角的坐标位置
X2,Y2 显示区域右下角的坐标位置

举例：

```
DRAWRECT(10,10,30,40)
LCD_UPDATE
```

3.10 DRAWLINE

类型： 显示指令
语法：DRAWLINE(X1,Y1,X2,Y2)
描述：画线。
参数：

X1,Y1 直线起始点的坐标位置
X2,Y2 直线结束点的坐标位置

举例：

```
DRAWLINE(10,10,50,50)
LCD_UPDATE
```

3.11 DRAWARC

类型： 显示指令
语法：DRAWARC(CentX, CentY, Radius, StartAngle, EndAngle)
描述：画圆弧。
参数：

CentX, CentY 圆心的位置
Radius 半径
StartAngle 起始角度，弧度单位
EndAngle 结束角度

举例：

```
DRAWARC (100,100,50, 0, PI*2)   ‘画一个整圆
LCD_UPDATE
```

3.12 DRAWBEZIER

类型： 显示指令
语法：DRAWBEZIER(X1,Y1,X2,Y2,X3,Y3,X4,Y4)
描述：画贝塞尔曲线。

参数：

X1,Y1	第 1 个控制点
X2,Y2	第 2 个控制点
X3,Y3	第 3 个控制点
X4,Y4	第 4 个控制点

举例：

```
DRAWBEZIER(X1,Y1,X2,Y2,X3,Y3,X4,Y4)
LCD_UPDATE
```

3.13 DRAWBSPLINE

类型： 显示指令

语法：DRAWBSPLINE (X1,Y1,X2,Y2,X3,Y3,X4,Y4)

描述：画 B 样条曲线。

参数：

X1,Y1	第 1 个控制点
X2,Y2	第 2 个控制点
X3,Y3	第 3 个控制点
X4,Y4	第 4 个控制点

举例：

```
DRAWBSPLINE (X1,Y1,X2,Y2,X3,Y3,X4,Y4)
LCD_UPDATE
```

3.14 DRAWREVERSE

类型： 显示指令

语法：DRAWREVERSE(X1,Y1,X2,Y2)

描述：。

参数：

X1,Y1	左上点
X2,Y2	右下点

举例：

```
DRAWREVERSE(10,10,50,50)
LCD_UPDATE
```

3.15 DRAWPIC

类型： 显示指令

语法：DRAWPIC (Picname, X1,Y1[,X2,Y2])

描述：绘制图片， 图片文件要加入工程，注意图片比较占空间，不需要的图片不要加入工程。

参数：
Picname: 图片文件名

3.16 RGB

类型： 显示指令
语法：COR = RGB(R,G,B)
描述：生成一个颜色。
参数：
R,G,B : 每个分量的颜色 0-255

3.17 SET_LCDVSIZE

类型： 显示指令
语法：SET_LCDVSIZE (xmax,ymax)
描述：设置 XY 的虚拟尺寸， 用于后面在不同的分辨率之间移植。
参数：

3.18 SET_COLOR

类型： 显示指令
语法：SET_COLOR (cor[,backcor])
描述：后面 draw 指令使用的颜色。
参数：
cor： 颜色，缺省为白色。
Backcor: DRAWTEXT 时的背景色，不填的时候为透明.
举例：
SET_COLOR(RGB(255,255,255))

3.19 SET_FONT

类型： 显示指令
语法：SET_FONT (width, height, [fontname])
描述：字体设置，缺省使用内置 16*16 的中文字体，英文为 16*8。但尺寸与字库的尺寸不

一致时，会出现缩放；如果需要自己的字体，请使用 zfontmaker 制作专门的字体文件。
参数：
Width： 字体宽度，英文自动减半。
Height：字体高度
Fontname：使用的字体文件名，不设置时不修改当前字体。

3.20 SET_LINE

类型： 显示指令
语法：SET_LINE (width)
描述：绘图线宽度设置，暂时不支持。
参数：

3.21 LCD_CLR

类型： 显示指令
语法：LCD_CLR([X1,Y1,X2,Y2])
描述：清除指定读取区域，无参数时全部清除。
参数：
X1,Y1 清除区域左上角的坐标位置
X2,Y2 清除区域右下角的坐标位置
举例：

```
LCD_CLR(10,10,50,50)
LCD_UPDATE
```

3.22 LCD_COPYBEFORE

类型： 显示指令
语法：LCD_COPYBEFORE
描述：把上次的屏幕显示内容拷贝作为当前的背景。
参数：
举例：

```
LCD_COPYBEFORE
LCD_CLR(10,10,50,50)
LCD_UPDATE
```

3.23 LCD_UPDATE

类型： 显示指令
语法：LCD_UPDATE
描述：把缓冲内容拷贝到显示屏，所有的 DRAW，和 CLR 指令都是先显示到缓冲。
参数：
无
举例：

```
DrawText(10,10, "运动控制器")          ‘向缓冲内写一串字符串
LCD_UPDATE                          ‘更新数据到显示屏
```

3.24 LCD_IFDRAW

类型： 显示参数
语法：value =LCD_IFDRAW()
描述：修改或读取是否要重新绘制屏幕，当系统有告警或输入框内容变化时，此参数自动赋值。
参数：
无
举例：

```
Dim if_draw
If_draw = LCD_IFDRAW()
```

3.25 ONEMGON:

类型： 中断函数
描述：急停按键按下时触发。
举例：

```
INT_ENABLE=1
END

GLOBAL ONEMGON:
    PRINT “EMG DOWN”, EMG_STATE
END SUB
```

3.26 ONEMGOFF:

类型： 中断函数
描述：急停按键按下时触发。
举例：

```
INT_ENABLE=1
END

GLOBAL ONEMGOFF:
    PRINT “EMG UP”, EMG_STATE
END SUB
```

3.27 输入框 TEXT_

指定类型， 指定 ID，输入框会自己维护绘图，TEXT_Cancel 前输入框总是显示。UPDATE 时会自动重新绘制，

一次只能显示一个输入框，显示新的的时候，自动取消旧的。因此旧的输入必须先保存起来。

当程序发现值超过了范围的时候，可以直接修改。

3.27.1 TEXT_SHOWNUM

类型： 显示指令
语法：TEXT_shownum(id, x，y，当前值，n，dot)
描述：显示一个文本输入框，输入数字。TEXT_Cancel 前输入框总是显示。
参数：

Id	当前显示框 ID 号
x,y	显示区域左上角坐标
当前值	显示的数字
n	总字符个数.
Dot	小数点后取几位

举例：

```
If keyscan() = key_ent then        ‘ENT 键按下显示
    TEXT_shownum(1, 10,10,10.2,4,2)
End if
```

3.27.2 TEXT_SHOWNUM2

类型： 显示指令
语法：TEXT_shownum2(id, x，y，当前值，n，dot)
描述：显示一个文本输入框，输入数字，在指定矩形内显示。
参数：

Id	当前显示框 ID 号

X1, Y1 显示区域左上角的坐标位置
X2, Y2 显示区域右下角的坐标位置
Align 对齐选项， 0-居中，
>0，左边对齐，值表示距离左边的距离.
<0，右边对齐，绝对值表示距离右边的距离.
当前值 显示的数字
n 总字符个数.
Dot 小数点后取几位

举例：

```
If keyscan() = key_ent then ‘ENT 键按下显示
    TEXT_shownum2(1, 10,10,200,100, 0,10.2,4,2)
End if
```

3.27.3 TEXT_SHOWSTRING

类型： 显示指令
语法：TEXT_showstring(id, x，y，当前值,max)
描述：显示一个文本输入框，输入字符串。
参数：
Id 当前显示框 ID 号
x,y 显示区域左上角坐标
当前值 显示的字符串
Max 显示的最大字符个数
举例：

```
If keyscan() = key_ent then ‘ENT 键按下显示
    TEXT_showstring (2, 10,10, "文件名",6)
End if
```

3.27.4 TEXT_SHOWSTRING2

类型： 显示指令
语法：TEXT_showstring(id, x，y，当前值,max)
描述：显示一个文本输入框，输入字符串，在指定矩形内显示。
参数：
Id 当前显示框 ID 号
X1, Y1 显示区域左上角的坐标位置
X2, Y2 显示区域右下角的坐标位置
Align 对齐选项， 0-居中，
>0，左边对齐，值表示距离左边的距离.
<0，右边对齐，绝对值表示距离右边的距离.
当前值 显示的字符串

Max 显示的最大字符个数

举例：

```
If keyscan() = key_ent then ‘ENT 键按下显示
    TEXT_showstring2 (2, 10,10, 200,100, 0, "文件名",6)
End if
```

3.27.5 TEXT_GETNUM

类型： 读取指令

语法：value = TEXT_GetNum()

描述：读取当前文本输入框数字。

参数：

无

举例：

```
Dim number
If TEXT_GetCurId <> -1 than ‘判断当前输入框是否有效
    Number = TEXT_GetNum() ‘读取当前输入框值
End if
```

 读取的输入框类型必须为数据类型

3.27.6 TEXT_GETSTRING

类型： 读取指令

语法：value = TEXT_GetString()

描述：读取当前文本输入框字符串。

参数：

无

举例：

```
Dim array(10)
If TEXT_GetCurId <> -1 than ‘判断当前输入框是否有效
    array= TEXT_GetString() ‘读取当前输入框值
End if
```

 读取的输入框类型必须为字符串类型

3.27.7 TEXT_CANCEL

类型： 取消指令
语法： TEXT_Cancel()
描述：取消显示当前输入框
参数：
无
举例：

```
If TEXT_GetCurId <>-1 then        ‘当前输入框
    TEXT_Cancel()                 ‘取消当前输入框显示
End if
```

3.27.8 TEXT_GETCURID

类型： 读取指令
语法：id = TEXT_GetCurId()
描述：读取当前文本输入框 id。 根据 id 是否=-1 可以判断是否有输入框显示。
参数：
无
举例：

```
Dim curid,number
If Curkey = key_ent then              ‘如果 ENT 键按下
    Curid = TEXT_GetCurId()           ‘查看输入框 id
    If curid<>-1 then
        Number = TEXT_GetNum()        ‘读取当前输入框值
    End if
End if
```

3.28 触摸屏 TOUCH_

3.28.1 TOUCH_ADJUST

类型： 触摸屏指令
语法：TOUCH_ADJUST ()
描述：进行触摸屏校正，此时不要刷新屏幕，校正后参数会自动保存。
参数：
无

3.28.2 TOUCH_SCAN

类型： 触摸屏函数
语法： TOUCH_SCAN (tablenum)
描述：扫描触摸按下动作，返回 1 表示扫描到按下，-1 表示有松开，0 没有变化，tablenum 存储对应的位置。
参数：
tablenum 存储触摸 XY 左边的 table 编号，X,Y 坐标分别存储在 tablenum, tablenum+1
举例：

```
While 1
    if touch_scan(0) = 1 then  ‘扫描按下操作，显示按下位置
        lcd_clr
        DRAWRECT(table(0)-2, table(1)-2, table(0)+2, table(1)+2)
        lcd_update
    endif
wend
```

3.28.3 TOUCH_STATE

类型： 触摸屏指令
语法： TOUCH_STATE (tablenum)
描述：读取触摸状态，返回 TRUE 表示按下，此时 tablenum 存储对应的位置。
参数：
tablenum 存储触摸 XY 左边的 table 编号，X,Y 坐标分别存储在 tablenum, tablenum+1
举例：

```
While 1
    lcd_clr
    if touch_state(0) then  ‘循环显示触摸位置
        DRAWRECT(table(0)-2, table(1)-2, table(0)+2, table(1)+2)
    Endif
    lcd_update
wend
```

第四章 例程

4.1 按键检测

```
while 1
```

```
  keycode = KEY_SCAN()  ’ 扫描按键
  if keycode <> 0 then      ’ 有按键按下或松开
    trace keycode
    lcd_ifdraw = 1  ‘此时要刷新屏幕
    if keycode >0 then speakout(2)  ’ 有按键按下时响喇叭 2ms
  endif
wend
```

4.2 显示例程

```
Runtask 1, showing  ‘启用多任务刷新显示
While 1
  x=x+1
  y=y+1
  if x>= 240 then x=0
  if y>= 120 then y=0
  lcd_ifdraw = 1  ‘此时要刷新屏幕
  delay 200  ‘ 延时再变化
wend

Showing:  ‘刷新
while 1
  wait until lcd_ifdraw  ‘等待需要刷新屏幕
  lcd_clr                    ‘清除缓冲区
  drawtext(x,y,"正运动")
  drawline(x,y,239,119)
  lcd_update        ’ 显示的内容更新到屏幕, 会自动清除 lcd_ifdraw
wend
```

4.3 通讯例程

4.3.1 自定义通讯

```
    Dim char1
    ’配置串口为 RAW 模式
    Setcom (38400,8,1,0,0 ,0)

    ’把串口收到的数据都从串口返回去
    while 1
        get #0, char1
```

```
    putchar #0, char1
wend
```

4.3.2 默认 modbus 通讯

```
setcom(38400, 8,1,0,0,14,2)          ’ 设置串口 0 为 modbus 主端，波特率 38400
modbusm_regget (100, 2, 100)     ‘从控制器上 modbus_ieee(100)读取 X 轴速度
text_shownum(1, 30,50,modbus_ieee(100),6,0)

while key_ent = keyscan                ‘ENT 键按下
   text_cancel()
   modbusm_long(100,text_getnum())        ‘将 X 轴速度写到控制器 modbus 寄存器
wend
```

4.4 常见问题

问题描述	解决方案
显示屏不亮，或亮度不够。	检查控制器的供电：USB 供电必须用质量很好的线并保证电脑的 USB 口供电足够，否则请使用串口头的 24V 供电；串口里面是 24V 供电与 ZHD100 的 5V 供电是不一样的。
通讯不上	检查串口的接线,2/3 需要交叉。 检查串口的参数设置，与控制器要一致。 串口要配置为 MODBUS 主端。
通讯有出错的情况	检查接线是否连接 OK； USB 供电会与控制器的电源冲突，与控制器通讯的时候建议拔掉 USB；如果需要 USB 调试，建议使用笔记本电脑并且使用电池供电。
程序反应速度慢	显示屏的刷新显示只在内容变化的时候才进行。 通讯使用多任务来进行。 按键的扫描使用独立的任务。

第五章 接线说明

5.1 DB 接口信号：

针脚号	名称	说明
1	CANH	预留 CAN
2	RXD	RS232 接收数据引脚
3	TXD	RS232 发送数据引脚
4	CANL	预留 CAN
5	EGND	外部电源地
6	485A	RS485 信号
7	EMG	常闭急停型号输出
8		预留 EGND
9	DC24V	电源输入，用于供电.
10		预留
11	485B	RS485 信号
12	TX-	预留 ETH
13	TX+	预留 ETH
14	RX-	预留 ETH
15	RX+	预留 ETH

5.2 USB 接口

标准的 MINIUSB 接口，需要使用我们提供的 USB 虚拟串口驱动程序。

下载程序时，可以通过 USB 对控制器供电，如果电脑的 USB 口供电不足，则需要使用 24V 从 DB 接头供电。

深圳市正运动技术有限公司
地　址：深圳市宝安区西乡钟屋 1 路 70 号大广发综合楼 5 楼
邮　编：518133
电　话：18927479495

传　真：0755-26066955
Email：tech@zmotion.com.cn
网　址：www.zmotion.com.cn