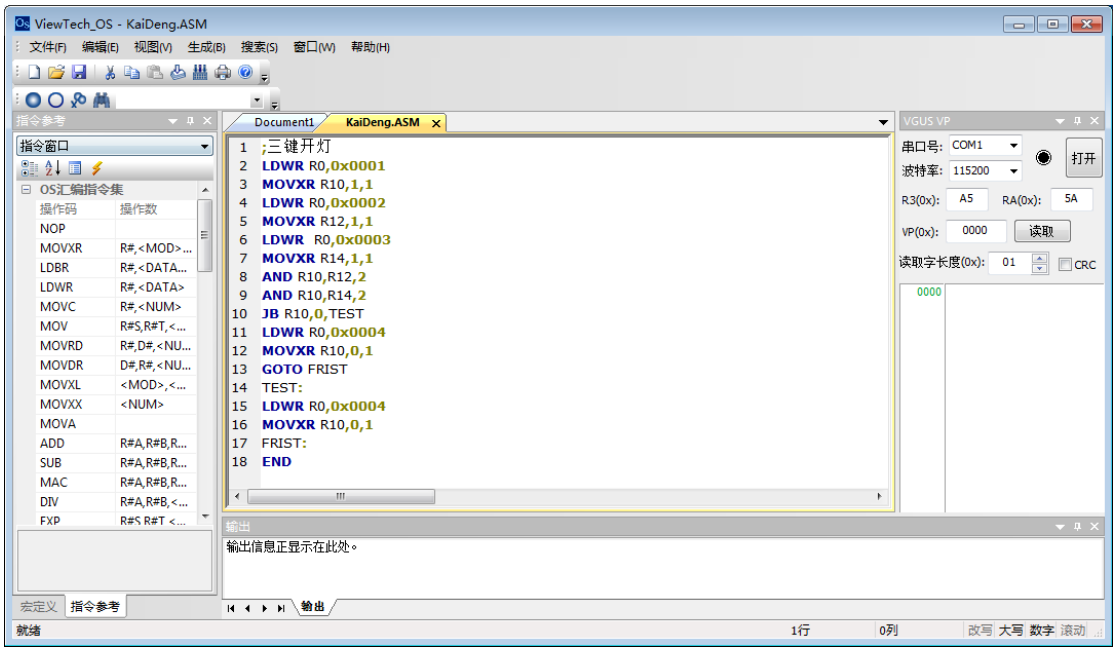
VIEWTECH_OS指令的理解与应用

VIEWTECH_OS平台采用类似汇编程序的编写规范，在VGUS稳定的GUI 平台下，方便用户针对自己的特殊需求快速，可靠地进行二次开发。

VIEWTECH_OS 的PC软件编译界面如下图所示：



基于VGUS的VIEWTECH_OS平台，用户最大代码空间是256KB（32764行）。VIEWTECH _OS程序在每个运行周期（80/120/160/200ms）都运行一次，VGUS+OS的运行时间构成整个运行周期，要求OS 程序中不能出现长的延时循环等待或死循环。

VIEWTECH _OS 的常见应用是解析用户的数据协议，数据处理，能代替工控机或HMI，不仅降低成本，同时极大地提升了可靠性。

编写OS 时的基本约定：

1）OS变量寄存器：R0—R255（只能OS访问），共256个。

2）VGUS寄存器：0x00-0xFF（由0x80/0x81访问），共256个。

3）VGUS变量：0x0000-0xFFFF(由0x82/0x83访问)。

4）字库空间：32-127号（0x20-ox7F）汉字库，24MB。

5）用户数据库空间：参考开发文档。

6）伪汇编指令（或宏指令）：

①EQU——替换，编译时直接替换

如： ONE EQU 1

TWO EQU 2

MOVDR ONE,R10,TWO 与 MOVDR 1,R10,2 等效。

②DB——在ROM中定义连续字节的数据，放在程序最后面（“END”后面）。

③TAB1: （标号）

DB 1 ,2 ,3 , 4

DB 0x00,0x01,0x02

DB “武汉中显”

④DW——在ROM中定义连续字的数据

TAB2:

DW 0x0001, 0x0002,0x0003,0x0004

7）注释用标示为“;”

8）R#为DWIN OS的256个寄存器中的任意一个或连续的一组。

9）< > 表示立即数。100,0x64,64H,064H 都是表示10进制的100。

指令集

指令	指令作用	指令操作数格式	指令说明与使用

LDBR	装载一个单字节立即数到 OS 寄存器或 OS 寄存器组，用于赋值	R#,<DATA>,<NUM>	R#：OS寄存器或OS寄存器组的首地址。
			<DATA>：待写入OS寄存器的数据，字节。
			<NUM>：待写入的OS寄存器个数。
			NUM=0，表示写入 256 个 OS 寄存器，由于OS 只有 256 个寄存器，NUM=0 时，R#必须为 R0。
			例： 1、把十进制数10装入R10
			LDBR R10,10,1
			执行后 R10=10
			2、把十六进制数0xFF装入R20
			LDBR R20,0xFF,1
			执行后 R20=0xFF
			3、把十六进制数AAH装入R30
			LDBR R30,AAH,1
			执行后 R30=AAH
			4、把十进制数10装入R40和R41
			LDBR R40,10,2
			执行后 R40=10
			R41=10
			5、把R0到R255清零
			LDBR R0,0,0
			执行后 R0=0
			……
			R255=0

LDWR	装载一个双字节(一个字)立即数到 OS 寄存器，用于赋值	R#,<DATA>	R#：OS寄存器。
			<DATA>：待写入OS寄存器的数据。
			例： 1、把十进制数10装入R10
			LDWR R10,10
			执行后 R10=0x00
			R11=0x0A
			2、把十六进制数0xAAFF装入R10
			LDWR R10, 0xAAFF
			执行后 R10=0xAA
			R11=0xFF

NOP	空操作指令	NOP	NOP：不执行任何操作，只占用程序行。
LDADR	加载地址 一般和查表指令 MOVC 配合使用	<ADDress>	ADDress：把地址ADDress装载到R5：R6：R7 例： 1、LDADR TAB1 执行后R5：R6：R7装入TAB1定义的字节常数起始地址。 2、LDADR TAB2 执行后R5：R6：R7装入TAB2定义的字常数起始地址。 TAB1： DB 0xFF,0xFE,0xFD,0xFC,0xFB,0xFA,0xF0 TAB2： DW 0xFFFF,0xFFFE,0xFFFD,0xFFFC,0xFFFB

MOVC	查表指令
	装入程序行中定义 的数据常数
	一般和加载地址 LDADR 配合使用
		R# ，<NUM>	R#：OS寄存器或OS寄存器组的首地址
			<NUM> ：待装载到OS寄存器的字节数。
			例： 1、LDADR TAB1 ;把地址装载到R5：R6：R7
			MOVC R10,2
			执行后 R10=0xFF
			R11=0xFE
			2、LDADR TAB2;把地址装载到R5：R6：R7
			INC R7,0,2 ;地址加入偏移量2
			MOVC R10,1
			执行后 R10=0xFD;查表结果相应偏移2个字节
			TAB1:
			DB 0xFF,0xFE
			TAB2:
			DW 0xFFFE,0xFDFC

MOVXR	VGUS 变量和 OS 寄存器之间数据传送	R#,<MOD>,<NUM>	R#：OS 寄存器或 OS 寄存器组的首地址；
			<MOD>：传送模式；
			传送模式=0 OS 寄存器-->VGUS 变量
			传送模式=1 VGUS 变量-->OS 寄存器
			<NUM>：待传送的数据长度，以字（双字节）计。
			NUM 最大 128（0x80 ，256 字节），NUM=0 时，由 R9 中的数据决定传送的长度
			使用本指令时 VGUS 变量地址由 R0：R1 指定。
			例： 1、 LDWR R0，0x0001 ; R0：R1 指定 VGUS
			;变量地址 0X0001
			MOVXR R10 ,1, 1 ;把 VGUS 变量地址
			;0X0001 中的数据读
			;到 R10：R11
			2、LDWR R0，0x0001 ; R0：R1 指定 VGUS
			;变量地址 0X0001
			MOVXR R10 ,1, 2 ;把 VGUS 变量地址
			;从 0X0001 起的数据读
			;到 R10:R11:R12:R13
			3、LDWR R0,0x0010 ; R0：指定 VGUS 变量 R1
			;地址 0x0010
			LDWR R10,0x0001
			MOVXR R10, 0, 1 ;把 R10：R11 数据写到
			;到地址为 0X0010
			;的 VGUS 变量中。
			4、 LDWR R0,0x0010
			LDWR R10,0x0001
			LDWR R12,0x0002
			MOVXR R10, 0, 2 ;把 R10:R11:R12:R13 中的数据写到地址为 0x0010，0x0011 的 VGUS 变量中。

MOVRD	OS 寄存器或 OS 寄存器组的数据传送到 VGUS 寄存器或 VGUS 寄存器组	R#, D# ,<NUM>	R#： OS 寄存器或 OS 寄存器组的首地址；
			D#： VGUS 寄存器或 VGUS 寄存器组的首地址；
			VGUS 寄存器具体定义见寄存器一览表。
			NUM：待传送的数据字节长度。
			例： 1、LDBR R10,10,1
			MOVRD R10,2, 1
			执行后蜂鸣器鸣叫
			2.、LDWR R10,0x0001 ;0X0001 待跳转页面号
			MOVRD R10,3,2
			执行后页面跳转到第一页，页面号由字（双字节）表示。

MOVDR	VGUS 寄存器或 VGUS 寄存器组的数据传送到 OS 寄存器或 OS 寄存器组	D# ,R# ,<NUM>	D#：VGUS 寄存器或 VGUS 寄存器组的首地址； VGUS 寄存器具体定义见寄存器一览表。 R#：OS 寄存器或 OS 寄存器组的首地址； NUM：待传送的数据字节长度。 例： MOVDR 3, R10,2;读当前页面号到 OS 寄存器 执行后 R10：R11 中是当前页面号。 若当前页面是第 2 页，则 R10=0x00，R11=0x02

MOV	OS 寄存器或 OS 寄存器组之间数据传送	R#S, R#T,<NUM>	R#S：OS 寄存器或 OS 寄存器组数据源；
			R#T：待传入数据的 OS 寄存器或 OS 寄存器组；
			<NUM>：待传送的数据字节长度。
			例： LDBR R10,10,1
			MOV R10,R20,1
			执行后 R20=10
			LDWR R10,0XAABB
			MOV R10,R20,2
			执行后 R20=0xAA
			R21=0xBB

MOVXX	VGUS 变量间数据传送	<NUM>	NUM：待传送的数据字长度。 NUM=0，若长度由 R8:R9 指定。R0:R1 指定数据源地址 R2:R3 指定目标地址。 例： LDWR R0,0x0005；R0R1 指定数据源地址 LDWR R2,0x0010；R2R3 指定目标地址。 MOVXX 1 执行后 VGUS 变量 0x0005 中的数据传送到 VGUS 变量 0x0010 中。
MOVA	OS 寄存器变址寻址（由地址对应的立即数寻址寄存器）	无	R2：数据源地址对应的立即数； R3：目的地址对应的立即数； R9：待传送的数据字长度。 OS 寄存器 R0--R255 对应的立即数地址分别为 0-255 例： LDBR R2,10,1； R2 装入了指向 R10 的立即数 LDBR R3,20,1； R3 装入了指向 R20 的立即数 LDBR R10,0xAA, 1 MOVA ;执行后 R20=0xAA INC R3, 0, 1 MOVA ;执行后 R21=0xAA 用于不定长数据处理时方便，如查表改变设定值。

MOVXL	VGUS 变量与字库，用户数据库之间数据传送	<MOD>,<NUM>	MOD：传送方向；
			MOD=0 字库数据读到 VGUS 变量
			MOD=1 VGUS 变量数据写到字库
			MOD=2 用户数据库数据读到 VGUS 变量
			MOD=3 VGUS 变量数据写到用户数据库
			NUM：待读或写的数据字长度 NUM=0 时长度由 R9：R10 定义。
			本指令执行时 VGUS 变量地址由 R0：R1 指定。
			MOD=0~1 时字库号由 R4 指定（范围：32~127）。字库首地址由 R5：R6：R7 指定。
			MOD=2~3 时用户数据库首地址由 R4：R5：R6：R7 指定。
			例： LDWR R10,0xAABB ;待写入字库的数据
			LDWR R0,0x0010 ; VGUS 变量缓存区地址
			MOVXR R10,0,1 ;R10：R11 的数据装入 VGUS 变量缓存区
			LDBR R4,40,1 ;R4 待写入数据的字库号。
			LDBR R5,0, 1
			LDWR R6,0x0020 ;R5：R6：R7 字库首地址
			MOVXL 1,1 ;数据写入字库。
			执行后 R10:R11 的数据写入到 40.HZK 字库的 0x000020（R5R6R7）地址内。

			例： LDWR R0, 0x0010 ; VGUS 变量缓存区地址
			LDBR R4, 40, 1 ;R4 字库号。
			LDBR R5, 0,
			LDWR R6, 0x0020 ;R5：R6：R7 字库首地址
			MOVXL 0, 1 ;字库数据读入 VGUS 变量缓存区。
			MOVXR R10,1,1 ;VGUS 变量缓存区的数据装入 R10：R11
			执行后 40.HZK 字库的 0x000020 地址内的数据读 到 R10:R11。
			例： LDWR R10, 0xAABB ;待写入用户数据库的数据
			LDWR R0, 0x0010 ; VGUS 变量缓存区地址
			MOVXR R10, 0, 1 ;R10：R11 的数据装入 VGUS 变量缓存区
			LDWR R4, 0
			LDBR R5, 0, 1
			LDWR R6, 0x0020 ;R4：R5：R6：R7 库首地址
			MOVXL 3, 1 ; 数据写入用户数据库。
			执行后 R10:R11 的数据写入到用户数据库的 0x00000020（R4R5R6R7）地址内。
			例： LDWR R0,0x0010 ; VGUS 变量缓存区地址
			LDWR R4,0
			LDWR R6,0x0020 ;R4R5R6R7 库首地址
			MOVXL 2,1 ;库数据读入 VGUS 变量缓存区
			MOVXR R10,1,1 ;VGUS 变量缓存区的数据装入 R10：R11
			执行后用户数据库库的 0x00000020 地址内的数据读到 R10:R11。

ADD	32BIT 整数加法	R#A,R#B,R#C	C=A+B A，B 为 32BIT 整数，C 为 64BIT 整数。
			例：计算 2+3=5
			R10:R11:R12:R13+R14:R15:R16:R17=
			R18:R19:R20:R21:R22:R23:R24:R25
			LDBR R10,0,16
			LDWR R12,2
			LDWR R16,3
			ADD R10,R14,R18
			执行后 R18~R24=0
			R25=5

SUB	32BIT 整数减法	R#A,R#B,R#C	C=A-BA , B 为 32BIT 整数，C 为 64BIT 整数。
			例：计算 5-3=2
			R10:R11:R12:R13 - R14:R15:R16:R17=
			R18:R19:R20:R21:R22:R23:R24:R25
			LDBR R10,0,16
			LDWR R12,5
			LDWR R16, 3
			SUB R10,R14,R18
			执行后 R18~R24=0
			R25=2
			计算 3-5=-2
			LDBR R10,0,16

			LDWR R12, 3
			LDWR R16, 5
			SUB R10,R14,R18
			执行后 R18~R24=0xFFFFFF
			R25=0xFE

MAC	64BIT 整数乘法	R#A,R#B,R#C	C= (A*B+C) A , B 为 32BIT 整数， C 为 64BIT 整数
			例：计算 5*3=1
			R10:R11:R12:R13 * R14:R15:R16:R17=
			R18:R19:R20:R21:R22:R23:R24:R25
			LDBR R10,0,16
			LDWR R12,5
			LDWR R16, 3
			MAC R10,R14,R18
			执行后 R18~R24=0
			R25=15

DIV	64BIT 整数除法	R#A,R#B,<MOD>	A/B：（A,B 64BIT）商放入 A ， 余数放入 B；
			MOD：四舍五入模式；
			MOD=0 不四舍五入，MOD=1 四舍五入。
			例： 计算 7/3=2.....1
			LDBR R10,0,16
			LDWR R16,7 ;R10R11R12R13R14R15R16R17
			LDWR R24,3 ;R18R19R20R21R22R23R24R25
			DIV R10,R18,0
			执行后 R10~R16=0
			R17=2 ； 商
			R18~R24=0
			R25=1 ； 余数

SMAC	32BIT 正整数乘法	R#A ,R#B ,R#C	C= (A*B+C) A , B 为 16BIT 整数， C 为 32BIT 整数
			例：计算 5*3=15
			R10:R11 * R12:R13= R14:R15:R16:R17
			LDBR R10,0,8
			LDWR R10,5
			LDWR R12, 3
			SMAC R10,R12,R14
			执行后 R14~R16=0
			R17=15

INC	OS 寄存器自增量	R# ,<MOD>,<NUM>	R#=R# - NUM 无符号数自减量 NUM=0~255。 MOD：R#数据模式 MOD=0 8BIT，MOD=1 16BIT 例： LDBR R10,10, INC R10, 0, 1 执行后 R10=11 LDWR R10,65534 INC R10, 1, 1 执行后 R10=0xFF R11=0xFF

DEC	OS 寄存器自减量	R# ,<MOD>,<NUM>	R#=R# - NUM 无符号数自减量 NUM=0~255 MOD：R#数据模式 MOD=0 8BIT，MOD=1 16BIT 例： LDBR R10,10,1 DEC R10, 0, 1 执行后 R10=9 LDWR R10,6553 DEC R10, 1, 1 执行后 R10=0xFF R11=0xFE

AND	与运算	R#A ,R#B,<NUM>	#A, #B：OS 寄存器或寄存器组的首地址；
			NUM ：运算的数据字节长度。
			A=A AND B 运算规则：见 0 得 0
			例： LDBR R10,10,1
			LDBR R11,11,1
			AND R10,R11,1
			执行后 R10=10
			LDWR R10,0xAAAA
			LDWR R12,0x5555
			AND R10,R12,2
			执行后 R10=0
			R11=0

OR	或运算	R#A ,R#B,<NUM>	#A, #B： OS 寄存器或寄存器组的首地址；
			NUM：运算的数据字节长度。
			A=A OR B 运算规则：见 1 得 1
			例： LDBR R10,10,1
			LDBR R11,11,1
			OR R10,R11,1
			执行后 R10=11
			LDWR R10,0xAAAA
			LDWR R12,0x5555
			OR R10,R12,2
			执行后 R10=0xFF
			R11=0xFF

XOR	异或运算	R#A ,R#B,<NUM>	#A, #B：OS 寄存器或寄存器组的首地址； NUM：运算的数据字节长度 A=A XOR B 运算规则：相同得 0,不同得 1 例： LDBR R10,10,1 LDBR R11,11,1 XOR R10,R11,1 执行后 R10=1
COMSET	串口配置	<MODE>,<BS>	MODE：串口模式设置 1）MODE 高 4 位选择要配置的串口号： ①高 4 位=0 0000 选择 COM1 ②高 4 位=1 0001 选择 COM2 2）MODE 低 4 位选择传输位数及校验方式 ①低 4 位=0 选择 8N1 ②低 4 位=1 选择 8E1 ③低 4 位=2 选择 8O1 ④低 4 位=3 选择 8N2

			波特率设定值(16bit) 1）对于 COM1： <BS> BS=6250000/设置的波特率 。 例： 6250000/115200=54 COMSET 0,54 ; COM1 8N1 115200 6250000/9600=651 COMSET 0,651 ; COM1 8N1 9600 2）对于 COM2： ①BS 高字节波特率因子代号（00,01,02） ②BS 低字节=波特率因子/波特率(波特率因子代号对应的波特率因子 00=5208000bps 01=15625000bps 02=1302000bps) 例： 设 COM2 波特率=9600 MODE00010000 即 0x10，转换成十进制为 16 BS 高字节=02 (1302000bps) BS 低字节=1302000/9600=0x87 COMSET 16, 0x0287
COMTXI	由立即数指定长度发送到串口	<COM>,R#,<NUM>	把 R#指向的<NUM>个寄存器内容发送到 COM。 <COM>：串口号选择 COM=0 COM1，COM=1 COM2； NUM：待发送的数据字节长度。 例： LDWR R10,0xAABB COMTXI 0,R10,1 执行后串口 1 会发送：0xAA LDWR R10,0xAABB COMTXI 0,R10,2 执行后串口 1 会发送：0xAA ，0xBB
COMTXD	由寄存器指定长度发送到串口	<COM>,R#S,R#N	<COM>：串口号选择 COM=0 COM1， COM=1 COM2； R#S：装入了待发送数据的寄存器组首地址； R#N：指定待发送数据字节长度的寄存器（如果 R#N=0x00,则发送 256 字节数据）。 例： LDWR R10,0xAABB LDBR R12,2,1;长度 2 长度 COMTXD 0,R10,R12 执行后串口 1 会发送 0xAA，0xBB
RDXLEN	查询串口接收缓冲区已接收到的数据字节长度	<COM>,R#	COM： 串口号； R#：返回的数据字节长度，R#=0 表示串口接收缓冲区没有接收到数据。 例： RDXLEN 0 R10 ；把数据长度读取到 R10 R10=0 缓冲区无数据
RDXDAT	读取串口接收缓冲区的数据	<COM>,R#A,R#B	COM：串口号； R#A：读取到的数据存放的寄存器组首地址； R#B：要读取的数据字节长度(8bit)； 例： LDBR R12,2,1; 读取长度 RDXDAT 0, R10,R12 执行后 R10,R11 读取到的数据

CRCA	ANSI CRC-16 计算(Modbus)	R#S , R#T, R#N 多项式 X16+X15+X2+1	R#S：要计算 CRC 的数据在 OS 寄存器组的首地址；
			R#T：计算得到的 CRC 结果(16bit) LSB(低位在前)保存；
			R#N：参与计算的数据长度指定寄存器(8bit)。
			例： LDBR R10, 0x01,1; R#S
			LDBR R11,0x03,1
			LDBR R12, 0x00,1

			LDBR R13,0x01,1
			LDBR R14, 0x00,1
			LDBR R15, 0x01,1
			LDBR R16,6,1; R#N
			CRCA R10, R17,R16
			执行后 R17=0xD5 ;R#T
			R18=0xCA
			01 03 0001 0001 D5CA

CRCC	CCITT CRC-16 计算 (xModbus)	R#S , R#T, R#N 多项式 X16+X12+X5+1	R#S：要计算 CRC 的数据在 OS 寄存器组的首地址；
			R#T：计算得到的 CRC 结果(16bit)MSB(高位在前)保存；
			R#N：参与计算的数据长度指定寄存器（8bit）。
			例： LDBR R10, 0x01,1 ;R#S
			LDBR R10, 0x01,1
			LDBR R12, 0x00,1
			LDBR R13,0x01,1
			LDBR R14, 0x00,1
			LDBR R15, 0x01,1
			LDBR R15, 0x01,1 ;R#N
			CRCC R10, R17,R16
			执行后 R17=0x8C ;R#T
			R18=0x63
			01 03 0001 0001 8C63

SUMADD	字节累加校验和（不进位）	R#S , R#T, R#N	R#S：要计算校验和的数据在 OS 寄存器组的首地址；
			R#T：计算得到的结果(8bit)；
			R#N：参与计算的数据长度指定寄存器（8bit）；
			例： LDBR R10, 0xAA,1 ; R#S
			LDBR R11,0xBB,1
			LDBR R12, 0x00,1
			LDBR R13,0x01,1
			LDBR R14, 0x00,1
			LDBR R15, 0x01,1
			LDBR R16,6,1; R#N
			SUMDD R10, R17,R16
			执行后 R17=0x67; R#T

SUMADDC	字节累加校验和（进位）	R#S , R#T, R#N	R#S：要计算校验和的数据在 OS 寄存器组的首地址；
			R#T：计算得到的结果(8bit)；
			R#N：参与计算的数据长度指定寄存器（8bit）；
			例： LDBR R10, 0xAA,1; R#S
			LDBR R11,0xBB,1
			LDBR R12, 0x00,1
			LDBR R13,0x01,1
			LDBR R14, 0x00,1
			LDBR R15, 0x01,1
			LDBR R16,6,1; R#N
			SUMADDC R10, R17,R16
			执行后 R17=0x68; R#T

SUMXOR	异或校验和	R#S , R#T, R#N	R#S：要计算校验和的数据在 OS 寄存器组的首地址； R#T：计算得到的结果(8bit)； R#N：参与计算的数据长度指定寄存器（8bit）。

			例： LDBR R10, 0xAA,1; R#S
			LDBR R11,0xBB,1
			LDBR R12, 0x00,1
			LDBR R13,0x01,1
			LDBR R14, 0x00,1
			LDBR R15, 0x01,1
			LDBR R16,6,1; R#N
			SUMXOR R10, R17,R16
			执行后 R17=0x11; R#T

CJNE	两个 8bitOS 寄存器比较	R#A , R#B,<TAB>	比较 A,B 两个 8bit OS 寄存器，不等跳转到 TAB，相等执行下一条，跳转范围：+/-127 行指令。
			例： LDBR R10,1,1
			LDBR R11,2,1
			CJNE R10,R11,NEXT1
			LDBR R10,3,1；不会执行
			GOTO NEXT2
			NEXT1: LDBR R10,2,1
			NEXT2: NOP
			.....
			例： LDBR R10,1,1
			LDBR R11,1,1
			CJNE R10,R11,NEXT1
			LDBR R10,3,1；会执行
			GOTO NEXT2
			NEXT1: LDBR R10,2,1
			NEXT2: NOP
			.....

JS	两 16bit OS 寄存器比较	R#A, R#B, <TAB>	比较 A,B 两个 16bit 整数大小, A<B 跳转到 TAB,A>=B 执行下一条 跳转范围：+/- 127 行指令。 JS R10,R12,TEST1
IJNE	OS 寄存器与立即数比较	R#,<INST>,<TAB>	8bit OS 寄存器与立即数<INST>比较，不等跳转到 TAB，相等执行下一条。 IJNE R10,2,TEST1

TESTS	序列比较	R#A, R#B, <NUM>	依次比较 A、B 两个寄存器序列值：
			值不同时，返回 A 序列此时的地址到 R0 寄存器；
			如果 A、B 相同则返回 0x00 到 R0 寄存器。
			R#A：A 序列寄存器，R#B：B 序列寄存器。
			<NUM>：最大比较数据字节长度。
			例： LDWR R10, 0xAABB
			LDWR R12, 0xAABB
			LDWR R14, 0xAABB
			LDWR R16, 0xAABB
			TESTS R10, R14, 4
			执行后 R0=0x00
			例： LDBR R10, 0xAA,1
			LDBR R11, 0xBB, 1
			TESTS R10, R11, 1
			执行后 R0=0x0A ;R10 对应的物理地址=10

BITS	位分解	R#, <VP>	把 R#的 8 个比特分解到 VP 指向的 8 个 VGUS 字变量，MSB 方式，bit1

			分解为 0x0001，bit 0 分解为 0x0000。 R#：需要进行位分解的寄存器，8bit； <VP>：VGUS 变量地址。 例： LDBR R10,0x5A,1；0101 1010 BITS R10,0x0010 执行后 0010=0x0000 0011=0x0001 0012=0x0000 0013=0x0001 0014=0x0001 0015=0x0000 0016=0x0001 0017=0x0000
BITI	位组合	R#,<VP>	把 VP 指向的 8 个 VGUS 字变量组合成 1 个字节位变量，MSB 方式，0x0000 为 bit 0，非 0x0000 数据为 bit 1。 R#：存储位组合数据的寄存器，8bit； <VP>：VGUS 变量地址。 例: R10=0 0010=0x0000 0011=0x0FFF 0012=0x0000 0013=0x0002 0014=0x0001 0015=0x0000 0016=0x0001 0017=0x0000 BITI R10,0x0010 执行后 R10=0x5A
JB	对 OS 寄存器的位判断	R#,<bit>,<TAB>	测试 R#寄存器的第<Bit>位，1 跳转， 0 继续执行下一条代码。 R#：位测试的寄存器，16bit； <Bit>：位测试位置，0x00-0x0F，MSB 方式； <TAB>：跳转位置。 例： LDWR R10, 2 JB R10,1,TEST1 LDBR R11,1,1 ..... GOTO ..... TEST1:NOP .....
HEXBCD	HEX 数据转换成压缩的 16 进制表示的 BCD 码	R#S,R#T,<MOD>	把 HEX 数据转换成压缩的 BCD 码，如数据 1000 0x3E8,将转换成 0x10,0x00。 R#S：输入 HEX 数据的寄存器组首地址； R#T：输出压缩 BCD 码数据的寄存器首地址； <MOD>：高 4bit 表示输入 HEX 字节数，0x01-0x08 低 4bit 表示输出 BCD 码字节数，0x01-0x0A。 例： LDBR R10,0X3E8 HEXBCD R10,R20,34 ； 0010 0010=34 各 2 字节 执行后 R20=0x10 R21=0x00

BCDHEX	16 进制表示的 BCD 码转换成 16 进制数。	R#S,R#T,<MOD>	把压缩的 BCD 码转换成 HEX 数据，比如数据 0x1000(16 进制表示的 BCD 码)将转换成 0x3E8（1000）。
			R#S：输入压缩 BCD 码的寄存器组首地址；
			R#T：输出 HEX 数据的寄存器首地址；
			<MOD>：高 4bit 表示输入 BCD 码字节数，0x01-0x0A,低 4bit 表示输出 HEX 字节数，0x01-0x08。
			例：从 VGUS 寄存器 0x20 中读取年，月，日，时，分，秒到 R20.—R26，假如是 2015-2-2218:16:40，则 R20=0x15
			R21=0x02
			R22=0x22
			R23=星期
			R24=0x18
			R25=0x16
			R26=0x40
			如果把数据直接放到数据变量显示(显示效果 BCD 码)显示 21 年 2 月 34 日 24 时 22 分 64 秒
			MOVRD R20,0x20,7
			MOV R24,R23,1
			MOV R25,R24,1
			MOV R26,R25,1
			LDBR R40,0,4
			MOV R20,R40,1
			BCDHEX R40,R41,0X11 ;00010001
			MOV R41,R20,1; 各 1 字节
			MOV R21,R40,1
			BCDHEX R40,R41,0X11
			MOV R41,R21,1
			MOV R22,R40,1
			BCDHEX R40,R41,0X11
			MOV R41,R22,1
			MOV R23,R40,1
			BCDHEX R40,R41,0X11
			MOV R41,R23,1
			MOV R24,R40,1
			BCDHEX R40,R41,0X11
			MOV R41,R24,1
			MOV R25,R40,1
			BCDHEX R40,R41,0X11
			MOV R41,R25,1
			执行后，再显示为 15-2-22 18:16:40

ASCHEX	ASCII 字符串转换成 64bit 带符号 HEX 数据	R#S,R#T,<LEN>	把 ASCII 字符串转换成 64bit 带符号 HEX 数据。 R#S：输入 ASCII 字符串寄存器首地址； R#T：输出 HEX 数据，64bit 寄存器； <LEN> ：ASCII 字符串数据长度，包括符号位和小数点，0x01-0x15。 例： LDADR TAB MOVCR10,2 ASCHEX R10,R20,2 TAB: DB "30"

			执行后 R20-R26=0 R27=0X1E LDADR TAB MOVC R10,3 ASCHEX R10,R20,3 TAB: DB "-30" 执行后 R20-R26=0XFF R27=0XE2 (负数补码)
HEXASC	整数转 ASCII 字符串	R#S,R#T,<MOD>	R#S：需要转换的 32bit 整数寄存器组； R#T：转换后的 ASCII 字符串寄存器组； <MOD>：转换模式控制，高 4bit 为整数位长度，低 4bit 为小数位数目。转换的 ASCII 串带符号，右对齐，空位用 0x20 填充。 对于数据 0x12345678 <MOD>=0x62 转换结果为+054198.96 <MOD>=0xF2 转换结果为+3054198.96 例： LDWR R10,0X1234 LDWR R12,0X5678 HEXASC R10,R20,0X62 执行后 R20=0X2B （+） R21=0X30 （0） R22=0X35 （5） …… 2B 30 35 34 31 39 38 2E 39 36

SQRT	平方根计算（结果已取整）	R#A, R#B	计算一个 64 位无符号数的平方根。
			R#A：待计算的 64 位无符号数；
			R#B：得到的结果(32bit)。
			例： LDBR R10,0,16
			LDBR R17,5,1 ;R10 R11 R12 R13 R14 R15 R16 R17
			SQRT R10,R18 ;R18 R19 R20 R21
			执行后 R18-R20=0 R21=2
			LDBR R10,0,16
			LDBR R17,9,1 ;R10 R11 R12 R13 R14 R15 R16 R17
			SQRT R10,R18 ;R18 R19 R20 R21
			执行后 R18-R20=0 R21=3

SCAN	读取键盘输入时的数据到 OS 寄存器	R#，<NUM>	把当前输入法下已录入到缓冲区的数据 NUM 个读到 R#+1。 <NUM>：R#保存数据的字节长度。 SCAN R20,6
SCANADD	在输入法下把字符数据添加到输入法缓冲区	R#A, R#B	文本输入法下，把 R#A 起的字符数据添加到输入法缓冲区。 R#B：添加的字节长度。 SCANADD R80，R90

WRLINE	给指定通道缓冲区写数据	R#S,R#I,<CH>	R#S：待写入数据（16bit 无符号整数）的起始地址；
			R#I:：3 字节寄存器组，从 R#I 开始连续定义 3 字节变量；
			R#I 待写入数据的长度，R#I+1-- R#I+1 待加入到数据中的偏移值。
			例： LDWR R20,0X0000
			LDWR R22,0X0030
			LDWR R24,0X0060
			LDWR R26,0X0090

			LDWR R28,0X0095
			LDWR R30,0X0098
			LDWR R32,0X009A
			LDWR R34,0X009F
			LDWR R36,0X0100
			LDWR R38,0X0105
			LDWR R40,0X0115
			LDWR R42,0X0200
			LDWR R44,0X0225
			LDWR R46,0X0235
			LDWR R48,0X0245
			LDWR R50,0X0255
			LDWR R52,0X0265
			LDWR R54,0X0266
			LDWR R56,0X0267
			LDWR R58,0X0268
			LDWR R60,0X0800
			LDWR R62,0X0900
			LDWR R64,0X0600
			LDWR R66,0X0030
			LDBR R12,24,1
			LDWR R13,100
			WRLINE R20,R12,1
			执行后
			R20-R66（len=24 字）的数据分别加上 100 的偏移值写到缓冲区显示

TIME	时间计算函数	R#A,R#B,<MOD>	R#A,R#B：保存的各 6 个字节的 OS 寄存器组（BCD 格式）；
			MOD=0： A=A-B（要求 A>B）；
			MOD=1： A= B-RTC；
			MOD=2： A= RTC-B；
			2015-2-14 13:23:20 - 2015-2-14 12:28:30= 54 分 50 秒
			例： LDBR R10,0X15,1
			LDBR R11,0X02,1
			LDBR R12,0X14,1
			LDBR R13,0X13,1
			LDBR R14,0X23,1
			LDBR R15,0X20,1
			LDBR R20,0X15,1
			LDBR R21,0X02,1
			LDBR R22,0X14,1
			LDBR R23,0X12,1
			LDBR R24,0X28,1
			LDBR R25,0X30,1
			TIME R10,R20,0
			执行后
			R10=0x00
			R11=0x00
			R12=0x00
			R13=0x00
			R14=0x54 分

			R15=0x50 秒
ERASE	擦除指定字库	<L_ID>	L_ID：要擦除的字库号 32-127，要保证工程文件中有字库 ERASE 40。 执行后 ID 为 40 号的字库被擦除
ADDL14	增加显示变量	R#A,R#B,<MOD>	R#A：保存 1 条显示变量（32 字节）的寄存器组首地址； R#B：变量添加的编号数 0-31 总共能添加 32 个； MOD=0x5A 把显示变量添加到指定位置； MOD=非 0x5A 删除指定位置的显示变量。 ADDL14 R10,R50,0x5A
EXIT	强制结束当前的输入法状态	R#A, R#B	R#A：控制是否切换页面，不切换=0，切换=1； R#B：要切换到的页面 ID（16BIT）。 EXIT R10，R12
FECEN	FEC 编码	R#A, R#B, R#C	曼彻施特编码。 对 R#A 指向，字节长度为 R#C 的数据串进行编码，编码输出保存在 R#B 位置。 FECEN R20，R80，R10
FECDE	FEC 解码	R#A, R#B, R#C	曼彻施特解码。 对 R#A 指向，字长度为 R#C 的数据串进行解码，编码输出保存在 R#B 位置。 FECDE R20，R80，R10
GOTO	直接跳转	<PC>	<PC>：跳转位置，0xFFFF 表示值在 R0:R1 中。 GOTO TEST1 NOP TEST1:NOP
CALL	子程序调用	<PC>	调用 PC 指针位置（0x0000-0x7FFB）的子程序，最多支持 32 级程序嵌套。 CALL TEST
RET	子程 调用返回		CALL 调用指令返回。 RET
END	程序结束		END