ICS-EDM-0401
ICS-EBM-0401
嵌入式通用工业控制器
技术手册

Version 1.0

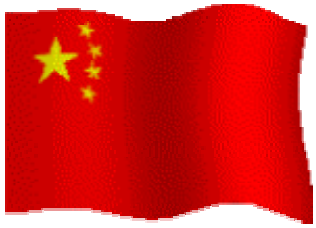
JohnZH@OverPrime.com

20005-11-11

创意时代

ZhuHai GD China

Support@overprime.com



Made in China

注意！

- 本文件 OverPrime 公司于 2005 年生效。版权归 OverPrime 公司所有。本手册信息将根据产品的升级而更改，不做另行通知。
- 除厂家许可外，不得擅自以任何行式再生、复印、翻译、修改、或传送手册中的任何部分。
- 本文中所有涉及的软件的版权由原有著作公司所有。
- OverPrime 公司不承担可能被包含在这份文件中的任何错误的责任.。
- OverPrime 公司保留更新或维持本用户手册中信息的权力。

任何其它目的之使用，均被法律明确禁止，并可导致严重的民事及刑事处罚。违反者将在可能的最大程度上受到指控。

目录

ICS-E 系列嵌入式通用工业控制器介绍

ICS-E 系列通用工业控制器是采用先进的电子技术、自动化技术、可靠性技术和开放式的设计思想设计而成的新一代微型模块化控制器，具有极高的可靠性、极高的性能/价格比和极其灵活的扩展能力等特点，可适用于各行各业，各种场合中的检测、监测及控制的自动化。使用范围可覆盖从替代继电器，智能 IO 的简单控制到 PLC 等复杂的自动化控制。

ICS-E 系列通用工业控制器功能强大，采用图形化方式的组态编程或者 Basic 语言方式编程，易学易用提高开发效率。

ICS-E 系列通用工业控制器可广泛用于冶金、电力、交通、化工、轻工、造纸、楼宇自动化、水处理等行业，以及食品与饮料机械、包装机械、纺织机械、烟草机械、木工机械、印刷机械、制冷设备、医疗设备、电梯、机器人等等。

ICS-E-0401 系列主要特点

- 运算能力强大的单芯片设计，丰富的外部资源；ICS-EDM-0401 ICS-EBM-0401 单芯片提供 8 路数字输出，１６路数字输入，８路模拟输入，２路ＰＷＭ，３路高速输入／计数，两路 Modbus 通讯口。
- 单芯片除了以上的外部资源外，还提供了 16K 用户ＲＡＭ数据存储器、64KＦＬＡＳＨ程序存储器、ＲＴＣ等资源，单芯片的设计也提供了极高的可靠性。同时单芯片的设计也极大地降低了系统成本，缩小了体积。每秒钟可以处理超过 4 千万条的指令（40MIPS）的 CPU 提供了极佳的性能表现。
- 工业级的温度范围。
- 内置实时多任务操作系统，在提供强大的处理能力和通讯能力的同时，对所有的用户数据都提供了完善的数据保护，绝对确保数据安全，保证系统长期可靠的运转。
- ICS-EDM 系列通用工业控制器采用图形化方式的组态编程。鼠标点击和拖放即可完成符合工艺流程的组态过程。易学好用，省时省事效率高；
- ICS-EDM 提供了数量几乎不受限制的逻辑运算，时间控制，算数运算（浮点），ＰＩＤ运算等运算控制软模块。并提供 2 路 PWM 或脉冲串输出。同时提供了ＳＬＯＷＰＷＭ模拟软模块，可以与每个普通的数字输出通道相连实现低速ＰＷＭ控制。

（温度，压力等控制）

- ICS-EBM 系列通用工业控制器采用 basic 语言方式编程，使您轻松实现过程控制。与类似汇编的ＰＬＣ语言相比，学习和使用都非常方便快捷，编程效率有了极大的提高，代码具有逻辑和流程清晰易读的特点。
- ICS-EDM、ICS-EBM 具有 Modbus 主 Modbus 从两套接口和协议，支持 Modbus - RTU / Modbus - ASCII 两种数据格式。支持 9600 19200 38400 57600 115200 五种通讯速率。
- ICS-EXX 专为嵌入式控制而设计，只有 40mm X 50 mm 左右，采用单一 5V 供电，52PIN 接口。可以非常方便的嵌入到各种机器人，控制设备，测量设备中。小巧的体积为您的产品设计提供了广阔的思路和创意源泉。缩短了产品设计周期，加快产品进入市场的时间。
- 由于所有技术均为自主产权，因此该模块拥有无人能比的新能价格比。

开发环境

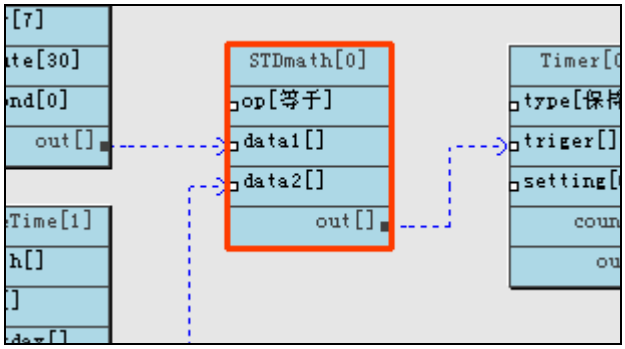
用户可通过以下软件和设备组态或者编程并向 ICS 设备下载程序：

1） 一台个人计算机，一套 FBDStudio 功能块图编辑软件(中文界面)或者 EB Studio Basic（英文界面）编程调试环境 及 WIN95 以上的操作系统。

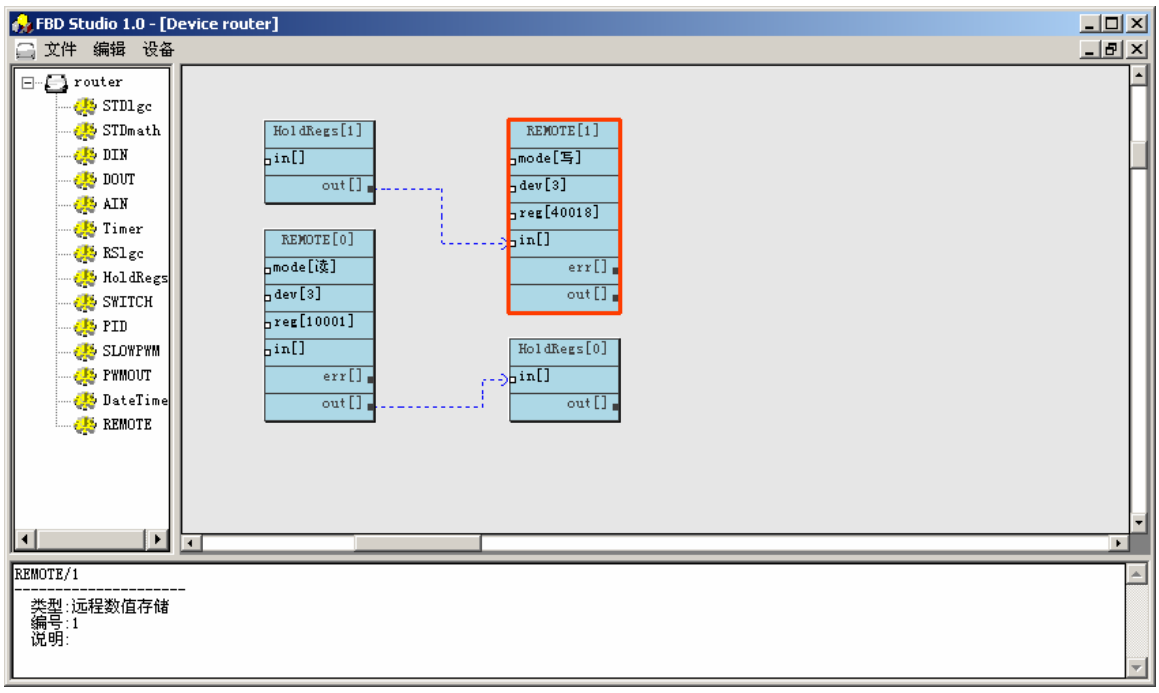
2） 计算机串口线（232 接口）或者一个具有全透明零延时转换的 RS232/RS485 转换器（485 接口）

3） 维护通过模组的通讯口（Slave）完成。

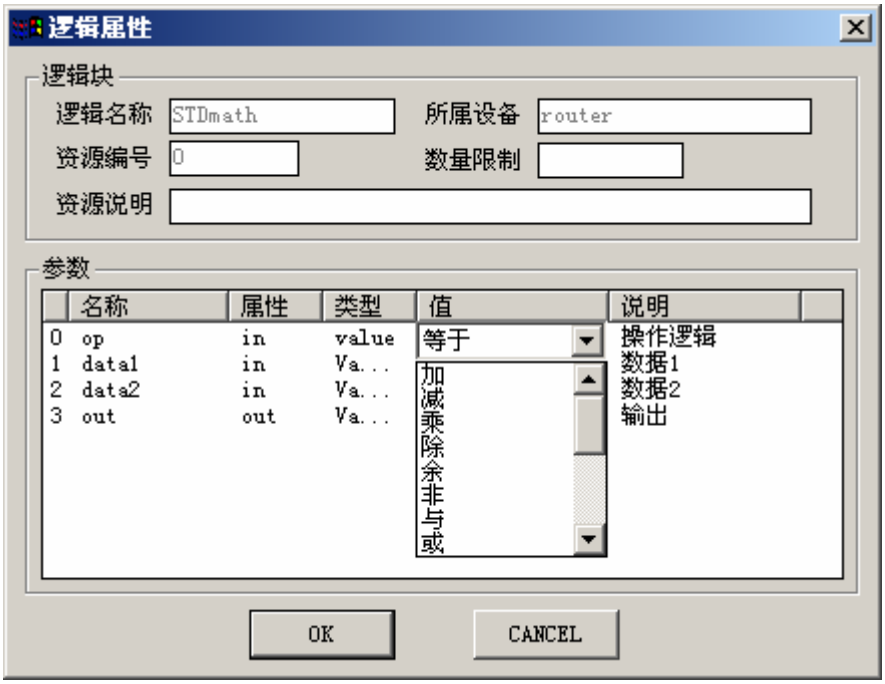
ICS-EDM、ICS-SDM 系列通用工业控制器组态过程(使用 FBDStudio)



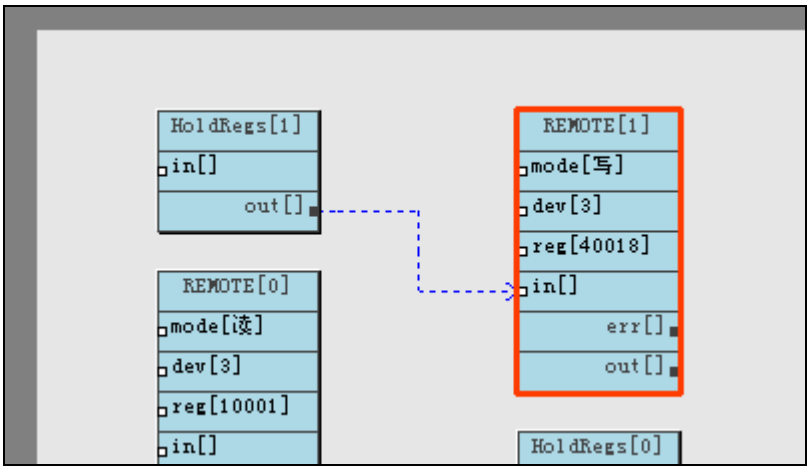
功能块图



FBDStudio 界面



设置功能块的参数

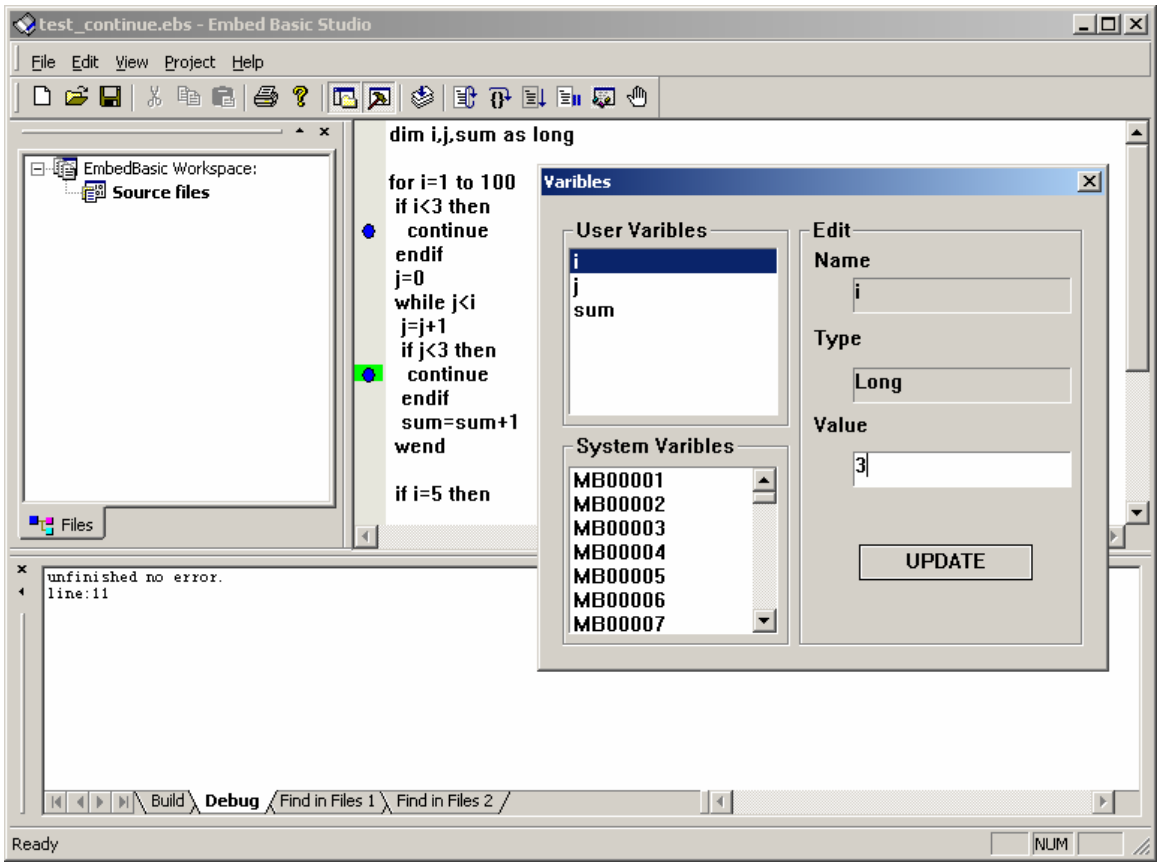


功能块图中使用网络(Remote，HoldRegs 块)

ICS-EDM、ICS-DNM 系列通用工业控制器组态过程简单，逻辑清晰

ICS-BNM 系列通用工业控制器编程

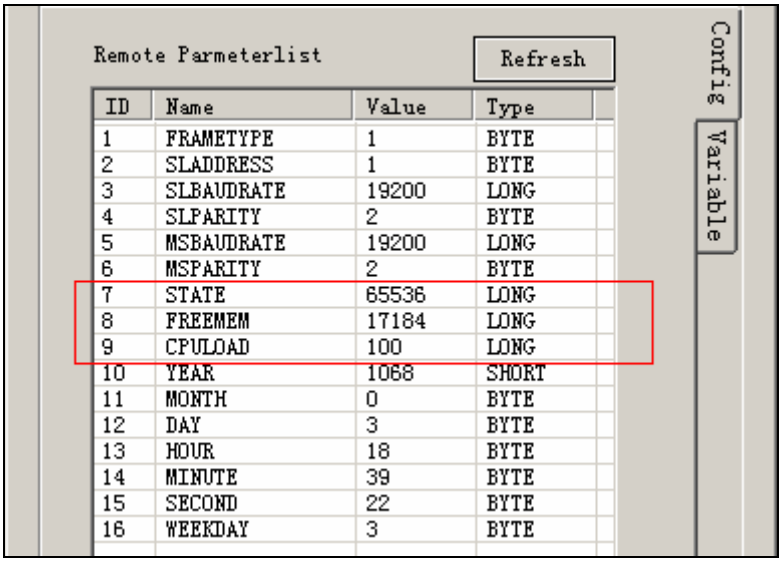
ICS-BNM 系列通用工业控制器采用 basic 语言方式编程，提供模拟调试环境。可以在开发环境中编辑，模拟调试，编译为设备中执行的.pc 代码。



Embed Basic Studio 界面

控制器维护使用标准的 MODBUS 网络。

ICS-DNM、ICS-ENM、ICS-BNM 系列通用工业控制器上下载维护工具使用标准的 Modbus 控制协议，可以查看ＣＰＵ工作状态，内存状态，网络配置信息。



ID	Name	Value	Type
1	FRAMETYPE	1	BYTE
2	SLADDRESS	1	BYTE
3	SLBAUDRATE	19200	LONG
4	SLPARITY	2	BYTE
5	MSBAUDRATE	19200	LONG
6	MSPARITY	2	BYTE
7	STATE	65536	LONG
8	FREEMEM	17184	LONG
9	CPULOAD	100	LONG
10	YEAR	1068	SHORT
11	MONTH	0	BYTE
12	DAY	3	BYTE
13	HOUR	18	BYTE
14	MINUTE	39	BYTE
15	SECOND	22	BYTE
16	WEEKDAY	3	BYTE

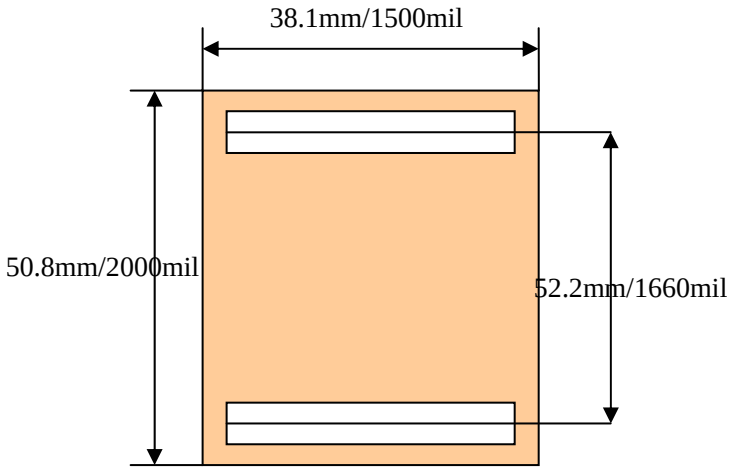
可以功过网络查看模块的信息

可以配置支持 modebus 协议的人机界面 如国产的 eView 等。

ICS-E 系列通用工业控制器详细说明

ICS-EDM-0401

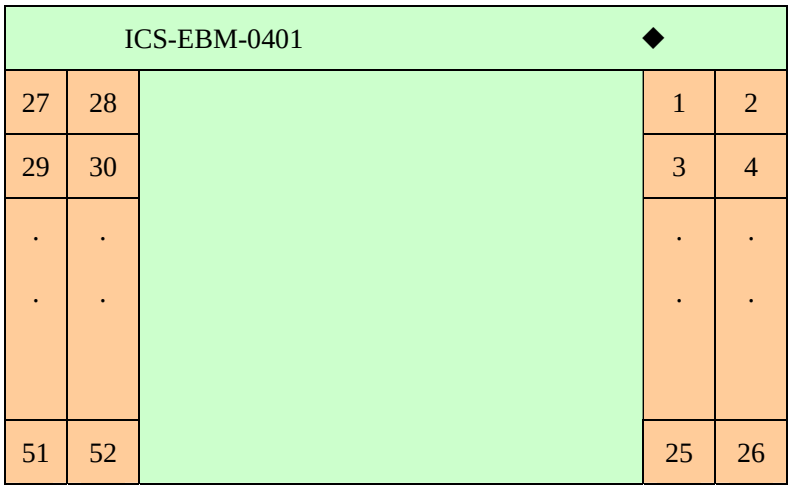
外形尺寸如图



ICS-EDM-0401 引脚说明

两组引脚，每组引脚有 26 针。丝印的箭头标出第一脚位置。

详细的引脚排列见下图：



顶视图

引脚定义如下：

引脚编号	符号	含义	备注
1	DI0	开入量	
2	DI1	开入量	
3	DI2	开入量	
4	DI3	开入量	
5	DI4	开入量	
6	DI5	开入量	
7	DI6	开入量	
8	DI7	开入量	
9	DI8	开入量	
10	DI9	开入量	
11	DI10	开入量	
12	DI11	开入量	
13	DI12	开入量	
14	DI13	开入量	
15	DI14	开入量	
16	DI15	开入量	
17	VCC	5V 电源输入	
18	GND	电源地	
19	DO0	开出量	
20	DO1	开出量	

21	DO2	开出量	
22	DO3	开出量	
23	DO4	开出量	
24	DO5	开出量	
25	DO6	开出量	
26	DO7	开出量	
27	AI7	模拟输入量	
28	AI6	模拟输入量	
29	AI5	模拟输入量	
30	AI4	模拟输入量	
31	AI3	模拟输入量	
32	AI2	模拟输入量	
33	AI1	模拟输入量	
34	AI0	模拟输入量	
35	GNDA	模拟输入地	输出
36	VDDA	模拟输入电源	输出 电压 3.3V
37	VBAT	电池输入	电压 3.3V 20uA
38	GND	电源地	
39	VDD	3.3V 电源输出	输出 电压 3.3V
40	REST	外部复位	输入
41	RUN	运行指示灯（低电平）	内部已提供限流电阻
42	ERROR	错误指示灯（低电平）	
43	FI0		
44	FI1		
45	FI2		
46	STOPMODE	停止模式	启动时接地模组进入 stop 模式
47	RSLAVE	MODBUS SLAVE 接收	9600----115200
48	RMASTER	MODBUS MASTER 接收	
49	TSLAVE	MODBUS SLAVE 发送	
50	TMASTER	MODBUS MASTER 发送	
51	PWM1		100-100KHz
52	PWM2		

ICS-EDM-0401 软件模组：

ICS-EDM-0401 提供下列功能模块

编号	符号	含义	说明
1	STDlgc	逻辑处理单元	完成与、或、非，异或 和 相等 等逻辑运

			算
2	STDmath	数值运算处理单元	能够实现 加，减，乘，除，余，非，与，或，异或，不等，等于，大于等于，小于等于，小于，大于，最大，最小 17 种算术运算和比较操作 支持逻辑，16 位整数，32 位整数和浮点类型。
3	DIN	开关量输入	与 modebus 寄存器 mb100xx 对应
4	DOUT	开关量输出	与 modebus 寄存器 mb000xx 对应
5	AIN	模拟量输入	10bit 精度 与 modebus 寄存器 mb300xx 对应
6	Timer	时间逻辑	能够实现 Timer（定时触发信号），保持器（延时），定时器，计数器
7	RSlgc	启动停止控制	
8	HoldRegs	Modbus 存储单元	对本地 Modbus 存储单元 mb400xx 的存取，用于通讯或者参数设置
9	SWITCH	数据选择	控制数据流向，动态参数调整
10	PID	PID 单元	有以下可配置项 Sp：设定目标，Kc：回路增益，Ti：积分常数，Td：微分常数，SAi：积分饱和点，ErCt：积分项切除误差
11	SLOWPWM	低速 PWM 单元	秒级的软件 PWM 模块,同 DOUT 配合实现低速的 PWM 控制。
12	PWMOUT	PWM 输出	实现 100Hz - 100KHz 的 PWM 调制。
13	DateTime	日期合成单元	实时时钟功能。支持年月日时分秒和礼拜。
14	REMOTE	远程数值存储	读写接入 ModBus Master 总线上其他设备寄存器
15	SQClgc	顺序触发器	根据规定的触发顺序产生状态信息。支持 16 个输入触发信号。

ICS-EDM-0401 Modbus 寄存器说明：

地址	功能	备注
00001----00008	开出量	对应 DOUT[0—7]
10001----10016	开入量	对应 DIN[0—15]
30001----30008	模入量	对应 AIN[0-7]
40001----40016	可读些寄存器	通过网络设定后断电保持 可用于参数设定
40017---40032	可读些寄存器	断电不保持，用作一般通讯

ICS-EDM-0401 文件说明：

info：型号信息文件，可以下载保存。文件是的后缀为 dex 的设备信息文件，可以选中后点击下载按钮进行下载。

usercode：用户的组态文件，只能使用上传方式更新，不能下载。

应当上载 FBDStudio 产生的相应设备的 .dc 格式文件。

如果没有用户组态，可以上载只包含一个数字输入的.dc 文件（用 FBDStudio 生成）即可。

ICS-EDM-0401 参数说明：

名称	含义	备注
FRAMETYPE	modbus 协议帧格式	1—ASCII / 2—RTU
SLADDRESS	从地址	
SLBAUDRATE	从波特率	
SLPARITY	从校验格式	0--无校验 1--奇校验 2--偶校验 3--强制 1 4—强制 0
MSBAUDRATE	主波特率	
MSPARITY	主校验格式	见从校验格式
STATE	状态	65536 运行 196608 停止 写入 0x01000000 系统会复位，重新启动 写入其他数据可能会**永久损坏**模块内的程序
FREEMEM	剩余内存	BYTE
CPULOAD	CPU 负载	0--100
YEAR	实时时钟	
MONTH	实时时钟	
DAY	实时时钟	
HOUR	实时时钟	
MINUTE	实时时钟	
SECOND	实时时钟	

WEEKDAY	实时时钟	

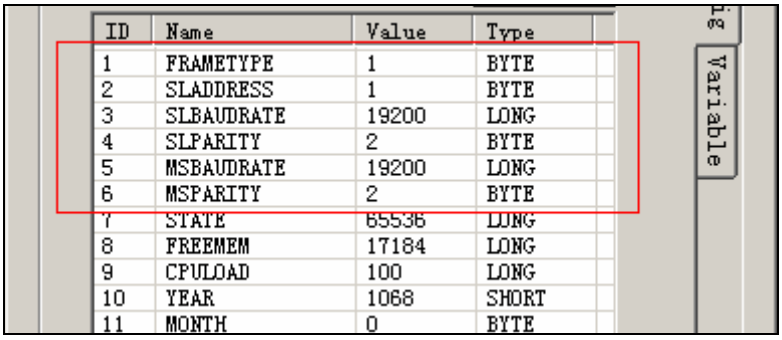
应用举例

由于小巧的体积，强大的功能，极高的可靠性以及低廉的价格。实际的应用范围只取决于大家的想象和创意。下面是几个完全没有创意的例子，目的只是说明基本的功能实现过程。

1 用作 Modbus IO 与组态王等的 HMI 软件共同实现 DCS 控制系统。

这是一个最简单的使用方式，只使用了通讯功能。采用如下方式进行设置即可，无需编程。还可以将模组的工作模式固定为 stop 模式，但是这种方法会使 slave 端口的协议格式固定为 19200 波特率 偶校验 和 Modbus ASCII 的帧格式（设定的参数无效）。

使用配置软件配置 Modebus Slave 的协议帧格式（modebus ascii 或者 modebus rtu）、网络的工作速率、奇偶校验、从站地址。



ID	Name	Value	Type
1	FRAMETYPE	1	BYTE
2	SLADDRESS	1	BYTE
3	SLBAUDRATE	19200	LONG
4	SLPARITY	2	BYTE
5	MSBAUDRATE	19200	LONG
6	MSPARITY	2	BYTE
7	STATE	65536	LONG
8	FREEMEM	17184	LONG
9	CPULOAD	100	LONG
10	YEAR	1068	SHORT
11	MONTH	0	BYTE

为 usercode 上载只包含一个数字输入或者逻辑处理之类的组态的.dc 文件（用 FBDStudio 生成）即可。

然后配置好参数后就可以接入 Modbus 网络内使用了。

2 智能传感器提供数据采集功能和部分控制

使用 FBDStudio。

先将采集通道的资源用鼠标拖放到设备组态区。

设置每个输入输出资源的参数。

对于超限等的逻辑控制可以使用 stdlgc 和 stdmath 等作比较和逻辑运算。

再将网络寄存器 Holdregs 用鼠标拖放到设备组态区,并输入相应的设备编号。编号是以如下的方式对应的：

Holdregs[0] ---------- 40001

…

Holdregs[31] ---------- 40032

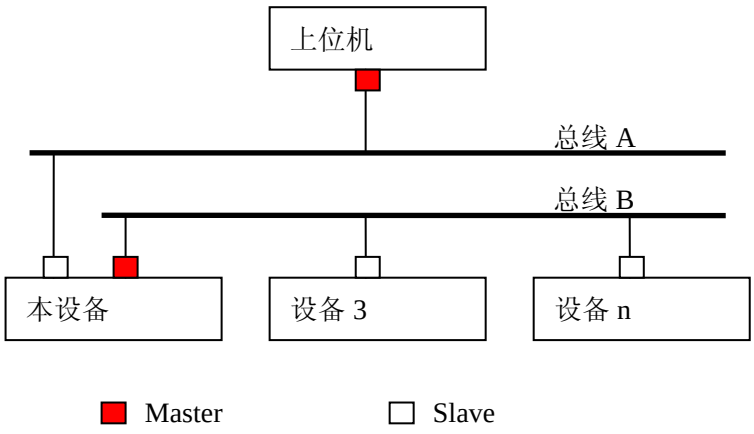
然后连接输入到 Holdregs 输入或者连接输出到 Holdregs 输出。数学运算、逻辑运算结果也可以连接到 Holdregs 输入或者直接驱动本地输出。

上载组态重新启动后新的逻辑组态即可被执行，通过网络远程应用程序可以访问传感器的值，状态。同时控制器也可以控制一些本地设备。

3 Modbus 路由

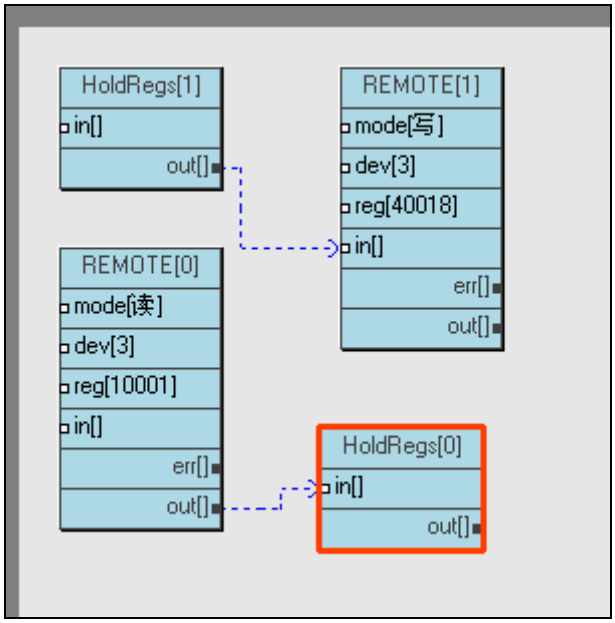
由于 0401 提供了 Modbus 主和 Modebus 从两路总线。因此通过 0401 还可以实现将主通讯口上的其他从设备的一些寄存器映射到本地的一些寄存器。也就间接实现了 Modebus 路由的功能。

下面是一个简单的例子，物理结构和本设备的部分组态分别如下：



device3 的 10001(DIN0)和 40018 分别映射到本地寄存器 40001 和 40002 。这样上位机就可以通过对本设备的 40001 和 40003 的访问实现对另外一条总线上的设备信息的存

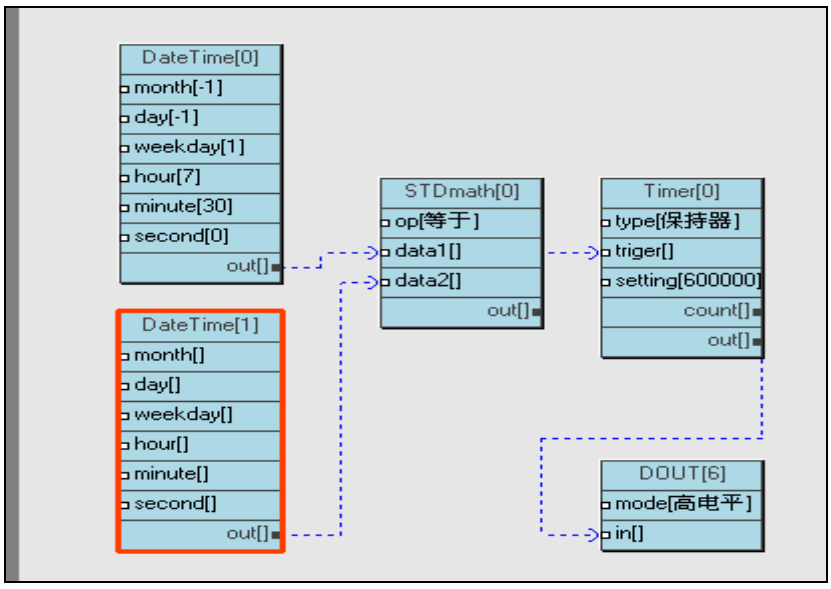
取。



4 时间控制

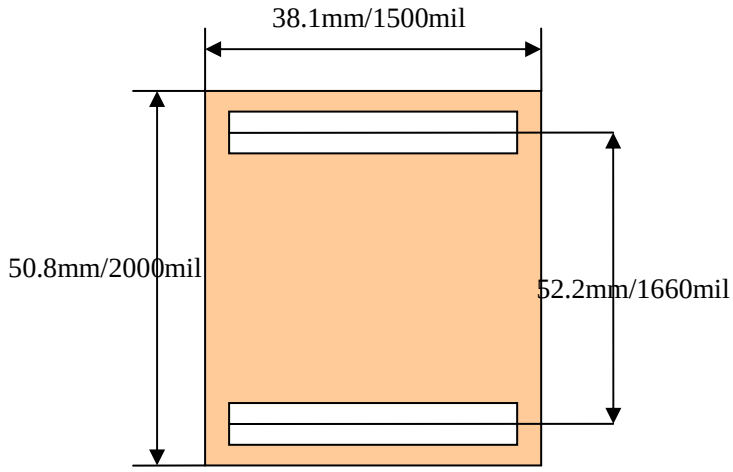
ICS-EDM-0401 系统提供了实时时钟模组，如果提供后备电池则可以确保时间的长期稳定。实时时钟提供了年月日时分秒以及星期。可以方便的实现时间控制。

下面的组态会在每个周一的早上 7：30 在 DOUT6 上启动 10 分钟。



ICS-EBM-0401

外形尺寸如图

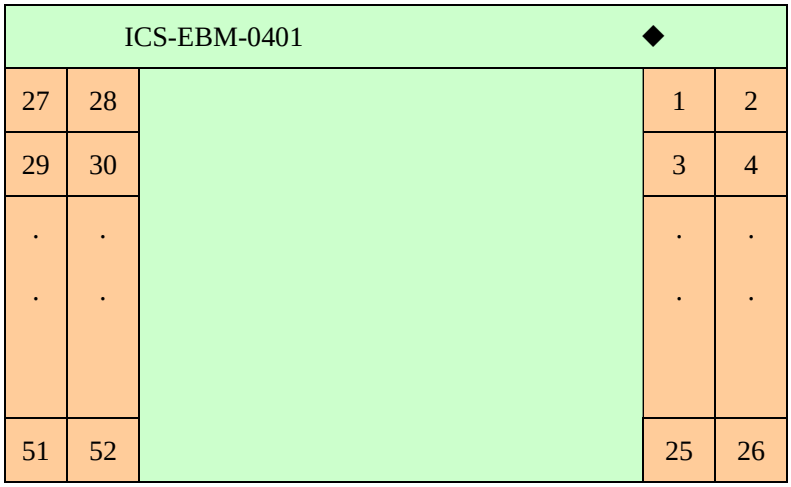


模组每秒可以处理 1 万行以上的 embed basic 脚本。

ICS-EBM-0401 引脚说明

两组引脚，每组引脚有 26 针。丝印的箭头标出第一脚位置。

详细的引脚排列见下图：



顶视图

引脚定义如下：

引脚编号	符号	含义	备注
1	DI0	开入量	

2	DI1	开入量	
3	DI2	开入量	
4	DI3	开入量	
5	DI4	开入量	
6	DI5	开入量	
7	DI6	开入量	
8	DI7	开入量	
9	DI8	开入量	
10	DI9	开入量	
11	DI10	开入量	
12	DI11	开入量	
13	DI12	开入量	
14	DI13	开入量	
15	DI14	开入量	
16	DI15	开入量	
17	VCC	5V 电源输入	
18	GND	电源地	
19	DO0	开出量	
20	DO1	开出量	
21	DO2	开出量	
22	DO3	开出量	
23	DO4	开出量	
24	DO5	开出量	
25	DO6	开出量	
26	DO7	开出量	
27	AI7	模拟输入量	
28	AI6	模拟输入量	
29	AI5	模拟输入量	
30	AI4	模拟输入量	
31	AI3	模拟输入量	
32	AI2	模拟输入量	
33	AI1	模拟输入量	
34	AI0	模拟输入量	
35	GNDA	模拟输入地	输出
36	VDDA	模拟输入电源	输出 电压 3.3V
37	VBAT	电池输入	电压 3.3V 20uA
38	GND	电源地	
39	VDD	3.3V 电源输出	输出 电压 3.3V
40	REST	外部复位	输入
41	RUN	运行指示灯（低电平）	内部已提供限流电阻
42	ERROR	错误指示灯（低电平）	
43	FI0		
44	FI1		

45	FI2		
46	STOPMODE	停止模式	启动时接地模组进入 stop 模式
47	RSLAVE	MODBUS SLAVE 接收	9600----115200
48	RMASTER	MODBUS MASTER 接收	
49	TSLAVE	MODBUS SLAVE 发送	
50	TMASTER	MODBUS MASTER 发送	
51	PWM1		100-100KHz
52	PWM2		

ICS-EBM-0401 的 Basic 语言说明

ICS-EBM-0401 使用 embed basic 语言编辑。同标准的 basic 语言相比 embed basic 有如下的特点和限制。

1 支持变量类型 long float int bool 四种；

2 关键字大小写敏感；

3 与标准 basic 相比，变量申明如 dim a,b,c as long 会申明 3 个 long 型变量而不是 1 个 Long 型变量；

4 变量名的最大长度是 8 个字符；

5 if 语句必须有 endif 语句作结束。endif 语句的 end 和 if 之间没有空格；

6 continue 语句能用于 for to next 语句和 while wend 语句中；

7 for 语句不支持 step ；

8 允许的嵌套深度为 30 级（30 组堆栈），每个 if,while,for 和 gosub 语句都需要一组堆栈；

9 如果模块有变动或者升级，支持的语句、函数以及系统变量名称等帮助内容在 Modbus 的文件的 info 中。

10 系统总共提供了约 12K RAM 的用户数据和 64K ROM 的用户代码空间（支持大约 500 个变量和 1000 行程序）。如果程序超过这个规模则不能被执行。下面是一段代码使用 EBStudio 工具 load 之后的信息：

```
=== Embedded Basic (version 1.0) ===
        Copyright 2005 John Zh

code tokenes:      100                    这个数据与代码大小有关
```

system varibles: 66 调试环境中的系统变量数
user varibles: 18 这个数据是变量和常量数量
stack size: 32 调试环境中允许嵌套的深度

RAM required: 1908 BYTE 内存开销（参考值）
ROM required: 1350 BYTE 代码开销（参考值）

具体的 rom 开销请察看编译后的.pc 文件的大小。

11 关键字和语法说明

dim **申明变量**
语法：dim <var1>[,<var2>…] as <type>
目前支持的 type 有 long int bool 和 float。
if **条件判断**
语法：if <expression> then <statemen> [else <statement>] endif
for **for 循环语句**
语法：for<exp1>to<exp2> <statement> [continue] next
表达式 1 和 2 必须是一个整型表达式，每次步长固定为 1
gosub **调用子过程**
语法：gosub <lable>
lable 子过程名 系统内按照变量来处理，所以必须符合变量命名规范。
return **子过程返回**
语法：lable:<statement> retrun
while **while 循环语句**
语法：while <expression> <statement>[continue <statement>] wend
expression 为一个逻辑表达式，如果是数字则非零及表示 True。
float **类型转换为浮点**
语法：float(<expression>)
long **类型转换为整数**
语法：long(<expression>)
wait **事件等待函数**
语法：wait (<timeout>,<expression>)
expression 逻辑表达式，当逻辑表达式成立（为 true）函数返回
timeout 最长等待时间，当时间超过 timeout 后函数返回。如果 timeout 为 0 则一直等待，直到表达式事件发生。
如果超时返回值为 ture
否则返回值为 false
sleep **延时函数**
语法：sleep <expression>
beep **“哔”音**
语法：beep
主要用于调试，在实际硬件中可能没有对应功能
timer **系统 tick 数（启动到目前的毫秒数）**

语法：timer()

sqrt 平方根

语法：sqrt(<expression>)

expression 可以使整型表达式或者浮点表达式

sin 正弦函数

语法：sin(<expression>)

expression 弧度值

sin 余弦函数

cos(<expression>)

expression 弧度值

rddev Modbus 读

语法：rddev(<deviceaddress>,<registeraddress>,<var>)

deviceaddress 被访问的设备的 modbus 地址

registeraddress 要访问的寄存器地址，地址要符合 modbus 协议的规范。

在 modbus 协议中

0xxxx 代表可读写开关量 比如 DO

1xxxx 代表只读开关量 比如 DI

3xxxx 代表只读寄存器（16bit 整型） 比如 AI

4xxxx 代表可读写寄存器（16bit 整型） 比如 AO

var 变量名

如果超时或错误返回值为 ture

否则返回值为 false

超时的时间为 200ms

wtdev Modbus 写

语法：wtdev(<deviceaddress>,<registeraddress>,<var>)

deviceaddress 被访问的设备的 modbus 地址

registeraddress 要访问的寄存器地址，地址要符合 modbus 协议的规范。

在 modbus 协议中

0xxxx 代表可读写开关量 比如 DO

1xxxx 代表只读开关量 比如 DI

3xxxx 代表只读寄存器（16bit 整型） 比如 AI

4xxxx 代表可读写寄存器（16bit 整型） 比如 AO

var 变量名或者常量

如果超时或错误返回值为 ture

否则返回值为 false

超时的时间为 200ms

end 结束

语法：end

运算符号

+	加	-	减
*	乘	/	除

%	模（余）	=	相等
>=	大于等于	<=	小于等于
<>	不等于	<	小于
>	大于	xor	异或
or	或	and	与
!	非	not	非

ICS-EBM-0401 Modbus 寄存器说明：

地址	功能	备注
00001----00008	开出量	对应 DOUT[0—7]
10001----10016	开入量	对应 DIN[0—15]
30001----30008	模入量	对应 AIN[0-7]
40001----40015	可读写寄存器	通过网络设定后断电保持 可用于参数设定
40016	可读写寄存器	Pwm 开关频率(50--32000) 断电保持。程序中改变无效。
40017	可读写寄存器	Pwm1 输出值（0--1023）
40018	可读写寄存器	Pwm2 输出值（0--1023）
40019---40032	可读写寄存器	断电不保持，用作一般通讯

ICS-EBM-0401 文件说明：

info：型号信息文件，可以下载保存。文件是后缀为 htm 的帮助文件。可以选中后点击下载按钮进行下载。

usercode：用户的 embed basic 可执行代码，使用 EB_Studio 工具编辑和编译生成。只能使用上传方式更新，不能下载。

应当上载 EB Studio 产生的相应设备的 .pc 格式文件。

如果没有用户代码，使用空行（一个回车）或者只有 “end” 的空的 basic 文件编译后上载

即可。

ICS-EBM-0401 参数说明：

名称	含义	备注
FRAMETYPE	modbus 协议帧格式	1—ASCII / 2—RTU
SLADDRESS	从地址	
SLBAUDRATE	从波特率	
SLPARITY	从校验格式	0--无校验 1--奇校验 2--偶校验 3--强制 1 4—强制 0
MSBAUDRATE	主波特率	
MSPARITY	主校验格式	见从校验格式
STATE	状态	65536 运行 196608 停止 写入 0x01000000 系统会复位，重新启动 写入其他数据可能会永久损坏模块内的程序
FREEMEM	剩余内存	BYTE
CPULOAD	CPU 负载	0--100
YEAR	实时时钟	
MONTH	实时时钟	
DAY	实时时钟	
HOUR	实时时钟	
MINUTE	实时时钟	
SECOND	实时时钟	
WEEKDAY	实时时钟	

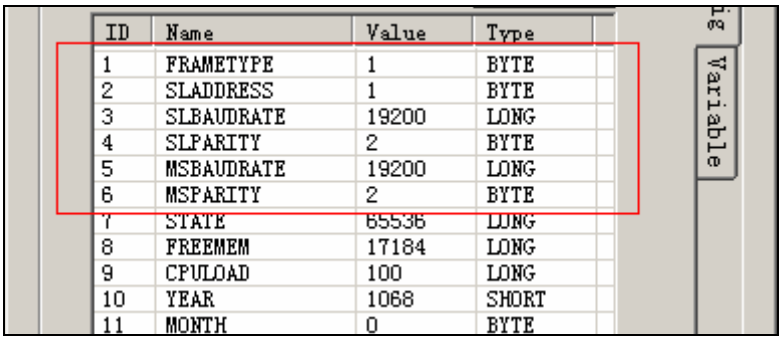
应用举例

由于小巧的体积，强大的功能，极高的可靠性以及低廉的价格。实际的应用范围只取决于大家的想象和创意。下面是几个完全没有创意的例子，目的只是说明基本的功能实现过程。

1 用作 Modbus IO

这是一个最简单的使用方式，只使用了通讯功能。采用如下方式进行设置即可，无需编程。还可以将模组的工作模式固定为 stop 模式，但是这种方法会使 slave 端口的协议格式固定为 19200 波特率 偶校验 和 Modbus ASCII 的帧格式（设定的参数无效）。

使用配置软件配置 Modebus Slave 的协议帧格式（modebus ascii 或者 modebus rtu）、网络的工作速率、奇偶校验、丛站地址。



ID	Name	Value	Type
1	FRAMETYPE	1	BYTE
2	SLADDRESS	1	BYTE
3	SLBAUDRATE	19200	LONG
4	SLPARITY	2	BYTE
5	MSBAUDRATE	19200	LONG
6	MSPARITY	2	BYTE
7	STATE	65536	LONG
8	FREEMEM	17184	LONG
9	CPULOAD	100	LONG
10	YEAR	1068	SHORT
11	MONTH	0	BYTE

为 usercode 上载如下的 ebs 程序（embed basic）编译后的.pc 执行文件（用 EBStudio.exe 生成）即可。

```
while true
   sleep 200
wend
end
```

然后配置好参数后就可以接入 Modbus 网络内使用了。

2 ICS-EBM-0401 中输入输出的访问

输入和输出的访问可以直接访问本地 Modebus 寄存器来实现。比如访问 DI 直接读取变量 MB00001 访问 DIN 1 。访问 DOUT 8 写变量 MB10008。

下面是一个启动停止的例子。MB00001 是停止按钮，MB00002 是启动按钮，MB10008 是负载。程序每隔 20ms 访问一次按钮状态并更新输出。

```
while true
   if    MB00001 then                    ‘停止按钮
         MB10008=false                   ‘停止负载
   else
      if    MB00002 then                 ‘按下启动和没有按下停止
         MB10008=true                    ‘启动负载
      endif
   endif
   sleep 20                             ‘停止 20ms
wend
end
```

不要对输入变量执行赋值操作，如果进行了这样的操作，可能会造成逻辑或者数据的错误。

3 ICS-EBM-0401 等待事件

等待事件可以使用 wait 语句或者 while 语句。如果使用 while 语句则程序以查询的方式等待某一个事件的发生。wait 语句中如果 timeout 为 0，则只有逻辑表达式为 true 时语句才会返回，否则如果条件不满足，语句也会在超时后返回。

下面是一个等待电机运行到停止位置的例子。MB00001 是停止位置开关，MB00002 是保护位置开关，MB10001 是错误报警，MB10008 是负载。首先我们使用查询方式编制程序。代码如下：

```
Waitok=true
while Waitok
   if    MB00002 then                   ‘异常保护
         MB10001=true                   ‘报警信号
         MB10008=false                  ‘停止电机
         Waitok=false
   else
      if    MB00001 then                ‘停止位置
         MB10008=false                  ‘停止电机
         Waitok=false
      endif
   endif
   sleep 1                              ‘停止 1ms
wend
end
```

我们再使用 wait 语句重新实现上面的功能

```
wait 0, MB00002 or MB00001          ‘等待停止位置或异常保护事件
                                    ‘如果需要这里还可以设置超时
MB10008=false                       ‘停止电机
if   MB00002 then                   ‘如果异常保护
        MB10001=true                ‘报警信号
endif
end
```

可以像下面一样判断 wait 语句是否是超时退出的

```
rv=wait(5000, MB00002 or MB00001)   ‘等待停止位置或异常保护事件
                                    ‘设置超时为 5 秒
MB10008=false                       ‘停止电机
if rv or MB00002 then               ‘如果异常保护或者超时
        MB10001=true                ‘报警信号
endif
end
```

4 ICS-EBM-0401 使用 Modbus 网络

在 embed basic 中使用 modbus slave 网络只需要简单的访问 modbus 寄存器变量即可。无需其他的编程工作。

可以使用 rddev 和 wtdev 函数通过 modbus 网络读写其他从设备的寄存器。比如远程的控制设备或执行设备。

下面是一个简单的网络根据远方的控制设备的采集数据启动另一个远方的设备。

设备 A 设备地址 2 远程采集设备 寄存器 40001 远程温度寄存器

设备 B 设备地址 3 另一个机房中的设备 寄存器 00003 2 号电机

```
dim valuea,valueb
deva=2                          ‘设备地址
devb=3                          ‘设备地址
rddev deva, 40001 ,valuea       ‘如果需要这里应该处理网络错误
if valuea>1000 then
    valueb=true
else
    valueb=fasle
endif
wtdev devb,00003,valueb         ‘启动或停止电机
end
```

网络和设备工作正常的情况下，在网络速度为 115200，读取函数大约在 10ms 到 20ms 左右返回。

在网络速度为 9600 时，读取函数大约在 14ms 到 24ms 左右返回。如果网络异常或者对方设备异常，函数会在 200ms 后返回。

判断网络异常可以像下面一样使用 rddev 或者 wtdev 函数。

```
rv=rddev( deva, 40001 ,valuea)
if rv then
        MB10001=true                ‘处理网络错误 报警
else
    if valuea>1000 then
        valueb=true
    else
    …
endif
```

5 ICS-EBM-0401 实现参数的设定

在使用网络中的例子中对需要访问的设备地址和寄存器地址是固定的写在了程序之中。如果设备地址等需要修改则需要重新修改程序。这样不是很灵活，也不是很方便。

0401 系列的 40001 到 40016 的 modbus 寄存器具有掉电保持的能力。适合实现这样的参数保存。

MB40005　　动作温度　　（1000 代表 100%）

```
rddev deva, 40001,valuea
if valuea> MB40005 then
      valueb=true
else
      valueb=fasle
endif
wtdev devab, 00003,valueb          ‘启动或停止电机
end
```

这样我们可以使用配置工具或者以后通过控制界面就可以在维护时或者运行时设定设备的控制器的工作参数。

注意：如果工作参数需要频繁的通过网络更新，则不适合使用具有掉电保持功能的寄存器。一方面这样会对程序的执行效率有一定的影响。同时掉电保持功能的寄存器只能存取有限的次数（一般可以可靠的存取 10 万次到 300 万次），频繁的更新操作会缩短控制器的寿命。如果需要频繁调整这种情况，可以使用没有掉电保持功能的寄存器实现。比如 40022。

6 用作机器人控制和教学

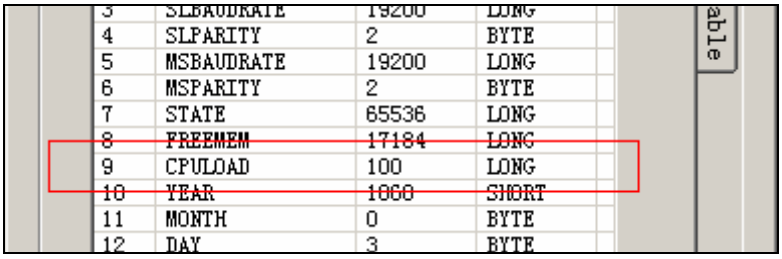
由于使用使用 basic 语言，可以方便的用于小型机器人的控制，用于计算机和手工等教学中。

7 脚本执行速度测试

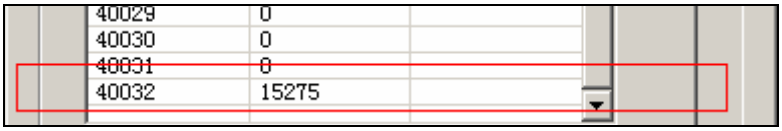
测试脚本如下：

```
'speed test    --passed
dim s,st as long
while true
   s=0
   st=timer()
   while (timer()-st)<1000
      s=s+1
      s=s+1
      s=s+1
      s=s+1
      s=s+1
   wend
   MB40032=s
wend
end
```

结果存放在 modbus 寄存器 40032 中。当进行速度测试时模组的运行指示灯闪烁会变慢，运行指示灯的点亮的时间不再短促，还可能闪烁的没有规律。这是因为上面的语句是一个死循环，这样的代码会高速的执行并且消耗所有的 CPU 资源，这对正常运行的指示灯的闪烁任务造成了影响从而产生了上述的现象。当然这时的网络通讯不会有任何问题，我们还可以通过网络访问模块的信息。



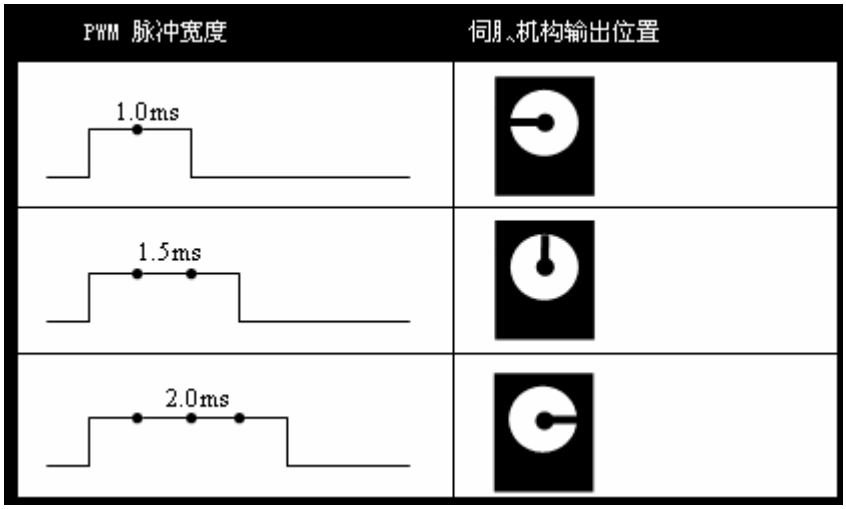
通过网络察看运行状态可以看到 CPU 负载为 100%。



这是我们实际读取到的结果 **15275** 。

8 机器人 Servo 控制（RC Servo Controller）

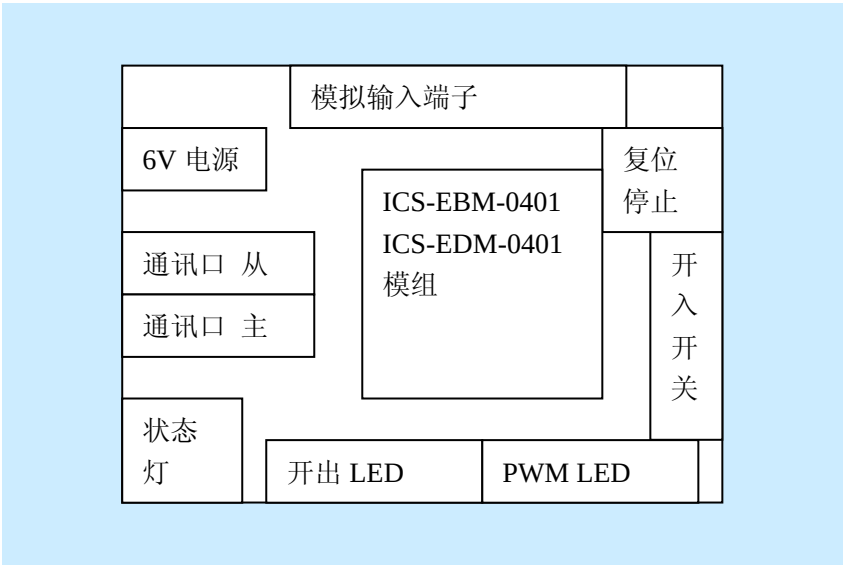
我们可以利用pwm通道来控制RC SERVO。一般的舵机可以设置pwm频率为50Hz也就是20ms。某些舵机可能需要设置为 100Hz。这时通过写 pwm 通道的值就可以控制了。需要注意的是，pwm 的值的范围需要控制在 51---102 之间（50Hz）或者 102---204 之间（100Hz），这时的 PWM 脉冲宽度在 1ms 到 2ms 之间。



开发板工具包 ICS-ENM-0401 KIT

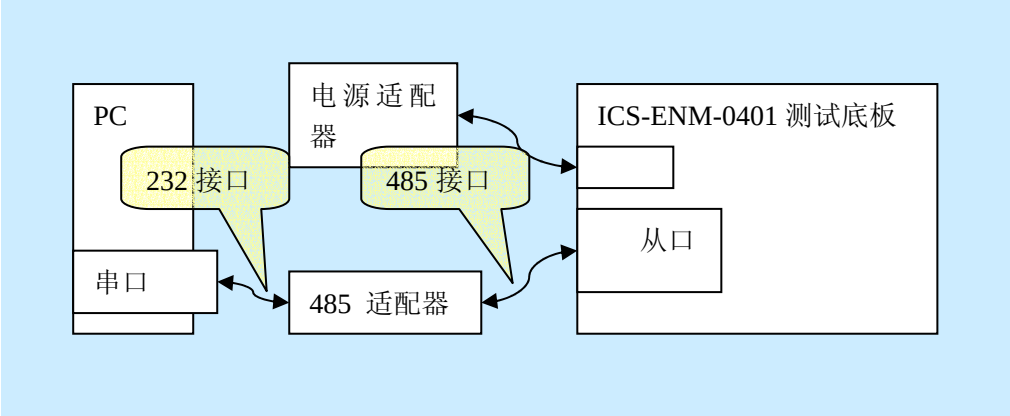
开发工具包包括开发底板，ICS-EBM-0401 , ICS-EDM-0401, 485 通讯适配器，电源适配器， EBStudio, FBDStudio 和 XBI_Utility 等开发软件。在安装和使用这些软件之前请详细阅读每个软件的安装说明和版权申明。

开发底板布局结构：



开发环境安装

将 ICS-EBM-0401 模组插入底板。注意引脚标记要对应。然后按照下图安装 485 通讯适配器和电源。

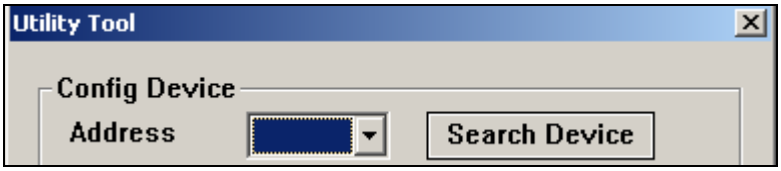


如果打算开发 EDM 请安装 FBDStudio 和 XBI_Utility。如果开发 EBM 请安装 EBStudio 和 XBI_Utility 软件。注意在运行 FBDStudio 前可能需要安装微软的 xml 组件。

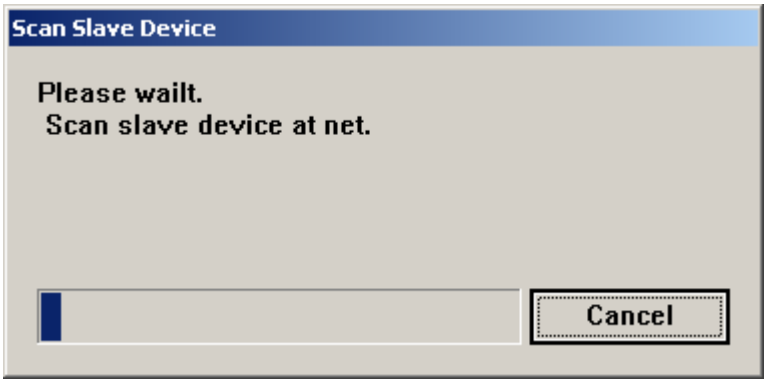
开发过程介绍

使用维护软件，首先按下测试底板上的 stop 按钮然后为底板上电或者按下 stop 后按复位按钮。在运行指示灯两次闪烁后释放 stop 按钮（约 1-2 秒）。这时 EDM 或 EBM 模组工作在 STOP 模式下。启动 XBI_Utility 软件选择 PC 所使用的串口，STOP 模式下波特率是 19200 偶校验,如果端口中没有正在使用的串口，这是因为该串口已经被其他的程序所使用造成的。退出其他程序，重新启动 XBI_Utility 即可找到使用的端口。

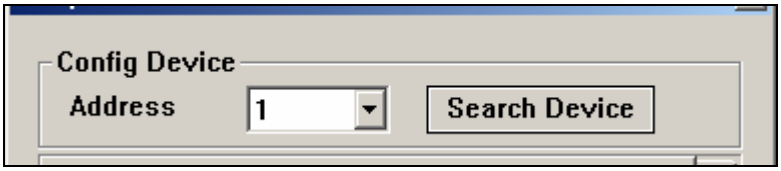
打开端口后出现配置界面。



按寻找设备寻找网络上的从设备。Stop 模式下的从设备地址范围被限定倒 1-16。所以不必等到全部搜索完毕。



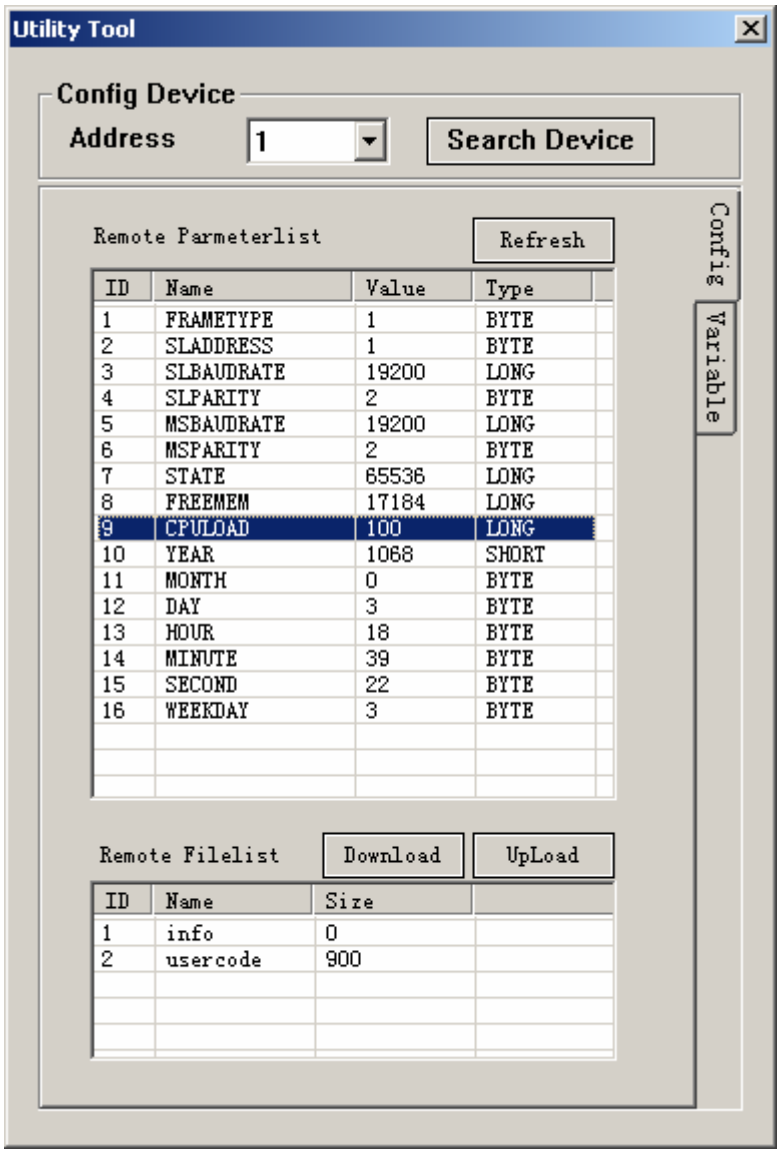
这时再在地址栏中选择配置的设备。



然后我们就可以看到配置信息了。我们可以通过按刷新按钮刷新当前信息。在配置信息了分了两个部分。一部分是参数，另一部分是文件。最主要的参数是通讯参数了。第一

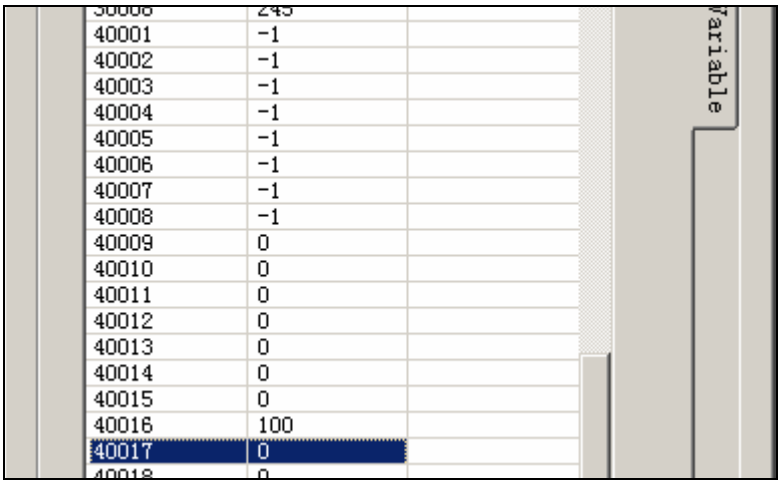
项是 modbus 协议的帧格式。然后依次是 从地址，从波特率，从校验格式，主波特率，主校验格式等等。

文件包含 Info 和 usercode 两个文件，分别支持上载和下载。这时我们选中 usercode 文件，然后按上载按钮。选择光盘中提供的 demo.pc（ICS-EBM 模块）或 demo.dc（ICS-EDM 模块） 文件。上载完毕后按底板上的复位按钮，这是系统会重新启动并且执行我们刚才上载的文件。



我们还可以通过选择寄存器标记查看 Modbus 寄存器的信息，对于可以写入的寄存器可以双击参数通过网络设定。

EBM-0401 中可以双击 40017。然后设置为 20 确定。



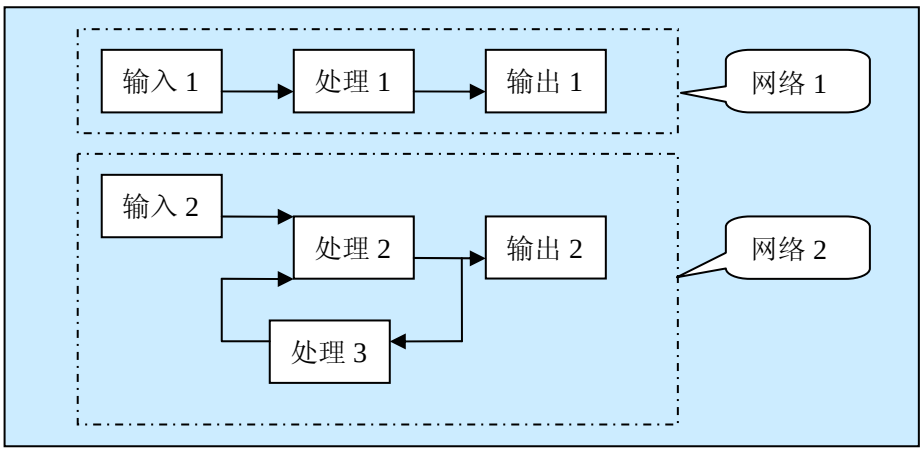
这时我们可以观察到开发板的 pwm2 微弱发亮。

EDM-0401 中可以设置 00008 为 1。可以观察到开发板的 LED8 发亮。

ICS-EBM　ICS-EDM 在编程上和执行上的区别

ICS-EBM- 采用 BASIC 语言编程。ICS-EBM-0401`系统只有一个用户程序在顺序的执行。所以更符合对过程性的控制过程进行处理。如果有多个任务需要同时控制则程序的编写需要一点技巧。在以后我们将推出 Embed Basic Multithread 版本(允许有多个用户程序同时执行)，方便并行任务的开发和执行效率。

ICS-EDM 采用图形化的方式编程，如果有多个逻辑网络，则这些逻辑网络是并行运行的（同时运行）。对于一个网络内的功能块则以数据流动的方向依次处理。如果网络内有环路，环路首先被按照一个节点的方式处理，环路内部系统会随机从某一个环路中的数据块开始处理，所以在有环路时必须设置好初始值。如下：（新版本的 FBD Studio 将会提供由用户指定功能块的处理顺序功能）



组态中 网络 1 和网络 2 同时运行。

网络 1 的处理顺序是：输入 1，处理 1，输出 1

网络 2 的处理顺序是：输入 2，处理 2，处理 3，输出 2

或者是输入 2，处理 3，处理 2，输出 2。

在 FBD 逻辑处理的过程中，本地输入的所有信号都不会被更新，只有当逻辑处理完毕才会进行本地 I/O 的更新。但是注意，**网络数据可能会在处理的过程中发生改变**。

而在 Embed BASIC 中，本地 I/O 会不断的更新。因为一般的 BASIC 应用都是按照下面的方式设计的,除了发生致命的错误，程序永远不会退出。也就不能在 basic 应用开始和结束时进行 I/O 的更新操作了。

```
‘这是一个典型的应用程序结构
变量申明
初始化工作
while true
    应用程序主体
wend
end

辅助子程序
```