

HR48 系列 PLC

HR48A & HR48B

主要配置：

- 所有输入输出包括开关量输入输出、模拟量输入输出、高速脉冲输入输出以及 PWM 输出通道均为光电隔离。
- 两个独立的通讯接口 RS232、RS485，其中 485 通讯口完全光电隔离。两通讯口都为自由编程通讯口，CRC 校验程序（MODBUS 标准）已经嵌入控制器内部，可自由编程取舍。
- 宽范围的通讯波特率：1200 B/s 至 115.2k B/S.
- 2 路高速脉冲计数，宽频率范围的脉冲输入 0 至 600kHz。
- 2 路高速脉冲输出，宽频率范围的脉冲输出 1 至 600kHz。
- 16k 字节闪存程序存储器，不需电池永久保持程序；16k 数据存储器掉电保持 10 年
- 全封闭铝合金外壳设计，最优良的散热性能和安全性。
- 软硬看门狗和高超的抗干扰性能和稳定性。
- 60 个定时器 定时范围：0.1 至 6553.5 秒
- 16 个普通计数器，计数范围：0 至 65535.
- 250 个输出继电器 Out0 至 Out249
- 250 个输入继电器 In0 至 In249
- 中间继电器 rly0 至 rly499
- 记忆中间继电器 mrly0 至 mrly99
- 基本指令周期 0.45us 程序循环周期 8.5ms
- 日历功能：年、月、日、星期、时、分、秒
- 电源：86V ~ 240V AC50 ~ 60 Hz
或 150V ~ 320VDC 功率消耗：2.5W



编程软件：恒日控制器编程软件 xc.exe

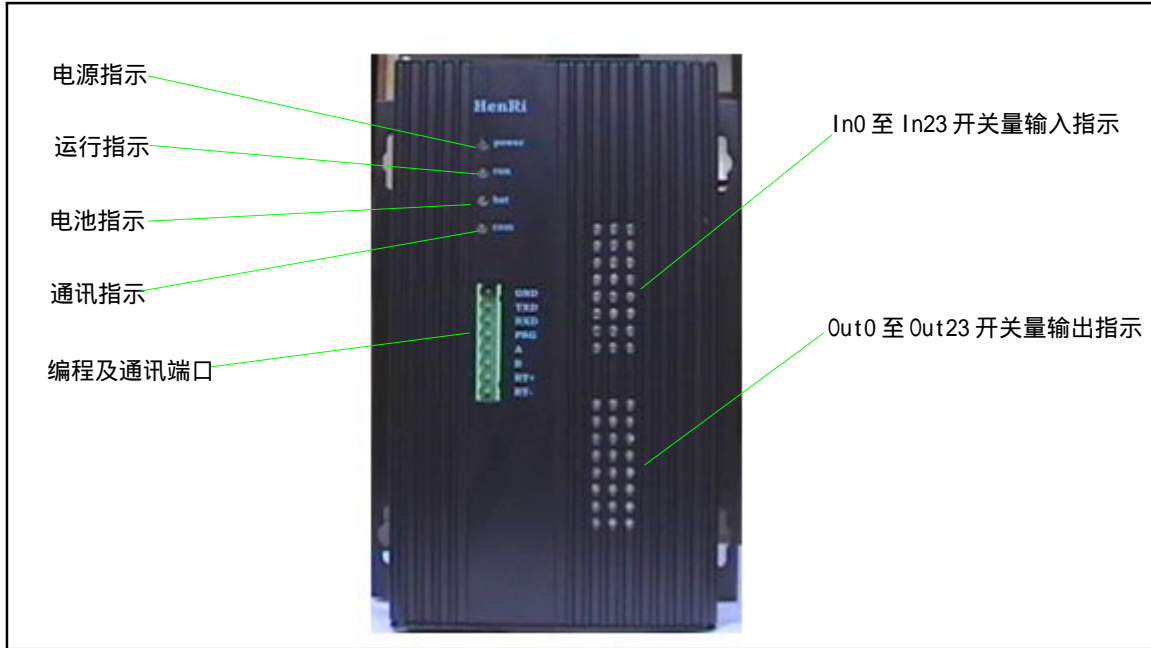
HR48A:

- 24 个开关量输入点
- 24 个 NPN 晶体管开关量输出点
- 2 个可加减高速脉冲输入通道(频率: 0 至 600kHz 计数范围: -4,000,000 至 4,000,000 ,
- 2 个高速脉冲输出通道(1Hz 至 100kHz)随时可以改变,是目前输出频率范围最宽的产品。
- 支持加(+)-减(-)-乘(*)-除(/)数学运算

HR48B:

- 24 个开关量输入点
- 24 个 NPN 晶体管开关量输出点
- 8 个 0 至 5V 的 12 位分辨率模拟量输入,最小分辨率 $5/4096=1.2\text{mV}$ 。
- 4 个 0 至 5V 的 8 位模拟量输出点(0 至 10V 可选)。
- 2 个 PWM 输出点 (定制)
- 2 个可加减高速脉冲输入通道(频率: 0 至 600kHz 计数范围: -4,000,000 至 4,000,000)
- 2 个高速脉冲输出通道(1Hz 至 600kHz)随时可以改变。
- 8 个 PID 函数通道
- 支持单字节、双字节、浮点数。
- 支持加(+)-减(-)-乘(*)-除(/)比较\数学运算,正弦函数、余弦函数、开平方函数、对数函数,

控制器信号指示：



bat 电池：

除了系统自身要求上电清零的数据以外，所有数据都是断电保持，数据至少保持十年，当电池不足时，bat 指示会闪烁，同时，控制器会关闭所有输出，取下控制器里面的钮扣电池，必须在 20 分钟之内更换新的电池，否则数据可能丢失。

run 运行指示：

常亮，发生故障时会闪烁，此时请联系经销商。

com 通讯指示：

当 RS232 和 RS485 两个通讯口任意一个通讯口有数据发送或接收时，该指示灯都会闪烁。

随机电源：

控制器的随机电源可以提供 12V 200mA 的电流，RT- 是电源负极，RT+ 是电源正极。

下载控制：

程序下载时利用了控制器的 RS232 通讯口，在 PRG 的控制下完成程序的下载，不影响用户对 232 口的编程。

RS232 通讯口：

GND TXD RXD 自由通讯口

RS232 收发缓冲区为 250，自由通讯口，兼容

MODBUS 协议，RS232 相关编程指令：

```
AddVariableTo232TB
GetVariableFrom232TB
Start232Transmit
Set232Port
AddCRCTo232TB
AddNumberTo232TB
AddVariableTo232TB
Rs232CRCCheck
```

RS485 通讯口：

A B 自由通讯口

RS232 收发缓冲区为 30，自由通讯口，兼容

MODBUS 协议，RS485 相关编程指令

```
AddVariableTo485TB
GetVariableFrom485TB
Start485Transmit
Set485Port
AddCRCTo485TB
AddNumberTo485TB
AddVariableTo485TB
Rs485CRCCheck
SetAddress
```

RT+, RT-：

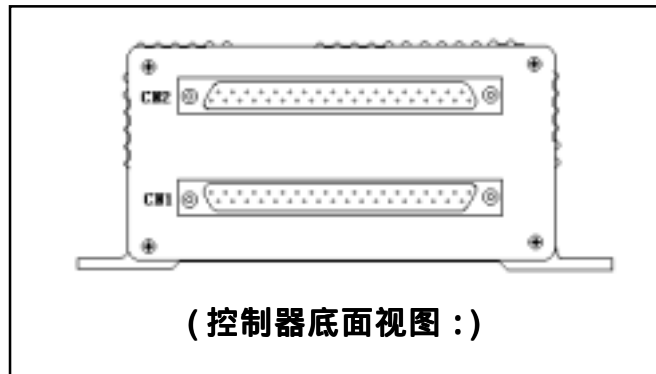
12V 直流电源输出，可以用于继电器、传感器电源等，容量 300mA

RT+：12V 直流电源输出正端

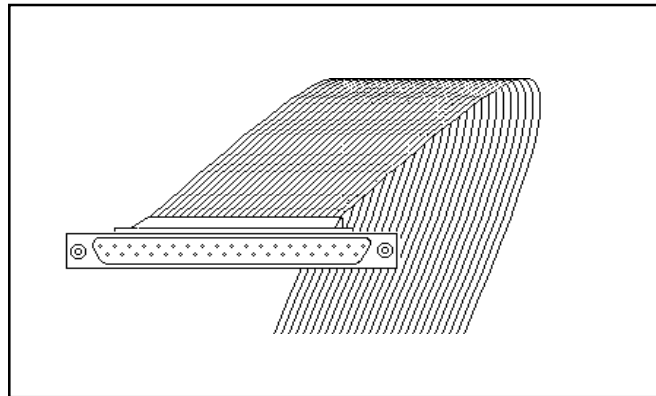
RT-：12V 直流电源输出负端，这个端子在控制器内部已经与控制器的 COM 端接在一起，所以，当用这个 12V 电源驱动直流继电器时，不再需要共地接线。

控制器的输入输出接线：

输入输出信号通过控制器下面的两个 DB37 接口 (male) 接入控制器内，控制器底面接线接口如下所示：



接口电缆 (DB37 female) 采用 1.5mm37 芯彩虹排线，长度:1m。

**CN1 接口的接线颜色和排号：**

1 黑 L(AC 220V 电源输入)	21 黑 In7(开关量输入 7)
2 白 N(AC 220V 电源输入)	22 白 In8(开关量输入 8)
3 灰 DIS NPN 输出尖峰电流泄放端	23 灰 In9(开关量输入 9)
4 紫 DA0_Output(HR48B 模拟量输出 0)	24 紫 In10(开关量输入 10)
5 蓝 DA1_Output(HR48B 模拟量输出 1)	25 蓝 In11(开关量输入 11)
6 绿 DA2_Output(HR48B 模拟量输出 2)	26 绿 In12(开关量输入 12)
7 黄 DA3_Output(HR48B 模拟量输出 3)	27 黄 In13(开关量输入 13)
8 橙 HSCounter1Vol Input(高速计数器 1 通道)	28 橙 In14(开关量输入 14)
9 红 HSCounter0Vol Input(高速计数器 0 通道)	29 红 In15(开关量输入 15)
10 棕 HSPulse1Out Output(高速脉冲输出 1 通道)	30 棕 In16(开关量输入 16)
11 黑 HSPulse0Out Output(高速脉冲输出 0 通道)	31 黑 In17(开关量输入 17)
12 白 PWM0Width(HR48B PWM 输出通道 0)	32 白 In18(开关量输入 18)
13 灰 PWM1Width(HR48B PWM 输出通道 1)	33 灰 In19(开关量输入 19)
14 紫 In0(开关量输入 0)	34 紫 In20(开关量输入 20)
15 蓝 In1(开关量输入 1)	35 蓝 In21(开关量输入 21)
16 绿 In2(开关量输入 2)	36 绿 In 22(开关量输入 22)
17 黄 In3(开关量输入 3)	37 黄 In 23(开关量输入 23)
18 橙 In4(开关量输入 4)	
19 红 In5(开关量输入 5)	
20 棕 In6(开关量输入 6)	

CN2 接口的接线颜色和排号：

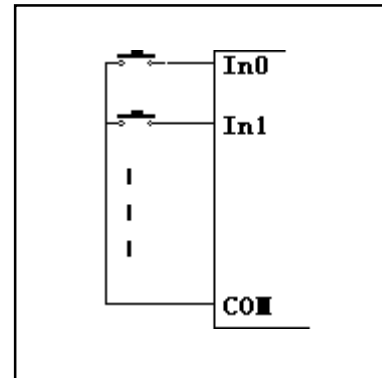
1 紫 COM	21 紫 Out7(开关量输出 7)
2 蓝 COM	22 蓝 Out8(开关量输出 8)
3 绿 COM	23 绿 Out9(开关量输出 9)
4 黄 COM	24 黄 Out10(开关量输出 10)
5 橙 AGND 模拟量地	25 橙 Out11(开关量输出 11)
6 红 ADO_Input (HR48B 模拟量输入通道)	26 红 Out12(开关量输出 12)
7 棕 AD1_Input (HR48B 模拟量输入通道 1)	27 棕 Out13(开关量输出 13)
8 黑 AD2_Input (HR48B 模拟量输入通道 2)	28 黑 Out14(开关量输出 14)
9 白 AD3_Input (HR48B 模拟量输入通道 3)	29 白 Out15(开关量输出 15)
10 灰 AD4_Input (HR48B 模拟量输入通道 4)	30 灰 Out16(开关量输出 16)
11 紫 AD5_Input (HR48B 模拟量输入通道 5)	31 紫 Out17(开关量输出 17)
12 蓝 AD6_Input (HR48B 模拟量输入通道 6)	32 蓝 Out18(开关量输出 18)
13 绿 AD7_Input (HR48B 模拟量输入通道 7)	33 绿 Out19(开关量输出 19)
14 黄 Out0(开关量输出 0)	34 黄 Out20(开关量输出 20)
15 橙 Out1(开关量输出 1)	35 橙 Out21(开关量输出 21)
16 红 Out2(开关量输出 2)	36 红 Out22(开关量输出 22)
17 棕 Out3(开关量输出 3)	37 棕 Out23(开关量输出 23)
18 黑 Out4(开关量输出 4)	
19 白 Out5(开关量输出 5)	
20 灰 Out6(开关量输出 6)	

开关量输入信号的接线：

控制器的开关量输入为电阻模式，不可以引入任何其他电压，所有输入点状态以开关量输入点和控制器公共端 COM 的电阻决定：

控制器 In0 至 In23 接通状态：接通电阻 < 1k 欧姆，通常是开关或按钮的接通电阻（一般阻值是几毫欧姆）

控制器 In0 至 In23 断开状态：断开电阻 > 15k 欧姆，通常是开关或按钮的断开电阻（一般阻值是无穷大）

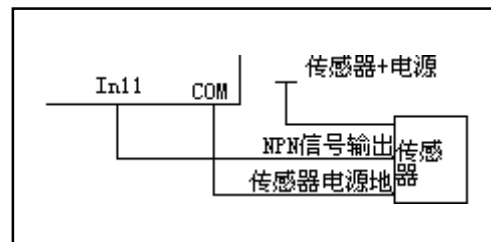


接线示意图

HR48 控制器的开关量输入只接受 NPN 传感信号输入：

1. 传感器吸入电流：6mA，传感器的导通门槛电压不得超过 1V。

2. 传感器电源地必须与控制器的公共端 COM 相连，如下图所示：



NPN 传感器信号

开关量输出信号的接线：**Out0 ~ Out23**

24 个开关量输出点

输出方式：NPN

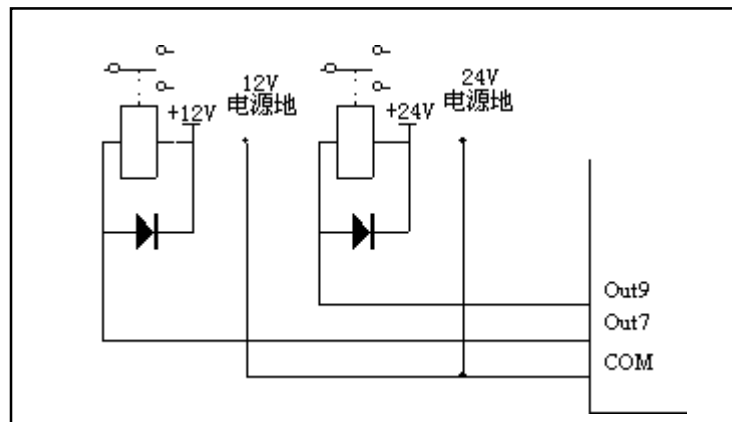
输出点容量：电流<500mA，电压<60V

接线：只能驱动直流负载，如下图所示，同时驱动一个 24V 继电器和一个 12V 继电器，上面的 12V 和 24V 都是外接直流电源，他们的低压侧即 0V 必须与控制器的公共端 COM 连接在一起，加上二极管是为了保护内部驱动电路。

编程方法：梯形图或者 C 语言

C 语言编程举例：**Out3 = on;**

结果：接通 Out3



对于每一个NPN输出点，控制器内部已经安装提供了一个泄放二极管，**当所有外接继电器采用统一一个直流电源时**，只需将控制器的DIS端与外接继电器的正电源相连即可对所有NPN输出产生尖峰电流起到泄放保护作用，不需要用户另外加接保护二极管。

高速计数器脉冲输入：

HR48 具有两路高速脉冲计数输入通道，计数频率可达几百千赫，是目前市场计数频率最高的可编程控制器。

高速计数器输入接受开关信号和 NPN 信号，接线方法与开关量输入接线相同。

低电平输入电阻： ≤ 20 欧姆

高电平输入电阻： $\geq 10K$ 欧姆

NPN 输入低电平： $\leq 1V$

HR48 提供两路高速加减脉冲计数，该计数器采用的是高性能的硬件计数器，计数频率可达 600kHz，其编程不同于其他计数器，只能在编程软件的 C 程序窗口对其读数，但不能改写，其 C 程序变量名定义是 **HSCounter0Vol** 和 **HSCounter1Vol**，只要调用这两个变量即可知道高速计数器的计数值。高速计数变量是个浮点数变量，而且高速计数通道都是加减可控，所以它们的计数范围是 -4,000,000 至 +4,000,000

高速计数器具有计停控制功能，其在 C 程序中的定义是 **HSCounter0Gate** 和 **HSCounter1Gate**，这是个 bool 型变量，开机默认状态：off，要启动计数功能，就必须将其置位 on 状态。

高速计数器的加减控制由另外一个 bool 型变量 **HSCounter0UpDown** 和 **HSCounter1UpDown** 来控制，当其为 on 状态时，计数器进行加计数，反之进行减计数，其开机默认状态是加计数。

编程举例：

```
main
{
HSCounter0Gate = on;
HSCounter0UpDown= on;
MainLoop:
    WatchDog();
    LadderProgram();
    if(HSCounter0Vol>100000){Out0 = on;}else{Out0 = off;}
    goto MainLoop;
}
```

结果是当高速计数通道 0 计数达到 100000 时，Out0 输出接通，否则就断开。

高速计数器的清零

高速计数器的计数数值具有停电记忆功能，并且只能被读取而不能被改写，但是恒日 C 提供了专门的清零功能函数，程序设计人员可以利用该函数进行设备的复位清零操作。

清零 0 号高速计数器：

```
ClearHSCounter(0);
```

清零 1 号高速计数器：

```
ClearHSCounter(1);
```

高速脉冲输出：

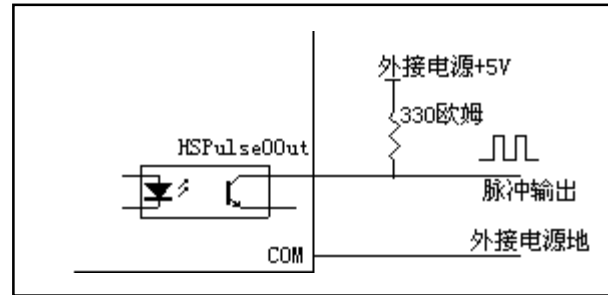
HR48 提供两路随时可变的脉冲输出，最高输出频率可达 600 千赫，是目前输出频率最高的可编程控制器。

接线：

高速脉冲输出通道都是 NPN 方式输出，要得到输出脉冲信号，必须外接负载电阻，负载电阻阻值：

5V 电源：330

10V 电源：660

**编程：**

HR48 的高速脉冲输出频率可达 600KHZ，脉冲输出控制必须在编程软件的 C 窗口编程，其脉冲输出频率由变量 **HSPulse0Out** 和 **HSPulse1Out** 决定。这两个脉冲输出变量是 word 型变量，能够表示的频率范围是 0 至 65535Hz，为了得到更高频率的脉冲输出，C 程序提供一个倍率设定函数：

SetHSPulse(LOW 或 MIDDLE 或 HIGH)；

当设置为 LOW 时，所有高速脉冲输出的频率等于变量 **HSPulse0Out** 和 **HSPulse1Out**。

当设置为 MIDDLE 时，频率将变成原来的 10 倍。

当设置为 HIGH 时，频率将变成原来的 100 倍。

恒日 C 对高速脉冲输出通道提供一个控制开关：**HSPulse0Gate** 和 **HSPulse1Gate**

当 **HSPulse0Gate = off**；时，通道 0 脉冲输出关闭；

当 **HSPulse0Gate = on**；时，通道 0 脉冲输出打开；

HSPulse0Gate 和 **HSPulse1Gate** 的开机默认状态是 off；

编程举例：(用高速脉冲输出通道 0 输出一个 1234Hz 的脉冲)

```
main
{
    HSPulse0Gate = on;

    HSPulse0Out = 1234;

    MainLoop:
        WatchDog();
        LadderProgram();

        goto MainLoop;
}
```

PWM 脉冲输出 : (定制)

PWM 脉冲输出通道都是 NPN 方式输出，接线方法与高速脉冲输出接线完全相同。

编程方法：C 语言

C 语言表示方法: PWMWidth PWM1Width

类型: byte

精度：8 位 脉冲占空比 = PWMiWidth/256;

HR24 控制器编程软件支持 2 个 PWM 输出点，
它们只可以在 C 语言窗口编程。

编程举例：(从 PWM 输出通道 0 输出一个 10% 占空比的脉冲)

程序如下：

```

main
{
    PWMWidth = 256*0.10;
MainLoop:
    WatchDog();
    LadderProgram();
    goto MainLoop;
}

```

PWM 基频设置语句： 恒日控制器 PWM 输出提供两种基本频率：7.2kHz
和 21.6kHz

```

SetPWMFrequencyHigh();    设置基频为 21.6kHz
SetPWMFrequencyLow();     设置基频为 7.2kHz

```

AD0_Input ~ AD7_Input:

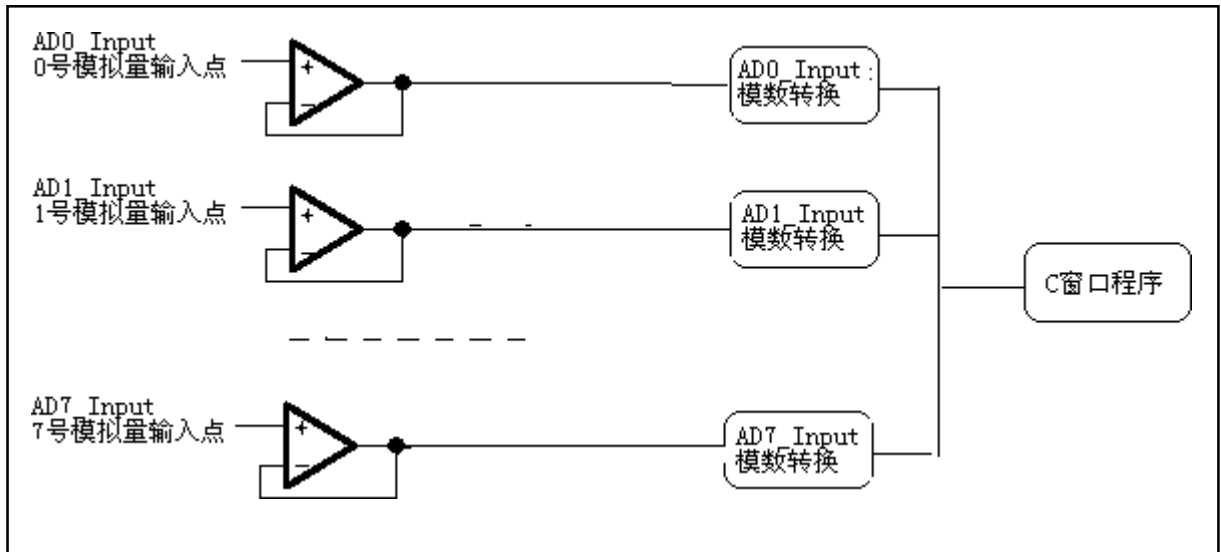
- 8 个模拟量输入点
- 精度：12 位
- 最小分辨率 $5/2048=2.4\text{mV}$
- 输入电压范围：0 ~ 5V (电流输入可以定制)
- 输入电阻：大于 100K
- 编程方法：在 C 窗口直接调用相应的名字即可
- 编程举例：if (AD2_Input>2.5){Out0 = on;}else{Out0 = off ;}
- 结果：当 2 号模拟量输入通道电压大于 2.5V 时，Out0 接通，否则，Out0 关断。

处理流程

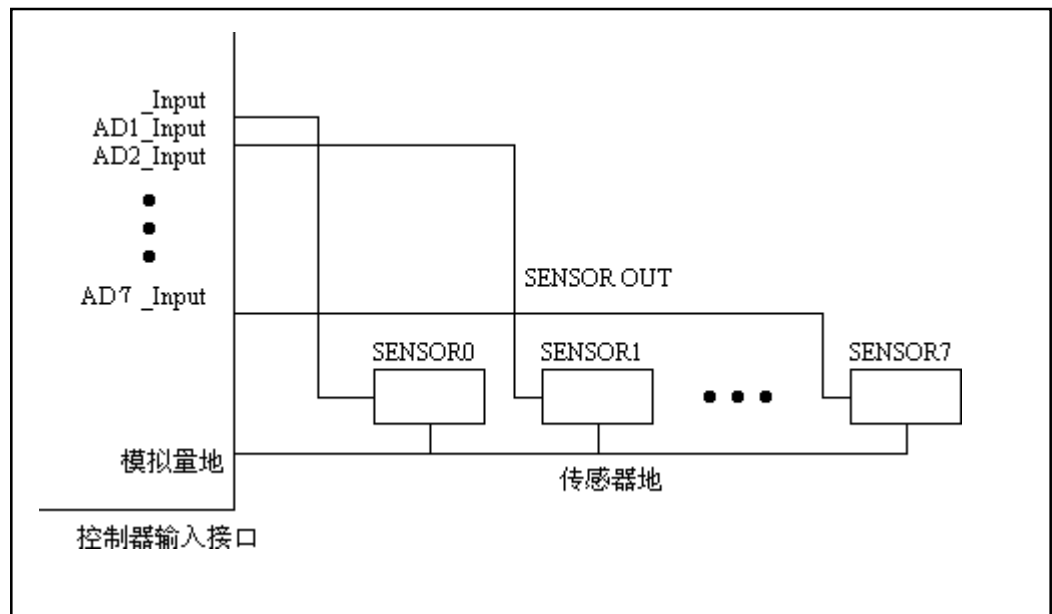
PLC 的模拟量转换模块
将输入接口的模拟量信
号转换成数字信号，并
且送给 PLC 的中央处理
器

C 窗口程序对 8 个模拟量信号：
AD0_Input ,AD6_Input
AD1_Input, AD7_Input
AD2_Input,
AD4_Input,
AD5_Input
进行判断和运算

模拟量输入通道结构示意图：



模拟量输入点的接线：



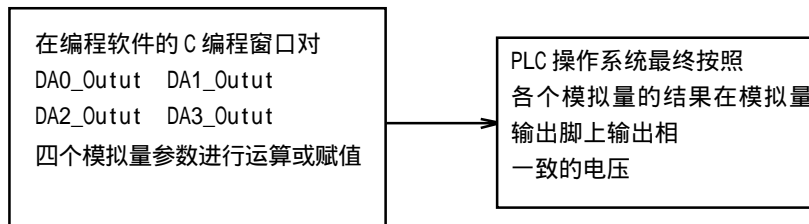
注意：

当模拟量输入点没有信号接线时，该通道将处于悬空状态，所对应的模拟量数值是一个不确定的数。

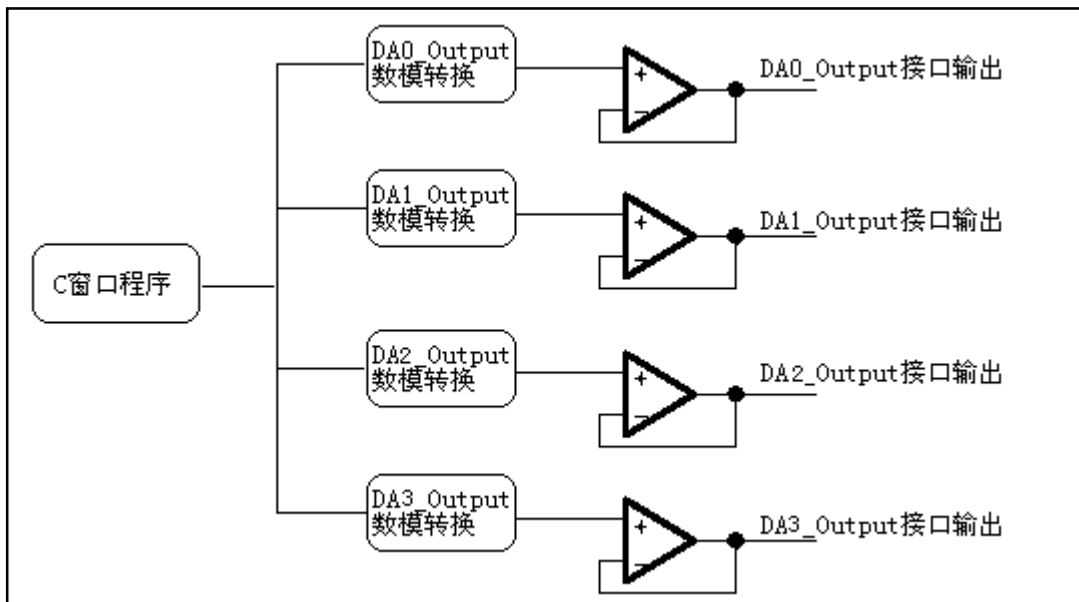
DA0_Output ~ DA3_Output

- 4 个模拟量输出点
- 精度：8 位
- 输出电压范围：0 ~ 5V 或者 0 ~ 10V 之间的任意数值
- 负载能力:10K
- 编程方法：在 C 窗口给相应的模拟量输出变量赋值
- 编程举例：DA1_Output = 2.77;
- 结果：在 DA1_Output 脚输出 2.77V 的直流电压

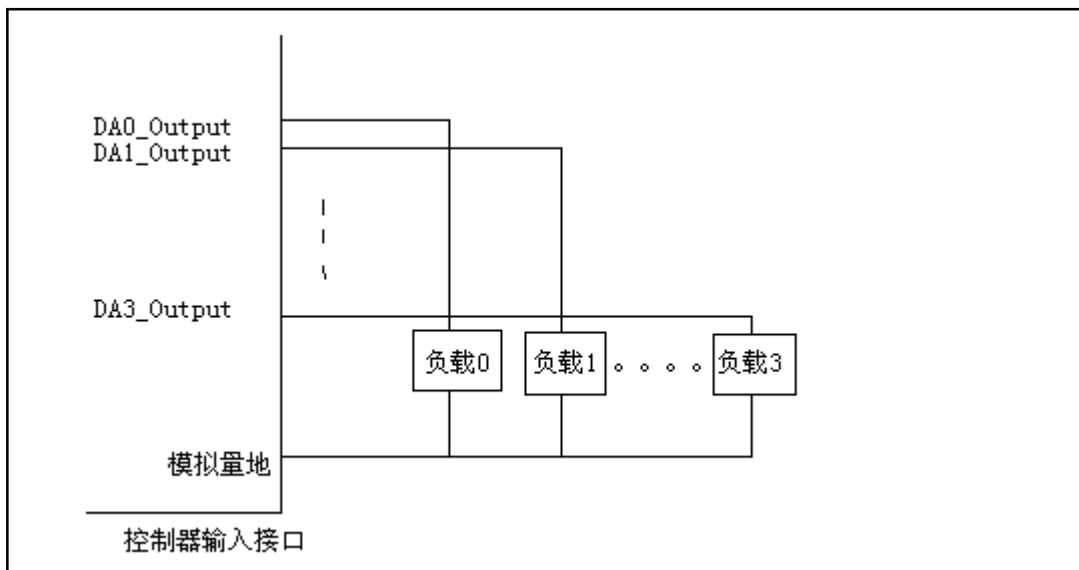
控制流程：



模拟量输出通道硬件结构：



模拟量输出接线：



日历和时钟

恒日控制器支持年、月、日、时、分、秒，在断电的情况下，日历可以运行至少十年，采用高精度定时晶体，每年的误差不会超过 3 分钟。

日历的 C 程序的表示方法：

年： Year (数字范围 00 至 99) 类型:byte, 前面的 20 被省略
月： Month (数字范围 1 至 12) 类型:byte
日： Day (数字范围 1 至 31) 类型:byte
星期：Week (数字范围 0 至 6) 类型:byte (星期日为 0)
时： Hour (数字范围 0 至 23, 24 小时制) 类型:byte
分： Minute (数字范围 0 至 59) 类型:byte
秒： Second (数字范围 0 至 59) 类型:byte

编程举例：

```
if (Year==4 && Month ==6 && Day >=8){Out8 = off;}else{Out8 = on;}
```

上面一段程序的结果是在 2004 年 6 月 8 日以后的 22 天内，Out8 将关闭，其他日期保持接通。

```
又如： if (Second==5){Out8 = on;}else{Out8 = off;}
```

结果是，每当 5 秒时，Out8 都会接通并且维持 1 秒

恒日 C 提供了一个时钟校对语句 SetClock(时,分,秒);该指令可更改时钟的时、分、秒，可以用来校正时钟。

```
例如：if (In1){ SetClock(0,0,0);}
```

该语句执行后，系统时钟的时、分、秒将从 0 点 0 分 0 秒开始，其他日期年、月、日等不受影响。

日历设定：

下载对话框的日历设置可以对控制器的日历功能进行预置，预置之后控制器将按照新的日历运行，更换新的日历必须输入控制器的编程密码方可进行更改。

数型转换

恒日控制器支持单字节正整数(byte)、双字节正整数(word)、浮点数(float)和布尔型(bool)四种变量。各种变量之间的属性转换除了 bool 型比较特殊以外,其他几种只需用等号表达式即可实现转换,恒日 C 在进行表达式运算时总是将等号右边的所有变量转换成等号左边的变量类型,然后在进行运算,例如,当等号左边是一个 word 型变量而等号右边是一个 float 型变量时,编译器会首先将浮点变量转换成 word 变量然后赋给左边变量,这样就完成了右边 float 型变量到左边 word 型变量的转换。

例如:下面表达式将一个 0 至 5V 的模拟量输入放大 50 倍,转换成 0 至 250 之间的浮点数,然后转换成 0 至 250 之间 byte 型正整数。

```
byte   my_byte;
float  my_float;
my_float = AD0_Input*50;
my_byte = my_float;
```

对于 bool 型变量,恒日 C 之提供了两个特殊的语句,以实现一个 byte 型变量到一系列 bool 型变量的转换和一系列 bool 型变量到一个 byte 型变量的转换。

1: byte 型变量转换成一系列 bool 型变量

指令格式: ByteToBool (byte 型变量, bool 型变量);

例如: ByteToBool (Second, Out4);

一个 byte 型变量由 8 位数据位组成,执行完上面的语句后,byte 变量的第一个数据位将赋给 bool 型变量 Out4,第二个数据位将赋给 Out5,以此类推,Second 的 8 个数据位将依次赋给 Out4、Out5、Out6、Out7、Out8、Out9、Out10、Out11。

2: 一系列 bool 型变量转换成 byte 型变量

指令格式: BoolToByte (byte 型变量, bool 型变量);这条语句实际上是上面的语句的逆变换。

例如: byte my_byte;

BoolToByte(my_byte, In0);

执行完上面的语句后,第一个 bool 型变量 In0 将赋给 my_byte 的第一个数据位,第二个 bool 型变量 In1 将赋给 my_byte 的第二个数据位,以此类推,从 In0 开始的 8 个开关量序列 In0、In1、In2、In3、In4、In5、In6、In7 将依次赋给 my_byte 的第一位、第二位。。。直至第八位。

注意:对于进行转换的左边参数必须是 byte 型变量,右边参数必须是 bool 型变量,bool 型变量仅限于 In0 至 In249 以及 Out0 至 Out249。

数学运算

恒日控制器支持加减乘除数学运算，恒日 C 窗口的运算语句中的 “=” 号是一个赋值符号，它将等号右边的结果赋予等号左边的变量。

例如：Tim0Set = 125; 该语句将Timer0的预设值更改为 125。

DA0_Output = Tim0Vol*0.01; 该语句将定时器 0 的定时数值乘以 0.01,最后，将结果送给模拟量输出通道 0,当定时器定时数值是 150 时，将在模拟量输出通道 0 得到 1.5V 的电压。利用这个语句可以很容易得到一个斜坡上升电压。

一个计算平均电压的例子：

```
float aaa;  
aaa = AD0_Input+AD1_Input+AD2_Input;  
aaa = aaa/3;
```

首先定义一个 float 类型变量 aaa，然后计算三个模拟量输入的和并且赋值给 aaa，最后将 aaa 除以 3 并且将结果再次赋给 aaa。

如果用带括号的表达式实现上面的运算结果则更加简洁明了：

```
aaa = (AD0_Input+AD1_Input+AD2_Input)/3;
```

由于计算表达式中的括号在运算时需要大量的费时的堆栈运算，所以，不建议过多使用括号运算符。

运算中的数型转换

恒日控制器支持 byte(单字节数) word(双字节数) float(浮点数)。

在运算时，程序会自动按照等号左边的变量类型首先将右边的各个运算因子转变成为相应的类型，再进行运算，最后将结果赋给左边变量。

仍以：DA0_Output = Tim0Vol*0.01;为例，左边变量 DA0_Output 是一个浮点变量，而右边 Tim0Vol 是一个 word 变量，在运算时，操作系统首先会把 Tim0Vol 转换成浮点数，乘上 0.01 之后，结果将赋给 DA0_Output

反过来，表达式：Tim0Vol = DA0_Output；将把浮点型的 DA0_Output 转变成 word 型数据，并且赋给左边的 Tim0Vol

事实上，由于 HenRi C 编程的灵活性，对于大多数程序设计人员来说，则不必考虑各种数型的转换，一切由编译器自动完成。

恒日控制器除了支持加、减、乘、除以外，恒日 C 还支持下列数学运算：

sin() 正弦 cos() 余弦 ln() 自然对数 sqr() 平方根 abs() 绝对值 INT() 取整

例如：计算 5 的各种结果

```
float MyData;  
MyData = sin(5);  
MyData = cos(5);  
MyData = ln(5);  
MyData = sqr(5);  
MyData = abs(5);  
MyData = INT(5);
```

上面数学运算的结果都是浮点数，他们都是独立运行语句，不可以在一个语句中连续出现。

例如：MyData = sin(5)*cos(4); 是错误的。

判断程序

恒日控制器主要采用 if else 判断程序结构

if 判断：

if 判断的一般形式为：if(条件判断表达式) { 程序 }

if 后面小括号里的内容是判断的条件,再后面的大括号里的内容是当条件满足时要执行的程序,当条件不满足时,大括号里的程序将被忽略。

例如:右边的几行程序首先是对定时器Tim0的定时数值进行判断,当其大于34时即执行后面的花括号内的程序语句,如果条件不成立,即Tim0Vol>34这个条件判断布尔结果为假,程序就会跳过花括号内的程序行。注:if的中文意思是:如果、假如。通俗一点说,这种判断语句就是:如果(小括号内的条件成立)那么就执行{花括号内的语句},否则,就跳过。

```
if(Tim0Vol > 34)
{
    Out0 = on;
}
```

在上面介绍的 if 判断程序后面加上 else{} 就成了另外一种结构
if(条件判断表达式)

{程序 1}

else

{程序 2}

这种判断语句就是:如果(小括号内的条件成立)那么就执行{第一个花括号内的语句},否则,就执行{第二个花括号内的程序行},二者必选其一,而且只能选一。

例如:右边的几行程序首先是对定时器Tim0的定时数值进行判断,当其大于34时即执行后面的花括号内的程序语句,然后跳过后面的else花括号内的程序语句,如果条件不成立,即Tim0Vol>34这个条件判断布尔结果为假,程序就会跳过第一个花括号内的程序行执行else后面的花括号内的程序行。注:else的中文意思是:别的,否则。通俗一点说,

```
if(Tim0Vol > 34)
{
    Out0 = on;
}
else
{
    Out0 = off;
}
```

判断程序中德条件设置

对控制参数的判断是编程过程常见的问题，恒日 C 主要采用 if 判断结构，例如：`if(AD0_Input > 2){Out0 = on;}`

if 后面的小括号的内容是判断的条件，大括号内的内容就是当条件成立时要执行的程序，当条件不成立时，就跳过大括号内的程序。

判断的条件必须是一个 bool 结果，他有如下几种形式：

1：单个 bool 变量，例如：`if(In0){Out0=on;}`，结果是当开关量输入点 In0 为真即接通时，开关量输出点 Out0 接通。

2：比较判断，例如：`if(Cnt0Vol >=100){Out0 = on;}`结果是当计数器 0 的计数数值超过 100 时，Out0 接通，这种判断有以下几种比较运算符

- A：“>” 大于判断
- B：“<” 小于
- C：“==” 等于
- D：“>=” 大于等于
- E：“<=” 小于等于
- C：“!=” 不等于

3：复合判断，例如：`if(AD0_Input >3 && In0){Out0 = on;}`结果是如果模拟量输入通道 0 电压高于 3V，而且开关量输入 In0 接通，那么 Out0 接通。

复合判断有逻辑与“&&”和逻辑或“||”两种运算符,&&可以读作“而且”

例如：`if(AD0_Input > 1.5 && Cnt0Vol <120 && Tim0Vol >45){Out0 = on;}`

上面程序可以读作：如果模拟量通道 0 大于 1.5V 而且计数器 0 计数数值小于 120 而且定时器 0 的定时数值超过 4.5 秒，那么，Out0 接通。可以看出，逻辑与判断是小括号里的所有条件都满足才能执行后面大括号的程序。

如果 && 换成 || 就变成逻辑或判断

例如：`if(AD0_Input > 1.5 || Cnt0Vol <120 || Tim0Vol >45){Out0 = on;}`

上面程序可以读作：如果模拟量通道 0 大于 1.5V 或者计数器 0 计数数值小于 120 或者而且定时器 0 的定时数值超过 4.5 秒，那么，Out0 接通，可以看出，逻辑或判断是只要其中一个条件满足，就执行大括号里的程序。

另外一个重要符号是取反符号“!”，它是将判断条件中的 bool 值取反，然后，由程序进行判断：例如 `if(!In0){Out0 = on;}`意思是当 In0 断开时，Out0 接通。取反符号也可以用在复合判断中，例如：`if(Tim0 && !Out100){Out0 = on;}`意思是当定时器 0 的触点接通而且 Out100 是断开时，Out0 就接通。需要注意的是与其他 C 编译器不同，在恒日 C 的判断条件中不支持括号，

例如：`if((Cnt0Vol>8)&&(Tim0Vol<7)){ }`在判断条件中又出现了括号，这是不允许的。

断电记忆

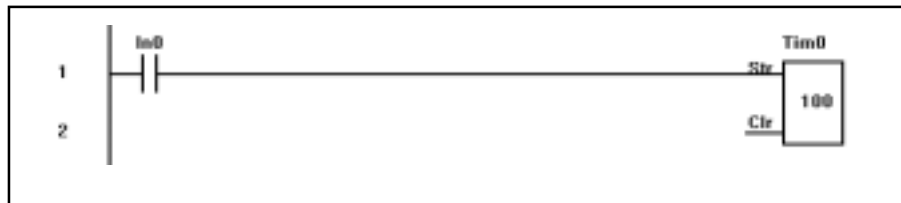
恒日控制器共有三种参数具有断电记忆特性

- 1: 记忆继电器 这类继电器断电时会保持其原来状态, 直至再次上电
- 2: 计数器的计数数值 包括高速计数器在内的所有计数器的计数数值都是断电保持
- 3: 自己定义的各种参数

这些数据采用电池支持, 在控制连续断电的情况下可以保持十年。

另外一种参数是不需电池永久保持, 它们是在梯形图中定义的计数器和定时器的预设值, 在控制器上电初始时, 控制器的定时器和计数器总是首先采用梯形图中的预设值, 但是, 用户可以在 C 窗口对其进行动态修改。

例如:



定时器Tim0的预设值是100即10秒, 控制器在上电接通时总是首先以梯形图中的100作为定时器0的预设指。但是当在 C 程序的初始化程序中加上语句 `Tim0Set = 65;` 后定时器0的预设值将被65取代。

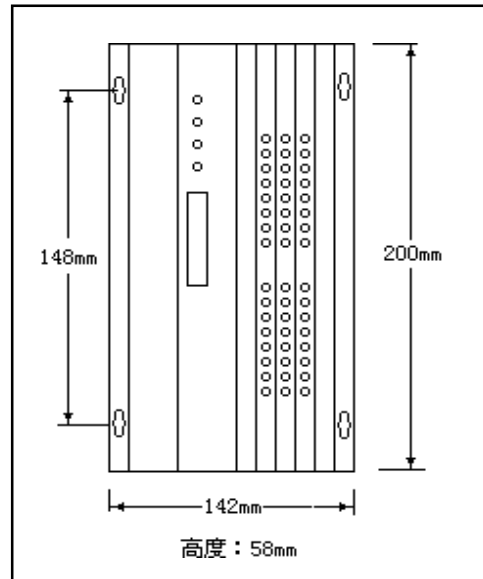
```
main
{
//Add initialization code here
Tim0Set = 65;
MainLoop:
    WatchDog();
    LadderProgram();
//Add normal program code here
    goto MainLoop;
}
```

用户可以在程序的运行中动态修改定时器预设值等所有变量, 例如当 In1 接通时将定时器0的预设值设为20, In2 接通时将其设为30, In3 接通时将其设为40, 程序如下:

```
main
{
//Add initialization code here
MainLoop:
    WatchDog();
    LadderProgram();
//Add normal program code here
    if(In1){Tim0Set = 20;}
    if(In2){Tim0Set = 30;}
    if(In3){Tim0Set = 40;}
    goto MainLoop;
}
```

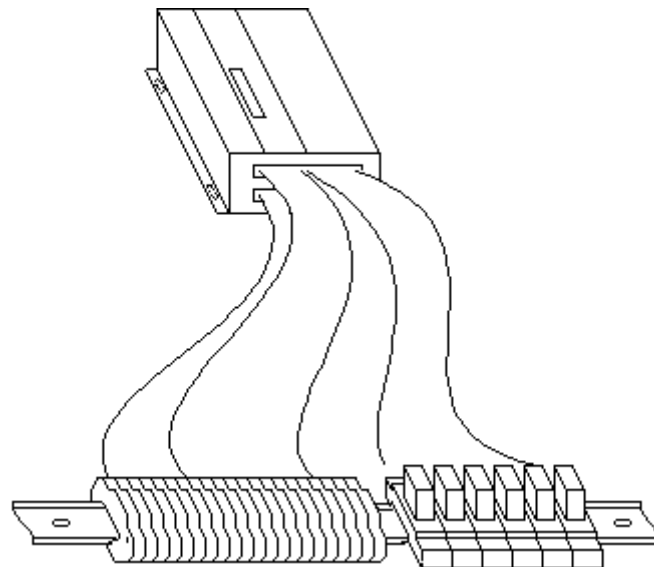
尺寸安装：

1. 控制器的安装和接线必须是懂得电气安全规范的专业技术人员。
2. 所有的输入信号应当尽可能地短，如果太长，应当采用屏蔽线缆。
3. 所有信号线应当尽可能远离高压大电流的动力线。
4. 控制器的安装位置应当远离变频器、接触器等大电流、电火花、电弧频繁的设备 and 电器，距离应当大于250mm以上。
5. 所有的NPN输出驱动继电器、电磁阀等电感性负载时必须加装保护二极管，或者直接采用控制器的尖峰泄放接线端子。
6. 技术人员应当仔细阅读本使用手册，严格按照手册规定的输入输出参数选用输入输出设备。



安装尺寸

电气控制箱安装示意图：



(DIN 导轨接线端子和继电器)