K-XPCIII-T可编程网络控制系统
用户手册

V1.2 版

广州赢溢电子有限公司

符号的意义

■ 安全指示

使用说明书和设备上都使用了符号，指出可能对用户或他人造成的伤害以及财产受损的风险，以便您能够安全、正确的使用设备。指示及其含义如下。请确保在阅读说明书之前正确理解这些指示。

	此为 A 级产品，在生活环境中，该产品可能会造成无线电干扰。在这种情况下，可能需要用户对干扰采取切实可行的措施。
	提醒使用者设备内出现未绝缘的危险电压可能会导致人遭受电击。
	CE 认证表示此产品已经达到了欧盟指令规定的安全要求，用户可放心使用。
	SGS 认证表示此产品已经达到了全球最大的瑞士通用公证的质检标准。
	本产品通过 ISO9001 国际质量认证（认证机构：德国莱茵TUV）。
	警告：为了避免电击，请不要打开机盖，也不要将无用的部分放在机箱内。请与有资格的服务人员联系。

■ **一般信息指示**

	列示了可能导致操作或设置不成功的内容及一些需要注意的相关信息。

重要说明


警告

为确保设备可靠使用及人员的安全，请在安装、使用和维护时遵守以下事项。

安装时的注意事项

◆ 请勿在下列场所使用本产品：有灰尘、油烟、导电性尘埃、腐蚀性气体、可燃性气体的场所；暴露于高温、结露、风雨的场合；有振动、冲击的场合。电击、火灾、误操作；

◆ 在进行螺丝孔加工和接线时，不要使用金属屑和电线头掉入控制器的通风孔内，这有可能引起火灾、故障、误操作：

◆ 产品在安装工作结束，需要保证通风面上没有异物，包括防尘纸等装物品，否则可能导致运行时散热不畅，引起火灾、故障、误操作；

◆ 避免带电状态进行接线、插拔电缆插头，否则容易导致点电击，或导致电路损坏；

◆ 安装和接线必须牢固可靠，接触不良可能导致误操作；

◆ 对于在干扰严重的应用场合，高频信号的输入或输出电缆应选用屏蔽电缆，以提高系统的抗干扰性能。

布线时的注意事项

◆ 必须将外部电源全部切断后，才能进行安装、接线等操作，否则可能引起触电或设备损坏；

◆ 本产品通过电源线的接地导线接地，为避免电击，必须将接地导线与大地相连，在对本产品的输入端或输出端进行连接之前，请务必将本产品正确接地；

◆ 在安装布线完毕，立即清除异物，通电前请盖好产品的端子盖板，避免引起触电；

运行和保养时的注意事项

◆ 请勿在通电时触摸端子，否则可能引起电击、误操作；

◆ 请在关闭电源后进行清扫和端子的旋紧工作，通电时这些操作可能引起触电；

◆ 请在关闭电源后进行通讯信号电缆的连接或拆除、扩展模块或控制单元的电缆连接或拆除等操作，否则可能引起设备损坏、误操作；

◆ 请勿拆卸设备，避免损坏内部电气元件；

◆ 务必熟读本手册，充分确认安全后，再进行程序的变更、试运行、启动和停止操作；

产品报废时的注意事项

◆ 电解电容的爆炸：电路板上的电解电容器焚烧时可能发生爆炸；

◆ 请分类收集和处理，不能投入生活垃圾中；

◆ 请按工业废弃物进行处理，或者按当地的环境保护规定处理。

前言

《K-XPCIII可编程网络控制系统用户手册》主要介绍了 K-XPCIII控制器的操作方法，主要性能参数以及常见故障解决办法。

本手册只作为用户操作指示，不作为维修服务用途。自发行日期起，此后之功能或相关参数若有改变，将另作补充说明，详情可向厂商或各经销商查询。

目录

目录

第一章、综述

K-XPCIII是网络通讯型控制主机，采用主频高达 667MHz 的 32 位内嵌式处理器，ARM11 CPU，256M 内存，1G Flash 闪存。

K-XPCIII可编程网络控制主机提供了三类网络（AV-NET、AV-LINK、Ethernet）及多种控制端口，含 IR（红外）、I/O（数字输入/输出）、RELAY（弱电继电器）、COM 口等。

采用先进的集成技术，提供高速准确的集中控制环境、开放式的用户编程界面，可完成各种复杂的控制接口编程。

1.1 功能特点

◆ ARM11 CPU，256M DDR 内存，1G Flash 闪存；

◆ 采用 667MHz 主频的 32 位内嵌式处理器；

◆ 8 路独立可编程的红外发射接口，支持控制多台相同或不同的红外设备；

◆ 8 路独立可编程 RS-232/422/485 控制接口，用户可编程设置多种控制协议和代码；

◆ 8 路弱电继电器接口；

◆ 8 路数字输入/输出 IO 接口；

◆ 三种网络通讯：AV-NET、AV-Link、Ethernet；

◆ USB2.0 编程通讯接口；

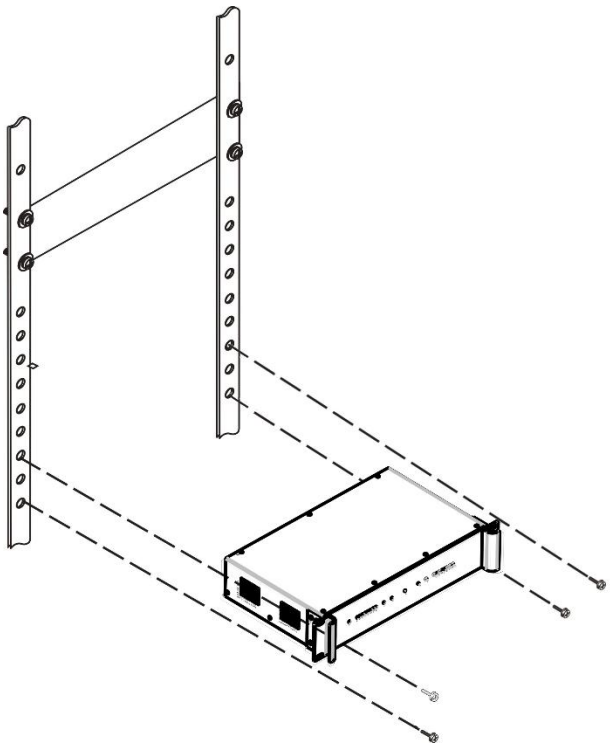
◆ 内嵌式红外学习器，方便调式和维护；

◆ 支持本地及远程多种控制方式；

◆ 国际通用宽适配电源设计（AC100~240V），适用任何国家和地区。

1.2 主机安装

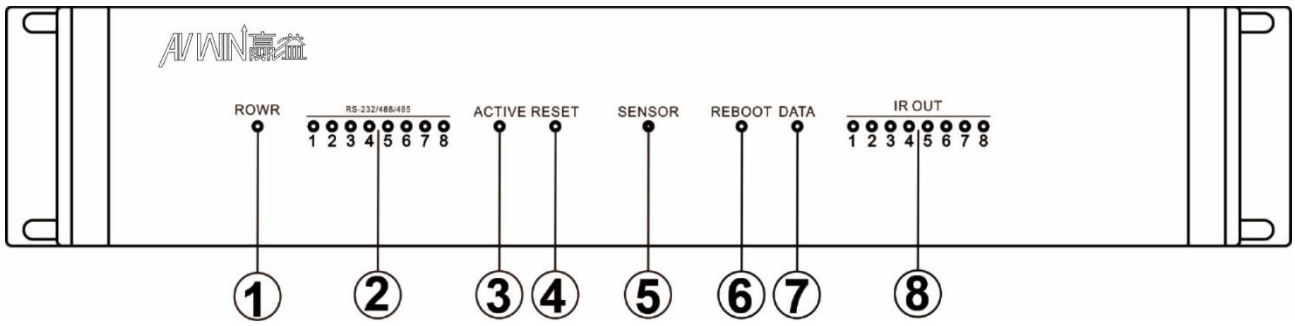
K-XPCIII可编程网络控制主机可以安装在标准 19 英寸机柜上，主机标配附件含一对机柜安装支架，装配方式见下图。



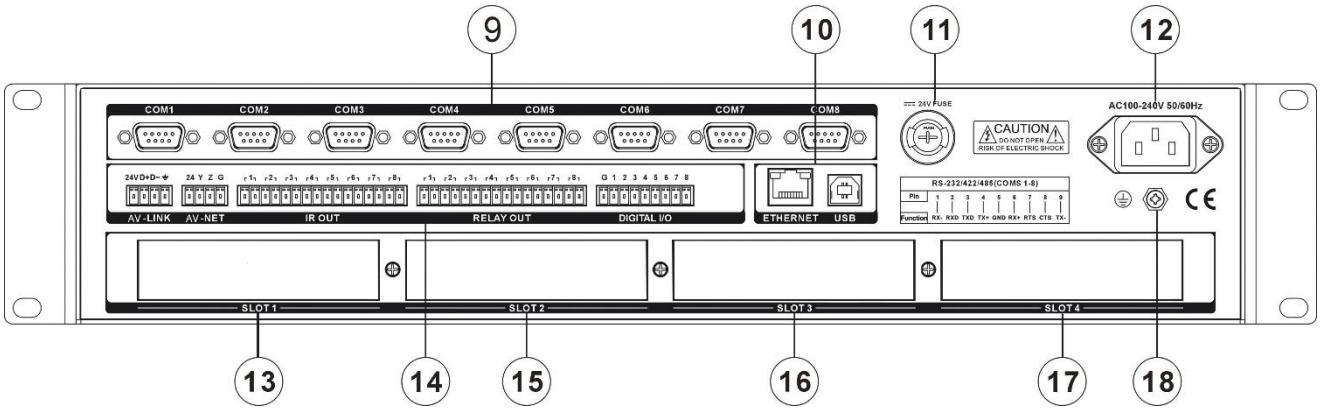
第二章、系统主机说明

2.1 面板功能说明

K-XPCIII前面板：



K-XPCIII后面板：



1）POWER——主机电源指示灯

2） RS-232/422/485——COM 数据收发指示灯支持 8 路 COM 接口信号指示，当 COM 口发送或接收数据时，相应的指示灯会闪亮， TX 是信号发送指示灯，RX 是信号接收指示灯。

3） ACTIVE——状态灯

4） RESET——复位按键

当主机被下载了非法的用户程序（如死循环）导致主机状态异常，可通过 RESET 清空用户程序。

具体操作方法：断开主机电源按住 RESET 键，给主机上电（不要松开 RESET 键），此时主机会连续发出“滴、滴”声响，约 7~8 次“滴”声后松开 RESET 键，此时主机内非法用户程序被清空，主机恢复正常。

5） SENSOR——红外接收窗口

K-XPCIII可编程网络控制主机提供了内置红外学习器， SENSOR 用于完成红外学习中的红外信号接收功能。

6） REBOOT——重启按键

当系统在操作不当时出现死机状态可按此键，重启系统。

7） DATA——数据信号指示灯

如数据信号正常传输，此指示灯亮起，反之不亮。

8） IR OUT——红外信号指示灯

支持 8 路红外信号指示，当主机对外部受控设备发送红外控制信号时，对应的指示灯闪亮。

9） COM 接口

包含 8 路可编程双向串行通讯接口 DB9（公）接口类型，支持 RS-232/422/485 通讯格式。

10） ETHERNET——以太网接口

用于连接外部网络实现与无线（WiFi）触摸屏 的通讯或以太网远程控制。

USB——USB2.0 通讯接口

与电脑的 USB 口连接完成各项操作，如用户程序下载、系统诊断、红外学习等。

11 ）24V 保 险

12 ）AC 100V~240V——系统电源

系统电源输入，开关电源 AC100V~240V 50/60Hz 自适应。

13） IR OUT——扩展红外控制接口

插入扩展卡可以完成扩展红外控制接口。

14） AV-LINK——高速总线接口、扩展用。

AV-NET——AV-NET 总线

内部通讯接口（类 RS-485 协议）4 芯凤凰接口类型，可连接各类外部网络设备，如电源控制器，调光器，音量控制器、无线接收器，有线触摸屏等。

IR OUT——红外控制接口

包含 8 路独立可编程红外控制接口（38KHE）载波，以控制多种相同或不同设备，如 DVD VCR、MD 等的播放、暂停、停止、进出仓等。

2 芯凤凰接口类型，需配合红外发射棒使用。将红外发射棒连接到红外控制接口，发射端放在受控设备红外接收窗口前使用（距离不大于 15CM）。

RELAY OUT——弱电继电器接口

提供 8 路继电器的常开接口，可驱动 AC 0.3A/125V 或者 DC 0.3A/110V、 DC 1A/30V 以下的负载，可以控制符合以上负载类型的各类相关电器设备的开关，实现以小电流低电压驱动大电流高电压的负载。

DIGITAL I/O——I/O（输入输出）接口

提供 8 路可由软件编程的 I/O 输入输出控制接口，可提供 5V/10mA 负载输出或接收 0~5V（10mA 负载电流）的信号输入。

15） RELAY OUT——扩展弱电继电器接口

插入扩展卡可以完成扩展弱电继电器接口。

16） DIGITAL I/O——扩展 I/O（输入输出）接口

插入扩展卡可以完成扩展 I/O 输入输出控制接口。

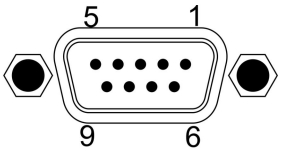
17） UART——扩展 COM 接口

插入扩展卡可以完成扩展可编程双向串行通讯接口。

18） 接地柱。

2.1. 接口说明

2.1.1. COM 口脚位功能说明

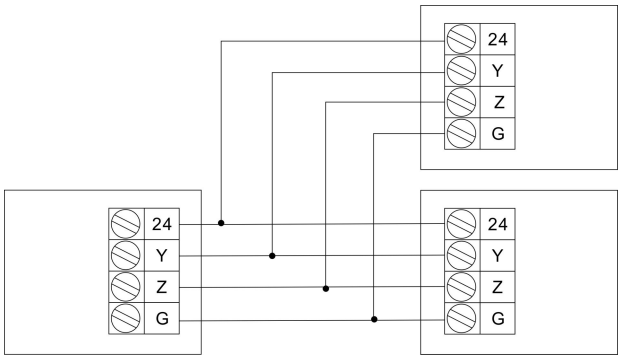


脚位	信号	说明
1	RXD	RS-485 协议用，和 9 脚接在一起作为 RS-485“-”
2	RXD	RS-232 协议用，接收数据
3	TXD	RS-232 协议用，发送数据
4	TXD+	RS-485 协议用，和 6 脚接在一起作为 RS-485“+”
5	GND	信号地
6	RXD+	RS-485 协议用，和 4 脚接在一起作为 RS-485“+”
7	RTS	RS-232 协议用，请求发送
8	CTS	RS-232 协议用，清除发送

9	TXD	RS-485 协议用，和 1 脚接在一起作为 RS-485“-”

2.1.2. AV-NET 连接

外部设备（AV-NET）的连接支持串、并联多类灵活的连接方式，连接时请注意 24、Y、Z、G 一一对应的连接关系，参考的连接见下图：



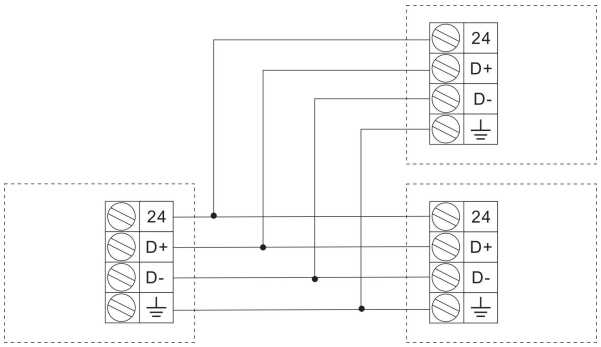
2.2.4 USB 接口连接说明

USB 接口用于编程调试时电脑与主机通讯用，连接示意图如下：

安装及使用中应尽量避免带电插拔网络设备， 以防因电压冲击造成网络设备与主机之间的通讯故障。

2.1.3. AV-LINK 连接

外部设备（AV-LINK）的连接支持串、并联多类灵活的连接方式，连接时请注意 24、D+、D-、⏚一一对应的连接关系，参考的连接见下图：

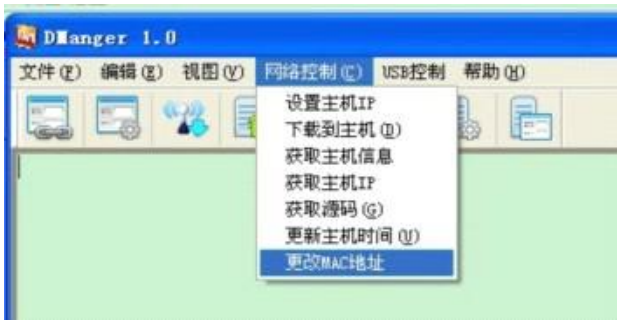


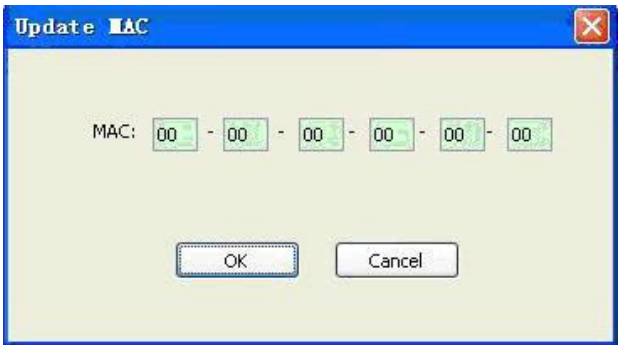
安装及使用中应尽量避免带电插拔网络设备， 以防因电压冲击造成网络设备与主机之间的通讯故障。

2.1.4. 修改主机 MAC 地址

MAC 不能和任何网络设备的 MAC 地址冲突（例如电脑，中控主机等），所以如果主机的 MAC地址和其他网络设备冲突时候就要修改 MAC 地址才可以通信，如果不冲突，可以不修改。

修改主机 MAC 地址方法：保证 PC 和主机正常通信下，根据如下的方框填写 MAC 地址，然后填写完毕后点击 OK，重启主机就行。

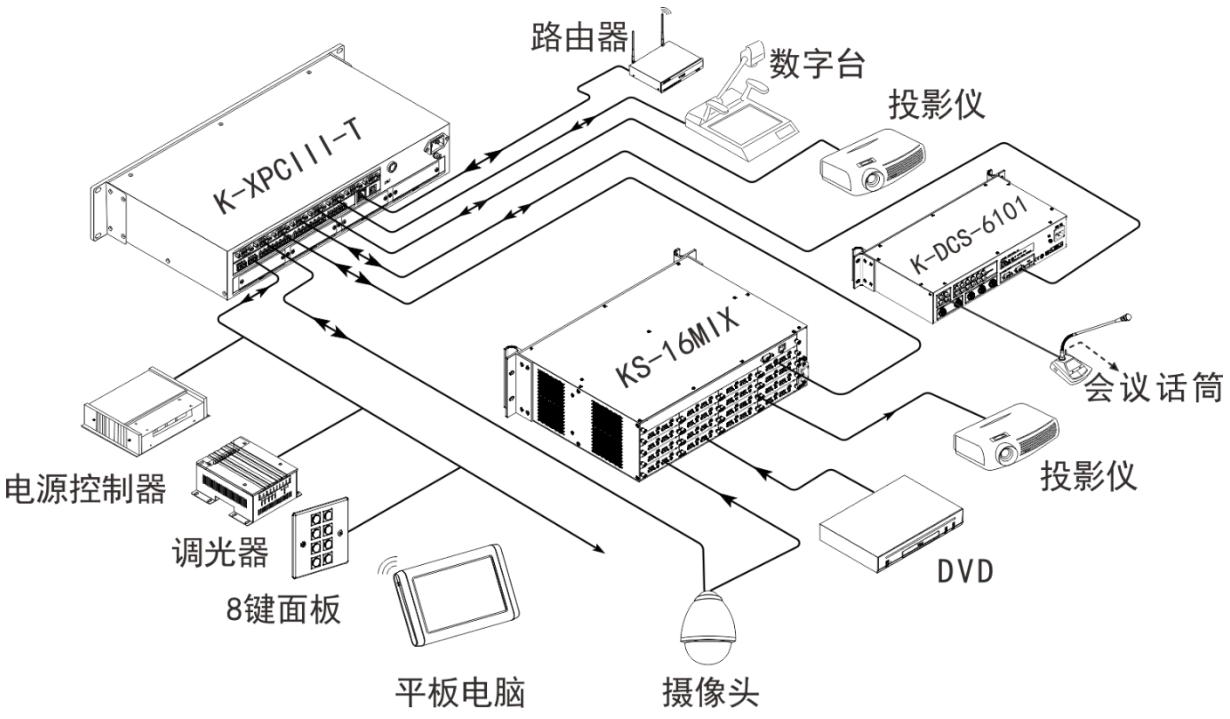




假设我们修改主机MAC 地 址 为 ：00:11:22:33:44:55，就在上面的六个方格里面填写进去对应的数字（注意主机 MAC 地址不能与电脑等其他网络设备 MAC 地址冲突）

第三章、连接示意图

3.1. 系统连接示意图



第四章、红外发射棒使用说明

5.1 功能特点

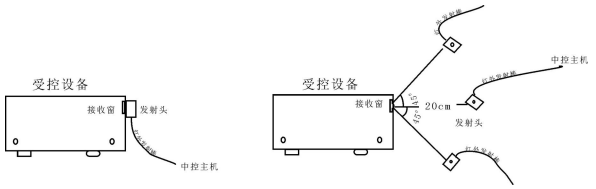
红外发射棒主要用于配合控制主机控制红外受控设备（如 DVD、VCR 等），它是一个装在黑色小型塑料外壳中的红外发射管，有正、负极之分， 当安装需要延长红外控制长度时，应按二极管“正向导通、反向截止”的特性判定正确的接线脚位。

控制系统用户可通过两种途径获得所需红外优码文件：

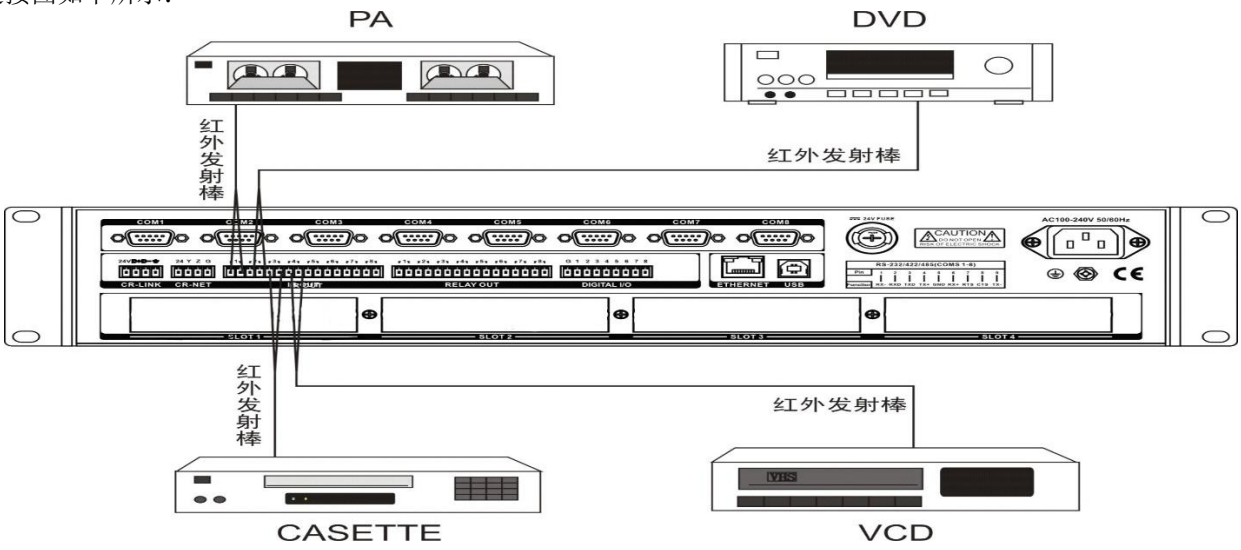
1、 从 Control System Builder 软件 “User IR Module”中搜寻（其中收录了目前市面上绝大多数各种品牌、各类型号设备红外代码文件）；
2、 利用控制主机内置红外学习器及 IR Manger 红外代码管理软件自行生成。

4.1. 连接说明

将红外发射棒一端连接至中控主机的红外发射端口，另一端固定至受控设备的红外接收窗口， 或与受控设备的红外接收窗口的垂线小于45°,距离红外接收窗口小于 20cm 的空间范围内。



连接图如下所示：



第五章、扩展卡说明

1.1. 模拟卡说明

5.1. 应用特性

- 输入：3 路高阻抗直流输入。
- 输出：3 路直流输出。
- 数/模、模/数转换：具备 A/D 和 D/A 转换器功能，可用于采样和控制；10 位 AD 采样精度，12 位输出精度。
- 输入电压范围：0V 到+12V。
- 输出电压范围：-12V 到+12V。
- 输出电压可通过软件调节。
- 输入最大可承受电压+36 伏直流电压。
- 每路输出最大电流为 5mA。

5.1.1. 外部参与输入

- 输入最大采样电压值为+12 伏直流电压。
- 过电压+36 伏直流电压。

5.1.2. 操作方法

- 输出电压控制方法（D/A 转换）

K-XPCIII主机向模拟卡发送第几路多少伏电压输出，模拟卡接收指令后做出电压输出。

- 读取输入电压值（A/D 转换）。

XPCIII主机向模拟卡发送读取模拟卡第几路电压值，模拟卡收到指令后将实际电压反馈给PGM III主机。

- 以下为使用模拟卡时 PGM3 的编程函数说明。

SEND_QACAR

Void SEND_QACAR (String dev,int channel)

功能：发送请求模拟卡电压值 ，发送请求后，将触发模拟卡的 DataEVENT 事件，在那里可以得到电压值，具体事例看其他函数的 BYTES_TO_INT

Parameters：

dev - ：模拟卡设备

channel -：设备通道号

示例：

```
Acar_m = M:8:ACAR:192.168.1.20; //定义主机板号为 8 的模拟卡
SEND_QACAR (Acar_m,1); //读取Acar_m 的第一路的电压值
```

BYTES_TO_INT

Int BYTES_TO_INT (byte[] b)

功能：将 byte 数组的前 4 位当做一个 int 数值处理，如 b 小于 4 位，则按多少位转换，大端模式。

Returns：

返回转换后的 int 型数值

示例：模拟卡电压数据返回，模拟卡实际电压=返回电压/100.00

```
DATA_EVENT(mcar,2)
  {
        ONDATA()
        {
            double tt = BYTES_TO_INT(DATA.Data)/100.0 // 当用 SEND_QACAR 发送请求时，这里触发。

SEND_COM(COM,1,DOUBLE_TO_STRING(tt));
        }
  }
```

SEND_ACAR

void SEND_ACAR(String dev,int channel,int val)

功能：控制模拟卡的电压输出

Parameters:

dev - ：模拟卡设备

channel - ：设备通道号

val - ：模拟量 (注：根据具体外接设备取值。）(一般取值范围为 double 型的 -12（伏） 到 12（伏））

示例：
acar_L = L:7:ACAR:192.168.1.20; // 定义 AVLINK(CAN)号为 7 的 ACAR 设备

SEND_ACAR(acar_L,1,-12);// 向 lilt_L 的第一路发送模拟量-12,即设置模拟卡输出-12 伏

第六章、软件说明

1. Think Control 1.0 编程软件

Think Control 1.0 是针对第三代可编程中控主机(K-XPCIII)开发的编程软件，该软件所实现的功能都是基于 XPCIII中控主机，故不能用该软件编写工程控制 XPC Ⅱ(第二代可编程中控主机)。而可编程就是用 Think Control 1.0 软件编写控制XPCⅢ主机各种电气设备的工程文件，并能根据用户所设定的特定逻辑进行控制。

操作系统

Windows XP 操作系统平台/ VISTA/ WIN7

6.2. Think Control 1.0 安装

Think Control 1.0 软件可以通过配套光盘获得。

安装 Think Control 1.0 步骤如下：双击 Think Control 1.0 安装程序，运行安装向导，如图：

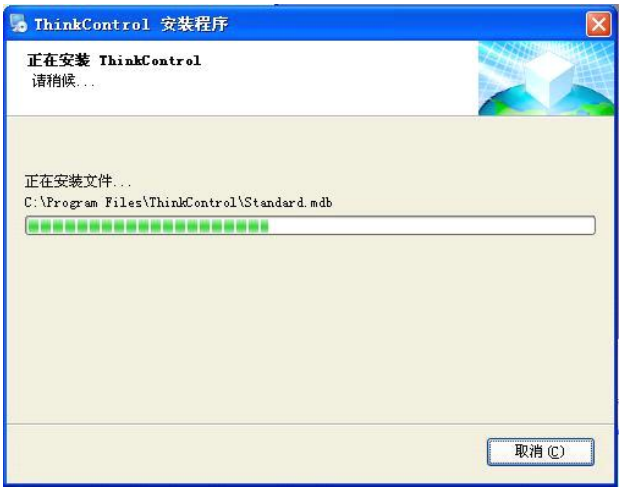
点击下一步

选择自己需要的语种，点击下一步

点击更改可选择安装路径，点击下一步

输入快捷方式名称（通常使用默认名称），点击下一步。

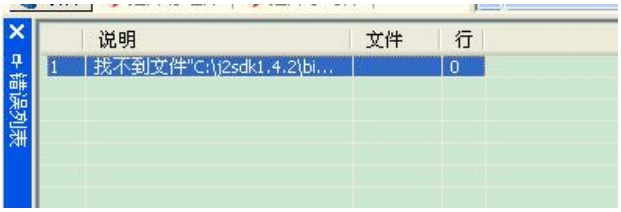
确认设置无误后点击下一步



安装向导提示安装进度，如需此时退出安装，可以按取消键退出。

程序安装成功，按 完成退出。这时桌面上会生成一个 Think Control 1.0 的快捷方式。如图：

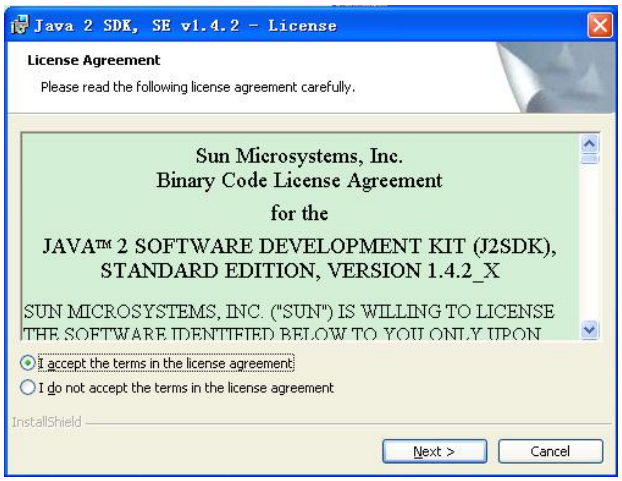
安装完毕后，我们还需要安装 JDK1.4 软件，不然我们编译工程的时候会出现编译错误，如图：



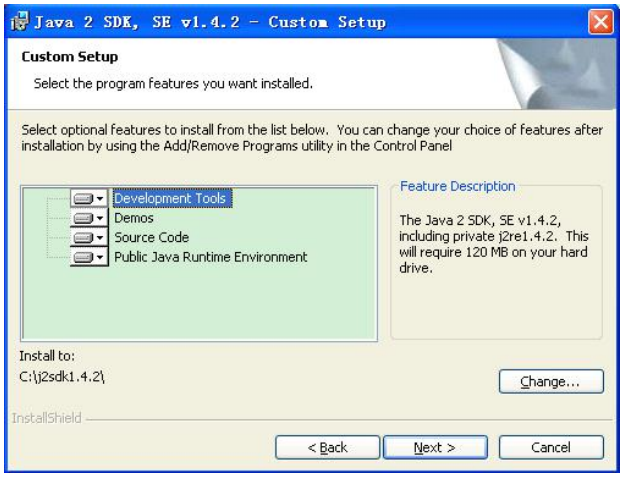
6.3. jdk1.4 安装

jdk1.4 软件可以通过配套光盘获得，也可以通过网上获得。

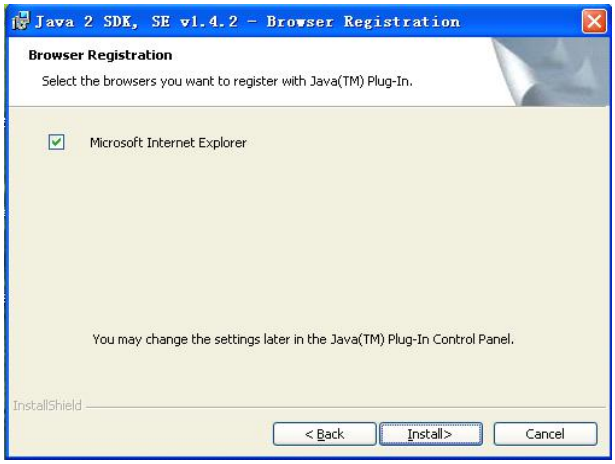
安装 jdk1.4 步骤如下：双击 jdk1.4 安装程序，运行安装向导，如图：



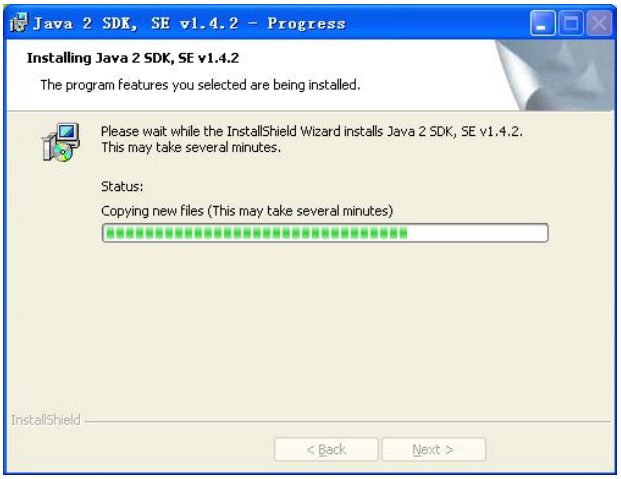
选 择 I accept the terms in the license agreement（我接受许可协议中的条款）后点击 Next

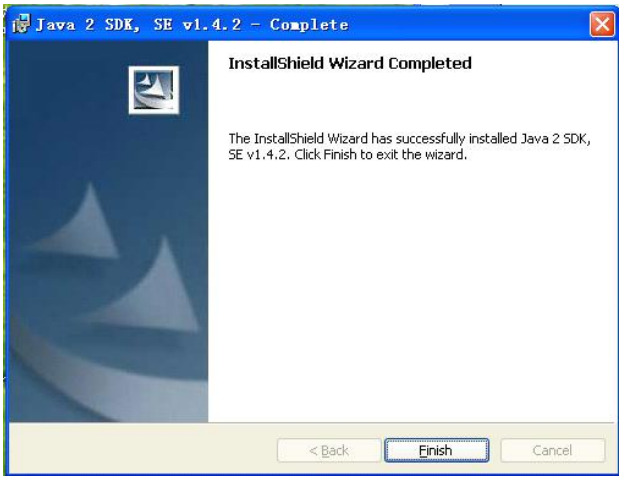


点击 Change…可选择想要安装路径，然后点击 Next



点击 Install

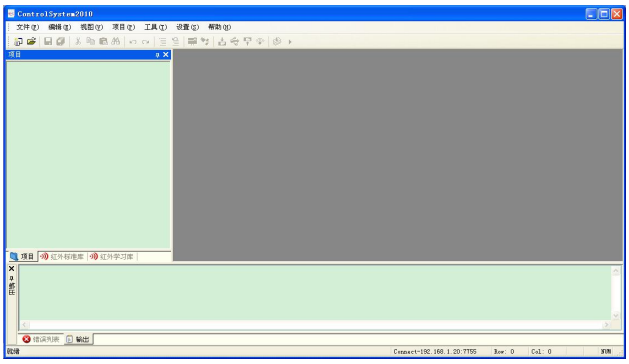




点击 Finish 完成安装。

6.4. 配置正确的 jdk 路径

安装完毕后我们双击桌面 Think Control 1.0 快捷方式进入 Think Control 1.0



选择菜单“设置”“选项”

选择“环境”

点击浏览选择正常的 jdk1.4 软件安装路径。

注：如果安装 jdk1.4 软件时没修改安装路径则不需要修改

6.5. 软件卸载

《Think Control 1.0》软件卸载方法如下：在【开始】/【控制面板】里点击“添加或删除程序”，在其弹出的对话框里找到里《Think Control 1.0》软件标识， 点删除， 系统会自动地删除《Think Control 1.0》的所有文件、程序组和快捷方式。

6.6. 代码组织与控制设备函数

Think Control 1.0 需要通过我们编制工程代码才能实现相应的功能。因此在编写 XPCIII 主机程序前，我们需要先了解一下 Think Control 1.0 代码组织与相应的控制函数（ 用户也可以通过 Think Control 1.0 帮助文档了解）。

6.6.1. 代码组织方式

1、一个程序是有以下若干个块组成的：

每个块都有其特定的功能；如“DEFINE_DEVICE”专门用来定义设备，程序里要用到的设备都必须在这个快里进行定义。不建议改变各块之间的顺序。

```
// 设备定义块
DEFINE_DEVICE

// 常量定义块
DEFINE_CONSTANT

// 变量定义块
DEFINE_VARIABLE

// 函数定义块
DEFINE_FUNCTION

// 程序初始化语句块
DEFINE_START

// 程序循环语句块
DEFINE_PROGRAME

// 事件定义块
DEFINE_EVENT
```

2、下面为大家解析一下各定义块：

DEFINE_DEVICE: 设备定义块

设备定义必须放在此块中。

DEFINE_COMBINE: 设备定义块

该模块只要是针对多触摸屏的情况。如实际中需要多个触屏屏设备，该模块可以很好的匹配。

DEFINE_VARIABLE: 变量定义块

变量定义必须放在此块中。

DEFINE_CONSTANT: 常量定义块

常量定义必须放在此块中。

DEFINE_FUNCTION: 函数定义块

函数定义必须放在此块中。

DEFINE_TIMER: 定时器定义块

定时器定义必须放在此块中。对于一些实际要求，可能需要定义一个时钟，一些动作需要每隔一 段时间执行一次。

DEFINE_START: 程序初始化语句块

当程序开始执行时，将首先执行这里的语句。可以利用程序初始语句块完成程序的一些初始化工作，如初始化变量、执行以下初始操作等。

DEFINE_EVENT: 事件定义块

事件定义必须放在此块中。

主要有以下 3 种事件：

◆ **按钮事件**

语法规则：参数可以是 0 个、1 个 、2 个。有 2 个参数，即制定设备名和 jionNumber 时，事件只对指定设备、指定 jionNumber 有效； 当有 1 个参数，即指定设备时，事件对指定设备有效。 当参数为 0 个时，事件对所有设备有效。事件中可以有响应“按下”、“松开”、“按住”、“整个按键过程”四种动作的函数。事件处理代码必须放到相应的函数当中。

```
BUTTON_EVENT([device] [JionNumber])
{
   PUSH()
   {
      // 当按钮按下去时执行的动作
   }
   RELEASE()

   {
      // 当按钮松手时执行的动作
   }
   HOLD(<TIME>[,TRUE|FALSE])
   {
      // 当按钮按住时过了多长时间/或每隔多长行的动作

   }
   REPEAT()
   {
      // 当按钮被按住时重复做的动作
   }
}
```

◆ **拉条事件**

语法规则：参数可以是 0 个、1 个、2 个。有 2 个参数，即制定设备名和通道号时，事件只对指定设备、指定 JionNumber 有效；当有 1 个参数，即指定设备时，事件对指定设备有效。 当参数为 0 个时，事件对所有设备有效。

```
LEVEL_EVENT([device] [, JionNumber])
{
   // 当按拉条变化时执行的动作
}
```

◆ 数据事件

```
DATA_EVENT([device])
{
   ONLINE()
   {
      // 当收到设备的数据在线指令时执行的动作
   } OFFLINE（）
   {
      // 当收到设备的数据离线指令时执行的动作
   } ONERROR（）
   {

      // 当收到设备的数据错误指令时执行的动作

   }
   ONDATA()

   {
      // 当收到设备的数据时执行的动作
   }
}
```

DEFINE_PROGRAME: 程序循环语句块

程序开始运行后，这里的语句将被不断的循环执行。可以在这里做一些监视工作，如监视某些设备的状态是否正常等。

6.6.2. 控制设备函数

如果你想设备实现各种功能，就是需要控制设备函数来实现的。可想而知了解设备控制函数对我 们来说是何等重要。

接下来我们来了解一下控制设备函数吧。（也可以通过 Think Control 1.0 帮助文档了解）。

6.6.2.2 SEND_IRCODE

Void
SEND_IRCODE(String dev,int channel,String str)
功能：发送红外数据

Parameters:
dev - ：红外设备
channel - ：设备通道号
str - ：红外数据 HEX String

示例：

```
IR_M = M:1000; //定义主机版上的红外设备 IR_M
//向 IR_M 的 1 通道发送红外码
// 其中 IRCODE<"StanderIRDb:3M:CODEC:VCS3000:POLYCOM1:6289:6 (MNO)">为标准库中 3M 公司，
// CODEC 类型，VCS3000 型号，POLYCOM1 受控终端，红外标本号 6289 下的（MNO）红外码，
// 调用该函数，将从标准库中提取匹配的红外码发送出去。
SEND_IRCODE(IR_M,1,IRCODE<"StanderIRDb:3M:CODEC:VCS3000:POLYCOM1:6289:6 (MNO)">);
```

6.6.2.3 ON_RELAY

void **ON_RELAY**(String dev,int channel)
功能：打开继电器

Parameters:
dev - :继电器设备
channel - :设备通道号

示例：

```
RELAY_M = M:1000:RELAY;  //定义主机板号为 1000 的继电器
ON_RELAY(RELAY_M,2);  // 打开主机板号为 1000 上的继电器的第 2 路。
```

6.6.2.1 OFF_RELAY

void **OFF_RELAY**(String dev,int channel)
功能：关闭继电器

Parameters:
dev - ：继电器设备
channel - :设备通道号

示例：

```
RELAY_M = M:1000:RELAY;  //定义主机板号为 1000 的继电器
OFF_RELAY(RELAY_M,2);  // 关闭主机板号为 1000 上的继电器的第 2 路。
```

6.6.2.4. SET_COM

void **SET_COM**(String dev,
int channel,
long sband,
int databit,
int jo,
int stopbit,
int dataStream,
int comType)

功能：设置 COM 口

Parameters:
dev - :设备名称channel
- :设备通道号sband -
：波特率databit - ：数据位 1~8
jo - ：奇偶位 0：无，1：奇数，2：偶数，3：标记，4：空格
stopbit-：停止位 10,15,20,对应 10=1,15=1.5，20=2
dataStream - ：数据流：0：无，1：xon/xoff，2：硬件

comType - ：串口通信方式 232、485、422 不为这三个值默认为 232

示例：

```
Com_m = M:1000:COM;  //定义主机板号为 1000 的串口
//  设置串口 Com_m 第 1 路（即定义主机板号为 1000 的第一个串口）的
// 波特率为 9600，数据位为 8，无奇偶，停止位为 1，数据流无，通信方式为 232
```

SET_COM(Com_m,1,9600,8,0,10,0,232);

6.6.2.5. SEND_COM

void **SEND_COM**(String dev,int channel, String str)

功能：串口数据发送

Parameters:

dev - ：串口设备

channel - :设备通道号

str - ：串口数据,支持两种格式。

1：直接透传字符串数据（将字符串原封不动发往串口），

2：转换 16 进制字符串（遇到 0x 或 0X 打头开始的字符串，将字符串作为 16 进制的字符串进行转换后发送。如发送 0x3132，串口收到为"12"的字符串）

示例：

Com_m = M:1000:COM; //定义主机板号为 1000 的串口

SEND_COM(Com_m,1,"1234"); // 向主机板的 1 路串口发送字符串 "1234"

SEND_COM(Com_m,1,"0x31323334"); // 向主机板的 1 路串口发送字符串"1234"

6.6.2.6. SEND_IO

void **SEND_IO**(String dev,int channel,int val)

功能：控制 IO 口

Parameters:

dev - ：io 设备

channel - :设备通道号

val - ：数值 0 | 1

示例：

Io_m = M:1000:IO; //定义主机板号为 1000 的 IO

SEND_IO(Io_m,1,0); //向 Io_m 的第一路输出低电平

6.6.2.7. READ_IO

int **READ_IO**(String dev,int channel)

功能：控制 IO 口

Parameters:

dev - ：io 设备

channel - :设备通道号

Return：返回 IO 口 channel 路的高低屏状态，为 0 或 1 值，其他为出错。

示例：

Io_m = M:1000:IO; //定义主机板号为 1000 的 IO

int iostate =READ_IO(Io_m,1); //读取 Io_m 的第一路的状态

6.6.2.8. SEND_LITE

void **SEND_LITE**(String dev,int channel,int val)

功能：控制灯光

Parameters:

dev - :灯光设备

channel - :设备通道号

val - ：模拟量 （注：该模拟量取值范围为 0 - 65535）

示例：

lite_n = N:8:LITE; //定义 CRNET 号为 8 上的的灯光设备

SEND_ LITE (lite_n,1,65535); //向 lite_n 的第一路发送模拟量 65535

6.6.2.9. SEND_DMX512

Void **SEND_DMX512**(String dev,int channel,int val)

功能：控制 DMX512

Parameters:

dev - :灯光设备

channel - :设备通道号

val - ：模拟量（注：该模拟量取值范围为 0 - 65535）

示例：

lilt_L = L:7:DMX512; //定义 CRLINK(CAN)号为 7 的 DMX512 设备

SEND_DMX512(lilt_L,1,65535);// 向 lilt_L 的第一路发送模拟量 65535

6.6.2.10. SEND_ACAR

void SEND_ACAR(String dev,int channel,int val)

功能：控制模拟卡的电压输出

Parameters:

dev - :模拟卡设备
channel - :设备通道号
val - ：模拟量 (注：根据具体外接设备取值。）(一般取值范围为 double 型的 -12（伏） 到 12（伏））

示例：

acar_L = L:7:ACAR; // 定义 CRLINK(CAN) 号为 7 的 ACAR 设备
SEND_ACAR(acar_L,1,-12);// 向 lilt_L 的第一路发送模拟量-12,即设置模拟卡输出-12 伏

6.6.2.11. SEND_QACAR

Void SEND_QACAR (String dev,int channel)
功能：发送请求模拟卡电压值 ，发送请求后，将触发模拟卡的 DataEVENT 事件，在那里可以得到电压值，具体事例看其他函数的 BYTES_TO_INT
Parameters:
dev - ：模拟卡设备
channel - :设备通道号

示例：

Acar_m = M:8:ACAR; //定义主机板号为 8 的模拟卡
SEND_QACAR (Acar_m,1); //读取 Acar_m 的第一路的电压值

6.6.2.12. ON_VOL

void **ON_VOL**(String dev,int channel)
功能：打开音量
Parameters:
dev - :音量设备
channel - :设备通道号

示例：

vol_N = N:9:VOL; //定义 CRNET 设备号为 9 的音量设备 vol_N

ON_VOL(vol_N,1); //打开 vol_N 的第一路

6.6.2.13. OFF_VOL

void **OFF_VOL**(String dev,int channel)
功能：关闭音量
Parameters:
dev - :音量设备
channel - :设备通道号

示例：

vol_N = N:9:VOL; //定义 CRNET 设备号为 9 的音量设备 vol_N
ON_VOL(vol_N,1); //关闭 vol_N 的第一路

6.6.2.14. SET_VOLTOTOL

void
SET_VOLTOTOL(String dev,int channel,int val)
功能：控制总音量部分
Parameters:
dev - :音量设备
channel - :设备通道号
val - ：模拟量（注：该模拟量取值范围为 0 - 65535）

示例：

vol_N = N:9:VOL; //定义 CRNET 设备号为 9 的音量设备 vol_N
SET_VOLTOTOL(vol_N,1,600);//设置 vol_N 设备第一路总音量值为 600

6.6.2.15. SET_VOLHIGHT

void
SET_VOLHIGHT(String dev,int channel,int val)
功能：控制高音部分
Parameters:
dev - :音量设备
channel - :设备通道号
val - ：模拟量（注：该模拟量取值范围为 0 - 65535）

示例：

vol_N = N:9:VOL; //定义 CRNET 设备号为 9 的音量设备 vol_N
SET_ VOLHIGHT (vol_N,1,600);//设置 vol_N 设备第一路高音部分音量值为 600

6.6.2.16. SET_VOLLOW

void
SET_VOLLOW(String dev,int channel,int val)

功能：控制低音部分

Parameters:
dev - ：音量设备
channel - :设备通道号
val - ：模拟量（注：该模拟量取值范围为 0 - 65535）

示例：
vol_N = N:9:VOL; //定义 CRNET 设备号为 9 的音量设备 vol_N
SET_VOLLOW (vol_N,1,600);//设置 vol_N 设备第一路低音部分音量值为 600

6.6.2.17. UP_WM

void UP_WM(String dev,int channel)
功能：给墙上面板发送弹起信息 ，用于触摸屏和墙上面板控制同一设备时的消息交互。

Parameters:
dev - :面板设备
channel - :设备通道号

示例：
wm_N = N:14:WM; // 定义 AVNET 设备号为 14 的墙上面板设备
UP_WM(wm_N,1);//要求 wm_N 设备的第一路是弹起状态

6.6.2.18. DOWMP_WM

void DOWN_WM(String dev,int channel)
功能：给墙上面板发送按下信息 ，用于触摸屏和墙上面板控制同一设备时的消息交互。

Parameters:
dev - :面板设备
channel - :设备通道号

示例：
wm_N = N:14:WM; // 定义 CRNET 设备号为 14 的墙上面板设备

DOWN_WM(wm_N,1);//要求 wm_N 设备的第一路是按下状态

6.6.2.19. DEV_REG

void **DEV_REG**(String dev, int channel)
功能：设备登记 ,主要用于 pgm2 代的墙上面板设备的登记

Parameters:
dev - 输入设备
channel - :设备通道号

示例：
wm_N = N:14:WM; // 定义 CRNET 设备号为 14 的墙上面板设备
DEV_REG(wm_N,1);//登记墙上面板 wm_N 的第一路

6.6.2.20. DEV_QUERY

void DEV_QUERY(String dev, int channel)
功能：设备查询 ，功能：设备登记 ,主要用于 pgm2 代的墙上面板设备的查询

Parameters:
dev - 输入设备
channel - :设备通道号

示例：
wm_N = N:14:WM; // 定义 AVNET 设备号为 14 的墙上面板设备
DEV_ QUERY (wm_N,1);//查询墙上面板wm_N 的第一路

6.6.2.21. 其他函数

我们可以通过其他函数来丰富我们控制设备的方法。程序就是通过逻辑指令才能完成复杂的功能。下面我们来了解一下其他函数。（这里只做简单的介绍，我们可以根据需要通过帮助文档来详细了解）

TRACE
功能：打印消息 msg

START_TIMER
功能：启动名为 name 的 Timer 的定时执行器,定时器间隔执行时间为 time 毫秒。与 CANCEL_TIMER(XXX)搭配使用

START_TIMER
功能：在时间 year, mouth, day, hh, minute, second 启动名为 name 的 Timer 的定时执行器 ，定时器间隔执行时间为 time 毫秒。

CANCEL_TIMER
功能：取消名为 name 的 Timer 的定时执行器。与 START_TIMER(XXX,t)搭配使用

WAIT
功能：类似 SLEEP 函数，将 WAIT 语句块里面的操作催迟到一定时间（WAIT 的最小单位为毫秒）才执行。与 SLEEP 不同的是，该语句块不会影响触摸继续操作其他操作。

CANCEL_WAIT
功能：取消名为 name 的 WAIT 语句

SLEEP
功能 ：让程序休眠一段时间。

BYTES_TO_STRING
功能 ：字节转换为字符串

STRING_TO_BYTES
功能 ：字符串转换为动态字节数组

STRING_EQ
功能 ：两个字符串比较，严格按大小写比较

STRING_EQNOCASE
功能 ：两个字符串比较，忽略大小写

STRING_STARTWITH
功能：匹配头部是否相等

STRING_ENDWITH
功能：匹配尾部是否相等

ATOI
功能：字符型转为 int 型

ITOA
功能：int 型数据转换为 String（字符串）

BYTES_ADD
功能：将参数 2 加到参数 1 的尾部，组成一个新的字节数字返回。

GET_BYTES_LENGTH
功能：获取动态字节数组的长度

BYTES_TO_HEX
功能：将动态字节数组转为 16 进制格式的字符串

HEX_TO_BYTES
功能: 将 16 进制字符串转为动态字节数组

GET_YEAR
功能：获取当前系统时间的年

GET_MONTH
功能：获取当前系统时间的月

GET_DATE
功能：获取当前系统时间的日

GET_HOUR_OF_DAY
功能：获取当前系统时间的时

GET_MINUTE
功能：获取当前系统时间的分

GET_SECOND
功能：获取当前系统时间的秒

GET_DAY_OF_WEEK
功能：获取当前系统是一个星期中的第几天

INT_TO_DOUBLE
功能：将 int 型转为 double 型

DOUBLE_TO_INT
功能：将 double 型转为 int 型

STRING_TO_DOUBLE
功能：将 string 型转为 double 型

DOUBLE_TO_STRING
功能：将 double 型转为 string 型

6.7. 编写工程

编写第一个 XPCIII 主机程序。我们以用触摸屏控制主机发送红外码给 DVD 为例，为大家讲解一下如何控制设备。

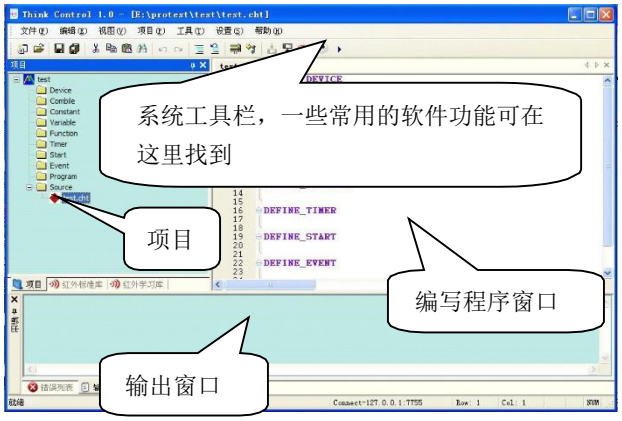
6.7.1. 新建工程

运行 Think Control 1.0。选择菜单“文件”→“新建”或工具栏按  ，新建一个工程，弹出如下对话框：

输入新的项目名如“test”，点解浏览选择项目存放的位置，描述属于工程辅助信息，可填可不填。点击确定。新的工程就创建好了。

我们知道，所有的软件都是基于硬件的，没有硬件，软件毫无意义，硬件是软件的一个平台、一个支撑面，任何软件不能脱离它的相应硬件平台运行。编写XPCIII主机程序所面向的对象是硬件，编程就是如何驱动、 控制、安排这些硬件工作的过程。 所以， 开始在 Think Control 1.0 上编写 PGM III主机程序，为 XPCIII控制软件搭建硬件平台是我们编写主机程序的第一步。

这也是编写主机程序的第一步：搭建控制软件的硬件平台，软件界面如图：



6.7.2. 添加设备

在编写程序前我们需要为工程添加需要的设 备。

选择菜单“项目”→“添加设备”或工具  按钮的添加设备。

我们先来熟悉一下界面

6.7.2.1. 设备名

可以由字母、数字以及下划线字符组成，须以字母或下划线开头。设备名的长度没有限制，但是为用户着想，它不应该过长。

6.7.2.2. 设备类型

设备类型表示是一个大设备，其上有其他设备。如主机板，就是一个大设备，其上有串口设备、继电器、红外口、io 口等。设备类型包括：

[设备元素类型]	描述
M	Main 主机板
T	Touch 触摸屏设备
N	CRNET 设备
L	CRLINK 设备

6.7.2.3. 设备 ID

网络设备都有个 ID 码，是有高、低两位十六制数表示的，分别为 H 高位和 L 低位。配置好的网络设备都有一个唯一的 ID 用来标示其身份，也以此用来区分相同的网络设备，在配置网络设备的时候，其设备的 ID 码要和实际硬件的 ID 相同。但配置过来的网络设备的 ID 是按配置顺序默认设置的， 所以我们还需要在软件上调整网络设备 ID

和实际网络设备硬件 ID 相同。

一定要保证并检查软件上的设备 ID 和实际硬件设备的 ID 一致，不然软件就无法控制，这是比较容易忽略的一个细节。

6.7.2.4. 设备元素类型

表示在载体设备上的小型设备，如串口，灯光，音量等。

[设备元素类型]	描述
RELAY	继电器
COM	COM 口
TP	触摸屏
IR	红外
IO	输入输出口
LITE	调光器
VOL	音量控制器
WM	墙上面板
DMX512	512 跑马灯

例：我们需要控制主机继电器。输入需要添加设备的设备名， 如“ relay_M ”； 选择设备类型“[M]:Main 主机板”；输入设备 ID，如“1000”；选择设备元素类型“[RELAY]:继电器”。

因为我们是通过触摸屏来控制主机，因此我们需要添加主机与触摸屏 2 个设备。

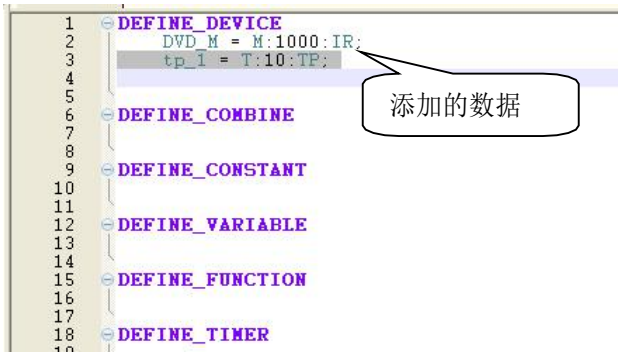
在这里我们添加设备类型为：[M]:Main 主机板，设备名为：DVD_M，设备元素类型为：[IR]:红外，设备 ID 为 1000 的主机设备 (如下图所示)

点击“添加”按钮完成设备添加。

然后我们再来添加设备类型为：[T]:Touch 触摸屏设备，名为：tp_1，设备元素类型为：[IR]:红外，设备 ID 为 10 的触摸屏设备 (如下图所示)

点击“添加”按钮即可完成设备添加。

这时我们会发现在编辑窗口区 DEFINE_DEVICE 下添加了两行数据。

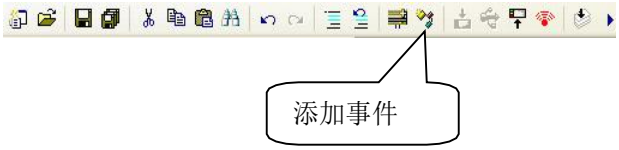


设备定义格式为：

设备名=[载体设备类型]:[载体设备 ID]:[元设备类型];

6.7.3. 添加事件

添加完设备后，接下来我们需要考虑控制设备做什么工作。点击工具栏中的添加事件图标。



我们先来了解一下 3 种事件的功能与作用。

6.7.3.1. 按钮事件

“事件类型”选择“[BUTTON_EVENT]：按钮事件”。

在“ 设备” 下拉列表里选择一个设备， 如“tp_1”，也可以选择“<空>”。

在“通道号”里输入目标按钮的通道号，如不需要通道号则勾选上“无通道号”，触屏设备的通道号即是其 join number 值。

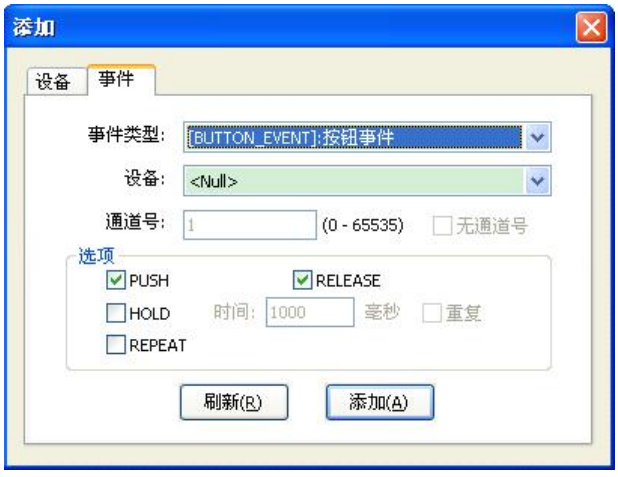
在“选项”里勾选需要的事件函数，

PUSH：表示按下按钮。

RELESE：表示释放按钮。

HOLD：可以设置其时间间隔和重复。

REPEAT：当按钮被按住时重复做的动作。



6.7.3.2. 拉条事件

“事件类型”选择“[LEVEL_EVENT]:拉条事件”。

在“设备”下拉列表里选择一个设备，也可以选择“<空>”。

在“通道号”里输入目标按钮的通道号，如不 需要通道号则勾选上“无通道号”。

6.7.3.3. 数据事件

“事件类型”选择“[DATA_EVENT]:数据事件”。

在“设备”下拉列表里选择一个设备，也可以选择“<空>”。

在“通道号”选择通道号，对于串口设备，这里的通道号为区分为第几个串口。如定义了主板上的串口设备 com_M，

这里选择通道号 1，表示为该主机板上的第一个串口，其他依此类推。

在“选项”里勾选需要的事件函数。

ONLINE：当收到设备的数据在线指令时执行的动作

OFFLINE：当收到设备的数据离线指令时执行的动作

ONERROR：当收到设备的数据错误指令时执行的动作

ONDATA：当收到设备的数据时执行的动作

在这里我们添加按钮事件。（通常情况下我们需要多个按钮事件，按照我们的实际需要添加相应的事件）

这时我们会发现在编辑窗口区 DEFINE_EVENT 下添加 BUTTON_EVENT 函数。

```
DEFINE_EVENT
    BUTTON_EVENT()
    {
        PUSH()
        {
        }
        RELEASE()
        {
        }
    }
```

6.7.4. 红外学习

6.7.4.1. 红外学习

在这之前我们先了解什么是红外设备？通俗的讲，采用红外遥控器控制的设备都属于红外设备，如 DVD、录像机、卡座、电视机、投影机这些通过红外遥控器控制的设备都属于红外设备，在中控行业里，红外设备的控制是非常常见的，无论高端可编程中控还是低端电教产品，都具备控制红外设备的能力。

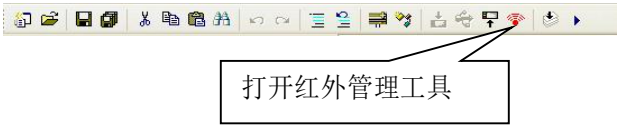
要采用中控系统控制红外设备，首先要采集红外设备的红外代码，采集红外设备的红外代码的过程就叫红外学习，不同的中控系统红外采集的技术也不同。概括的讲，是通过一种叫红外学习器的硬件设备来采集红外代码。XPCIII 内置就有一个红外学习器，XPCIII前面板的 Sensor 口就是红外学习器的红外接收口。至于红外采集的原理技术不必关心，我们要做的工作就是把遥控器对着XPCIII前面板的Sensor 口像平时控制设备一样按遥控器上的功能按键即可把红外遥控器上的功能按键的红外代码采集到红外学习器，再存储到电脑里，生成一个后缀名为 cir 的红外文件。

XPCIII的红外学习是通过 Think Control 1.0 下的一个红外管理器工具 IRL 管理的。下面我们就详细学习 XPCIII的红外学习全过程。

6.7.4.2. IRL 工具

IRL 是 Think Control 1.0 下的一个工具软件，通过 IRL 可以把设备的红外文件录入成一个文件存储在电脑上，然后通过控制程序一起上传到 PGM III主机里。

首先我们先熟悉 IRL 的界面，点击工具栏中的‘调用红外管理器’进入 IRL。如图：

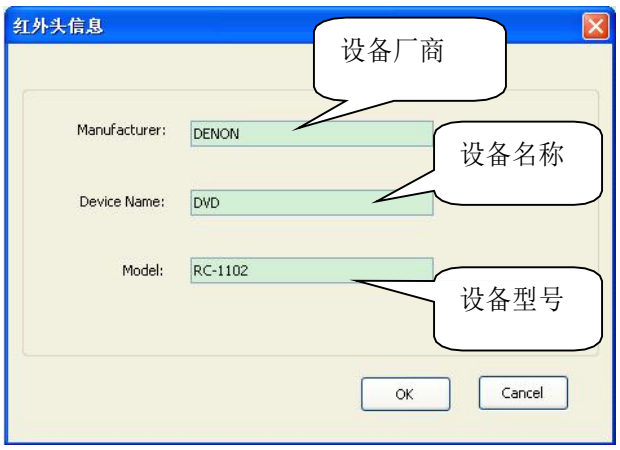


- **设置红外连接 IP 地址**

首先用网线将电脑与 XPCIII连接起来。打开 IRL 后，输入XPCIII主机 IP 地址与端口后按Apply 如图：（主机 IP 地址默认为 192.168.1.20，端口号固定为 100）

- **新建红外文件**

点击工具  图标，或单击【文件】菜单选择【新建】。新建一个文件，在弹出信息对话框，填写设备的厂商、设备种类、型号信息。如图：



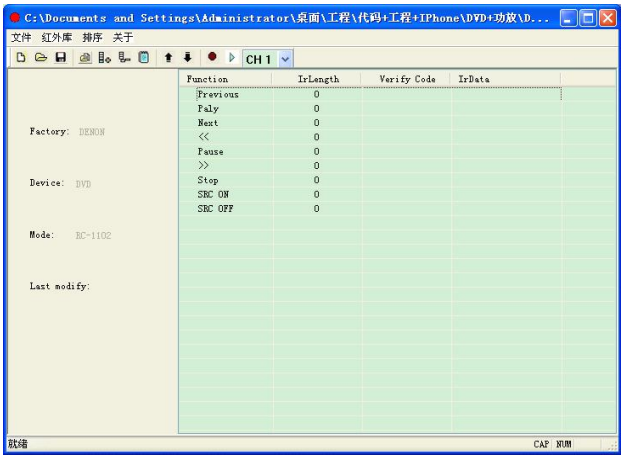
填写红外设备的厂商、名称、型号完毕后点击 OK。一个空白的红外文件建立成了。

◆ **增加功能键**

刚建立的红外文件是空白的。首先我们必须增加功能键，以 DVD 为例，常用的应该有二三十个。

点击工具栏  增加红外码图标，或单击【红外库】菜单选择【增加红外码】弹出一个信息输入框。只需为这个功能键取个名字，为方便辨认和以后维护，建议以有象征性的单词或缩写， 如电源 取 Power，播放为 Play。

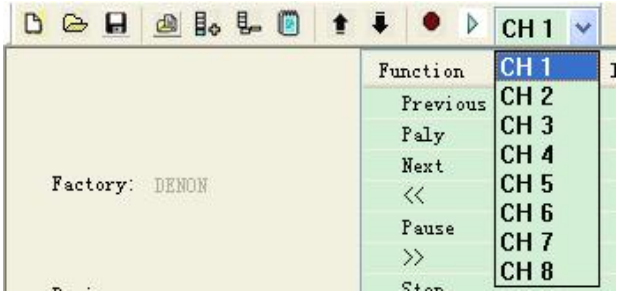
按照以上步骤，添加其他红外码。如图：



◆ **选择红外输出通道**

选择红外输出通道的作用就是：选择红外码从 XPCⅢ主机的哪一个输出口输出。

点击工具栏 CH1 下拉窗口选择需要的通道。一共有 8 个通道选择。如图：



◆ **开始录码**

新建的红外码是没有存储红外代码的，除了有名称其他参数都是空白，必须通过电脑与主机学习 进去。红外学习前保证主机通电和通过传输线电脑 与 XPCⅢ主机正确连接。在准备工作保证无误后，就可进行红外学习了。

在红外学习之前，我们先讨论下什么是长码和短码。XPCⅡ红外学习技术支持学习红外长码和短码，短码比较普遍，一般像 DVD 的常用控制键如电源、播放、暂停等按键都属于短码，而长码比较少用，最典型的是有些带红外控制功能的功放设备中的音量控制，如果用短码来录这种功放的音量控制， 每控制一次调节的幅度很小， 这让用户控制起来非常不方便，所以 XPCⅡ采用了长码技术，采用长码来录的话，由于红外代码长度增长，每次控制音量的时候幅度调节的就相对要大，这让用户控制一下就可以感觉到音量的变化。

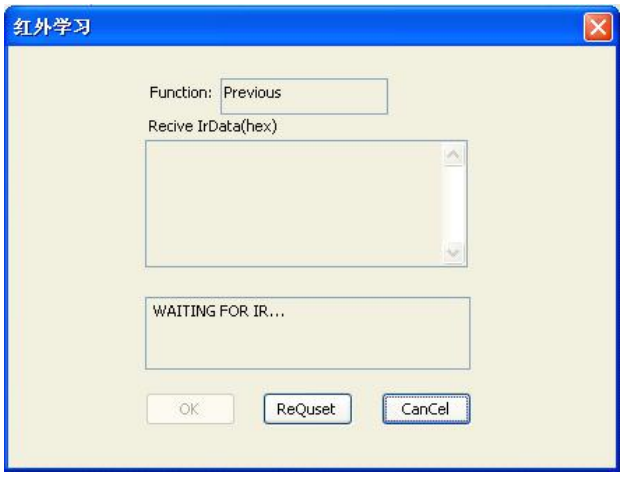
现在我们要学习的 DVD 红外，都属于短码。

我们先大概了解下红外学习的整个流程：在软件中按下红外学习键，这时候软件会提示等待红外代码 输入，XPCIII主机的前面板的红色指示灯也会开始闪烁，在按下红外学习键开始计时 10 秒钟内，拿着遥控器对着 XPCIII主机前面板的红外接收口按下您想学习的功能键，按完键后软件会询问您是否 保存该红外代码，点击是系统会自动把红外代码通 过传输线保存到电脑上的红外文件。 这时软件上如果还有其他没有录码的功能键，会询问您是否进 行下一个红外功能键的学习。点击“OK”又开始重复上述的步骤，学习完红外文件中的所有功能键，系统会提示您按 Cancel 退出。最后保存红外文件到电脑上，会生成一个后缀名为 cir 的红外文件。

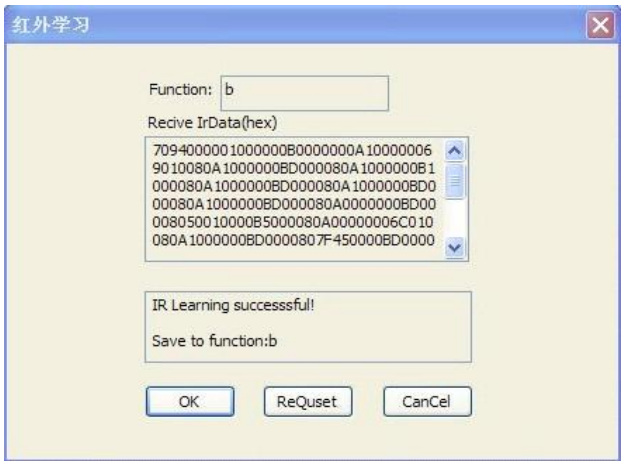
我们再以学习 DVD 红外代码的步骤说明下上述红外学习的过程

A：点击工具栏中的图  打开红外学习，或单击【红外库】菜单选择【红外学习】。

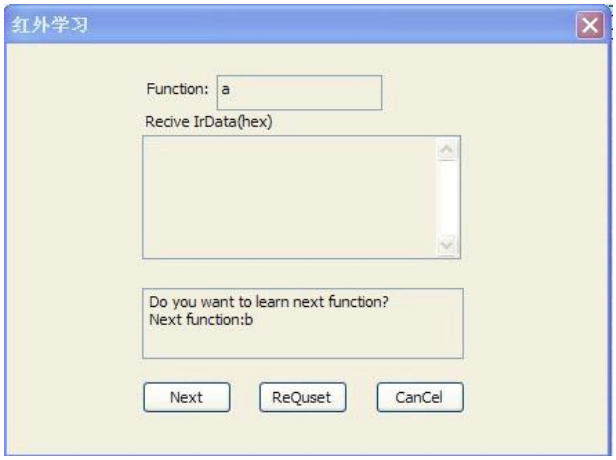
B：点击红外学习键，弹出如下信息对话框：



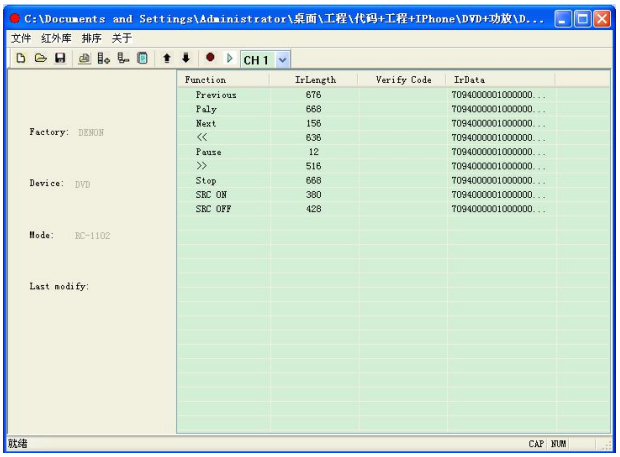
C：看到此对话框，提示等待录入红外，表示可以用遥控器对着主机前面板的红外接收口录入红外了，红外遥控器对着主机前面板的红外接收口按完红外遥控器的按键后，弹出如下对话框：



D：提示红外代码录入成功，我们点 OK。这时系统会提示您是否继续录入下一个功能键：



E:点击 OK 进行下一个功能键的红外学习。Cancel 表示不进行下一个功能键的红外学习。 红外文件中的功能按键全部学习完毕后会提示您按 Cancel 退出。



录码注意事项：

◆ 遥控器要正对着主机的红外接收口，离主机的距离大概保持在 3-5 厘米左右。

◆ 学习的时候按遥控器的按键不宜太久，按键的时间长短和正常操作的一样。

◆ 功能键上的红外代码长度保持一致或相差不大，特别注意 DVD 菜单的上下左右键不宜太长，对着红外接收口的时候按住键的时间要短。保持平常操作的按键方式一般都能成功。

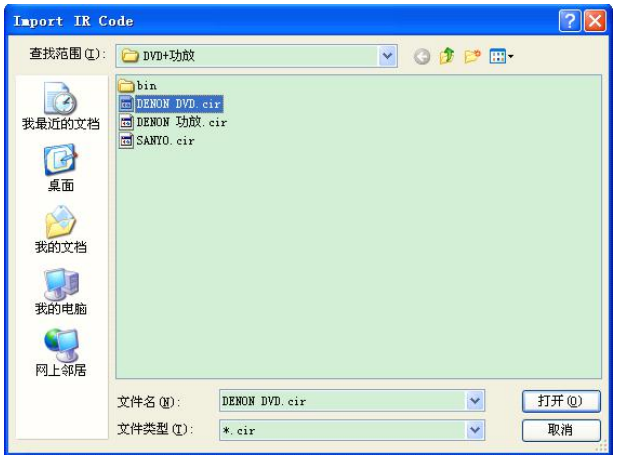
◆ 必须保证每个功能按键都录入红外，多余空的功能键要删除。

录码完毕后保存，系统会生成一个红外文件，后缀名为 cir。

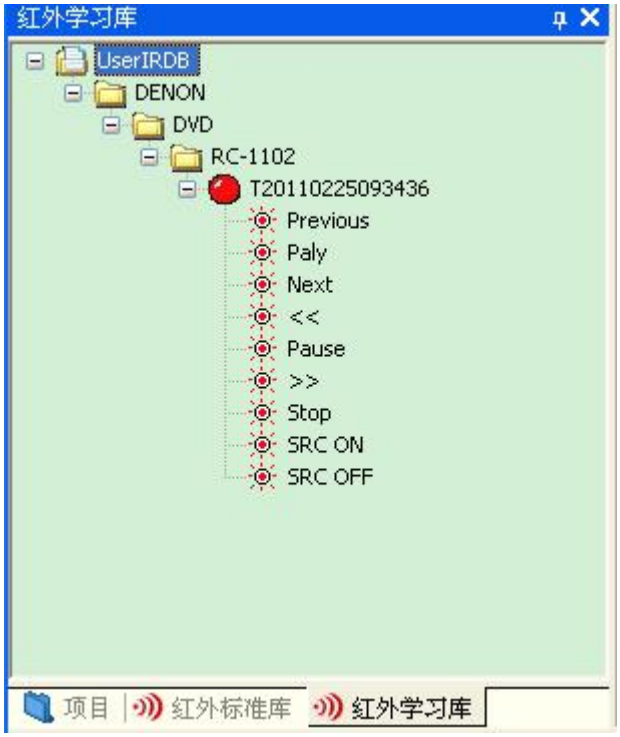
6.7.5. 倒入红外文件

点击 Think Control 1.0 菜单栏【视图】在【项目窗口】中把红外学习库勾上。并选中外学习库窗口。如图：

在红外学习库窗口的空白处点击鼠标右键，在弹出菜单中选中【红外数据导入（IPM）】。弹出如下对话框，选中刚刚录入的红外文件点击打开如图：



导入成功后，打开红外库可以看到我们刚才录入的红外码。如下图所示：



如果发现漏了有代码没录或者代码不正确，需要重新在 IRL 打开该红外文件重新录码，录好以后需要重新倒入到 Control system 的红外学习库。

6.7.6. 编辑程序

完成以上步骤，配置阶段全部完成。这就开始进入编程阶段。

我们希望按下 tp_1 触摸屏的 JOBMONBER 1，就从主机的红外通道 1 发送 Play 红外码。按下

tp_1 触摸屏的 JOBMONBER 2，就从主机的红外通道 2 发送 Pause 红外码。

在这里我们需要用到控制函数 SEND_IRCODE（）来控制主机发送红外码。

根据要求，我们编写了程序：

```
BUTTON_EVENT(tp_1,1)        // tp_1 触摸屏的 JionNumber1
{
    PUSH()
    {
    //从 DVD_M 主机的红外通道 1 发送 Play 红外码
    SEND_IRCODE(DVD_M,1,
IRCODE<"UserIRDb:DENON:DVD:RC-1102

:T20110225093436:Paly");
    }
    RELEASE()
    {
    }
}
BUTTON_EVENT(tp_1,2)        //tp_1 触摸屏的 JionNumber2
{
    PUSH()
    {
    // 从 DVD_M 主机的红外通道 2 发送 Pause 红外码
    SEND_IRCODE(DVD_M,2,
IRCODE<"UserIRDb:DENON:DVD:RC-1102:
T20110225093436:Pause">);
    }
    RELEASE()
    {
    }
}
```

7.7.6.1 如何插入红外码

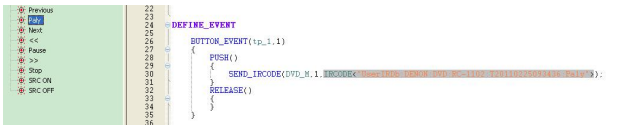
您可以直接从“红外数据库”中选择对应的红外码插入到文件中，而不需要手写。

A：首先把光标移动到需要出入红外码的位置上；

```
DEFINE_EVENT

    BUTTON_EVENT(tp_1,1)
    {
        PUSH()
        {
            SEND_IRCODE(DVD_M,1,)
        }
        RELEASE()
        {
        }
    }
```

B：在“红外数据库”里选择需要的红外码并双击即可完成红外码的插入；

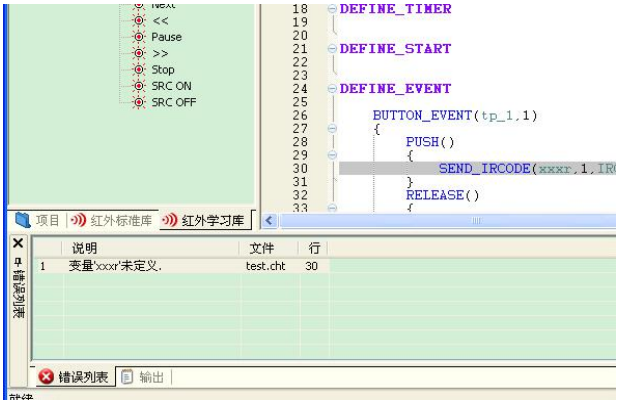


6.7.7. 编译工程

程序编写完后，就准备上传到主机里，在上传之前还需对我们编写的程序进行编译，点击常用工具栏里的编译小图标，如图：

编译时可以从“输出”窗口了解编译的具体信息；

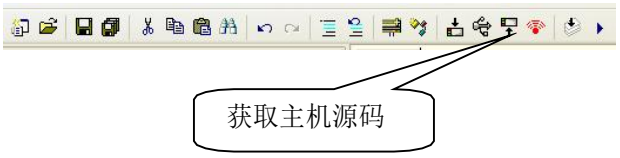
若编译时遇到了错误，则在编译结束后自动跳转到“错误”窗口显示编译所遇到的错误；双击某一条错误信息可以跳转到引发该错误的某行代码出；



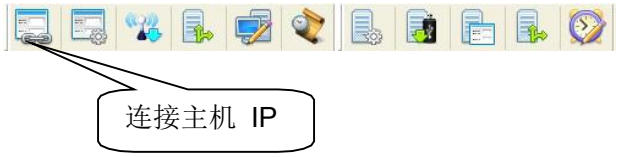
这样我们就编写完成了一个 XPCIII主机程序了。

6.7.8. 使用网络下载主机程序

编译完成后，点击在 Think Control 1.0 的常用工具栏里获取主机源码图标，打开 DManger 软件：



1、点击工具工具栏连接主机 IP 按钮



2、设置连接主机 IP

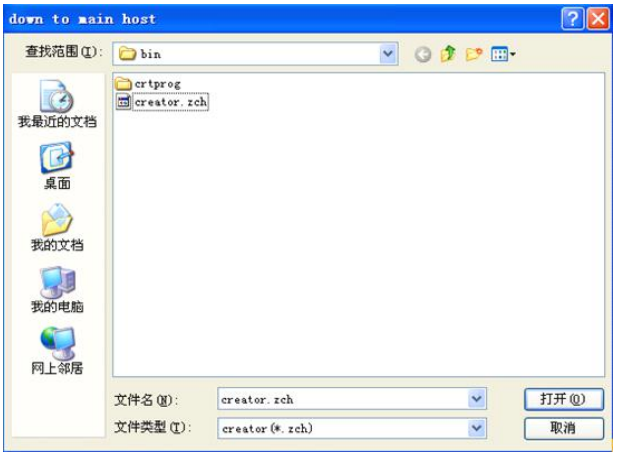
主机 IP 地址：根据主机 IP 地址填写（通常为 192.168.1.20）

端口：固定为 7755。填

写完毕后点击 Apply。

这里我们假设主机 IP 地址为 192.168.1.20(注意主机 IP 地址不能与电脑 IP 地址冲突)。

3. 选择“网络控制”→“下载到 0 主机”，或点击工具  图标。在弹出的对话框选择需要下载的工程文件，如图：



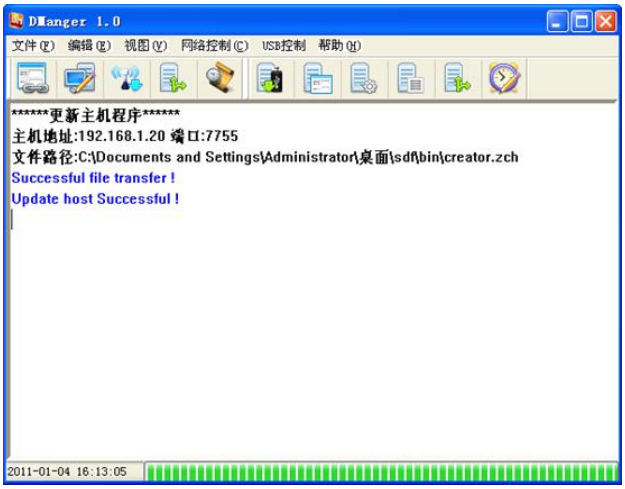
4、点击打开下载工程到主机

如果下载成功，窗口显示：

Successful file transter!

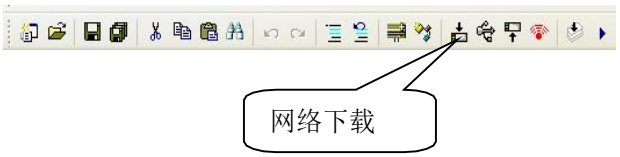
Updata host Successful!

如图：



如果下载不成功，重启主机后重复上述 2-4 步骤再次下载。

如果不是首次下载工程，且确定主机连接 IP 正确，可直接点击工具栏网络下载图标下载，而不需要重复上面所述步骤。



6.8. 工程例子 }

下面来为大家提供一些工程例子，以便参考：

6.8.1. 控制主机继电器

```
    tp = T:1000:TP;  //定义 ID 号为 1000 的触屏 tp
    m_relay = M:1002:RELAY;  //定义主机继电器 m  _relay
DEFINE_EVENT
    BUTTON_EVENT(tp,1)
    {
        PUSH()
        {
            ON_RELAY(m_relay,1); //第一路继电器开
        }
    }

    BUTTON_EVENT(tp,2)
    {
        PUSH()
        {
            OFF_RELAY(m_relay,1);//第一路继电器关
        }
    }

    BUTTON_EVENT(tp,3)
    {
        PUSH()
        {
            ON_RELAY(m_relay,2); //第 2 路继电器开
        }
    }
    BUTTON_EVENT(tp,4)
    {

        PUSH()

        {
            ON_RELAY(m_relay,2); //第 2 路继电器关
        }
    }

    BUTTON_EVENT(tp,5)
    {
        PUSH()
        {
            ON_RELAY(m_relay,3); // 第 3 路继电器开
        }
    }

    BUTTON_EVENT(tp,6)
    {
        PUSH()
        {
            ON_RELAY(m_relay,3); // 第一路继电器关
        }
    }
```

6.8.2. 级联和调用模块实例

```
DEFINE_DEVICE
    tpx = T:1002:TP;
    mtp = M:0:TP;
    com = M:1003:COM;
    mr = M:1000:IR;
    io = M:1000:IO;
    tp7600 = N:07100:TP;
    acar = L:12:ACAR;

DEFINE_COMBINE
[tpx,mtp];

DEFINE_EVENT
    // 收第一个串口数据 发往第二个串口
    DATA_EVENT(com ,1)
    {

        ONDATA()
        {
            SEND_COM(com,2,"0x"  +
BYTES_TO_HEX(DATA.  Data));
```

```
        }
    }

    BUTTON_EVENT(tpx,12)
    {
        PUSH()
        {
            SEND_ACAR(acar,1,12);
            SEND_ACAR(acar,1,12);

        }
    }

    LEVEL_EVENT(tp7600,1)
    {

    SEND_M2M_LEVEL("192.168.1.10",2,LEVEL.Value);

    SEND_M2M_LEVEL("192.168.1.2",12,LEVEL.Value); // 向主机 192.168.1.2 发送触屏的 joinmuber12 的拉条数据 13
    }

    BUTTON_EVENT(tp7600,1)
    {
        PUSH()
        {

    SEND_M2M_DATA("192.168.1.2","11111"); // 向主机 192.168.1.2 发送字符 11111

    SEND_M2M_JNPUSH("192.168.1.2",12);  // 向主机 192.168.1.2 发送触屏的 joinmuber12 的按下
    SEND_M2M_JNRELEASE("192.168.1.2",12

);//向主机 192.168.1.2 发送触屏的 joinmuber12 的松开

            //
        }
    }

    // 串口 com 第 2 路数据发往主机 192.168.1.10
    DATA_EVENT(com,2)
    {
        ONDATA()
        {

    SEND_M2M_DATA("192.168.1.10",BYTES_TO_STRING(DATA. Data));
        }

    }

    M2MDATA_EVENT()
    {
        ONDATA()
        {
            TRACE("all ip:"+DATA.STR_M2MIPADDR +" data:" +DATA.DataString);
        }
    }

    M2MDATA_EVENT("127.0.0.1")
    {
        ONDATA()
        {
            TRACE("dai ip ip:"+DATA.STR_M2MIPADDR +" data:" +DATA.DataString);
            int p = DATA.B1;
            string ip = DATA.STR_M2MIPADDR;
        }
    }

    M2MDATA_EVENT("127.0.0.11")
    {

        ONDATA()
        {
```

```
        }
    }

DEFINE_CALL_TEMPLATE
    abc(tpx,3,5,com,4);
```

6.8.3. 墙上面板

DEFINE_DEVICE

CRNET_RELAY= N:6:RELAY; // 定义 CRNET 的 ID 为 6 的继电器

CRNET_WallBoard = N:9:WM; // 定义 CRNET 的 ID 为 9 的墙上面板

DEFINE_EVENT

//墙上面板 CRNET_WallBoard 第 1 路返回数据是控制继电器 CRNET_RELAY 的开关。

//对应墙上面板 DATA. Data[0]==1 表示按下 == 0 表示弹起。DATA. Data 为返回的字节数组

```
DATA_EVENT(CRNET_WallBoard,1)
{
    ONDATA()
    {
        TRACE("recive data from zhu gong");
        //////DATA. Data 为 con 口返回的数据，字节数组表示 //注：暂时不支持 0x8 表示
        if(DATA. Data[0]==1)
        {

ON_RELAY(CRNET_RELAY,1);
            //ON_VOL(CRNET_VOL_01);

//SET_VOLTOTOL(CRNET_VOL_01,"56666");
        }

        else if(DATA. Data[0]==0)
        {

OFF_RELAY(CRNET_RELAY,1);

        }
    }
}

//同上，为第 2 路
DATA_EVENT(CRNET_WallBoard,2)
{
    ONDATA()
    {
        TRACE("recive data from zhu gong");
        //DATA. Data 为 con 口返回的数据，字节数组表示 //注：暂时不支持 0x8 表示
        if(DATA. Data[0]==1)
        {

ON_RELAY(CRNET_RELAY,2);
            //ON_VOL(CRNET_VOL_01);

//SET_VOLTOTOL(CRNET_VOL_01,"56666");
        }
        else if(DATA. Data[0]==0)
        {

OFF_RELAY(CRNET_RELAY,2);
        }
    }
}

//同上，为第三路
DATA_EVENT(CRNET_WallBoard,3)
{

    ONDATA()
    {
        TRACE("recive data from zhu gong");

        //DATA. Data 为 con 口返回的数据，字节数组表示 //注：暂时不支持 0x8 表示
        if(DATA. Data[0]==1)

        {
```

```
                ON_RELAY(CRNET_RELAY,3);

            }
            else if(DATA. Data[0]==0)
            {

    OFF_RELAY(CRNET_RELAY,3);
            }
        }
    }
```

第四路，第五路，第六路……如此类推

6.8.4. 灯光与音量控制

DEFINE_DEVICE

```
    tp_1 = T:20:TP;    //定义 ID 为 20 的触摸屏设备
    CRNET_VOL = N:4:VOL;        //定义 CRNET 的 ID 为 4 的音量控制器
    CRNET_light = N:5:LITE;    //定义 CRNET 的 ID 为 5 的调光器
```

DEFINE_VARIABLE

```
    int cr_light;          //定义 CRNET 的调光器的亮度变量值
    int cr_vol;                 //定义 CRNET 的音量控制器音量值变量值
```

DEFINE_EVENT

```
    //tp_1 按钮按下到弹起之间一直执行 REPEAT 块内的代码。即按住将 CRNET_light 的 1 路变亮
    BUTTON_EVENT(tp_1,52)
    {
        REPEAT()
        {
            cr_light=cr_light+100;
            if(cr_light>65535)
            {
                cr_light=65535;
            }

    SEND_LITE(CRNET_light,1,cr_light);
        }
    }

    // 同上，灯光减
    BUTTON_EVENT(tp_1,53)
    {
        REPEAT()
        {

    SEND_LITE(CRNET_light,1,cr_light);
            cr_light=cr_light-100;
            if(cr_light<0)
            {
                cr_light=0;
            }
        }
    }

    // 同上，控制总音量加
    BUTTON_EVENT(tp_1,50)
    {
        REPEAT()
        { SET_VOLTOTOL(CRNET_VOL,1,(cr_v

    ol));

    SET_VOLTOTOL(CRNET_VOL,2,(cr_vol));
            cr_vol=cr_vol+100;
            if(cr_vol>65535)
            {
                cr_vol=65535;
            }
        }
    }

    // 同上，控制总音量减
    BUTTON_EVENT(tp_1,51)
    {
        REPEAT()
        { SET_VOLTOTOL(CRNET_VOL,1,(cr_v

    ol));
```

```
SET_VOLTOTOL(CRNET_VOL,2,(cr_vol));
            cr_vol=cr_vol-100;
            if(cr_vol<0)
            {
                cr_vol=0;
            }
        }
```

6.8.5. 两路继电器互锁

```
DEFINE_DEVICE
    RL_M = M:1000:RELAY;          //定义主机继电器
    tp_1 = T:10:TP;               // 定 义 ID 为 10 的触摸屏设备

DEFINE_EVENT
    BUTTON_EVENT(tp_1,1)
    {
        PUSH()
        {
            ON_RELAY(RL_M,1);          // 打开主机第 1 路继电器
             OFF_RELAY(RL_M,2);        // 关闭主机第 2 路继电器
        }
        RELEASE()
        {
        }
    }
    BUTTON_EVENT(tp_1,2)
    {
        PUSH()
        {
            ON_RELAY(RL_M,2);          // 打开主机第 2 路继电器

             OFF_RELAY(RL_M,1);        // 关闭主机第 1 路继电器
        }
        RELEASE()
        {
        }
    }
```

7.9.6 重复按下按钮不影响延时操作

```
DEFINE_DEVICE
    REL = N:8:RELAY;              // 定 义 CR-NET，ID 号为 8 的继电器
    tp_1 = T:10:TP;               // 定 义 ID 为 10 的触摸屏设备

DEFINE_VARIABLE
    int py=1;       //定义 CRNET 继电器的变量值

DEFINE_EVENT
    BUTTON_EVENT(tp_1,1)
    {
        PUSH()
        {
            //打开 CR-NET 继电器第 1 路
            ON_RELAY(REL,1);
            //等待 5 秒后关闭 CR-NET 继电器第 1 路；等待时间内重复按下按钮不影响操作
            if(py==1)
            {
                py=0;
                WAIT 5000
                {
                OFF_RELAY(REL,1);
                py=1;
                }
            }
        }

        RELEASE()
        {
        }
    }
```

7.9.7 矩阵的控制

```
DEFINE_DEVICE
    tp = T:10:TP;
    mcom = M:1000:COM;

DEFINE_VARIABLE
    string checkMode ="";
    string outstr = "";
```

```
    string instr ="";

DEFINE_FUNCTION
    //矩阵输出函数
    void shuchu()
    {

    SEND_COM(mcom,1,instr+checkMode+outs
tr+".");
    }

DEFINE_START
    SET_COM(mcom,1,9600,8,0,10,0,232);

DEFINE_EVENT
    //模式选择BUTTON_EVENT(tp,4)
    {
        PUSH()
        {
            checkMode = "B";//AV 模式
        }
    }

    BUTTON_EVENT(tp,6)
    {
        PUSH()
        {
            checkMode = "A"; //A 模式
        }


    }

    BUTTON_EVENT(tp,5)

    {
        PUSH()
        {
            checkMode = "V"; //V 模式
        }
    }

    //选择输出
BUTTON_EVENT(tp,1)
{
    PUSH()
    {
        outstr ="1";
    }
}

BUTTON_EVENT(tp,2)
{
    PUSH()
    {
        outstr ="2";
    }
}

BUTTON_EVENT(tp,3)
{
    PUSH()
    {
        outstr ="3";
    }
}

// 选择输入
BUTTON_EVENT(tp,11)
{
    PUSH()
    {
        instr ="1";
        shuchu();
    }
}

BUTTON_EVENT(tp,12)

{
    PUSH()
    {
        instr ="2";
        shuchu();
    }
}
```

```
BUTTON_EVENT(tp,13)
{
    PUSH()
    {
        instr ="3";
        shuchu();

    }
}


BUTTON_EVENT(tp,100)
{
    PUSH()
    {
        SEND_COM(mcom,1,instr   +"All.");

    }
}
```

第七章、技术参数

功能说明	CR-PGMIII
内存	256M DDR-RAM，1G FLASH
AV-NET 、 AV-LINK及 以 太 网 (TCP/IP)	有
红外独立发射端口	8 路
数字 I/0 口	8 路
弱电继电器接口	8 路
RS-232/422/485 通讯串口	8 路
USB 接口	1 路
扩展插口	有
接地线孔	有
机箱尺寸	2U
重量	约 4.5KG
AC100—240V 自适应电源	有

第八章、常见问题与排除

故障问题	解决方法
触屏无法控制主机	● 检查触屏工程的按钮 Join No.值是否与主机程序中触屏的 JOIN NO.相同；
	● 检查触屏 ID 是否分别与主机程序中触屏 ID 相同；
	● 检查接收器与控制主机之间的连线是否正确无误；
	● 检查接收器是否装上天线，触屏与接收器是否工 作在有效的距离范围里；
	● 请检查触屏与接收器近距离控制是否能控，不能控制可能接收器或触屏其中一个设备有问题，请送专业人员检修。
电脑无法给触屏下载程序	● 检查 USB 数据线是否连接正确；
	● 检查触屏的驱动是否安装成功，此情况可重新安装驱动程序解决；
	● 检查触屏用户程序是否与触屏型号相同；
	● 检查电脑 USB 口是否能正常工作使用；
	● 检查触屏是否已接通电源并点亮触屏。
触屏无显示	● 触屏电池的电量可能用完，此时请充电；
	● 触屏充电时请确定电源适配器已经接正确；
	● 检查触屏电池是否安装不良，此时请取出电池重新安装可解决。
触摸触屏按钮无反应	● 可能触屏出现漂移现象，请进入设置页重新校准坐标可解决问题；
	● 在触屏用户程序中查按钮是否画错成文本，重新修正按钮可解决；
	● 检查触屏是否电量不足，请及时充电。