I-7188XC 系列中文用户手册

保固说明

泓格科技股份有限公司 (ICP DAS) 对于所生产的产品有瑕疵之材料，均保证原始购买者于交货日起保有为期一年的保固。

免责声明

泓格科技股份有限公司对于应用本产品所造成的损害并不负任何法律上的责任。本公司保留有任何时间未经通知即可变更与修改本文件内容之权利。本文所含信息如有变更，恕不予另行通知。本公司尽可能地提供正确与可靠的信息，但不保证此信息的使用或其他团体在违反专利或权利下使用。此处包含的技术或编辑错误、遗漏，概不负其法律责任。

版权所有

商标识别

本文件提到的所有公司商标、商标名称及产品名称分别属于该商标或名称的拥有者所有。

技术服务

如有任何问题，请与本公司客服联络，我们将尽速为您服务。

Email 信箱 ： service@icpdas.com

目　录

1. 产品简介

I-7188XC(D)可扩展嵌入式控制器系列，适用于工业应用并可取代 PC 或恶劣环境下的 PLC 设备。此外，它还支持 I/O 扩展总线，并可方便地扩展各种功能，例如：D/I、D/O、A/D、D/A、UART、Flash 记忆体、具有后备电池的 SRAM、AsicKey 和其它 I/O 功能。大多数 I/O 类型功能该扩展总线均可支持。ICP DAS 提供多于 10 种 I/O 扩展板以使 I-7188XC(D)适于各种不同应用功能的需求。

产品清单

除该产品用户手册外，产品包装内还包含如下项目：

- 一个 I-7188XC(D)模块
- 一条程序下载线(CA0910)
- 一个随机赠送的 CD，包含软件驱动及相关用户手册电子版
- 一张版本注释说明

注意：若以上项目有任何损害或遗失，请联系购买产品的经销商。推荐保存好所有配件及硬纸盒，以备未来可能的运输使用。

1.1 产品特色

- 嵌入式 80188 CPU，20M 或相兼容
- I-7188 系列的低成本版
- 内置 EEPROM
- 内建双串口串口：COM1 和 COM2
- 串口驱动为中断方式工作并有 1K 顺序缓存
- 支持 I/O 扩展总线（可安装一片 IO 扩展板）
- 三路数字量输入通道
- 三路集电极开路输出通道
- 在 485 端口上内置自适应 ASIC 控制芯片
- 7 段 LED 显示可选
- 操作系统 ICP DAS　MiniOS7
- 程序下载端口：COM1

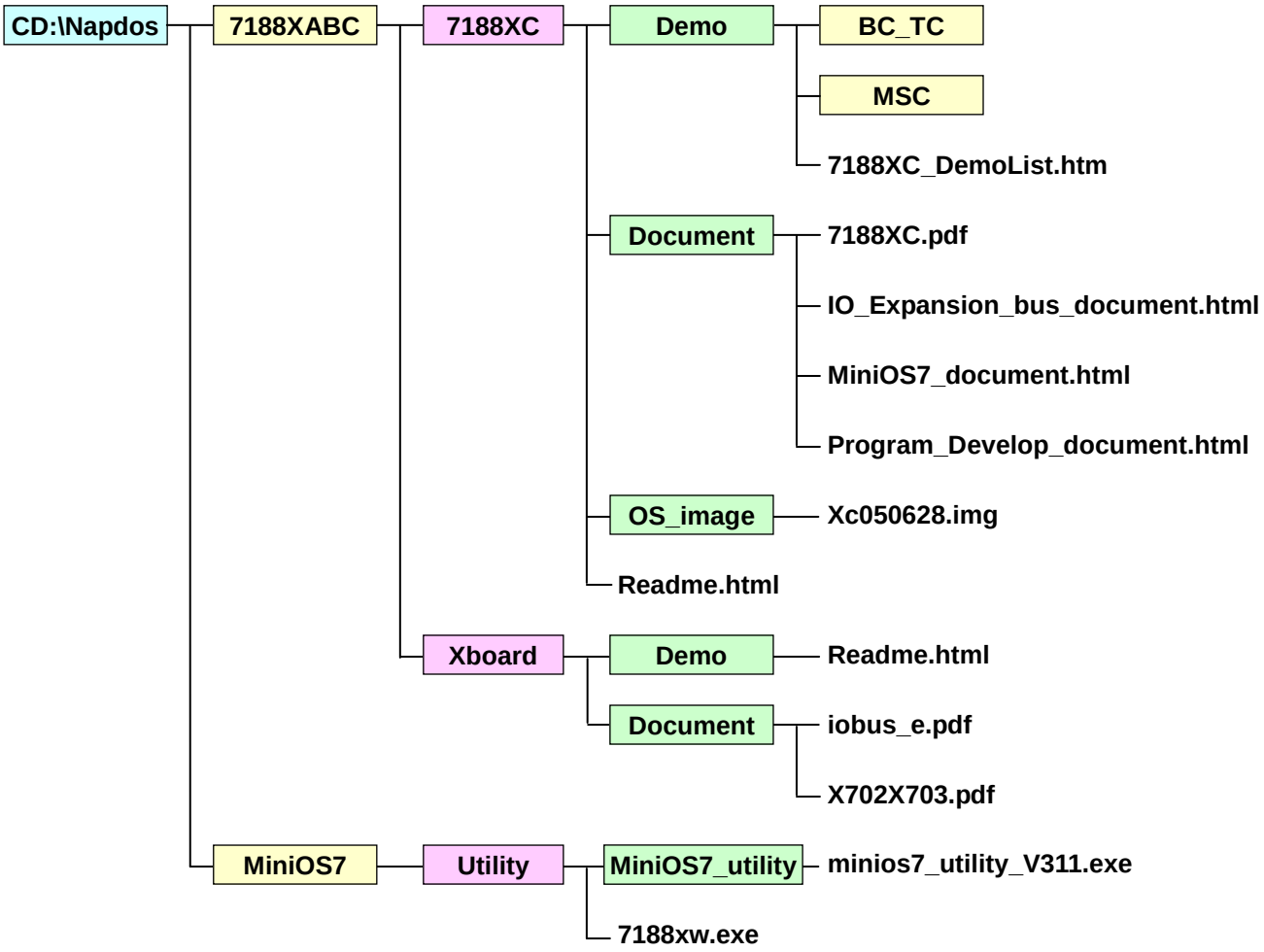
1.2 产品规格

CPU 模块	
CPU	80188 CPU, 20MHz 或相兼容
SRAM	128K 字节
Flash	256K 字节(通过 OEM 可扩展至 512K 字节)
EEPROM	2K 字节
NVRAM	不支持
RTC (实时时钟)	不支持
硬件序列号	不支持
内置看门狗	支持
通讯接口	
COM 1	RS-232/RS-485
COM 2	RS-485
COM 3	无
COM 4	无
以太网接口	无
数字量输入	
输入通道	2
Contact	支持干接点
低电平	接地
高电平	悬空
数字量输出	
输出通道	3
输出类型	集电集开路
最大负载电流	100mA
负载电压	+30V/DC 最大
LED 显示	
1 LED 作为电源/通讯指示器	
5 位 7 段 LED 显示 (仅 I-7188XCD 支持)	
尺寸	
119mm x 72mm x 33mm	
操作环境	
运行温度	-25°C to +75°C
存储温度	-30°C to +80°C
相对湿度	10 to 90% RH (非凝露)
电源	
电源需求	10 to 30V/DC (非固定)

功耗	2.0W —— I-7188XC　　3.0W —— I-7188XCD

1.3 软件文件资料

所有有关 I-7188XC(D) 文档及软件路径可在如下目录树中获得。

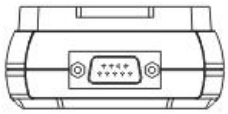


以上所有相关文档及软件均可在 ICP DAS 主页上下载：
http://ftp.icpdas.com/pub/cd/8000cd/napdos.
此外也可从随机赠送的光盘中获得。

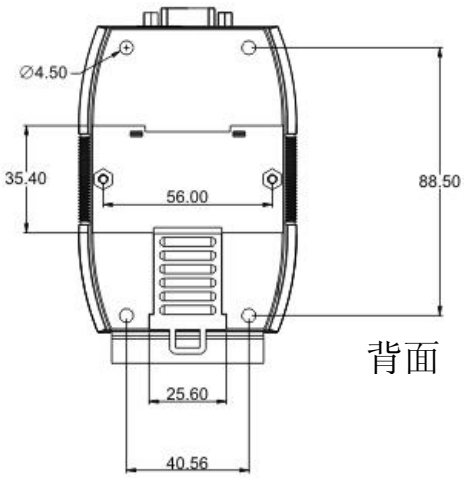
有关 I/O 扩展总线相关资料可从 **iobus_e.pdf**
(CD:\Napdos\7188XCBC\Xboard\Document\)中获得。
此外，使用前请务必阅读文档 Readme.html
(CD:\Napdos\7188XCBC\7188XC\)。.

1.4 硬件信息

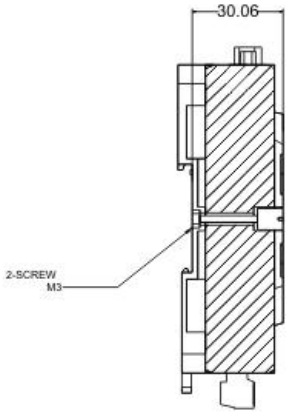
1.4.1 I-7188XC(D)硬件尺寸图表



正面

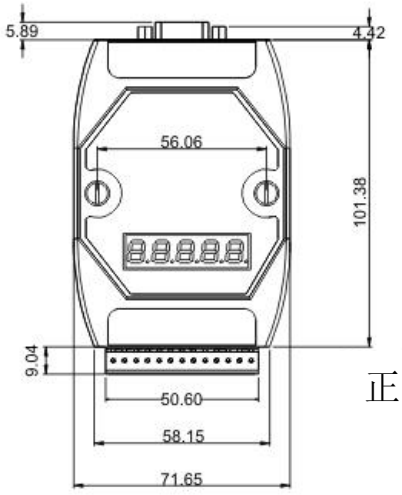


背面

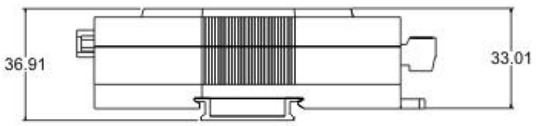


侧面

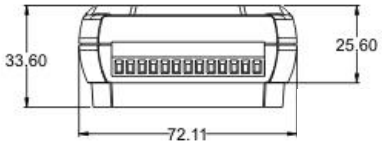
单位: mm



正面

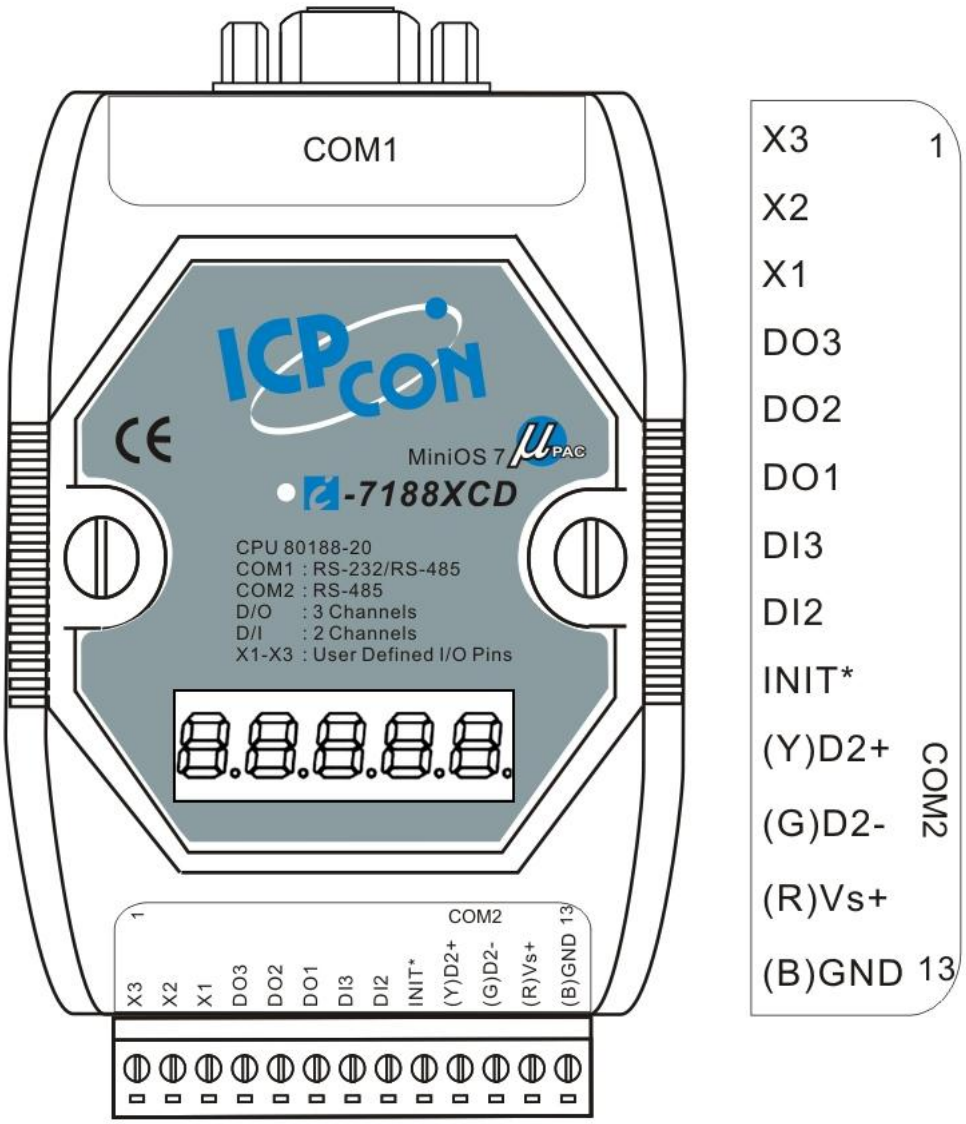


DIN-RAIL 装备支架



下面

1.4.2 引脚定义

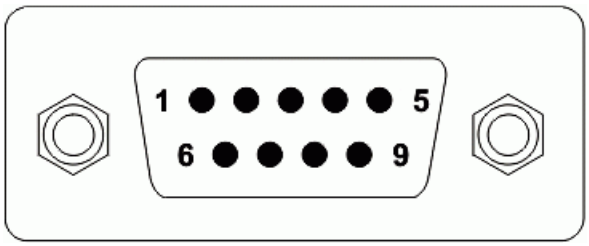


14 针引脚分配如下

引脚	名称	说明
1	X3	连接 I/O 扩展板
2	X2	连接 I/O 扩展板
3	X1	连接 I/O 扩展板
4	DO3	数字量输出，最大 100mA, 30V
5	DO2	数字量输出，最大 100mA, 30V
6	DO1	数字量输出，最大 100mA, 30V
7	DI3	数字量输入, 3.5V ~ 30V

8	DI2	数字量输入, 3.5V ~ 30V
9	INIT*	初始化引脚
10	D2+	串口 COM2 (RS-485)引脚 DATA+
11	D2-	串口 COM2 (RS-485)引脚 DATA-
12	+VS	供电电源正极+V (+10 到 +30V/DC, 非固定)
13	GND	供电电源地

串口 COM1 连接器(DB-9 Male)引脚分配如下：



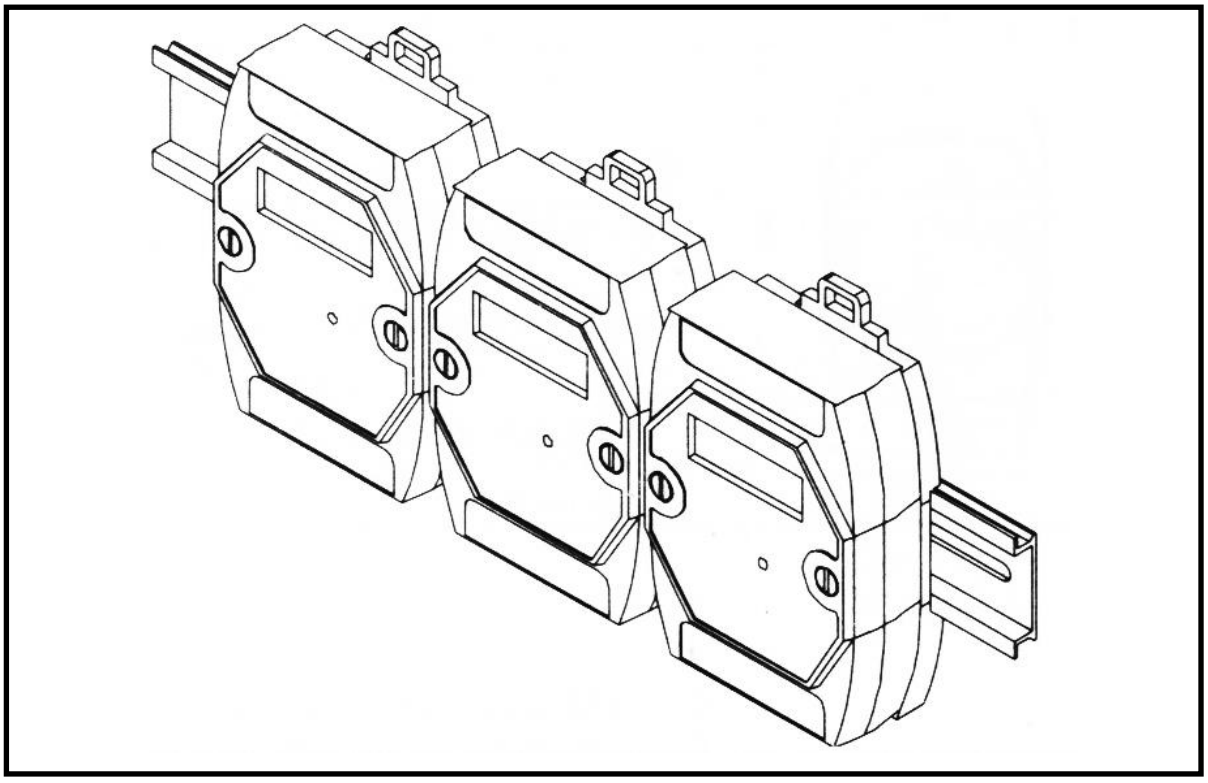
DB9-Male (COM1)

引脚	名称	说明
1	D1+	DATA+ 适用于 RS-485
2	TXD	传输数据
3	RXD	接收数据
4	N/C	空
5	GND	RS-232 信号地
6	N/C	空
7	CTS	清除发送(RS-232)
8	RTS	应答发送(RS-232)
9	D1-	DATA-适用于 RS-485

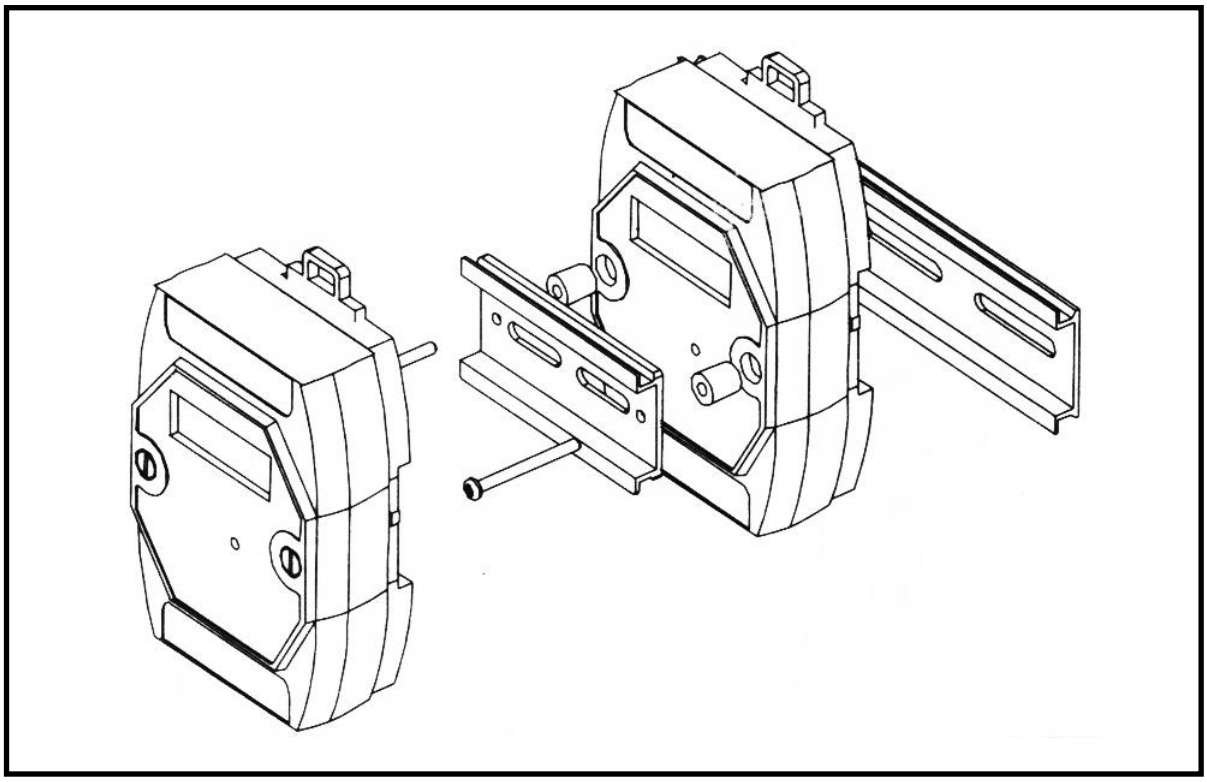
注意： 串口 COM1 即可用于 RS-232 接口也可用于 RS-485 端口，但并不意味着可同时支持 RS-232 及 RS-485。

1.4.3 I-7188XC(D)安装

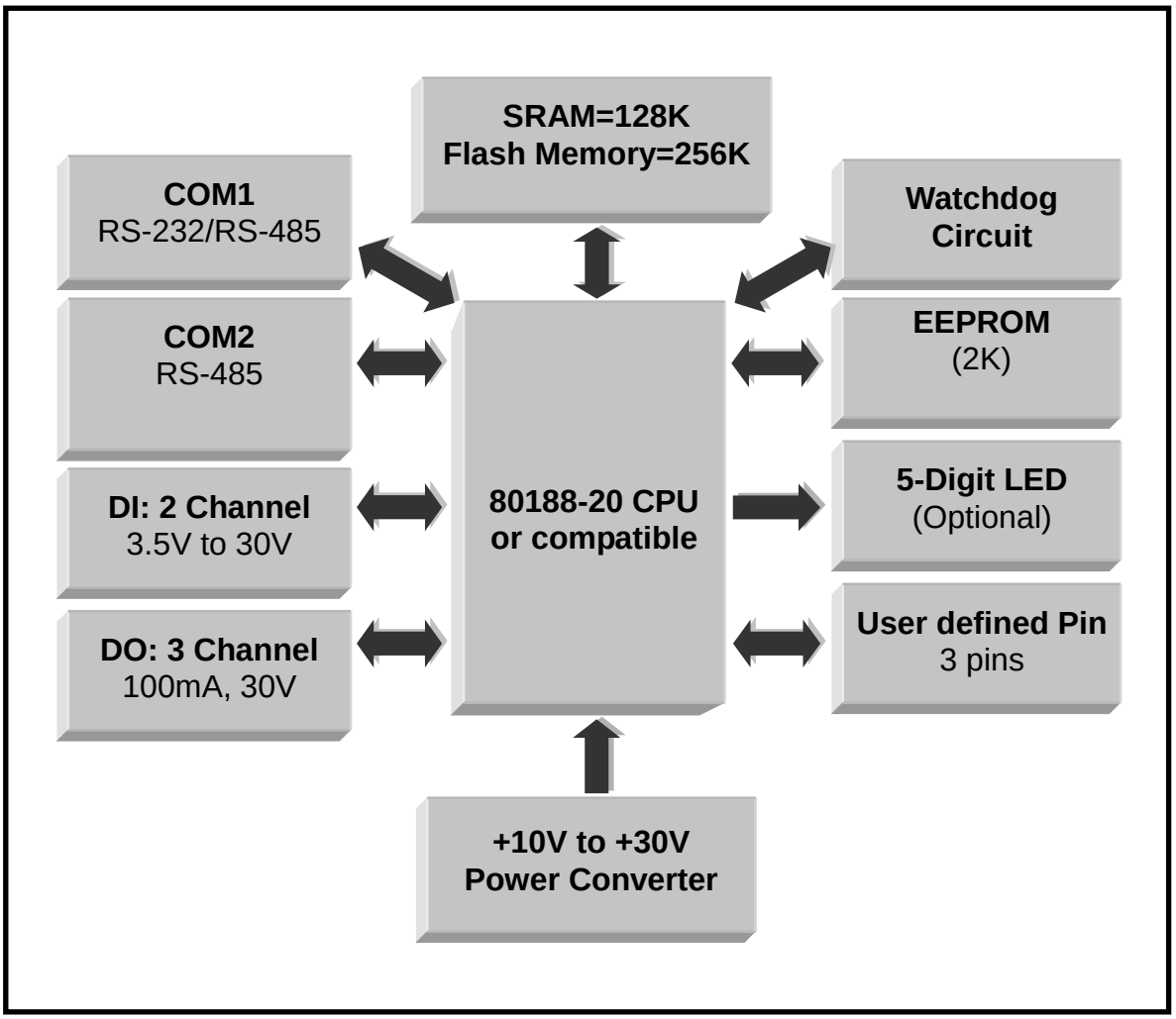
1. 导轨安装



2. 叠加安装

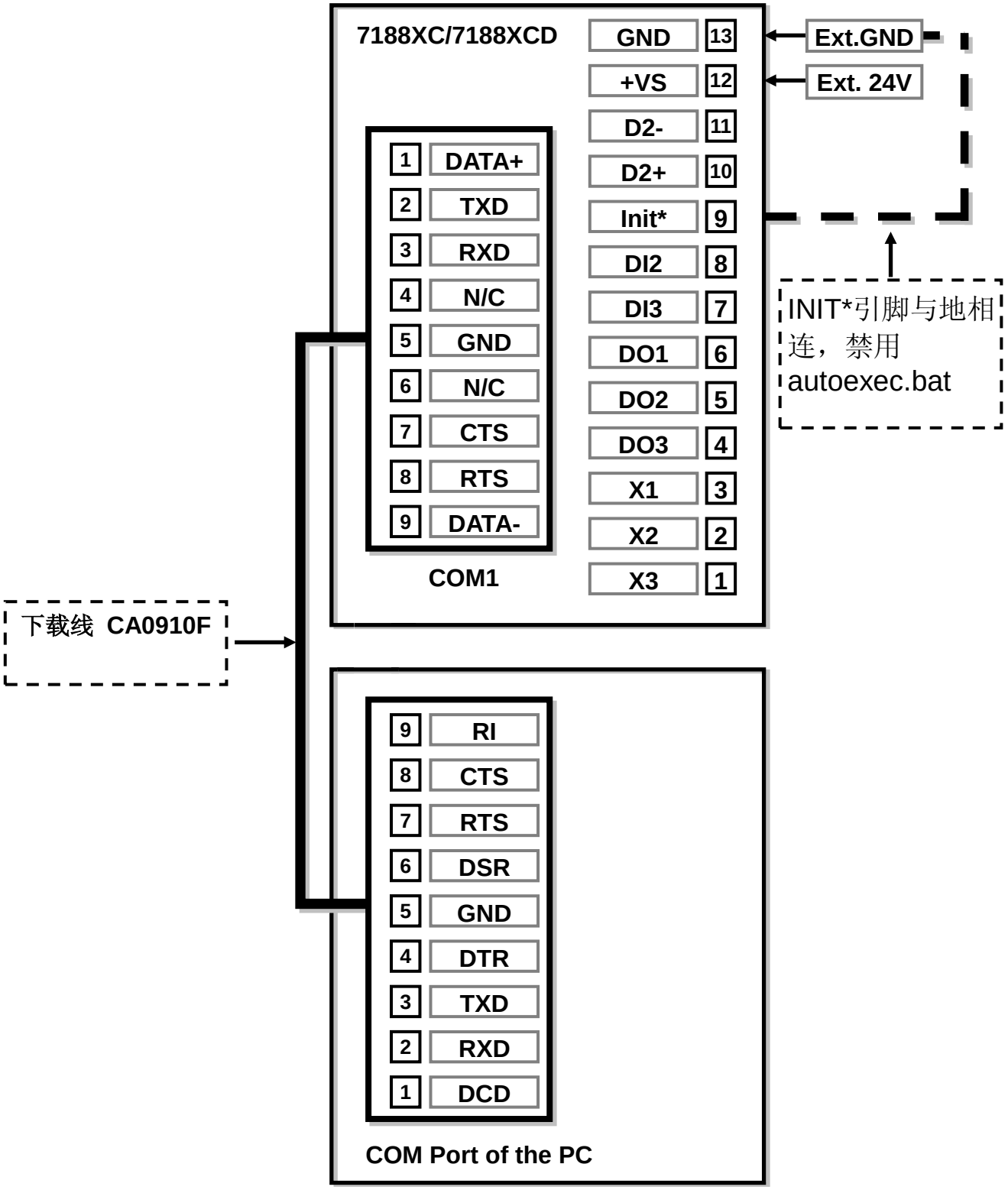


1.4.4 硬件结构图


SRAM=128K
Flash Memory=256K
COM1
RS-232/RS-485
COM2
RS-485
DI: 2 Channel
3.5V to 30V
DO: 3 Channel
100mA, 30V
80188-20 CPU
or compatible
Watchdog
Circuit
EEPROM
(2K)
5-Digit LED
(Optional)
User defined Pin
3 pins
+10V to +30V
Power Converter

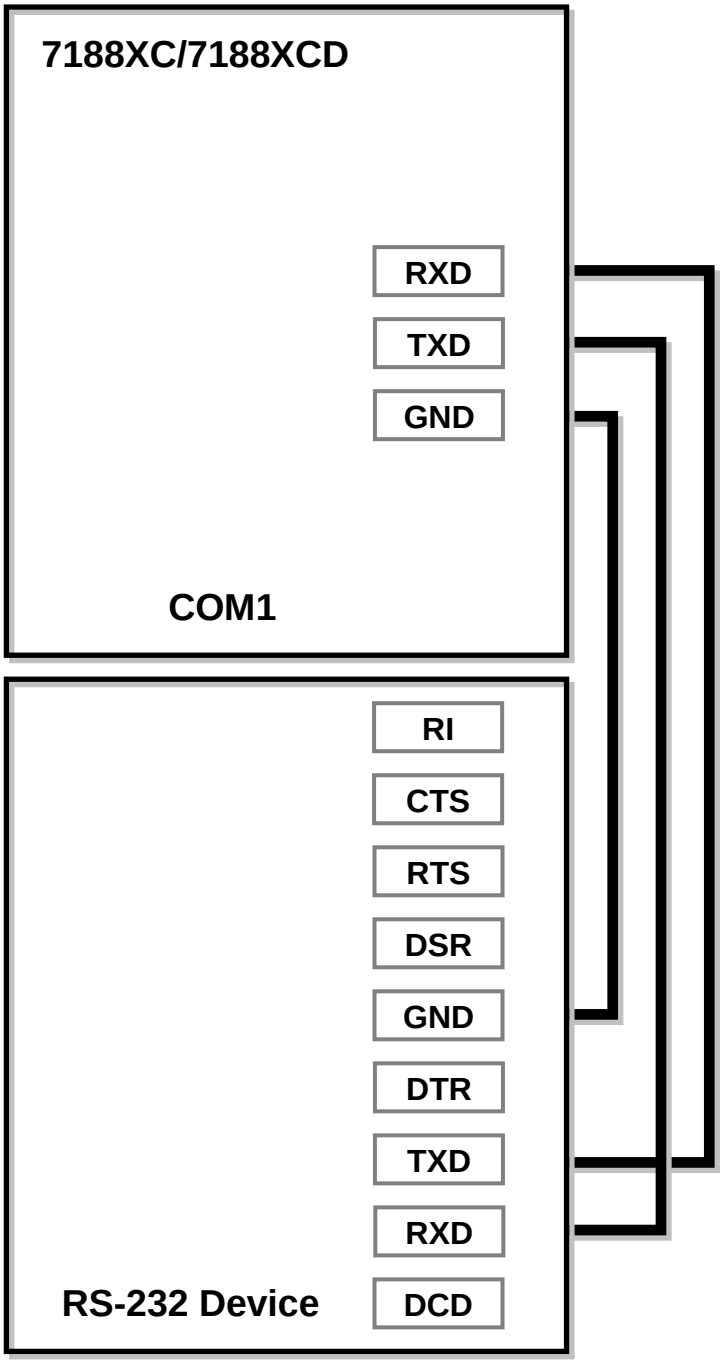
1.4.5 端口接线图

程序下载



注意：使用数据下载线 DB-9 将端口与 PC 串口相连。

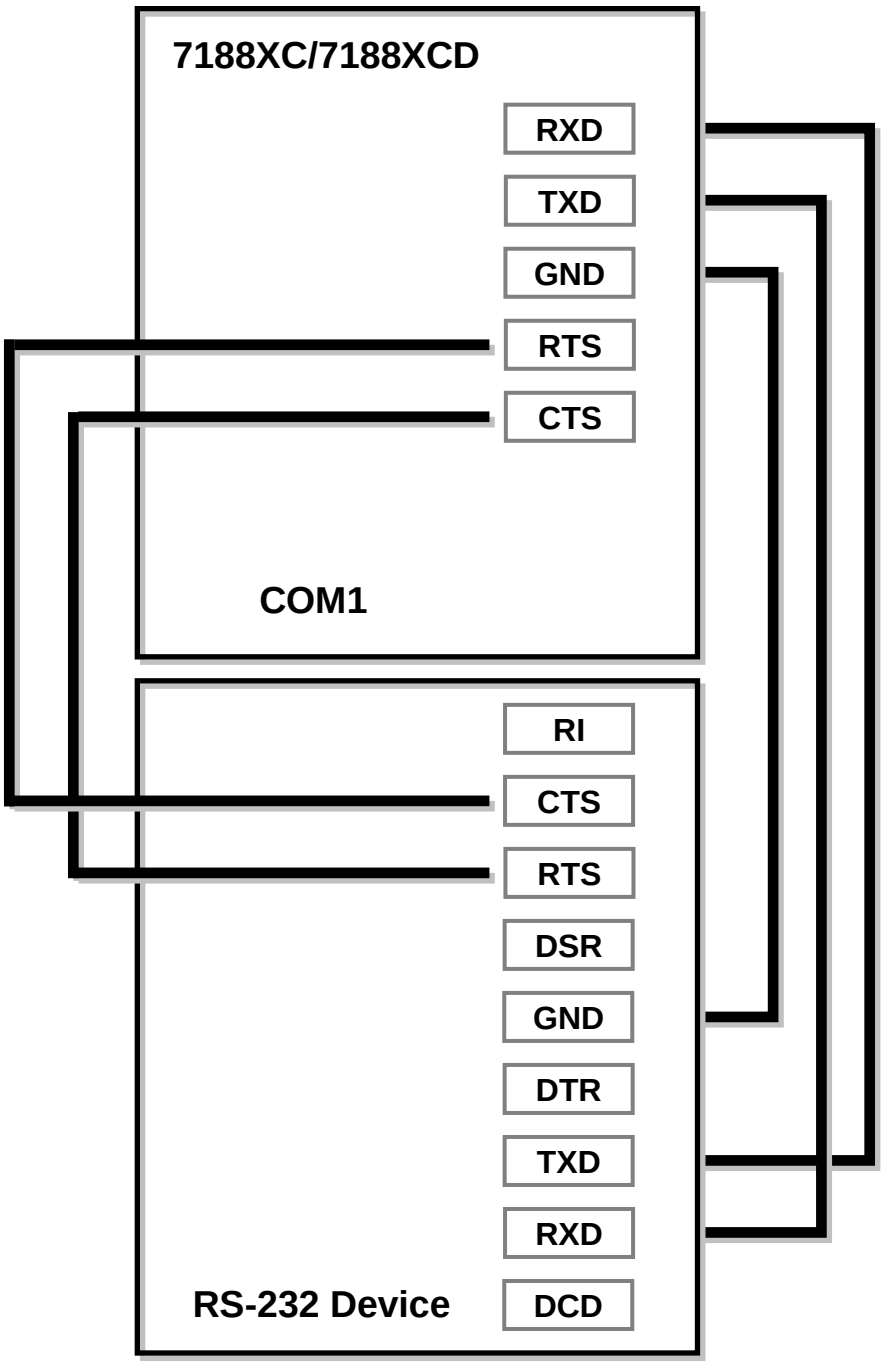
使用 3 线制 RS-232 接口



注意：3 线制接法如下

- 将 RXD 与 RS-232 设备的 TXD 相连
- 将 TXD 与 RS-232 设备的 RXD 相连
- 将 GND 与 RS-232 设备的 GND 相连

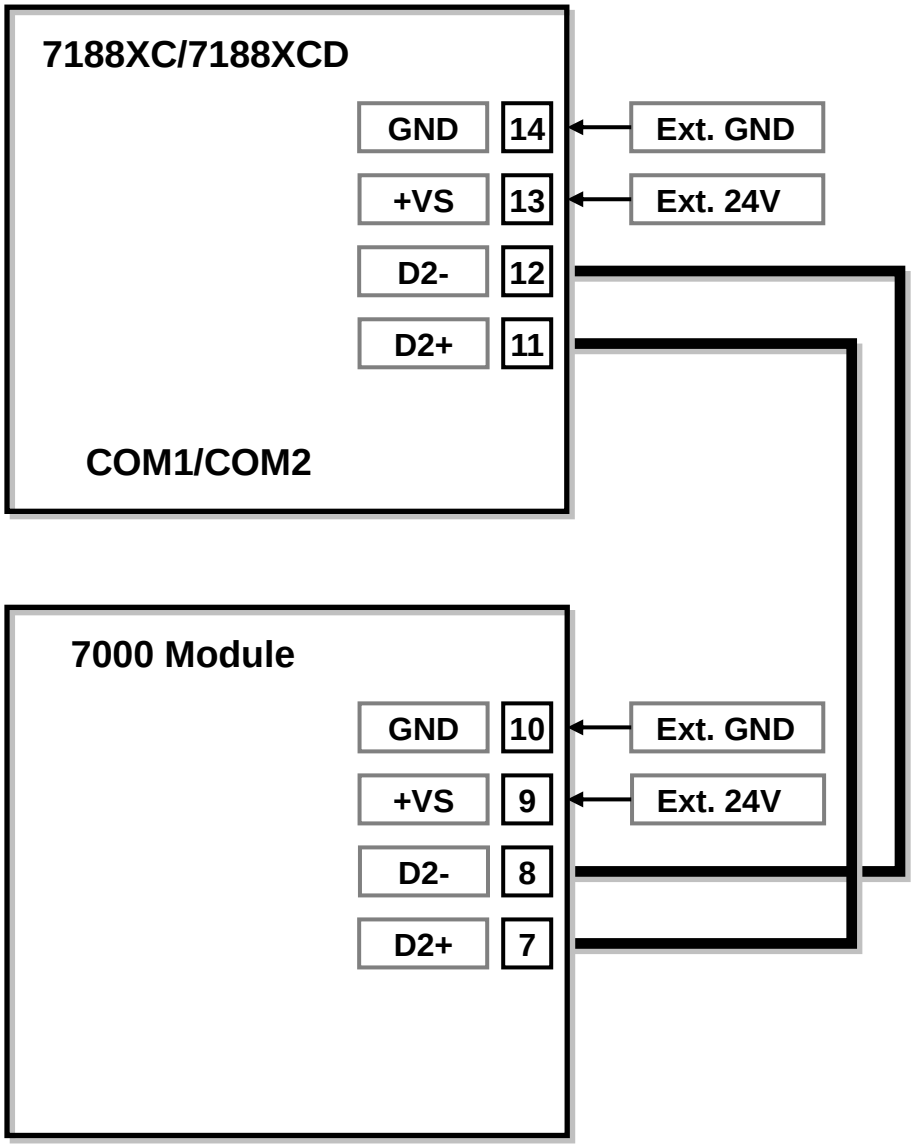
使用 5 线制 RS-232 接口



注意：5 线制接法如下

- 将 RXD 与 RS-232 设备的 TXD 相连
- 将 TXD 与 RS-232 设备的 RXD 相连
- 将 RTS 与 RS-232 设备的 CTS 相连
- 将 CTS 与 RS-232 设备的 RTS 相连
- 将 GND 与 RS-232 设备的 GND 相连

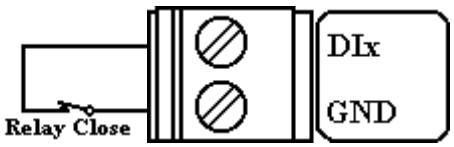
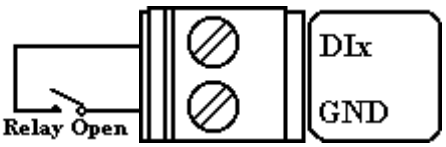
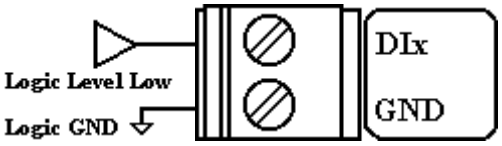
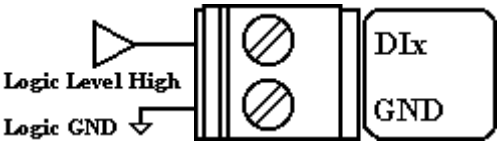
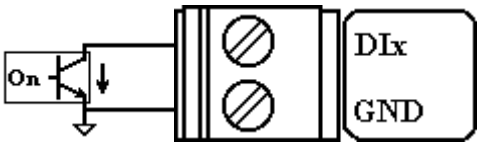
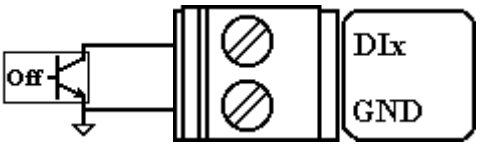
使用 RS-485 接口



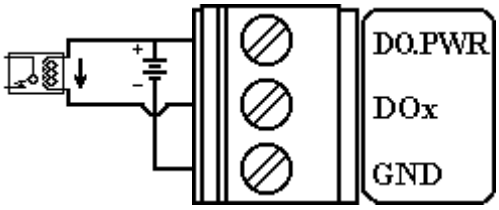
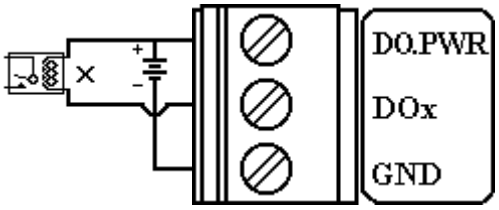
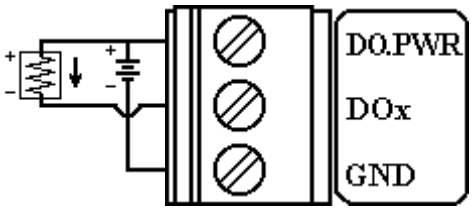
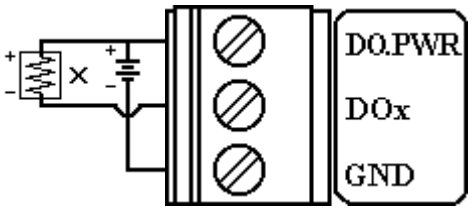
注意：RS-485 平台可直接驱动高达 256 个 I-7000 系列模块而无需任何中继。

1.4.6 DI/DO 接线图

数字量输入接线图

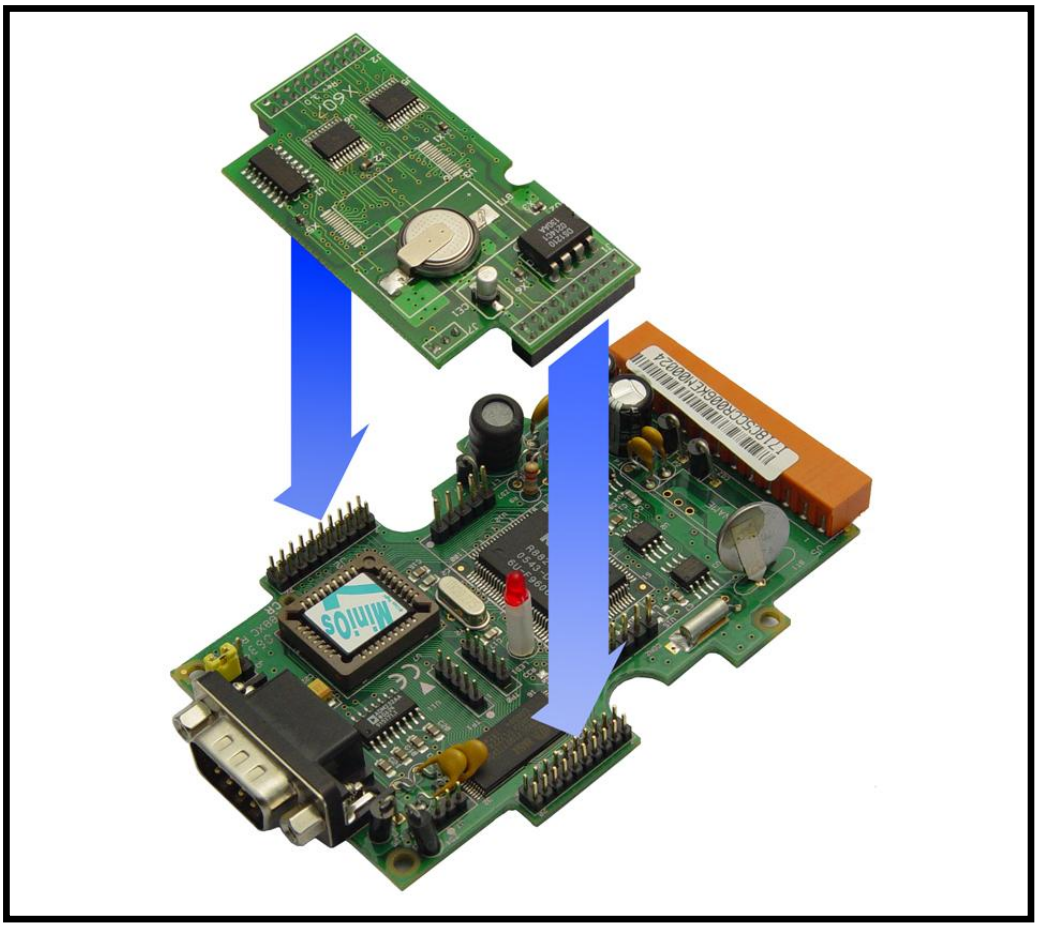
Input Type	ON State DI value as 0	OFF State DI value as 1
Relay Contact		
TTL/CMOS Logic		
Open Collector		

数字量输出接线图

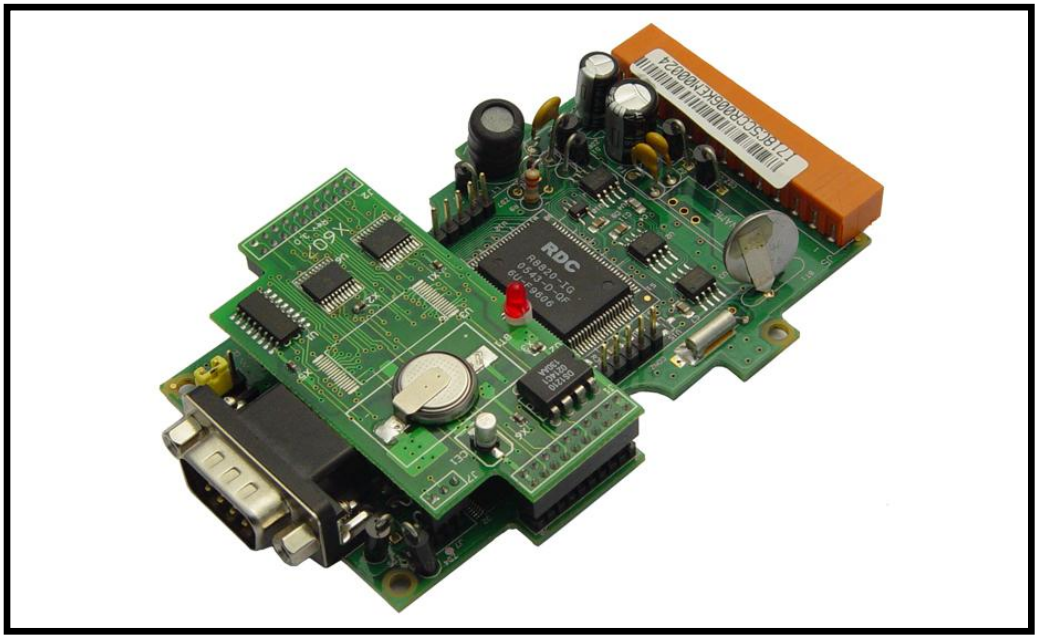
Input Type	ON State DO value as 1	OFF State DO value as 0
Drive Relay		
Resistance Load		

1.4.7 安装 I/O 扩展板

安装前：



安装后：



2. 快速上手

2.1 软件安装

步骤 1:随机赠送 CD 插入光驱中。

步骤 2: 从光盘 **CD:\Napdos\7188XABC** 中复制 7188XC 相关文件到主机硬盘上。

步骤 3: 安装 MiniOS7 Utility.
执行文件 **minios7_utility_v311.exe** 从 CD:\NAPDOS\MINIOS7\UTILITY\MiniOS7_utility\ 或 http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/minios7_utility

步骤 4: 从光盘 CD:\Napdos\MiniOS7\utility\ 复制文件 **7188xw.exe**

当所有软件复制到主机上，7188XC 文件目录参考如下：

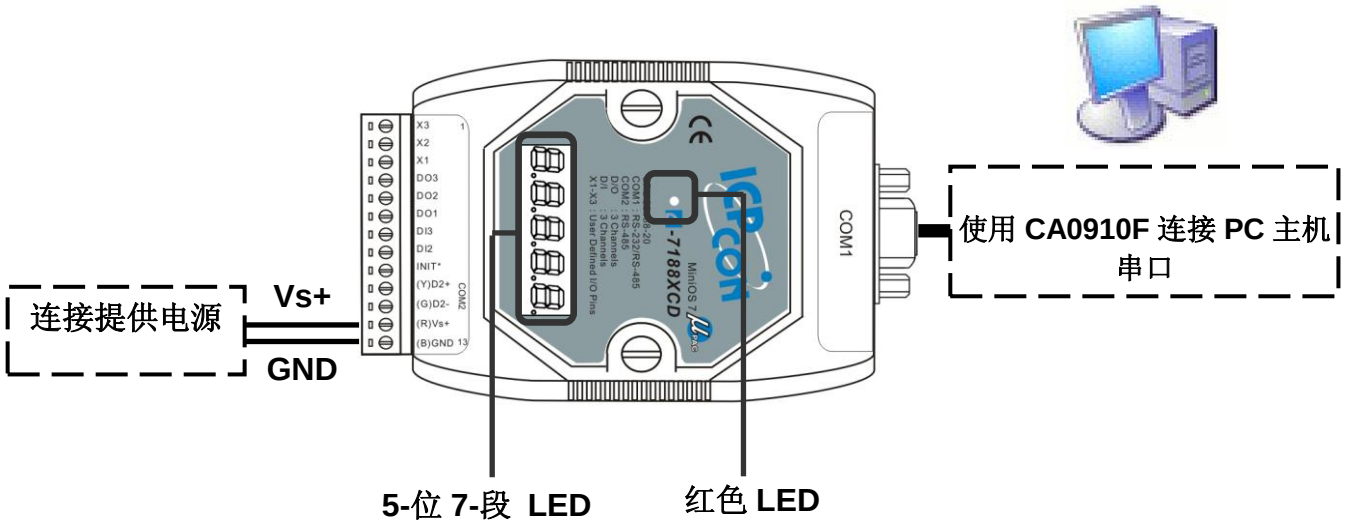
- **7188XC**
  - **Demo** → I-7188XC(D)示例程序。
    - **BC_TC** → 基于 BC++及 TC++ 示例程序
    - **MSC** → 基于 MSC 示例程序
    - **7188XC_DemoList.htm** → I-7188XC(D)示例清单
  - **Document** → I-7188XC(D)相关文档
  - **OS_image** → 操作系统 MiniOS7 镜像
  - **Readme.html** → **7188XC（D）**文档相关说明

注意： 7188xw.exe 作为 I-7188XC(D)与 PC 机桥梁，7188xw.exe 需复制到文件夹“C:\Windows\”以便在任何目录下都可以运行。

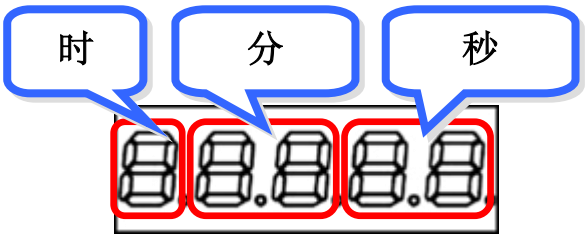
2.2 与 PC 相连

步骤 1:使用 CA0910 下载电缆将 I-7188XC(D)的 COM1 口与 PC 主机串口相连，接线图参考如下。

步骤 2: 给 I-7188XC(D)供电，(Vs+,GND) 输入范围（+10V 到 +30V DC）。

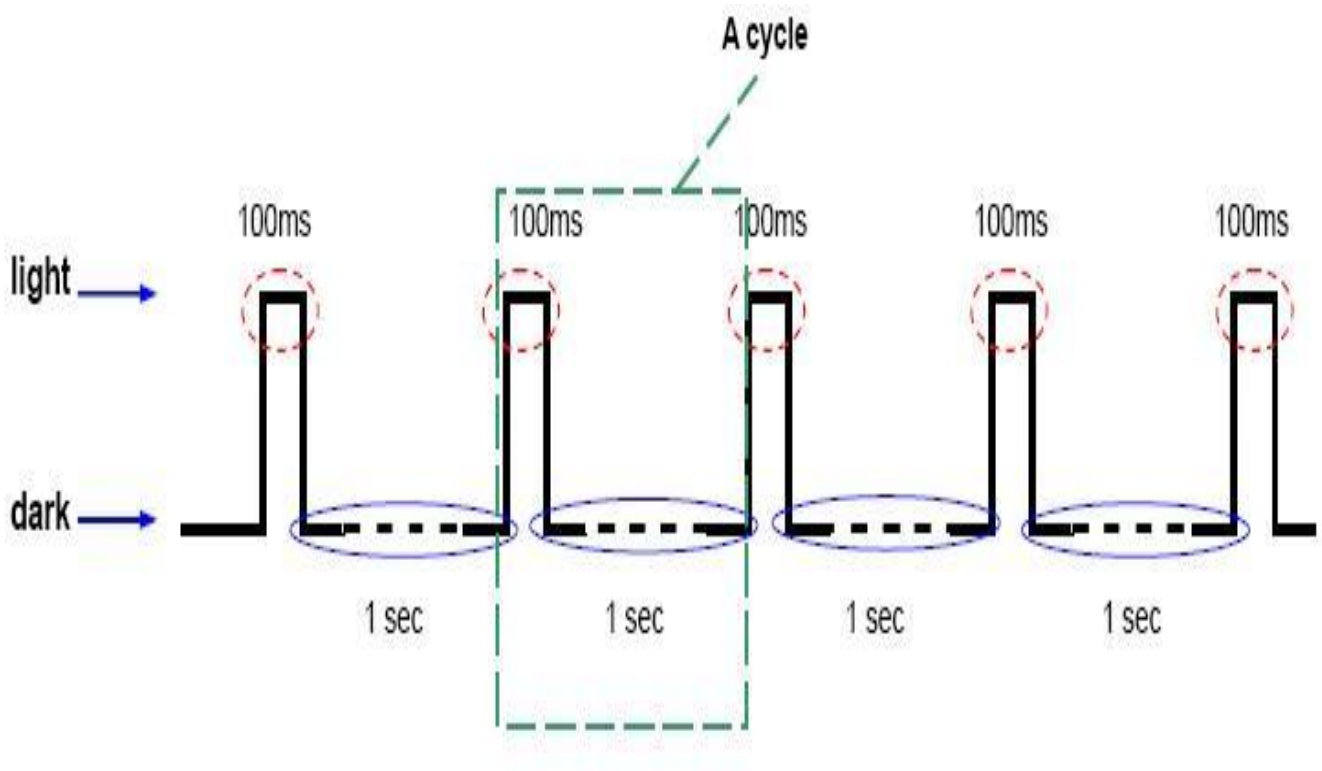


步骤 3: 供电后，5 位 7 段 LED 将持续显示如下内容：



若模块不含有 7 段 LED 显示，可跳过此环节。

步骤 4: 检查红色 LED 不断的一秒钟闪烁 1 次。时序表如下：



注意： 仅 I-7188XCD 带有 5 位 7 段 LED 显示。

2.3 I-7188XC(D)的程序下载

在使用 MiniOS7 Utility 前，确定数据线与 PC 主机和 I-7188XC(D)已连接，且 I-7188XC(D)无其它程序正在运行。有关如何连接 I-7188XC(D)与主机串口请参考章节 1.4.5 。

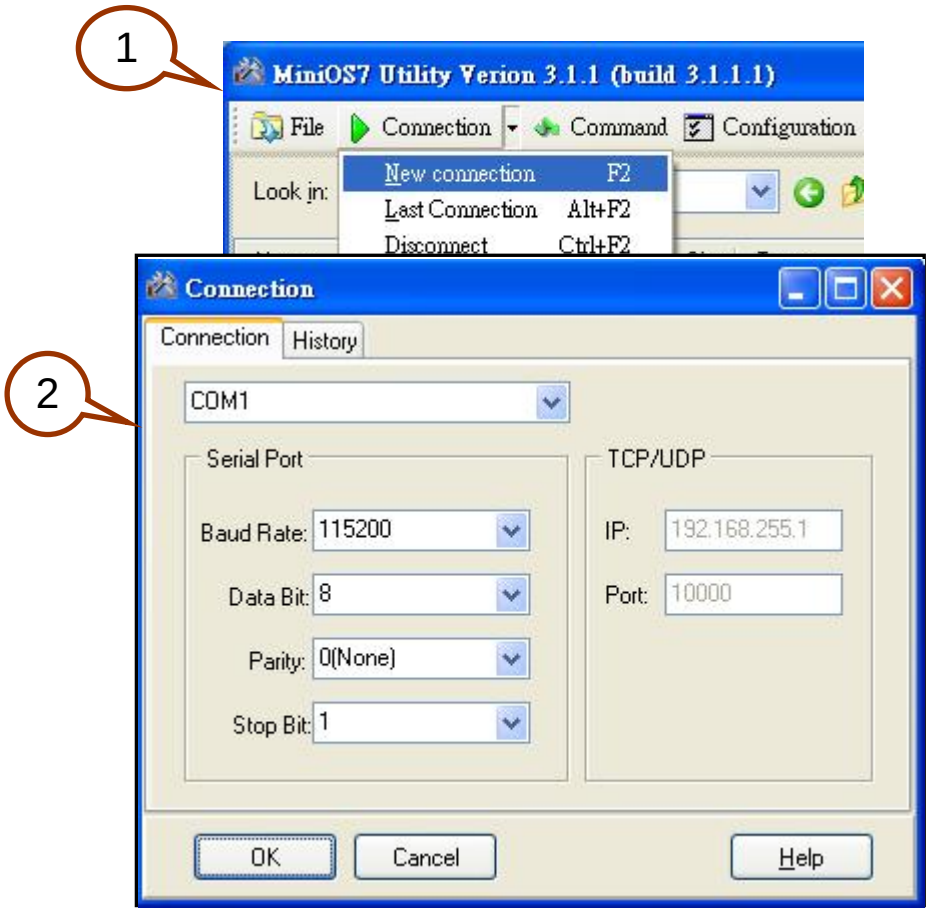
注意：7188xw.exe 也可取代 MiniOS7 Utility 来作为 I-7188XC(D)程序下载工具。有关程序下载步骤请参考**附录 B：MiniOS7 Utility 和 7188XW。**

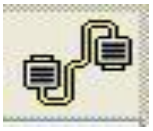
程序下载步骤参考如下 (以章节 2.1 中安装的 MiniOS7 Utility Ver 3.11 为例)：

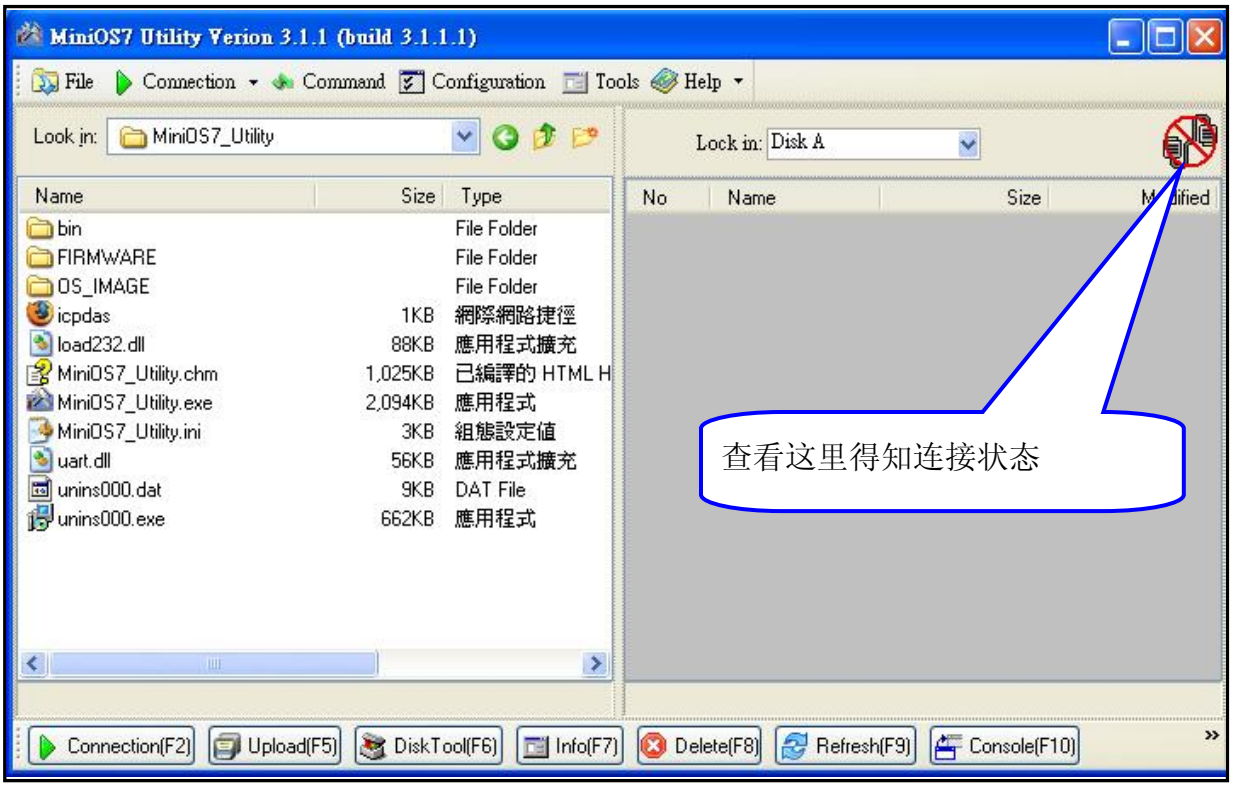
步骤 1: 从 Windows 开始菜单中获得 MiniOS7 Utility Ver 3.11 文件（程序/ICPDAS/**MiniOS7 Utility Ver 3.11/**）。

步骤 2: 点击 

并选择“New connection”。 选取合适的串口并设置其它参数。单击 **OK** 按钮应用程序将自动搜索模块。

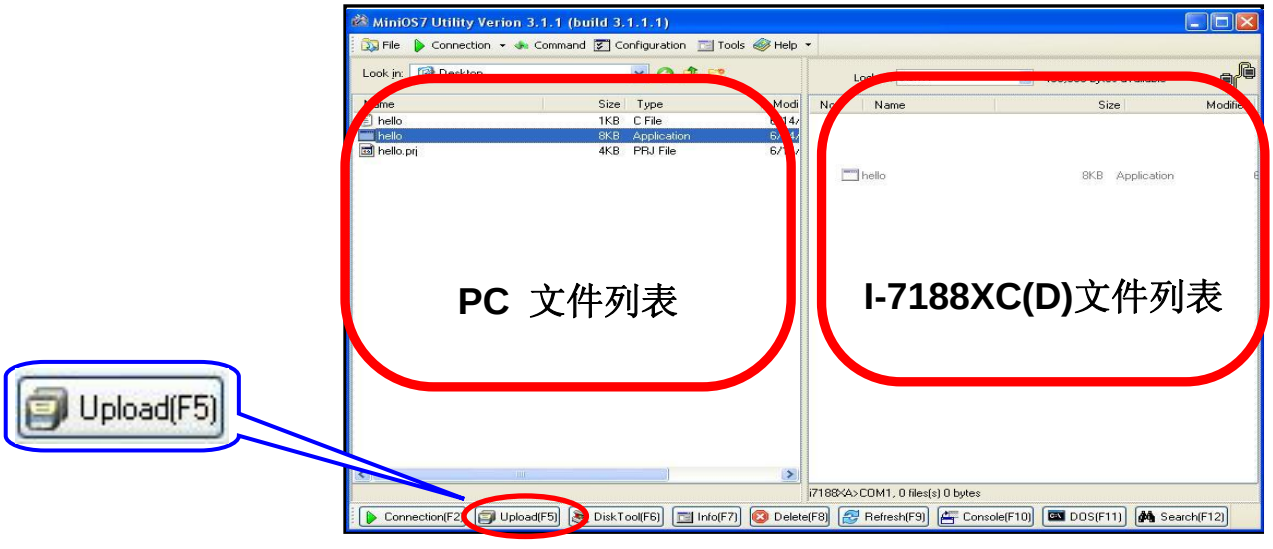


步骤 3: 可通过右上角图标得知 MiniOS7 Utility 是否连接到 I-7188XC，图标  为已连接；图标  为断开连接。

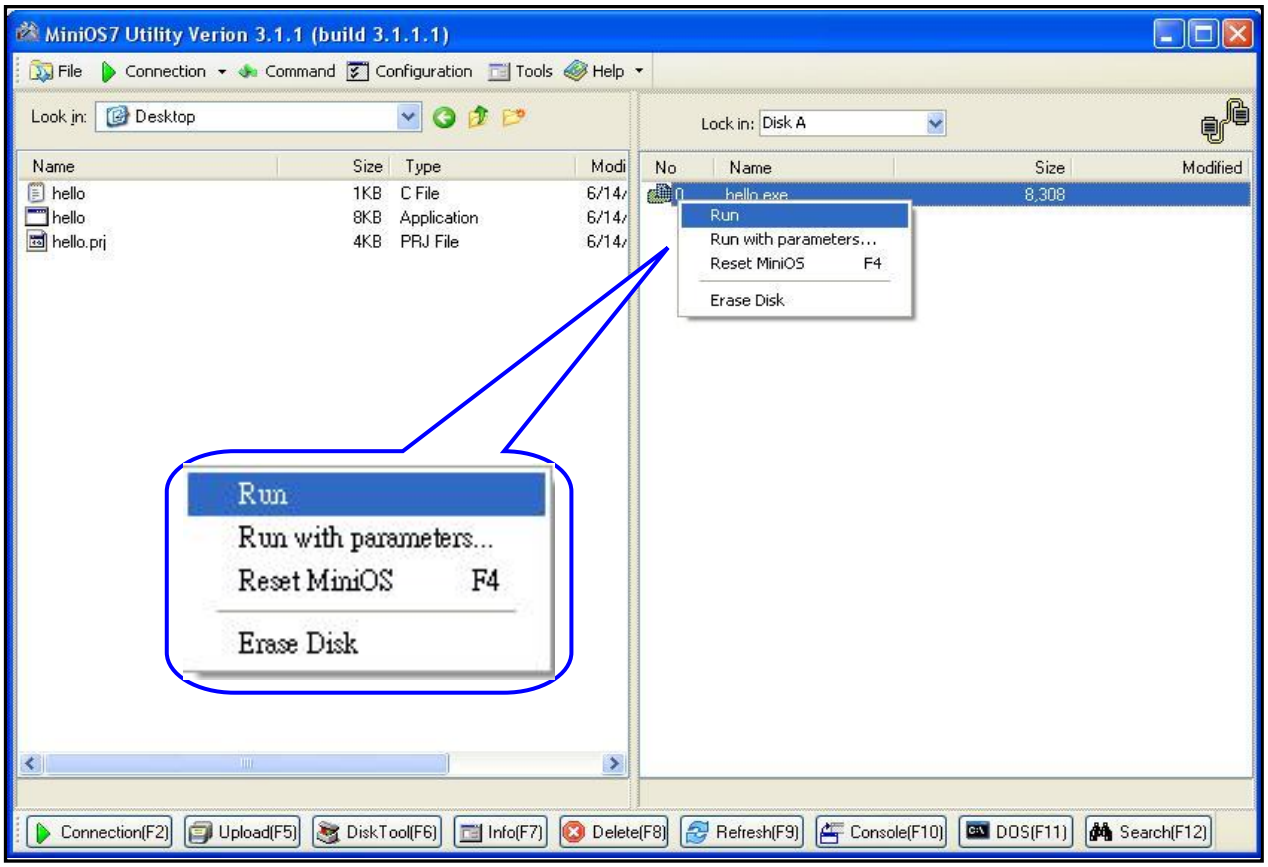


步骤 4: 从左边文件选择栏中选中文件，单击 

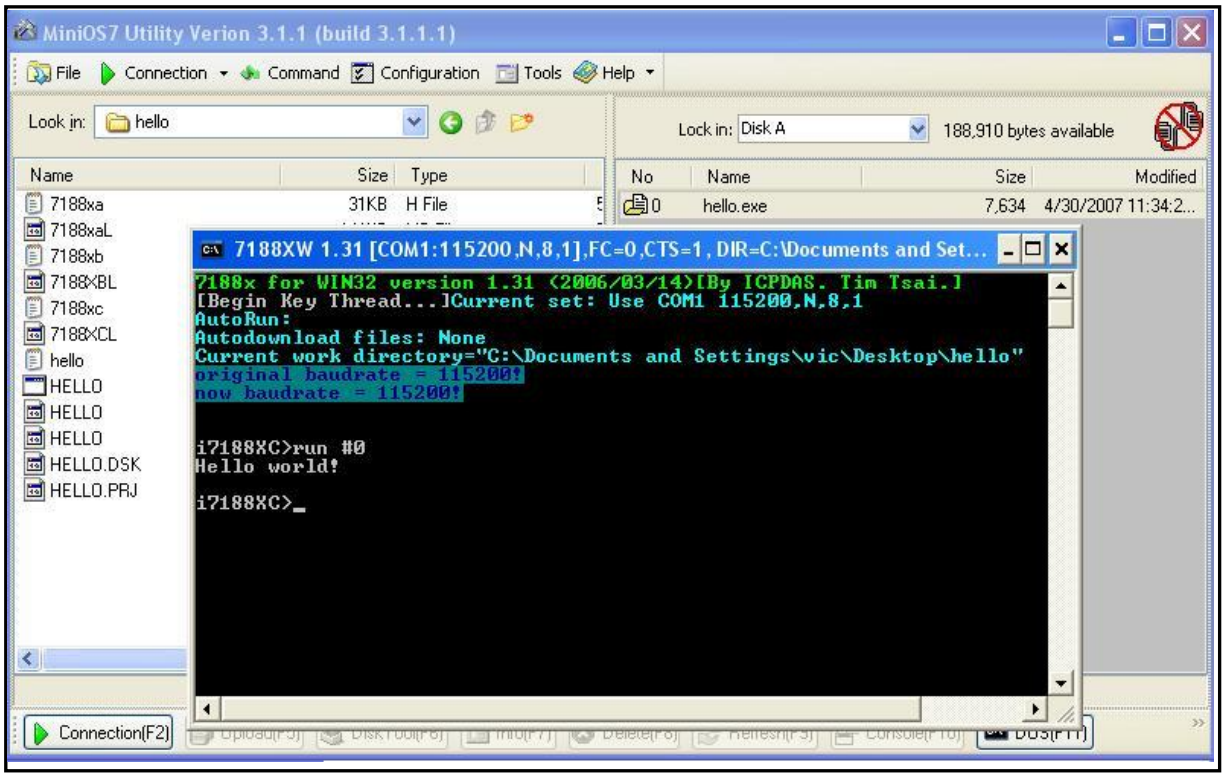
下载文件，下载成功后， 7188 文件栏即显示相关文件。



步骤 5: 选择文件并单击鼠标右键，选择 **Run** 执行程序。



步骤 6: 程序运行结果将在 7188xw 窗口中显示。



注意：当再次执行下载操作时，**7188xw** 窗口必须关闭。

Hello.c 程序代码如下：

```
#include “7188xc.h”       /* Include the headers to use 7188xcl.lib
                                functions */
void main(void)
{
   InitLib();                       /* Initiate the 7188xc library */

   Print("Hello world!\r\n");       /* Print the message on the screen */
}
```

2.4 MiniOS7 升级

泓格科技将持续更新 MiniOS7，以添加更多实用功能。因此，用户可定期访问泓格官方网站查询是否有最新版本 MiniOS7 发布。

注意：更多 MiniOS7 详细说明请参考**附录 A：什么是 MiniOS7**？

通过 MiniOS7 Utility 可方便地更新 MiniOS7 操作系统，具体方法参考如下：

步骤 1: 获得最新 MiniOS7 操作系统镜像文件

镜像名格式为：**TTYYMMDD.img**

TT: 产品类型

YY: 镜像发行年份

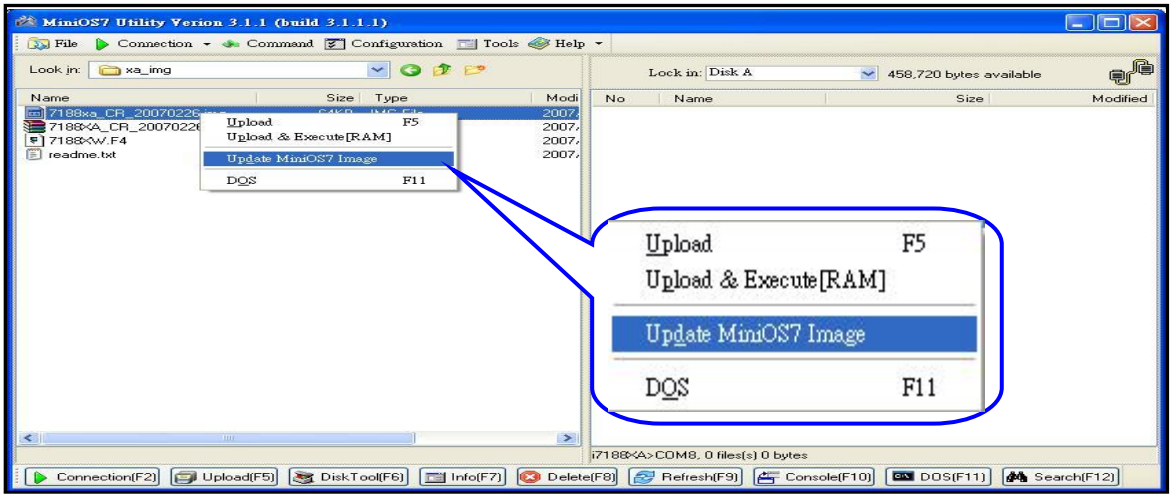
MM: 镜像发行月份

DD: 镜像发行日期

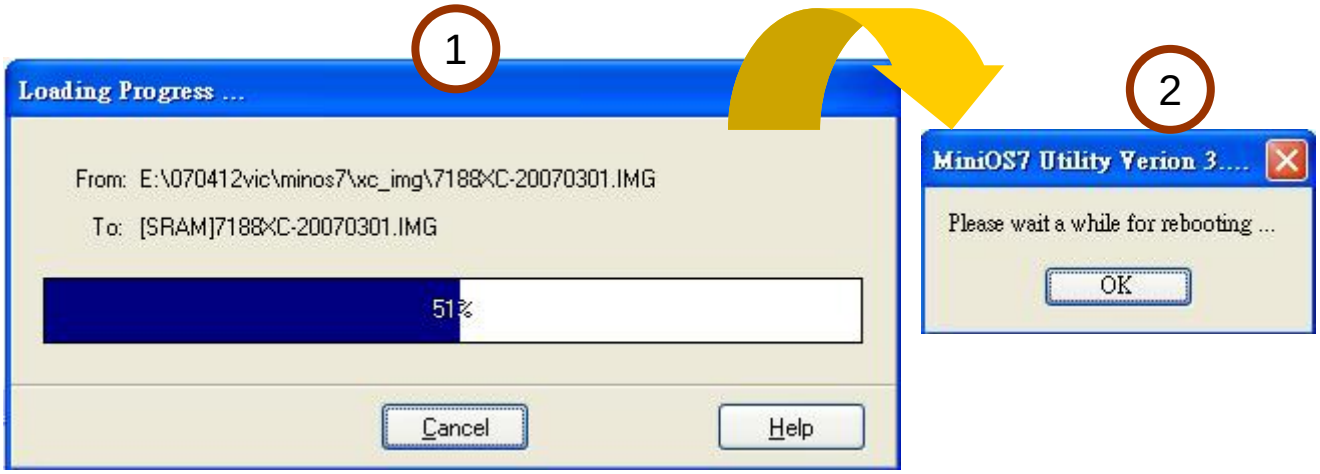
注意： MiniOS7 镜像文件可在随机赠送光盘中获得，文档地址为 CD:\NAPDOS\MiniOS7\. 最新 MiniOS7 可访问泓格官方网站获得，下载地址：

http://ftp.icpdas.com/pub/cd/8000cd/napdos/7188Xabc/7188XC/os_image/

步骤 2: 执行 MiniOS7 Utility. 参考章节 2.3 连接 I-7188，选中你需要更新的 MiniOS7 镜像文件，单击鼠标右键选“**Update MiniOS7 Image**”。

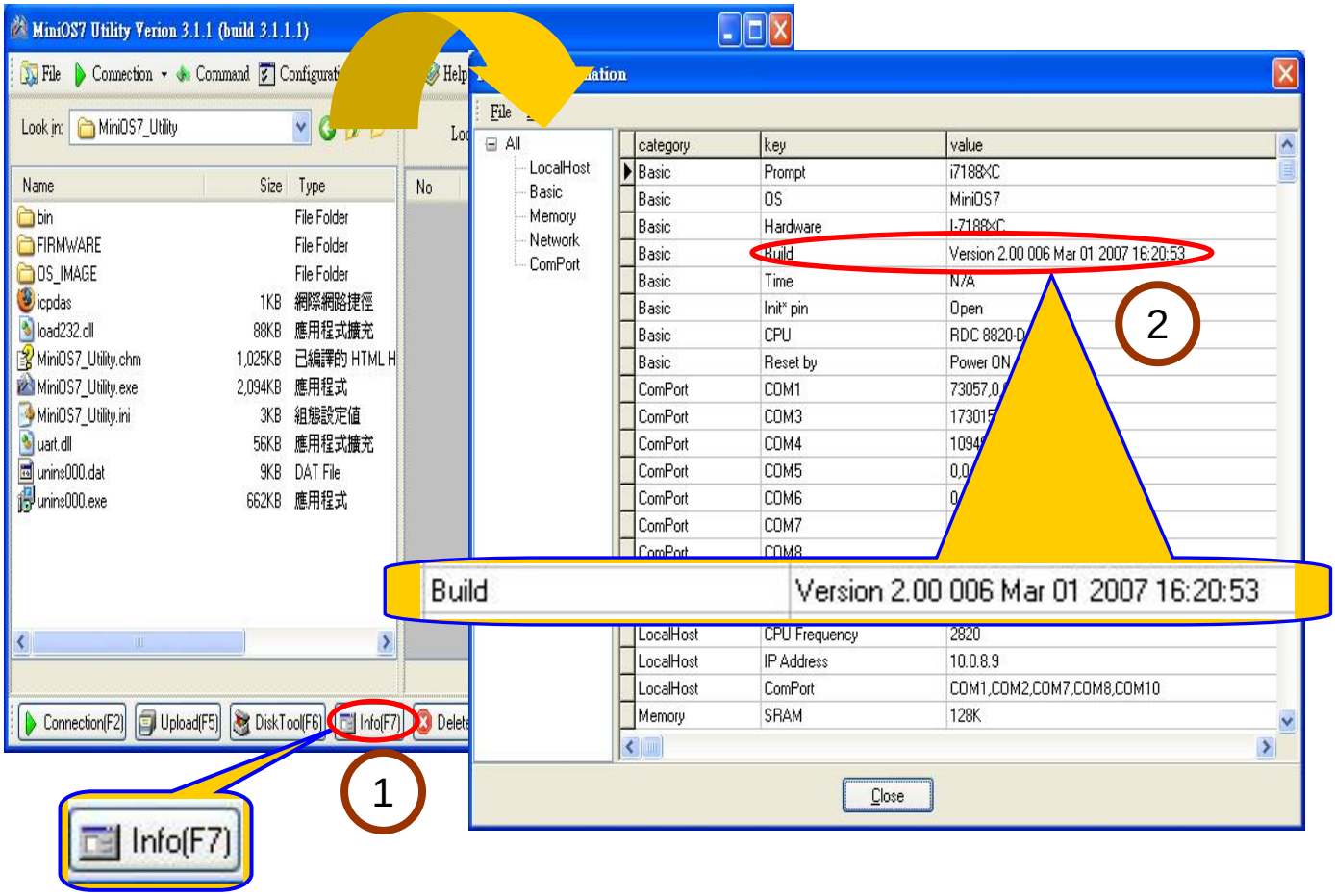


步骤 3: 全部更新过程大约需要 10 秒钟。若更新成功，则一个配置对话框将出现。



步骤 4: 点击 

，即可查有关 I-7188 “Build” 项目，检测 MiniOS7 版本号。参考如下：



注意：除使用 MiniOS7 Utility 更新 MiniOS7 操作系统外, 7188xw.exe 也可完成相关操作。详情请参考**附件 B: MiniOS7 Utility 和 7188XW**

3.编写第一个程序

3.1 库文件

I-7188XC(D) 拥有两个函数库文件：

- **7188xcs.lib** 适用于小内存模式
- **7188xcl.lib** 适用于大内存模式

两个库文件均可适用于 TC, BC++, MSC 和 MSVC++编译器。所有函数声明都在头文件 **7188xc.h** 中。

最新库文件地址：
http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/minios7_2.0/i-7188xc/lib/ 或 CD: \Napdos\MiniOS7\MiniOS7_2.0\i-7188xc\lib

7188xcs.lib/7188xcl.lib 支持上百条函数：

函数说明	例如
COM port	InstallCOM0, InstallCOM1, InstallCOM2 …… IsCOM0, IsCOM1, IsCOM2 …… ToCOM0, ToCOM1, ToCOM2 …… ReadCom0, ReadCom1, ReadCom2 ……
EEPROM	WriteEEP, ReadEEP, EnableEEP, ProtectEEP
LED and 5-digit LED	LedOn, LedOff, Init5DigitLed, Show5DigitLedWithDot
Flash Memory	FlashReadId, FlashErase, FlashRead, FlashWrite ……
Timer and Watchdog Timer	TimerOpen, TimeClose, TimerResetVlaue, TimerReadValue, StopWatchReset, StopWatchRead, StopWatchStop
File	GetFileNo, GetFileName, GetFilePositionByNo, GetFilePositionByName
Connect to 7000 series modules	SendCmdTo7000, ReceiveResponseFrom7000
Programmable I/O	SetDio4Dir, SetDio4High, SetDio4Low, GetDio4
Others	Kbhit, Getch, Putch, LineInput, Scanf ……

注意：更多有关函数详情请参考**附件 D: 库函数清单**

3.2 编译及链接

C 语言编译常常用来开发各式应用程序，其中有效的编译器包含有：

- BC++ 3.1~5.02
- TC++ 1.01
- TC 2.01
- MSC
- MSVC++ (版本 1.52 或更早).

建议客户使用 Borland C++ 3.1 编译器，
本产品随货光盘提供的 Library 也是由此编译器编译完成，使用编译器开发应用程序之前，有下列几点特别注意事项：

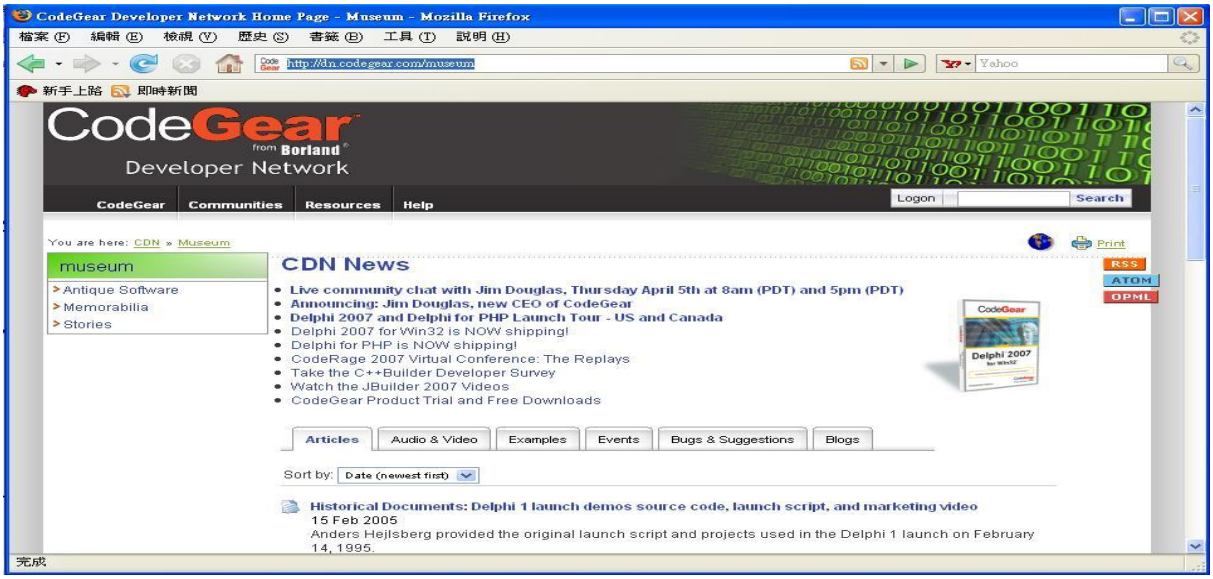
- 选择生成标准的 DOS 可执行程序.
- 设置 CPU 选型为 80188/80186
- 如有浮点运算需求，设置浮点类型为 EMULATION(注意不要设定为 8087)
-
- 取消调试信息功能以降低程序空间大小(MiniOS7 支持该功能)

3.3 程序编写详细步骤

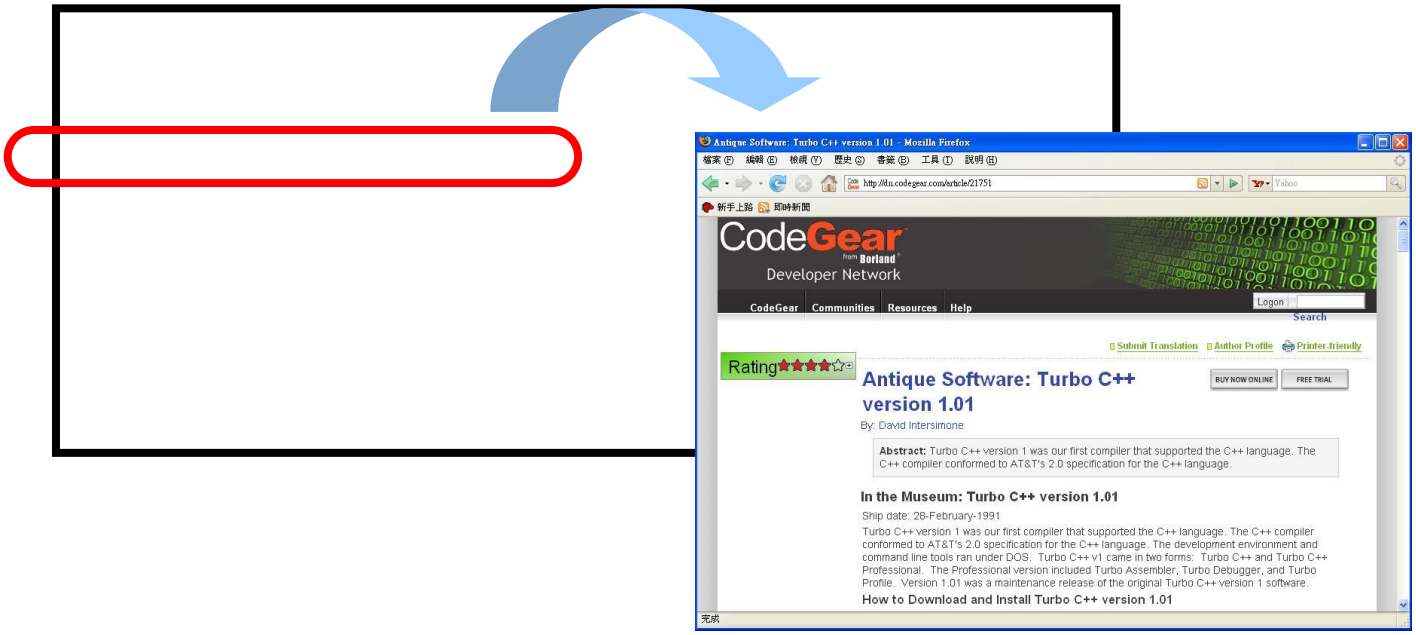
3.3.1 下载 Turbo C++ version 1.01

免费版本的 Turbo C 2.01 和 Turbo C++ 1.01 编译器可从 Borland 公司网页下载。参考如下步骤在 Windows 操作系统中安装 Turbo C++ 1.01：

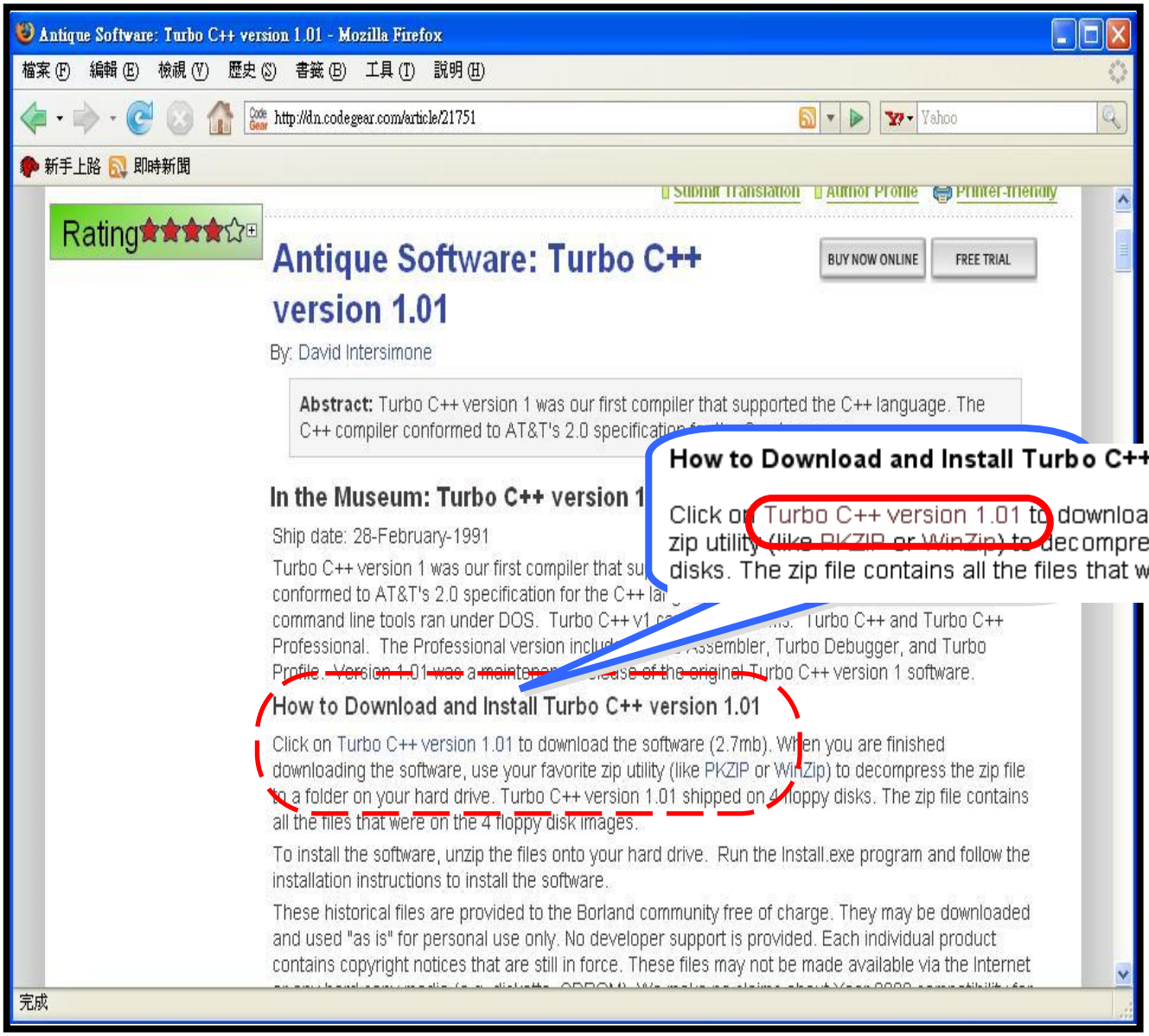
步骤 1: 访问 CodeGear 公司网站(http://dn.codegear.com/museum).



步骤 2: 点击链接 **Antique Software: Turbo C++ version 1.01** 进入下载页：



步骤 3: 点击连接 **Turbo C++ version 1.01**, ， 并下载文件 **tcpp101.zip** 到本地硬盘上。

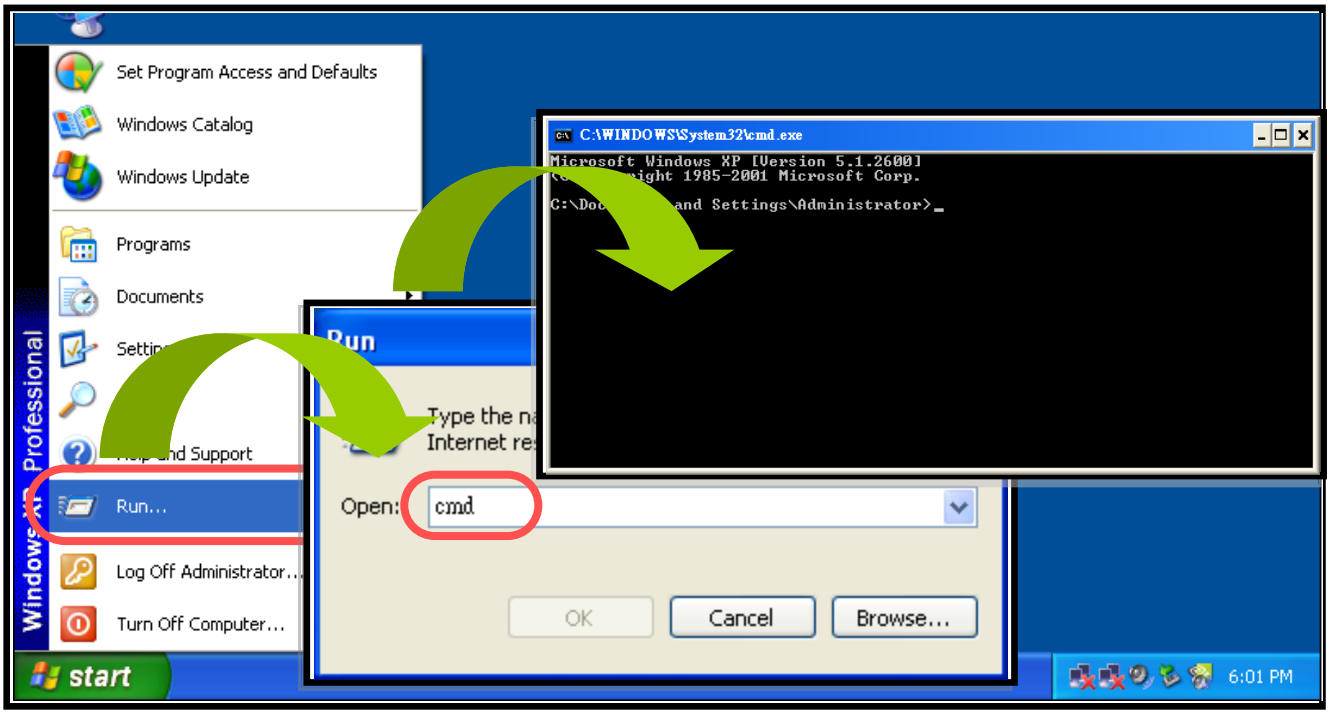


3.3.2 安装 Turbo C++ version 1.01

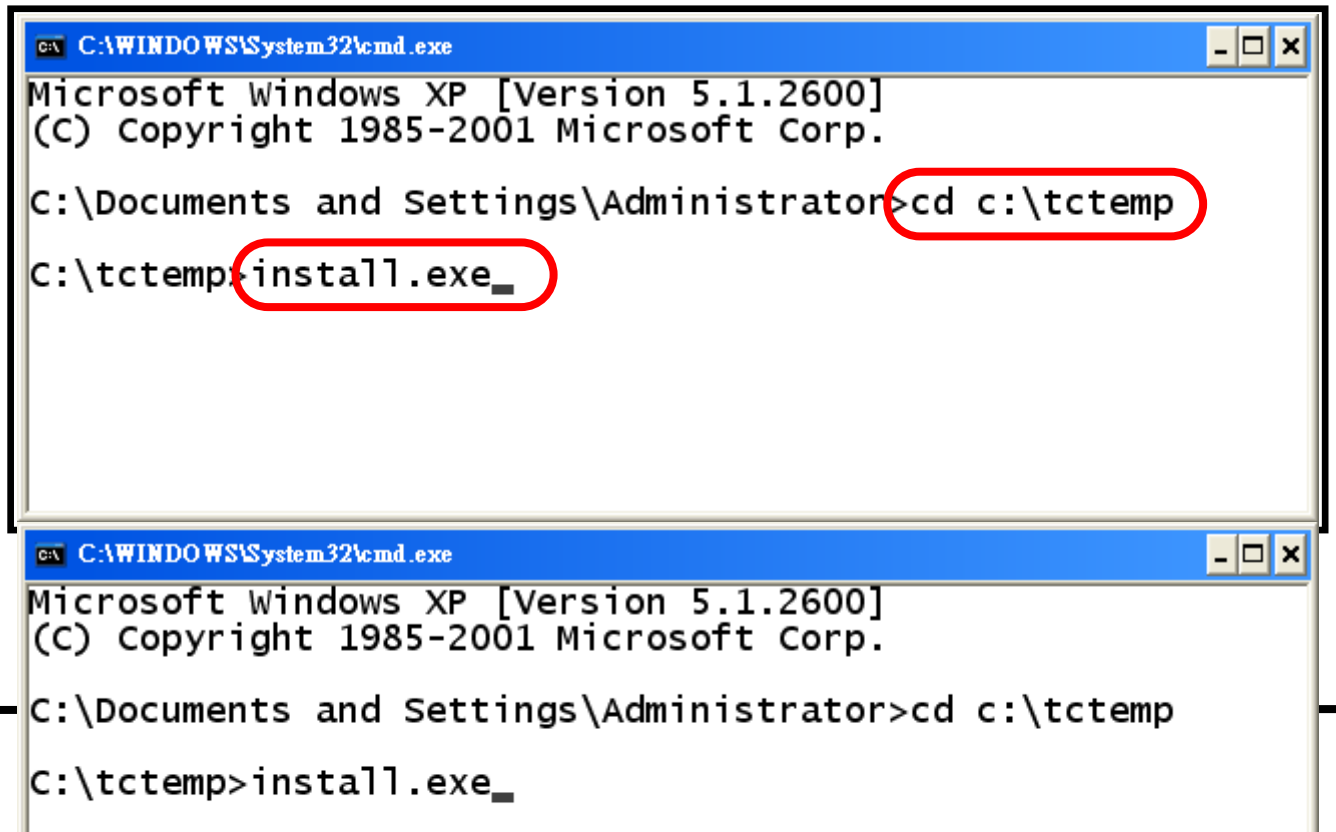
步骤 1:解压文件到本地硬盘。默认解压地址为 C:\tctemp。

步骤 2:解压后，退出解压软件窗口。

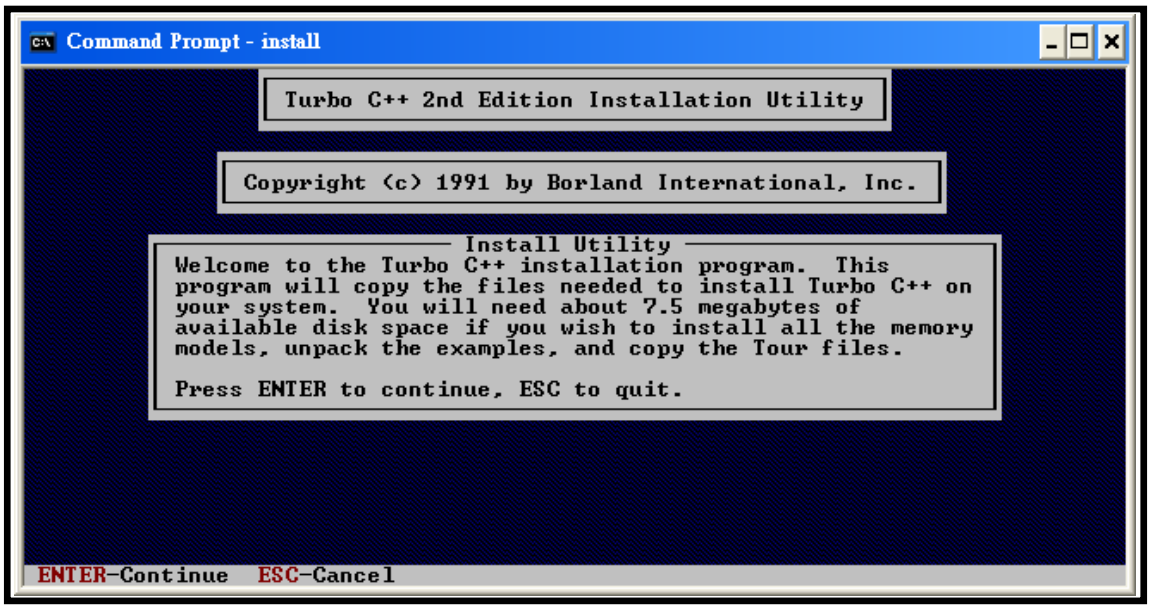
步骤 3:进入 MS-DOS 命令提示符窗口。



步骤 4:改变目录到 **c:\tctemp** (或解压文件地址)，并执行 **INSTALL.EXE** 安装文件。



步骤 5:依照安装向导安装软件。



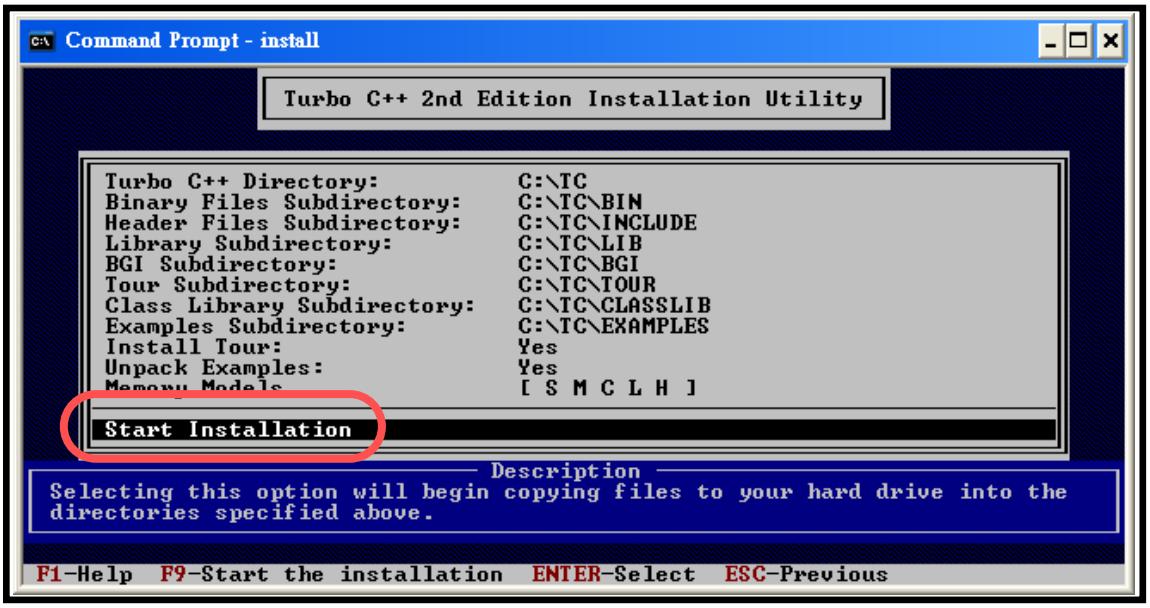
步骤 5.1: 点击 **<ENTER>**开始安装

步骤 5.2: 选择解压文件后的**驱动器**。默认为 “A”，因此输入 **“C”**,然后点击**<ENTER>**

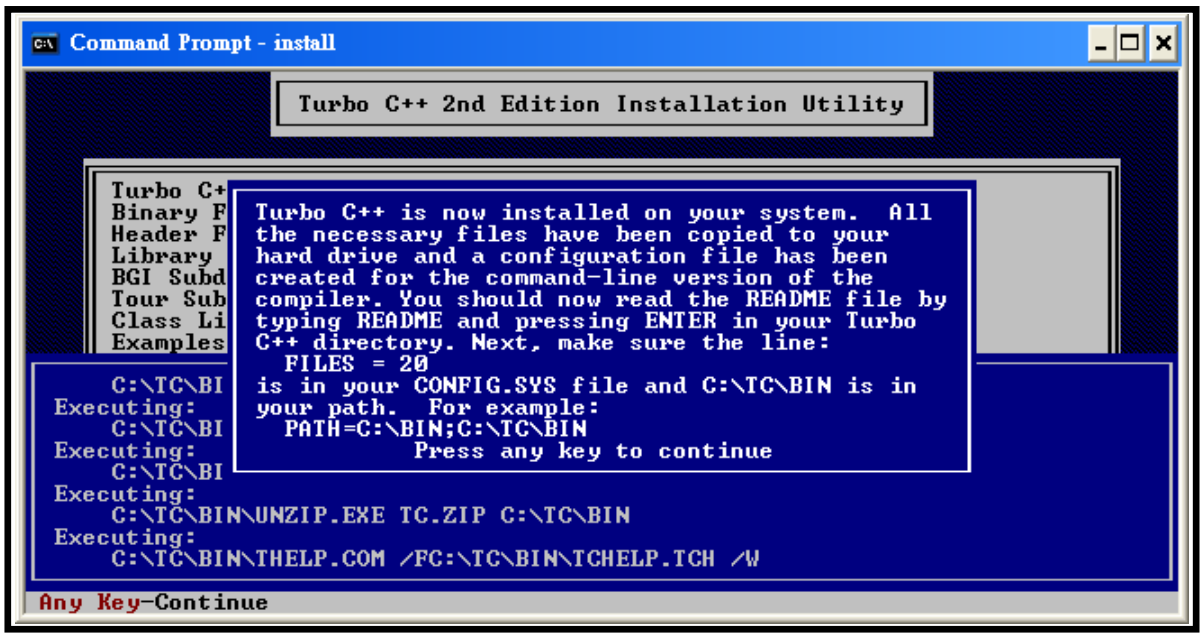
步骤 5.3: 点击**<ENTER>**，将从文件夹\tctemp 中安装软件

步骤 5.4: 再次点击**<ENTER>**，将 Turbo C 安装于硬盘上

步骤 5.5: 使用方向键 Up/Down 把选择 **Start Installation**，并再次点击 **<ENTER>**



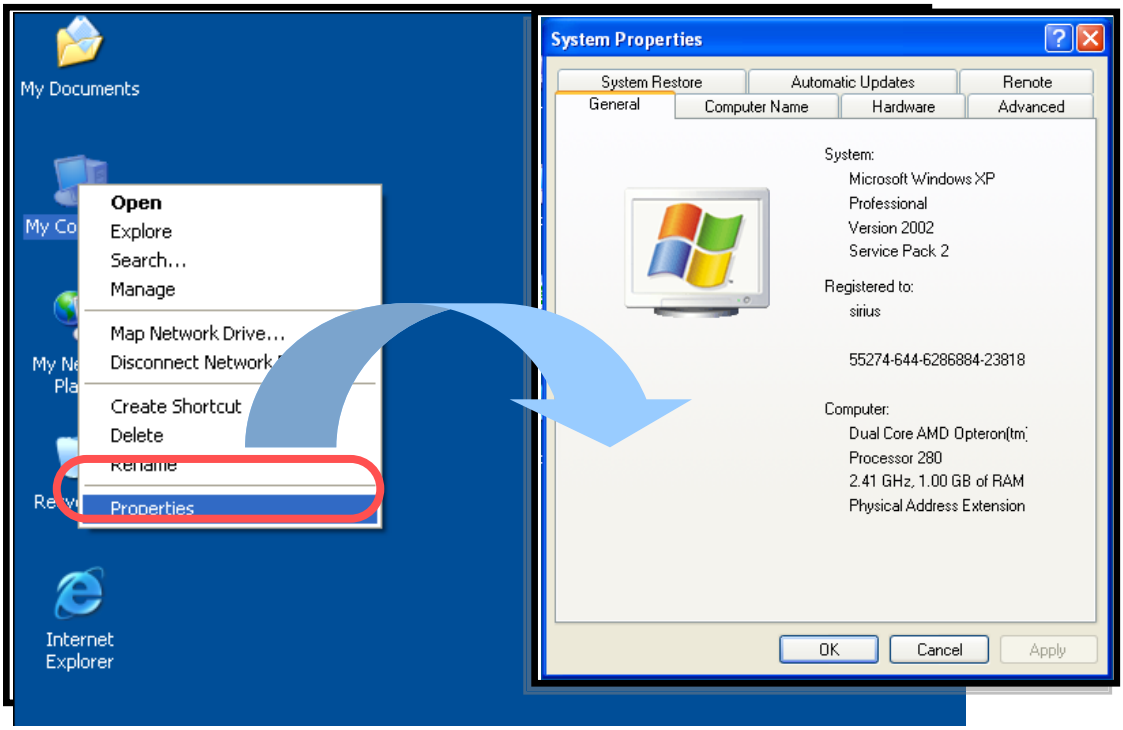
步骤 5.6: 此时，Turbo C++ version 1.01 编译器安装于 C:\TC，可自执行文件 tcc.exe 同时也在本地。



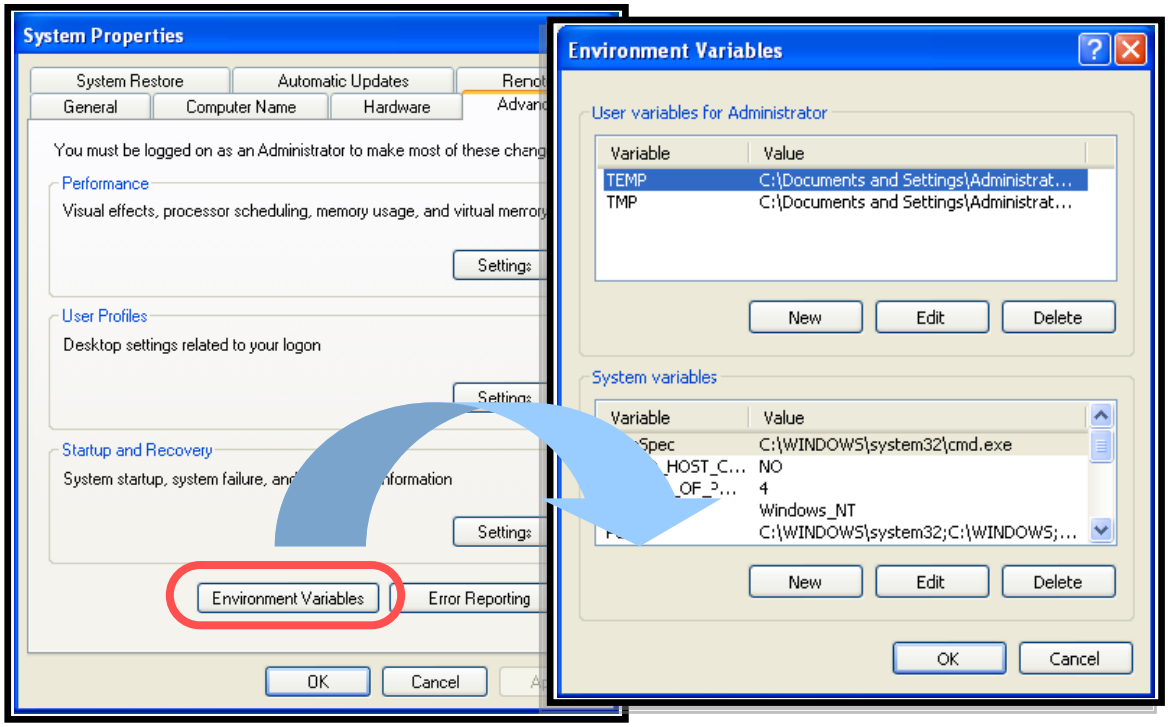
3.3.3 设置系统环境变量

安装后，可以把 C:\TC 添加到系统环境变量中,这样一来，在命令行输入简单名称，即可执行编译器，添加方法如下：

步骤 1: 右键单击桌面上**我的电脑**图标，并选择**属性**。

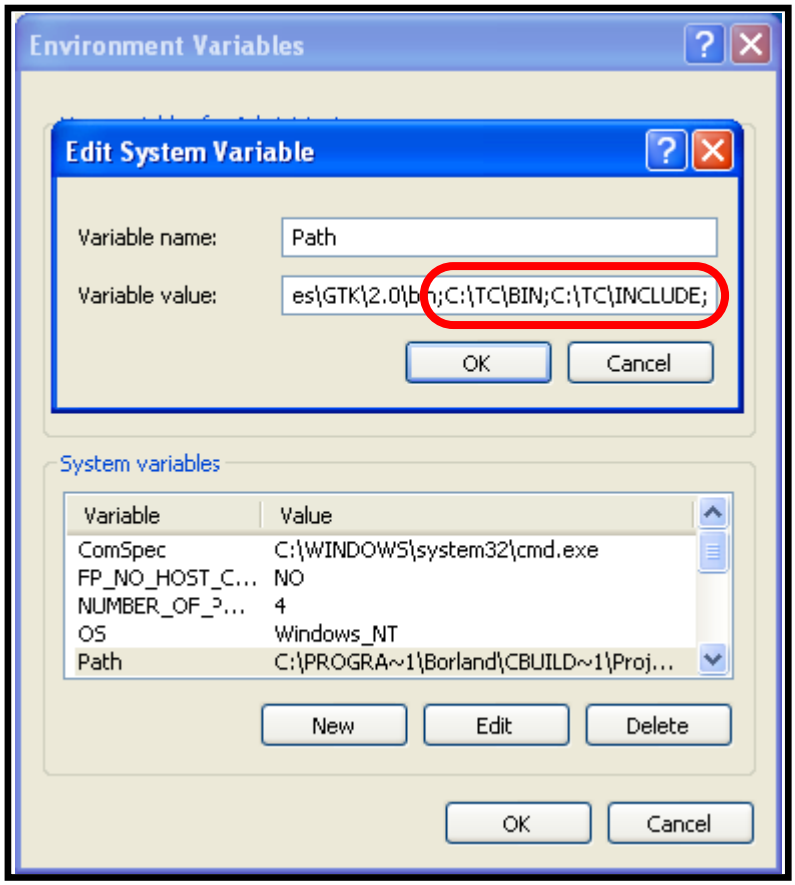


步骤 2: 选择**高级**标签，单击**环境变量**按钮。



步驟 3: 在**系统变量**中，选择变量 **Path** 并单击**编辑**。

步驟 4: 在**变量值**末尾处用分号隔离后添加路径。例如：
”**C:\TC\BIN;C:\TC\INCLUDE;**”.



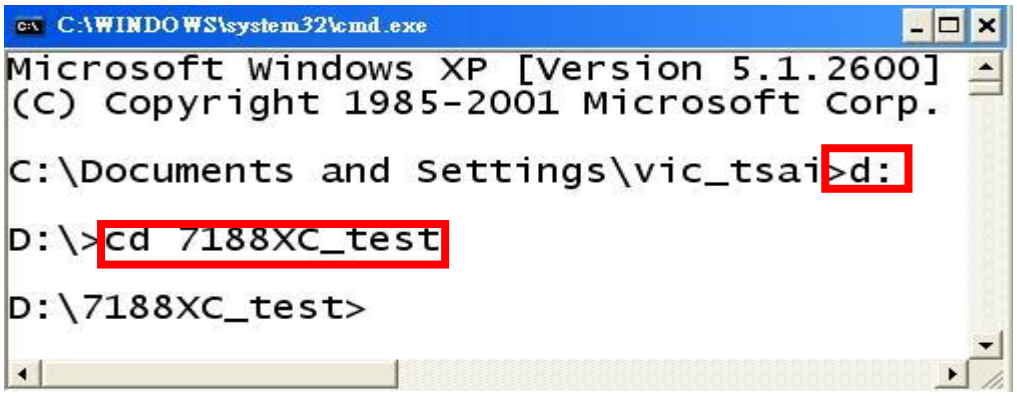
步驟 5: 单击**确定**按钮，并重启计算机以让设定生效。

3.3.4 编译及执行程序

步骤 1: 打开 MS-DOS 命令提示符窗口

注意: 开启新窗口前，必须先将已打开的 MS-DOS 命令提示符窗口关闭。

步骤 2: 输入“d:”后按<**Enter**>进入 D 盘
步骤 3: 输入“cd 7188XC_test” 后按<**Enter**>

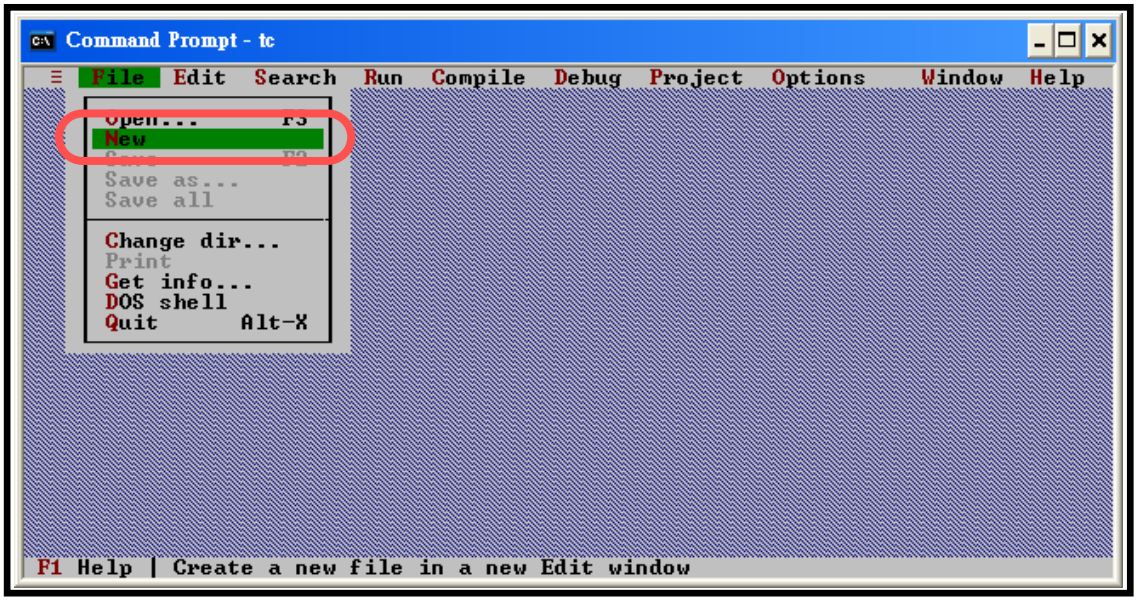


注意: 假定存在文件夹 7188XC_test 建立在 d:\. 并在文件夹 7188XC_test 包含有 7188xc.h 和 7188xcl.lib。

步骤 4: 输入 **tc** 后单击 <**ENTER**>运行 TC++ 1.01 编译环境。该命令可从任务路径下执行。

步骤 5:创建源文件(*.c).

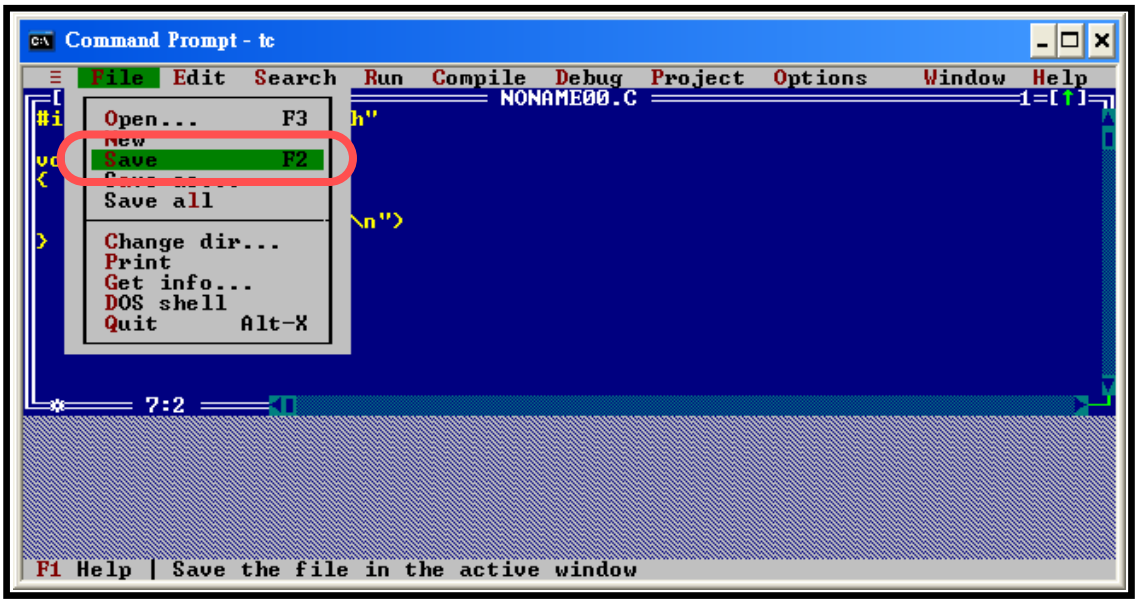
步骤 5.1: 从 File 菜单选择 **New**。



步骤 5.2: 输入如下代码，注意区分代码大小写。

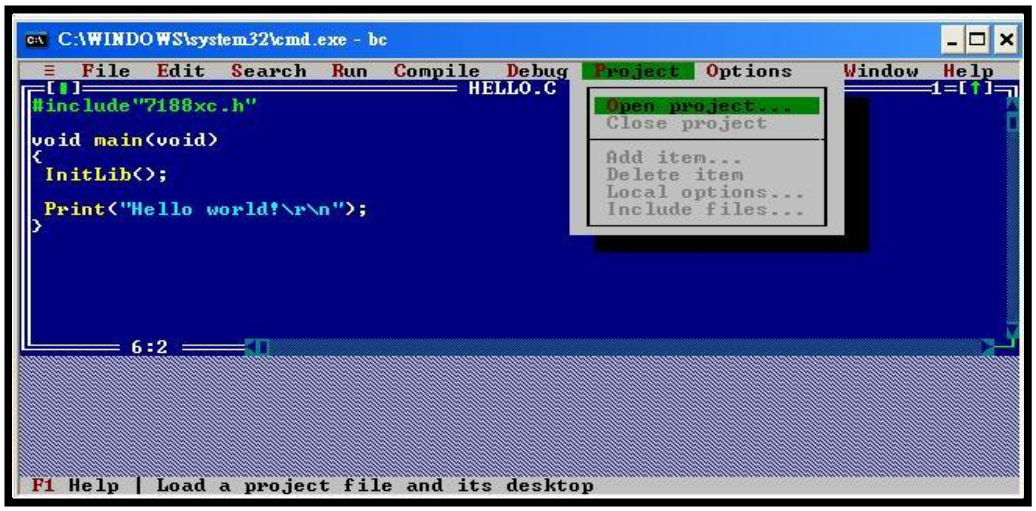
```
#include “7188xc.h”

void main(void)
{
   InitLib();

   Print("Hello world!\r\n");
}
```

5.3: 从 File 菜单中选择 **Save**，然后输入文件名 **Hello.C**。

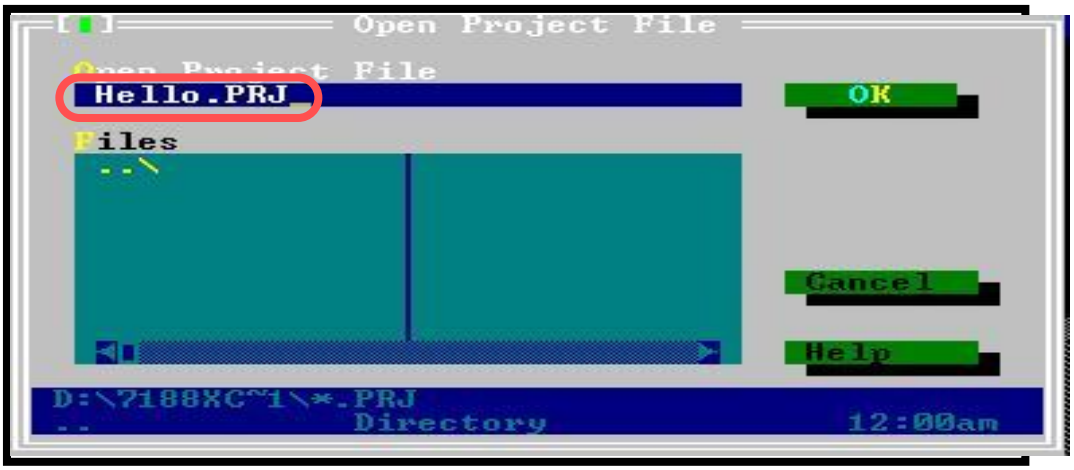


注意: 若您有任何熟悉的文字编辑器，可先行将上面代码输入进去，但要注意代码不可用诸如 WORD 这样的文字处理软件，使用文字编辑器必须是保存为纯文本类型，如 Windows 自带的记事本等。C 语言程序文件后缀必须为“**.C**”。

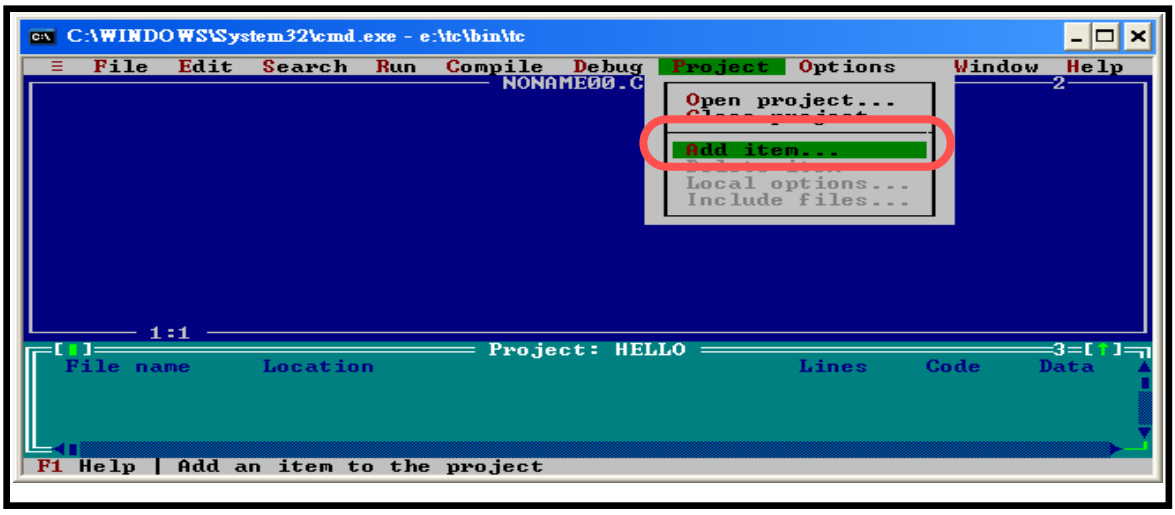
步骤 6: 创建一个新的工程文件(*.prj).



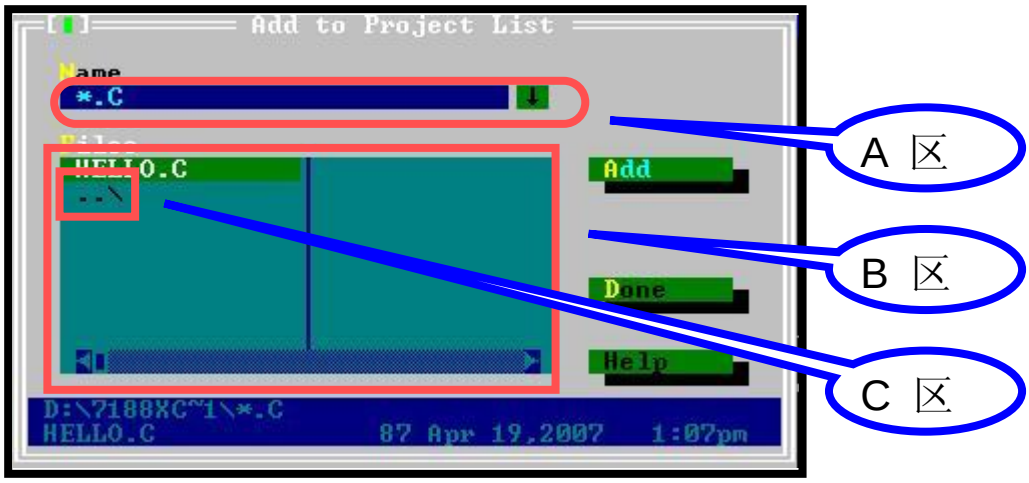
步骤 6.1: 输入工程文件名称然后点击 **OK**。



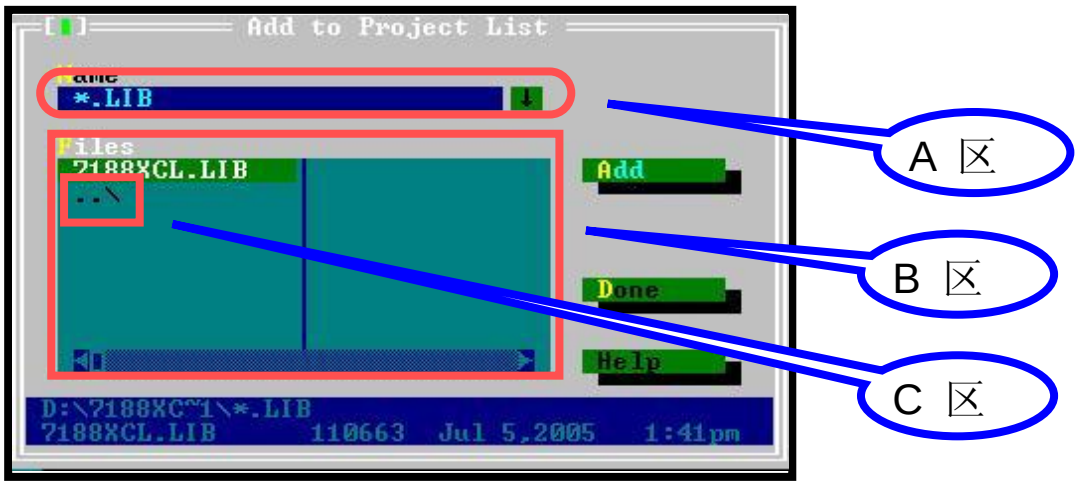
步骤 7: 加入工程所需要的文件。



步骤 7.1: 选择程序源文件，输入 “*.c” 在区域 A 点击 **Enter**。若所需文件在 B 区域中，移动绿色滑条选定文件，单击按钮 **Add**；若没有,则移动 C 区域中绿色滑条，重新选择文件。

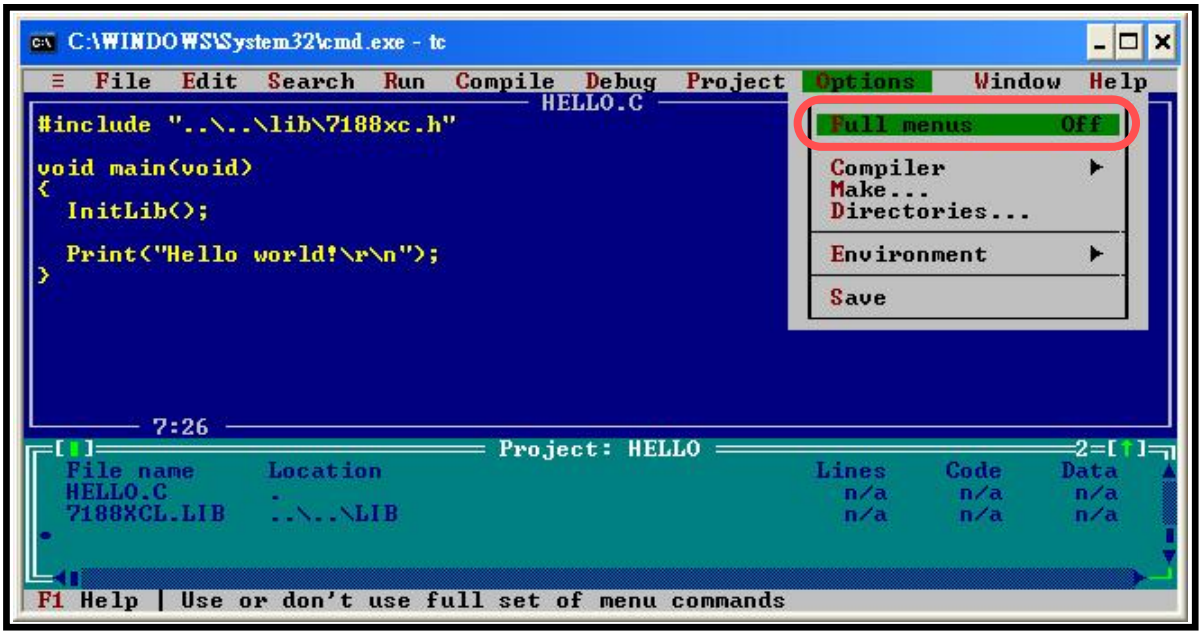


步骤 7.2: 选择函数库。输入“*.lib” 在区域 A 点击 **Enter**。若所需文件在 B 区域中，移动绿色滑条选定文件，单击按钮 **Add**；若没有,则移动 C 区域中绿色滑条，重新选择文件。

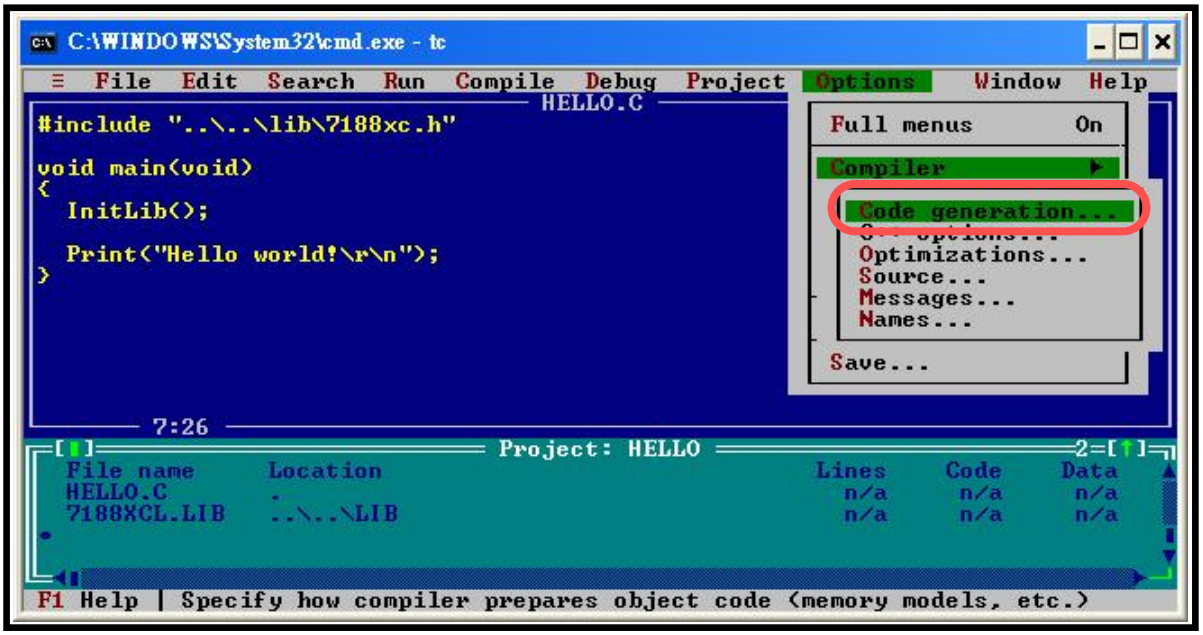


步骤 8: 点击 **Done** 确定并退出窗口

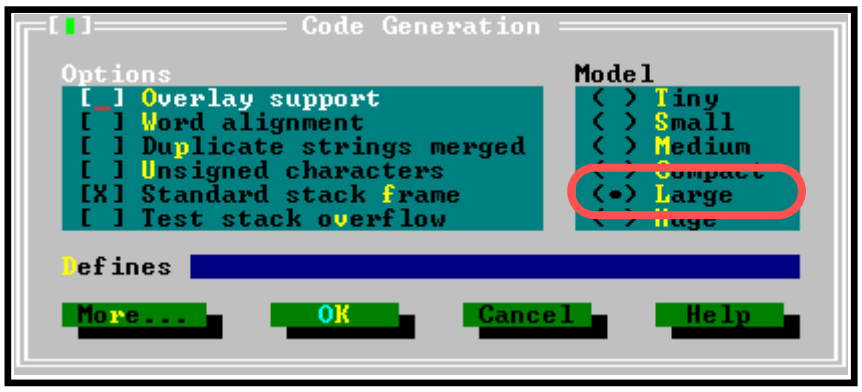
步骤 9: 点击 “**Options**”选择 **full menus**



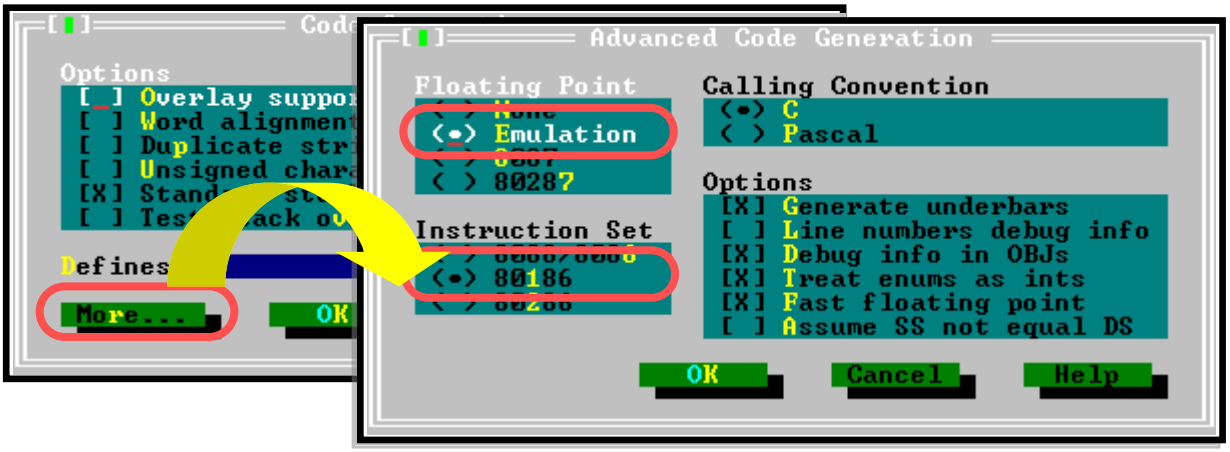
步骤 10: 点击“**Options**”选择 **compile** 菜单项目，再选择 **Code generation options**.



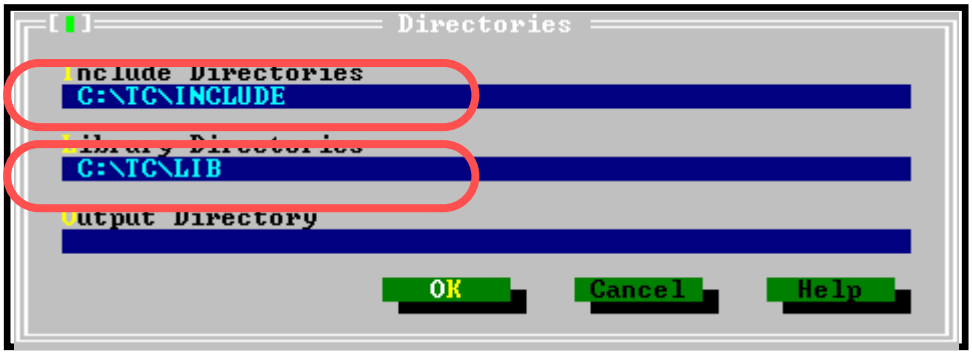
步骤 10.1: 改变内存模式 (Small 适用于 7188xcs.lib, large 适用于 7188xcl.lib).



步骤 10.2: 点击“**More…**”, 设置 Floating Point 为 **Emulation**, **Instruction Set** 为 **80186**.

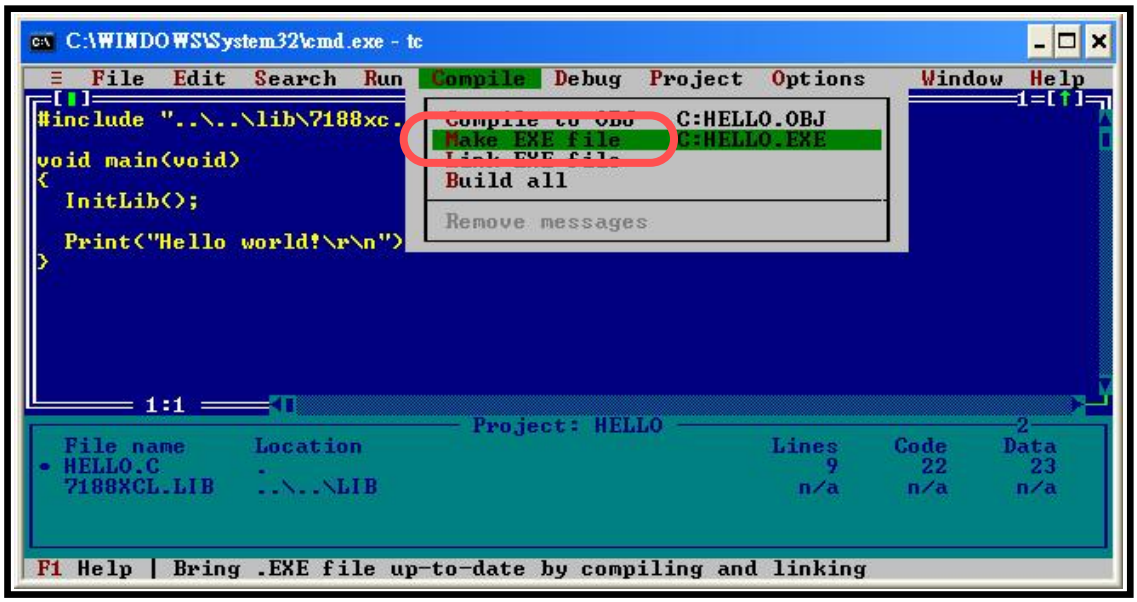


步骤 11: 单击“**Options**” 选择“**Directories…**” 进入 TC++ 1.01 include 及 library 目录路径。默认设置为 TC++ 1.01 安装文件夹。

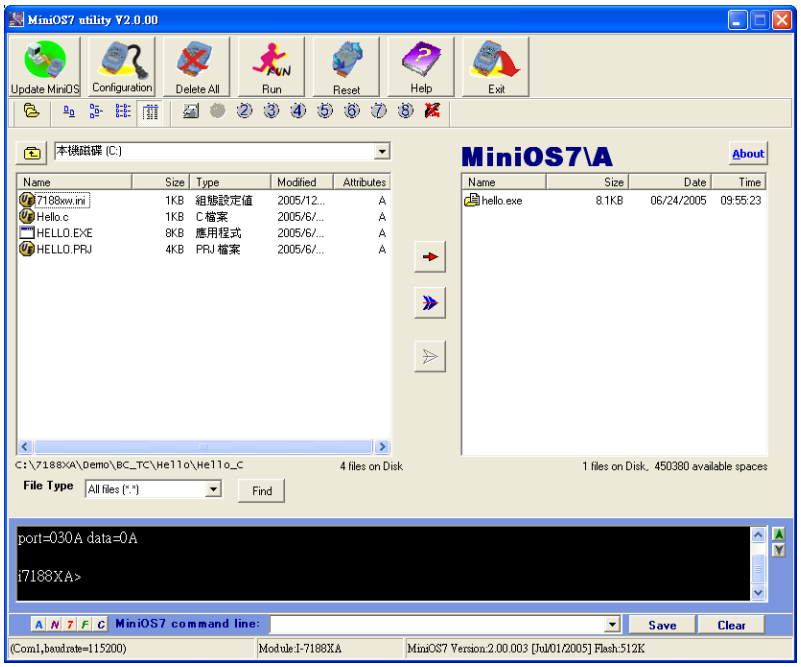


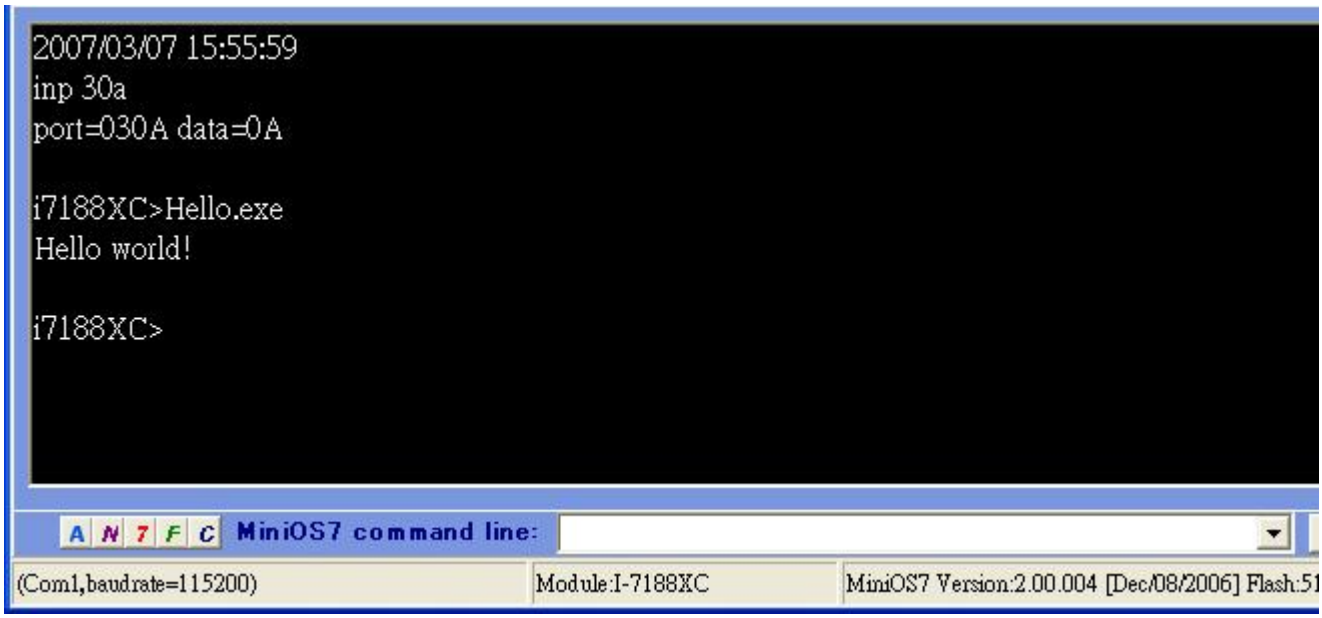
注意: Include Directories 特指包含有标准 include 文件目录; Library Directories 特指包含 TC++ 1.01 启动目标文件及 run-time 库文件。

步骤 12: 单击“**Compile**” 选择“**Make EXE file**”生成工程文件。



有关下载与执行应用程序，详情请参考章节 2.3.





有关各种不同 C 语言编译器(TC/BC/MSC/MSVC)详情编译和链接说明请参考**附录 E: 编译和链接**

3.4 在 64 位平台建立工程(Project)

假如要在 Windows 7 或 Windows 8 这种有支持 64 位的 OS 平台上使用 16 位的编译器(像是 BC3.1 或是 TC++3.0)来创建 MiniOS7 工程时，将会告知有 64 位平台兼容性问题信息出现。

请参阅**附录 E: 编译和链接** 的子章节“在 64 位平台编译”.

4.操作原理

4.1 系统映射

设备	地址映射
Flash ROM	256K: C000:0000 ~ F000:FFFF
SRAM	128K: 0000:0000 ~ 1000:FFFF
COM1 BASE	0XFF80 ~ 0XFF88
COM2 BASE	0XFF10 ~ 0XFF18

中断号	中断映射
0	Divided by zero
1	Trace
2	NMI
3	Break point
4	Detected overflow exception
5	Array bounds exception
6	Unused opcode exception
7	ESC opcode exception
8	Timer 0
9	Reserved
0A	DMA-0
0B	DMA-1
0C	\INT0 of the I/O expansion bus
0D	\INT1 of the I/O expansion bus
10	Reserved
11	COM2
12	Timer 1
13	Timer 2
14	COM1

4.2 程序调试串口 COM1

I-7188XC(D)的串口 COM1 (程序下载口) 有三大功能：

- 从主机 PC 下载程序
- 与主机 PC 相连调试程序
- 作为通用串口

当 I-7188XC(D)拨动开关开启,将在控制台模式下初始化 COM1 为如下标准配置：

- **Start Bit=1, Data Bit=8, Stop Bit=1, no parity**
- **Baud Rate=115200 bps**

I-7188XC(D)将自行检测引脚 INIT*状态。**若引脚 INIT*与 GND 短接**, 则 I-7188XC(D) 将发送启动信息， COM1 将进入**控制台模式，**允许用户下载/调试程序, 通过以下步骤可显示启动信息：

- I-7188XC(D)及主机 PC 断电
- 使用串口线连接 I-7188XC(D)的 COM1 与主机 PC 串口 (详情请参考章节 2.2)
- 启动主机 PC 并执行 7188xw.exe
- 将 I-7188XC(D)上电
- 所有初始化信息将在主机 PC 上显示

若引脚 INIT* 悬空, I-7188XC(D)将会自动搜索 **autoexec.bat** 文件，若文件 autoexec.bat 存在, I-7188XC(D)将执行该文件； 若文件 autoexec.bat 不存在, 则 I-7188XC(D)将返回控制台模式，以允许用户下载/调试程序。

当初始化阶段完成后，I-7188XC(D)的串口 COM1 将成为标准的输入/输出接口。I-7188XC(D)标准输出将在主机 PC 的显示器上显示，如果在主机 PC 上输入的任何按键信息,按键信息代码将做为 I-7188XC(D)标准输入信息。因此主机 PC 的显示屏及键盘可视为 I-7188XC(D) 标准输出/输入设备：

- 使用 7188xw.exe 或 MiniOS7 Utility 作为连接 I-7188XC(D)与主机 PC 的桥梁。
- 在主机 PC 上执行 7188xw.exe 或 MiniOS7 Utility 以建立桥梁。

- 主机 PC 键盘 → I-7188XC(D)标准输入设备
- 主机 PC 显示器 → I-7188XC(D)标准输出设备

通过该方式，I-7188XC(D) 可从键盘读取数据及在显示器上显示数据。因而，可以方便而有效的对程序进行调试。

注意: 7188xw.exe 和 MiniOS7 Utility 可在随机赠送的 CD 获得。有关接线信息及如何下载程序，详情请参考章节 2.2 及章节 2.3 。

4.3 下载端口作为通用串口使用

I-7188XC(D)下载端口(COM1)也可作为通用串口来使用，方法如下：
步骤 1: 首先下载应用程序及 autoexec.bat 到 I-7188XC(D)中。
步骤 2: 断开 I-7188XC(D)电源，将下载线从主机 PC 上移除。
步骤 3: 若引脚 INT* 与 GND 已短接，将其断开。
步骤 4: 给 I-7188XC(D) (无标准输入输出及调试信息)供电。
步骤 5: 将 I-7188XC(D)的串口 COM1 与其它 RS-232 设备相连。
步骤 6: 以新的配置初始化 COM1 。
步骤 7: 这样，I-7188XC(D) 的 COM1 可即作为 RS-232 接口来应用。

4.4 功能及范例程序列表

我们为 I-7188XC(D)设计有大量范例程序以供使用。更多有关例程详细信息请参考下一章。例程功能列表如下：

文件夹	范例程序	说明	章节
Hello	Hello_C	适用于操作系统为 MiniOS7. **注意：** MSC 并不支持 C++语言。Hello_C++ 仅适用于 BC.	3.3.4
	Hello_C++		
COM_Port	C_Style_IO	1. 如何写代码完成数据输入 2. 如何得到数据字符串 3. 如何使用 C 语言函数: sscanf, or just use Scanf().	4.6
	Receive	从串口接收数据	
	Slv_COM	从 PC 端发送命令给 I-7188XC(D),并接收 I-7188XC(D)反馈信号，同时具有 LED 功能，以帮忙客户调试程序。	
	ToCom_In_Out	从串口读/写数据	
IO_PIN		读/写 I-7188XC(D)的 DO 及 DI 数据	4.11
LED	Led	如何使用 DelayMs 函数调节 LED 状态 ON 或 OFF	4.7
	Seg7led	控制红色 LED 及 5 位 7 段 LED.	
File	Config_1_Basic	在许多应用中，针对一些关键数据常常存储在文本里，因而程序往往需要读取其中数据. FSeek 可用于检索文本中关键信息。	
	Config_2_Advanced	除去 config_1_Basic 之外,同时还加入了函数 GetProFileInt, GetProFileFloat 及 GetProFileStr.。这些函数可确定文本文件类型。	
	EEPROM	向 EEPROM 写入数据并显示到屏幕上。	
	EEPROM-r	读取已经写入 EEPROM 中数据	
	EEPROM-w	输入一值，并存储到 EEPROM 中 block 1 每一地址中 (数值将以 1 累加).	
	Flash	读,写和清除 Flash 存储器	
	Flash-r	读取已经写入 Flash 存储器中数据	
	Flash-w	输入数据并写入 Flash 存储器中(数值将以 1 累加).	
	Demo5	如何访问 NVRAM.	
	NVRAM-r	从 NVRAM 读取已写入数据	

	NVRAM-w	向 NVRAM 中写入数据 (数值将以 1 累加).	
	Top-Mem	AllocateTopMemory 函数使用例程	
Misc	Reset	复位初始值.	
	Runprog	使用 Ungetch 函数运行另外一个程序	
	Watchdog	激活 Watchdog 或绕过激活 Watchdog.	4.9
7K87K_Module	7K87K_demo_for_com	如何通过 COM2 连接 7K 及 87K 系列模块	4.6.3
	7K87K_AI_for_Com		
	7K87K_DI_for_Com		
	7K87K_DIO_for_Com		
	7k87K_DO_for_Com		
	AO_024_for_Com		
	AO_22_26_for_Com		
Timer	Demo90	Timer 函数使用例程.	4.10
	Demo91	利用 CountDownTimer 函数调节 LED 状态 ON 或 OFF.	
	Demo92	利用 StopWatch 函数调节 LED 状态 ON 或 OFF.	
	Demo96	使用 InstallUserTimer 函数控制 5 位 7 段 LED.	
	Demo97	使用 DelayMs 函数调节 LED 状态 ON 或 OFF.	
	Demo98	如何利用 timer 函数通过 I-7188XC(D) 发送/接收 7000 系列模块数据.	
XBoard		该项目包含所有 I/O 应用于 I-7188XC(D).扩展板例程	4.12

4.5 串口对照表

I-7188XC(D) 拥有串口如下表所示：

串口	硬件
COM1	芯片 80188 上 UART-0, 5 线制 RS-232 或 2 线制 RS-485
COM2	芯片 80188 上 UART-1, 2 线制 RS-485

I-7188XC(D)上 RS-232 或 RS-485 串口设置如下：

串口类型	引脚名称
2 线制 RS-485	Data+, Data-
3 线制 RS-232	TXD, RXD, GND
5 线制 RS-232	TXD, RXD, GND, RTS, CTS

使用 16C550 与使用芯片 80188-UART 的程序方式有着很大的差异，而 80188 上的中断操作则与 PC 主的 8259 芯片也是迥然不同。因此，PC 上基于 RS-232 的应用程序并不能在 I-7188XC(D)上运行。

I-7188XC(D)的软件驱动可调用中断驱动库，并可对每个串口提供 1K 字节的缓冲区。这样，软件更易于设计和使用。

软件驱动对于两个串口提供同样的接口，因此每一个串口可使用相同的方式进行操作，并不会因串口不同而增加开发难度。

4.6 使用串口

7188XC(D)拥有两个通讯端口：

- COM1 可以作为 RS-232（默认） 或 RS-485 通讯口
  RS-232: TXD, RXD, RTS, CTS 和 GND
  RS-485: DATA+, DATA- (内置 ASIC 自适应技术)
- COM2 是个 RS-485 通讯口 (D2+, D2-,内置 ASIC 自适应技术)

在使用串口操作前，必须调用函数 ***InstallCom()*** (或 ***InstallCom1/2***) 对串口进行初始化。而在退出程序前，须调用函数 ***RestoreCom()*** (或 ***RestoreCom1/2***)释放驱动。

在调用函数 ***InstallCom()***后，才可对串口进行数据的读取，发送等操作。

在读取串口数据前，函数 ***IsCom()*** 可地检测串口上是否有收到数据。若为真，函数 ***ReadCom()***即可用于读取在串口输入缓存的数据。
而向串口发送数据前,可使用函数 ***ClearCom()*** 来确定串口输出缓存是否已清空，之后再使用函数 ***ToCom()*** 从串口发送数据。

例：使用串口 COM1 (RS-232)读取/发送数据代码参考如下：

```
int port=1;   /*to use COM1*/
int quit=0, data;

InitLib();    /* Initiate the 7188xc library */
InstallCom(port, 115200L, 8, 0, 1);   /*install the COM driver*/
while(!quit){
   if(IsCom(port)){ /*check if any data is in the COM Port input
                      buffer*/
      data=ReadCom(port);   /*read data from the COM Port*/
      ToCom(port, data);   /*send data via the COM Port*/

      if(data=='q') quit=1;   /*if 'q' is received, exit the program*/
   }
}
RestoreCom(port);   /*uninstall the COM driver*/
```

可简单的把 “port” 参数从 **port=1** 更换为 **port=2** 即可从串口 COM1 修改为 COM2。

若程序使用串口 COM1 进行操作，修改代码如下：

```
int quit=0, data;

InitLib();    /* Initiate the 7188xc library */
InstallCom1(115200L, 8, 0, 1);    /*install the COM1 driver*/
while(!quit){
   if(IsCom1()){    /*check if any data is in the COM1 input buffer*/
      data=ReadCom1();    /*read data from COM1*/
      ToCom1(data);    /*send data via COM1*/
      if(data==’q’) quit=1;    /*if ‘q’ is received, exit the program*/
   }
}
RestoreCom1();    /*uninstall the COM driver*/
```

4.6.1 从串口输出显示

I-7188XC(D)库文件也支持诸如函数 ***printf()***这样的标准 C 语言库来进行以指定格式的输出

函数 ***printCom()*** 可用于所有串口，而函数 printCom1/2 则分别适用于指定串口号。在使用 ***printCom()*** 函数前，须调用 ***InstallCom()*** 函数进行串口初始化操作。具体代码如下：

```
int port=2;   /*to use COM2*/
int i;

InitLib();    /* Initiate the 7188xc library */
InstallCom(port, 115200L, 8, 0, 1);   /*install the COM2 driver*/
for(i=0; i<10; i++){
   printCom(port, "Test %d\r\n", i);   /*print data from COM2*/
}
RestoreCom(port);   /*uninstall the COM driver*/
```

4.6.2 使用 COM1/COM2 完成 RS-485 应用

串口 COM1/COM2 为 2 线制 RS-485 串口，引脚如下：

- D+:连接至 RS-485 网络的 Data+
- D-:连接至 RS-485 网络的 Data－

COM1/COM2 为半双工 2 线制 RS-485 网络，
可直接驱动 I-7000 系列模块，但不能用于全双工 4 线制应用场合。

在 2 线制的 RS-485 网络中发送/接收的方向控制就显现非常重要。为了解决这个问题，I-7188XC(D)内置有基于 RS-485 端口的 Self-Tuner ASIC 自适用芯片，可自动侦测并控制 RS-485 网络的数据流方向并进行控制。以该方式，应用程序可自由的发送/接收 RS-485 网络的数据，而不必考虑其数据流的方向及控制。

4.6.3 向 I-7000 系列模块发送指令

I-7000 系列模块的命令与 I-7188XC(D)指令有着天壤之别，但命令仍可使用函数 ***ToCom()*** 从 I-7188XC(D)进行发送。

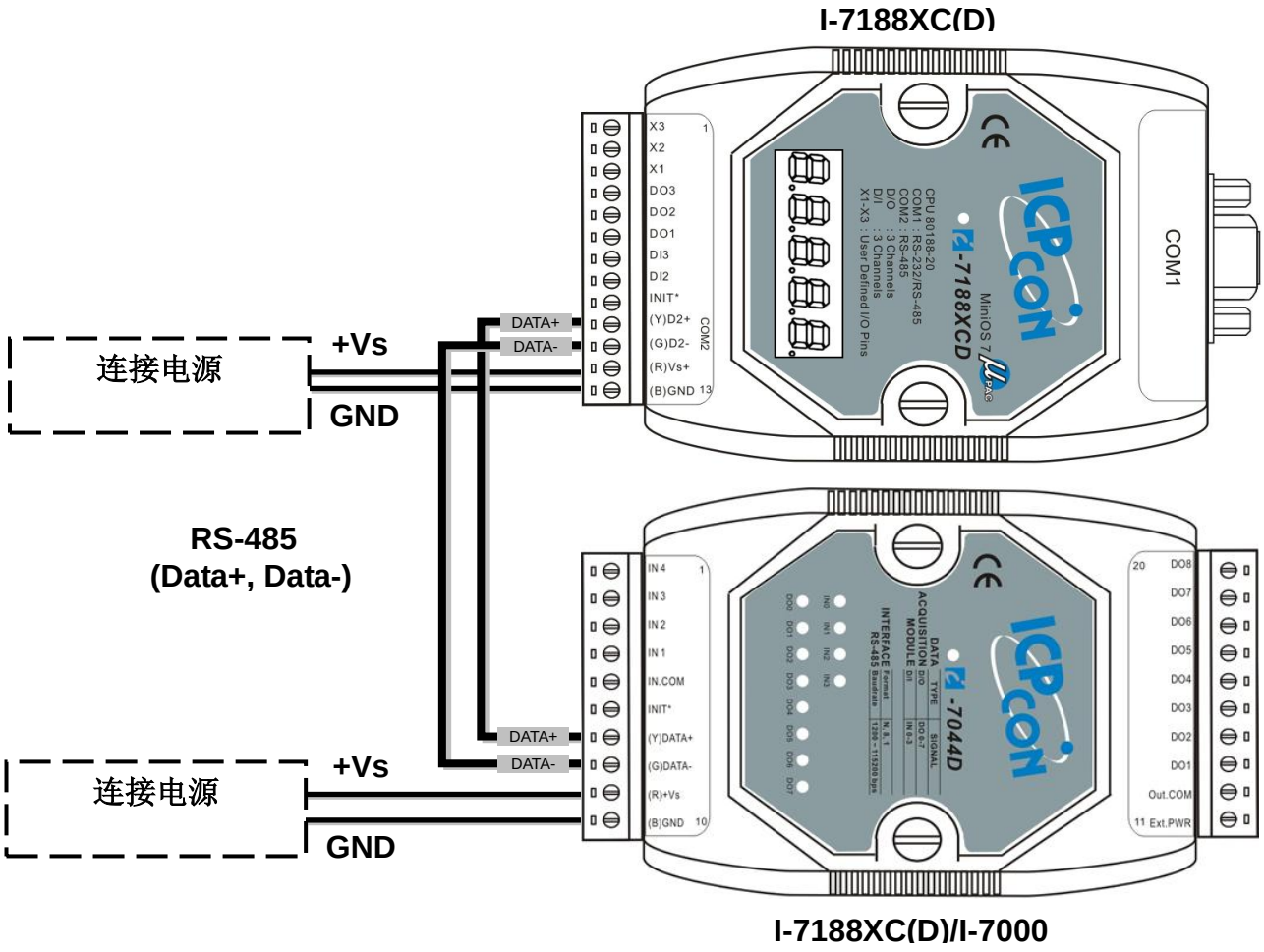
使用 COM1/COM2 连接并控制 I-7000 系列模块

有关 I-7000 系列模块应用步骤如下：

步骤 1: I-7188XC(D)发送命令字符串到 I-7000 系列模块
步骤 2: I-7000 模块执行命令
步骤 3: I-7000 模块可设定延时进行回应(1 个字节)

步骤 4: I-7000 模块返回数据给 I-7188XC(D).

注意: 步骤 3 中延时参数仅为 1 个字节长.



从串口 COM2 (RS-485)发送命令代码如下所示：

```
int port=2;   /*to use COM2*/
int i;
char data[ ]="$01M\r";   /*command to read a module's name*/

InitLib();    /* Initiate the 7188xc library */
InstallCom(port);   /*install the COM2 driver*/
for(i=0; i<5; i++)
   ToCom(port, data[i]);   /*send a command to the I-7000 module*/
……   /*program code*/
RestoreCom(port);   /*uninstall the COM driver*/
```

除函数 ***ToCom()***之外，函数 ***SendCmdTo7000()*** 亦可完成发送指令给 I-7000 系列模块的功能， ***ReceiveResponseFrom7000()*** 函数则用于接收来自 I-7000 系列模块的返回数据。

用于连接 I-7000 系列模块函数：

- ***SendCmdTo7000(int iPort, unsigned char *cCmd, int iChksum);***
  该函数用于向 I-7000 系列模块发送命令。若校验位被激活，函数将在指令尾部添加 2 字节的校验码。
- ***ReceiveResponseFrom7000_ms(int iPort, unsigned char *cCmd, long lTimeout, int iChksum);***
  在调用 ***SendCmdTo7000()*** 函数后，用户可利用函数 ***ReceiveResponseFrom7000_ms()*** 接收来自 I-7000 系列模块的返回数据。

更多相关例程请参考如下：
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\7K87K_Module

注意:更多 I-7000 命令，请参考 **7000** 用户手册**.**

4.7 使用红色 LED 及 7 段 LED 显示

I-7188XCD 带有一个 5 位 7-段 LED，并且每个位数的右下角皆有一个小数点，其可利用软件设定是否显示。为使各 LED 容易区分，每位 LED 从左到右使用数字 1 到 5 加以编号，且每个 LED 可独立编程。这样将在实际应用中非常实用，一定程度上可取代显示屏或触摸屏。

在使用 LED 前，须调用函数 ***Init5DigitLed()*** 进行初始化，之后方可使用函数 ***Show5DigitLed()*** 显示数据。有关 “7188d”的 5 位 7 段 LED 显示代码，参考如下：

```
InitLib();    /* Initiate the 7188xc library */
Init5DigitLed();
Show5DigitLed(1, 7);
Show5DigitLed(2, 1);
Show5DigitLed(3, 8);
Show5DigitLed(4, 8);
Show5DigitLed(5, 13);    /* The ASCII code for ‘d’ is 13 */
```

更多相关例程请参考如下：
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\LED

4.8 访问 I-7188XC(D)存储介质

4.8.1 使用 Flash 存储器

I-7188XC(D)模块包含有 256K 的 Flash memory，其中部分空间（0xF000 地址段）已用于储存操作系统 MiniOS7。因此，用户仅能使用其它剩余的 448K 字节。

Flash memory 每个位仅能从 1 写到 0，而不能从 0 写到 1，唯一将数据从 0 写到 1 的方式是调用 ***EraseFlash()*** 函数来清除 Flash Memory 的一个存储段(64K 字节)。这样，用户须要决定哪一段是否需要清除。

向 Flash Memory 0xD000 段，偏移量为 0x1234 写入一个整数，代码参考如下：

```
int data=0xAA55, data2;
char *dataptr;
int *dataptr2;

InitLib();    /* Initiate the 7188xc library */
dataptr=(char *)&data;
FlashWrite(0xd000, 0x1234, *dataptr++);
FlashWrite(0xd000, 0x1235, *dataptr);

/* read data from the Flash Memory method 1 */
dataptr=(char *)&data2;
*dataptr=FlashRead(0xd000, 0x1234);
*(dataptr+1)=FlashRead(0xd000, 0x1235);

/* read data from the Flash memory method 2 */
dataptr2=(int far *)_MK_FP(0xd000, 0x1234);
data=*data
```

从 Flash Memory 读取数据与从 SRAM 读取方式相同。用户首先须调用指针指向存储器单元，然后使用该指针访问存储器进行存取。在将数据写入 Flash Memory 之前，用户须调用函数 ***FlashWrite()***来验证数据能否正确写入。之后，可用 ***EraseFlash()*** 函数清除该存储段。

更多相关例程请参考如下：
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\Memory

4.8.2 使用 EEPROM

EEPROM 应用于存储并不频繁发生改变的数据，例如：

- 模块 ID, 配置设定
- 串口设置参数
- 小型数据库

另外，EEPROM 的读/写次数限制在 1,000,000 次，因此当测试时最好不要经常改变其中数据。而 EEPROM 可以每单字节为单位清除/写入数据，这样将非常适用于现实的大多数应用。

I-7188XC(D)拥有 2K 字节的 EEPROM 存储空间，包含 8 段，每段拥有 256 字节，以此计算 EEPROM 总存储空间为 2048 字节。通常，在默认情况下 EEPROM 为保护模式，数据不能写入。而利用函数 ***EE_WriteEnable()***写入之前调用，即可取消保护模式。
例如: 向 EEPROM 第 1 段，地址号为 10 写入数据，须首先调用函数 ***EE_WriteEnable()*** 。相关代码如下所示：

```
int data=0x55, data2;

InitLib();    /* Initiate the 7188xc library */
EE_WriteEnable();
EE_MultiWrite(1, 10, 1, &data);
EE_WriteProtect();

EE_MultiRead(1, 10, 1, &data2);       /* now data2=data=0x55 */
```

注意: 在向 EEPROM 写入整形数据时，函数 ***EE_WriteEnable()*** 须调用两次，与向 NVRAM 写入方法相同。

更多例程请参考如下信息：
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\Memory

4.9 使用 Watchdog 计时器

I-7188XC(D)的 watchdog 计时器在 MiniOS7 2.0 中固定为 0.8 秒。当 I-7188XC(D) 初次上电时，watchdog 计时器将自行激活。若 watchdog 计时器在 0.8 秒内没有刷新，则系统将会自动重启。

I-7188XC(D)的操作系统 MiniOS7 将在上电后，自行刷新 watchdog。用户可利用程序调用软件驱动停止 MinOS7 刷新 watchdog 计时器，但是之后程序必须人为进行刷新 watchdog 计时器。若程序在每个 0.8 秒内不刷新 watchdog，将会使 I-7188XC(D) 重启。

程序必须告知 MiniOS7，让其将 watchdog 计时器复位，然后停止并返回 MiniOS7 命令提示符。

使用函数 ***EnableWDT()*** 可激活 watchdog 计时器，而函数 ***DisableWDT()*** 则可将之禁用。在 watchdog 激活后，程序可在每达到 0.8 秒之前调用 ***RefreshWDT()*** 函数 ，否则 watchdog 将使 I-7188XC(D)模块重启。相关代码如下所示：

```
InitLib();   /* Initiate the 7188xc library */
EnableWDT();
while(!quit)
{
   RefreshWDT();
   User_function();
}
DisableWDT();
```

函数 ***IsResetByWatchDogTimer()*** 用于检测 I-7188XC(D)模块是否已经被 watchdog 计时器重启 。该函数必须在程序之初调用。相关代码如下所示：

```
main()
{
   InitLib();    /* Initiate the 7188xc library */

   if(IsResetByWatchDogTimer())
   {
      /* do something here to check the system */
   }
   quit=0;
   EnableWDT();
   while(!quit)
   {
      RefreshWDT();
      User_function();
   }
```

更多范例程序请参考如下信息
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\Misc

4.10 使用计时器函数

I-7188XC(D) 能支持一个主时间计时器，8 个 StopWatch 计时器和 8 个 CountDown 计时器。I-7188XC(D)使用一个 16 位计时器实现这些计时器函数(有效单位 1ms)。函数 InstallUserTimer() 用于安装用户计时器函数且该函数在每 1ms 时间间隔内调用。MiniOS7 系统计时器将在每隔 1ms 调用 INT 9 ，每隔 55ms 调用 INT 0x1C。库文件的计时器函数与相 INT9 联系，并调用用户计时器函数。

函数 ***TimerOpen()*** 用于启动 I-7188XC(D)计时器，且该函数必须在程序之初调用。函数 ***TimerClose()*** 用于停止计时器。相关代码如下所示：

```
unsigned long time iTime;

InitLib();   /* Initiate the 7188xc library */
TimerOpen();    /* Begin using the 7188XC timer function */
while(!quit)
{
   if(Kbhit())
      TimerResetValue();      /* Reset the main time ticks to 0 */

   iTime=TimerReadValue();   /* Read main time ticks */
}
TimerClose();   /* Stop using the 7188XC timer function */
```

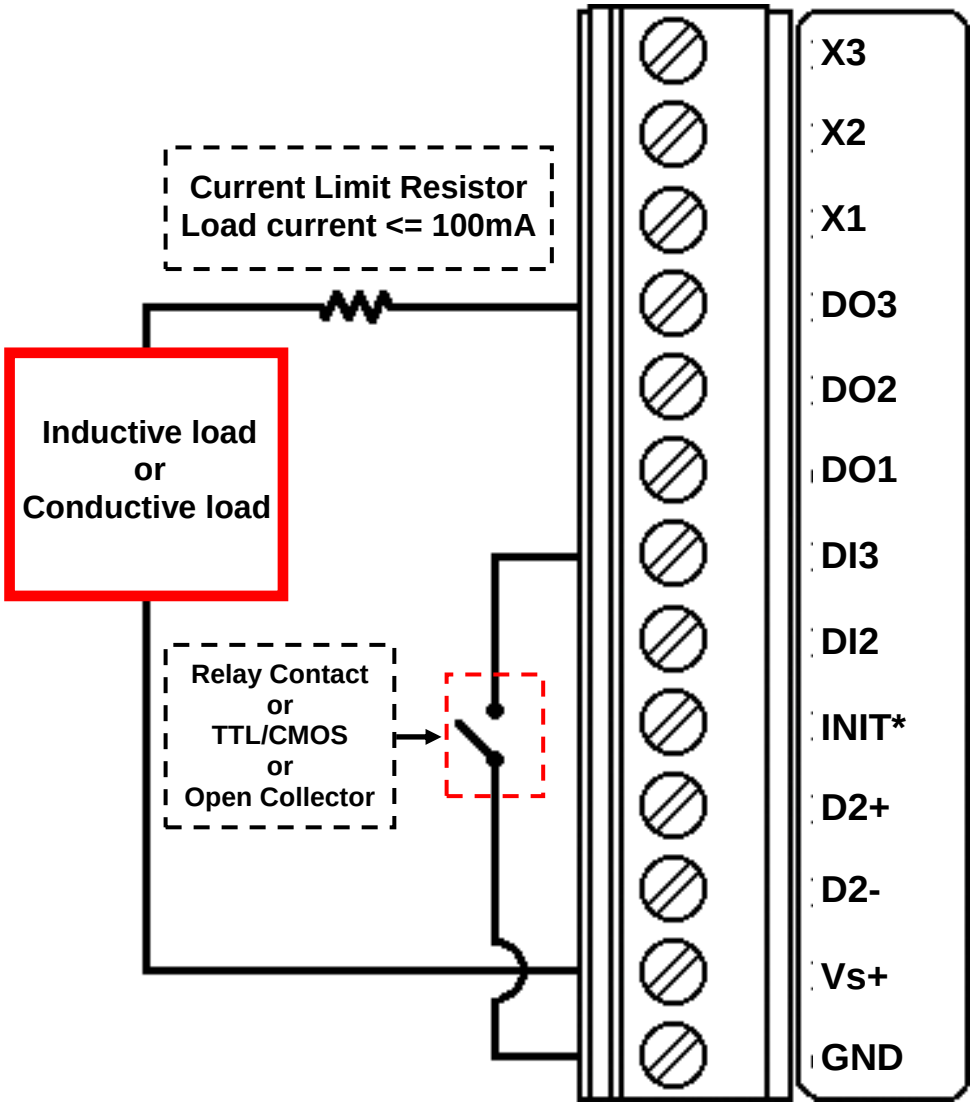
更多范例程序信息如下:
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\Timer

4.11 使用数字量输入及数字量输出

I-7188XC(D) 提供 2 路 DI 信道和 3 路 DO 信道。函数 ***SetDo1High ()***,***SetDo1Low ()***,***SetDo2High ()***,***SetDo2Low ()***,***SetDo3High()*** 及 ***SetDo3Low()***可用于控制三路 DO 通道，而函数 ***GetDi2()*** 和 ***GetDi3()*** 则用于读取两路 DI 通道状态。

有关 DI 和 DO 连线方式，请参考**章节 1.4.6 DI 和 DO 连接方式**

一路 DI/DO 配线方式如下：



有关读取及设定 DI 及 DO 代码如下：

```
int Do1, Do2, Do3;

InitLib();    /* Initiate the 7188xc library */

Print("DI=%s\n\r", GetDi2()?"High":"Low"); /* Read the state of DI2 */
Print("DI=%s\n\r", GetDi3()?"High":"Low"); /* Read the state of DI3 */

Do1=GetDo1();    /* Read the state of DO1 */
Print("DO1=%s\n\r", Do1?"High":"Low");
if(!Do1)
   SetDo1High();    /* Set the DO1 to ON */
else
   SetDo1Low();    /* Set the DO1 to OFF */

Do2=GetDo2();    /* Read the state of DO2 */
Print("DO2=%s\n\r", Do2?"High":"Low");
if(!Do2)
   SetDo2High();    /* Set the DO2 to ON */
else
   SetDo2Low();    /* Set the DO2 to OFF */

Do3=GetDo3();    /* Read the state of DO3 */
Print("DO3=%s\n\r", Do3?"High":"Low");
if(!Do3)
   SetDo3High();    /* Set the DO3 to ON */
else
   SetDo3Low();    /* Set the DO3 to OFF */
```

更多范例程序参考如下信息：
CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\IO_Pin

4.12 使用 I/O 扩展总线

在众多串行接口设备大行其道的今天，I/O 扩展总线包含有串行和并行接口。并行接口与 ISA 总线类似，在以旧版 ISA 总线设计的基础上进行小部分修改，并将其移植到 I/O 扩展总线。而串行总线 I/O 引脚可进行编程也可用于 D/I 或 D/O。

这些串行设备特点如下：

- 与并行设备相比尺寸更小
- 与并行设备相比成本更低
- 易于在应用中进行隔离保护

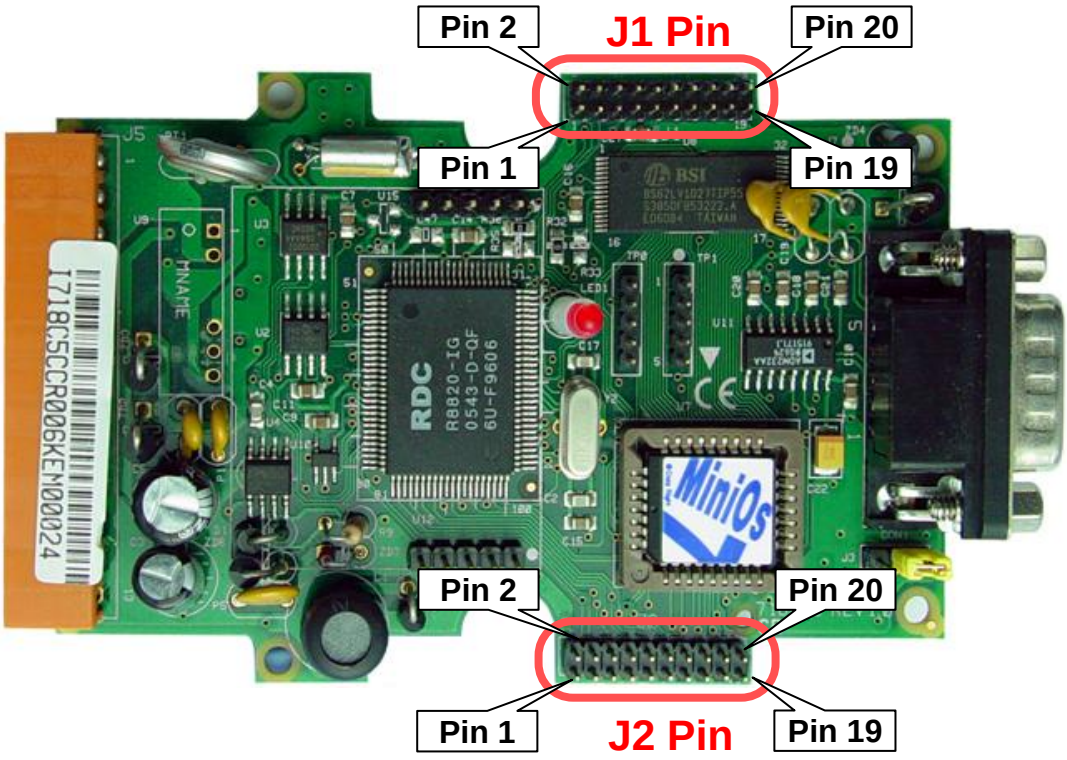
I/O 扩展总线的串行接口可非常便捷的与串行设备相连接。

4.12.1 I/O 扩展总线定义

I-7188XC(D)模块的 I/O 扩展总线可分为如下 3 大部分：

1. 电源及复位信号：VCC, GND, RESET 及 /RESET
2. 并行总线：
   - 系统时钟： CLOCKA
   - 异步控制： ARDY
   - 地址总线： A0 ~ A7
   - 数据总线： D0 ~ D7
   - 中断控制：INT0 和 INT1
   - 芯片选择及读/写：/CS, /WR and /RD
3. 串行总线： TO_0, TO_1, TI_0, TI_1, SCLK, DIO9, DIO4 and DIO14

I/O 扩展总线定义如下：



J1 引脚定义及说明:

序号	名称	描述
1	GND	PCB 地
2	GND	PCB 地
3	CLOCKA	CPU 同步时钟输出
4	ARDY	异步输入(灵敏等级, OPEN=ready)
5	INT0	通道 0 中断请求输入(异步, 高态有效)
6	INT1	通道 1 中断请求输入(异步, 高态有校)
7	VCC	PCB 电源提供
8	RESET	电源复位脉冲 (高态有效)
9	GND	PCB 地
10	/RESET	电源复位脉冲 (低态有效)
11	TO_0	CPU 计时器输出 0(可编程 D/I/O)
12	TO_1	CPU 计时器输出 1(可编程 D/I/O)
13	TI_0	CPU 计时器输入 0(可编程 D/I/O)
14	TI_1	CPU 计时器输入 1(可编程 D/I/O)
15	SCLK	7188 系列模块通用串行同步脉冲输出
16	DIO9	可编程 D/I/O
17	DIO4	可编程 D/I/O
18	DIO14	可编程 D/I/O

19	VCC	电源供应
20	VCC	电源供应

- **CLOCKA:** 20M
- **ARDY:**该引脚在没有异步需求的应用时，通常为 OPEN
- **INT0 及 INT1:**在没有中断需求的应用时，为 OPEN
- **TO_0 及 TO_1:**可作为 CPU 或可编程 DI/O 计时器输出
- **TI_0 及 TI_1:** 可作为 CPU 或可编程 DI/O 计时器输入
- **DIO4, DIO9 及 DIO14:** 可编程 I/O
- **SCLK:** I-7188XC(D)使用该信号为作时钟脉冲源来驱动所有板载串行设备，因此该信号通常可作为可编程的 DO。改变该信号来进行其它设置将会产生严重错误。该信号可驱动外部串行设置而无任何副作用。

J2 引脚定义和描述

序号	名称	描述
1	A0	地址总线
2	D0	数据总线
3	A1	地址总线
4	D1	数据总线
5	A2	地址总线
6	D2	数据总线
7	A3	地址总线
8	D3	数据总线
9	A4	地址总线
10	D4	数据总线
11	A5	地址总线
12	D5	数据总线
13	A6	地址总线
14	D6	数据总线
15	N/C	无连接
16	D7	数据总线
17	N/C	无连接
18	/WR	写入选通输出(异步，低态有效)
19	/CS	芯片选择输出 (异步，低态有效)
20	/RD	读取选通输出 (异步，低态有效)

- **地址总线 (输出):** A0 ~ A6
- **数据总线 (三态, 双向):** D0 ~ D7
- ***/CS, /RD* 及 */WR:*** 这 3 个信号均为同步时钟(JP1 引脚 3)及异步 ARDY (JP1 引脚 4)
- 如果程序需要来自 I/O 地址 0 到 0xff 的输入/输出数据，/CS 将被激活。

注意：I-7188XC(D)JP2 的引脚 15 和引脚 17 是被保留的。用户需要这两个引脚无连接。更多信息更多详情更多详情请参考" **iobus_e.pdf** "

4.12.2 I-7188XC(D)重新配置

I-7188XC(D)的三个 DO 信道和两个 DI 信道来自引脚 4 到引脚 8。定制 7000 模块应用，这 5 个引脚可以硬件重新装配成其它功能，如下

步骤 1: 重新配置电阻移除如下：

- 如果是 DO3 重新配置，移除 R19
- 如果是 DO2 重新配置，移除 R20
- 如果是 DO1 重新配置，移除 R21
- 如果是 DI3 重新配置，移除 D13
- 如果是 D12 重新配置，移除 R23
- 那么内置 DI/DO 功能就被禁用

步骤 2:安装一个 5-pin 公头跳线到 I-7188XC(D)的 TP0 上。

步骤 3:安装一个 5-pin 母头跳线在扩展板的 TP1 上。那么 pin-4 到 pin-8 的外部信号就能够连接到扩展板上。用户可以认为这 5 个 D/I/O 引脚是用于他们的指定需求。

注意：如果 DO2 重新配置成 DI，它的初始状态需要是高电平，如果它的初始状态是低电平，系统时钟将被减小成 10M，因此所有时钟相关的库只会在中速。

I-7188XC(D)的 X100 是一个 8 通道 DI 板。移除所有 5 个电阻，R19 到 R23，并且重新配置这些引脚作为 DI 引脚。安装一个 5-pin 公头跳线到

I-7188XC(D)的 TP1 上。所以，通过设定跳线到不同的点上，用户能够选择最初的 3_DO_2_DI 功能或 5_DI 功能。更多信息参考“**iobus_e.pdf** ”

4.12.3 I/O 扩展板

I/O 扩展面包板和检测板：

扩展板	描述
X000	面包版 (小型)
X001	面包版 (大型)
X002	面包板
X003	自检板

I/O 扩展板 DI 和 DO：

扩展板	描述
X100	8 路 DI
X101	8 路 DO
X102	2 路继电器输出
X103	7 路 DI
X104	8 路 DI 或 8 路 DO (每路可以配置成 DI/DO)
X105	8 路 DI 或 8 路 DO (每路可以配置成 DI/DO)
X106	3 路 DI channels 或 2 路 DO
X400	3 路 16-bit timer/counter

I/O 扩展板 A/D,D/A,DI 和 DO

扩展板	描述
X200	1 路 A/D (0~2.5V)
X300	2 路 D/A (0~4.095V)
X301	1 路 A/D (0~2.5V) + 1 路 D/A (0~4.095V)
X302	1 路 A/D (+/-5V) + 1 路 D/A (+/-5V)

I/O 扩展板 RS-232/422/485, DI 和 DO

扩展板	描述
X500	1 个 9 线制 RS-232
X501	1 个 5 线制 RS-232
X502	1 个 3 线制 RS-232 + 1 个 5 线制 RS-232

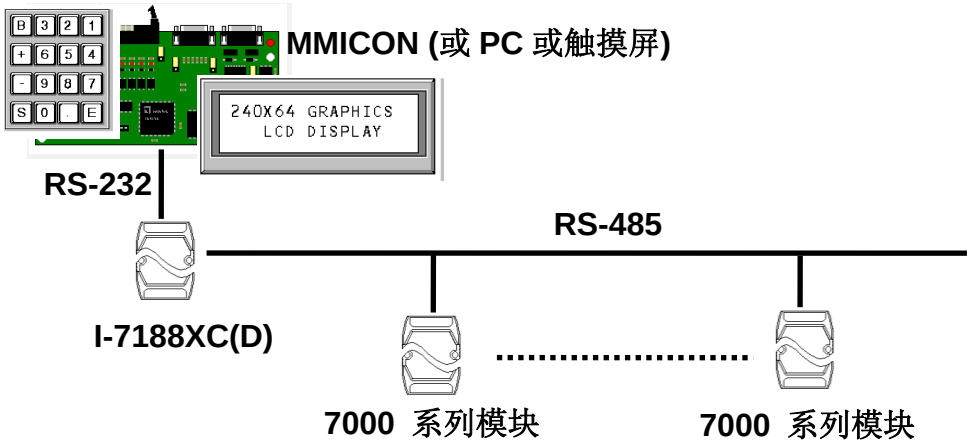
I/O 扩展板存储设备

扩展板	描述
X600	4M byte NAND Flash
X601	8M byte NAND Flash
X607	128K 带后备电池的 SRAM
X608	512K 带后备电池的 SRAM

注意: 更多信息参考“ **iobus_e.pdf** ”

5. 产品应用

5.1 嵌入式控制器



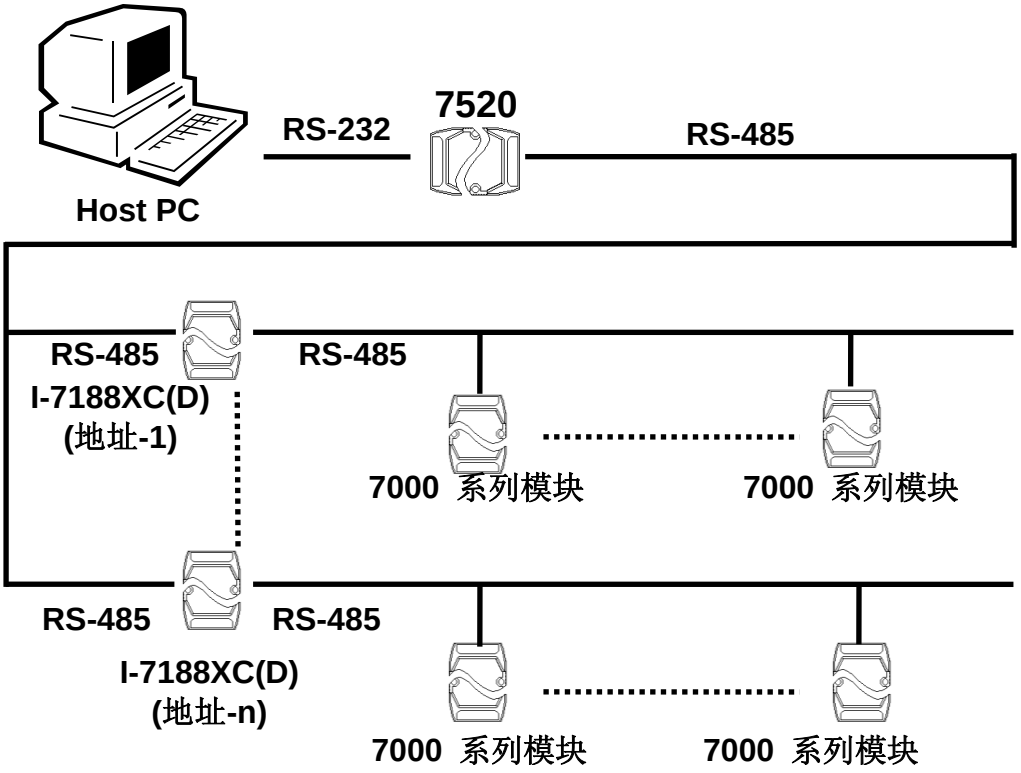
应用:

- 替代 **PC-based** 构架控制器
- 替代 **PLC**
- 替代其它专用控制器

I-7188XC(D)在多数应用中可作为嵌入式控制器，并可取代主机 PC, PLC 及其它专用控制器。

编程工具	TC/BC/MSC
调试工具	标准输入/输出(PC 键盘及显示器)
人机界面	• MMICON • PC 键盘及显示器 • 触摸屏 (RS-232 或 RS-485 接口)
程序	存储在 Flash Memory 中
输入/输出	• 板载 DI 或 DO • 来自 I/O 扩展总线 • 可直接控制高达 256 块 7000 系列模块，即为成百上千个 I/O 点

5.2 本地实时控制器 (RTC)



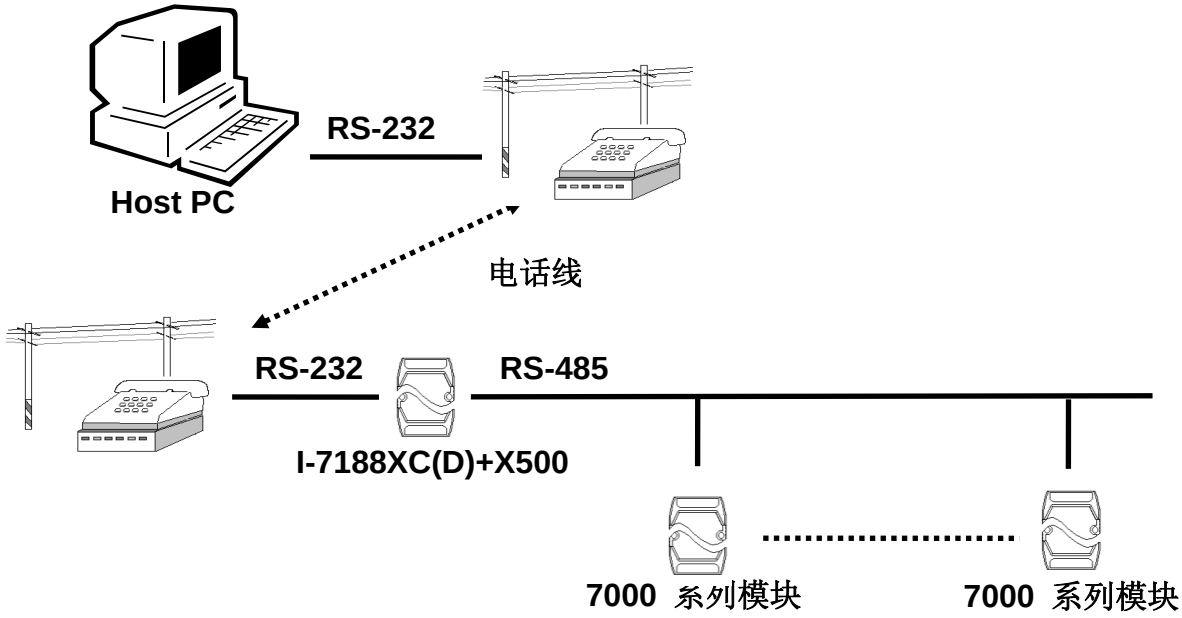
在一般应用架构中下，7000 系列模块作为从机设备，可在主机 PC 上执行控制程序。具体操作步骤如下：

- PC 发送命令给 7000 系列模块并接收输入数据
- PC 分析该输入数据并产生输出数据
- PC 发送命令给 7000 系列模块作为输出数据

若大量 7000 系列模块在系统中，将花费主机 PC 长时分析并控制这些模块，因此控制程序可在本地 I-7188XC(D)中执行。然后 PC 仅发送控制参数给 I-7188XC(D)， I-7188XC(D)即可基于这些参数控制本地 7000 系列模块。在该方式下，成百上千的 7000 系列模块可让 PC 经由 I-7188XC(D)进行控制。

所以现场 I-7188XA(D)可进行实时控制，而无需由主机 PC 参与。

5.3 远程控制器



在该应用架构中，控制程序在 I-7188XC(D)中执行，I-7188XC(D)可直接控制现场的 7000 系列模块基于本地的控制策略。

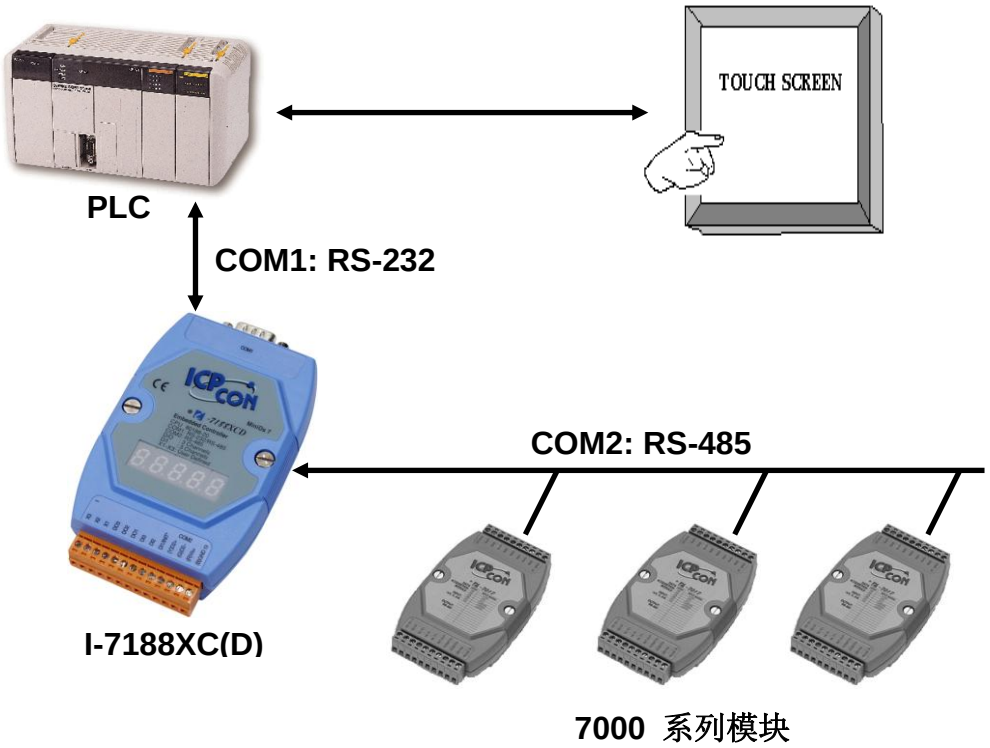
主机 PC 访问远程 I-7188XC(D) ，详细方式参考如下：

- 查询并记录远程系统状态
- 将控制参数下载至远程 I-7188XC(D)

远程 I-7188XC(D)与主机 PC 通讯，详细方式参考如下：

- 紧急事件通知及响应
- 远程系统状态通知

5.4 PLC I/O 扩展应用



大多数 PLC 针对人机一体化应用包含有人机界面，I-7188XC(D) 可使用该接口在 PLC 与 7000 系列模块之间建立通讯桥梁。

I-7188XC(D)可直接读/写 PLC 内存储器，这样 PLC 可按如下方式访问 7000 系列输入模块

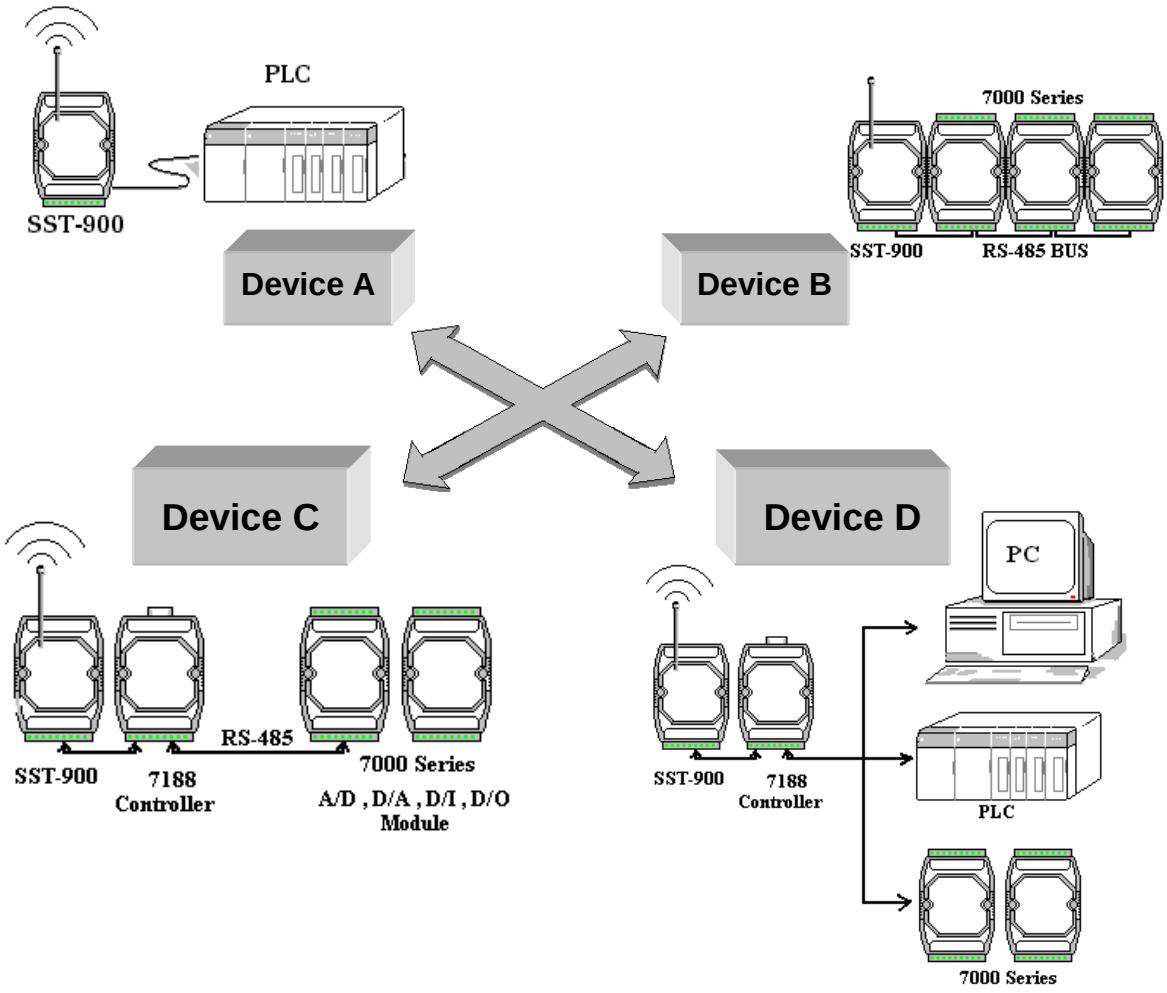
- I-7188XC(D)发送指令给 7000 系列输入模块
- I-7188XC(D)将数据写入 PLC 内存储器
- PLC 通过内存储器访问该数据

PLC 可按如下方式控制 7000 系列输出模块：

- PLC 将数据写入其内存储器
- I-7188XC(D) 从 PLC 的内存储器读取输出数据
- I-7188XC(D) 发送指令给 7000 输出模块

以该方式，从 7000 系列模块获得的输入数据可显示在触摸屏。另外，7000 系列模块的输出信号可由触摸屏控制。

5.5 无线调制解调器应用



SST-900/SST-2400 设定: (设备 A)

- RS-232
- 半双工模式
- 同步运行
- 从机状态
- 波特率=9600
- 通道=3
- 频率=915.968MHz

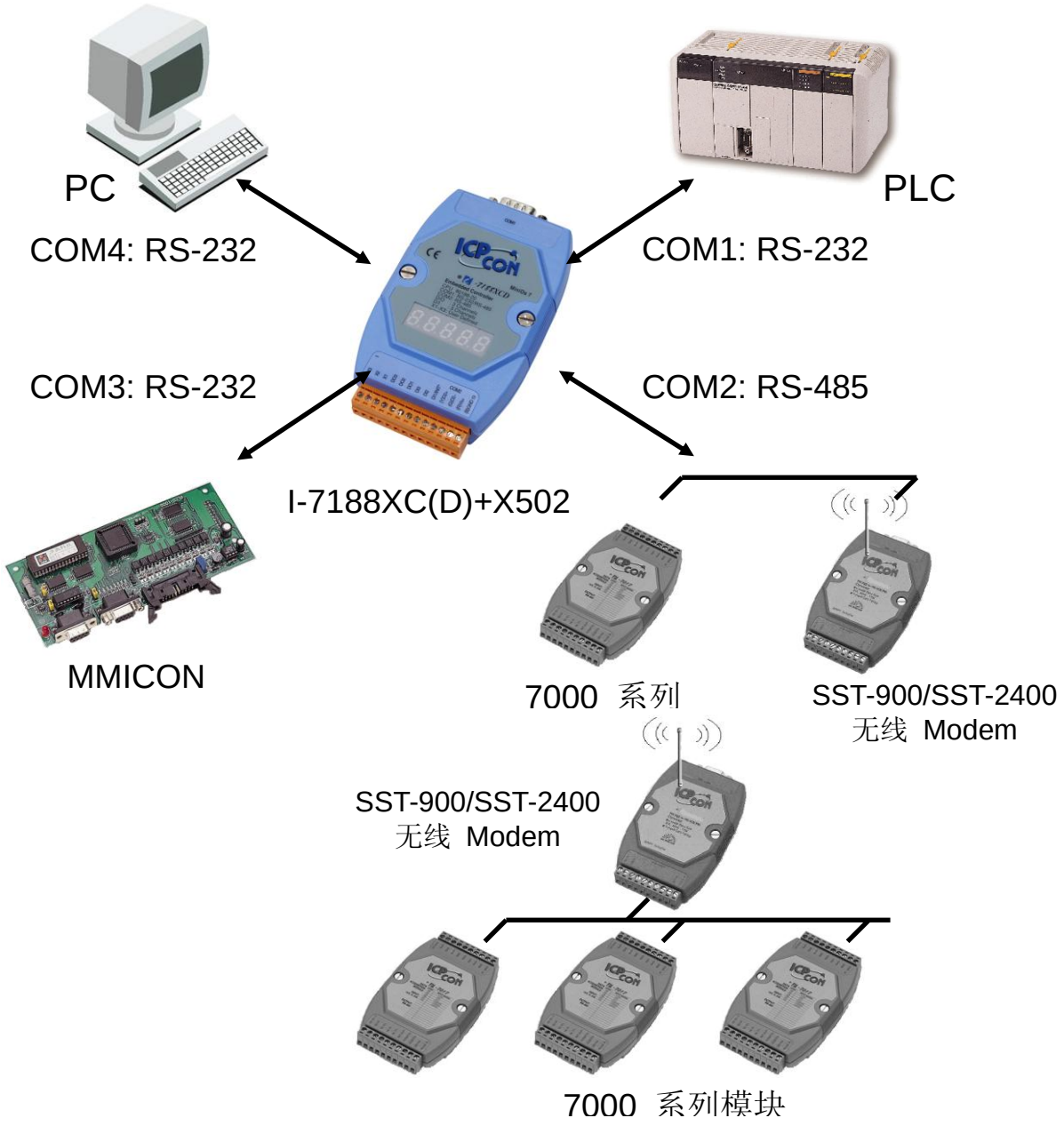
SST-900/SST-2400 设定: (设备 B/C/D)

- RS-485 或 RS-232
- 半双工模式

- 同步运行
- 从机状态
- 波特率=9600
- 通道=3
- 频率=915.968MHz

I-7188XC(D)是一个可编程嵌入式控制器，**可在 SST-900 和其它外部设备(**例如：PLC，控制器和其它 7000 模块**)构建有效的通讯桥梁。**

5.6 一个应用中使用 4 个串口



COM1: PLC 可以访问 7000 系列模块 I/O 状态。
COM2:直接控制 7000 系列 输入/输出 模块。
COM3:本地人机界面
COM4: PC 来监视并且记录系统数

附录 A: 什么是 MiniOS7

MiniOS7 是由泓格科技(ICP DAS)开发的嵌入式操作系统，它用来在 7188 系列模块中取代 ROM-DOS 操作系统。一些公司已经创建了很多品牌的 DOS 操作系统，所有的 DOS 不管是 PC-DOS、MS-DOS 还是 ROMDOS) 都是用来告诉计算机如何处理信息的指令集或代码，它可以执行程序、管理文件、控制信息处理、指定输入和输出，并执行许多其它相关的功能。

MiniOS7 提供与 ROM-DOS 相同的功能， 另外 I-7188XC(D)还可为用户提供很多特别的功能。

MiniOS7 与 ROM-DOS 对照表如下所示：

	MiniOS7	**ROM-DOS**
启动时间	0.1 秒	4 ~ 5 秒
占用存储空间	<64K 字节	64K 字节
支持 I/O 拓展总线	Yes	No
支持 ASIC Key	Yes	No
Flash ROM 管理	Yes	No
更新 O.S. (由串口线下载更新)	Yes	No
内嵌硬件诊断功能	Yes	No
直接控制 7000 系列模块	Yes	No
用户 ODM 功能	Yes	No
免费	Yes	No

注意: ICP DAS 有权改变 MiniOS7 的规格而无需声明。

MiniOS7 的典型命令套件：

命令	**说明**
LED5 pos value	在 5 位 LED 的指定位置，显示一个十六进制数值
USE NVRAM	读/写 NVRAM
USE EEPROM	读/写 EEPROM
USE FLASH	读/写 Flash Memory
USE COM2 /option	从 COM2 端口发送/接收数据
DATE [mm/dd/yyyy]	设置 RTC 的时间
Time [hh:mm:ss]	设置 RTC 的时间

MCB	检测当前存储器
UPLOAD	在 I-7188XC(D)的 SRAM 中存储 MiniOS7 镜像文件 (该命令用来更新操作系统)
BIOS1	在 I-7188XC(D)的 Flash memory 中存储 MiniOS7 镜像文件 (该命令用来更新操作系统)
LOAD	下载用户程序到 I-7188XC (D)的 Flash Memory
DIR [/crc]	显示 I-7188XC (D)的 Flash Memory 上的文件信息
RUN fileno	通过文件编号执行文件
Filename	通过文件名称执行文件
DELETE or DEL	删除所有文件。
RESET	复位
DIAG [option]	执行硬件诊断
BAUD baudrate	设置 COM1 的波特率
TYPE filename [/b]	显示文件内容
REP [/#] command	重复执行同样的指令
RESERVE [n]	为储存程序保留 n 个 Flash Memory 扇区
LOADR	将文件下载到 SRAM
RUNR [option]	运行储存在 I-7188XC(D)的 SRAM 中程序 (使用 LOADR 命令下载程序)
I/INP/IW/INPW port	从硬件端口读取数据
O/OUTP/OW/OUTPW port value	向硬件端口发送数据
更多……	

注意: 更多有关 MiniOS7 详细信息，请参考：
CD:\Napdos\MiniOS7\document\Lib_Manual_For_7188XABC\index.htm
或
http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/document/lib_manual_for_7188xabc/index.htm

附录 B: MiniOS7 Utility 及 7188XW

MiniOS7 Utility 和 7188xw.exe 应用程序都可以帮助用户简单地更新 MiniOS7 到最新版本。MiniOS7 Utility 及 7188xw.exe 应用程序适用于执行基本配置及下载应用程序到嵌入于 I-7188XC(D)控制器中。

MiniOS7 Utility

MiniOS7 Utility 程序提供以下三个主要功能：

- 更新 MiniOS7 镜像文件
- 下载程序到 Flash Memory
- 配置 COM 端口设置

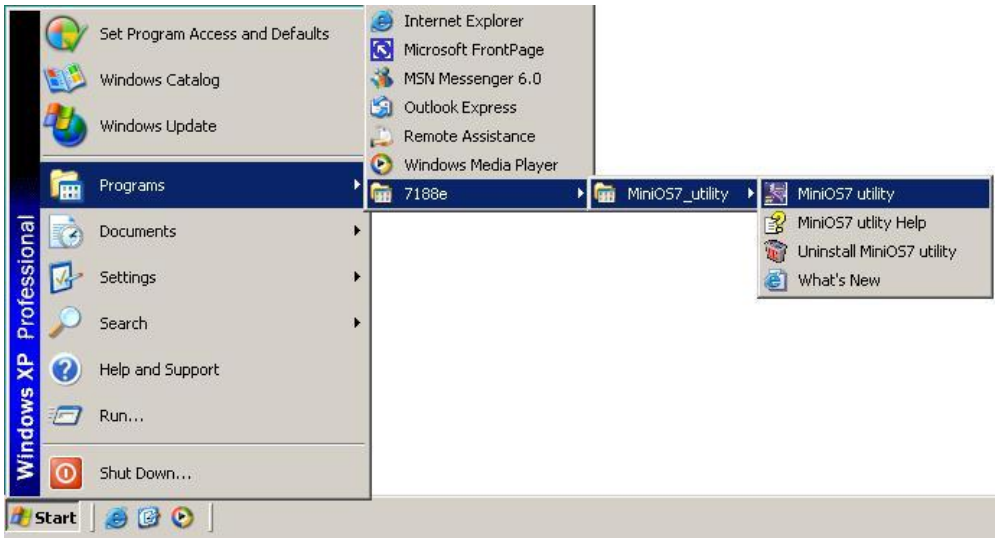
MiniOS7 utility 地址

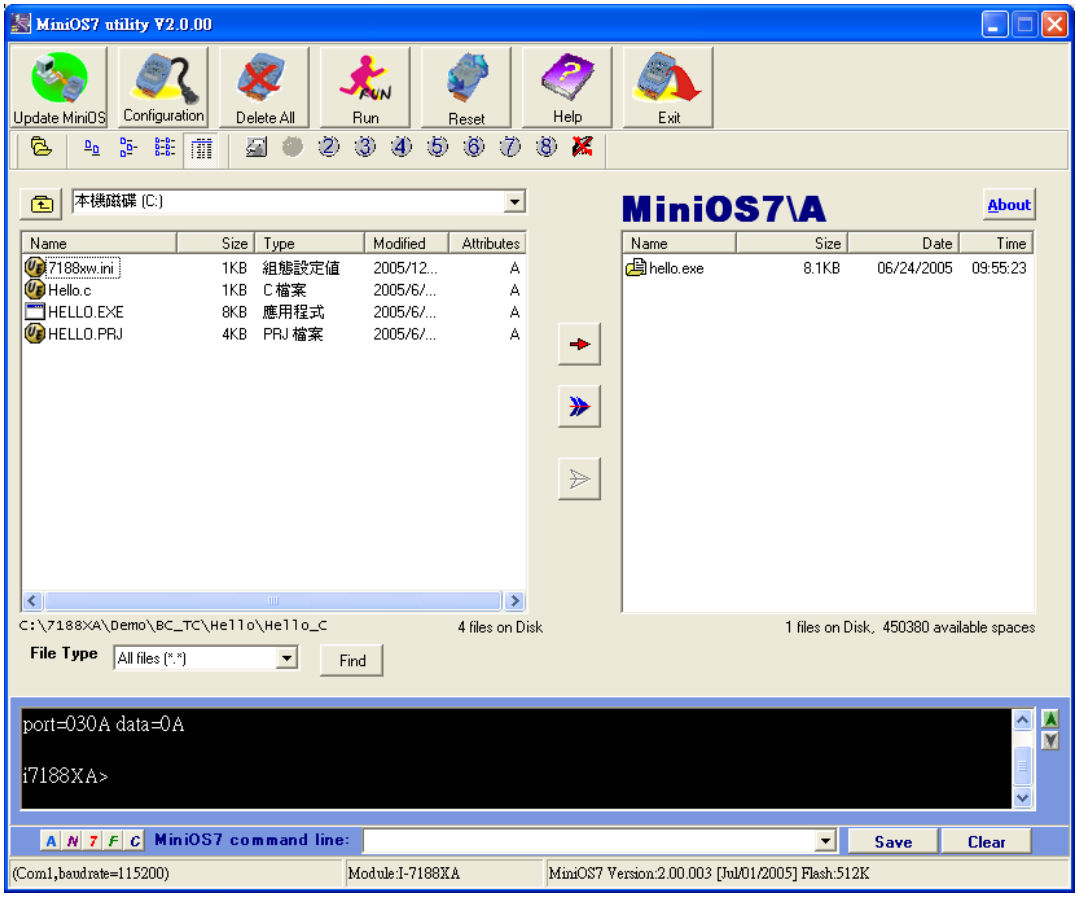
MiniOS7 utility 可以如下方式获得：
CD:\NAPDOS\MINIOS7\UTILITY\MiniOS7_utility\ 或
http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/minios7_utility/

安装步骤

步骤 1: 找到并执行 **minios7_utility_v2000.exe** 在 **CD:\Napdos\MiniOS7\utility\MiniOS7_utility** 文件夹中

步骤 2: 安装完成后一个新的文件夹**7188E**将出现在PC的开始目录的程序选项中。展开这个文件夹即可访问 MiniOS7 Utility 文件。详细信息请见如下图表：





7188XW

7188xw.exe 应用程序作为 I-7188XC (D)最主要的工具，可用来执行如下功能：

- **从主机 PC 下载程序到 I-7188XC (D)模块的存储单元**
- **下载 MiniOS7 镜像文件到 I-7188XC (D)控制器的 Flash Memory，从而来升级 MiniOS7 操作系统**
- **在主机 PC 的显示器上显示调试字符串**
  三个标准的输出功能库，比如 Putch、Print 和 Puts 函数，可让主控单元发送输出字符串到主机 PC 的显示器。
- **使用主机 PC 的键盘发送数据到 I-7188XC (D) 模块**
  三个标准的输入功能库，比如 Getch、Scanf 和 LineInput 函数，可让主控单元接收主机 PC 输入的字符串。

7188xw.exe 地址：

7188xw.exe 可以如下方式获得：

文件夹 CD:\Napdos\MiniOS7\utility\ 或网页

http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/

7188xw.exe 的 MiniOS7 命令行操作：

命令选项	说明
/c#	使用主机 PC 的 COM#端口
/b#	设置主机 PC 上串口的波特率(默认为 115200)
/s#	设置显示器显示的行数 (默认 25，最大 50)

7188xw.exe 热键清单：

使命选项	说明
F1	显示 7188xw.exe 的帮助信息
Alt_F1	用中文（big5 码）进行显示 7188xw.exe 的帮助信息
Ctrl_F1	用中文（GB2312 码）进行显示 7188xw.exe 的帮助信息
Alt_1	使用主机 PC 的串口 COM1
Alt_2	使用主机 PC 的串口 COM2
Alt_3	使用主机 PC 的串口 COM3
Alt_4	使用主机 PC 的串口 COM4

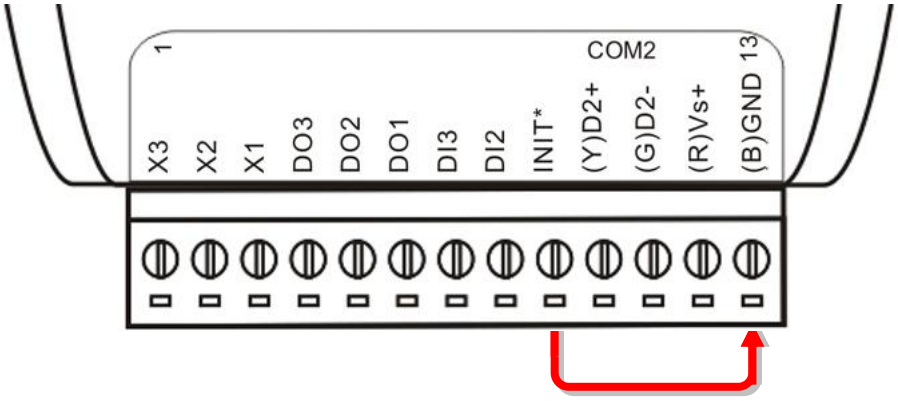
Alt_5	使用主机 PC 的串口 COM5
Alt_6	使用主机 PC 的串口 COM6
Alt_7	使用主机 PC 的串口 COM7
Alt_8	使用主机 PC 的串口 COM8
Alt_9	使用主机 PC 的串口 COM9
Alt_A	在普通模式和 ANSI-Escape-code-support 模式间切换
Alt_C	切换到命令模式，支持命令： b#:设置主机 PC 上串口的波特率 c#:使用主机 PC 的 COM#端口 n/e/o: 设置奇偶位为 none/even/odd 5/6/7/8: 设置数据位到 5/6/7/8 p#: 设置主机 PC 的工作目录 q: 推出命令模式
Alt_D	将 RTC 的日期设置成主机 PC 的日期
Alt_T	将 RTC 的时间设置成主机 PC 的时间
Alt_E	用于下载文件到存储器。屏幕上出现“Press ALT_E to download file!” 信息后点击 Alt_E
Alt_H	切换 Hex/ASCII 显示模式
Alt_L	切换 normal/line 模式。在 line 模式中，在点击 ENTER 键前所有的参数将不会发送到串口，它可用来测试 7000 系列模块
Alt_X	退出 7188xw.exe 应用程序
F2	设置下载文件名 (无需初始化下载操作)
F5	运行由 F2 指定的程序，并使用由 F6 设置的变量参数
F6	设置由 F2 指定的程序的变量(最大 10 个变量，如果设置少于 10 个需要在结尾加‘*’)
Ctrl_F6	清空屏幕
F8	F8=F9+F5
F9	将由 F2 下载的程序下载到 FLASH 存储器
Alt_F9	将所有用 ALT_F2 选定的文件下载到 FLASH 存储器
F10	将所有用 ALT_F2 选定的文件下载到 SRAM 并执行
Alt_F10	将所有用 ALT_F2 选定的文件下载到 SRAM 存储器
Ctrl_B	发送 BREAK 信号到主机 PC 表示此端口正被 7188xw.exe 占用
更多…	

更多关于 7188xw.exe 应用程序的详细信息请参考一下目录的 index.htm 文件

CD:\Napdos\MiniOS7\document\Lib_Manual_For_7188XABC\ 或 http://ftp.icpdas.com.tw/pub/cd/8000cd/napdos/minios7/document/lib_manual_for_7188xabc/

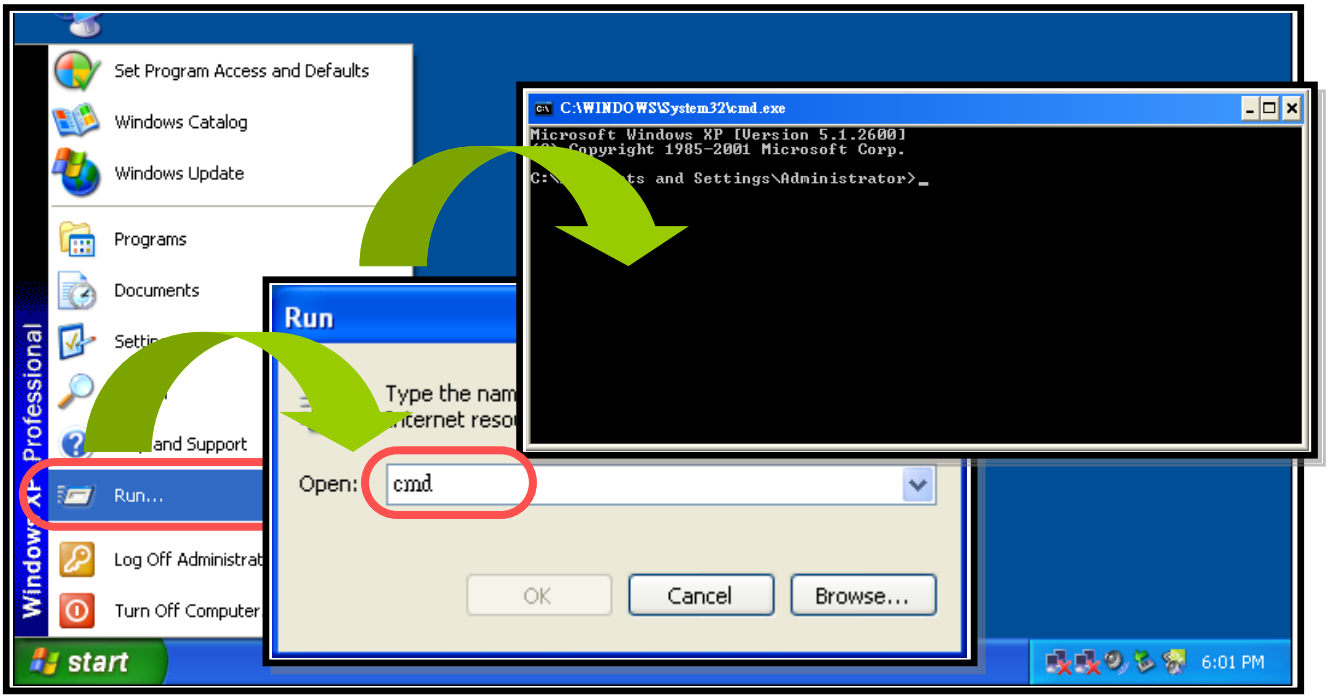
使用 7188xw.exe 程序下载文件到 I-7188XC (D)控制器：

步骤 1: 关闭 I-7188XC (D)的电源，将 INIT*引脚 与 GND 引脚相连，然后接通 I-7188XC (D) 的电源。

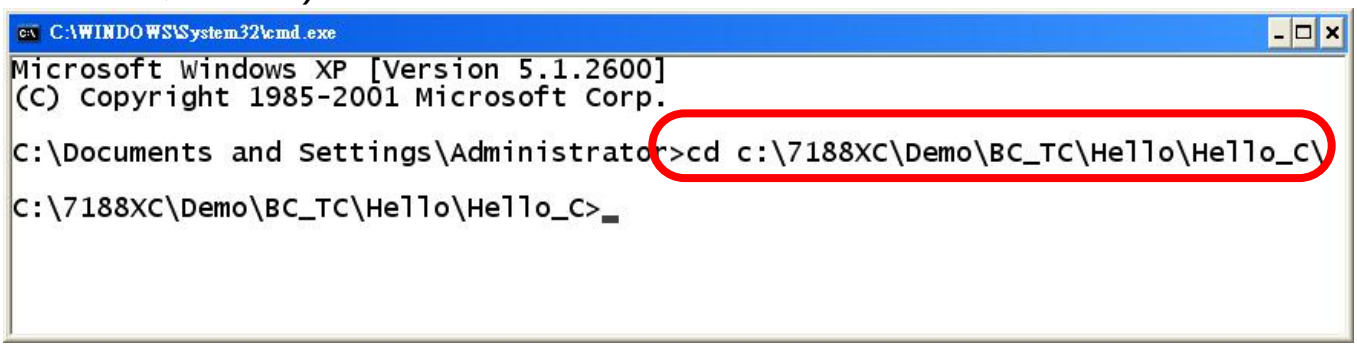


步骤 2: 当 I-7188XC (D) 启动后，断开 INIT*和 GND 引脚的连接

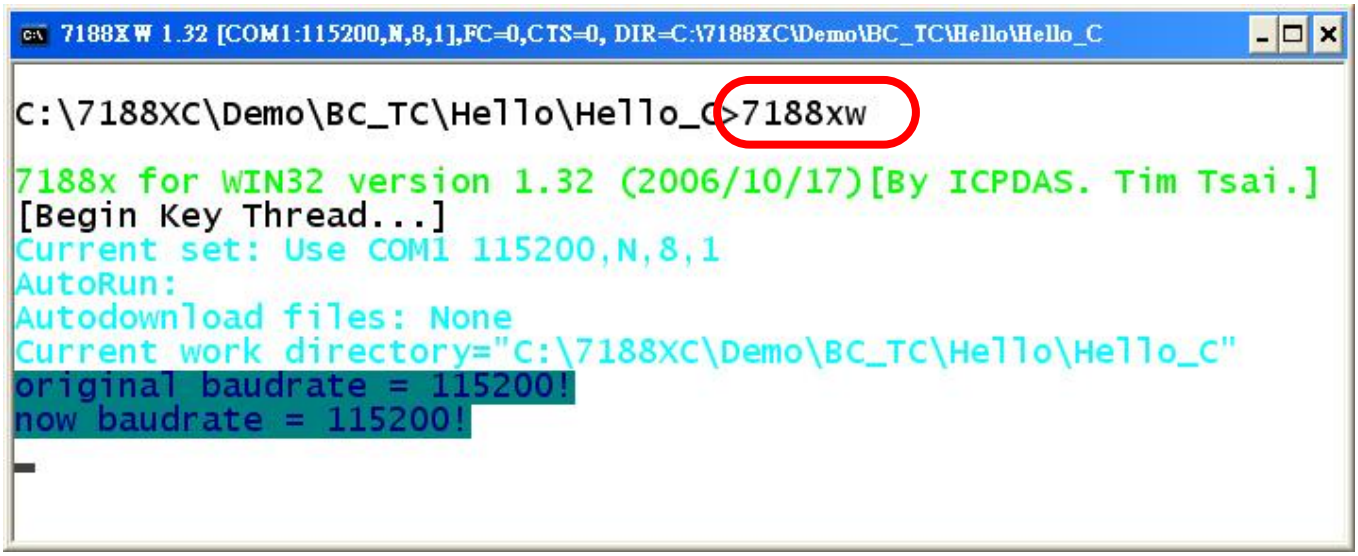
步骤 3: 打开 MS-DOS 命令提示符窗口并按如下图步骤操作：



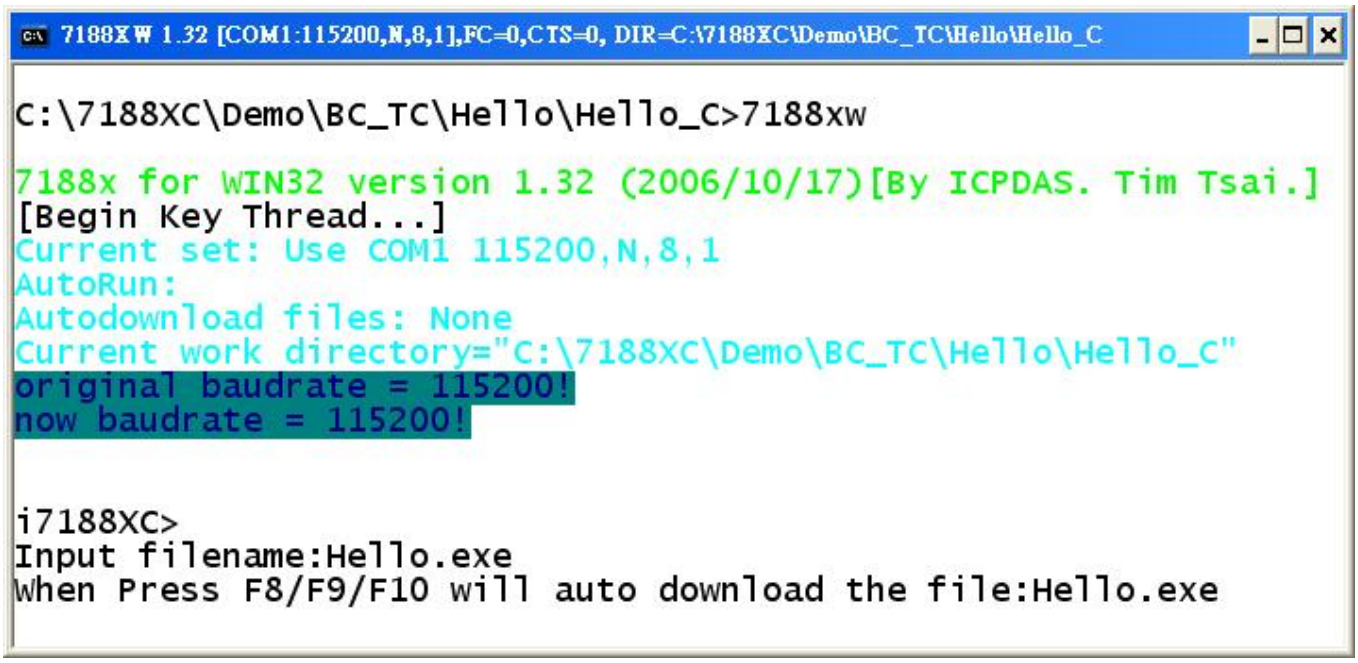
步骤 4: 输入 “cd c:\7188XC\Demo\BC_TC\Hello\Hello_C\”然后点击 **<Enter>**。(假定用户复制 7188XC 文件夹到 C 盘，参考步骤 2 在章节.2.1)



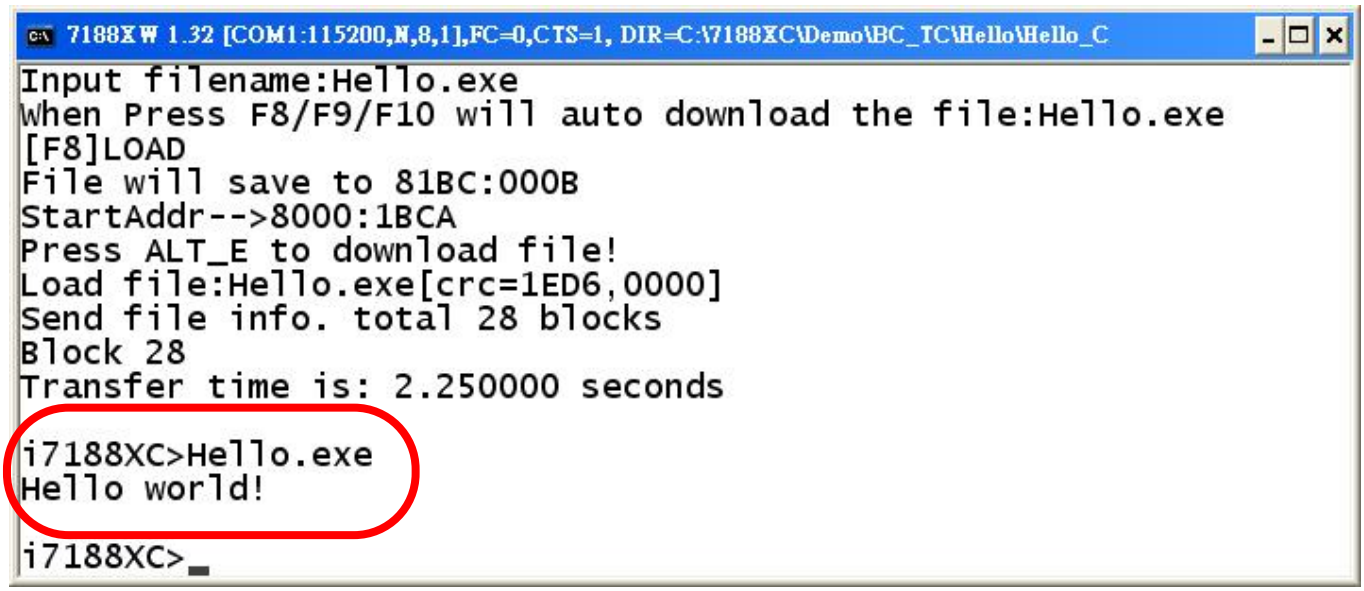
步骤 5: 执行 **7188xw.exe 应用程序出现如下界面：**



步骤 6: 点击 **<F2>**并且输入“**Hello.exe**” 并点击 **<Enter>**。

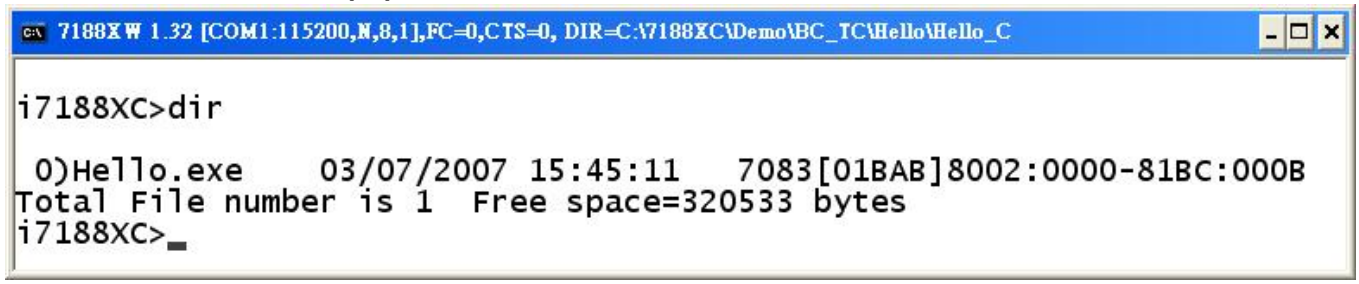

7188XW 1.32 [COM1:115200,N,8,1],FC=0,CTS=0, DIR=C:\7188XC\Demo\BC_TC\Hello\Hello_C
C:\7188XC\Demo\BC_TC\Hello\Hello_C>7188xw
7188x for WIN32 version 1.32 (2006/10/17)[By ICPDAS. Tim Tsai.]
[Begin Key Thread...]
Current set: Use COM1 115200,N,8,1
AutoRun:
Autodownload files: None
Current work directory="C:\7188XC\Demo\BC_TC\Hello\Hello_C"
original baudrate = 115200!
now baudrate = 115200!
i7188XC>
Input filename:Hello.exe
When Press F8/F9/F10 will auto download the file:Hello.exe

步骤 7: 点击 **<F8>** 来下载 Hello.exe 文件到 I-7188XC (D) 并执行这个程序。

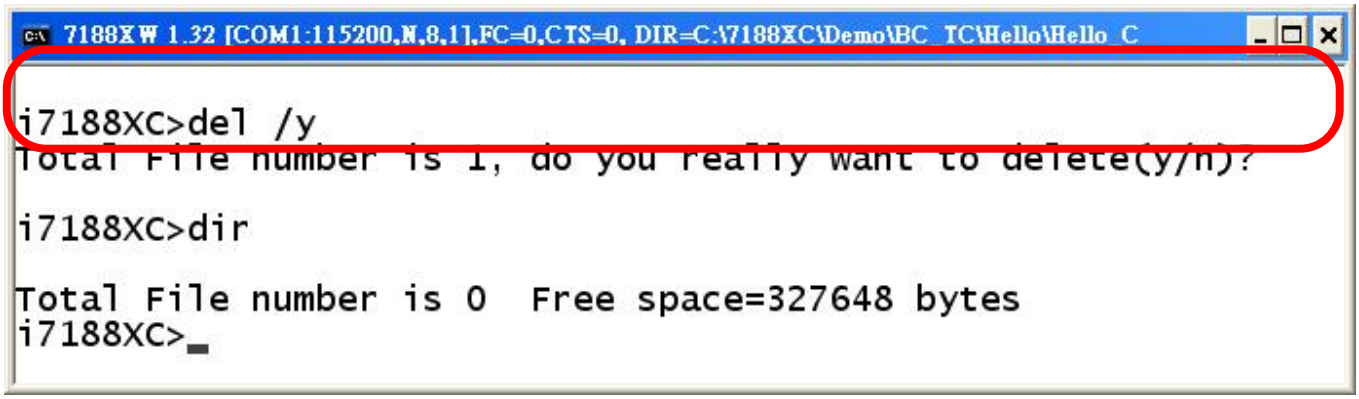


注意: 热键功能的描述如下所示：
F8: 下载一个文件到 FLASH 存储器并执行该文件
F9: 下载一个文件到 FLASH 存储器
F10: 下载一个文件到 SRAM 并执行这个程序

步骤 8: 输入 “dir” 并点击 **<Enter>**确认文件是否已经保存到了 I-7188XC (D)的 Flash 存储器。



步骤 9: 输入 “**del /y**” 并点击 **<Enter>**删除 I-7188XC (D)的 Flash 存储器上已经存储的文件。



注意: MiniOS7 只支持 **delete all** 命令，单独的文件无法被选择删除。

Step 10: 点击 **<Alt + X>** 推出 MiniOS7。

使用 7188xw.exe 程序升级 MiniOS7

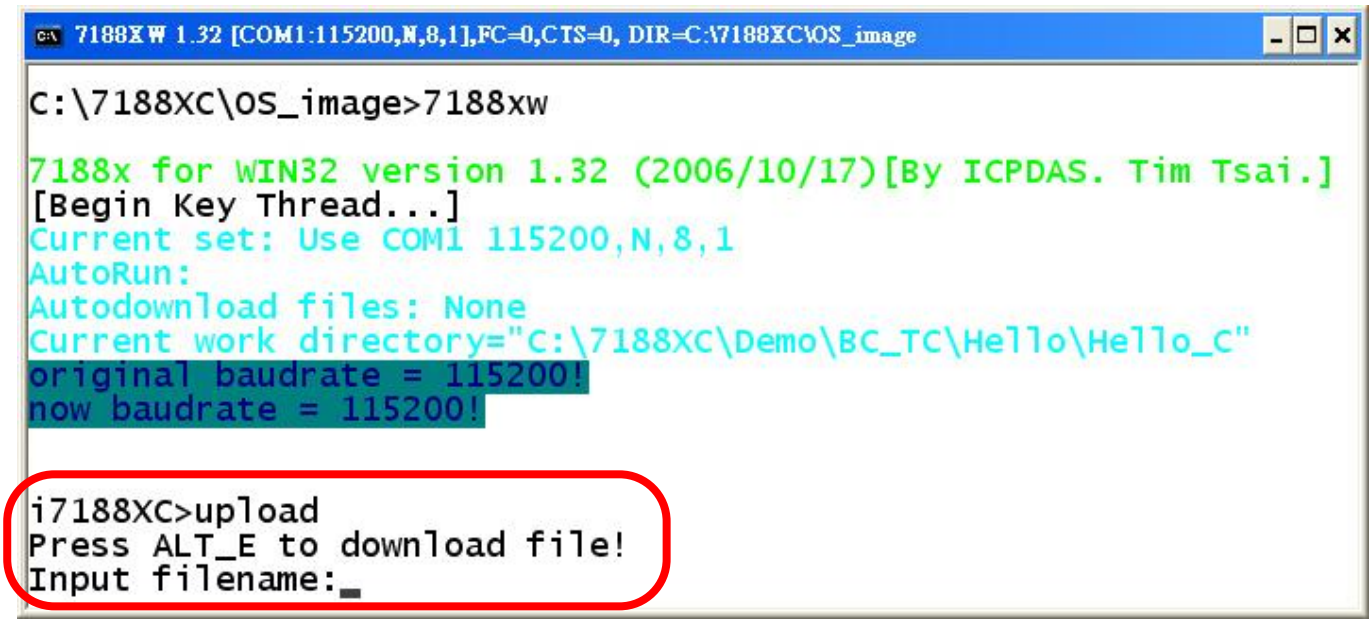
步骤 1: 使用 CA0910 电缆将 I-7188XC (D)连接到主机 PC 的 串口， 详细信息请参考 2.2 章的内容。

步骤 2: 确定 MiniOS7 最新的镜像文件
镜像文件名的格式是： **TTYYMMDD.img**
TT: 产品类型
YY: 镜像文件于哪年发布的
MM: 镜像文件发布的月份
DD: 镜像文件发布的日期

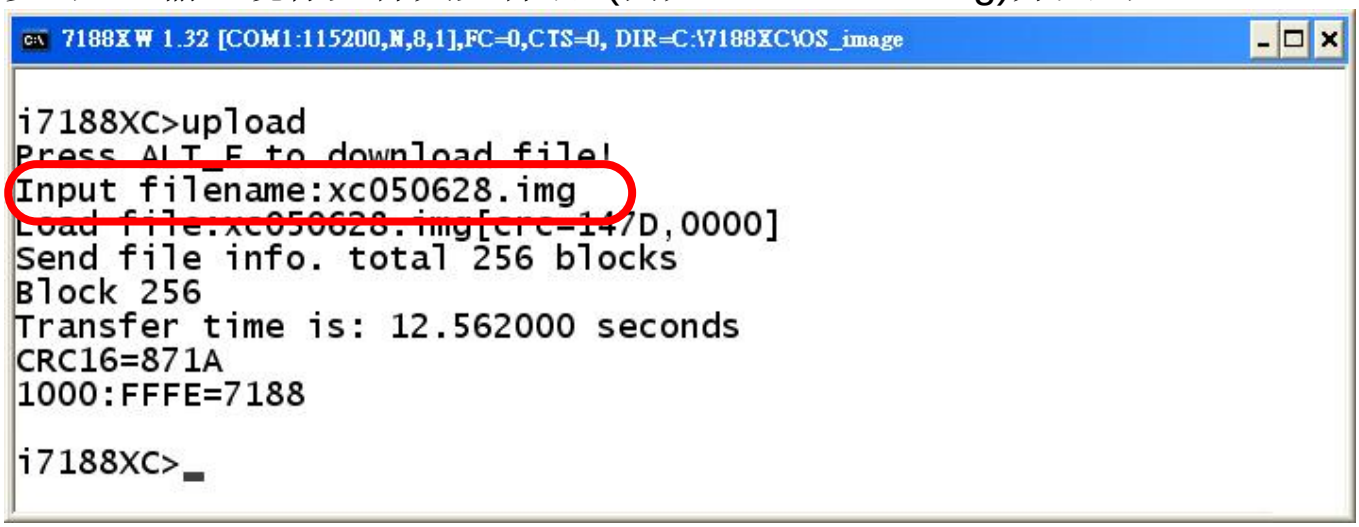
注意: MiniOS7 镜像文件可以如下方式获得： CD:\NAPDOS\MiniO7\.
最新版本 MiniOS7 可从如下网址下载：
http://ftp.icpdas.com/pub/cd/8000cd/napdos/7188xabc/7188xc/os_image/

步骤 3: 从主机 PC 打开镜像文件存储的文件夹，执行该文件夹中的 7188xw.exe 程序来连接主机 PC 和 I-7188XC (D) 控制器。

步骤 4: 使用 “**UPLOAD**”命令，在屏幕中出现“Press ALT_E to download file!”信息后点击**<ALT + E>** 。

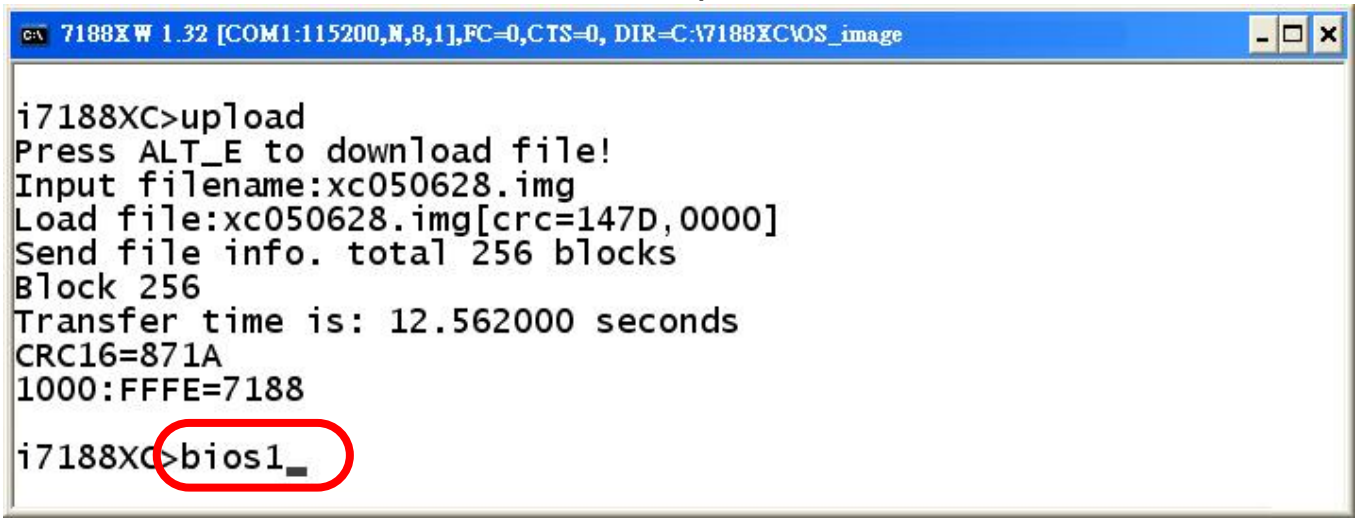


步骤 5: 输入镜像文件夹文件名 (例如: xb050701.img)并点击**<ENTER>**。

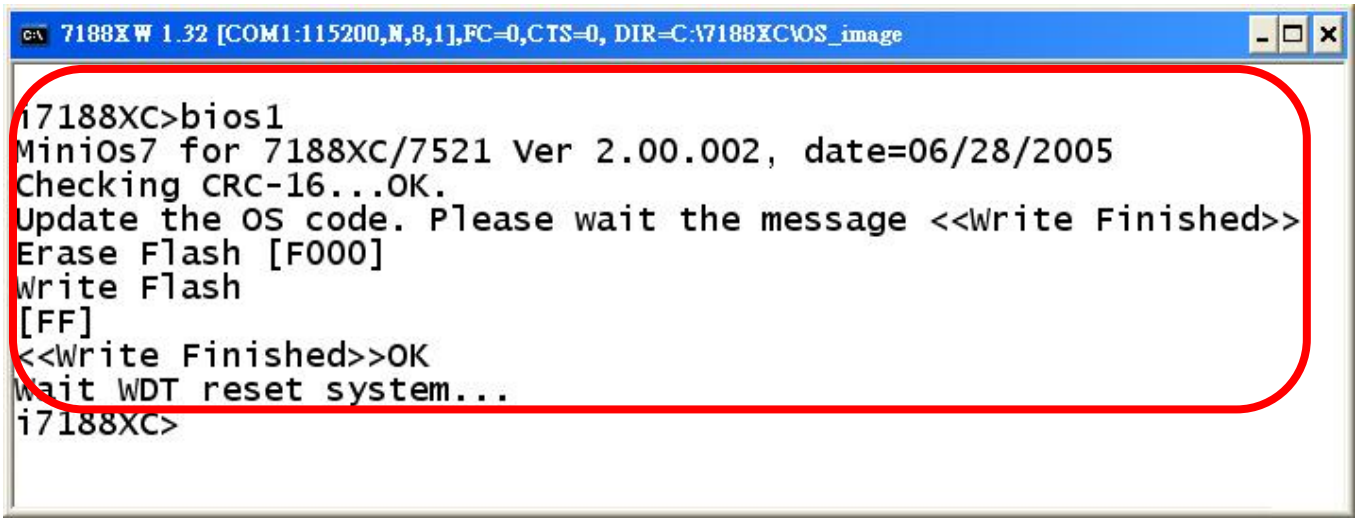


步骤 6: 等待升级完成 (镜像文件将存储在 SRAM)

步骤 7: 在 I-7188XC (D)的命令行中输入 “**bios1**”命令 (OS 将检查存储在 SRAM 中的镜像文件，并且显示版本信息，如果镜像文件是正确的则将被写入 Flash 存储器)



步骤 8: 升级 MiniOS7 将占用 10 秒时间。在升级完成后系统将自动重新启动，如果没有重新启动则需手动重启。



步骤 9: 输入“**ver**” 确认 MiniOS7 的版本号

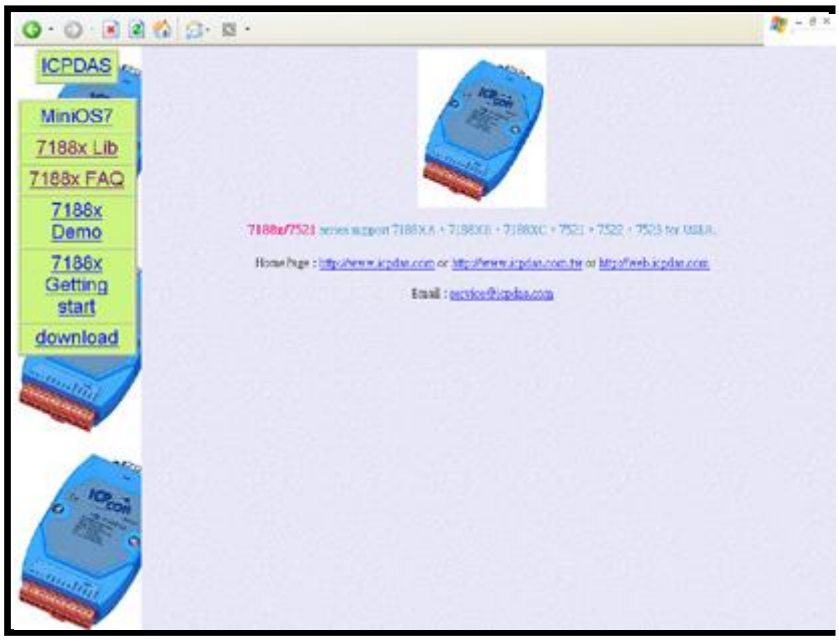
附录 C: 对照表

7188 和 7188X 系列的特征对照表:

	I-7188XA(D)	I-7188XB(D)	I-7188XC(D)	I-7188(D)
CPU 时钟	80188, 40MHz	80188, 40MHz	80188, 20MHz	80188, 40MHz
SRAM	512K	256K(I-7188XB) 512K(I-7188XB/512)	128K	256K
Flash Memory	512K	512K	256K (512K for ODM)	256K/512K
COM1	RS-232 带 modem 模式或 RS-485 含 self-tuner	RS-232 或 RS-485 含 self-tuner	RS-232 或 RS-485 含 self-tuner	RS-232 带 modem 模式或 RS-485
COM2	S-485 含 self-tuner，3000V 隔离	S-485 含 self-tuner	S-485 含 self-tuner	RS-485
COM3	RS-232 (TxD, RxD)	No COM	No COM	RS-232 (TxD, RxD)
COM4	RS-232 (Txd, Rxd)	No COM	No COM	RS-232 (TxD, RxD)
用户自定义引脚	0	14	3	0
Modem 控制	COM1	No	No	COM1
RTC	Yes	Yes	No	Yes
64 唯一硬件序列号	Yes	Yes	No	No
EEPROM	2K 字节	2K 字节	2K 字节	2K 字节
D/I (3.5V~30V)	2 通道	1 通道	2 通道	0
D/O (100mA, 30V)	2 通道	1 通道	3 通道	0
I/O 扩展总线	Yes	Yes	Yes	No
支持 ASIC Key	Yes	Yes	Yes	No
操作系统	MiniOS7	MiniOS7	MiniOS7	MiniOS7
编程语言	TC/MSC/BC	TC/MSC/BC	TC/MSC/BC	TC/MSC/BC
程序下载口	COM4	COM1	COM1	COM4

附录 D: 库函数清单

下面的表中列举了最常用的函数，更详细信息请参考 CD:\Napdos\MiniOS7\document\Lib_Manual_For_7188XABC\index_e.htm 或网页 http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/document/lib_manual_for_7188xabc/index_e.htm.



类型 1: 标准 IO

函 数	说 明
Kbhit	检查是否有键盘输入的数据在串口 COM1 的缓存
Getch	等待到接收从键盘输入的 1 个单字符
Ungetch	返回 1 个单独的字符到串口 COM1 的输入缓存
Putch	发送 1 个单独的字符到串口 COM1
Puts	发送字符串到串口 COM1
Scanf	类似 C 语言中的 Scanf 检索格式化的数据(无法用在 MSC/VC++，只能 TC/BC)
Print	类似 C 语言中的 printf 打印格式化的数据
ReadInitPin	读取 INIT*脚状态
LineInput	从 StdInput 输入单独的线
…更多…	更多基于标准 IO 用户函数信息详情请参考文件 7188xc.h 及 CD:\Napdos\MiniOS7\document\Lib

注意：函数 **Print** 和 **printCom** 不可同时在一个程序中使用.

- **Kbhit()**

  功能: 检测输入缓存是否有键盘输入的数据

  语法: **int Kbhit(void);**

  头文件: #include ”7188xc.h”

  描述: 检测输入缓存是否有键盘输入的数据

  返回值: 0: 无数据

  Other: 缓存中有数据，并且返回缓存中的数据。如果下个数据是 “\0”函数将返回 -1 (0xFFFF)。

  例程:

```
#include <7188xc.h>
void main()
{
   int quit=0, data;
   InitLib();
   Puts("\n\rPress any key to show ASCII ('Q' to quit):\n\r");
   while(!quit){
      if(Kbhit()){
         data=Getch();
         if(data=='Q') quit=1;
         Putch(data);
         Print(" ASCII is: %d\n\r", data);
         Puts("\n\rPress any key to show ASCII ('Q' to quit):\n\r");
      }
   }
}
```

- **Getch()**

  功能: 等待到接收从键盘输入的单字符

  语法: **int Getch(void);**

  头文件: #include ”7188xc.h”

  描述: 从输入缓存读取一个单字符，如果缓存没有输入数据，函数将一直等待从缓存中接收到数据为止

  返回值: 0 ~ 255

  例程: 更多详情请参考“Kbhit()”

- **Ungetch()**

  功能: 在输入缓存放入单字符。
  语法: **int Ungetch(int data);**
  头文件: #include ”7188xc.h”
  描述: 调用 Ungetch()时，如果没有数据在输入缓存，函数 Getch() 将在下周期继续调用，直到收到返回数据为止。
  数据： 0 ~ 255，如果数据 > 255，仅发送低字节。
  返回值: 如果成功返回 NoError，否则(例如缓存已满)返回 1。
  例程: 更多详情请参考“Kbhit()”

- **Putch()**

  功能: 在屏幕中显示单字符。
  语法: **void Putch(int data);**
  头文件: #include ”7188xc.h”
  描述: 数据: 0~255，如果数据 > 255，仅发送低字节。
  例程: 更多详情请参考“Kbhit()”

- **Puts()**

  功能: 在屏幕中显示字符串。
  语法: **void Puts(char *str);**
  头文件: #include ”7188xc.h”
  描述: Puts 将调用 Putch()来发送字符串
  str: 字符串的指针将被发送
  例程: 更多详情请参考“Kbhit()”

- **Scanf()**

  功能: 类似于 C 语言的 scanf()函数从输入区扫描一个字符 (此函数无法用于 MSC /VC++)
  语法: **int Scanf(char *fmt, ...);**
  头文件: #include ”7188xc.h”
  描述: 在输入信号扫描、转换和存储后返回数据，而不包含任何未被储存的投扫描域。

返回值: 0: 没有存储文件
EOF: 尝试读到字符串的结尾
例程: 参考 CD:\Napdos\7188XABC\7188XC\Demo\MSC\COM_Ports\C_Style_IO\

- **Print()**

功能: 在屏幕上打印一个格式符，用法类似于 C 语言的 printf() 函数。
语法: **int Print(char *fmt,...);**
头文件: #include ”7188xc.h”
描述: 该函数可用于取代函数 printf()，而函数 Print() 和 printf() 两者不同之处在于，函数 Print() 并不转换字符 “\n” 为 “\n” + “\r”。字符“\n”仅仅只发送代码 0x0A，而并非 0x0A + 0x0D，因此“\n\r”必须在执行“换行并置首”操作时，全部完成。显示信息将从 COM1(默认参数： 115200, N, 8, 1)发送。
输入参数: 请参考 C 语言的标准函数 printf() 。
返回值: 发送的字符数
例程: 更多详情请参考“Kbhit()”

类型 2: 串口

函 数	说 明
InstallCom	初始化串口驱动，串口号未赋值
InstallCom1	安装串口 COM1 驱动
InstallCom2	安装串口 COM2 驱动
RestallCom	卸载串口 COM 驱动。串口号未分派。
RestallCom1	卸载串口 COM1 驱动。串口号分派为 1。 RestallCom2 与该函数相同
IsCom	检查串口的缓存是否有数据。串口号未分派
IsCom1	检查串口 COM1 的缓存是否有数据
IsCom2…	检查串口 COM2 的缓存是否有数据
ClearCom	清除所有串口的缓存上的数据。 串口号未分派
ClearCom1	清除所有串口 COM1 的缓存上的数据
ClearCom2	清除所有串口 COM2 的缓存上的数据
ReadCom	读取串口的缓存上的数据。 串口号未分派
ReadCom1	读取串口 COM1 的缓存上的数据
ReadCom2	读取串口 COM2 的缓存上的数据
ToCom	向串口发送数据。串口号未分派
ToCom1	向串口 COM1 发送数据
ToCom2	向串口 COM2 发送数据
printCom	打印串口的缓存上存储的数据。串口号未分派
printCom1	打印串口 COM1 的缓存上存储的数据
printCom2	打印串口 COM2 的缓存上存储的数据
…更多…	更多有关串口函数请参考头文件 7188xc.h 及 CD:\Napdos\7188\miniOS7\manual\index.html

注意：Print 和 **printCom** 不能在程序中同时使用。

- **InstallCom()**

  功能: 安装串口驱动。

  语法: **int InstallCom(int port, unsigned long baud, int data, int parity, int stop);**

  头文件: #include ”7188xc.h”

描述: 安装串口驱动。串口号未分派，但可以通过“port”参数进行修改。
port: 分派端口号
baud: 波特率， I-7188XC(D) 默认是 115200

例程：

```
#include <7188xc.h>
void main()
{
  int quit=0, data, i, port=1;   /*port=1, uses COM1*/
  InitLib();
  InstallCom(port,115200,8,0,1);   /*installs the COM port driver*/
  for(i=0; i<10; i++)
    printCom(port,"Test %d\n\r",i);   /*prints data to the COM Port*/
  while(!quit) {
    if(IsCom(port)) {   /*checks if any data is in the COM Port buffer*/
      data=ReadCom(port);   /*reads data from the COM Port buffer*/
      ToCom(port,data);   /*sends data to the COM Port buffer*/
      ClearCom(port);   /*clears all the data in the COM Port buffer*/
      if(data=='Q') quit=1;   /*if 'Q' is received, exit the program*/
    }
  }
  RestoreCom(port);   /*uninstalls the driver for COM Port */
}
```

- **InstallCom1()**

功能: 安装串口 COM1 驱动。

语法: **int InstallCom1(unsigned long baud, int data, int parity, int stop);**

头文件: #include ”7188xc.h”

描述: 安装串口驱动。
波特率， I-7188XC(D) 默认是 115200

例程:

```
#include <7188xc.h>
void main()
{
  int quit=0,data;
  InitLib();
```

```
    InstallCom1(115200,8,0,1);    /*install the driver for COM1*/
    while(!quit) {
          if(IsCom1()) {    /*checks if any data is in the COM1 buffer*/
             data=ReadCom1();    /*reads data from COM1*/
             ToCom1(data);    /*sends data to COM1*/
             if(data=='q') quit=1;    /*if 'q' is received, exit the program*/
          }
       }
    RestoreCom1();    /*uninstalls the driver for COM1*/
}
```

- **RestoreCom()**

  功能: 卸载串口驱动。串口号未分派。

  语法: **int RestoreCom(int port);**

  头文件: #include "7188xc.h"

  描述: 卸载串口驱动。串口号未分派，但可以通过"port" 参数进行修改。

  port: 分派串口号

  例程: 更多详情请参考"InstallCom()"

- **RestoreCom1()**

  功能: 卸载串口 COM1 驱动.

  语法: **int RestoreCom1(void);**

  头文件: #include "7188xc.h"

  描述: 卸载串口 COM1 驱动

  例程: 更多详情请参考"InstallCom()"

- **IsCom()**

  功能: 检查串口的缓存是否有数据。串口号未分派

  语法: **int IsCom(int port);**

  头文件: #include "7188xc.h"

  描述: 检查串口的缓存是否有数据。串口号未分派，但可以通过"port" 参数进行修改。

  port: 分派串口号

  例程: 更多详情请参考"InstallCom()"

- **IsCom1()**

  功能: 检查串口 COM1 的缓存是否有数据。
  语法: **int IsCom1(void);**
  头文件: #include ”7188xc.h”
  描述: 检查串口 COM1 的缓存是否有数据。
  例程: 更多详情请参考“InstallCom()”

- **ReadCom()**

  功能: 从串口的换粗读取数据，串口号未分派。
  语法: **int ReadCom(int port);**
  头文件: #include ”7188xc.h”
  描述: 从串口的换粗读取数据，串口号未分派，但可以通过 “port” 参数修改
  port: 分配的串口。
  例程: 更多详情请参考“InstallCom()”

- **ReadCom1()**

  功能: 从串口 COM1 的缓存读取数据。
  语法: **int ReadCom1(void);**
  头文件: #include ”7188xc.h”
  描述: 从串口 COM1 的缓存读取数据
  例程: 更多详情请参考“InstallCom()”

- **ClearCom()**

  功能: 清除所有串口的缓存上的数据。串口号未分派
  语法: **int ClearCom(int port);**
  头文件: #include ”7188xc.h”
  描述: 清除所有串口的缓存上的数据。 串口号未分派，但是可通过“port” 参数修改
  port: 分配串口号。
  例程: 更多详情请参考“InstallCom()”

- **ClearCom1()**

  功能: 清除串口 COM1 的缓存上的所有数据。
  语法: **int ClearCom1(void);**
  头文件: #include ”7188xc.h”
  描述: 清除串口 COM1 的缓存上的所有数据。
  例程: 更多详情请参考“InstallCom()”

- **ToCom()**

  功能: 向串口发送数据。串口号未分派。
  语法: **int ToCom(int port);**
  头文件: #include ”7188xc.h”
  描述: 向串口发送数据。串口号未分派，但是可以通过“port” 参数修改
  port: 分配串口号。
  例程: 更多详情请参考“InstallCom()”

- **ToCom1()**

  功能: 向串口 COM1 发送数据。
  语法: **int ToCom1(void);**
  头文件: #include ”7188xc.h”
  描述: 向串口 COM1 发送数据。
  例程: 更多详情请参考“InstallCom()”

- **printCom()**

  功能: 打印串口的缓存上存储的数据。串口号未分派
  语法: **int printCom(int port,char *fmt,...);**
  头文件: #include ”7188xc.h”
  描述: 打印串口的缓存上存储的数据。串口号未分派，但是可以通过“port” 参数修改.。可生产格式化的数据输出，类似于 C 语言的 printf()函数。
  例程: 更多详情请参考“InstallCom()”

- **printCom1()**

  功能: 打印串口 COM1 的缓存上存储的数据

  语法: **int printCom_1(char *fmt,...);**

  头文件: #include ”7188xc.h”

  描述: 打印串口的缓存上存储的数据，可生产格式化的数据输出，类似于 C 语言的 printf()函数。

  例程: 该函数类似于 printCom()。更多详情请参考“InstallCom()”。

类型 3: EEPROM

函 数	说 明
EE_WriteEnable	设置 EEPROM 为可写入模式
EE_MultiWrite	向 EEPROM 写入数据
EE_WriteProtect	设置 EEPROM 为写保护模式
EE_MultiRead	读取 EEPROM 中的数据
…更多…	更多有关 EEPROM 函数请参考头文件 7188xc.h 及随机赠送 CD 中的用户手册，地址： CD:\Napdos\MiniOS7\document\Lib_Manual_For_7188XABC\index.htm

- **EE_WriteEnable ()**

  功能: 设置 EEPROM 为可写入模式。

  语法: **void EE_WriteEnable (void);**

  头文件: #include ”7188xc.h”

  描述: 设置 EEPROM 为可写入模式。EEPROM 默认为写保护模式。在向 EEPROM 中写入数据前必需调用 EE_WriteEnable()函数。

  例程:

```
#include <7188xc.h>
void main()
{
   Int data=55, data2;
   InitLib();
   EE_WriteEnable ();
   EE_MultiWrite(1,10,1,&data);
   EE_WriteProtect();
   EE_MultiRead(1,10,1,&data2);
   Print("data=%d, Data2=%d", data,data2);
}
```

- **EE_MultiWrite ()**

  功能 向 EEPROM 写入数据

  语法: **int EE_MultiWrite(int Block,unsigned Addr,int no,char *Data);**

  头文件: #include ”7188xc.h”

  描述: 向 EEPROM 写入数据。

Block: 0 到 7 (Block 共有 8 个).
Addr: 0 到 255 (每个 block 可包含 256 bytes).
no: 1 到 16
Data:存储在缓存的起始地址的数据

返回值: 操作成功，返回 NoError.
操作失败，返回-1，表示 EEPROM 忙、 Block 损坏或者地址错误。

例程: 更多详情请参考“EE_WriteEnable()”

- **EE_WriteProtect ()**

功能: 设置 EEPROM 为写保护模式。

语法: **void EE_WriteProtect(void);**

头文件: #include ”7188xc.h”

描述: 设置 EEPROM 到写保护模式，EEPROM 默认为写保护模式。在向 EEPROM 中写入数据前必需调用 EE_WriteEnable() 函数。在写完数据后，推荐使用 EE_WriteProtect () 将 EEPROM 恢复为写保护模式。

Example: 更多详情请参考“EE_WriteEnable()”

- **EE_MultiRead ()**

功能: 读取 EEPROM 中的数据。

语法: **int EE_MultiRead(int StartBlock,unsigned StartAddr,int no,char *databuf);**

头文件: #include ”7188xc.h”

描述: 从 EEPROM 读取多位字节的数据。
StartBlock: 0 到 7 (总共 8 个 block).
StartAddr: 0 到 255 (每个 block 可包含 256 bytes)
no: 1 到 2048
databuf: 存储数据的地址

返回值: 操作成功，返回 NoError。操作失败，返回-1，表示 EEPROM 忙、 Block 损坏或者地址错误。

例程: 更多详情请参考“EE_WriteEnable()”

类型 4: NVRAM 及 RTC

函 数	说 明
ReadNVRAM	从 NVRAM 读取数据
WriteNVRAM	在 NVRAM 中写入数据
GetTime	自 RTC 中取得系统时间
SetTime	为 RTC 设置系统时间
GetDate	自 RTC 取得系统日期
SetDate	为 RTC 设置系统日期
GetWeekDay	自 RTC 取得星期信息
…更多…	更多有关 NVRAM 及 RTC 函数请参考头文件 7188xc.h 及随机赠送 CD 中的用户手册，地址：CD:\Napdos\minios7\document\lib_manual_for_7188xabc\index.htm

- **ReadNVRAM()**

  功能: 从 NVRAM 读取数据。

  语法: **int ReadNVRAM(int addr);**

  头文件: #include ”7188xc.h”

  描述: 从 NVRAM 读取数据。
  addr: 0 到 30, 共 31 个字节

  返回值: 成功则返回在指定地址存储的数据 (0-255) ，
  失败则返回 AddrError (-9)。

  例程:

```
#include <7188xc.h>
void main()
{
   int data=55, data2;
   InitLib();
   WriteNVRAM(0,data);
   data2=ReadNVRAM(0);    /* now data2=data=55 */
   Print("data=%d, data2=%d",data,data2);
}
```

- **WriteNVRAM()**

  功能: 在 NVRAM 中写入数据。

  语法: **int WriteNVRAM(int addr, int data);**

  头文件: #include ”7188xc.h”

描述: 在 NVRAM 中写入一个字节的数据。
addr: 0-30.
data: 1 个字节的数据(0-255).
如果数据>255，仅有低位的数据会被写入 NVRAM。
返回值: 成功则返回 NoError，
失败则返回 AddrError (-9)。
例程: 更多详情更多详情 “ReadNVRAM()”

- **GetTime()**

功能: 自 RTC 中取得系统时间
语法: **void GetTime(int *hour, int *minute, int *sec);**
头文件: #include ”7188xc.h”
描述: hour: 用于存储小时数据的地址 (0-23) 。
minute: 用于存储分钟数据的地址(0-59)。
sec: 用于存储秒数据的地址 (0-59) 。
例程:

```
#include <7188xc.h>
void main()
{
  int year, month, day, hour, min, sec, wday;
  InitLib();
  SetDate(2006,1,12);   /*sets the system date for the RTC*/
  SetTime(15,35,50);   /*sets the system time for the RTC*/
  SetWeekDay(4);   /*sets the system day of the week for the RTC*/
  GetDate(&year,&month,&day);   /*reads the system date from the RTC*/
  GetTime(&hour,&min,&sec);   /*reads the system time from the RTC*/
  wday=GetWeekDay();
  Print("Date=%02d/%02d/%04d(%d) Time=%02d:%02d:%02d\n\r",
  month,day,year,wday,hour,min,sec);
}
```

- **SetTime()**

功能: 为 RTC 设置系统时间
语法: **int SetTime(int hour,int minute,int sec);**
头文件: #include ”7188xc.h”
描述: hour: 0-23.
minute: 0-59.

sec: 0-59.

返回值: 成功则返回 NoError，失败则返回 TimeError (-19)。

例程: 更多详情请参考“GetTime()”

- **GetDate()**

  功能: 自 RTC 取得系统日期

  语法: **void GetDate(int *year,int *month,int *day);**

  头文件: #include ”7188xc.h”

  描述: year: 2000-2080
  month: 1-12
  day: 1-31

  例程: 更多详情请参考“GetTime()”

- **SetDate()**

  功能: 为 RTC 设置系统日期

  语法: **int SetDate(int year,int month,int day);**

  头文件: #include ”7188xc.h”

  描述: year: 2000-2080
  month: 1-12
  day: 1-31
  返回值: 成功则返回 NoError，
  失败返回 DateError (-18).

  例程: 更多详情请参考“GetTime()”

- **GetWeekDay()**

  功能: 自 RTC 取得星期信息

  语法: **int GetWeekDay(void);**

  头文件: #include ”7188xc.h”

  描述: 自 RTC 取得星期信息

  返回值: 0=>周日
  1-6=>周一到周六

  例程: 更多详情请参考“GetTime()”

注意: GetWeekDay() 并非验证星期几是否正确，只是从 RTC 读取数据。当使用 MiniOS7 的“date” 命令去设置日期时，MiniOS7 将计算正确的星期信息并设置 RTC 时间。当调用 SetDate()函数时，该函数也会计算正确的星期信息并设置 RTC 时间。然而调用 SetWeekDay() 函数时，该函数会自己计算正确星期数。

类型 5: Flash Memory

函 数	说 明
FlashReadId	读取 Flash 存储器的信息
FlashErase	擦除 Flash 存储器的 1 个扇区
FlashWrite	在 Flash 存储器中写入数据的 1 个字节
FlashRead	自 Flash 存储器中读取数据的 1 个字节
…更多…	更多有关 Flash Memory 函数请参考头文件 7188xc.h 及随机赠送 CD 中的用户手册，地址：CD:\Napdos\minios7\document\lib_manual_for_7188xabc\index.htm

I-7188XC(D)系列模块具有 256K 字节的 Flash 存储器。MiniOS7 占用其中最后的 64K 字节，其它的空间可用来存储用户程序和数据。

应用开发人员可应用这些函数来向 Flash 存储器写入数据。当写入数据时，数据必须从“1” 到 “0”写入，而不能从“0” 到“1”。所以，在向 Flash 存储器写入数据前必须进行擦除动作，擦除操作将使数据转化成 0xFF，这样所有的数据为将变成“1”，只有这样数据才能被写入。FlashErase() 函数每次擦除 1 个扇区的数据 (64K bytes)。

- **FlashReadId()**

  功能: 读取 Flash 存储器的信息
  语法: **int FlashReadId(void);**
  头文件: #include ”7188xc.h”
  描述: 读取 Flash 存储器的设备代码（高位）和生产代码（低位）。
  返回值: 0xA4C2 (MXIC 29f040), 0xA401 (AMD 29f040)
  例程: 更多详情请参考 CD:\Napdos\7188XABC\7188XC\Demo\BC_TC\Memory\

- **FlashErase()**

  功能: 擦除 Flash 存储器的 1 个扇区
  语法: **int FlashErase(unsigned seg);**
  头文件: #include ”7188xc.h”
  描述: 擦除 Flash 存储器的 1 个扇区(64K bytes)。 所有这个扇区

上数据的值将变成 0xFF。
seg: 0x8000, 0x9000, 0xA000, 0xB000, 0xC000, 0xD000 or 0xE000.

返回值: On success, returns NoError (0).
On error, returns TimeOut (-5).

例程: 详细信息请参考 “\Demo\BC_TC\Memory\Flash\FLASH.C”

注意: 0xF000 区段用来存储 MiniOS7 操作系统，如果尝试擦除该区段，FlashErase() 函数将不会起作用。

- **FlashWrite()**

功能: 在 Flash 存储器中写入数据的 1 个字节

语法: **int FlashWrite(unsigned int seg, unsigned int offset, char data);**

头文件: #include ”7188xc.h”

描述: seg: 0x8000, 0x9000, 0xA000, 0xB000, 0xC000, 0xD000 及 0xE000
offset: 0 to 65535 (0xffff)
data: 0 to 255 (8 位数据)

返回值: 成功则返回 NoError(0)
失败返回 TimeOut(-5)或 SegmentError(-12).

例程:

```
#include <7188xc.h>
void main()
{
  int data=0xAA55, data2;
  char *dataptr;
  InitLib();
  dataptr=(char *)&data;
  FlashWrite(0xd000,0x1234, *dataptr++);      /*writes data to the Flash
                                               memory*/
  FlashWrite(0xd000,0x1235, *dataptr);
  dataptr=(char *)&data2;      /*reads data from the Flash memory*/
  *dataptr=FlashRead(0xd000, 0x1234);
  *(dataptr+1)=FlashRead(0xd000, 0x1235);
}
```

注意: 当向 Flash 存储器写入数据时，数据位只能由 1 变到 0 ，FlashWrite()函数不会检查 Flash 存储器状态，只会写入数据，如果尝试将数据由 0 变成 1，将会出现 TimeoutError 错误信息。在调用 FlashErase() 函数后数据才可被写入。

- **FlashRead()**

  功能: 自 Flash 存储器中读取数据的 1 个字节。

  语法: **int FlashRead(unsigned int seg, unsigned int offset);**

  头文件: #include ”7188xc.h”

  描述: seg: 0-65535(0xffff).
  offset: 0 to 65535(0xffff).

  返回值: FlashRead() 只会返回地址的值
  seg:区距。地址可以是 SRAM、Flash 存储器或其它地址(都会返回 0xff)。

  例程: 更多详情请参考 “FlashWrite()”

类型 6: 计时器和 Watchdog 计时器

函 数	说 明
TimerOpen	开启计时器功能
TimerClose	关闭计时器功能
TimerResetValue	设置计时器为 0
TimerReadValue	读取主要计时点
DelayMs	在指定的时间段插入延时，时间单位为 ms，使用系统计时点
Delay	在指定的时间段插入延时，时间单位为 ms，使用 CPU 的 Timer 1 特征
Delay_1	在指定的时间段插入延时，时间单位为 0.1 ms, 使用 CPU 的 Timer 1 特征
Delay_2	在指定的时间段插入延时，时间单位为 0.01 ms, 使用 CPU 的 Timer 1 特征
StopWatchStart	使用 1 个 StopWatch 通道
StopWatchReset	重置 StopWatch 的值为 0
StopWatchStop	停止 StopWatch 通道
StopWatchPause	暂停 StopWatch
StopWatchContinue	重新启动 StopWatch
StopWatchReadValue	读取当前 StopWatch 值
CountDownTimerStart	开始使用 CountDownTimer
CountDownTimerReadValue	读取当前 CountDownTimer 值
InstallUserTimer	装入用户 timer 功能，该函数在 1 ms 内调用
InstallUserTimer1C	在中断 0x1c 装入用户 timer 功能，系统 timer 在 55 ms 内使用该中断点
EnableWDT	使用 Watchdog timer
DisableWDT	停止使用 Watchdog timer
RefreshWDT	刷新 Watchdog timer
…更多…	还有更多计时器和 Watchdog 计时器函数提供，更多详情请参考 7188xc.h 头文件或参考用户手册 CD:\Napdos\minios7\document\lib_manual_for_

	7188xabc\index.htm

- **TimerOpen()**

  功能: 打开 function 功能

  语法: **int TimerOpen(void);**

  头文件: #include ”7188xc.h”

  描述: 在使用任何 timer 功能前必须调用 TimerOpen(函数。

  返回值: 成功则返回 NoError，如果 Timer 已经打开则返回 1。

  例程:

```
#include <7188xc.h>
void main()
{
   unsigned long time;
   int quit=0;
   InitLib();
   Print("\n\rPress any key to start the timer");
   Print("\n\rthen Press '0' to Reset the timer, ’1’~’4’ to insert a delay, 'q' to
            quit\n\r");
   Getch();
   TimerOpen();   /*open the timer function*/
   while(!quit){   /*sets the key function*/
      if(Kbhit()){
         switch(Getch()){
            case '0':
               TimerResetValue();   /*reset the timer*/
               break;
            case '1':
               DelayMs(1000);   /*delay unit is ms, uses system timeticks. */
               break;
            case '2':
               Delay(1000);   /*delay unit is ms, uses the Timer 1 feature of the
                                 CPU. */
               break;
            case '3':
               Delay_1(1000);   /*delay unit is 0.1 ms, uses the Timer 1 feature
                                   of the CPU.*/
               break;
            case '4':
               Delay_2(1000);   /*delay unit is 0.01 ms, uses the Timer 1
                                   feature of the CPU .*/
```

```
            break;
          case 'q':
            quit=1;
            break;
        }
      }
      time=TimerReadValue();   /*reads the timer*/
      Print("\r\nTime=%8.3f sec", 0.001*time);
    }
    TimerClose();   /*closes the timer function*/
}
```

- **TimerClose()**

  功能: 停止使用计时器功能。
  语法: **int TimerClose(void);**
  头文件: #include ”7188xc.h”
  描述: 在使用 OpenTimer() 函数前需先调用 TimerClose() 函数。
  返回值: 总返回 NoError。
  例程: 更多详情请参考“TimerOpen()”

- **TimerResetValue()**

  Function: 将计时器复位为 0。
  语法: **void TimerResetValue(void);**
  头文件: #include ”7188xc.h”
  描述: 重置计时点为 0。
  例程: 更多详情请参考“TimerOpen()”

- **TimerReadValue()**

  功能: 读取主要计时点
  语法: **unsigned long TimerReadValue(void);**
  头文件: #include ”7188xc.h”
  描述: 读取主要计时点，时间单位为 ms.当 TimerOpen,TimerReset 函数被调用，计时点将变成 0。
  例程: 更多详情请参考“TimerOpen()”

- **DelayMs()**

  功能: 在指定的时间段插入延时，时间单位为 ms，使用系统计

时点

语法: **void DelayMs(unsigned t);**

头文件: #include ”7188xc.h”

描述: 延时时间单位为 ms.
t: 延时时间

例程: 更多详情请参考“TimerOpen()”

- **Delay()**

功能: 在指定的时间段插入延时，时间单位为 ms，使用 CPU 的计时器 Timer 1

语法: **void Delay(unsigned ms);**

头文件: #include ”7188xc.h”

描述: 在指定的时间段插入延时，时间单位为 ms，使用 CPU 的 Timer 1 特征
ms: 延时时间

例程: 更多详情请参考“TimerOpen()”

- **Delay_1()**

功能: 在指定的时间段插入延时，时间单位为 0.1ms，使用 CPU 的计时器 Timer 1

语法: **void Delay_1(unsigned ms);**

头文件: #include ”7188xc.h”

描述: 在指定的时间段插入延时，时间单位为 0.1ms，使用 CPU 的 Timer 1 特征
ms: 延时时间

例程: 更多详情请参考“TimerOpen()”

- **Delay_2()**

功能: 在指定的时间段插入延时，时间单位为 0.01，使用 CPU 的计时器 Timer 1

语法: **void Delay_2(unsigned ms);**

头文件: #include ”7188xc.h”

描述: 在指定的时间段插入延时，时间单位为 0.01，使用 CPU 的 Timer 1 特征

ms: 延时时间

例程: 请参考“TimerOpen()”

- **StopWatchStart()**

  功能: 使用 1 个 StopWatch 通道并重置 StopWatch 值为 0。

  语法: **int StopWatchStart(int channel);**

  头文件: #include ”7188xc.h”

  描述: 系统 timer ISR 将在每 1 ms 为 StopWatch 值增加 1。
  channel: 0-7, a total of 8 channels.

  返回值: 成功则返回 NoError，
  如果超出通道范围返回 ChannelError (-15)。

  例程:

```
#include <7188xc.h>
void main(void)
{
   unsigned long value;
   int quit=0; InitLib();
   Print("\n\rTest the StopWatch ... Press 'q' to quit\n\r ");
   TimerOpen();
   StopWatchStart(0);    /*start using the StopWatchStart function*/
   while(!quit){
      if(Kbhit()){ switch(Getch()){ case 'q': quit=1; break; } }
      StopWatchReadValue(0,&value);
      Print("SWatch=%d \r",value);
      if(value==2000){
         StopWatchPause(0);
         DelayMs(2000);
         StopWatchContinue(0); }
         if(value==4000){
            StopWatchStop(0);
            DelayMs(2000);
            StopWatchReset(0);
            StopWatchStart(0);
         }
      }
      TimerClose();
   }
```

- **StopWatchReset()**

  功能: 将 StopWatch 复位为 0
  语法: **int StopWatchReset(int channel);**
  头文件: #include ”7188xc.h”
  描述: channel: 0-7，共 8 个 channel。
  返回值: 成功则返回 NoError，
  如果超出通道范围返回 ChannelError (-15)。
  例程: 更多详情请参考“StopWatchStart()”

- **StopWatchStop()**

  Function: 停止 StopWatch 通道
  语法: **int StopWatchStop(int channel);**
  头文件: #include ”7188xc.h”
  描述: 系统 timer ISR 将停止增加 StopWatch 的值，
  channel: 0-7，共 8 个 channel。
  返回值: 成功则返回 NoError，
  如果超出通道范围返回 ChannelError (-15)。
  例程: 更多详情请参考“StopWatchStart()”

- **StopWatchPause()**

  功能: 暂停使用 StopWatch
  语法: **int StopWatchPause(int channel);**
  头文件: #include ”7188xc.h”
  描述: 在调用 StopWatchPause()用，必须调用
  StopWatchContinue() 函数恢复时间计时。
  channel:0-7，共 8 个 channel。
  返回值: 成功则返回 NoError，
  如果超出通道范围返回 ChannelError (-15)。
  例程: 更多详情请参考“StopWatchStart()”

- **StopWatchContinue()**

  功能: 重新启动 StopWatch。
  语法: **int StopWatchContinue(int channel);**

头文件: #include ”7188xc.h”
描述: channel:0-7，共 8 个 channel。
返回值: 成功则返回 NoError，
如果超出通道范围返回 ChannelError (-15)。
例程: 更多详情请参考“StopWatchStart()”

- **StopWatchReadValue()**

Function: 读取当前 StopWatch 值
语法: **int StopWatchReadValue(int channel,unsigned long *value);**
头文件: #include ”7188xc.h”
描述: 这个值表示自 StopWatchStart()或 StopWatchReset()函数被调用后占用的时间。
channel: 0-7，共 8 个 channel。
返回值: 成功则返回 NoError，
如果超出通道范围返回 ChannelError (-15)。
例程: 更多详情请参考“StopWatchStart()”

- **CountDownTimerStart()**

功能: 开始使用 CountDownTimer。
语法: **int CountDownTimerStart(int channel,unsigned long count);**
头文件: #include ”7188xc.h”
描述: channel: 0-7，共 8 个 channel。
count: 已经计数的数值
返回值: 成功则返回 NoError，
如果超出通道范围返回 ChannelError (-15)。

例程:

```
#include <7188xc.h>
void main(void)
{
  unsigned long value;
  int quit=0;
  InitLib();
  Print("\n\rTest the CountDownTimer...");
```

```
    Print("\n\rPress 'q' to quit\n\r");
    TimerOpen();
    CountDownTimerStart(0,1000);   /*use the CountDownTimer*/
    while(!quit){
      if(Kbhit()&&(Getch()=='q')) quit=1;
      CountDownTimerReadValue(0,&value);   /*reads the
                                             CountDownTimer*/
      Print("Test CountDown=%d\r",value);
      if(value==0)
        CountDownTimerStart(0,1000);   /*restarts the CountDownTimer*/
    }
    TimerClose();
}
```

- **CountDownTimerReadValue()**

  功能: 读取当前 CountDownTimer(count)值。

  语法: **int CountDownTimerReadValue(int channel,unsigned long *value);**

  头文件: #include ”7188xc.h”

  描述: 如果值为 0 代表时间已经终止。
  channel: 0-7，共 8 个 channel。
  value: 值存储位置的指针

  返回值: 成功则返回 NoError，
  如果超出通道范围返回 ChannelError (-15)。

  例程: 请参考 “CountDownTimerStart ()”

- **InstallUserTimer()**

  功能: 启动用户 timer 功能，该函数在每 1ms 内调用

  语法: **void InstallUserTimer(void (*fun)(void));**

  头文件: #include ”7188xc.h”

  描述: fun:用户函数的指针
  该函数无法使用输入变量，并且不会提供返回值

  例程:

```
#include <7188xc.h>
int Data[3]={0,0,0};
void MyTimerFun(void)   /*custom timer function*/
{
   static int count[3]={0,0,0};
   int i;
   for(i=0;i<3;i++){
      Print("count[%d]=%d\r",i,count[i]);
      count[i]++;
   }
   if(count[0]>=200){   /*LCD lamp1 blinks each 200 units*/
      count[0]=0;
      if (Data[0]==0) Data[0]=1;
      else Data[0]=0;
      lamp(1,1,Data[0]);
   }
   if(count[1]>=500){   /*LCD lamp2 blinks each 500 units*/
      count[1]=0;
      if (Data[1]==0) Data[1]=1;
      else Data[1]=0;
      lamp(2,1,Data[1]);
   }
   if(count[2]>=1000){   /*LCD lamp3 blinks each 1000 units*/
      count[2]=0;
      if (Data[2]==0) Data[2]=1;
      else Data[2]=0;
      lamp(3,1,Data[2]);
   }
}
void main(void)
{
   int quit=0;
   Print("\n\rTest the LCD lamp blink using user timer ");
   Print("\n\rPress 'q' to quit\n\r");
   InitLib(); InitLCD();   /*initial Lib & LCD*/
   ClrScrn();   /*clear the LCD screen*/
   TimerOpen();   /*open timer function*/
   InstallUserTimer(MyTimerFun); /*install and call user timer */
   while(!quit){
      if(Kbhit() && Getch()=='q') quit=1;
   }
   TimerClose();}
```

- **InstallUserTimer1C()**

  功能: 在中断 0x1c 装入用户计时器功能，系统计时器在 55 ms 内使用该中断点

  语法: **void InstallUserTimer1C(void (*fun)(void));**

  头文件: #include ”7188xc.h”

  描述: fun: 用户函数的指针。该函数无法使用输入变量，并且不会提供返回值

  例程: 请参考“InstallUserTimer()”

- **EnableWDT()**

  功能: 启用 WatchDog timer.

  语法: **void EnableWDT(void);**

  头文件: #include ”7188xc.h”

  描述: WatchDog Timer (WDT) 一直处于激活状态，并由系统 Timer ISR 不断的刷新。当用户程序调用 EnableWDT() 函数时，系统 timer ISR 将停止刷新 WDT，直到在程序中调用 RefreshWDT()函数，否则系统将被 WDT 重新启动。. MiniOS7 2.0 下 WDT 的超时时间为 0.8 秒。

  例程:

```
#include"7188xc.h"
void main(void)
{
   int quit=0,k;
   InitLib();
   if(IsResetByWatchDogTimer())   /*test whether the system has been
                                    reset by the WDT*/
      Print("reset by WatchDog timer\n\r");
   EnableWDT();   /*after callng EnableWDT, Refresh WDT must be called
                    within 0.8s*/
   while(!quit){
      if(Kbhit()) {
         k=Getch();
         if(k=='q') {
            Print("quit the program\r\n");
            quit=1;   /*quit the program*/
         }
```

```
        else {
          Print("more than 0.8s has elapsed reset the system\r\n");
          Delay(1000);   /*There has been a delay for more than 0.8s. Reset
              the system*/
        }
      }
      RefreshWDT();   /*Refresh WDT must be called within 0.8s*/
      Print("call Refresh WDT\n\r");
    }
    DisableWDT(); /*Disable the WDT. The system will refresh the WDT*/
    Print("Call DisableWDT\n\r");
}
```

- **DisableWDT()**

  功能: 停止 WatchDog 计时器
  语法: **void DisableWDT(void);**
  头文件: #include ”7188xc.h”
  描述: 更多详情请参考 EnableSDT()的描述。
  例程: 更多详情请参考 “EnableWDT()”

- **RefreshWDT()**

  功能: 刷新 WatchDog 计时器
  语法: **void RefreshWDT(void);**
  头文件: #include ”7188xc.h”
  描述: 更多详情请参考 EnableSDT()的描述。
  例程: 更多详情请参考“EnableWDT()”

- **IsResetByWatchDogTime()**

  功能: 检查系统是否被 WatchDog 计时器重启。
  语法: **int IsResetByWatchDogTime(void);**
  头文件: #include ”7188xc.h”
  描述: “是”则返回 0
  例程: 更多详情请参考 “EnableWDT()”

类型 7: 文件操作

函 数	说 明
GetFileNo	得到存储在 Flash 存储器上的文件数
GetFileName	使用文件索引得到文件名
GetFilePositionByNo	使用文件号得到文件位置
GetFileInfoByNo	使用文件号得到文件信息
GetFileInfoByName	使用文件名得到文件信息
…更多…	还有更多 timer 和 Watchdog Timer 函数提供，请参 7188xc.h 头文件 或参考用户手册 CD:\Napdos\minios7\document\libndex.htm

注意: MiniOS7 操作系统支持用户程序读取按文件，但并不支持以文件方式写入。

- **GetFileNo()**

  功能: 得到存储在 Flash 存储器上的文件数。
  语法: **int GetFileNo(void);**
  头文件: #include ”7188xc.h”
  描述: 返回文件数
  例程: 更多详情请参考“GetFilePositionByNo()”

- **GetFileName()**

  功能: 使用文件索引得到文件名。
  语法: **int GetFileName(int no,char *fname);**
  头文件: #include ”7188xc.h”
  描述: no: 文件索引 (第一个文件的索引为 0)
  fname: 存储文件名的缓存
  返回值: 成功则返回 NoError，并将文件名存入 fname。
  失败则返回-1，不在 fname 中存入任何数据。
  例程: 更多详情请参考“GetFilePositionByNo()”

- **GetFilePositionByNo()**

  功能: 使用文件号得到文件位置
  语法: **char far *GetFilePositionByNo(int no);**

头文件: #include ”7188xc.h”
描述: 地址可用来得到文件数据。
no: 文件索引 (第一个文件的索引为 0)
返回值: 成功则返回文件的起始地址，失败则返回 NULL。

注意: 如果文件大于 64K-16, 超出部分必须使用一个巨型指针来得到文件数据。

例程:

```
#include"7188xc.h"
static FILE_DATA far *fdata;   /*file_data structure, please see the file.c
                                 for details*/
char far *fp_no;
void main()
{
  int fileno,i;
  char fname[13];
  InitLib();   /*Initialize the Library*/
  fileno=GetFileNo();   /*get file number*/
  Print("Total file number=%d\n\r",fileno);
  fname[12]=0;
  for(i=0;i<fileno;i++){
    fdata=GetFileInfoByNo(i);   /*get file information using the file
                                  number*/
    if(fdata) {
      GetFileName(i,fname);   /*get file name*/
      Print("[%02d]:%-12s start at %Fp "
            "%02d/%02d/%04d %02d:%02d:%02d size=%lu\n\r", i,fname,
      fdata->addr,fdata->month,fdata->day,(fdata->year)+1980,
      fdata->hour,fdata->minute,fdata->sec*2,fdata->size);
    }
  }
  for(i=0;i<fileno;i++){
    fp_no=(char far *)GetFilePositionByNo(i);   /*get file position*/
    if(fp_no){
      GetFileName(i,fname);
      Print("file %d [%-12s] position: [ %Fp ]\r\n",i,fname,fp_no);
    }
  }
}
```

- **GetFileInfoByNo()**

  功能: 使用文件数索引得到文件信息
  语法: **FILE_DATA far *GetFileInfoByNo(int no);**
  头文件: #include ”7188xc.h”
  描述: no: 文件索引(第一个文件的索引为 0)
  返回值: 成功则返回文件的起始地址，失败则返回 NULL。
  例程: 更多详情请参考“GetFilePositionByNo()”

- **GetFileInfoByName()**

  功能: 使用文件名得到文件信息
  语法: **char far *GetFileInfoByName(char *fname);**
  头文件: #include ”7188xc.h”
  描述: fname: 文件名
  返回值: 成功则返回文件的起始信息，失败则返回 NULL。
  例程: 更多详情请参考“GetFilePositionByNo()”

类型 8: 连接 I-7000/I-87K 系列模块

函 数	说 明
SendCmdTo7000	向 I-7000/I-87K 系列模块发送数据
ReceiveResponseFrom7000_ms	接收 I-7000/I-87K 系列模块的回应
ascii_to_hex	将 ASCII 码换算成 16 进制值
hex_to_ascii	将 16 进制值换算成 ASCII 码
…More…	还提供很多为 COM 端口应用的函数。 更多详情请参考 7188xc.h 文件和 CD:\Napdos\minios7\document\lib

- **SendCmdTo7000()**

  功能: 向 I-7000/I-87K 系列模块发送数据

  语法: **int SendCmdTo7000(int iPort, unsigned char *cCmd, int iChksum);**

  头文件: #include ”7188xc.h”

  描述: 如果启用数据校验，函数将在命令最后加 2 个直接的校验位。

  iPort: 0/1/2/3/4 代表 COM0/1/2/3/4.

  cCmd: 将被发送的命令(如果没有则在 cCmd 最后加“\r”就像 SendCmdTo7000()函数在最后加校验位一样 。

  iChecksum: 1 代表启用数据校验, 0 代表不校验。

  返回值: 成功则返回 NoError，失败则返回错误代码。请参考 I-7000 产品用户手册。

  例程:

```
#include <7188xc.h>
void main()
{
   int port=2,quit=0,x;
   char k;
   InitLib();
   InstallCom(port,115200L,8,0,1);    /*install a COM Port for the I-7065D*/
   ClearCom(port);
   SendCmdTo7000(port, "@0100", 0);    /*send a command to DO to off*/
   if(ReceiveResponseFrom7000_ms(port,"@0100",20,0))
      Print("I-7065D is not available\r\n");
   while(!quit){    /*control Do*/
```

```
        Print("\n\r Enter 1~5 to set [Do] on...'9' to quit\n\r");
        k=Getch();
        x=ascii_to_hex(k);    /*convert ASCII code to hex*/
        ClearCom(port);
        switch(x){    /*send a command to set the I-7065D Do1~5 light to on*/
          case 1:    /*for command details, refer to the "I-7000 DIO manual"*/
            SendCmdTo7000(port, "@0101", 0);Print("[%x]=ON",x);break;
          case 2:
            SendCmdTo7000(port, "@0102", 0); Print("[%x]=ON",x);break;
          case 3:
            SendCmdTo7000(port, "@0104", 0); Print("[%x]=ON",x);break;
          case 4:
            SendCmdTo7000(port, "@0108", 0); Print("[%x]=ON",x);break;
          case 5:
            SendCmdTo7000(port, "@0110", 0); Print("[%x]=ON",x);break;
          case 9:
            quit=1; Print("*quit*");break;
        }   /*end of switch*/
      }   /*end of while loop*/
    }
```

- **ReceiveResponseFrom7000_ms()**

  功能: 接收来自 I-7000 模块的返回数据。

  语法: **int ReceiveResponseFrom7000_ms(int iPort, unsigned char *cCmd, long lTimeout, int iChksum);**

  头文件: #include "7188xc.h"

  描述: 在调用 SendCmdTo7000()函数后，即使不需要接受回应数据，也请调 ReceiveResponseFrom7000_ms() 函数。
  iPort: 1 代表 COM1、2 代表 COM2
  cCmd: 从 I-7000 发挥的回应。如果启用数据校验，函数将校验并移动校验码，CR 也将被移动。
  lTimeout: 设置超时，单位 ms。
  iChecksum: 1 代表启用数据校验，0 代表不启用。

  返回值: 成功则返回 NoError，失败则返回错误代码。请参考 I-7000 产品用户手册。

  例程: 更多详情请参考 endCmdTo7000()

- **ascii_to_hex()**

  功能:　　将 ASCII 转换成 16 进制值

  语法:　　**int ascii_to_hex(char ascii);**

  头文件:　　#include ”7188xc.h”

  描述:　　返回 1 个整数代表 16 进制值

  ascii: ASCII 代码字符

  例程:　　更多详情请参考“ SendCmdTo7000()”

附录 E: 编译和链接

使用 TC 编译器

使用 TC 2.01 有 2 种编译过程，参考方式如下：

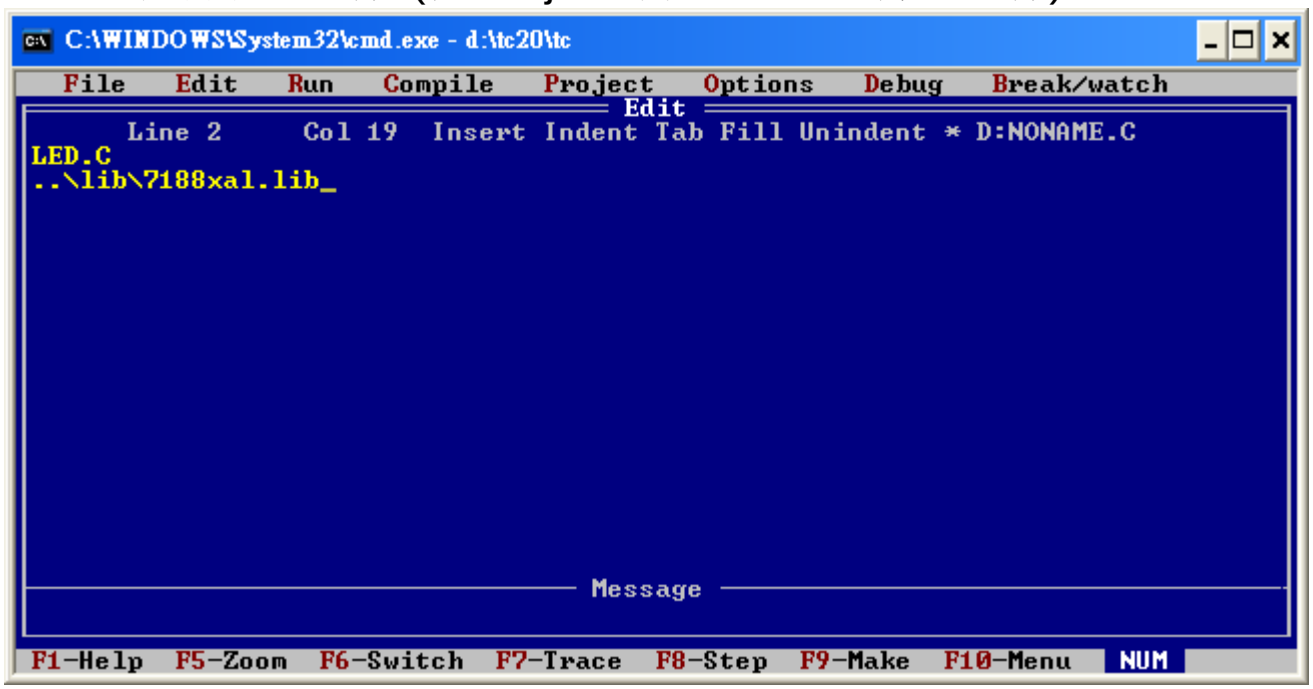
方法 1: 使用命令行 (详情请参考 **CD:\8000\NAPDOS\7188XABC\7188XC\Demo\BC_TC\Hello_C\gotc.bat**)

tcc -Ic:\tc\include -Lc:\tc\lib hello1.c ..\lib\7188XCs.lib

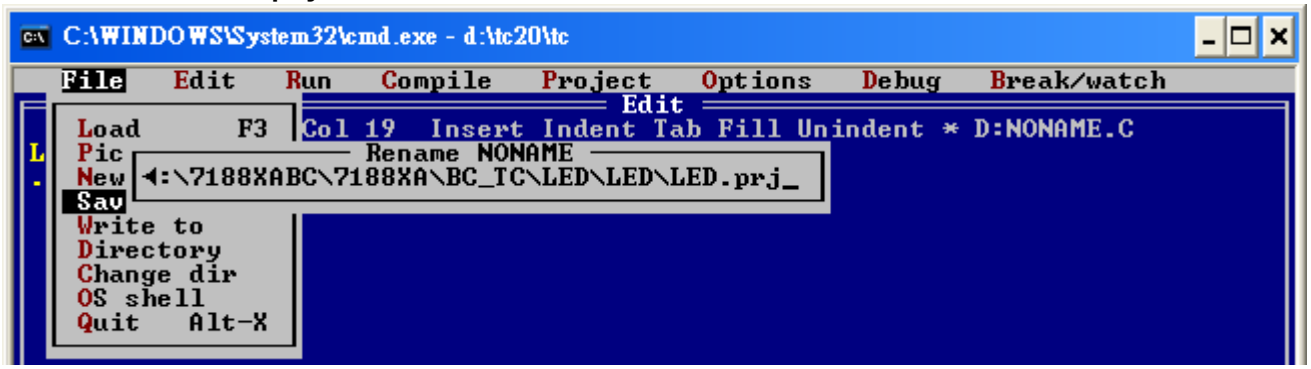
方法 2: 使用 TC 集成编译环境

步骤 1: 执行 TC.EXE 运行 TC 2.01 使用 TC 集成编译环境。

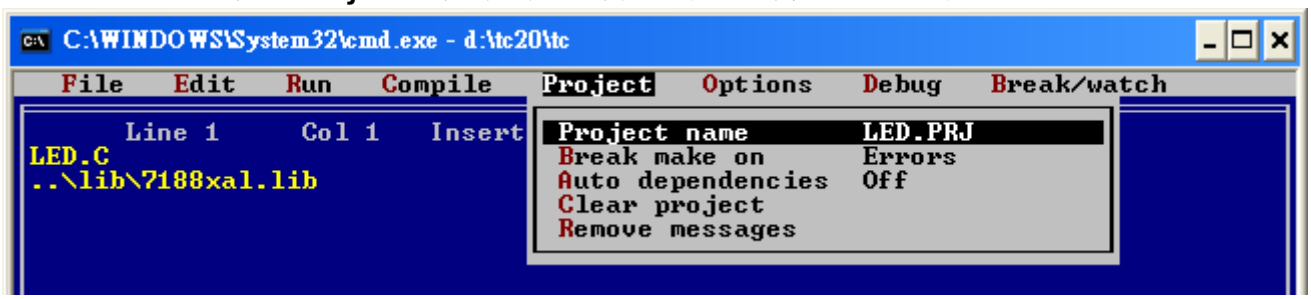
步骤 2: 编辑项目文件 (在 Project 中加入必要的库和文件)。



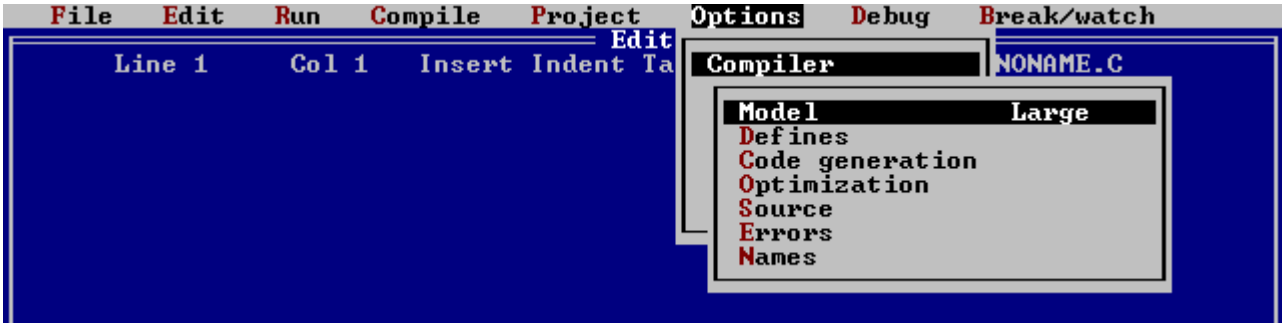
步骤 3: 在 File 目录中选择“save”将项目保存为工程文件，并输入文件名，如 LED.prj。

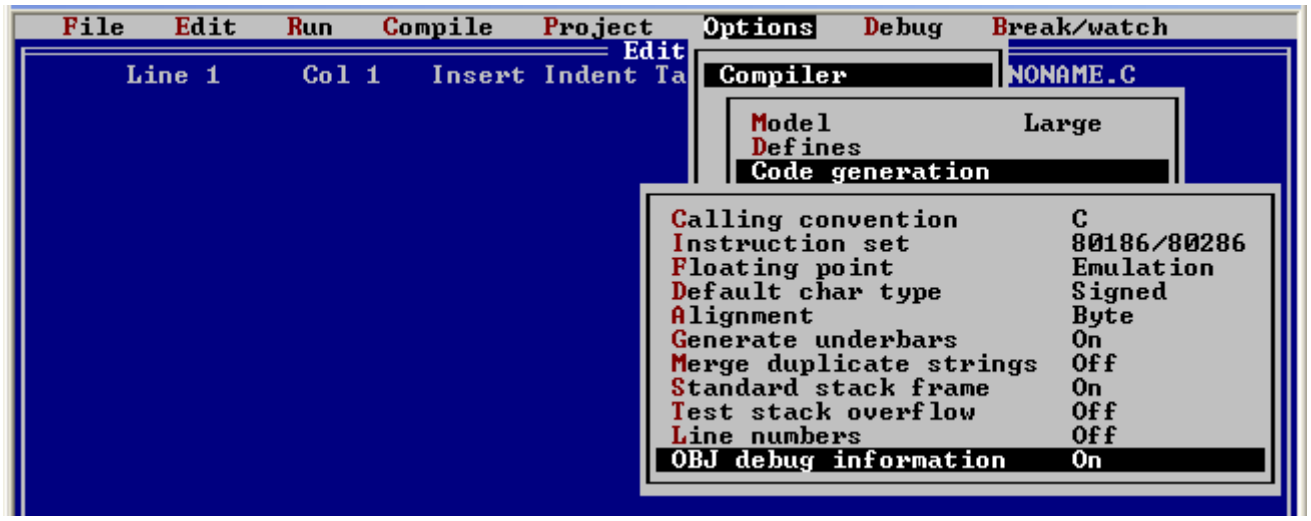


步骤 4: 通过在 Project 菜单中选择工程文件导入工程。

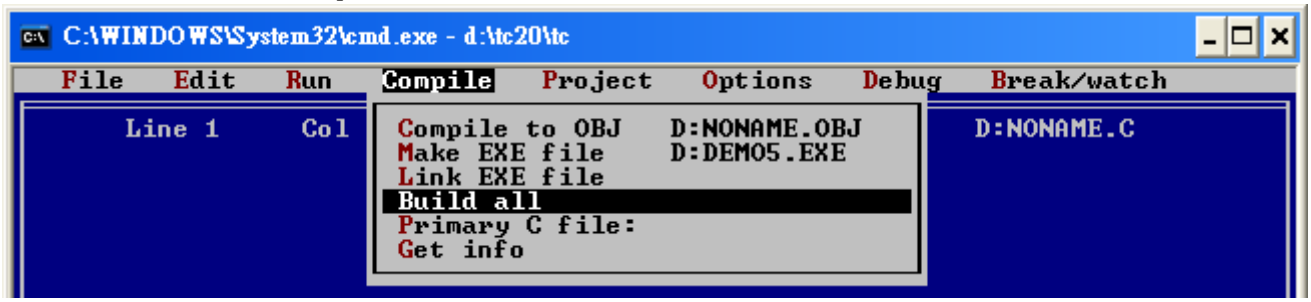


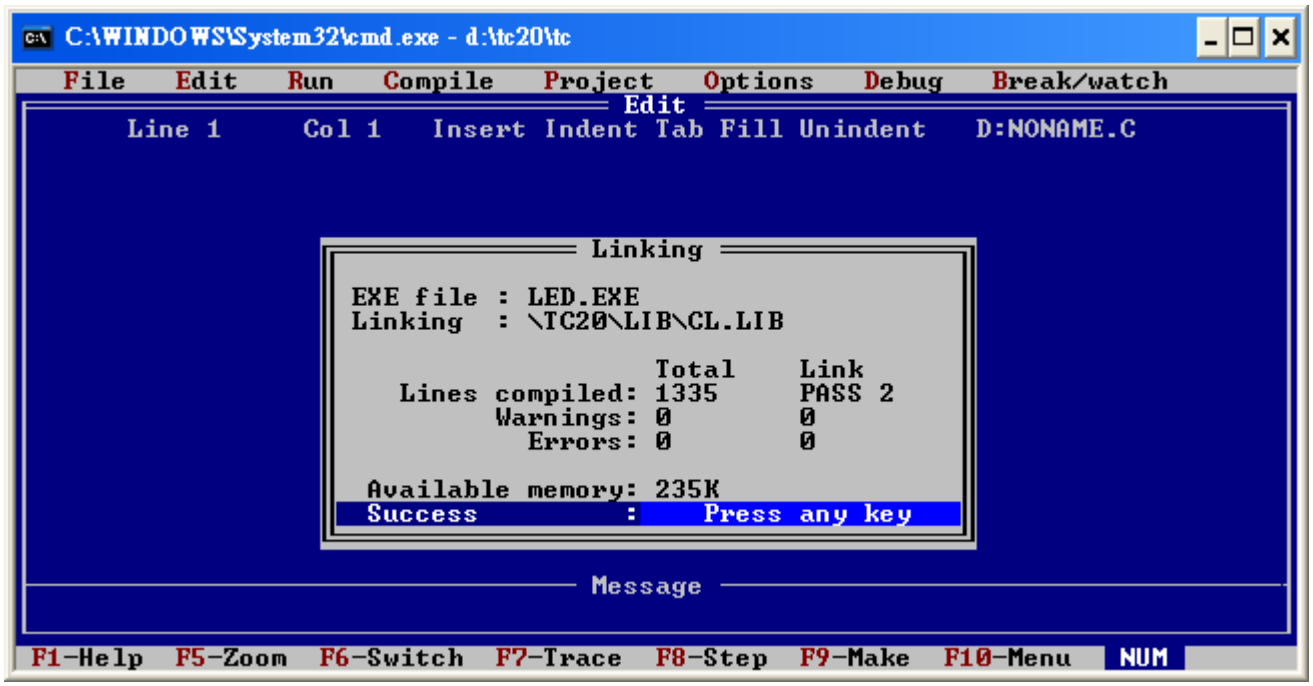
步骤 5: 在编译器的 Options 目录改变存储模式 (Small for 7188xcs.lib, large for 7188xcl.lib) 并设置代码生成为 80186/80286，请见如下对话框：





步骤 6: 选择 **Compile** 的“**Build all**” 选项建立项目



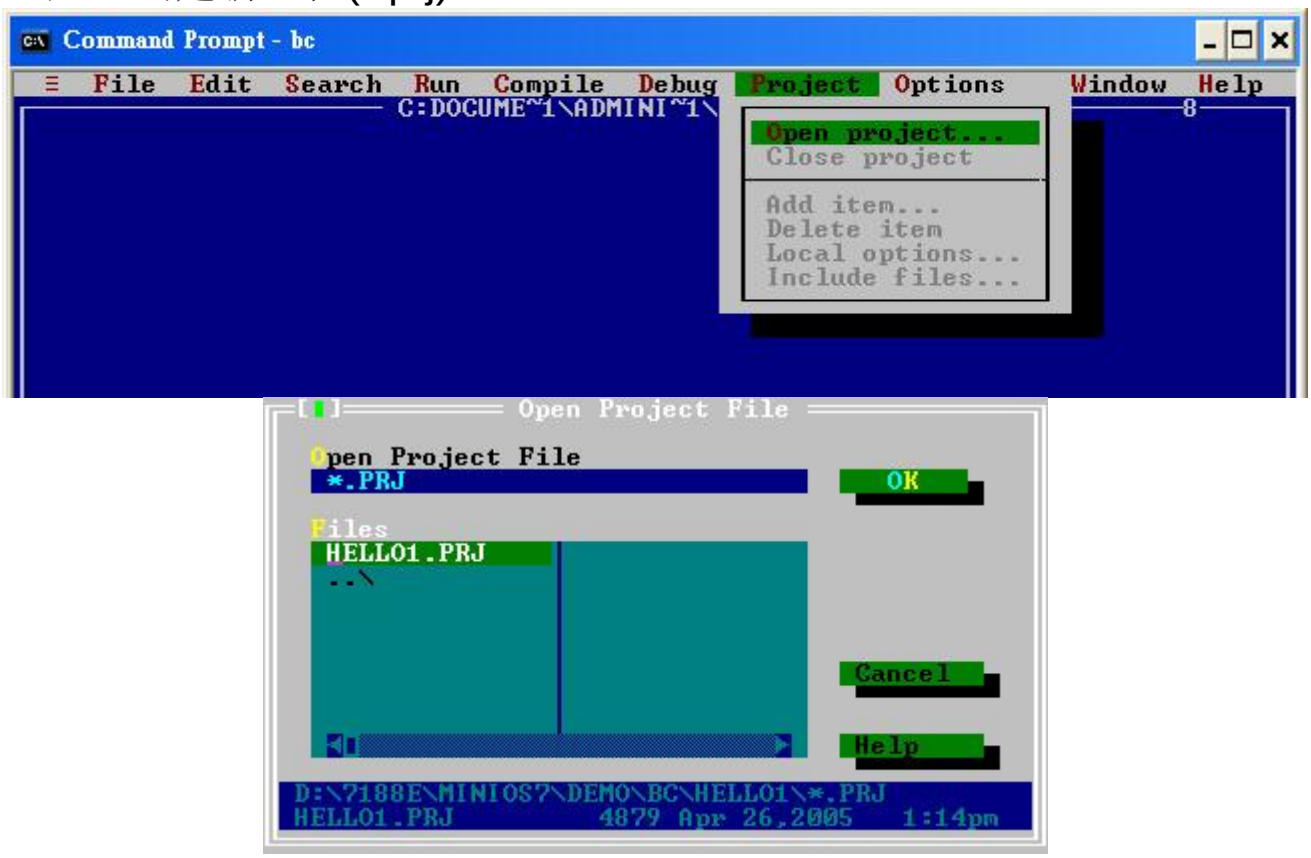

C:\WINDOWS\System32\cmd.exe - d:\tc20\tc
File Edit Run Compile Project Options Debug Break/watch
Edit
Line 1 Col 1 Insert Indent Tab Fill Unindent D:NONAME.C
Linking
EXE file : LED.EXE
Linking : \TC20\LIB\CL.LIB
Total Link
Lines compiled: 1335 PASS 2
Warnings: 0 0
Errors: 0 0
Available memory: 235K
Success : Press any key
Message
F1-Help F5-Zoom F6-Switch F7-Trace F8-Step F9-Make F10-Menu NUM

使用 BC++ 编译器

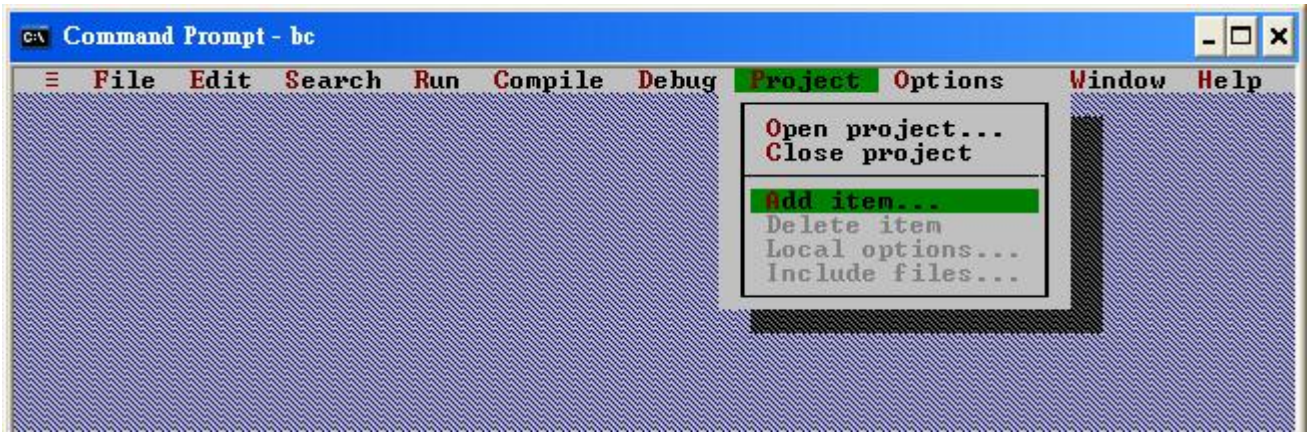
使用 BC++ 编译器的编译步骤如下：

步骤 1: 执行 Borland C++ 3.1.

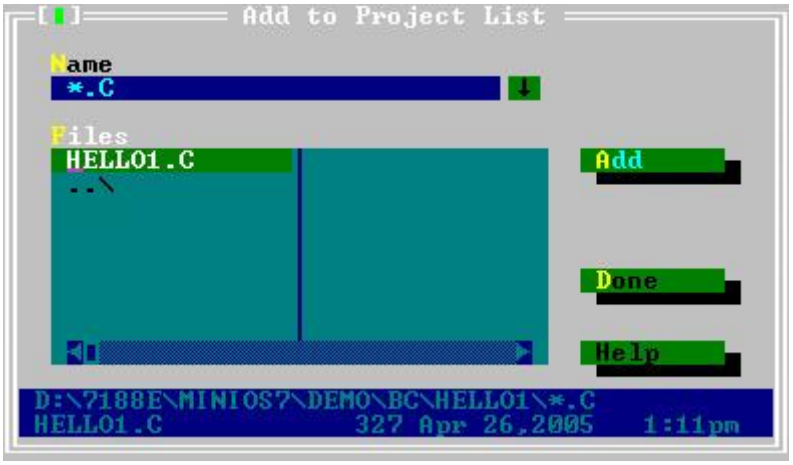
步骤 2: 创建新工程(*.prj).



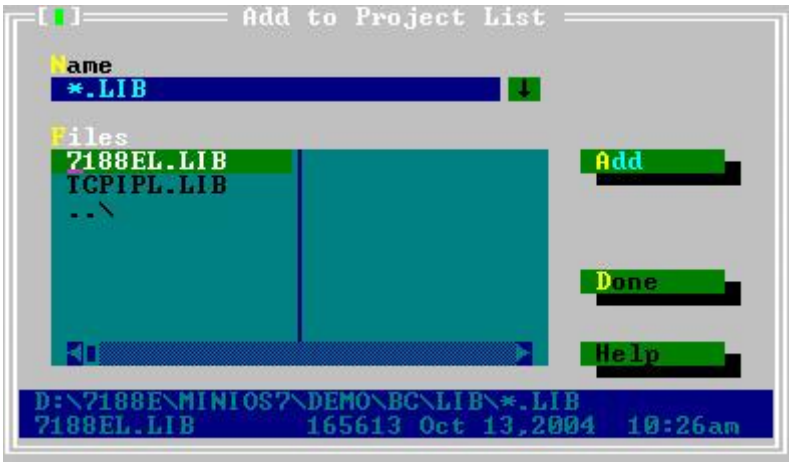
步骤 3: 在工程文件中添加必要的文件。



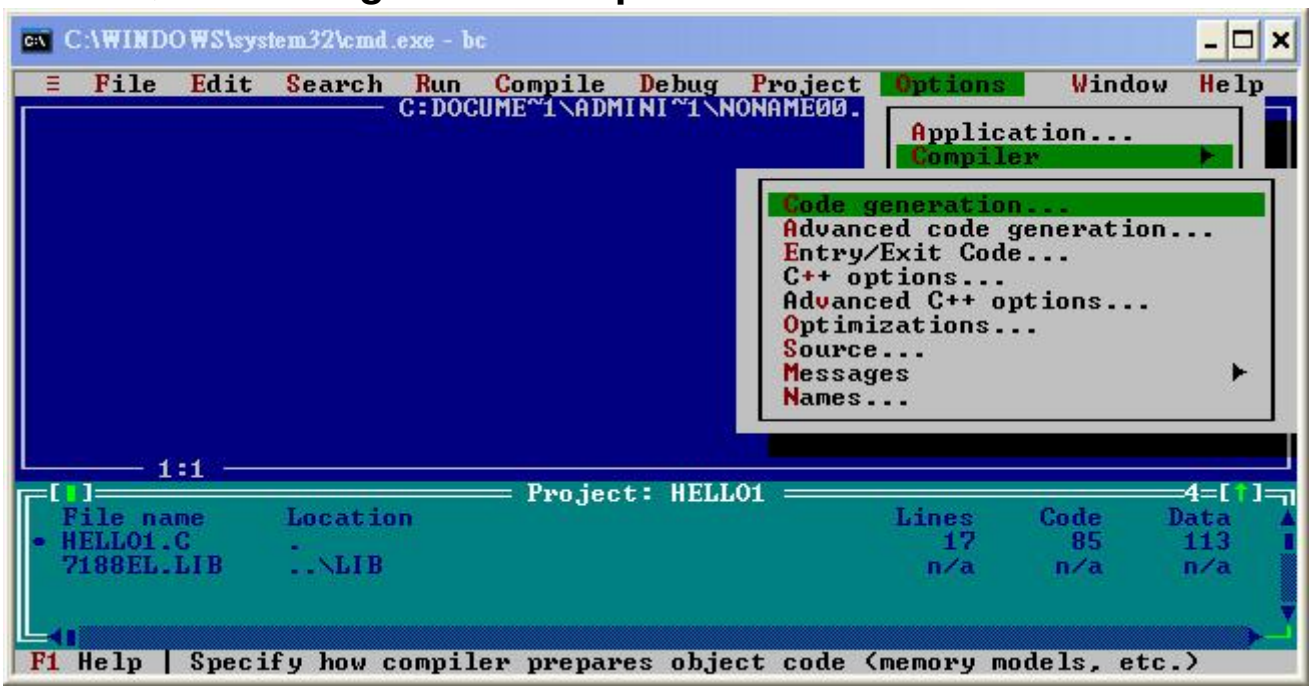
步骤 3.1: 选择代码文件。



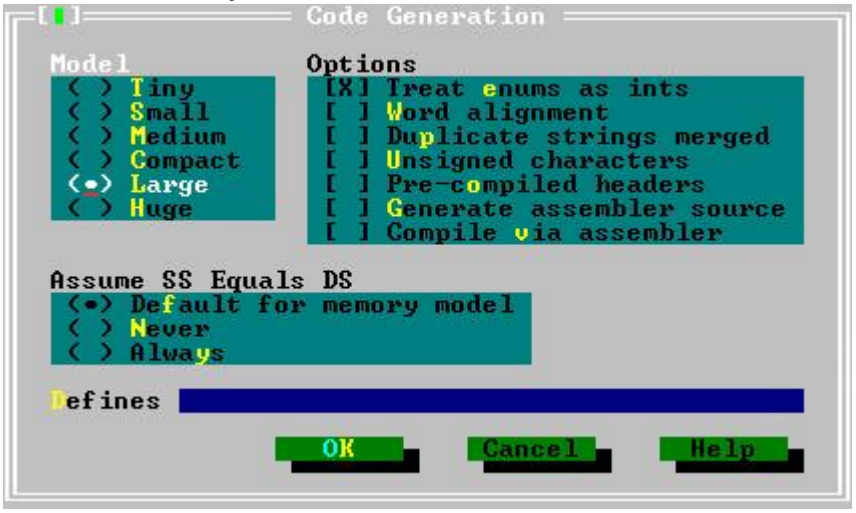
步骤 3.2: 选择函数库并点击按钮 **Done**。



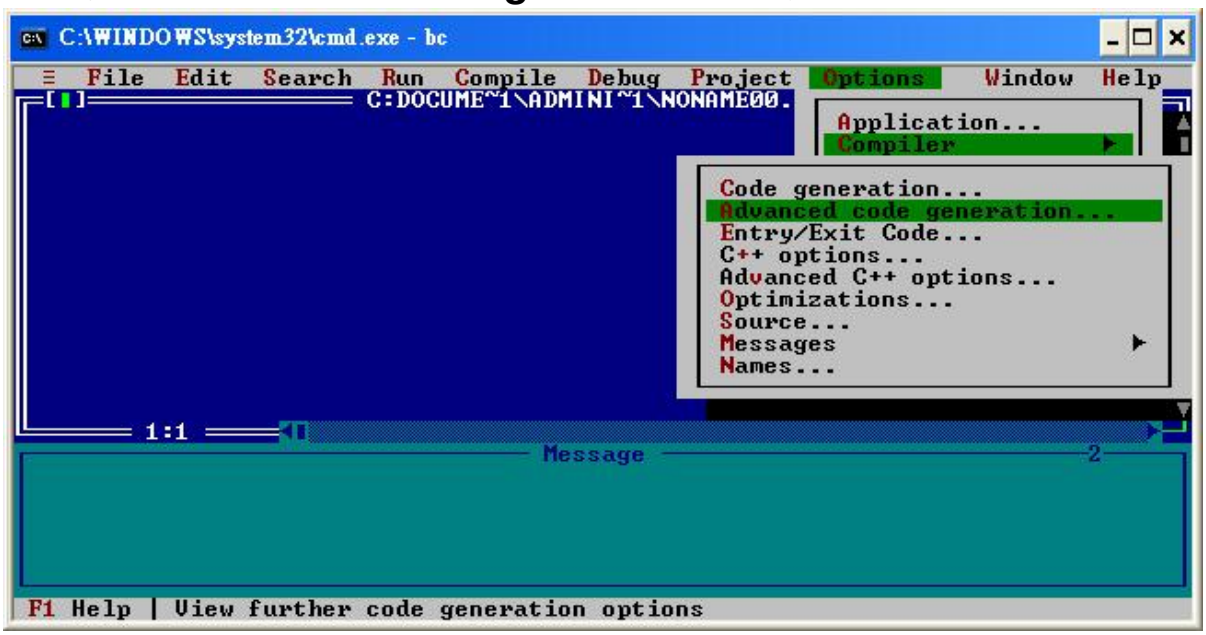
步驟 4: 设置 **Code generation option**



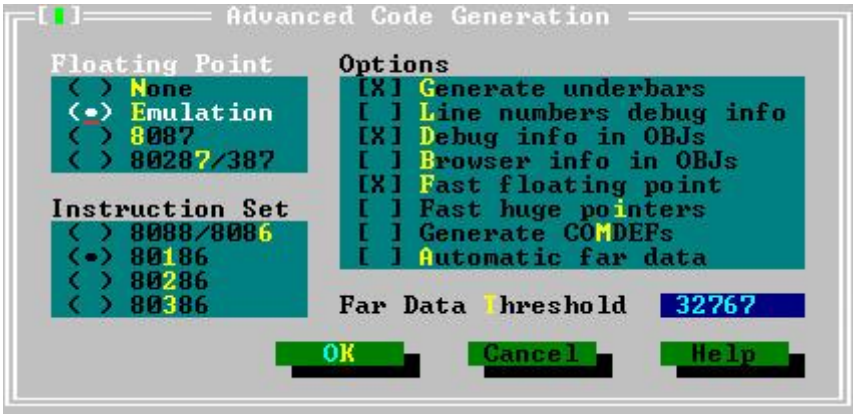
步驟 4.1: 改变存储模式(7188XCs.lib 适用于小型模式, 7188XCl.lib 适用于大型模式)。



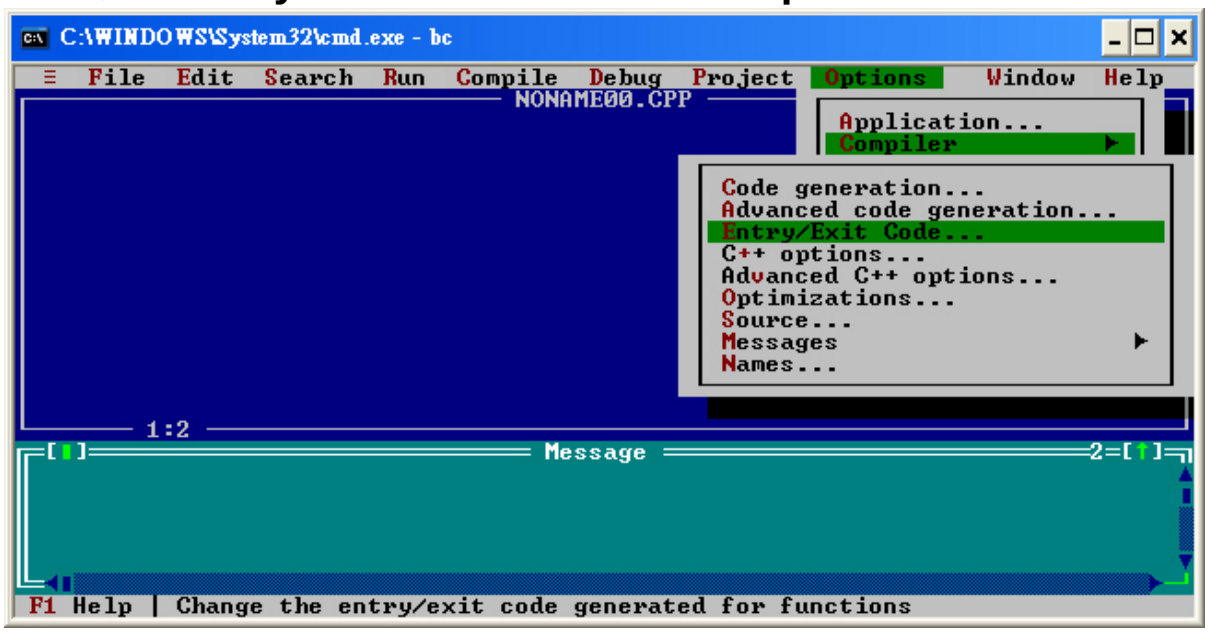
步骤 5: 设置 **Advanced code generation** 选项。



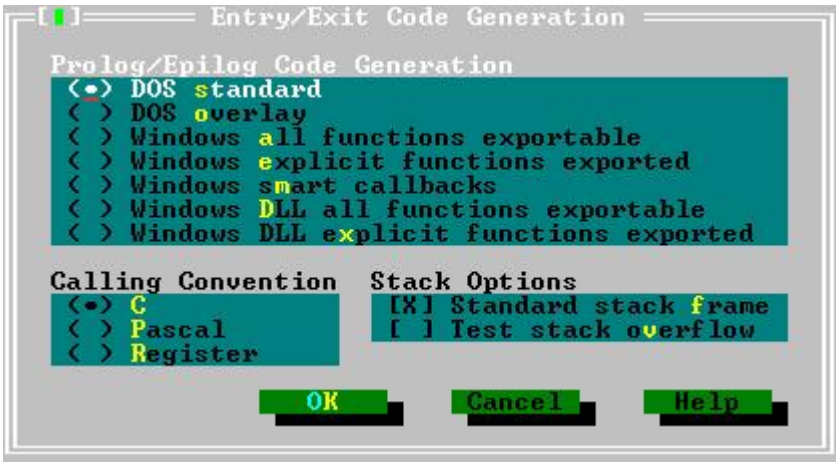
步骤 5.1: 设置 **Floating Point** 为 **Emulation**，设置 **Instruction Set** 为 80186。



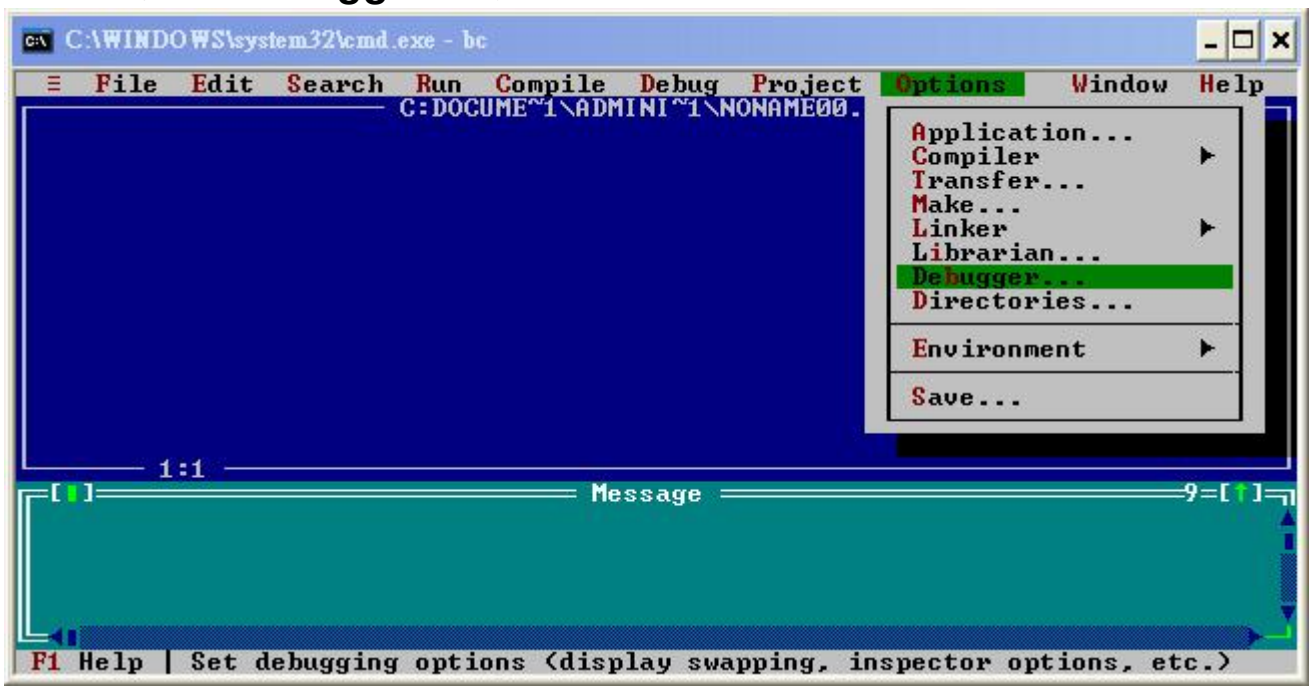
步骤 6: 设置 **Entry/Exit Code Generation option**



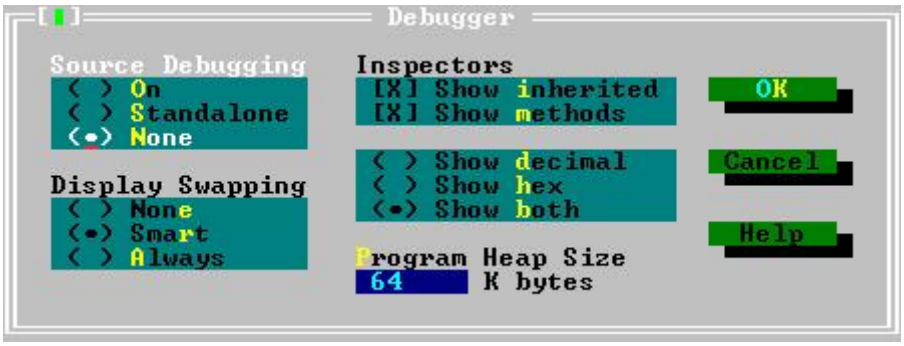
步骤 6.1: 设置为 **DOS standard**。



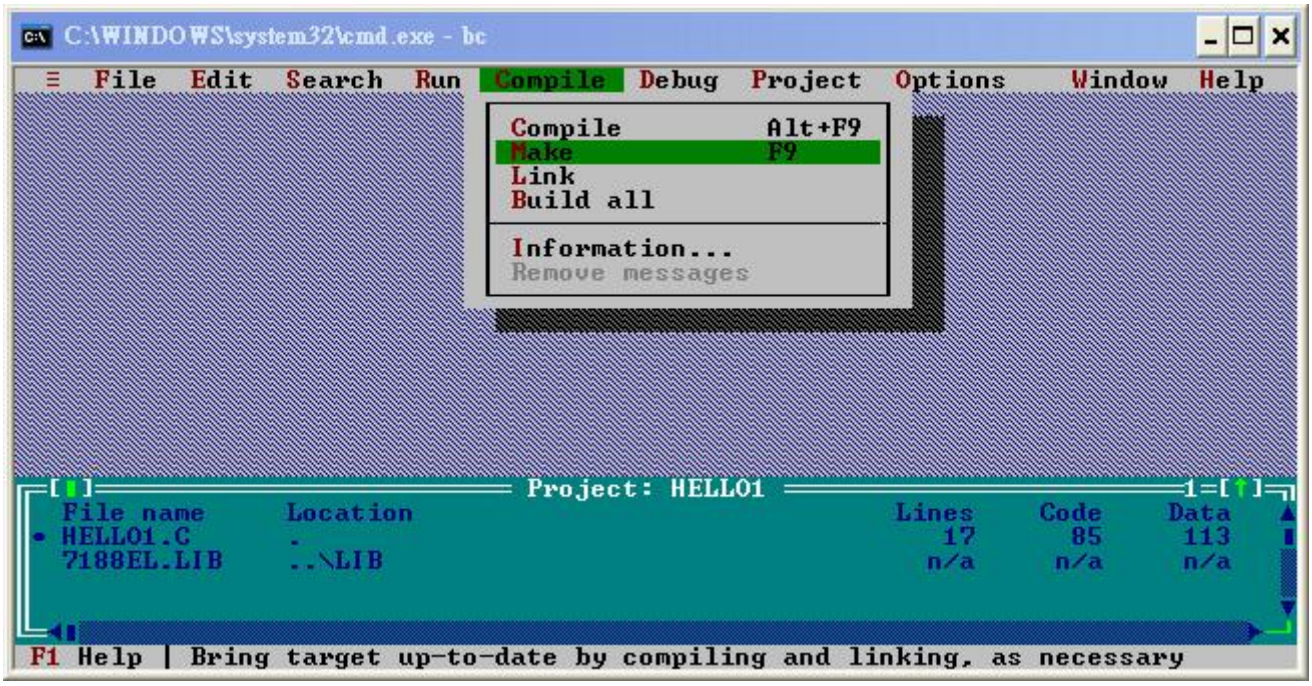
步骤 7: 设置 **Debugger** 选项。

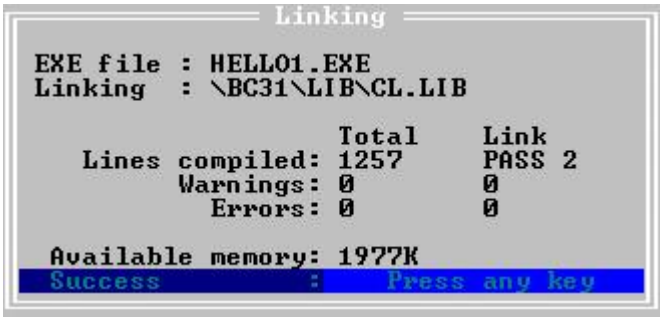


7.1 设置 **Source Debugging** 为 **None**。



步骤 8: 编译工程生成工程。

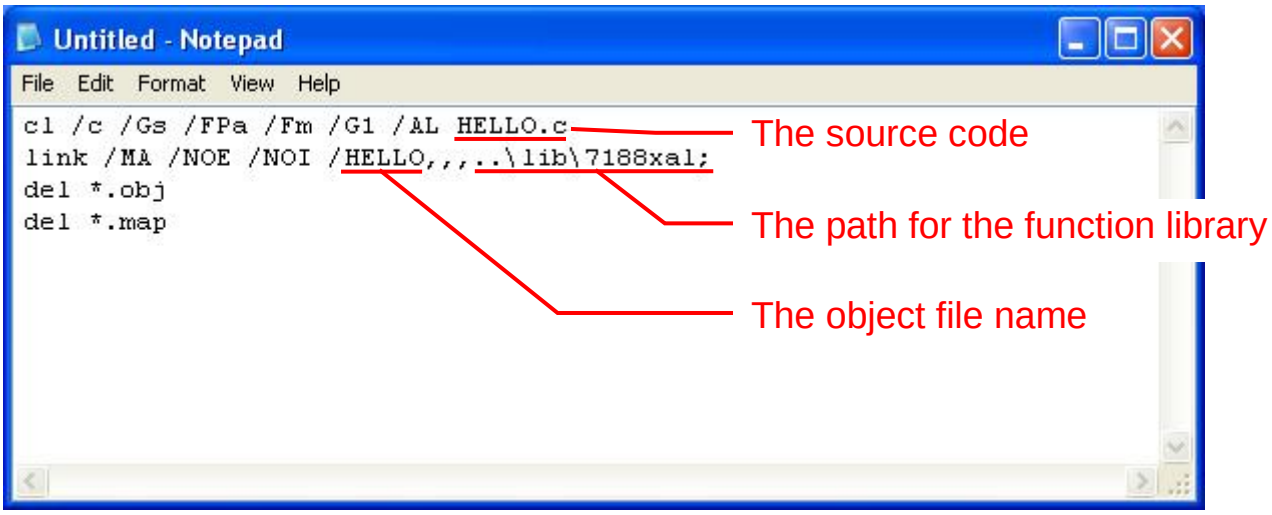




使用 MSC 编译器

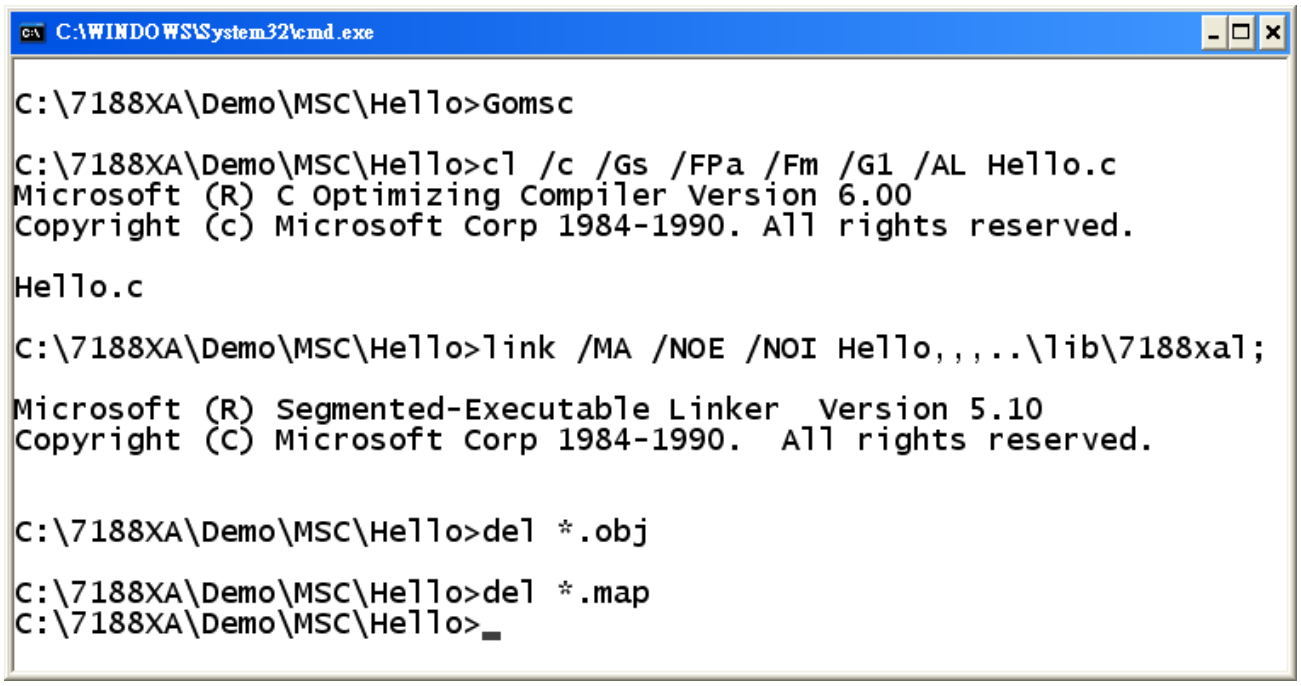
使用 BC++ 编译器的编译过程如下：

步骤 1: 在代码文件所在文件夹使用文本编译器创建一个文件名为 Gomsc.bat 的批处理文件。

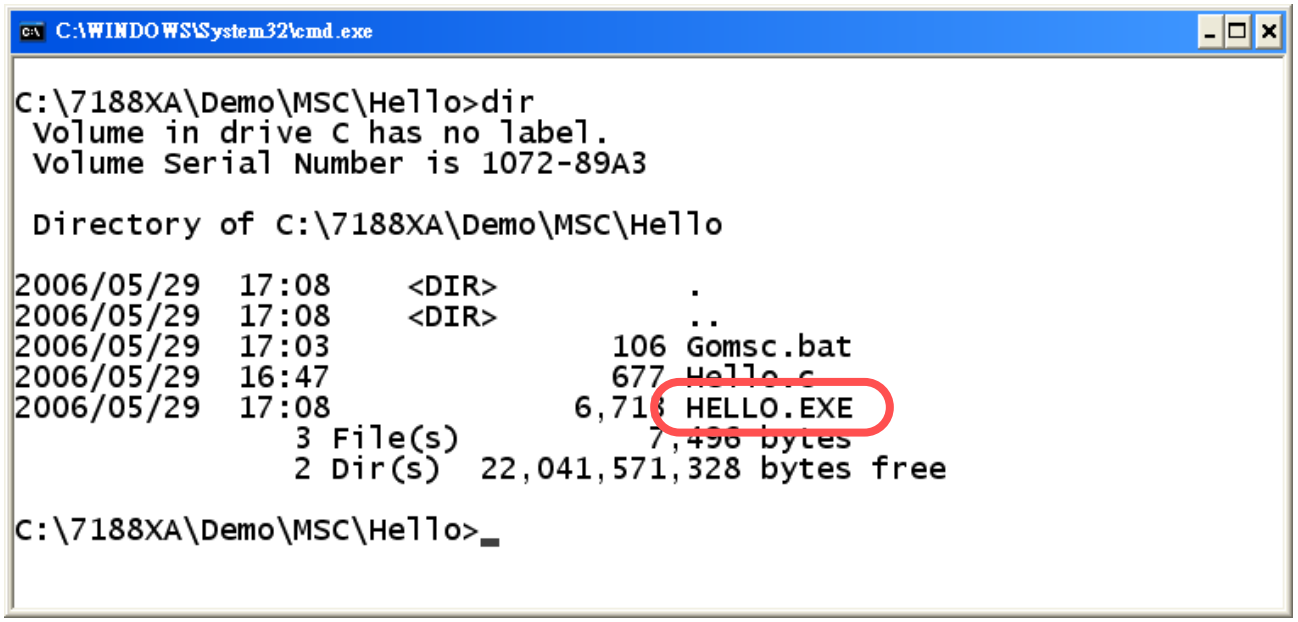


注意: **/C:** 不列出注释
/Gs: 不检查堆栈**/Fpa:** 在目标文件中放入浮点运算库名称**/Fm:** [map 文件]
/G1: 186 指令
/AL: 大型模式

步骤 2: 运行 Gomsc.bat。



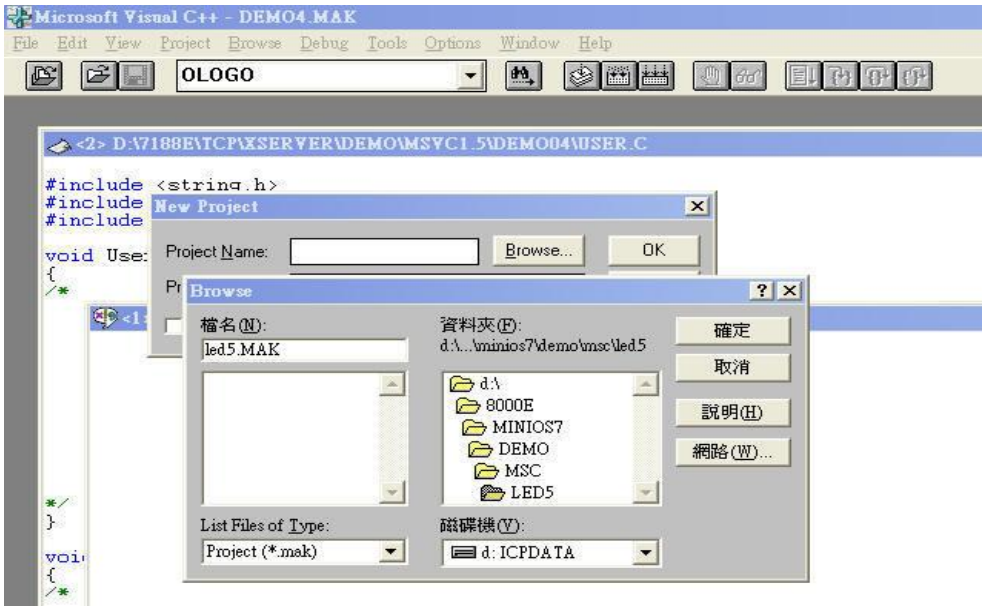
步骤 3: 如果成功编译将创建一个新的可执行文件。



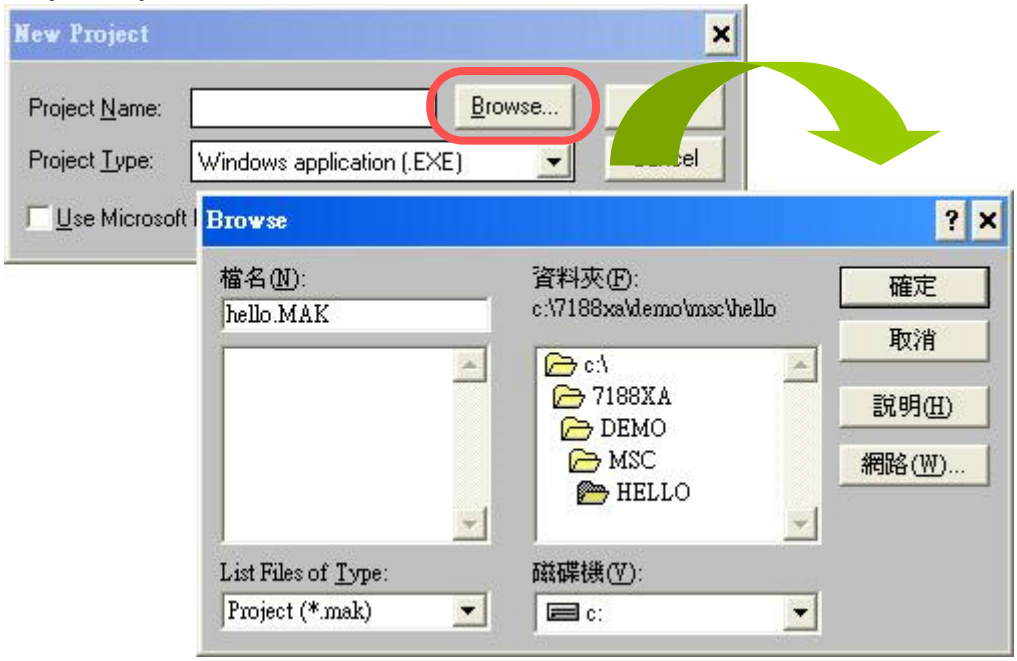
使用 MSVC++编译器

使用 MSVC 1.50 编译器的编译过程如下：

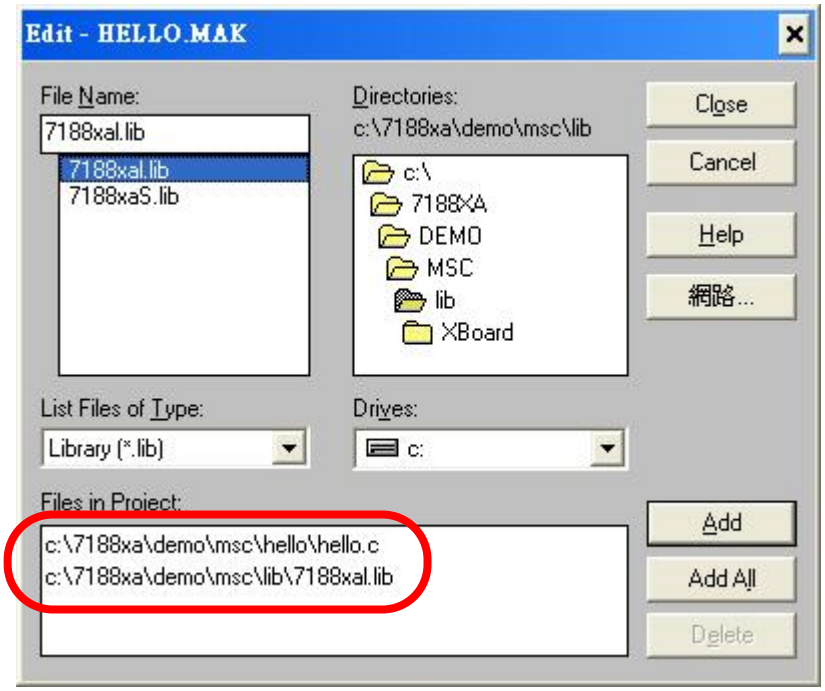
步骤 1: 运行 MSVC.exe。



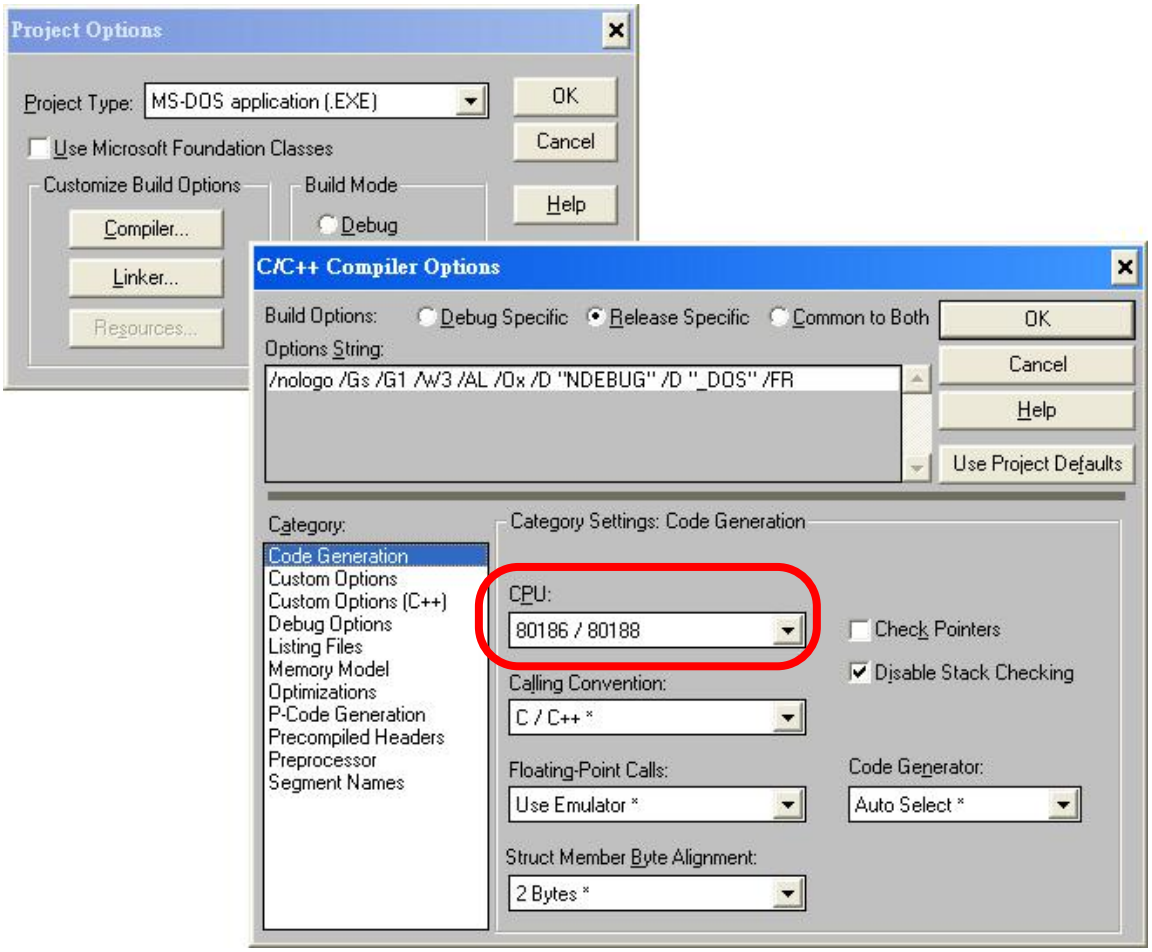
步骤 2: 在工程名区域输入工程名创建一个新的项目，并选择 MS-DOS 应用(EXE) 作为项目类型。



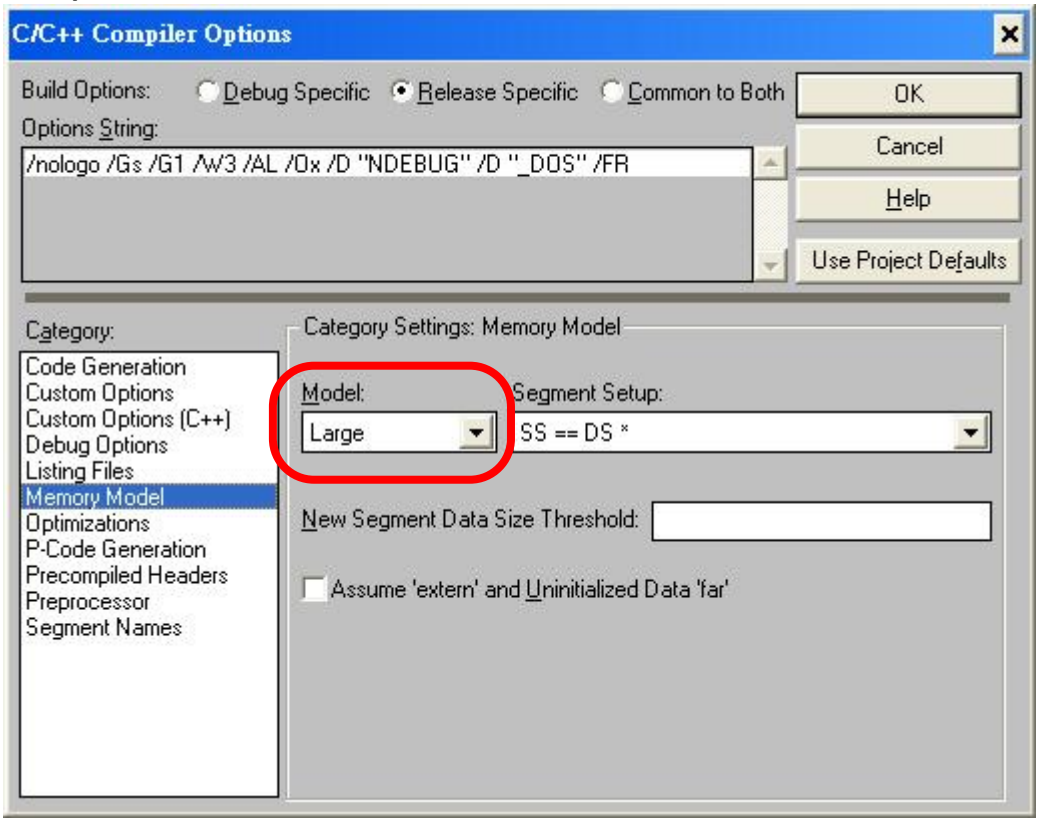
步骤 3: 在项目中加入用户程序和必要的库文件。



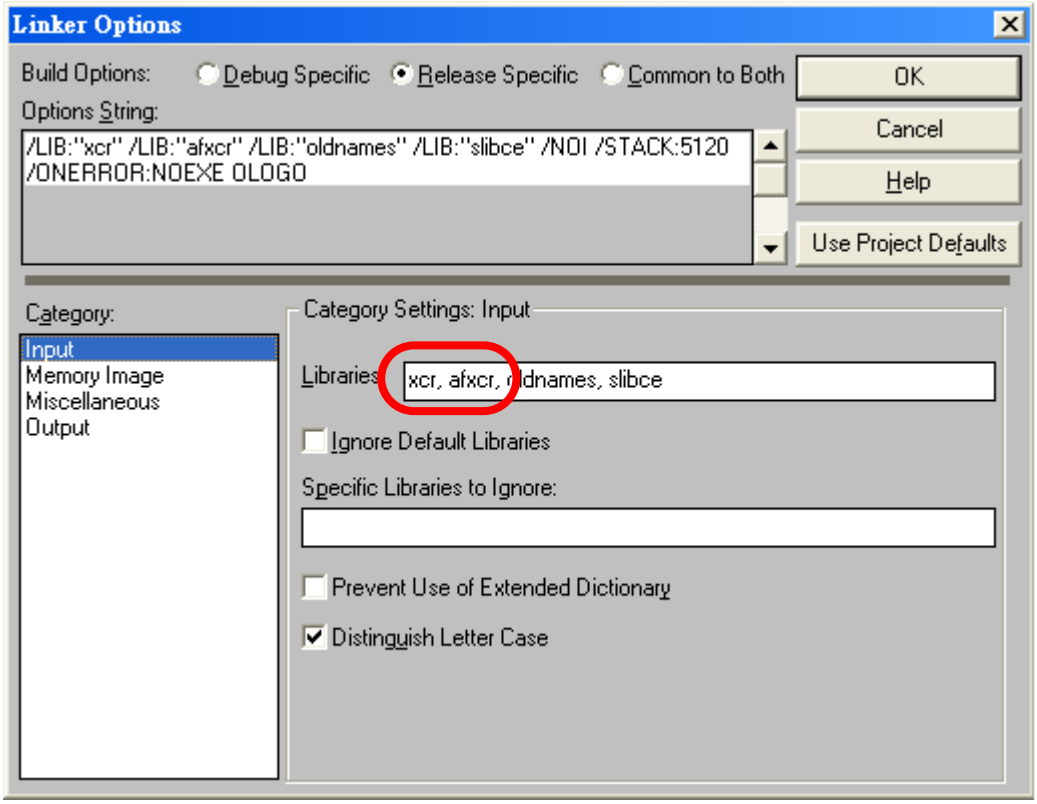
步骤 4: 在编译器中设置 **Code Generation**。

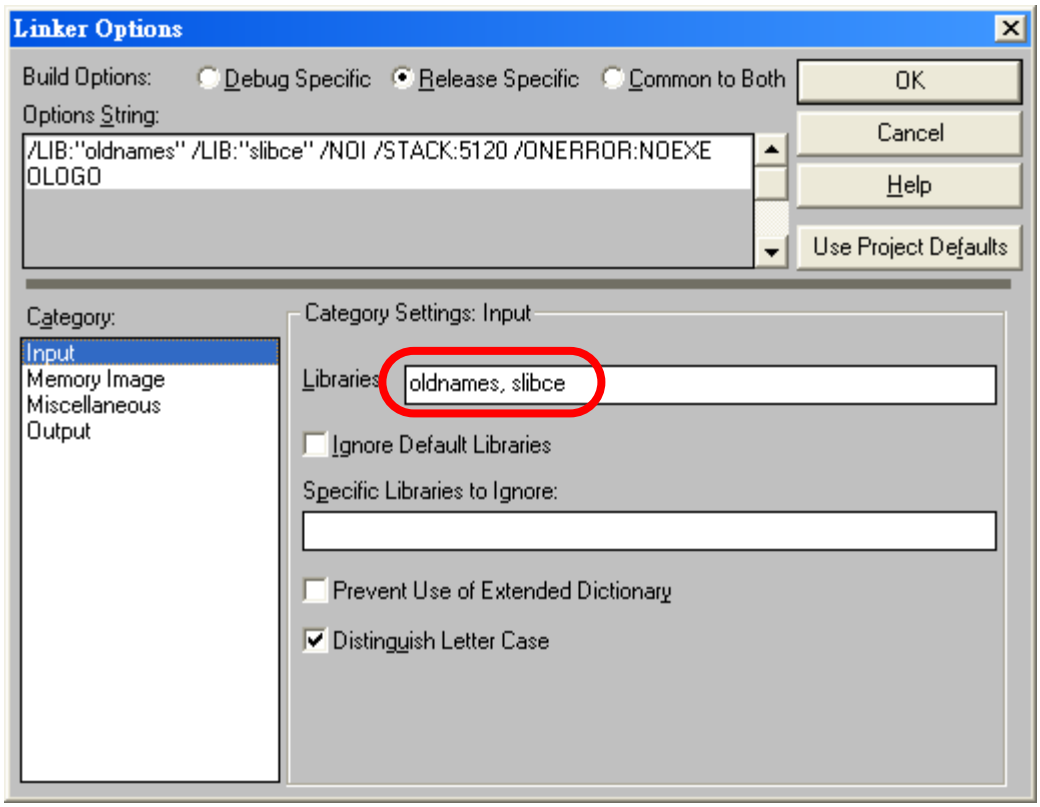


步骤 5: 变换 **Memory model** (7188xcs.lib 适用于小型,7188xcl.lib 适用于大型)。

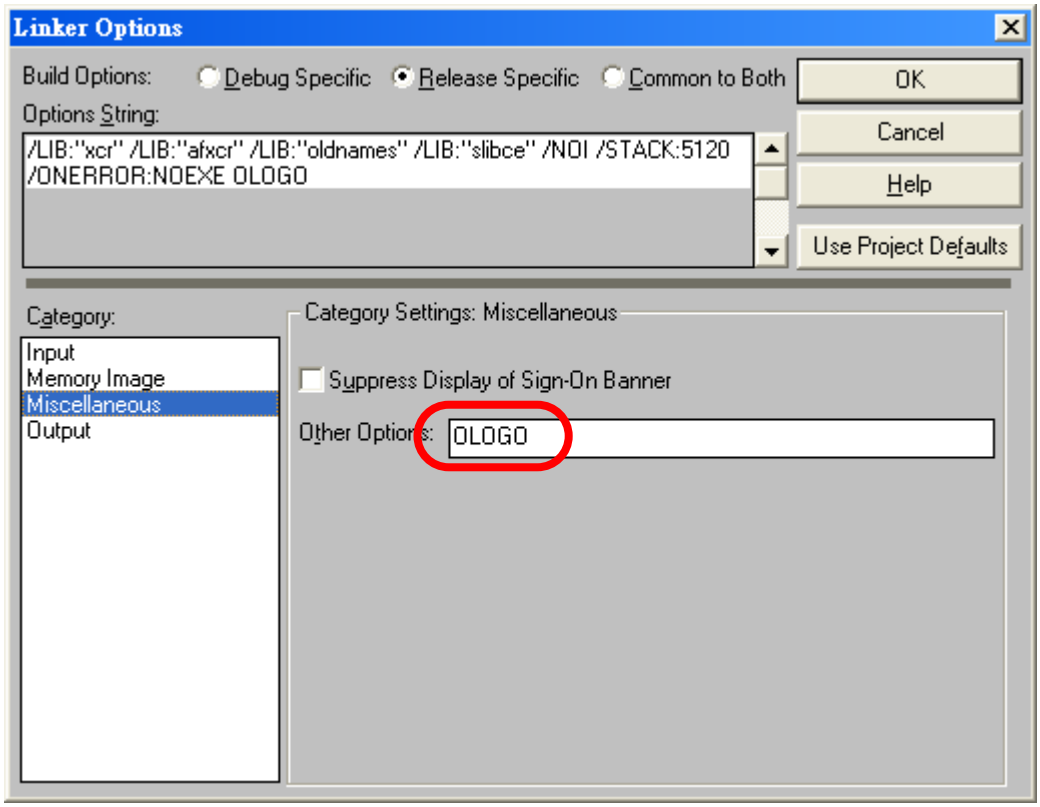


步骤 6: 在 **Input** 类中移除 xcr, afxcr 库。

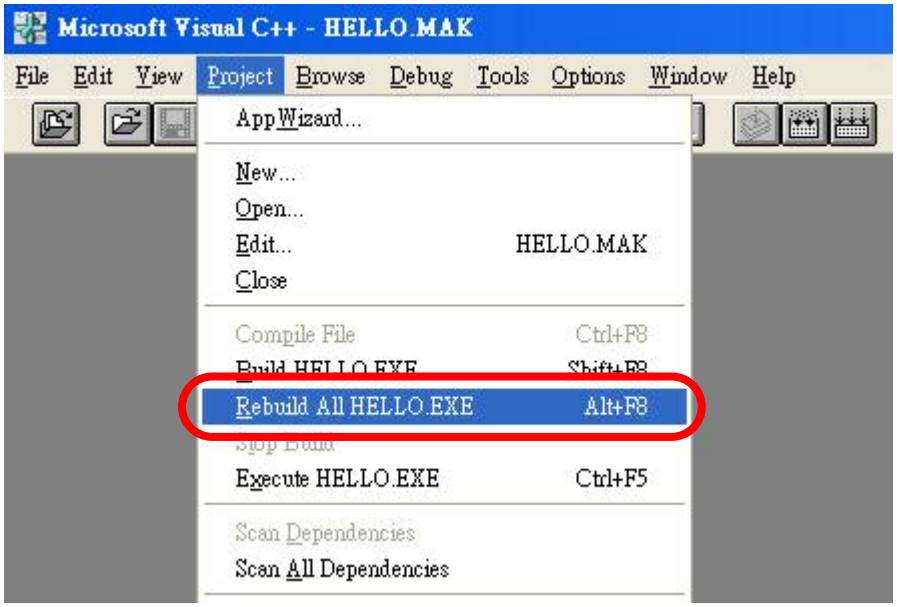


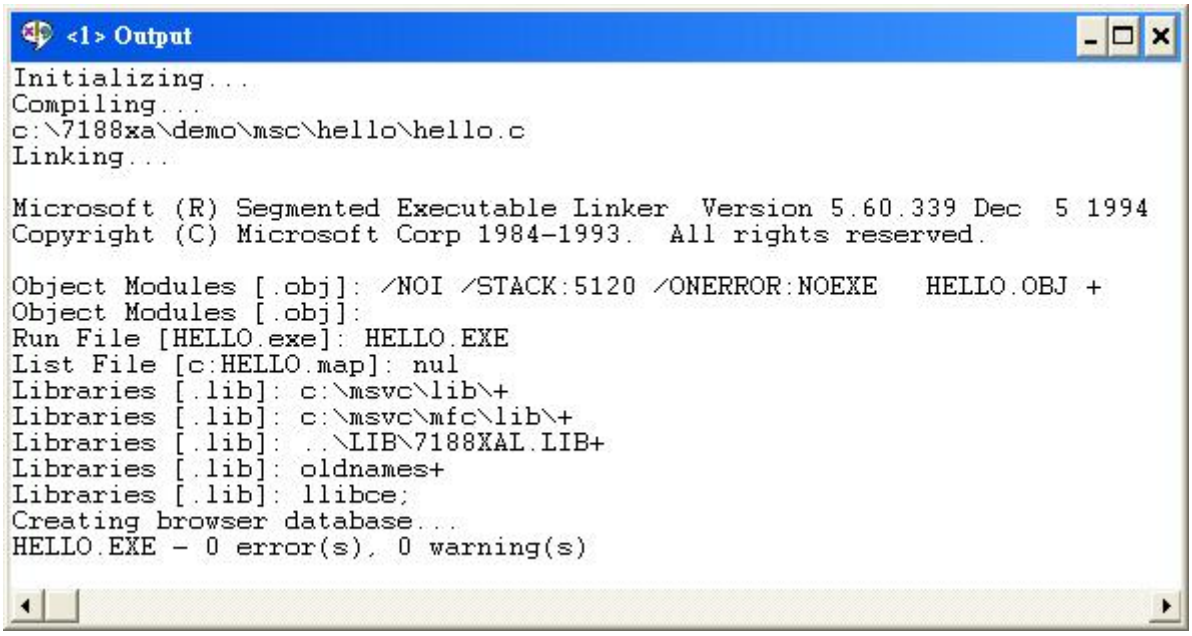


步驟 7: 在 **Miscellancous** 类中移除 OLOGO 选项。



步驟 8: **Rebuild** 项目。



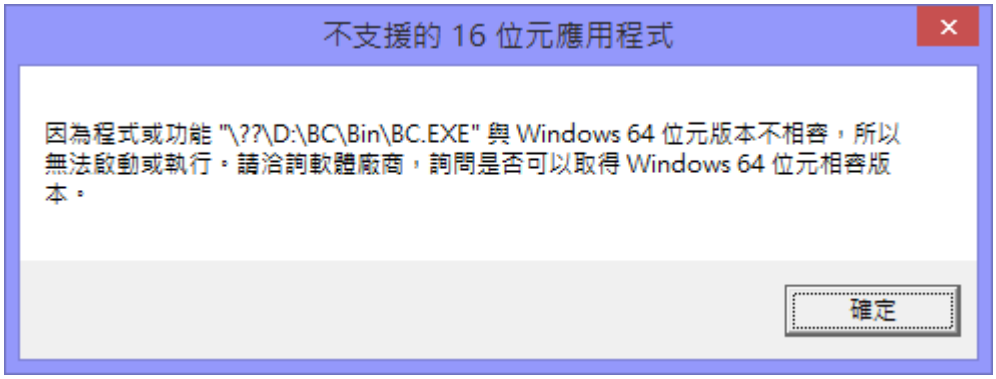


在 64 位平台编译

前言:

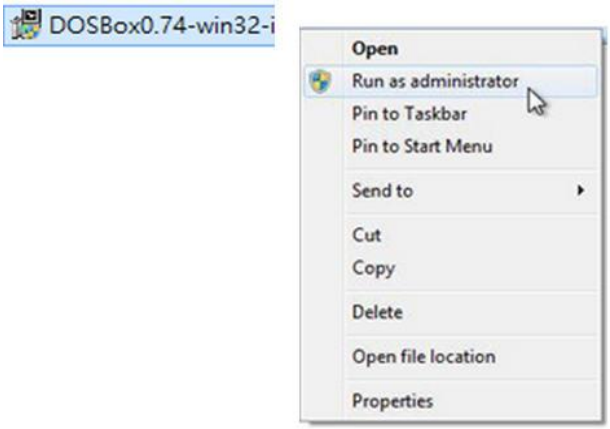
假如要在 Windows 7 或 Windows 8 这种有支持 64 位的 OS 平台上使用 16 位的编译器(像是 BC3.1 或是 TC++3.0)创建 MiniOS7 项目时，将会告知有 64 位平台兼容性问题的信息出现。

“**Application can’t run on your PC. Contact with your software publisher**” 或是下列截图信息.



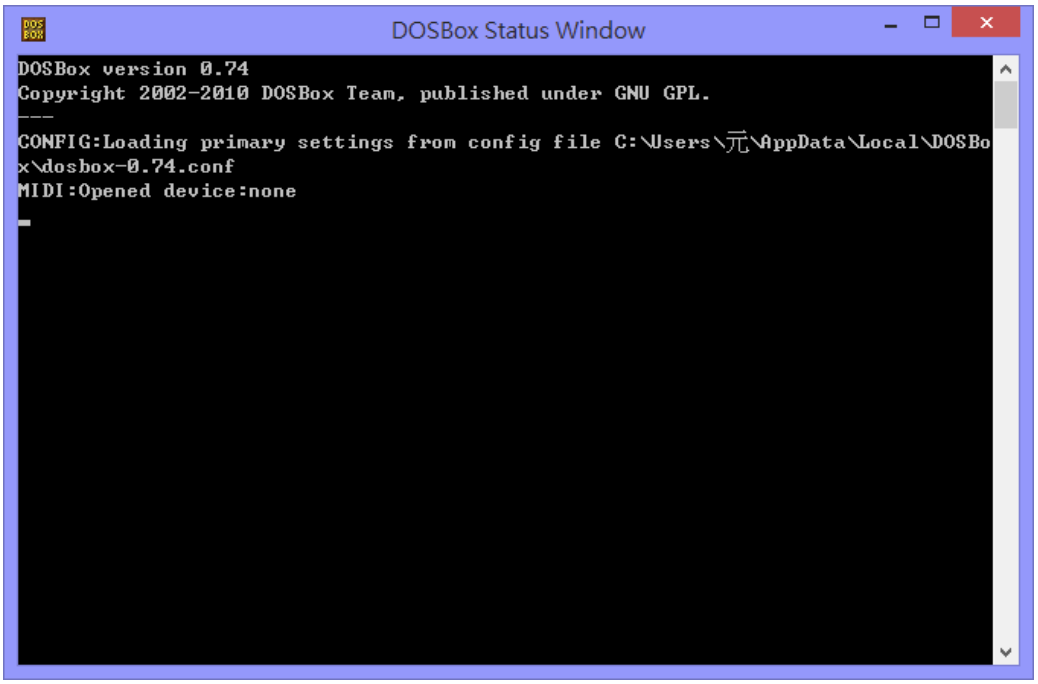
请遵循如下的步骤来解决上述的问题。

步骤 1: 下载和安装 Windows 环境的 DosBox. 你可以很容易在 64-bit 的 OS 上安装 32-bit 的版本。 下载的网页如下：
http://sourceforge.net/projects/dosbox/files/dosbox/0.74/DOSBox0.74-win32-installer.exe/download
注意：强烈建议以系统管理员身份安装 DOSBox。

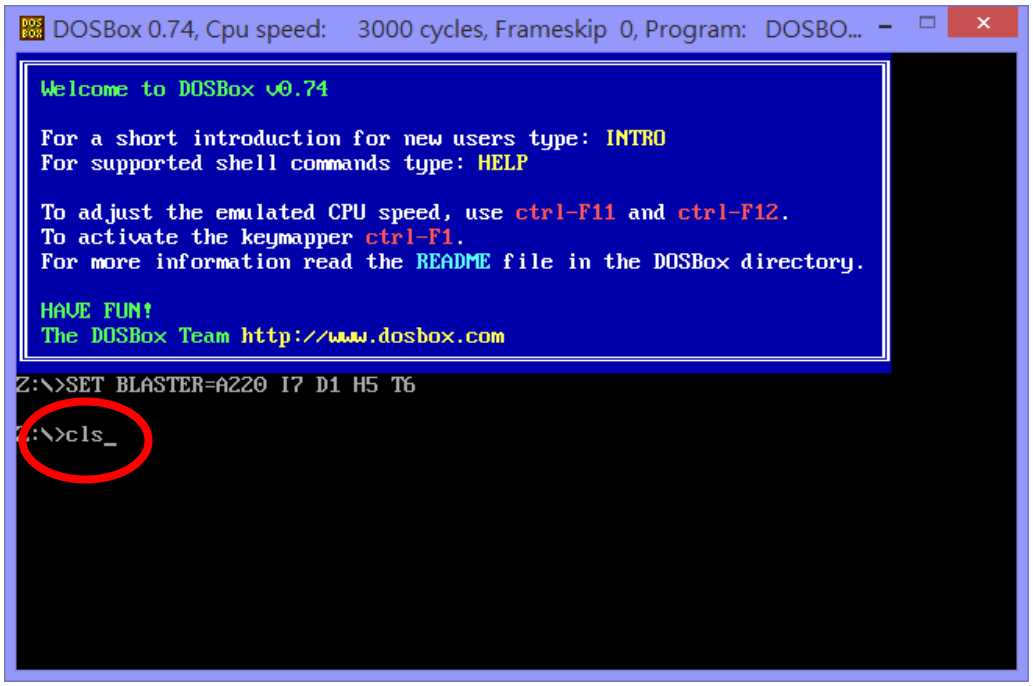


步骤 2: 启动 DOSBox. 双击 DOSBox 

，将会产生二个窗口，一个是 DOSBox Status Window, 另一个是 DOSBox Console. 请参考如下的图。

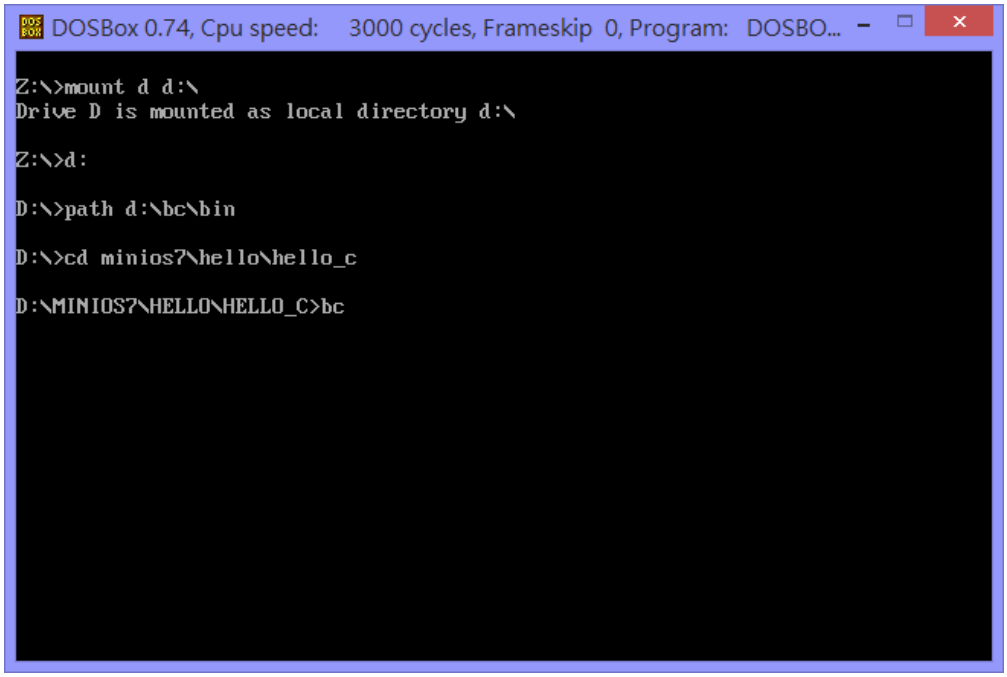


首先输入命令“CLS”或“cls”来清除屏幕上的文字。

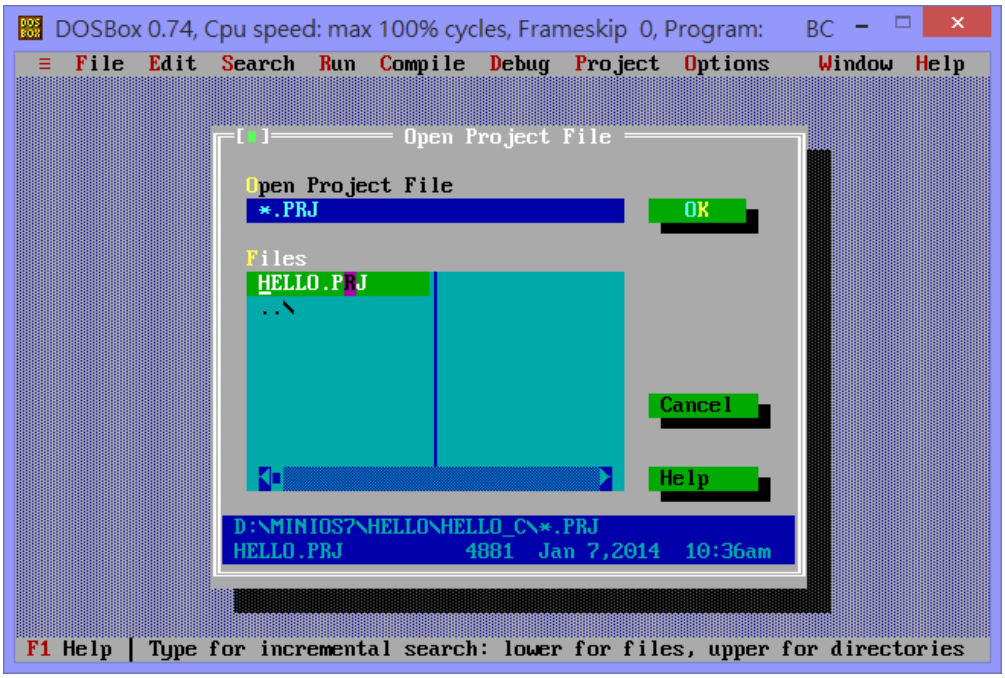


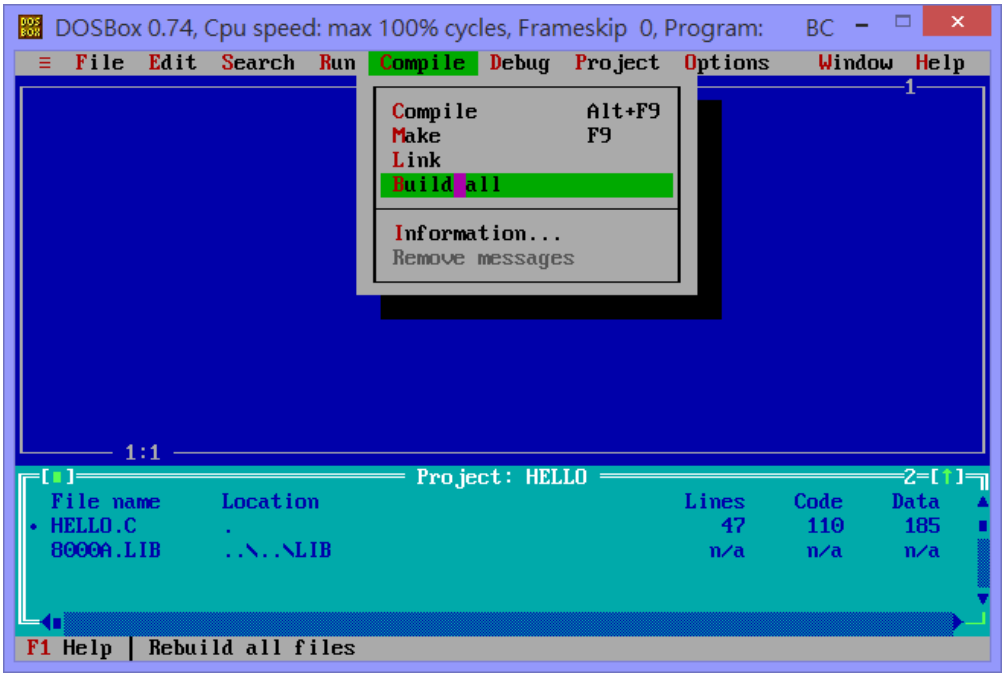
步骤 3: 设定 DOSBox 的环境设定.

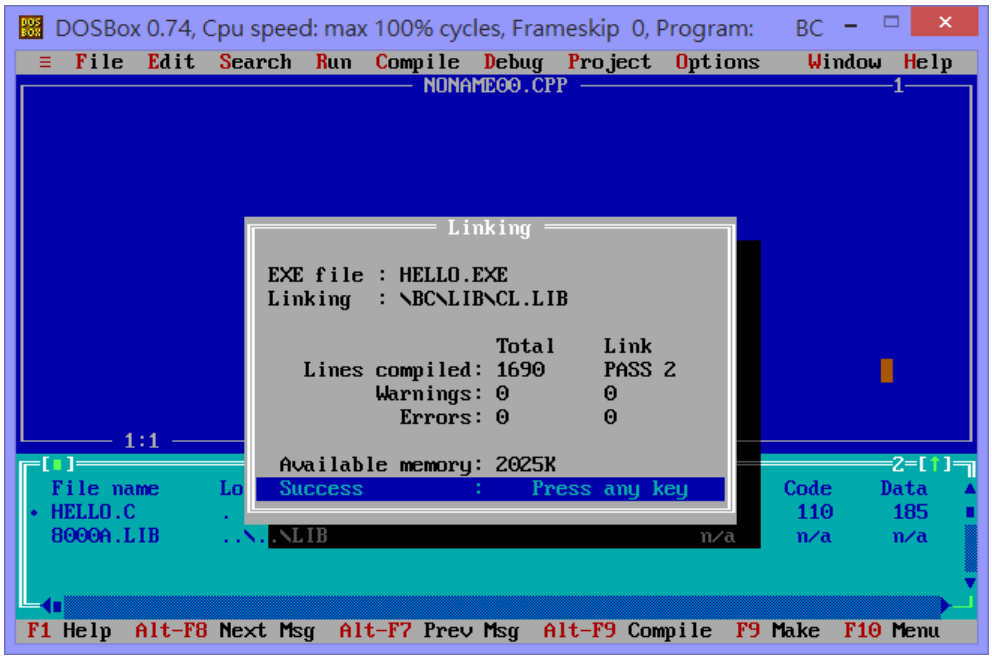
1. 确定 Demo 档案路径在哪，例如 D:\MiniOS7\hello\hello_c
2. 确定 BC\Bin 的档案路径，例如 D:\BC\Bin
3. 使用”mount”命令加载磁盘到 DOSBox，例如“mount d d:\”
4. 加载磁盘之后，输入“D:” ,可将“Z:\>”变成“D:\>”.
5. 设定编译器路径到 BC\Bin，例如 “path d:\bc\bin”
6. 变换路径到 Demo 所在的数据夹，然后输入“bc”

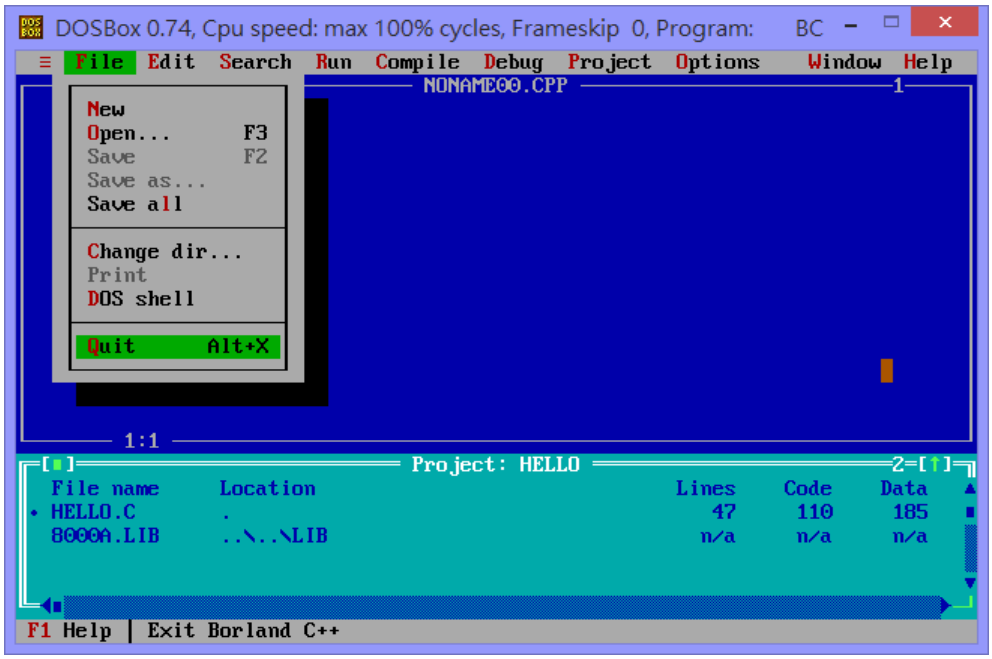


Step 4: 点击”Project” ➔ 选择 “*. prj “ 来打开专案档 ➔ 点击”Compile” 编译项目.


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: BC
File Edit Search Run Compile Debug Project Options Window Help
Open Project File
Open Project File
*.PRJ
OK
Files
HELLO.PRJ
..\
Cancel
Help
D:\MINIOS7\HELLO\HELLO_C*.PRJ
HELLO.PRJ 4881 Jan 7,2014 10:36am
F1 Help | Type for incremental search: lower for files, upper for directories


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: BC
File Edit Search Run Compile Debug Project Options Window Help
Compile Alt+F9
Make F9
Link
Build all
Information...
Remove messages
1:1
Project: HELLO
File name Location Lines Code Data
HELLO.C . 47 110 185
8000A.LIB ..\..\LIB n/a n/a n/a
F1 Help | Rebuild all files


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: BC
File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP
Linking
EXE file : HELLO.EXE
Linking : \BC\LIB\CL.LIB
Total Link
Lines compiled: 1690 PASS 2
Warnings: 0 0
Errors: 0 0
Available memory: 2025K
Success : Press any key
1:1
File name Lo Code Data
HELLO.C . 110 185
8000A.LIB ..\..\LIB n/a n/a
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

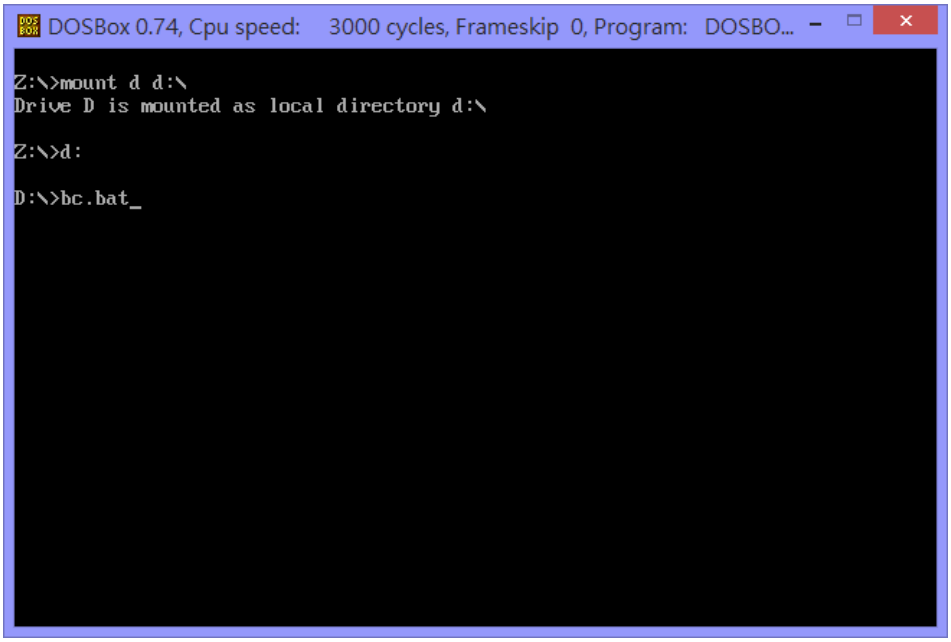

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: BC
File Edit Search Run Compile Debug Project Options Window Help
NONAME00.CPP
New
Open... F3
Save F2
Save as...
Save all
Change dir...
Print
DOS shell
Quit Alt+X
1:1
Project: HELLO
File name Location Lines Code Data
HELLO.C . 47 110 185
8000A.LIB ..\..\LIB n/a n/a n/a
F1 Help | Exit Borland C++

小技巧——使用 batch 檔:

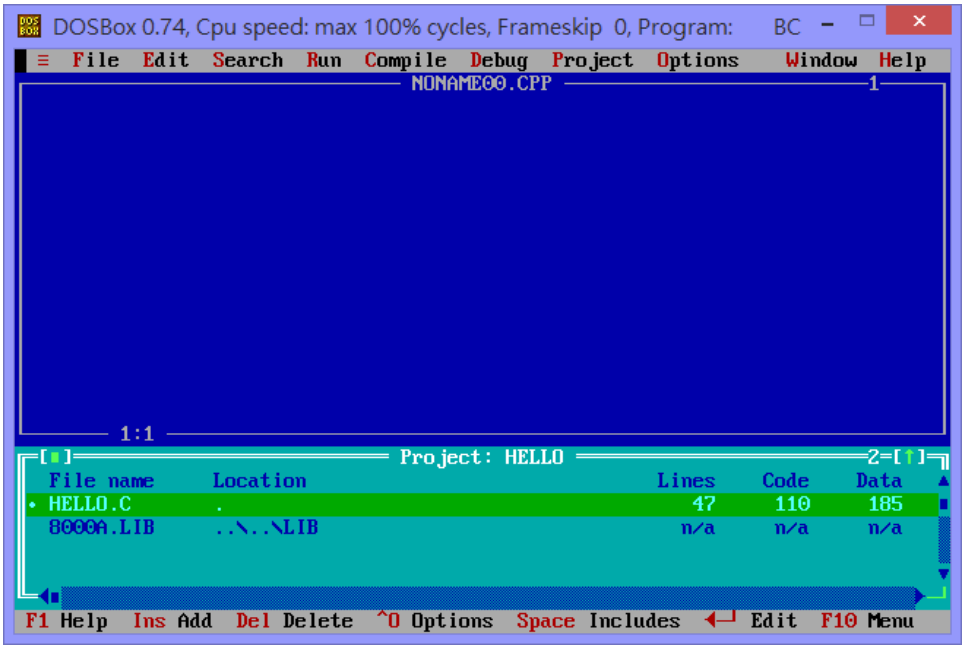
在 DOSBox Console 窗口中设定路径和呼叫编译器是有一点点令人感到麻烦的。我们可以制作一个 batch 档来省却麻烦,

```
path d:\bc\bin
cd minios7\hello\hello_c
bc
```

将此 batch 档储存在“D:\”路径下，当 D 槽被加载到 DOSBox 后，更换路径到 d:，执行 batch 檔。

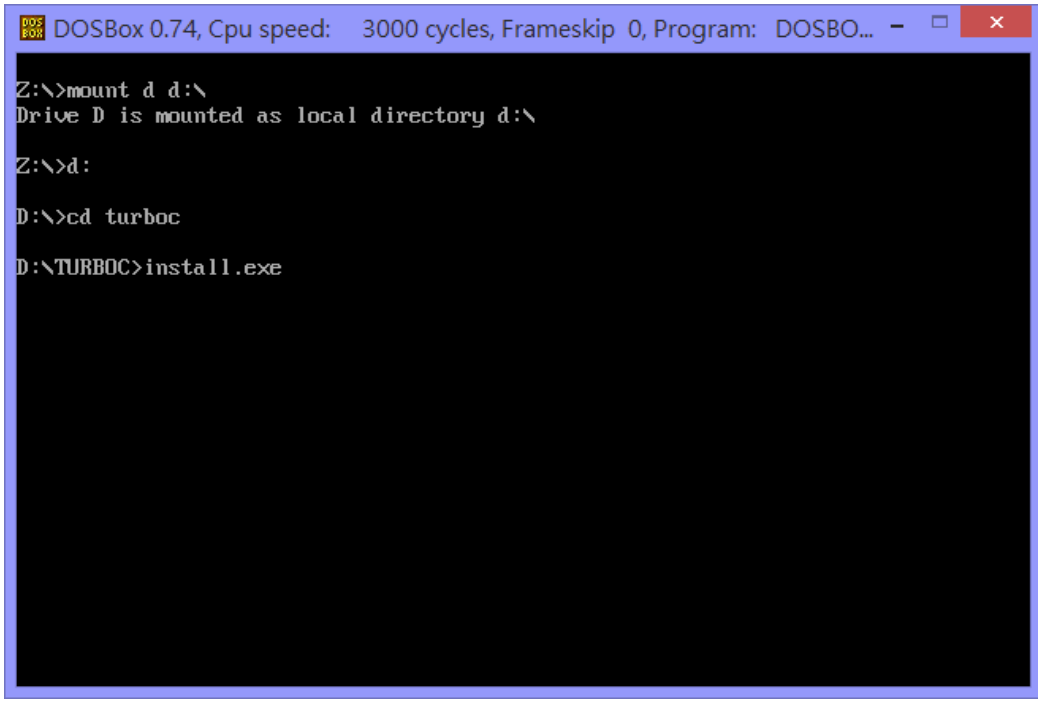


当执行 bc.bat 档后, Demo 项目会成功的被呼叫起来.

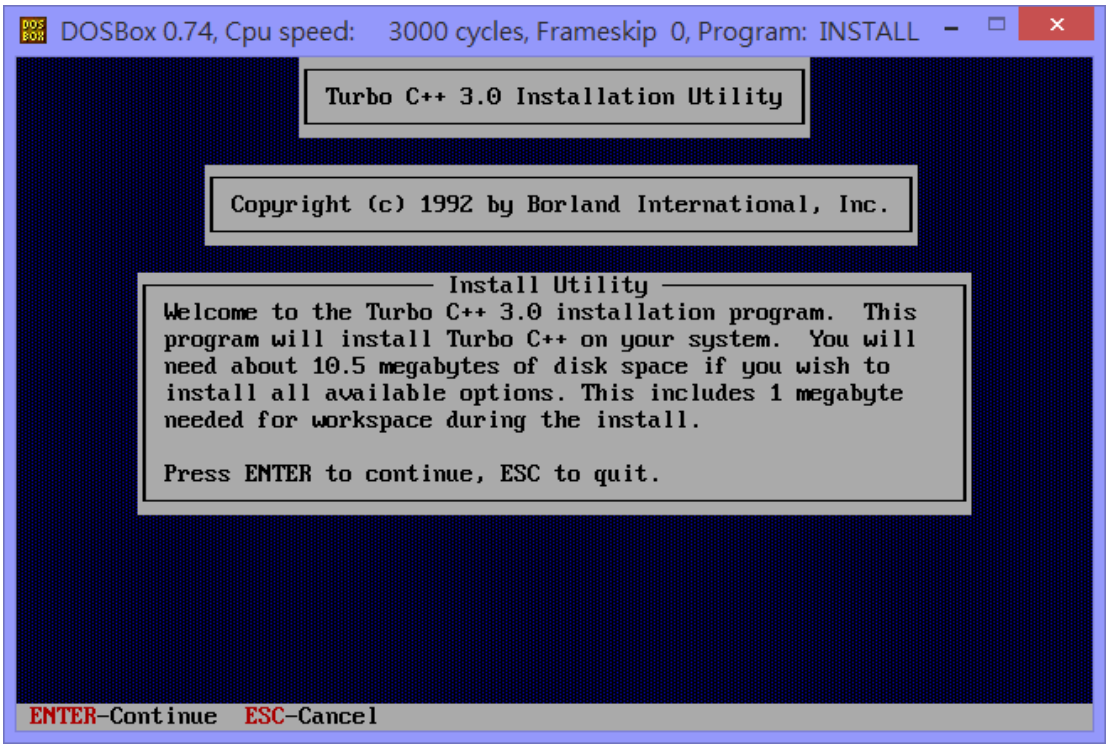


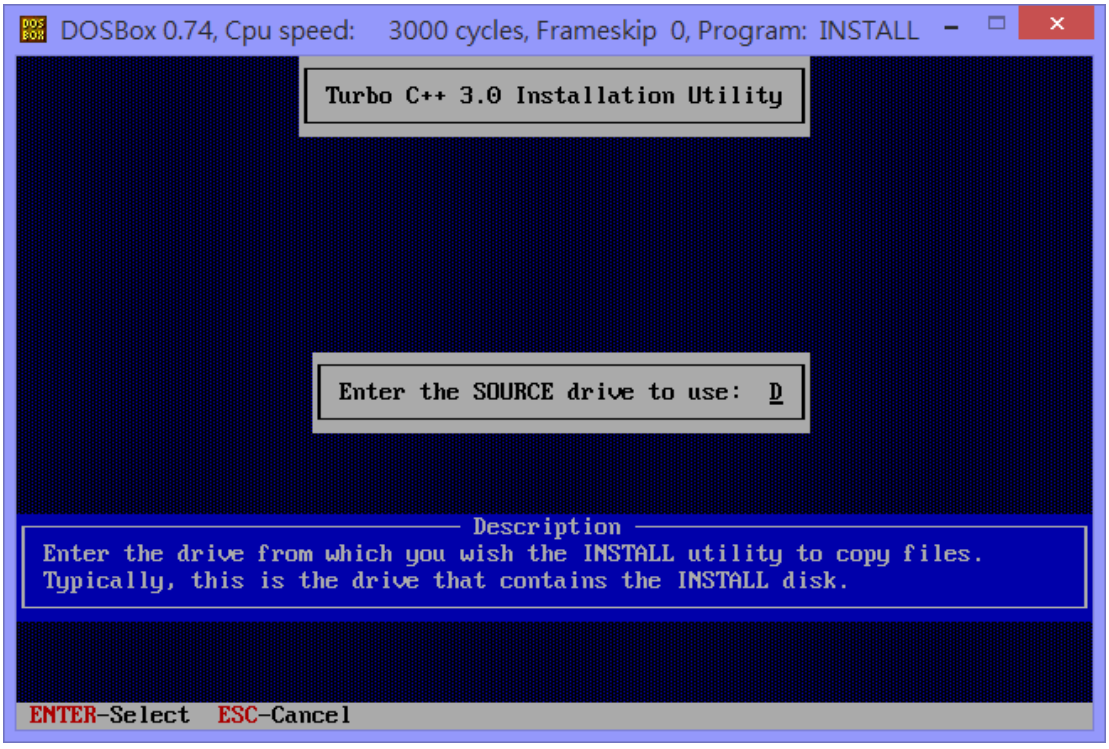
假设 BC3.1 编译器无法取得，可以从网络下载免费的 TurboC++3.0.
可以从如下的链接下载免费的 **turboc.zip**
http://www.bestfreewaredownload.com/download/t-free-turbo-c--freeware-flggsdpz.html

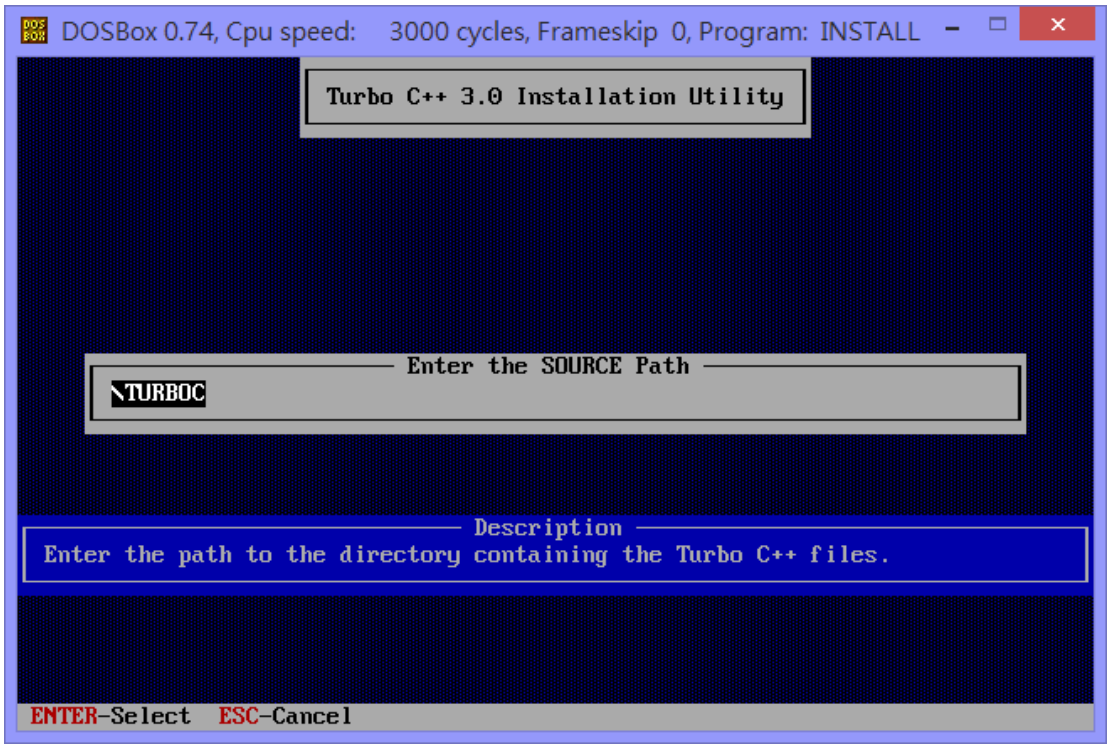
将 turboc.zip 解压缩后放到 D:\ 下，执行 DOSBox ➔ 执行 install.exe 安装 TC++3.0

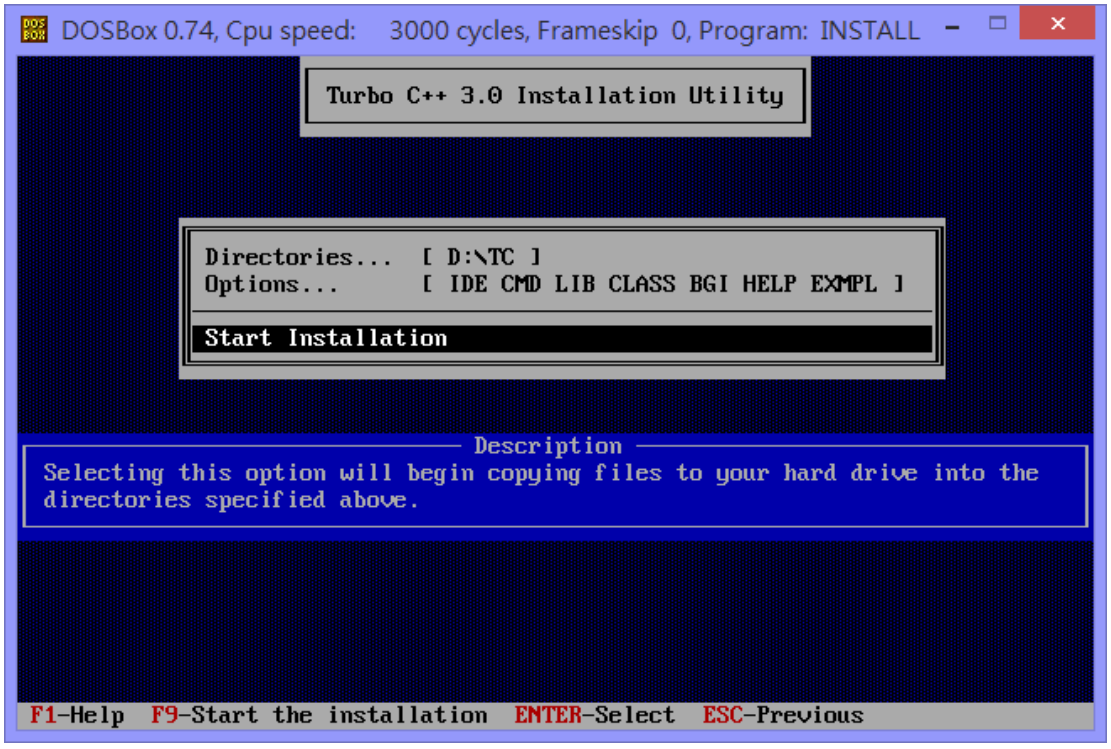


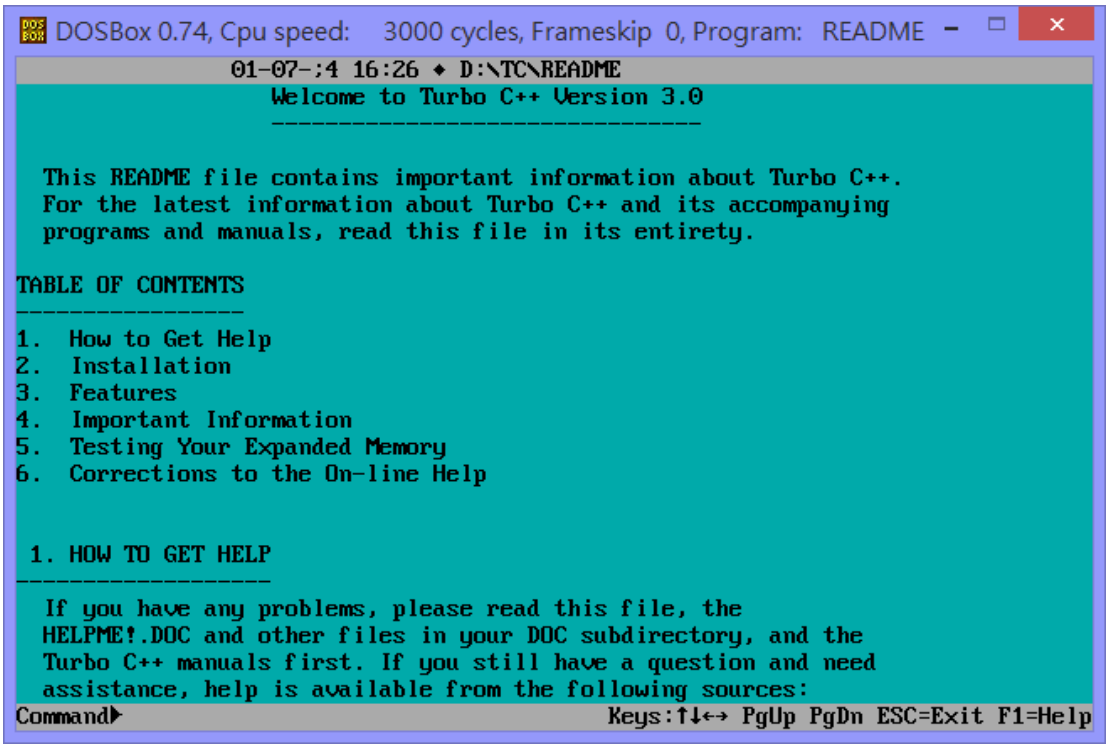
下面是 TC++3.0 安装步骤图解



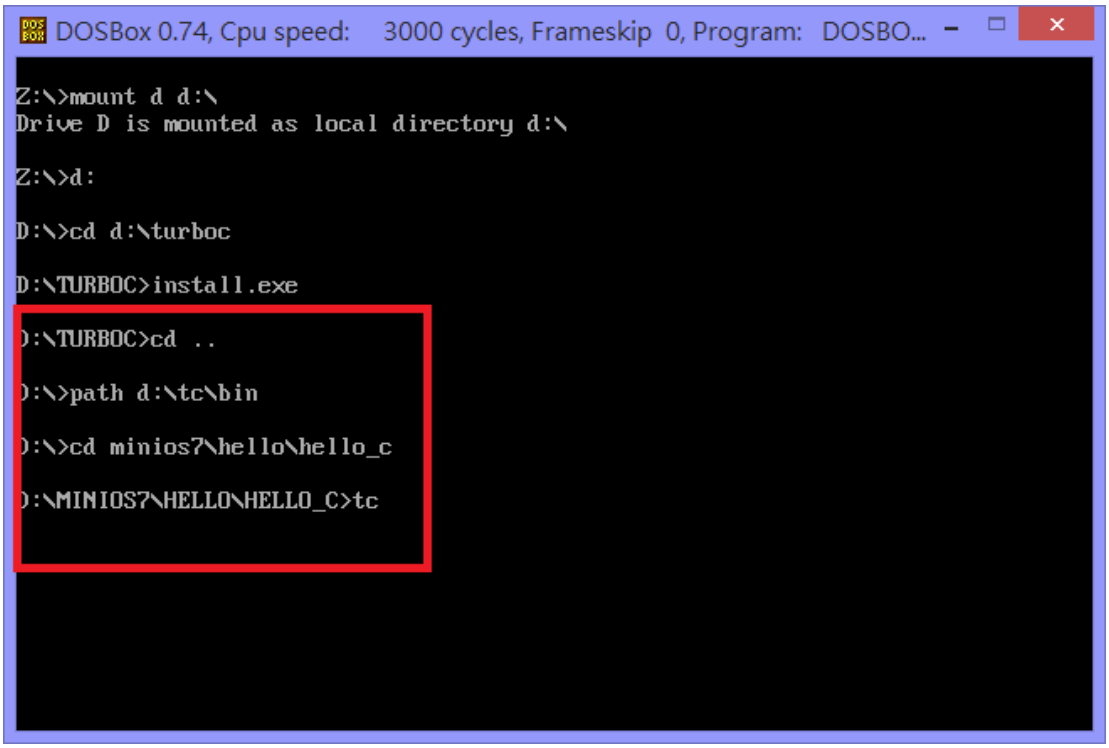


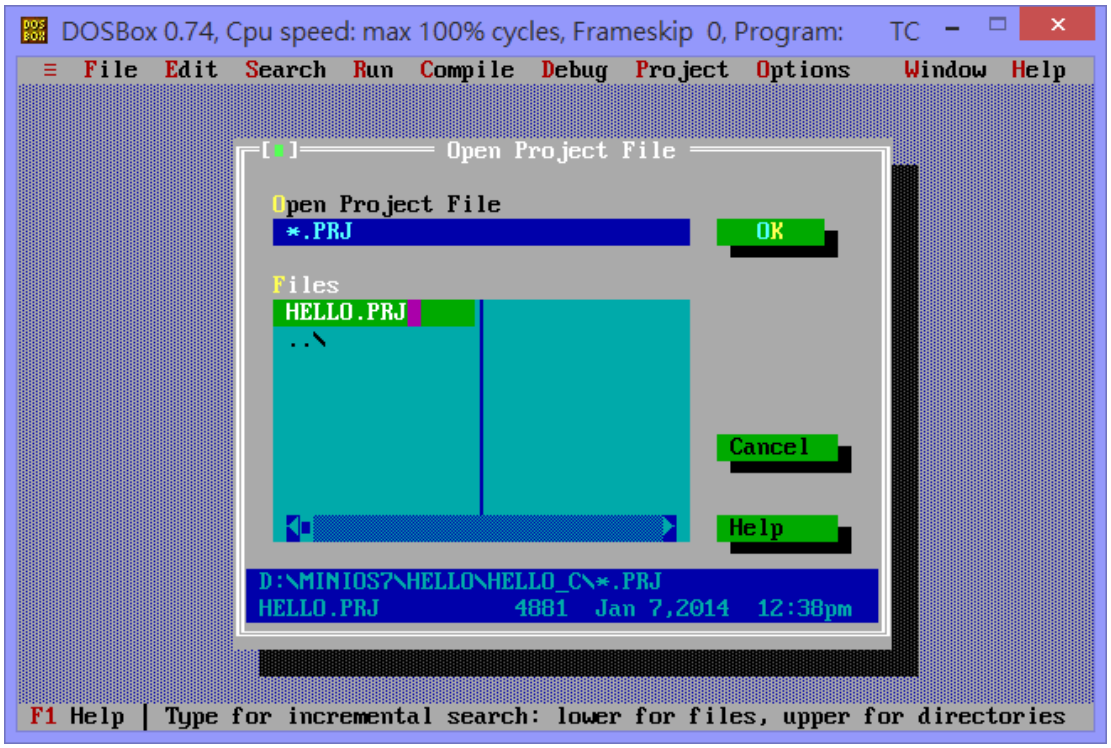

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: INSTALL
Turbo C++ 3.0 Installation Utility
Enter the SOURCE Path
\TURBOC
Description
Enter the path to the directory containing the Turbo C++ files.
ENTER-Select ESC-Cancel


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: INSTALL
Turbo C++ 3.0 Installation Utility
Directories... [D:\TC]
Options... [IDE CMD LIB CLASS BGI HELP EXMPL]
Start Installation
Description
Selecting this option will begin copying files to your hard drive into the directories specified above.
F1-Help F9-Start the installation ENTER-Select ESC-Previous



安装完 TC++3.0 之后，设定编译器路径和切换到 Demo 所在的数据夹下 ➔ 执行 TC 编译器.




DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
File Edit Search Run Compile Debug Project Options Window Help
Open Project File
Open Project File
*.PRJ
OK
Files
HELLO.PRJ
..\
Cancel
Help
D:\MINIOS7\HELLO\HELLO_C*.PRJ
HELLO.PRJ 4881 Jan 7,2014 12:38pm
F1 Help | Type for incremental search: lower for files, upper for directories

附录 F: 术语说明

1. 64 位唯一硬件序列号:

I-7188XC(D)具有一个 64 位唯一硬件序列号。这个数字是唯一的，并且不会与其它的 I-7188XC(D)控制器重号。应用软件可使用这个数字来验证硬件的有效性和软件的合法性。它是当前最低成本的 I-7188XC(D)保护装置。

2. AsicKey:

I/O 拓展总线支持 AsicKey。AsicKey 具有一个复杂的机制来完成有效性检查。为实现同样的目的其中还包括 128 字节的专用数据。它为软件的违法复制提供非常强的保护。每个合法用户具有一个唯一的和唯一的软件库，用户可自己检查这个 key，或者通过软件自动进行。总的来说，不可能移除 AsicKey 保护。

附录 G: 文件修订纪录

版本	出版日期	修正内容
1.0	12 月 2007	第一次出版
1.1	2 月 2012	1. 修改 DI 规范 2. 修改储存温度规格： 原：-40° C 至+80° C 新：-30° C 至+80° C 3. 修改湿度规格： 原：0～90％ 新：10～90％RH（无冷凝）
1.2	2 月 2014	1. 新增章节 3.4 “在 64 位平台建立工程(Project)” 2. 新增章节”在 64 位平台编译”于附录 E:编译和链接 3. 新增章节”附录 G: 文件修订纪录”