

# μPAC-7186EX User Manual

Version 1.1, January 2009

Service and usage information for



μPAC-7186EX



μPAC-7186EX-FD



μPAC-7186EX-SM



μPAC-7186EXD



μPAC-7186EXD-FD



μPAC-7186EXD-SM

Written by Liam Lin

Edited by Anna Huang

# Important Notices

## Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

## Warning

ICP DAS assumes no liability for any damage resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

## Copyright

Copyright © 2007 by ICP DAS Co., Ltd. All rights are reserved.

## Trademark

The names used for identification only may be registered trademarks of their respective companies.

# Table of Contents

---

Table of Contents.....	3
1. Introduction.....	6
1.1. Features.....	8
1.2. Specifications.....	13
1.3. Overview.....	19
1.4. Dimension.....	20
1.5. Companion CD .....	21
1.6. Comparing $\mu$ PAC-7186 Series with I-7188 .....	22
2. Quick Start.....	23
2.1. Hardware installation.....	23
2.1.1. Understanding terminal pin assignment/wiring diagram .....	23
2.1.2. Installing the $\mu$ PAC-7186EX.....	27
2.2. Software installation .....	29
2.3. MiniOS7 Utility for downloading programs .....	31
2.3.1. Establishing a connection between Host PC and the $\mu$ PAC-7186EX ..	31
2.3.2. Uploading and executing programs on $\mu$ PAC-7186EX .....	50
2.3.3. Making programs start automatically .....	51
2.4. MiniOS7 Utility for updating OS image.....	52
3. Your First Program on $\mu$ PAC-7186EX .....	55
3.1. Setting up the compiler .....	55
3.1.1. Installing the Compiler .....	57
3.1.2. Setting up the environment variables .....	62
3.2. API for $\mu$ PAC-7186EX.....	65
3.3. Build and run your first program.....	70
4. API and Demo Program Reference .....	83

4.1.	API for COM port .....	86
4.1.1.	Types of COM port functions .....	87
4.1.2.	API for MiniOS7 COM port .....	88
4.1.3.	API for standard COM port .....	93
4.1.4.	Comparing with MiniOS7 COM port function and Standard COM port function .....	97
4.1.5.	Request/Response protocol define on COM port .....	99
4.2.	API for I/O modules .....	100
4.3.	API for EEPROM .....	102
4.4.	API for Flash Memory .....	104
4.5.	API for NVRAM and RTC .....	107
4.6.	API for 5-Digit LED .....	110
4.7.	API for Timer .....	112
4.8.	API for WatchDog Timer (WDT) .....	115
4.9.	API for MiniOS7 File System (For $\mu$ PAC-7186EX-FD series only) .....	117
Appendix A.	Frame Ground .....	123
Appendix B.	What is MiniOS7 .....	125
Appendix C.	What is MiniOS7 Utility .....	126
Appendix D.	What is MiniOS7 File System (MFS) .....	127
Appendix E.	What is VxComm Utility .....	131
Appendix F.	More C Compiler Settings .....	132
F.1.	Turbo C 2.01 .....	132
F.2.	BC++ 3.1. IDE .....	136
F.3.	MSC 6.00 .....	140
F.4.	MSVC 1.50 .....	143
Appendix G.	Application of RS-485 Network .....	147
G.1.	Basic RS-485 network .....	147

G.2. Daisy chain RS-485 network.....	148
G.3. Star type RS-485 network.....	149
G.4. Random RS-485 network.....	151
G.5. $\mu$ PAC-7186EX Master-Slave Mode.....	152
G.5.1. $\mu$ PAC-7186EX as a Master .....	152
G.5.2. $\mu$ PAC-7186EX as a slave .....	153

# 1. Introduction

---

The  $\mu$ PAC-7186EX is a palm-size programmable automation controller that with Ethernet, RS-232 and RS-485 communication. ICP DAS provides easy-to-use Software development tool kits (Framework, Xserver, VxComm, Modbus function Library). Users can use them to easily integrate serial devices to have Ethernet/Internet communication ability and through the standard Modbus protocol to Communicate with SCADA software (Indusoft, ISaGARF, DasyLab, Trace Mode, Citect, iFix and so forth).

For the hardware, it also supports for I/O expansion bus interface. The I/O Expansion bus can be used to implement various I/O functions such as D/I, D/O, A/D, D/A, Timer/Counter, UART, flash memory, battery backup SRAM, ASIC key and other I/O functions. This I/O expansion bus can implement nearly all kinds of I/O functions, but only one expansion board can be added. There are more than 50 boards available for  $\mu$ PAC-7186EX series module so far.

## Package List

In addition to this manual, the shipping package includes the following items:

- One  $\mu$ PAC-7186EX module
- One download cable (CA-0910)
- One companion CD containing software drivers and digital versions of the user manuals
- One copy of the release notes



## 1.1. Features

---

### ➤ Support for Virtual COM technology

PC can create virtual COM ports to map the RS-232, RS-485 of  $\mu$ PAC-7186EX series module using the VxComm technology. The software running on the PC can operate the virtual COM ports like a standard COM port to access the serial devices connect to the  $\mu$ PAC-7186EX. In other words, the original software developed for the serial devices can access the serial devices via the Ethernet/Internet without any modification. Each PC can control up to 256 COM ports (including real COM ports). Using the I/O expansion board, each  $\mu$ PAC-7186EX can have up to 8 COM ports.

### ➤ Support Modbus Protocol

Using the Modbus firmware,  $\mu$ PAC-7186EX offers following Modbus features:

1. Modbus/TCP/RTU/ASCII slave
2. Modbus/TCP/RTU/ASCII master
3. Gateway for Modbus/TCP to Modbus/RTU

### ➤ Ethernet Protocols

TCP, UDP, IP, ICMP and ARP.



➤ **VxComm Technique Supported**

VxComm technique is used to create virtual COM ports on PC (for windows 2K/XP to map remote COM ports of PDS-700, I-7188E, I-8000 and  $\mu$ PAC-7186EX over the Ethernet. Using the technique, RS-232/485 software can access devices locally (via the physical RS-232/485 bus) or remotely (via the Ethernet). The RS-232/485 software only needs to change COM port number from the physical COM port to virtual COM port.

➤ **Easy-Use Software Development Tool Kits (Using C Language)**

The custom firmware can be developed for  $\mu$ PAC-7186EX series module using the SDK (Framework, Xserver, Modbus function libray) provided by ICP DAS.

➤ **Support Web configuration**

$\mu$ PAC-7186EX series module has a build-in web server for configuration. You can use standard web browsers (such as IE, Netscape, Firefox, and etc) to configure its Ethernet and COM ports configurations.

➤ **Remote Configuration/Maintenance**

$\mu$ PAC-7186EX series module can be operated via the Ethernet (TCP/IP or UDP) or RS-232, to allow tasks such as downloading files, configuration updating the MiniOS7 image etc.

➤ **Built-in Watchdog Timer (WDT)**

μPAC-7186EX series module includes an internal watchdog timer (WDT). The watchdog timer will trigger a system reset if the main program fails or neglects to regularly service the watchdog. The intention is to bring the system back from the hung state into normal operation.

➤ **I/O Expansion Bus Interface**

The μPAC-7186EX series module supports the use of an I/O Expansion bus to add a single I/O Expansion Board. ICP DAS provides all function libraries for I/O Expansion Boards to enable easy use of the I/O Expansion Board functions.

➤ **MinOS7 File System (MFS) – For μPAC-7186EX-FD series only**

MFS implements a reliable file system with C language API for embedded data logger applications on MiniOS7

1. Can dynamically read/write/append data to files continuously

The 64MB flash memory is divided to 2 disks, each disk can store 456 files max.. You can create files and then write/append data to it. Then read data in the file and forward to PC for posted analysis when the data is complete collected.

## 2. Provides C language API

Following functions are similar to the functions that turbo C and Borland C provide. This helps users short the learning of MFS.

mfs\_OpenFile, mfs\_CloseFile, mfs\_ReadFile, mfs\_WriteFile, mfs\_Gets, mfs\_Puts, mfs\_Getc, mfs\_Putc, mfs\_EOF, mfs\_Seek, mfs\_Tell, mfs\_DeleteFile, mfs\_DeleteAllFiles, mfs\_GetFileInfoByName, mfs\_GetFileInfoByNo, ... etc.

## 3. Writing Verification

Data written to flash memory are read back to verify its correction.

The function can be disabled to increase writing speed. But for data safety, we recommend users to enable the function.

## 4. Automate file system recovery in the event of unexpected reset or power losses

When an unexpected reset or power loss occurs, closed files, and files opened for reading are never at risk. Only writing data has risk to be lost.

MFS writes data to the flash memory just after executing writing functions (such as mfs\_WriteFile, mfs\_Puts, mfs\_Putc, etc.). And meanwhile, MFS stores important information (such as file name, pointer, flash location, etc) to NVRAM (non-volatile random access memory). When an unexpected reset or power loss occurs, only data written since the last writing operation (such as mfs\_WriteFile, mfs\_Puts, mfs\_Putc, etc.) could be lost. When the MFS reboots, it refers the information stored in the NVRAM to restores the file system. The un-closed writing file will be automatically closed and all its data written before the last writing operation will be safe.

μPAC-7186EX series module has more features as followings:

➤ RoHS Compliance and CE Certification

- Low Power Input (10 to 30VDC) according to industrial environment
- Frame-Ground design for ESD protection
- Fire Retardant Materials (UL94-V0 Level) and Robust Case
- VxComm Driver for Windows NT 4.0, 2000/XP/2003 and Vista32

## 1.2. Specifications

---

### ➤ $\mu$ PAC-7186EX/ $\mu$ PAC-7186EXD

PACs	$\mu$ PAC-7186EX	$\mu$ PAC-7186EXD
<b>CPU Specification</b>		
CPU	80186 CPU, 80MHz or compatible	
SRAM	512K Bytes	
Flash	512K Bytes Erase unit is one sector (64K bytes); 100,000 erase/write cycles	
EEPROM	16K Bytes Data retention: 40 years; 1,000,000 erase/write cycles	
NVRAM	31 Bytes Battery backup, data valid up to 10 year	
RTC (Real Time Clock)	Year-2000 compliance; seconds, minutes, hours, date of the month; month, year, valid up from 1980 to 2079	
Hardware Serial Number	Yes	
Build-in Watchdog Timer	Yes	
<b>Communication Interface</b>		
COM 1	RS-232 (TXD, RXD, CTS, RTS and GND); Non-isolation	
COM 2	RS-485 (D2+, D2-; self-tuner ASIC inside); Non-isolation	
Ethernet Port	10/100Base-TX Ethernet Controller	

	(Auto-negotiating, Auto_MDIX, LED indicator)	
<b>COM Port Formats</b>		
Data bit	7, 8	
Parity	None, Even, Odd, Mark, Space	
Stop bit	1	
Baud Rate	115200 bps Max.	
<b>LED Display</b>		
5-digit 7-segment LED	No	Yes
System LED Indicator	Yes	
<b>Hardware Expansion</b>		
I/O expansion bus	Yes	
User defined I/O pins	14 pins	
<b>Operating Environment</b>		
Operating temperature	-25°C to +75°C (-13°F to +167°F)	
Storage Temperature	-40°C to +80°C (-40°F to +176°F)	
Humidity	5% to 95%, Non-condensing	
<b>Power</b>		
Protection	Power reverse polarity protection	
Frame Ground	Yes (for ESD Protection)	
Required Supply Voltage	+10VDC to +30VDC (non-regulated)	
Power Consumption	1.5W	2.5W
<b>Dimensions</b>	123mm x 72mm x 33mm	

➤ **μPAC-7186EX-FD/μPAC-7186EXD –FD**

PACs	μPAC-7186EX-FD	μPAC-7186EXD-FD
<b>CPU Specification</b>		
CPU	80186 CPU, 80MHz or compatible	
SRAM	512K Bytes	
Flash	512K Bytes Erase unit is one sector (64K bytes); 100,000 erase/write cycles	
<b>NAND Flash</b>	<b>64M Bytes</b> Data retention: 10 years; 100,000 erase/write cycles	
EEPROM	16K Bytes Data retention: 40 years; 1,000,000 erase/write cycles	
NVRAM	31 Bytes Battery backup, data valid up to 10 year	
RTC (Real Time Clock)	Year-2000 compliance; seconds, minutes, hours, date of the month; month, year, valid up from 1980 to 2079	
Hardware Serial Number	Yes	
Build-in Watchdog Timer	Yes	
<b>Communication Interface</b>		
COM 1	RS-232 (TXD, RXD, CTS, RTS and GND); Non-isolation	
COM 2	RS-485 (D2+, D2-; self-tuner ASIC inside); Non-isolation	

Ethernet Port	10/100Base-TX Ethernet Controller (Auto-negotiating, Auto_MDIX, LED indicator)	
<b>COM Port Formats</b>		
Data bit	7, 8	
Parity	None, Even, Odd, Mark, Space	
Stop bit	1	
Baud Rate	115200 bps Max.	
<b>LED Display</b>		
5-digit 7-segment LED	No	Yes
System LED Indicator	Yes	
<b>Hardware Expansion</b>		
I/O expansion bus	Yes	
User defined I/O pins	14 pins	
<b>Operating Environment</b>		
Operating temperature	-25°C to +75°C (-13°F to +167°F)	
Storage Temperature	-40°C to +80°C (-40°F to +176°F)	
Humidity	5% to 95%, Non-condensing	
<b>Power</b>		
Protection	Power reverse polarity protection	
Frame Ground	Yes (for ESD Protection)	
Required Supply Voltage	+10VDC to +30VDC (non-regulated)	
Power Consumption	2W	3W
<b>Dimensions</b>	123mm x 72mm x 33mm	

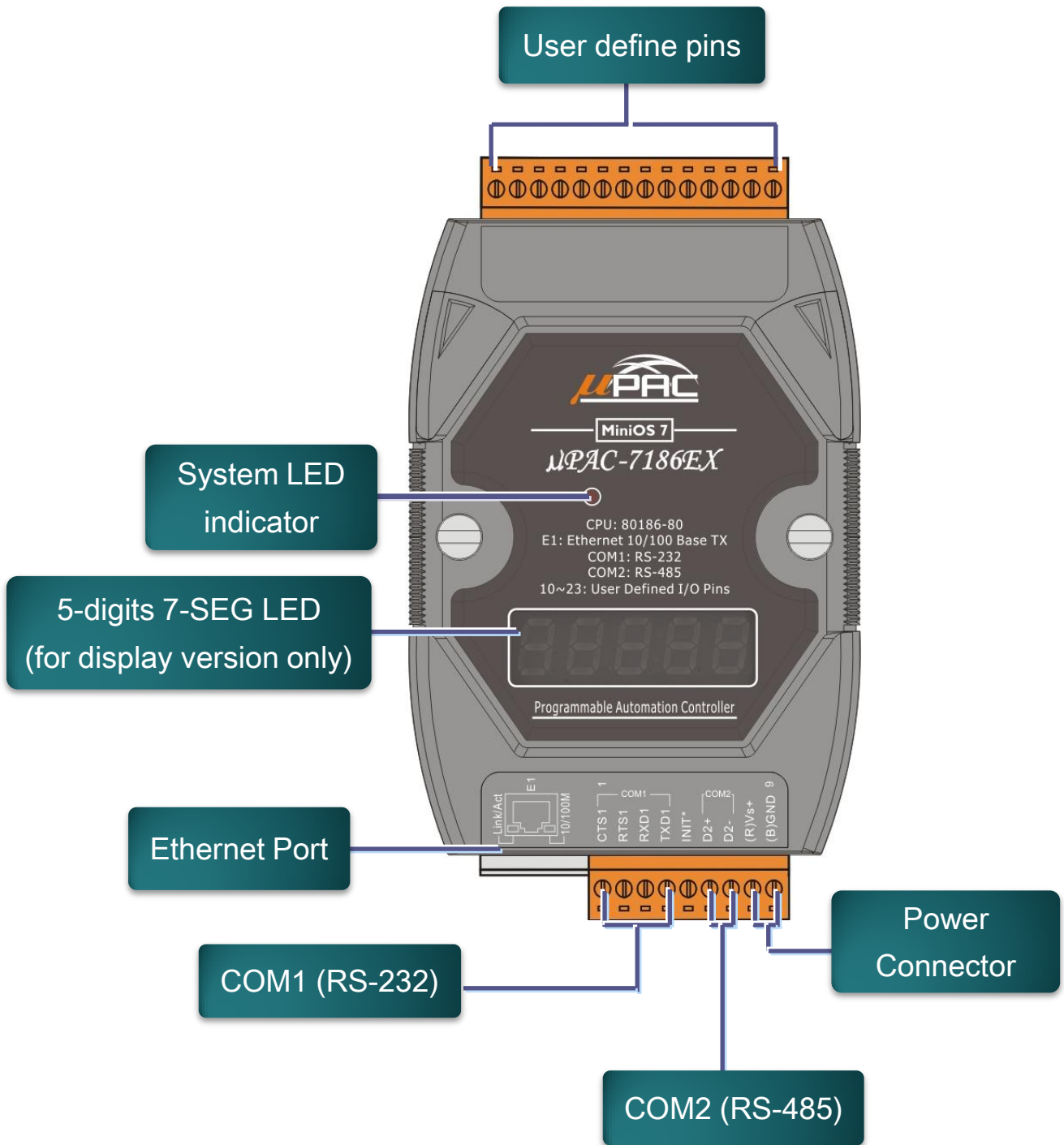


➤ **μPAC-7186EX-SM/μPAC-7186EXD-SM**

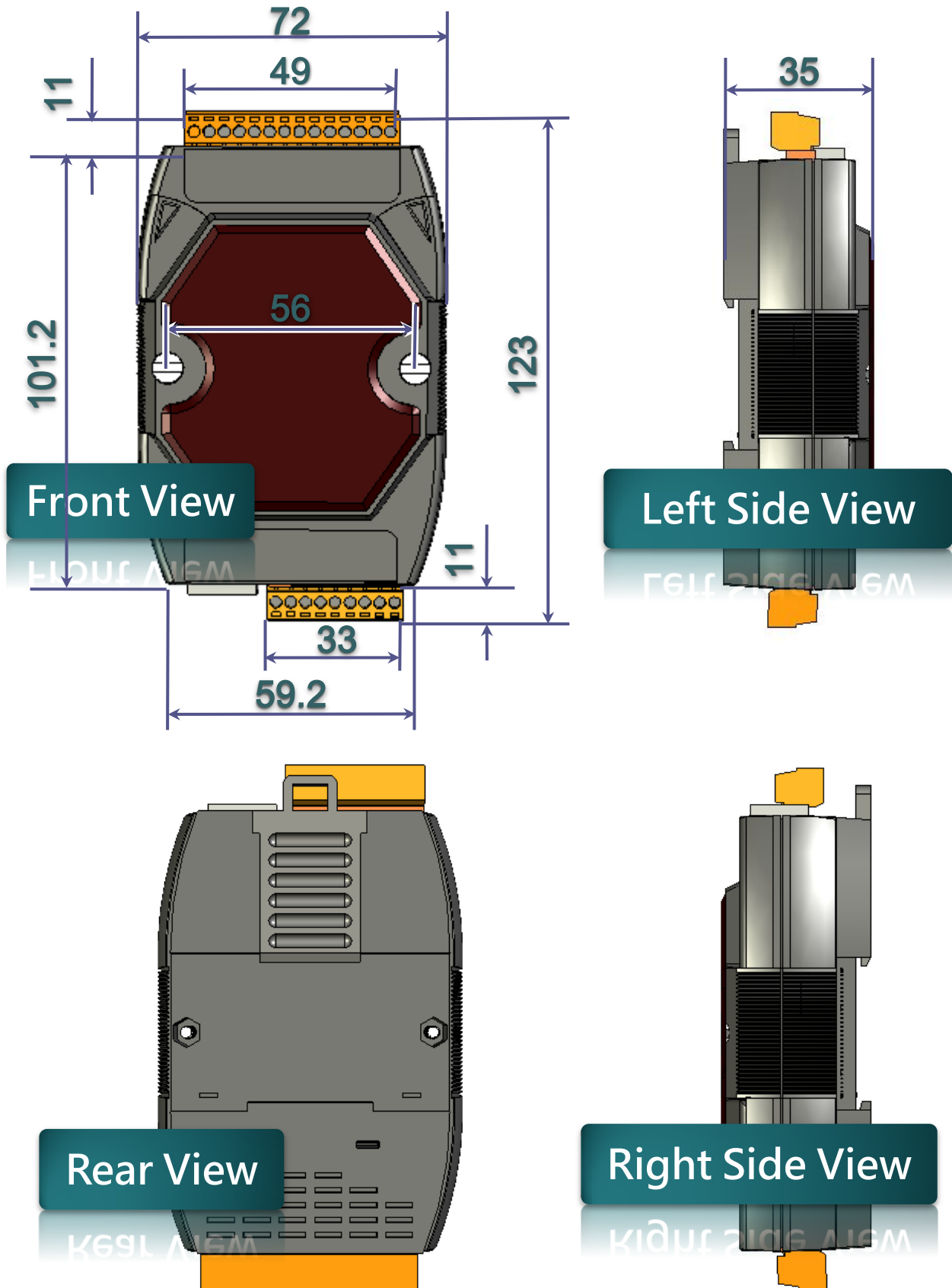
PACs	μPAC-7186EX-SM	μPAC-7186EXD-SM
<b>CPU Specification</b>		
CPU	80186 CPU, 80MHz or compatible	
SRAM	<b>640K Bytes</b>	
EEPROM	16K Bytes Data retention: 40 years; 1,000,000 erase/write cycles	
NVRAM	31 Bytes Battery backup, data valid up to 10 year	
RTC (Real Time Clock)	Year-2000 compliance; seconds, minutes, hours, date of the month; month, year, valid up from 1980 to 2079	
Hardware Serial Number	Yes	
Build-in Watchdog Timer	Yes	
<b>Communication Interface</b>		
COM 1	RS-232 (TXD, RXD, CTS, RTS and GND); Non-isolation	
COM 2	RS-485 (D2+, D2-; self-tuner ASIC inside); Non-isolation	
Ethernet Port	10/100Base-TX Ethernet Controller (Auto-negotiating, Auto_MDIX, LED indicator)	
<b>COM Port Formats</b>		
Data bit	7, 8	
Parity	None, Even, Odd, Mark, Space	
Stop bit	1	

Baud Rate	115200 bps Max.	
<b>LED Display</b>		
5-digit 7-segment LED	No	Yes
System LED Indicator	Yes	
<b>Hardware Expansion</b>		
I/O expansion bus	Yes	
User defined I/O pins	14 pins	
<b>Operating Environment</b>		
Operating temperature	-25°C to +75°C (-13°F to +167°F)	
Storage Temperature	-40°C to +80°C (-40°F to +176°F)	
Humidity	5% to 95%, Non-condensing	
<b>Power</b>		
Protection	Power reverse polarity protection	
Frame Ground	Yes (for ESD Protection)	
Required Supply Voltage	+10VDC to +30VDC (non-regulated)	
Power Consumption	2W	3W
<b>Dimensions</b>	123mm x 72mm x 33mm	

# 1.3. Overview



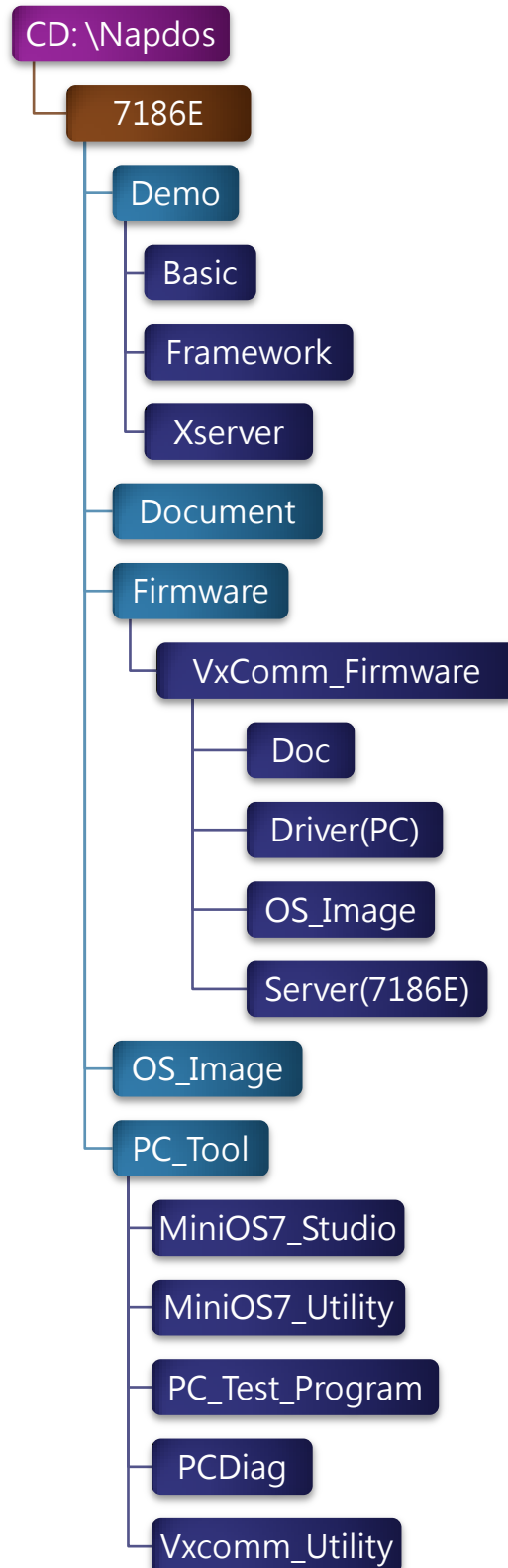
## 1.4. Dimension



## 1.5. Companion CD

---

This package comes with a CD that includes the following software and documentation:



## 1.6. Comparing $\mu$ PAC-7186 Series with I-7188

---

The table below shows a comparison of differences between I-7188 and  $\mu$ PAC-7186 series device families.

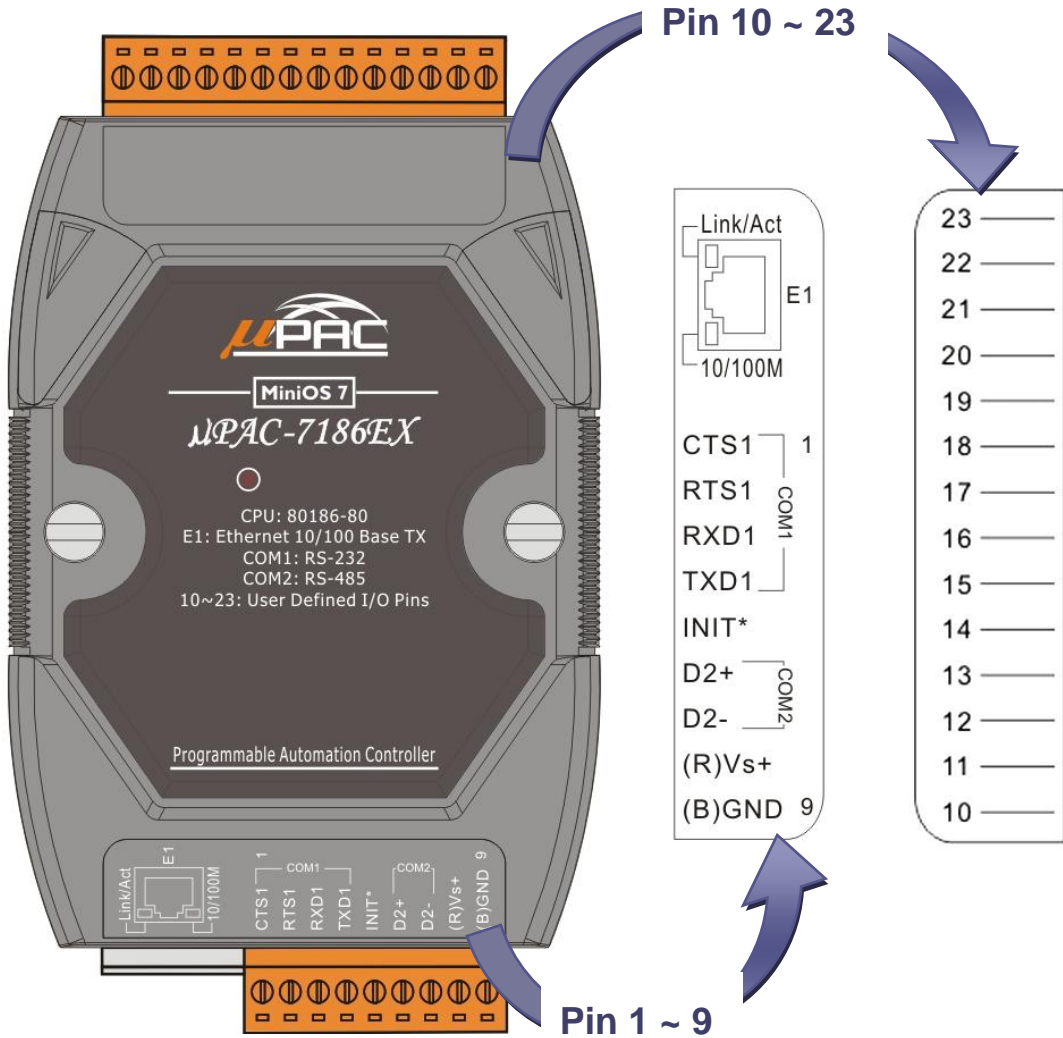
Specification Device	CPU	SRAM	EEPROM	NAND Flash	I/O Expansion Bus	DI	DO	Ethernet Port
I-7188EA(D)	40M Hz	512K	2K	-	For memory board only	6	7	10 Base-T
I-7188EX(D)	40M Hz	512K	2K	-	Yes	-	-	10 Base-T
I-7188EF(D)- 016	40M Hz	512K	2K	-	No	-	-	10 Base-T
$\mu$ PAC- 7186EX(D)	80M Hz	512K	16K	-	Yes	-	-	10 Base-T
$\mu$ PAC- 7186EX(D)-FD	80M Hz	512K	16K	640MB	Yes	-	-	10/100 Base
$\mu$ PAC- 7186EX(D)-SM	80M Hz	512K	16K	-	Yes	-	-	10/100 Base
$\mu$ PAC- 7186EX(D)- CAN	80M Hz	640K	16K	-	Yes	-	-	10/100 Base

# 2. Quick Start

This chapter provides users with basic information needed to begin installing and maintaining the  $\mu$ PAC-7186EX.

## 2.1. Hardware installation

### 2.1.1. Understanding terminal pin assignment/wiring diagram



The below table shows pin assignment for pin 1 ~ 9:

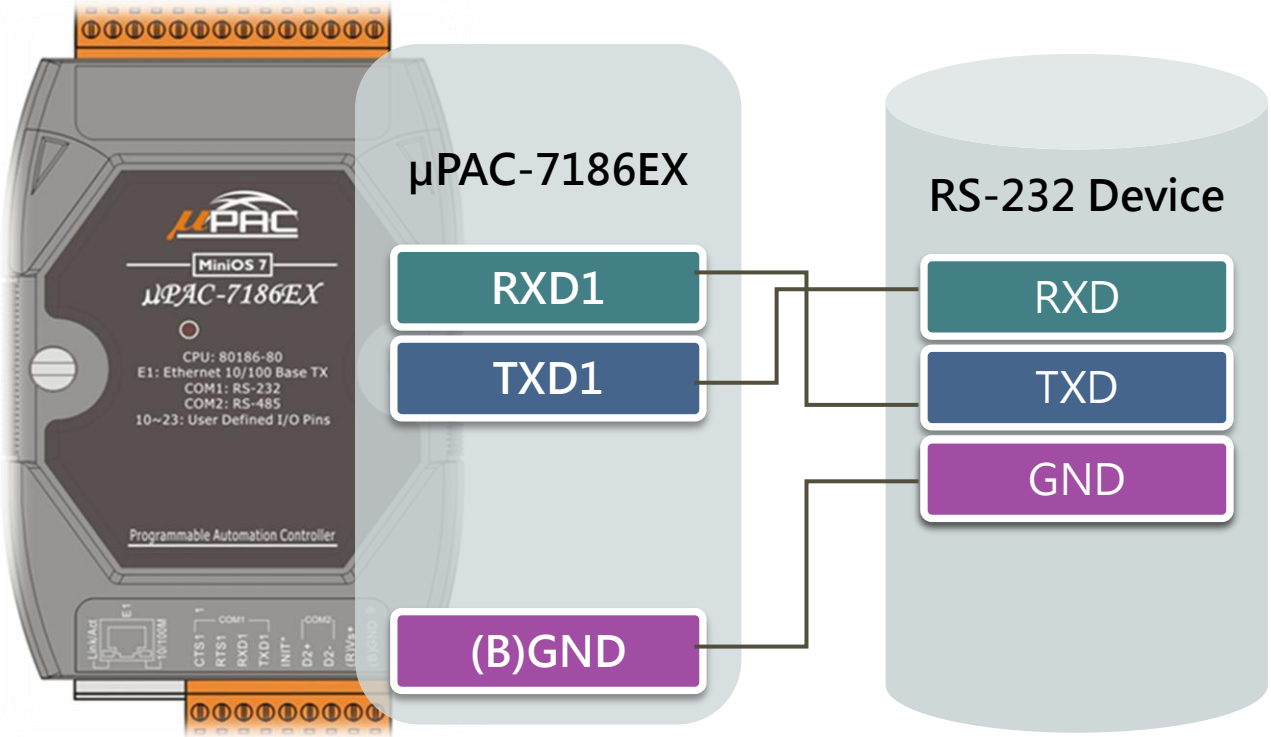
Pin	Name	Description
1	CTS1	CTS pin for COM1
2	RTS1	RTS pin for COM1
3	RXD1	RXD pin for COM1
4	TXD1	TXD pin for COM1
5	INIT*	Initial pin
6	D2+	DATA+ pin for COM2
7	D2-	DATA- pin for COM2
8	Vs+	V+ of power supply (+10 to +30VDC, unregulated)
9	GND	GND for the power supply

The below table shows pin assignment for pin 10 ~ 23:

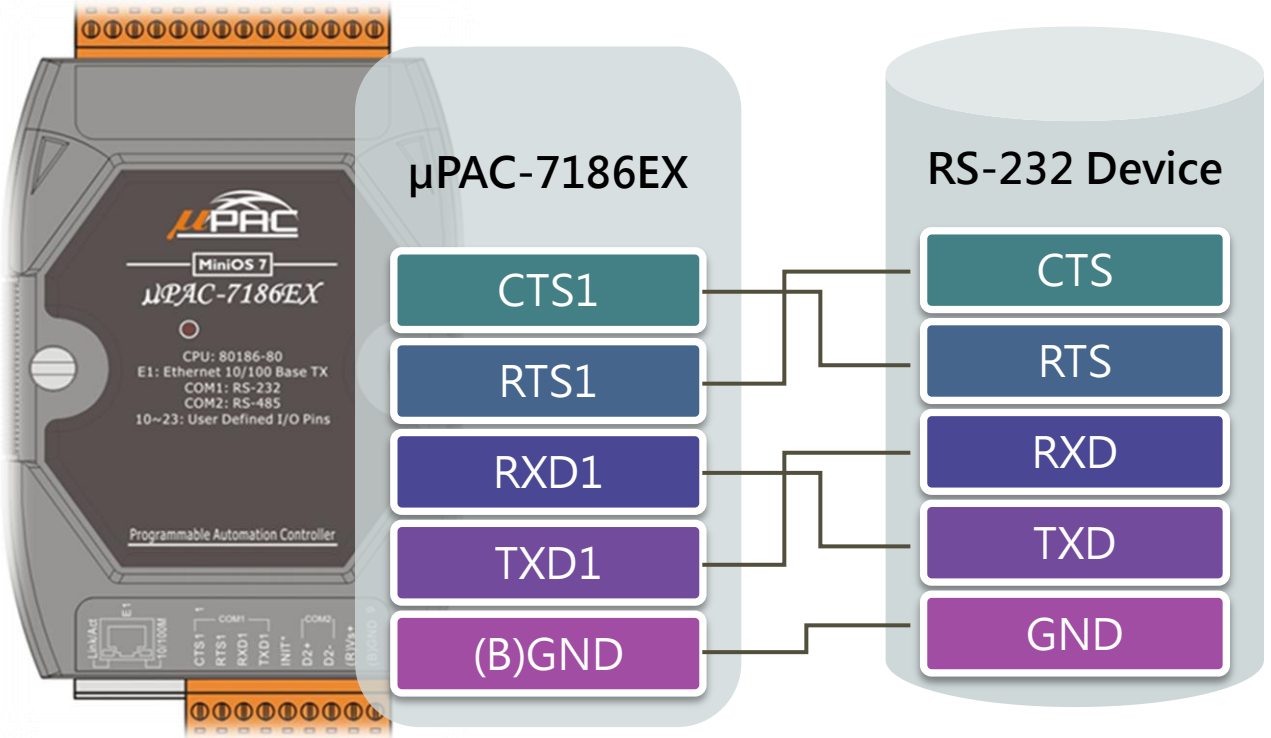
Pin	Name	Description	Pin	Name	Description
10	Pin 10	User defined pin 10	17	Pin 17	User defined pin 17
11	Pin 11	User defined pin 11	18	Pin 18	User defined pin 18
12	Pin 12	User defined pin 12	19	Pin 19	User defined pin 19
13	Pin 13	User defined pin 13	20	Pin 20	User defined pin 20
14	Pin 14	User defined pin 14	21	Pin 21	User defined pin 21
15	Pin 15	User defined pin 15	22	Pin 22	User defined pin 22
16	Pin 16	User defined pin 16	23	Pin 23	User defined pin 23



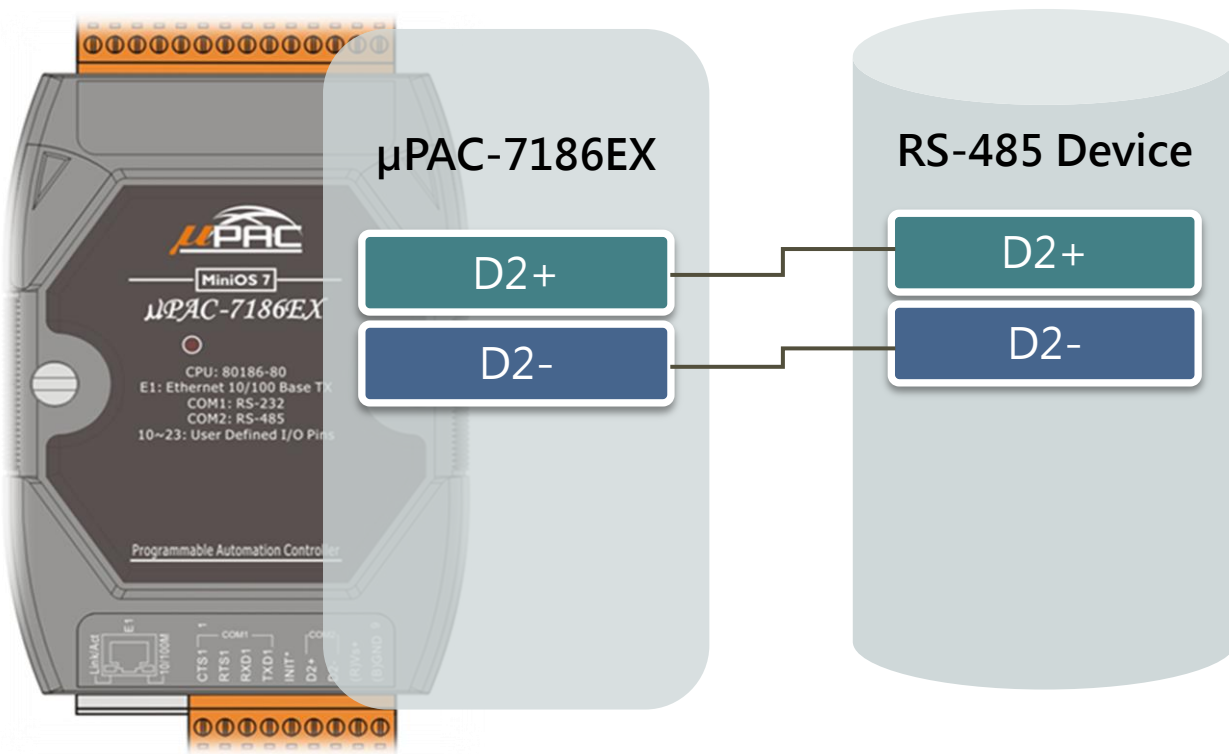
The figure below shows the wiring connections used for the 3-wire RS-232 port:



The figure below shows the wiring connections used for the 5-wire RS-232 port:



The figure below shows the wiring connections used for the RS-485 port:

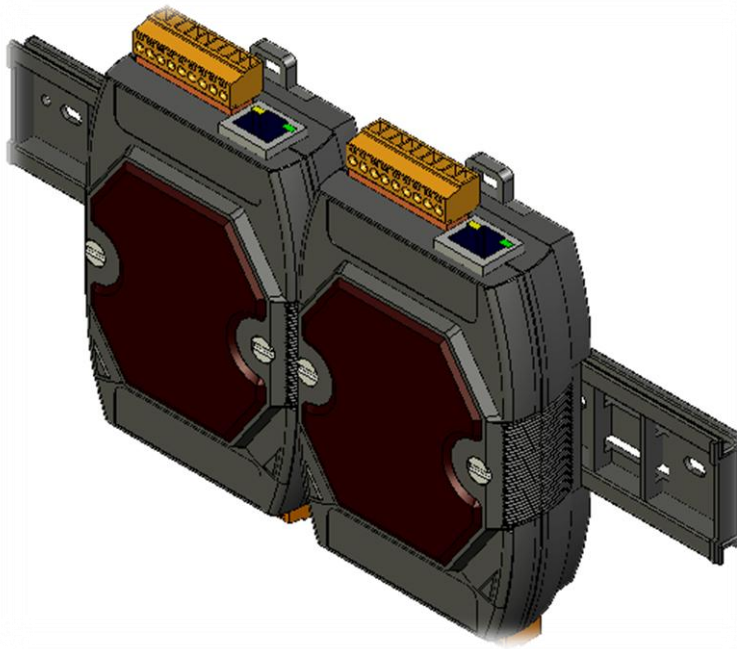


## 2.1.2. Installing the $\mu$ PAC-7186EX

### Step 1: Mounting the $\mu$ PAC-7186EX

The  $\mu$ PAC-7186EX can either be mounted on DIN-rail or stack.

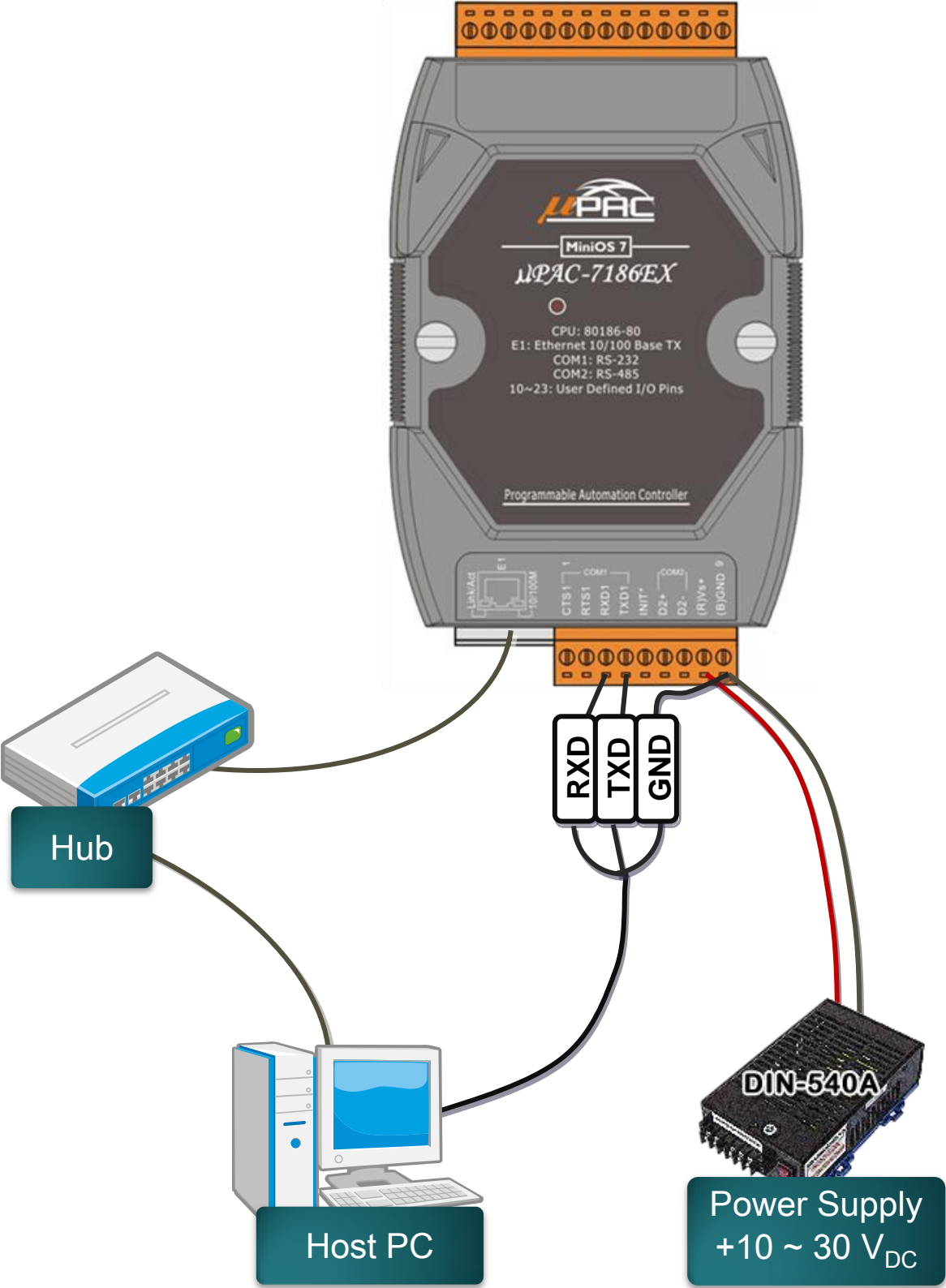
#### 1: DIN-rail mounting



#### 2: Stack mounting



Step 2: Connecting the Host PC to the  $\mu$ PAC-7186EX



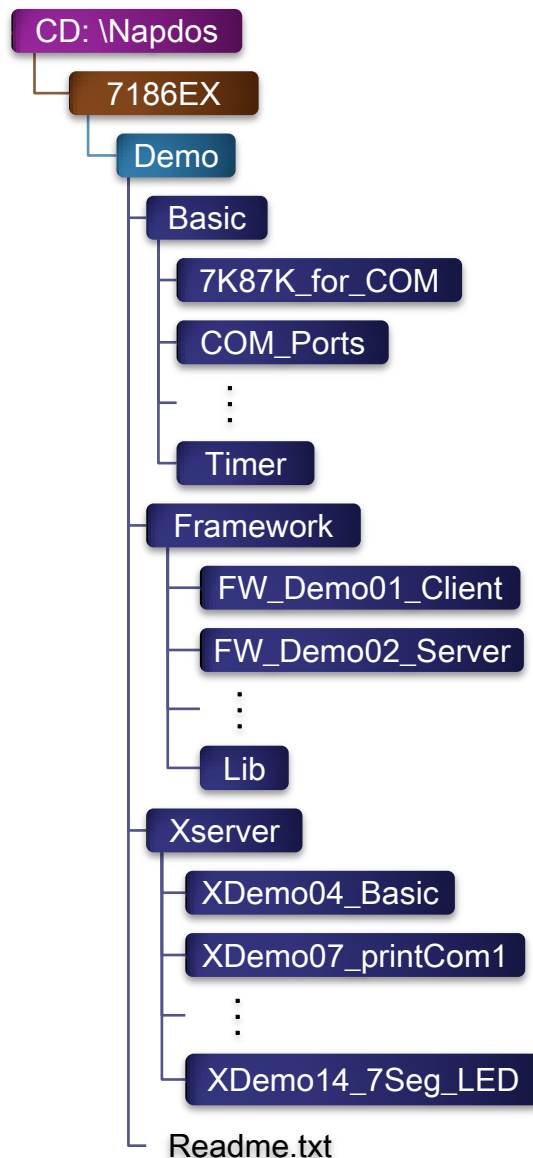
## 2.2. Software installation

---

All software resources are included on the companion CD, the following steps will help you to install the resources and software from the companion CD.

### Step 1: Copy the “Demo” folder from the companion CD to the Host PC

The folder is an essential resource for users developing custom programs which contains libraries, header files, demo programs and more information as shown below:



## Step 2: Install the MiniOS7 Utility

The MiniOS7 Utility is a tool that can be used to configure and upload files to the controller and is located at:

CD:\Napdos\minios7\utility\minios7\_utility\

[ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/minios7\\_utility/](ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/minios7_utility/)

## 2.3. MiniOS7 Utility for downloading programs

---

Before you begin using the MiniOS7 Utility to download programs, ensure that the controller is connected to the Host PC.

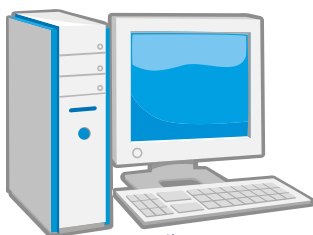
The download process has the following main steps: .

1. Establishing a connection
2. Download and executing programs on the controller
3. Making programs start automatically

All of these main steps will be described in detail later.

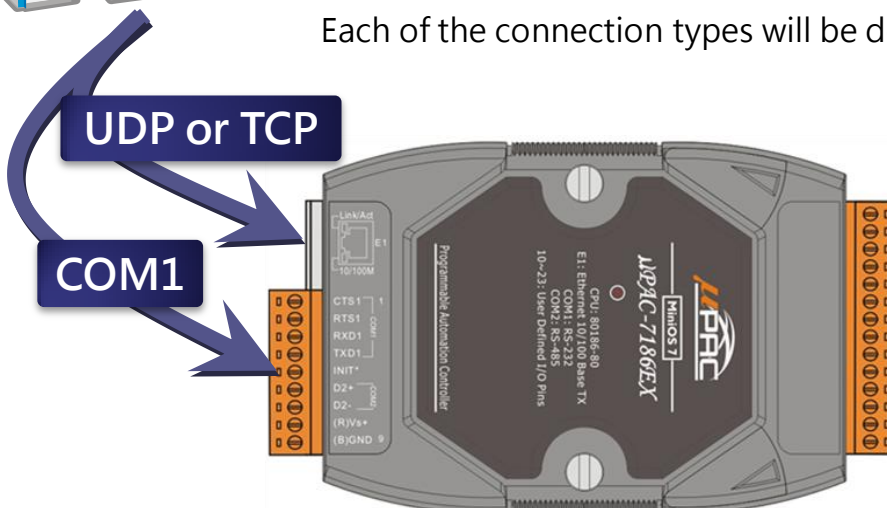
### 2.3.1. Establishing a connection between Host PC and the $\mu$ PAC-7186EX

Connect Host PC to the  $\mu$ PAC-7186EX with the following two connection types:



1. COM1 connection
2. UDP connection
3. TCP connection

Each of the connection types will be described in detail later.

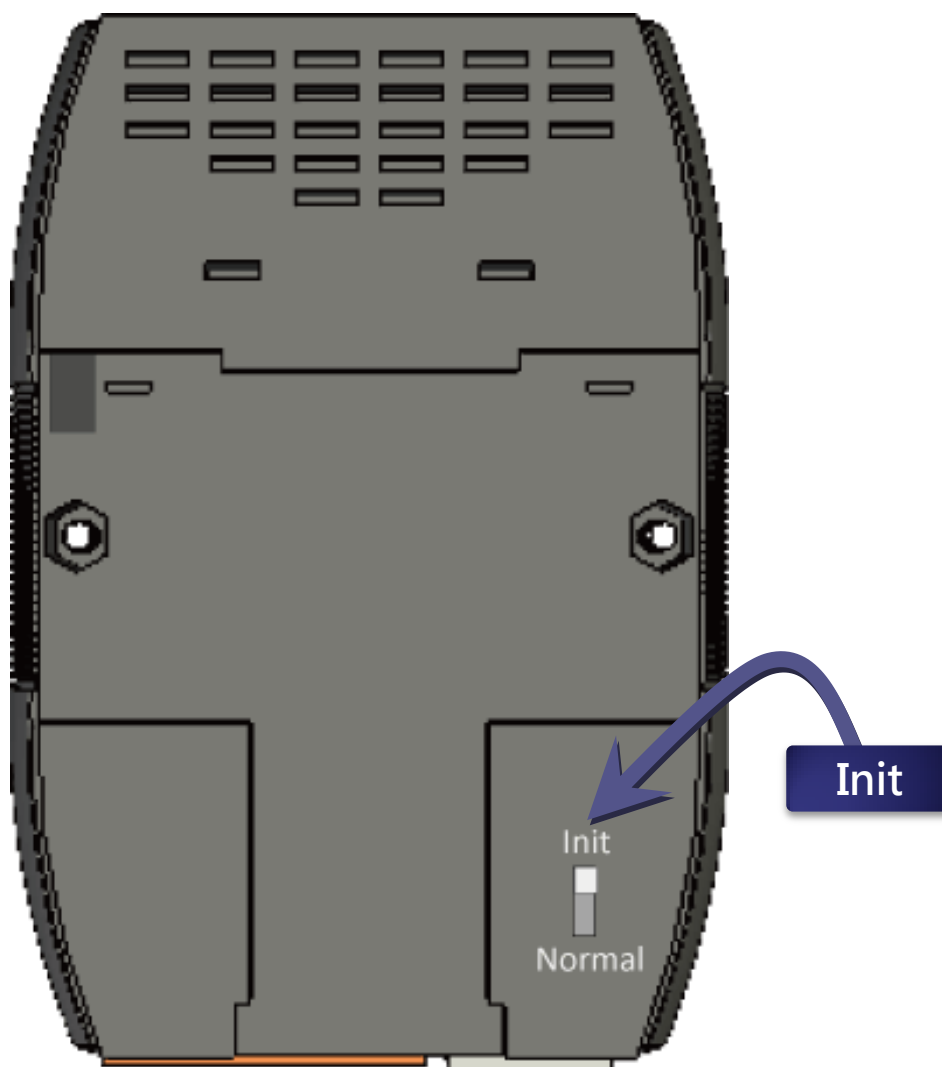


### 2.3.1.1. Steps to use a COM1 connection

---

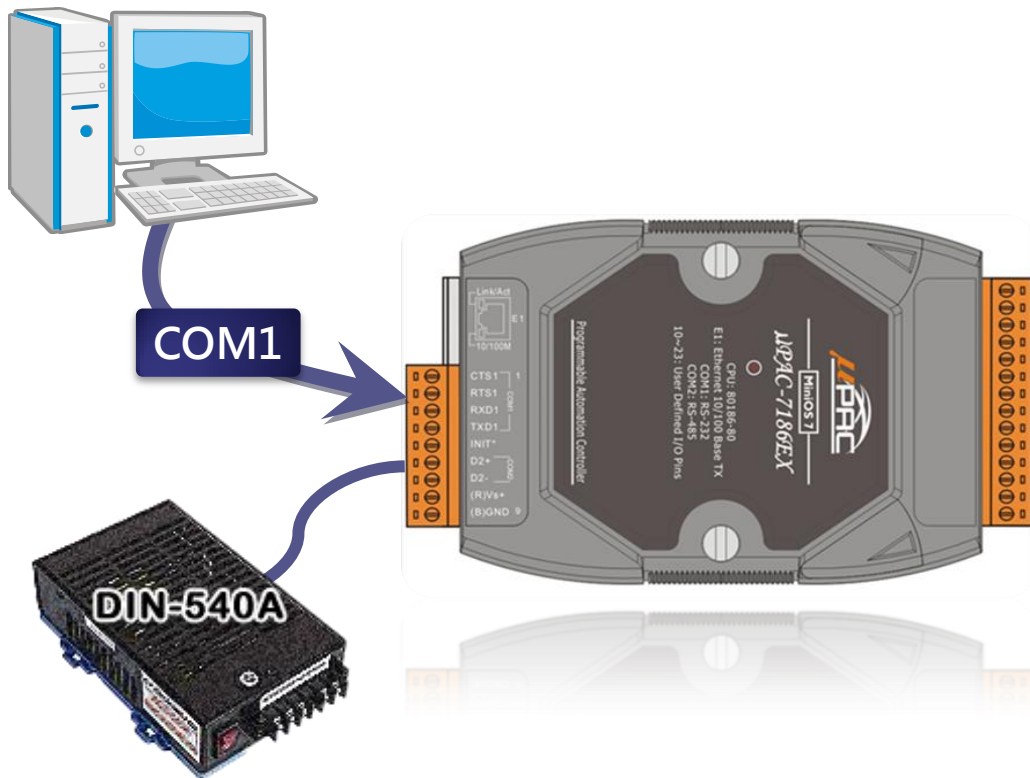
To connect to the host PC using a COM1 connection, please follow the instructions below.

Step 1: Turn the switch to "Init" position





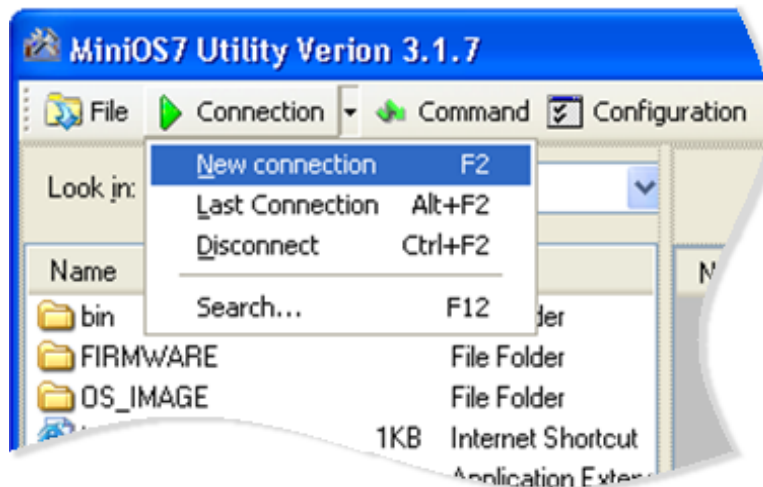
Step 2: Connect the  $\mu$ PAC-7186EX to the host PC using a COM1 connection



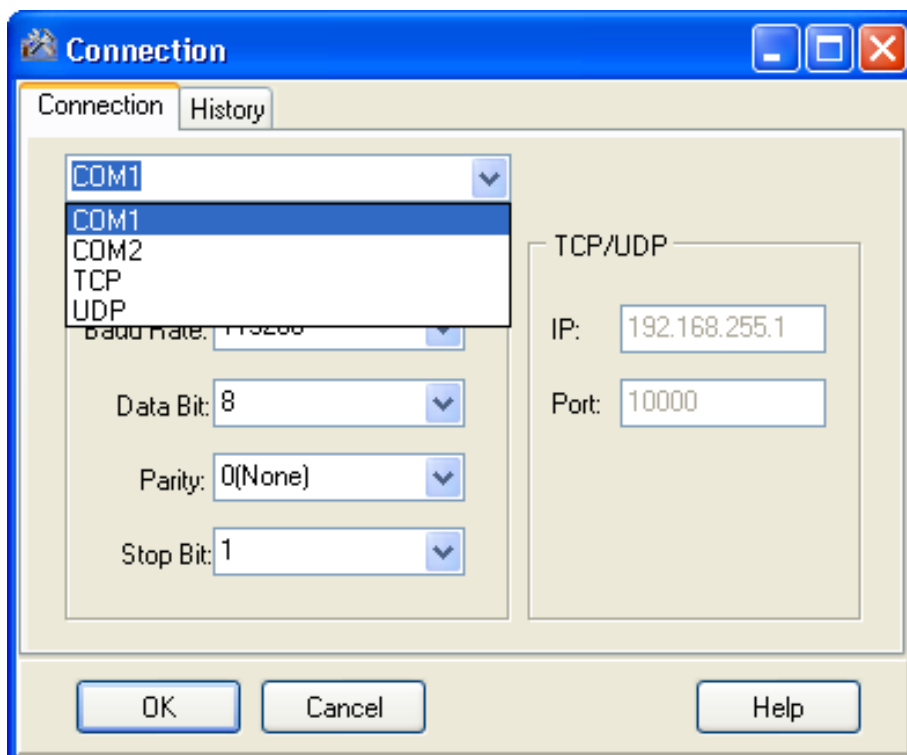
Step 3: Run the MiniOS7 Utility



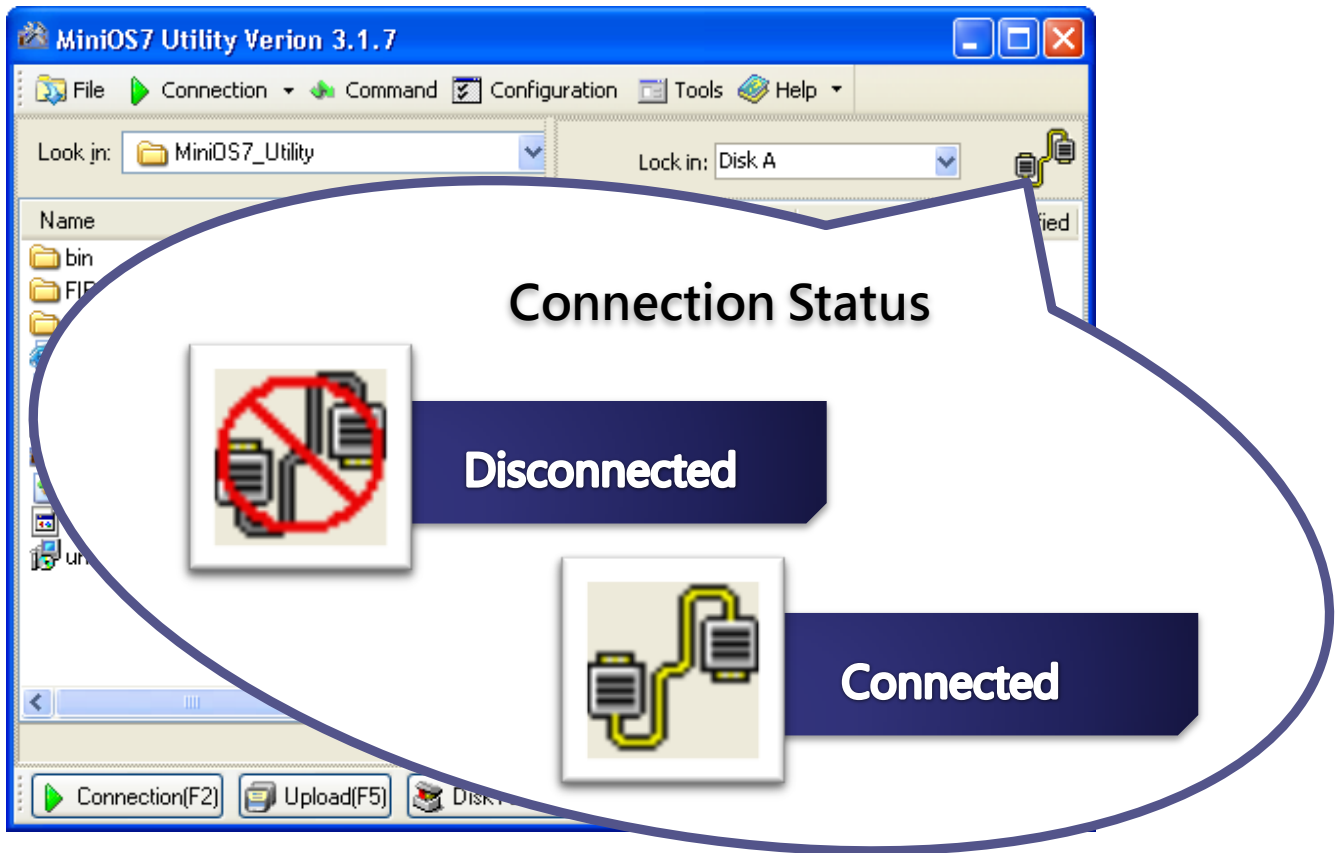
Step 4: On the "Connection" menu, click "New connection" function



Step 5: On the "Connection" tab of the "Connection" dialog box, select "COM1" from the drop down list, and then click "OK" button



Step 6: The connection has already established

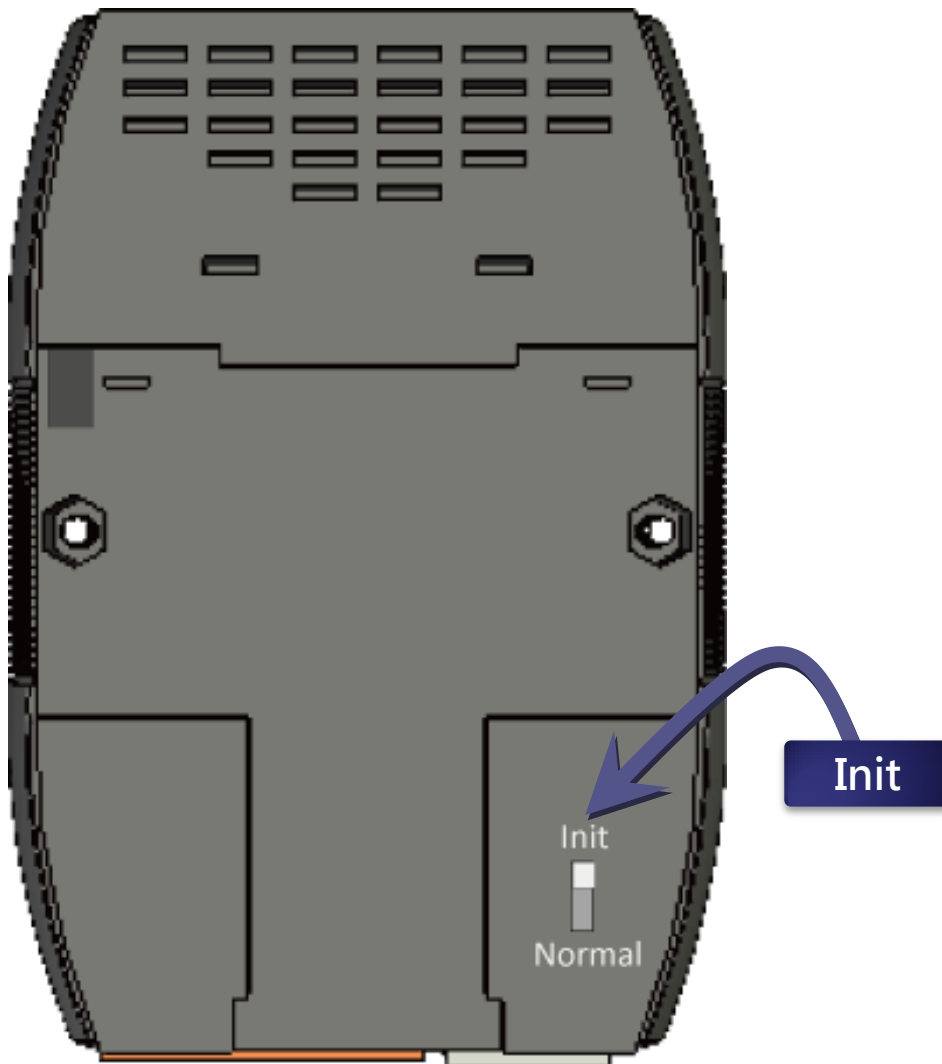


### 2.3.1.2. Steps to use a UDP connection

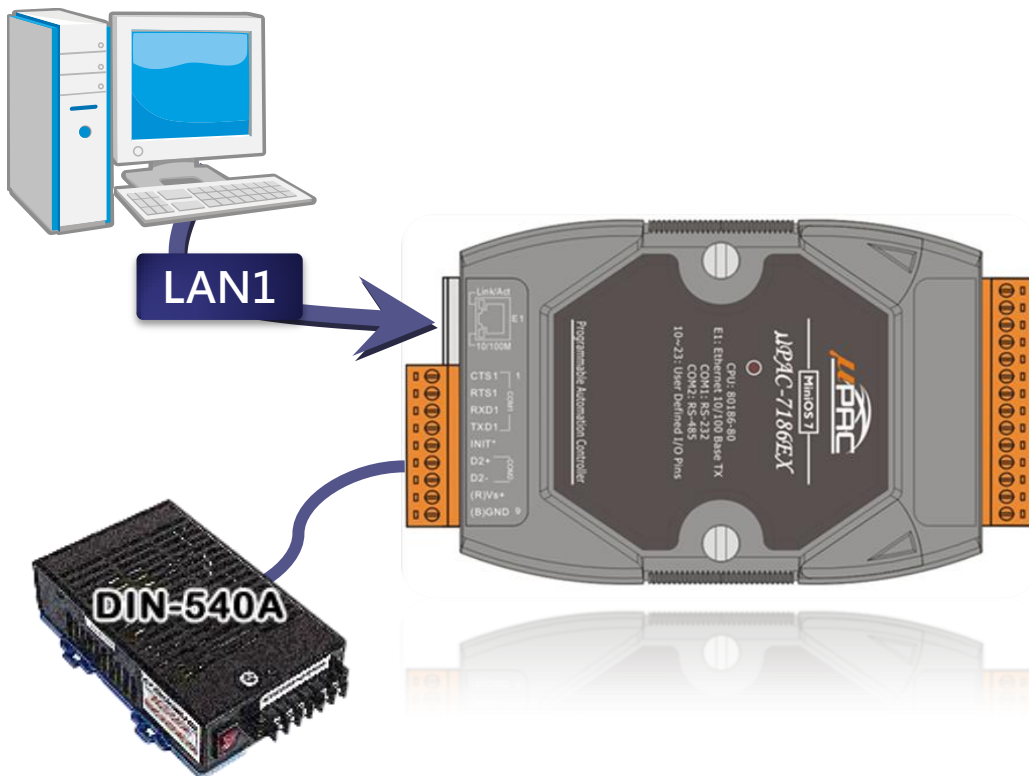
---

To connect to the host PC using a UDP connection, please follow the instructions below.

Step 1: Turn the switch to "Init" position



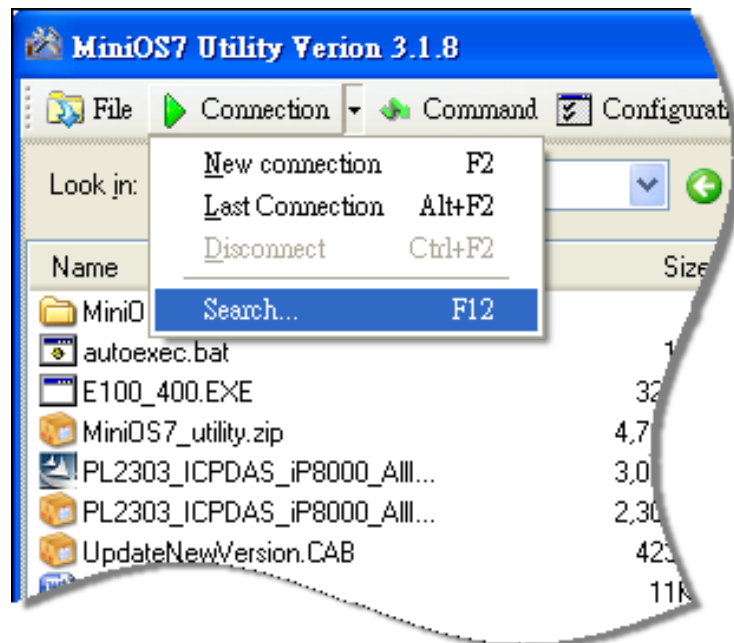
Step 2: Connect the  $\mu$ PAC-7186EX to the host PC using a LAN1 connection



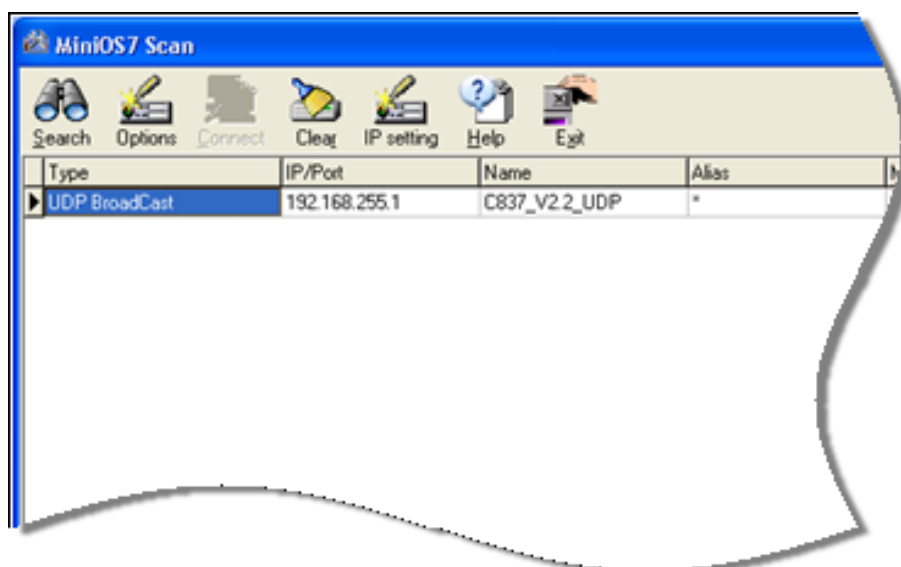
Step 3: Run the MiniOS7 Utility



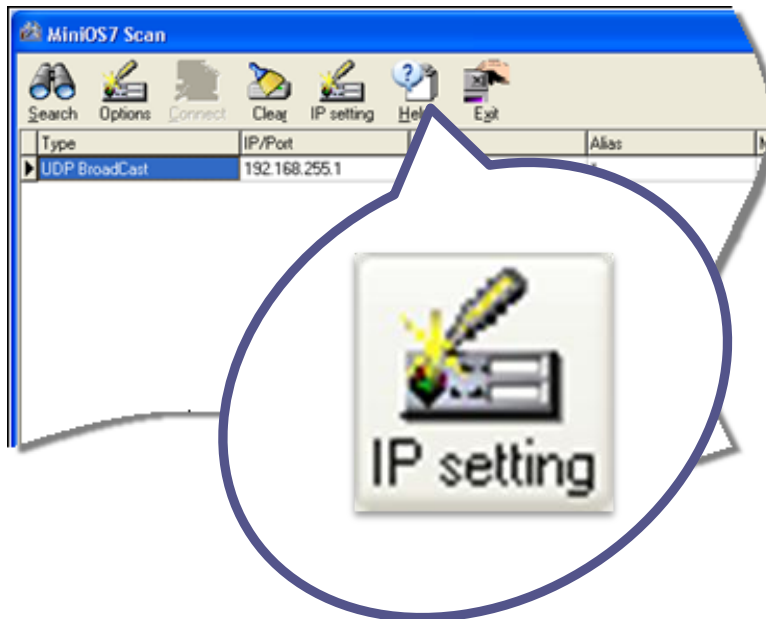
Step 4: On the "Connection" menu, click "Search" function



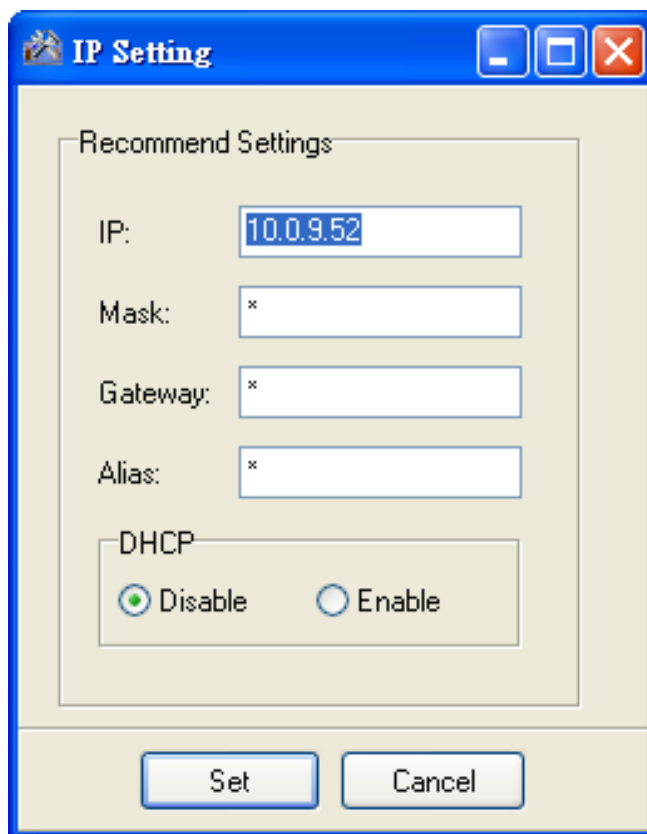
Step 5: On the "MiniOS7 Scan" dialog box, select "192.168.255.1" item from the list



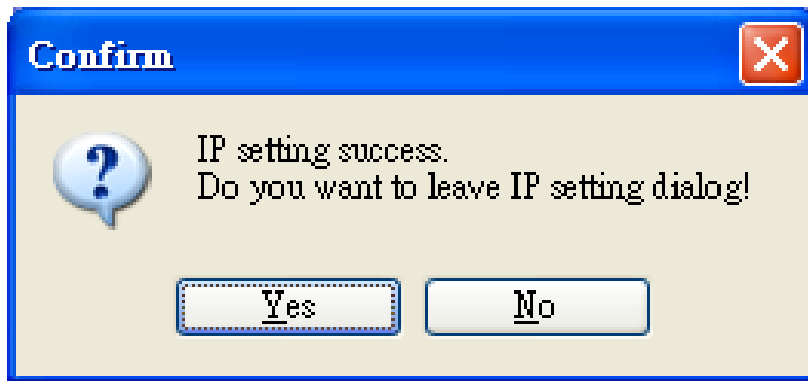
Step 6: On the toolbar, click "IP setting" button



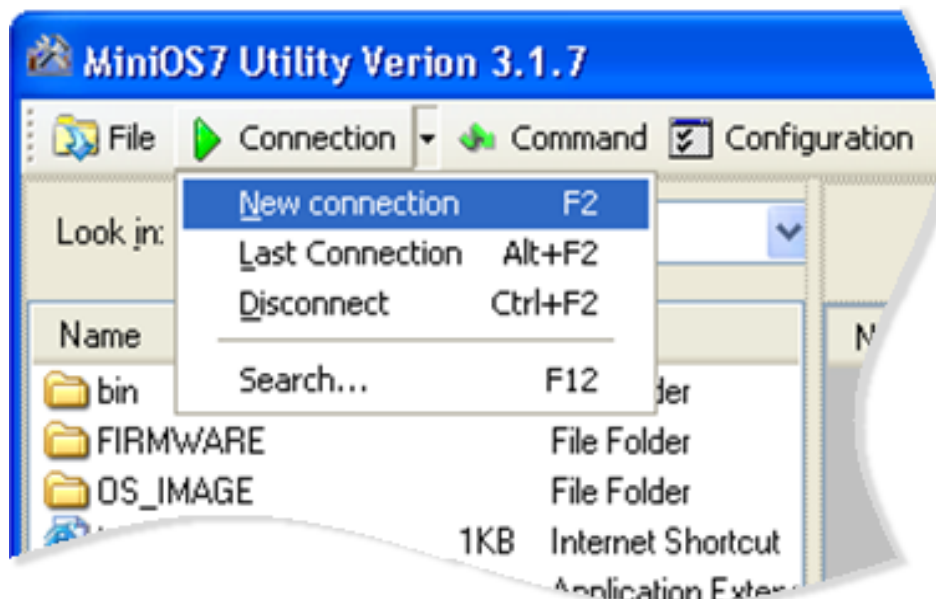
Step 7: On the "IP Setting" dialog, configure the "IP" settings and then click the "Set" button



Step 8: On the "Confirm" dialog box, click "Yes" button

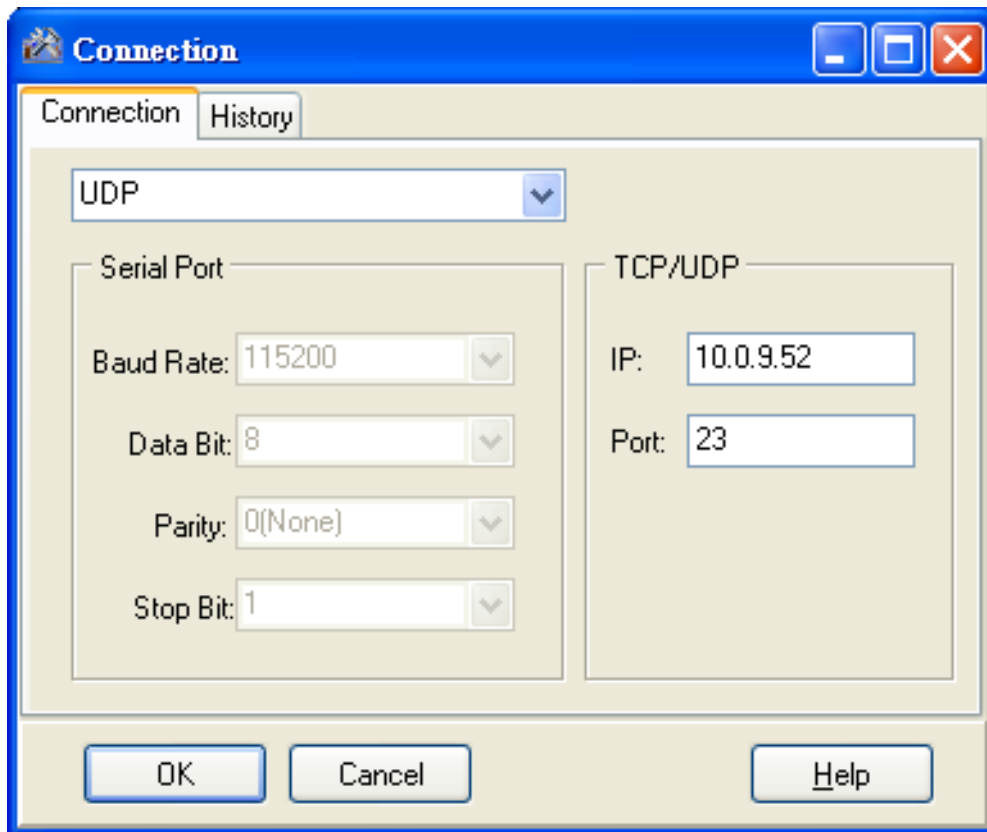


Step 9: On the "Connection" menu, click "New connection" function

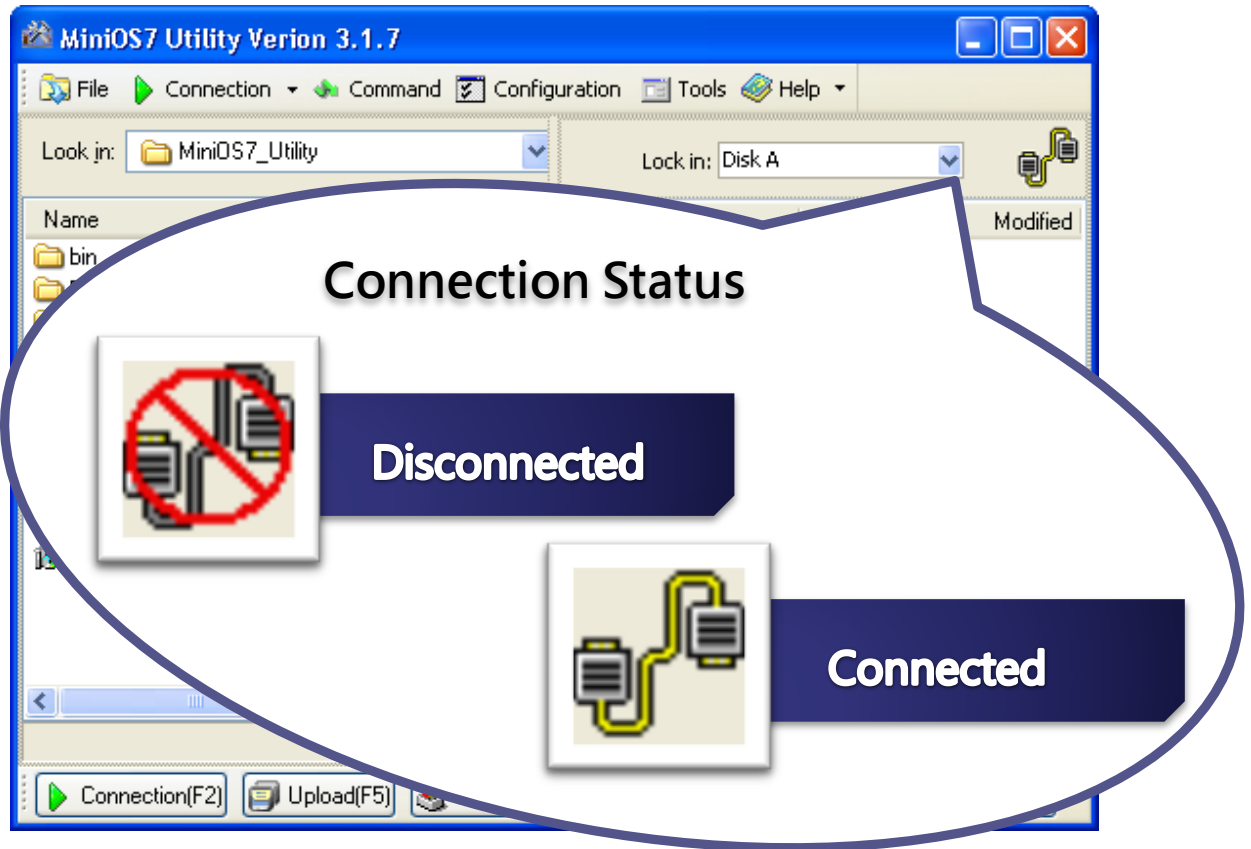




Step 10: On the "Connection" tab of the "Connection" dialog box, select "UDP" from the drop down list , type the IP address which you are assigned, and then click "OK" button



Step 11: The connection has already established

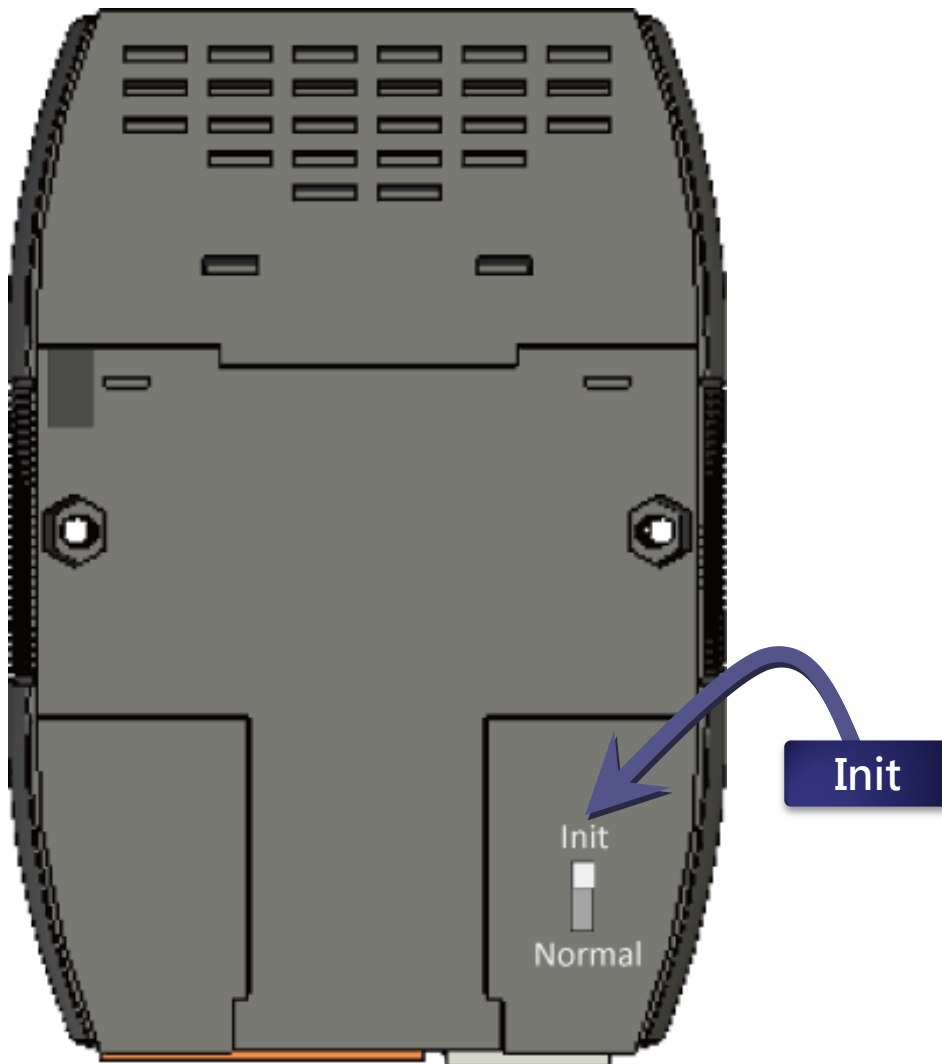


### 2.3.1.3. Steps to use a TCP connection

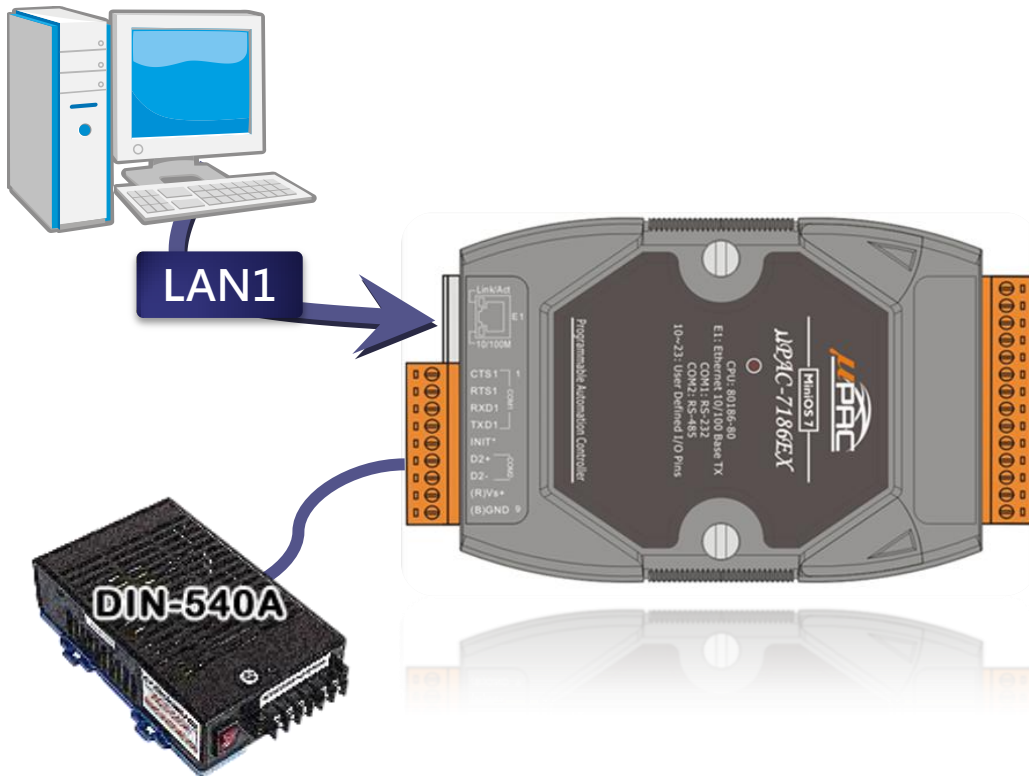
---

To connect to the host PC using a TCP connection, please follow the instructions below.

Step 1: Turn the switch to "Init" position



Step 2: Connect the  $\mu$ PAC-7186EX to the host PC using a LAN1 connection



Step 3: Run the MiniOS7 Utility



#### Step 4: Run the VxComm driver

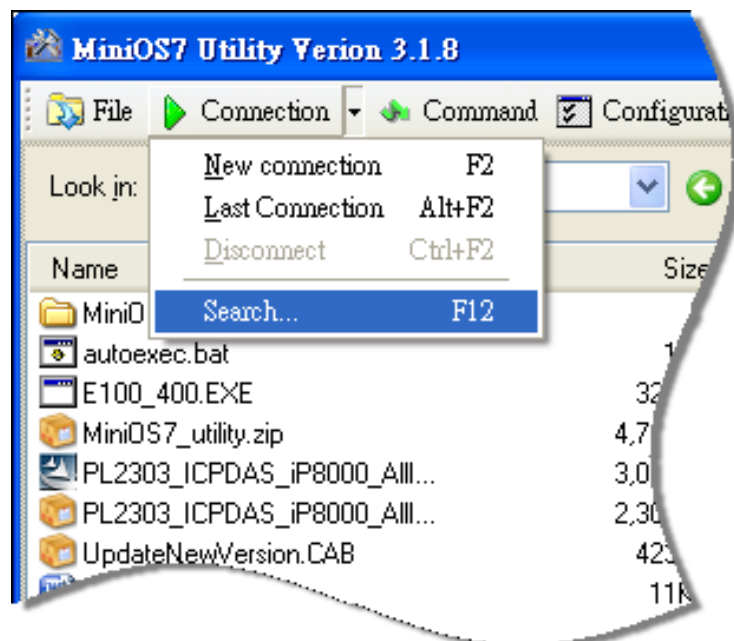
The VxComm driver is located at:

CD:\Napdos\7186e\firmware\vxcomm\server(7186e)\7186ex\

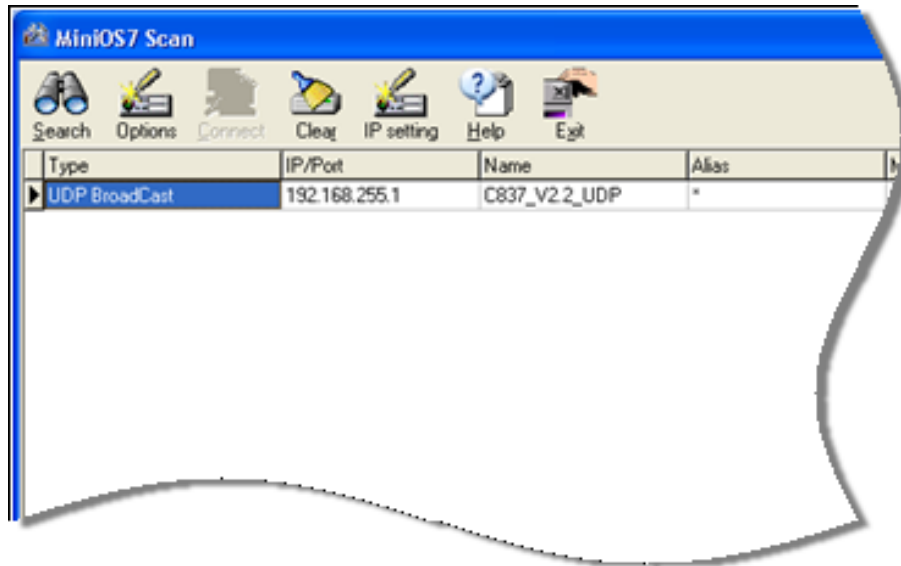
[ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/firmware/vxcomm/server\(7186e\)/7186ex/](ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/firmware/vxcomm/server(7186e)/7186ex/)



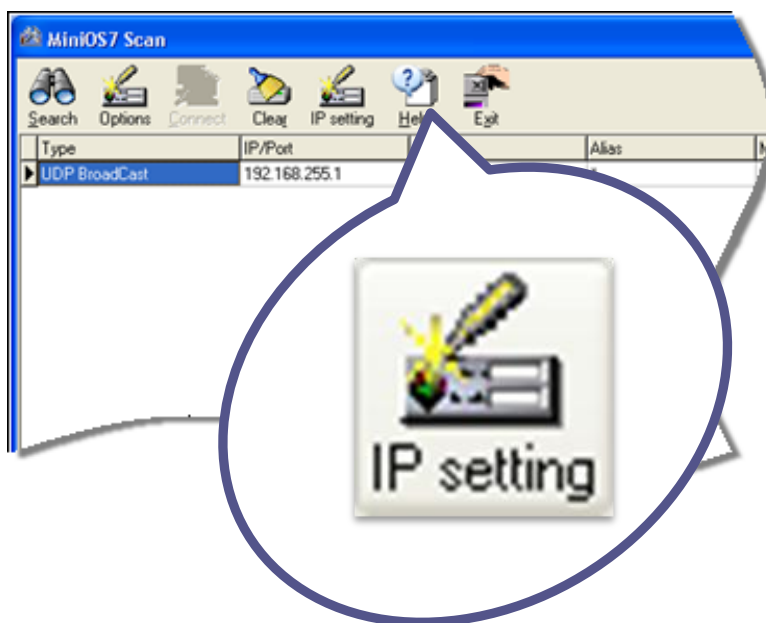
#### Step 5: On the "Connection" menu, click "Search" function



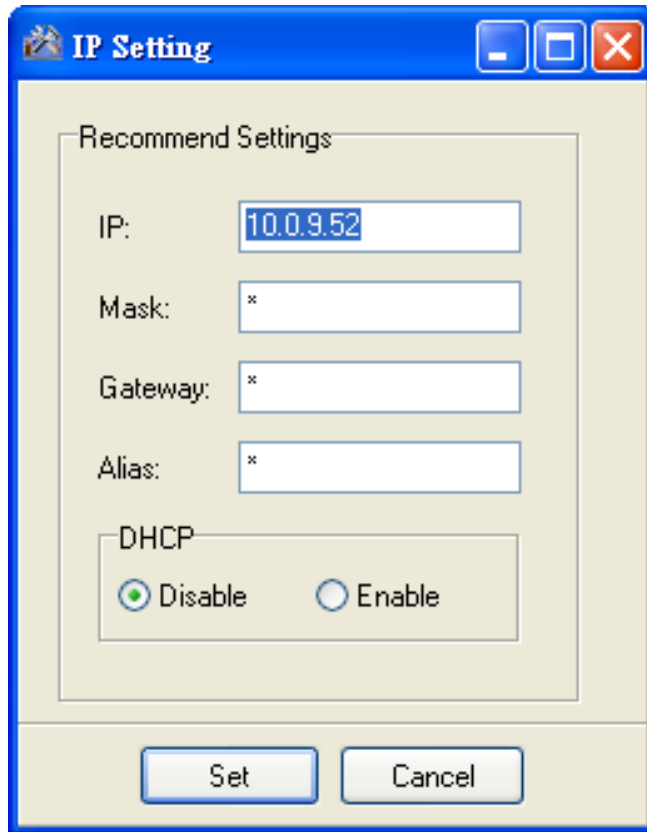
Step 6: On the “MiniOS7 Scan” dialog box, select “192.168.255.1” item from the list



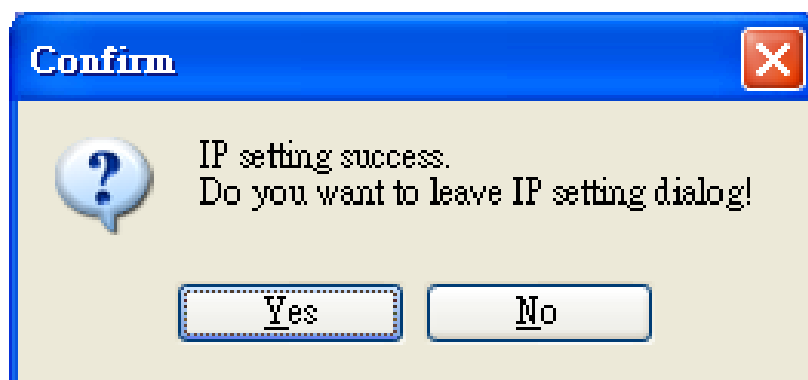
Step 7: On the toolbar, click “IP setting” button



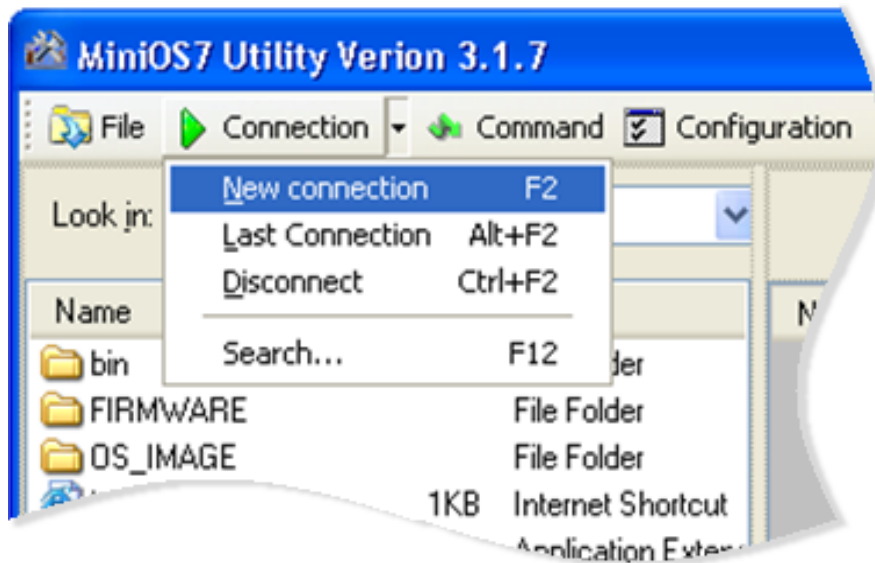
Step 8: On the "IP Setting" dialog, configure the "IP" settings and then click the "Set" button



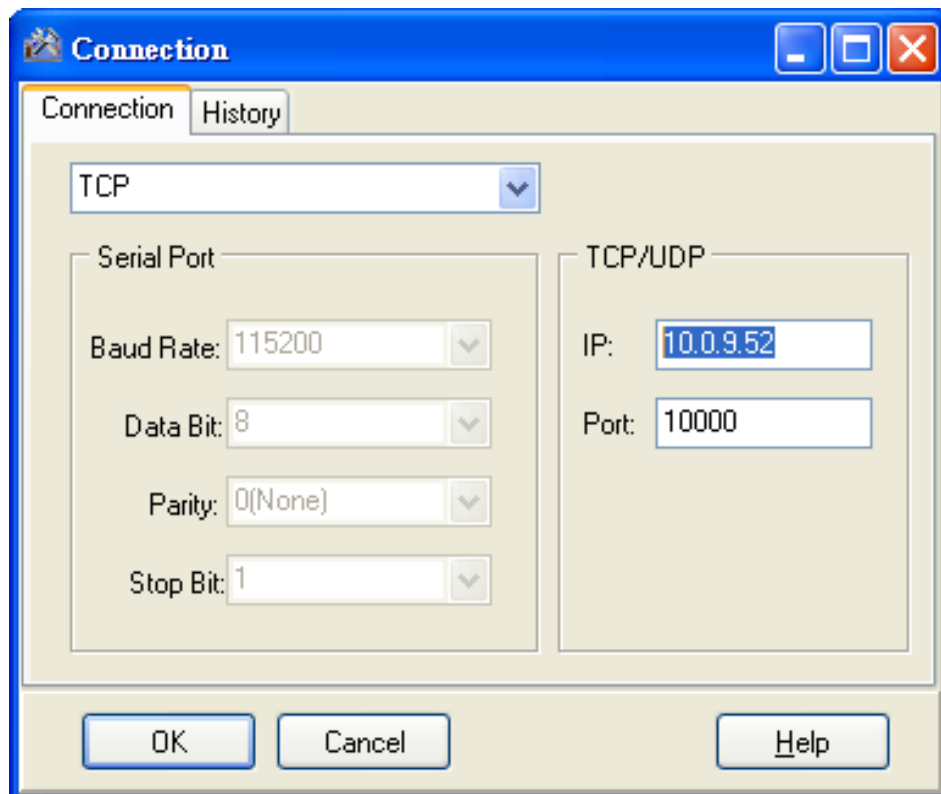
Step 9: On the "Confirm" dialog box, click "Yes" button



Step 10: On the "Connection" menu, click "New connection" function

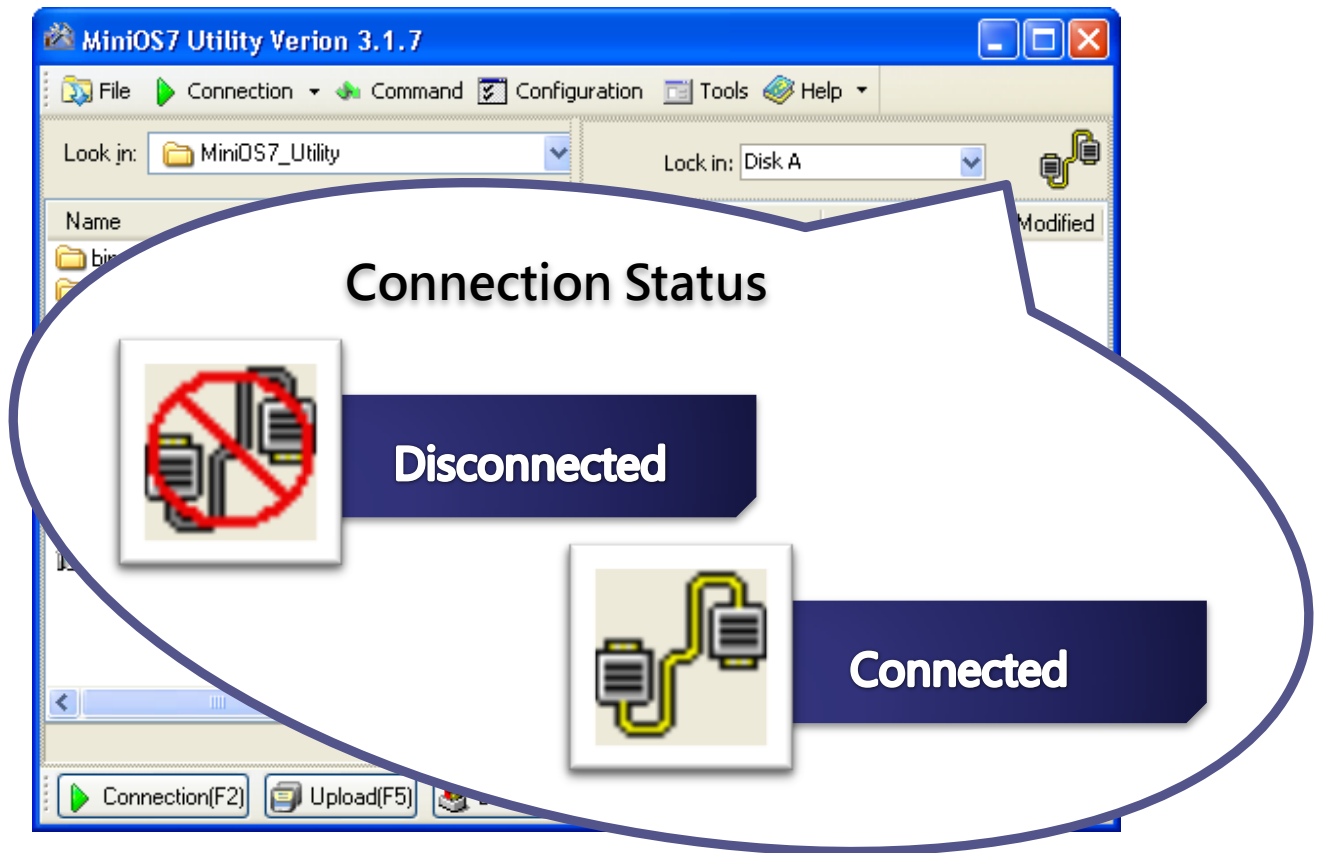


Step 11: On the "Connection" tab of the "Connection" dialog box, select "TCP" from the drop down list, type the IP address which you are assigned, and then click "OK" button



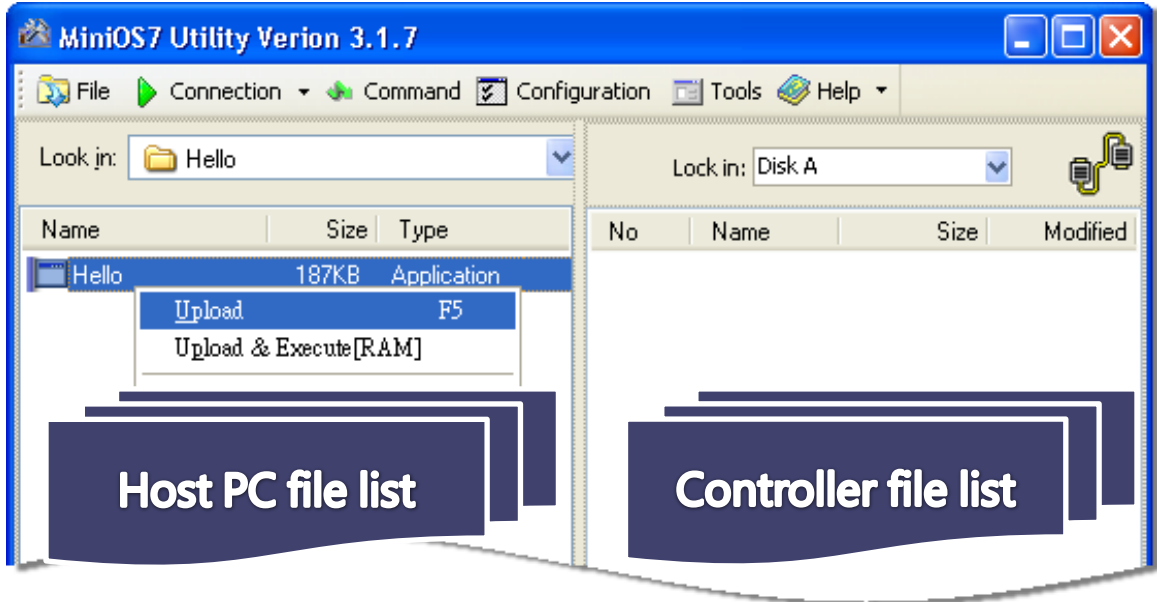


Step 12: The connection has already established

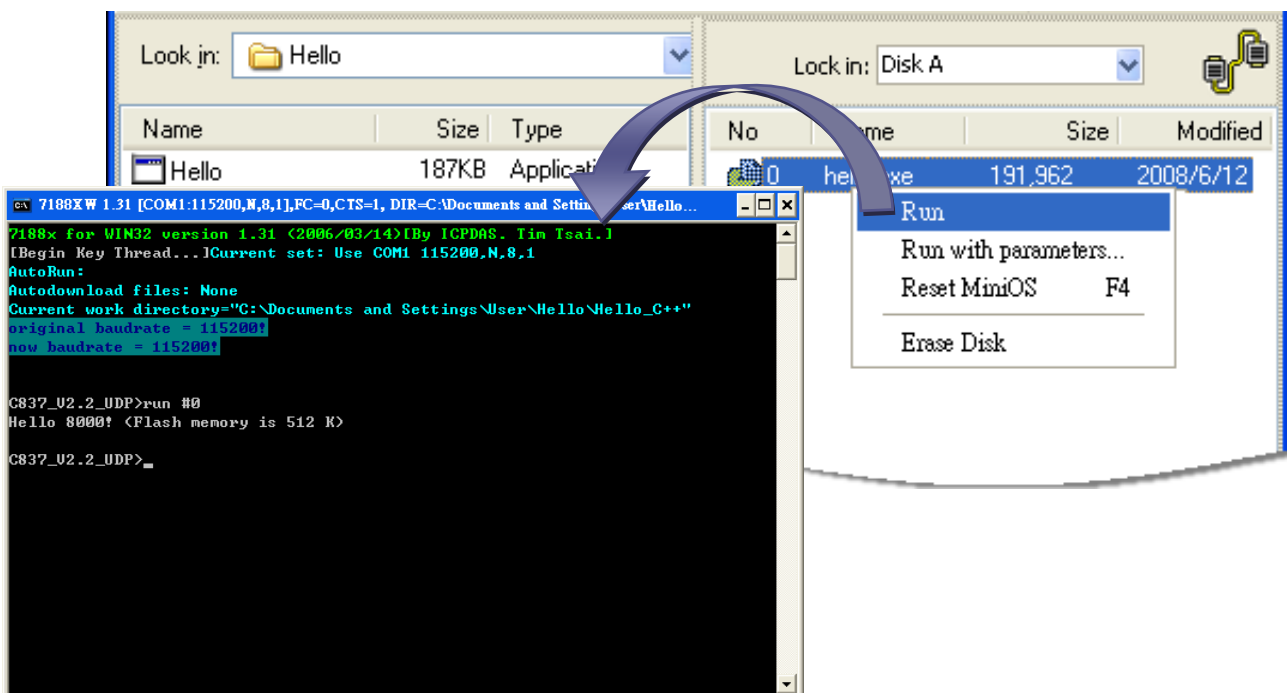


### 2.3.2. Uploading and executing programs on $\mu$ PAC-7186EX

Step 1: On the host pc file list, Right click on the file name that you wish to download and then select the "Upload" option



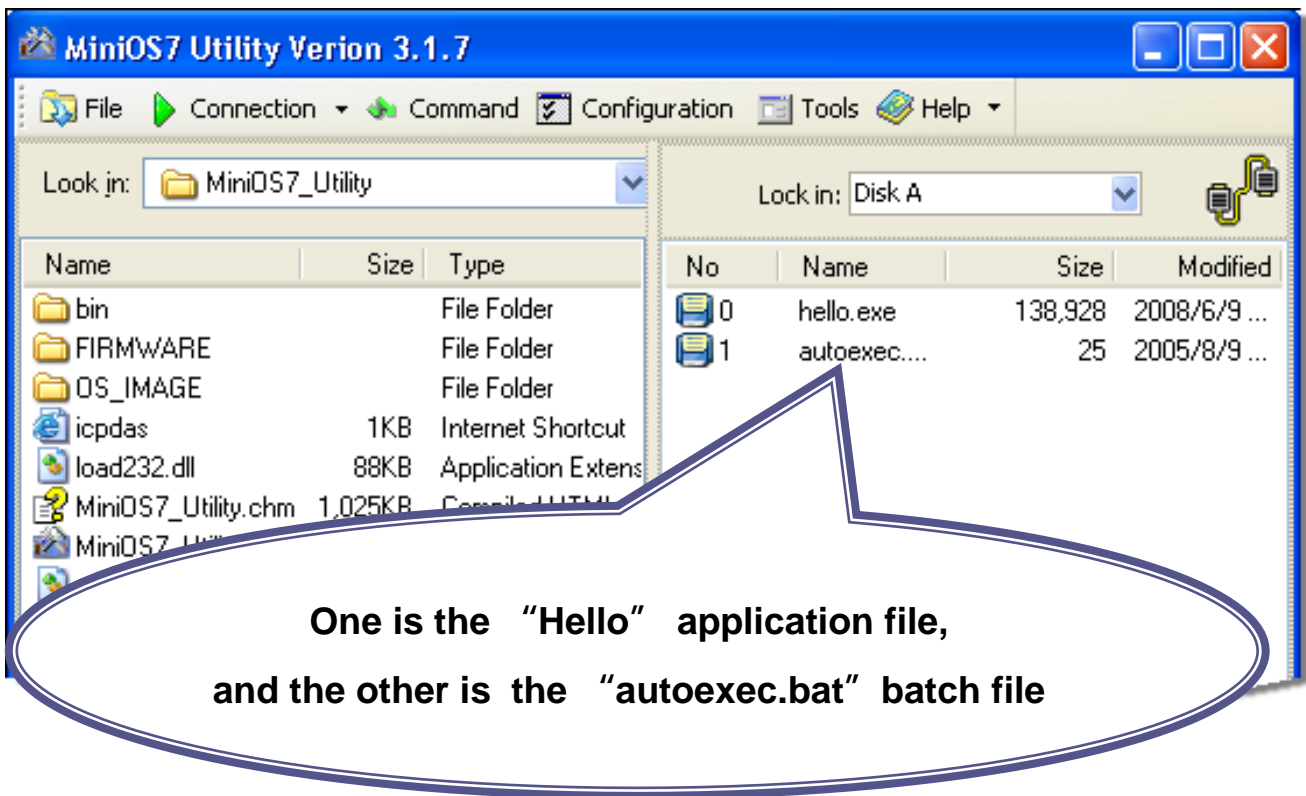
Step 2: On the controller file list, Right click on the file name that you wish to execute and then select the "Run" option



### 2.3.3. Making programs start automatically

After download programs on the  $\mu$ PAC-7186EX, if you need programs to start automatically after the  $\mu$ PAC-7186EX start-up, it is easy to achieve it, to create a batch file called autoexec.bat and then upload it on the  $\mu$ PAC-7186EX, the program will start automatically in the next start-up.

For example, to make the program "hello" run on start-up.



## 2.4. MiniOS7 Utility for updating OS image

---

ICP DAS will continue to add additional features to MiniOS7 in the future, we advise you periodically check the ICP DAS web site for the latest update to MiniOS7.

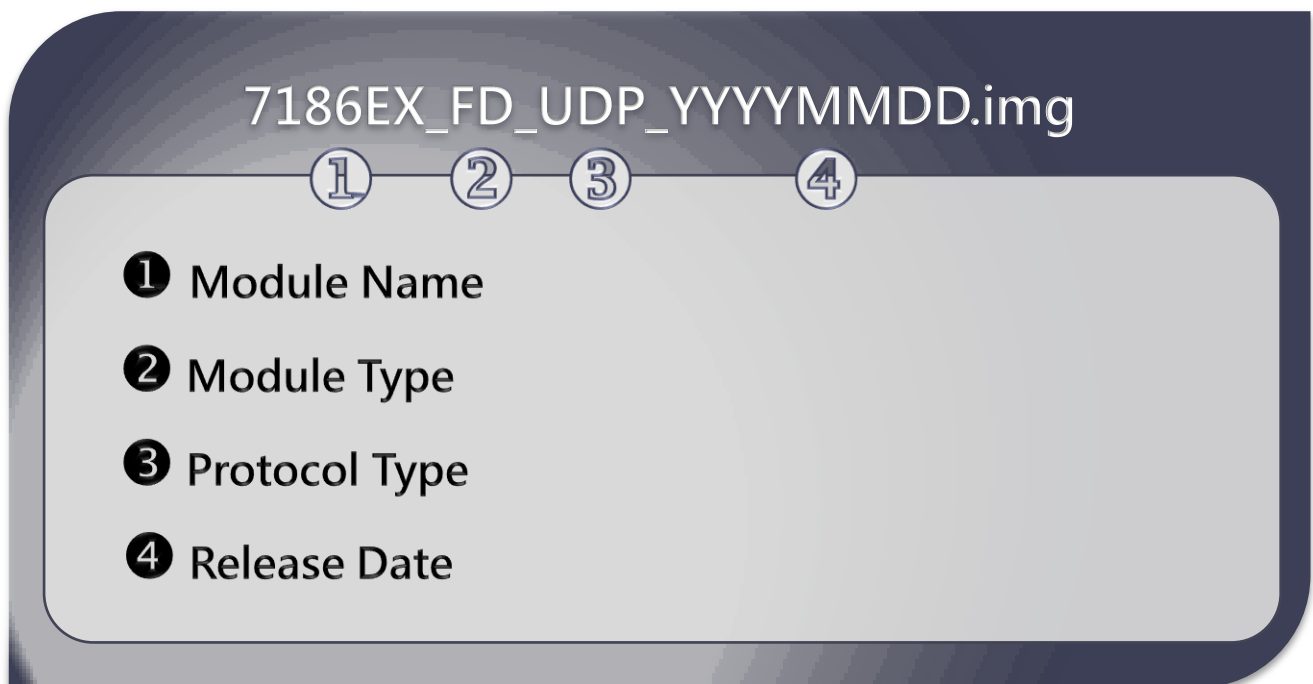
### Step 1: Get the latest version of the MiniOS7 OS image

The latest version of the MiniOS7 OS image can be obtain from:

CD:\NAPDOS\7186e\OS\_Image

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/OS\\_Image/](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/OS_Image/)

#### ➤ For 7186EX-FD series



- For 7186EX and 7186EX-SM series

**7186EX\_UDP\_HR\_YYYYMMDD.img**

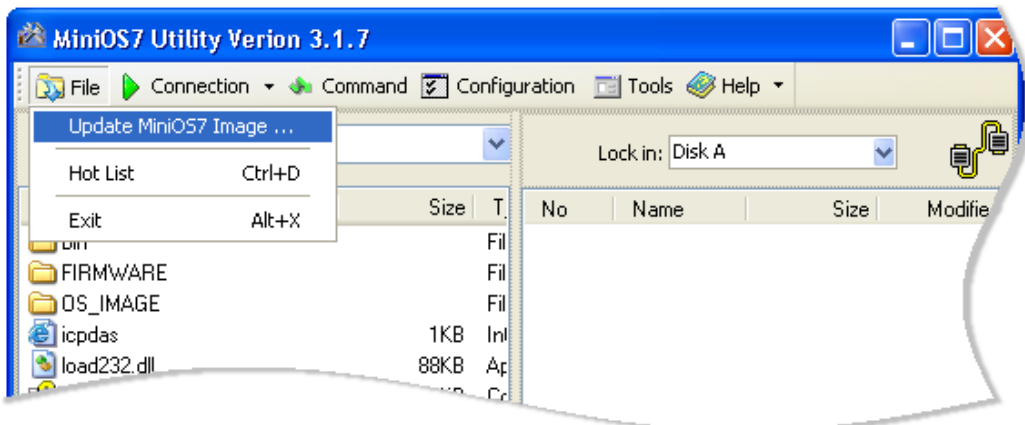
①      ②      ③      ④

- ① Module Name
- ② Protocol Type
- ③ Module Type
- ④ Release Date

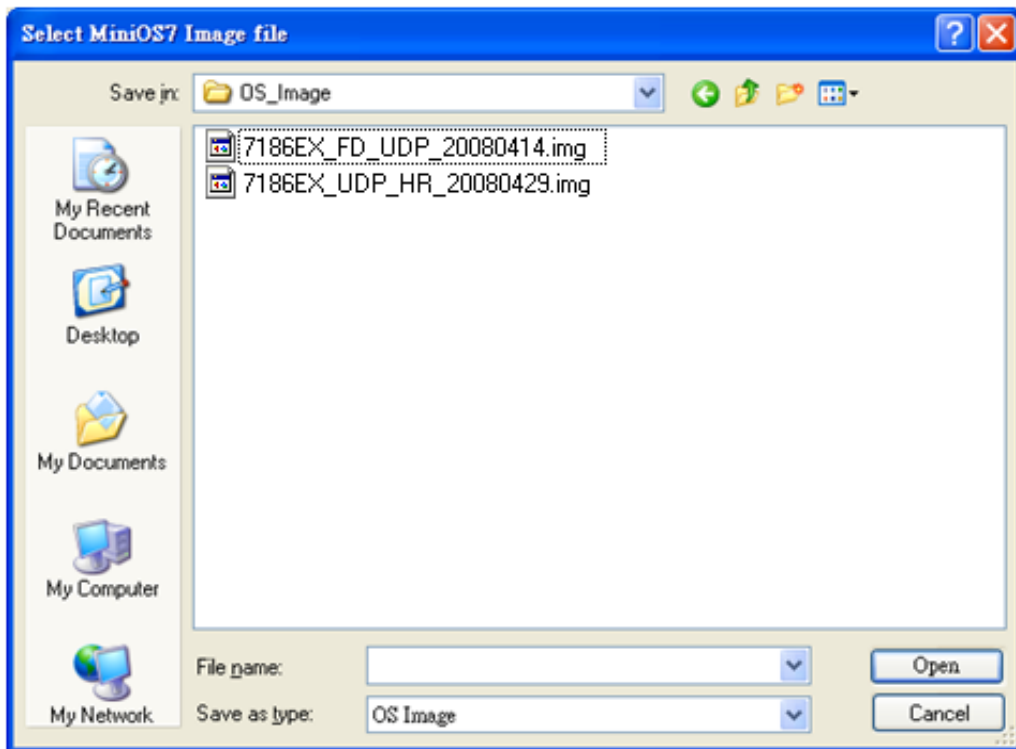
**Step 2: Establish a connection**

For more detailed information about this process, please refer to section “2.3.1. Establishing a connection” .

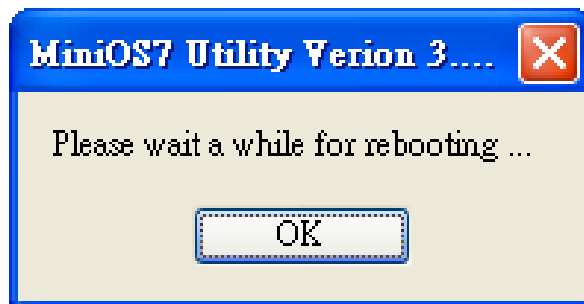
**Step 3: Click on the “Update MiniOS7 Image ...” from the “File” menu**



Step 4: Select the latest version of the MiniOS7 OS image



Step 5: Click on the "Update MiniOS7 Image ..." from the "File" menu



Step 6: Click on the "Info" button to check OS image version

## 3. Your First Program on $\mu$ PAC-7186EX

---

Before writing your first program, ensure that you have the necessary C/C++ compiler and the corresponding functions library on your system.

### 3.1. Setting up the compiler

---

The following compilers are available for  $\mu$ PAC-7186EX.

- Turbo C++ Version 1.01 (Freeware)
- Turbo C Version 2.01 (Freeware)
- Borland C++ Versions 3.1 - 5.2.x
- MSC
- MSVC ++



ICP DAS suggests that the Borland C++ version compiler is used as the libraries provided on the companion CD have been created using this compiler.

Special attention should be paid to the following items before using the compiler to develop custom applications:

- Generate a standard DOS executable program

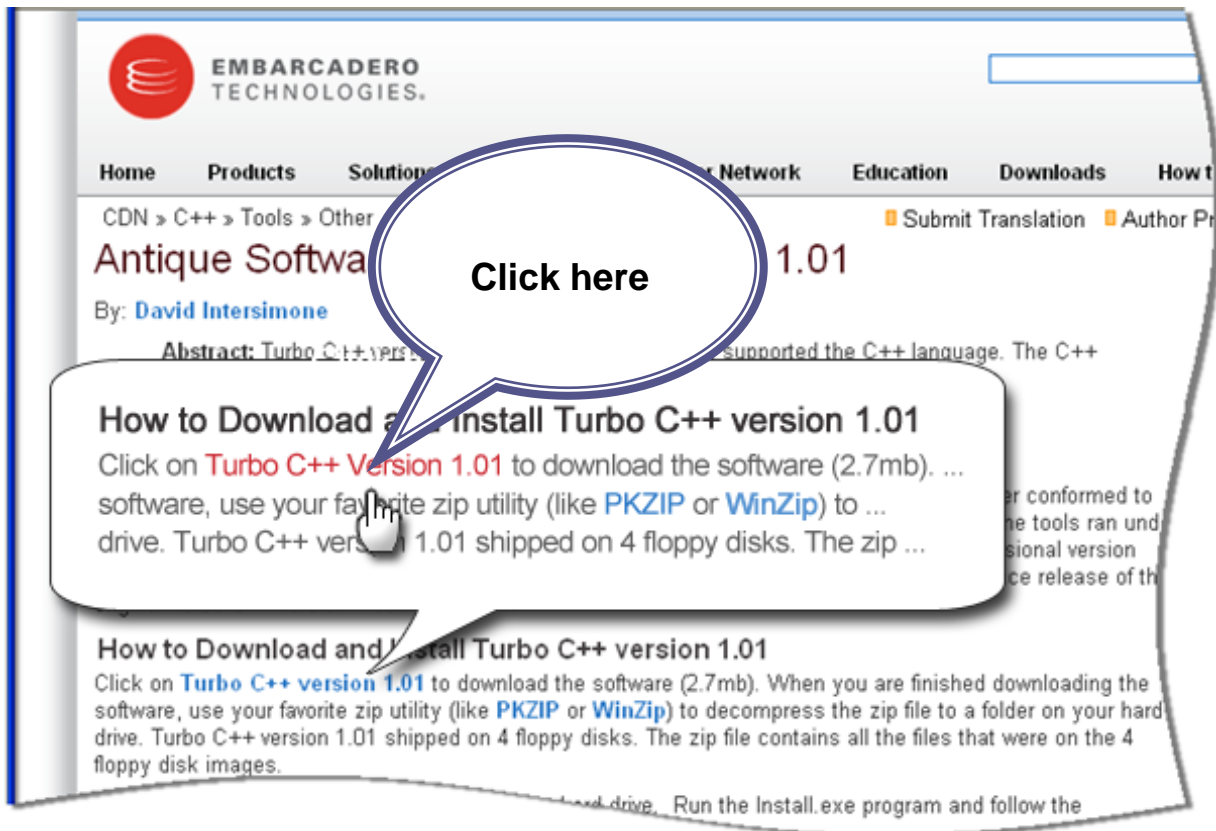
- Set the CPU option to 80188/80186
  - Set the floating point option to EMULATION if floating point computation is required. (Be sure not to choose 8087)
  - Cancel the Debug Information function as this helps to reduce program size. (MiniOS7 supports this feature.).
-



### 3.1.1.Installing the Compiler

If there is no compiler currently installed on your system, installation of the compiler should be the first step. The following section guides you to install Turbo C++ Version 1.01 on your system.

**Step 1: Go to the Borland web site and download Turbo C++ version 1.01**





Free versions of the Turbo C++ version 1.01 and Turbo version 2.01 Compilers can be downloaded from the Borland web site.

➤ Turbo C++ version 1.01

<http://dn.codegear.com/article/21751>

➤ Turbo C version 2.01

<http://dn.codegear.com/article/20841>

---

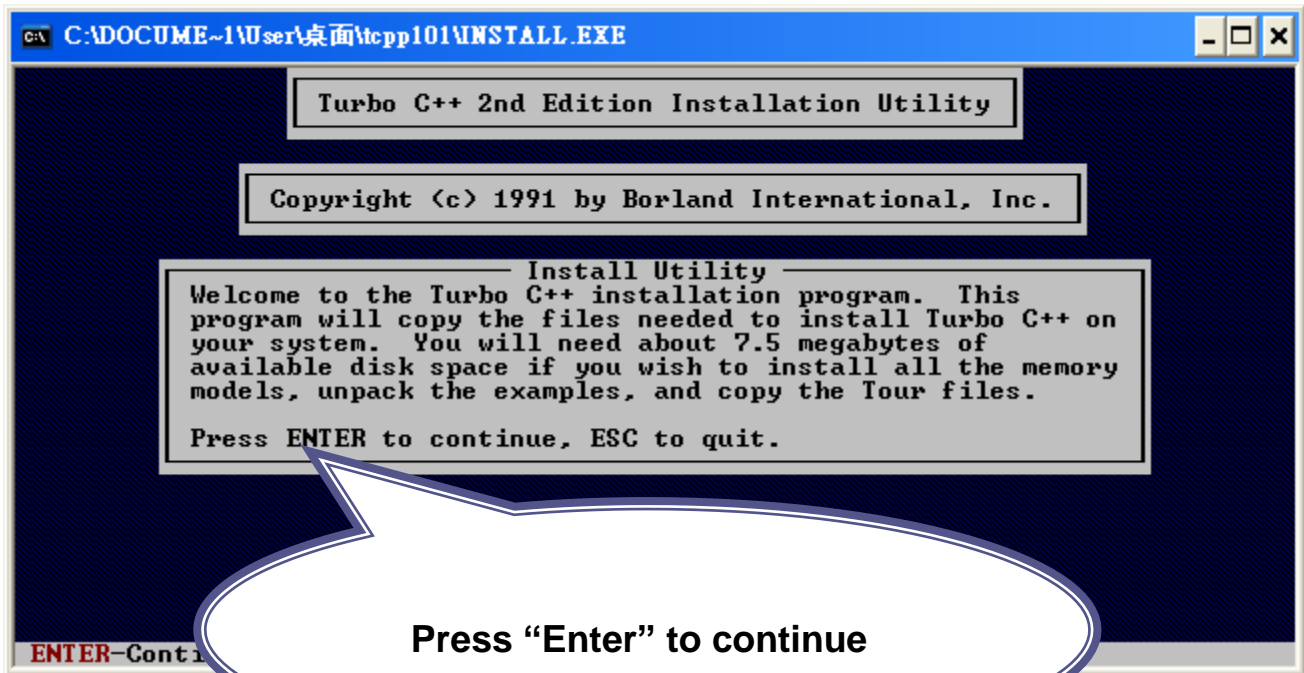
**Step 2: Unzip the downloaded zip file to the temporary folder**



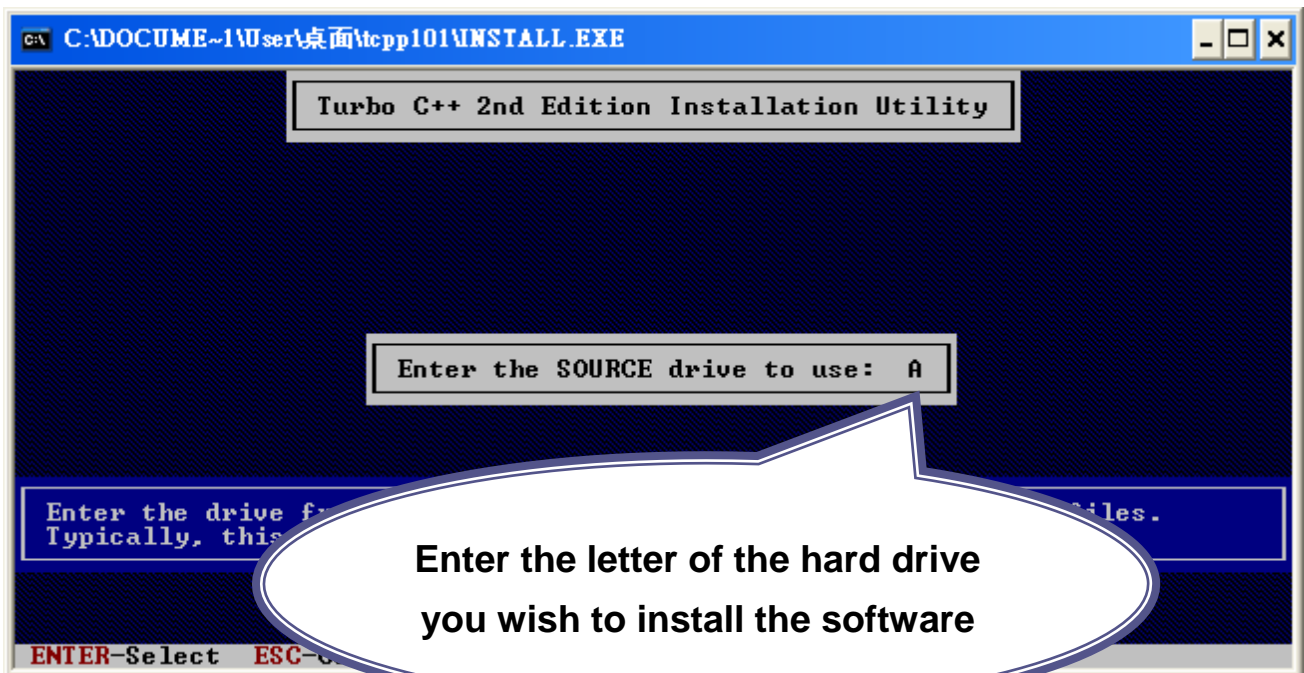
**Step 3: Double click the executable file to start setup wizard**



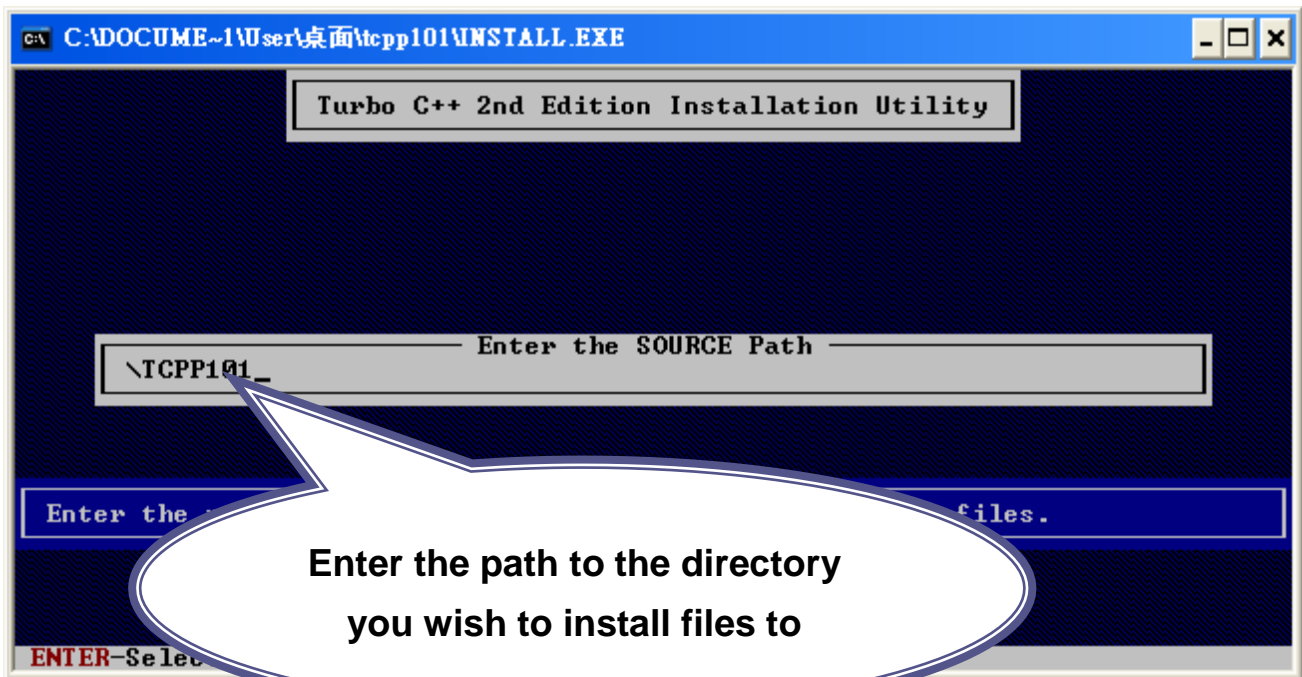
Step 4: Press "Enter" to continue



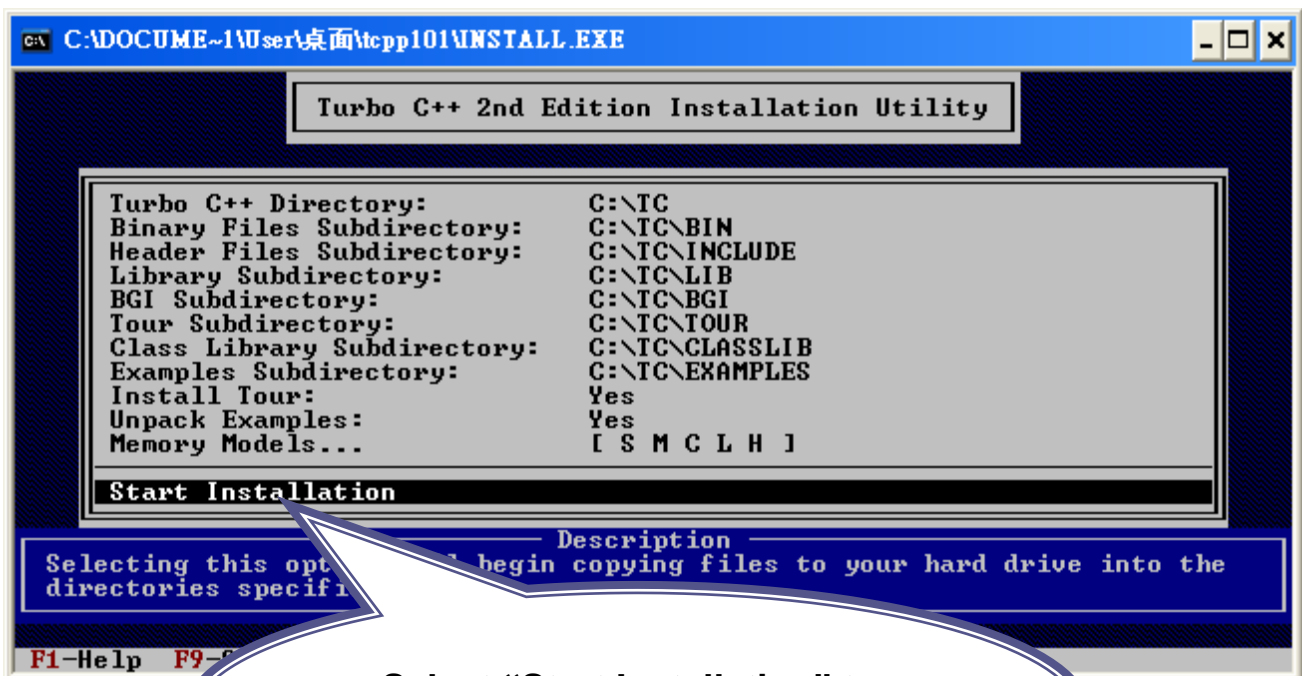
Step 5: Enter the letter of the hard drive you wish to install the software



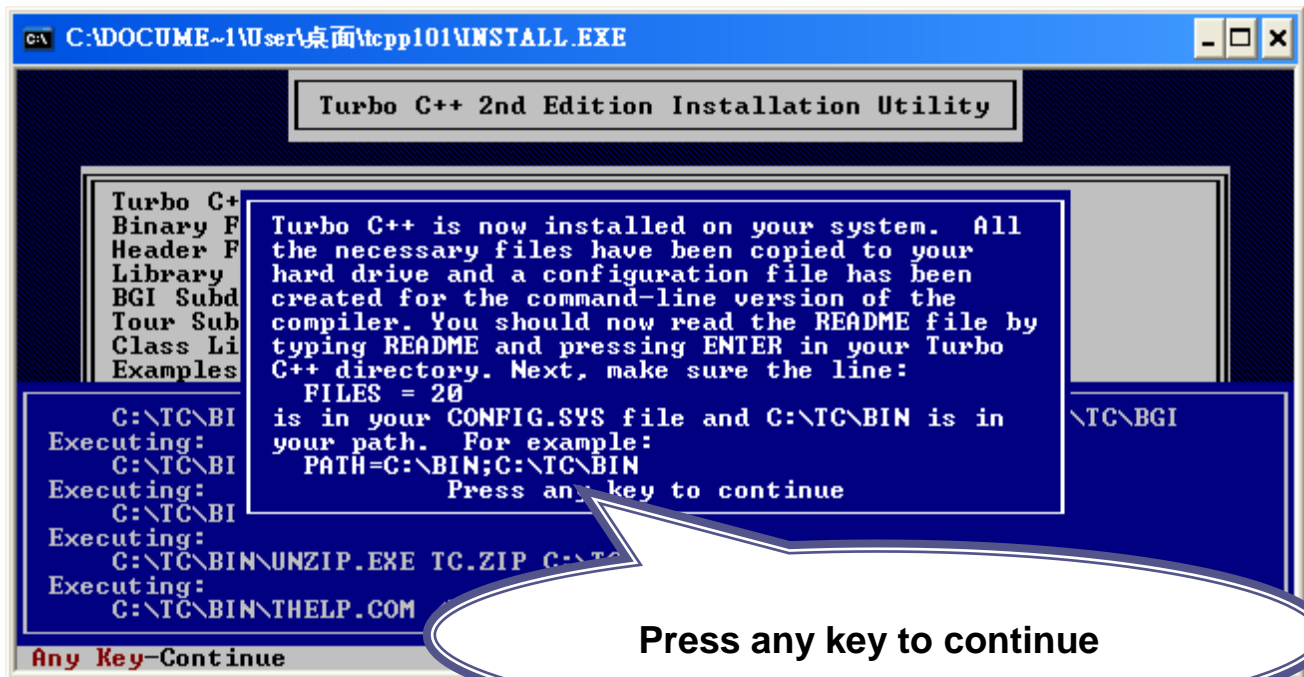
Step 6: Enter the path to the directory you wish to install files to



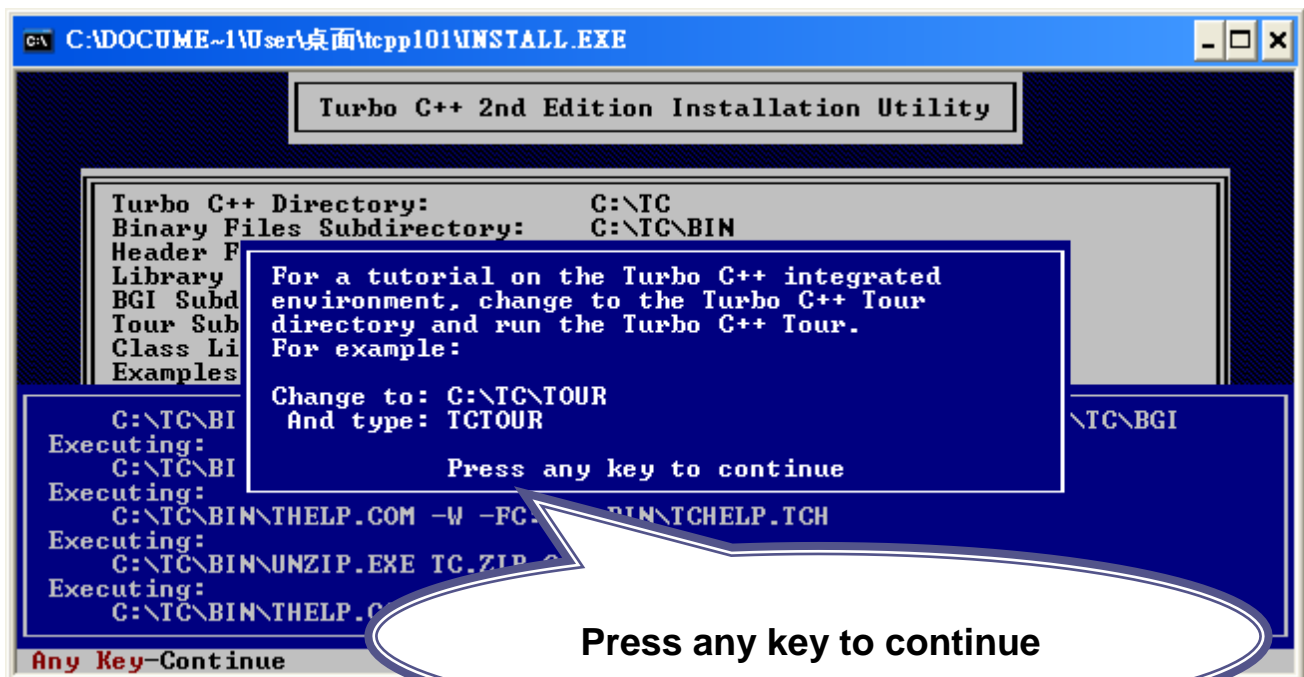
Step 7: Select "Start Installation" to begin the install process



Step 8: Press any key to continue



Step 9: Press any key to continue

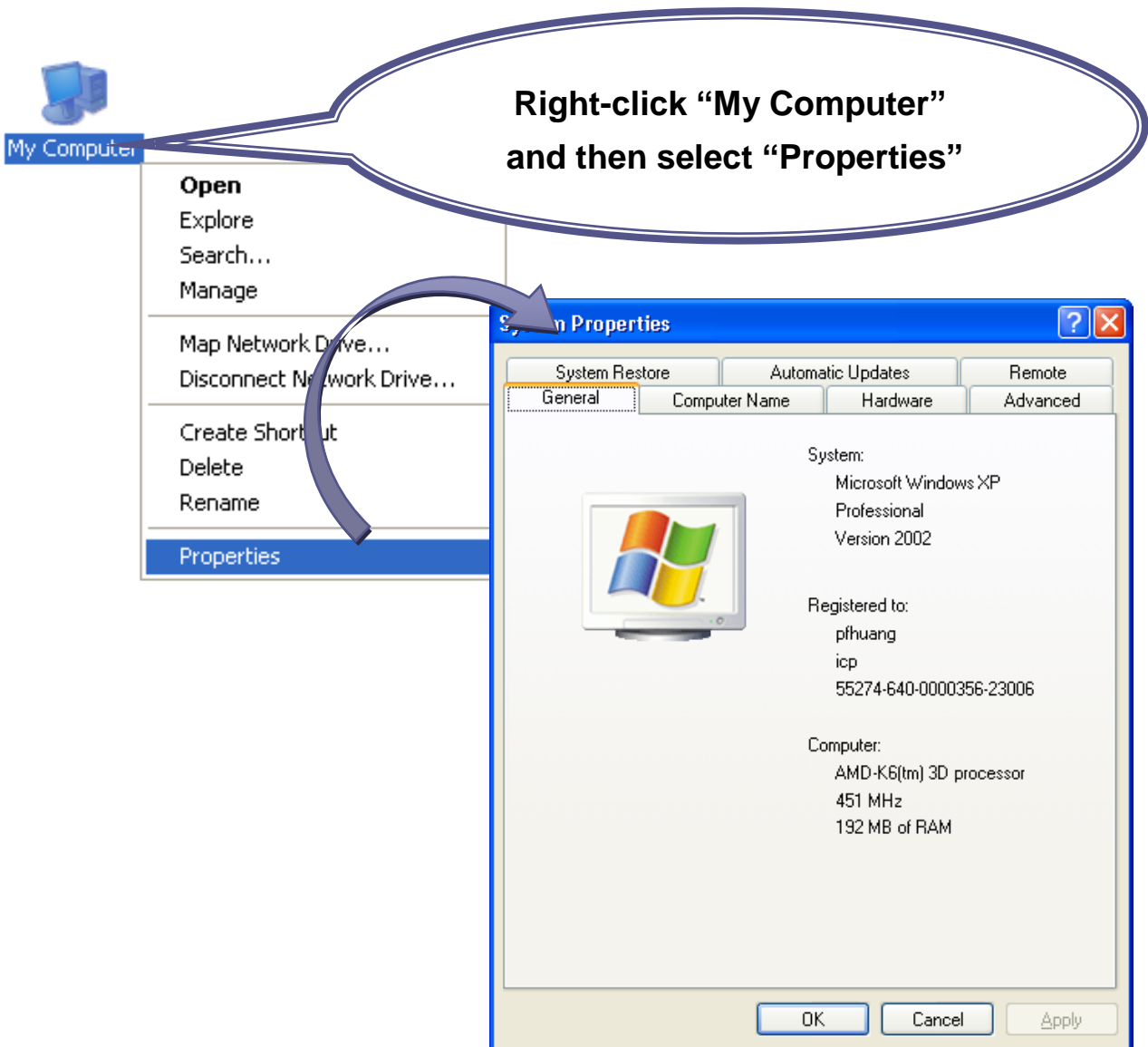


Step 10: Installation is complete

### 3.1.2. Setting up the environment variables

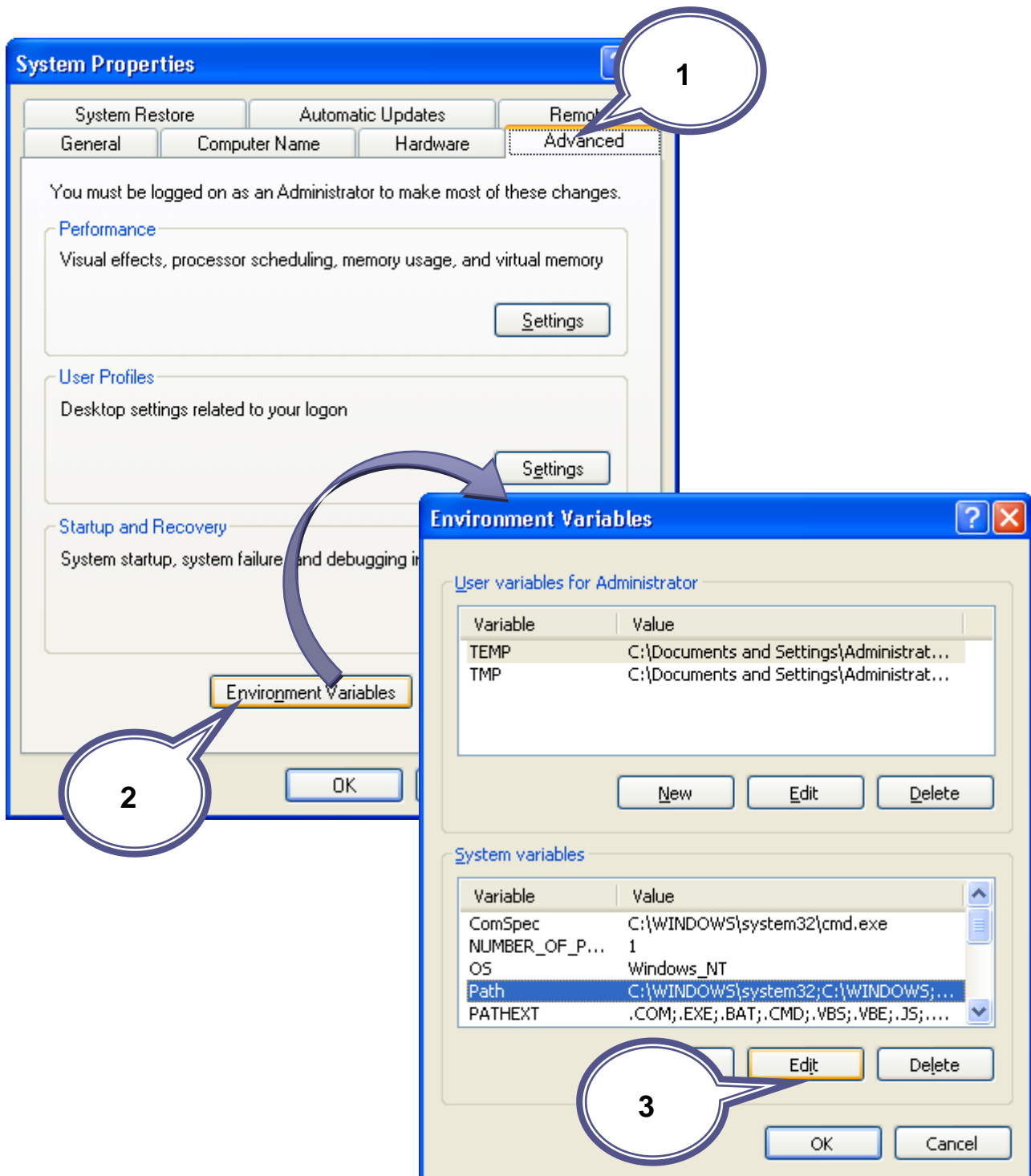
After installing the compiler, several compilers will be available from the Windows Command line. You can set the path environment variable so that you can execute This compiler on the command line by entering simple names, rather than by using Their full path names.

**Step 1: Right click on the “My Computer” icon on your desktop and select the “Properties” menu option**



Step 2: On the “System Properties” dialog box, click the “Environment Variables” button located under the “Advanced” sheet

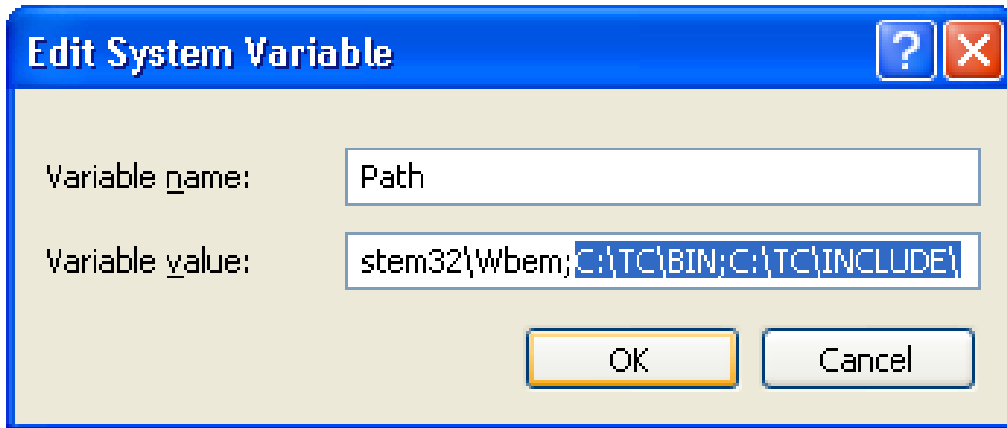
Step 3: On the “Environment Variables” dialog box, click the “Edit” button located in the “System variables” option



**Step 4: Add the target directory to the end of the variable value field**

A semi-colon is used as the separator between variable values.

For example, " ;c:\TC\BIN;c:\TC\INCLUDE\"



**Step 5: Restart the computer to allow your changes to take effect**



## 3.2. API for $\mu$ PAC-7186EX

---

To develop a custom program, ensure that the files below are installed the PC.

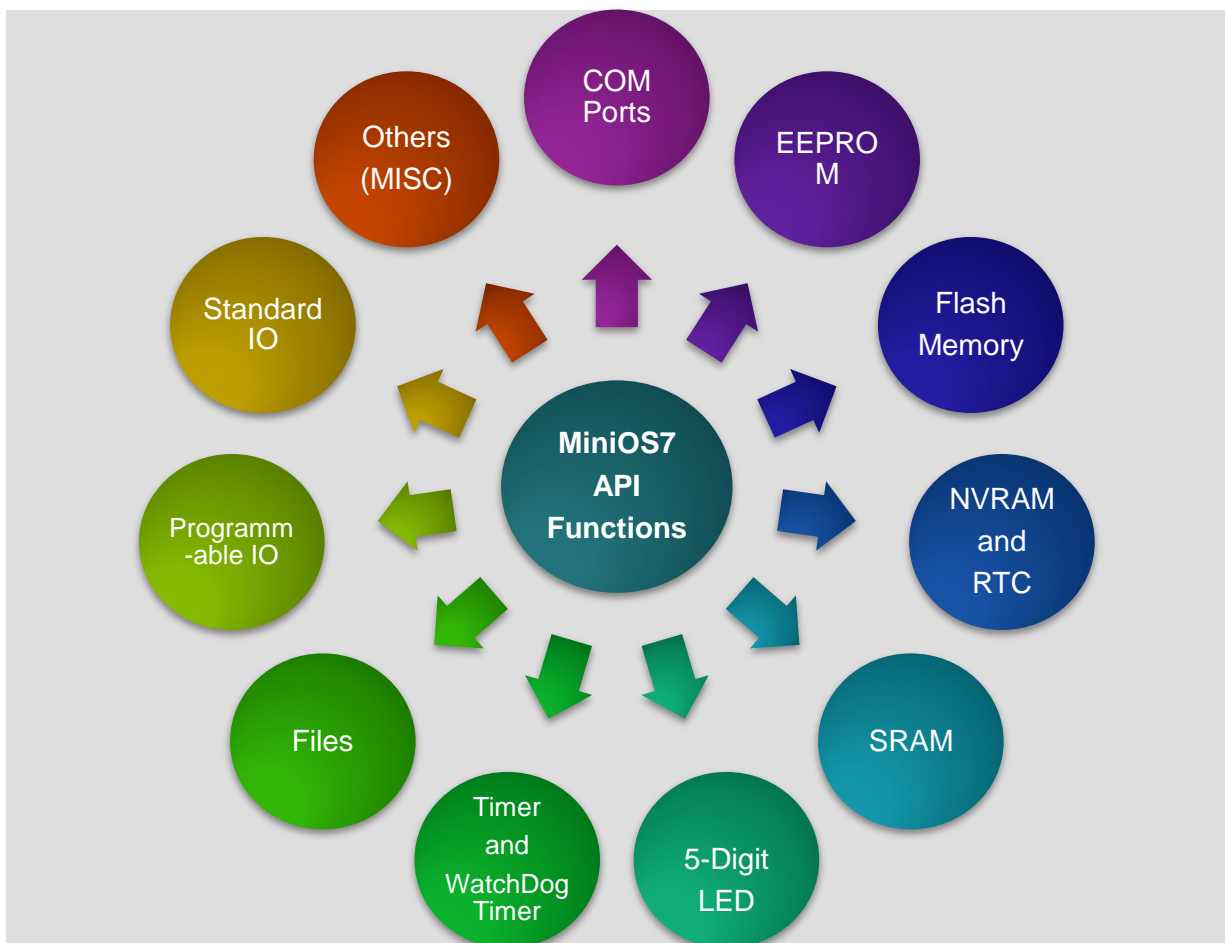
If they are not installed, refer to "section 2.2. Software Installation" .

### ➤ Functions Library – 7186e.lib

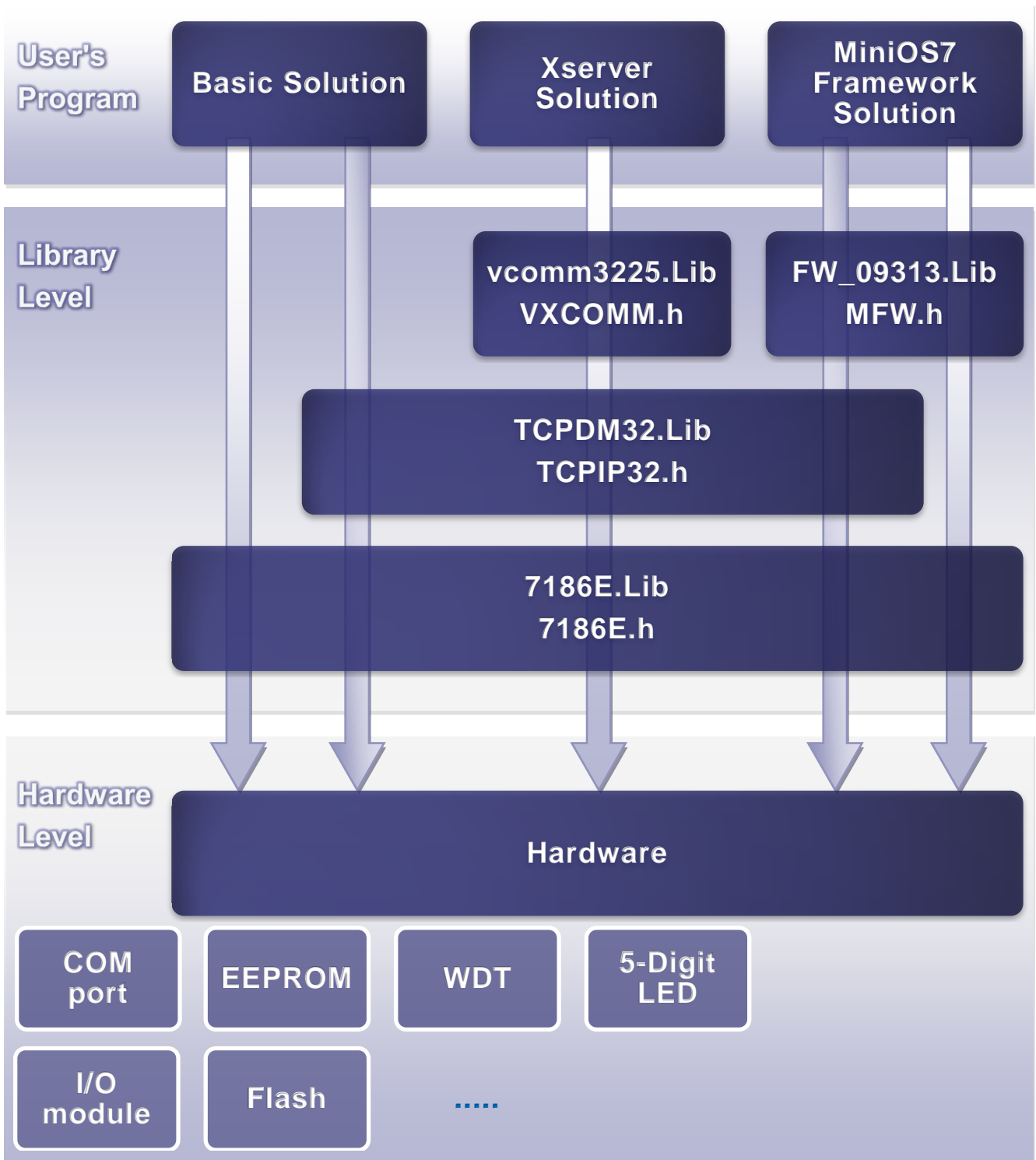
This file contains the MiniOS7 API (Application Programming Interface) and has hundreds of pre-defined functions related to your controller.

### ➤ Header File – 7186e.h

This file contains the forward declarations of subroutines, variables, and other identifiers used for the MiniOS7 API.



➤ System Structure



➤ **Bisic:**

Demos developed from main().

➤ **Xserver:**

Demos developed based on Xserver library.

Xserver is a library for TCP/IP server applications.

With an addional modbus library, users can develop programs with

1. Modbus/TCP slave
2. Modbus/TCP client
3. Modbus/RTU slave
4. Modbus/RTU client
5. Modbus/ASCII slave
6. Modbus/ASCII client
7. Modbus/TCP to Modbus/RTU gateway

The Modbus library and demos can be found in

CD:\Napdos\Modbus\7186e\Demo\

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/>

## ➤ **MiniOS7 Framework Solution:**

Demos developed based on MiniOS7 framework.

MiniOS7 framework is a library for general TCP/IP applications.

It is second generation library of Xserver and is much flexible and powerful than the Xserver.

Base on it, users can quickly and easily develop programs with

1. TCP Client
2. TCP Server
3. Web Server
4. UDP Client
5. UDP Server

With an additional modbus library, users can develop programs with

1. Modbus/TCP slave
2. Modbus/TCP client
3. Modbus/RTU slave
4. Modbus/RTU client
5. Modbus/ASCII slave
6. Modbus/ASCII client
7. Modbus/TCP to Modbus/RTU gateway

The Modbus library and demos can be found in

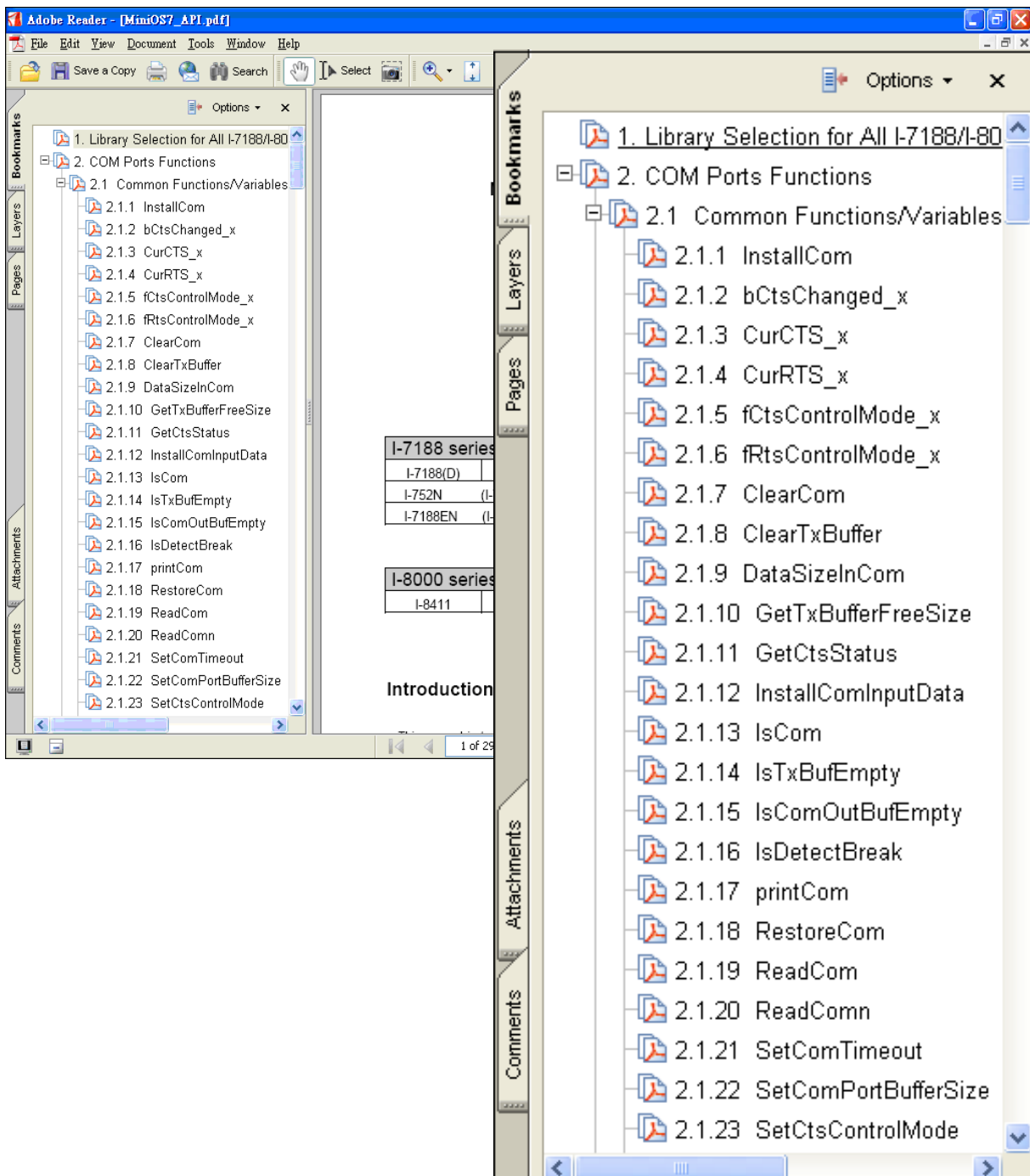
CD:\Napdos\Modbus\7186e\Demo\

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/>

For full usage information regarding the description, prototype and the arguments of the functions, please refer to the “MiniOS7 API Functions User Manual” located at:

CD:\Napdos\MiniOS7\Document

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/document/>



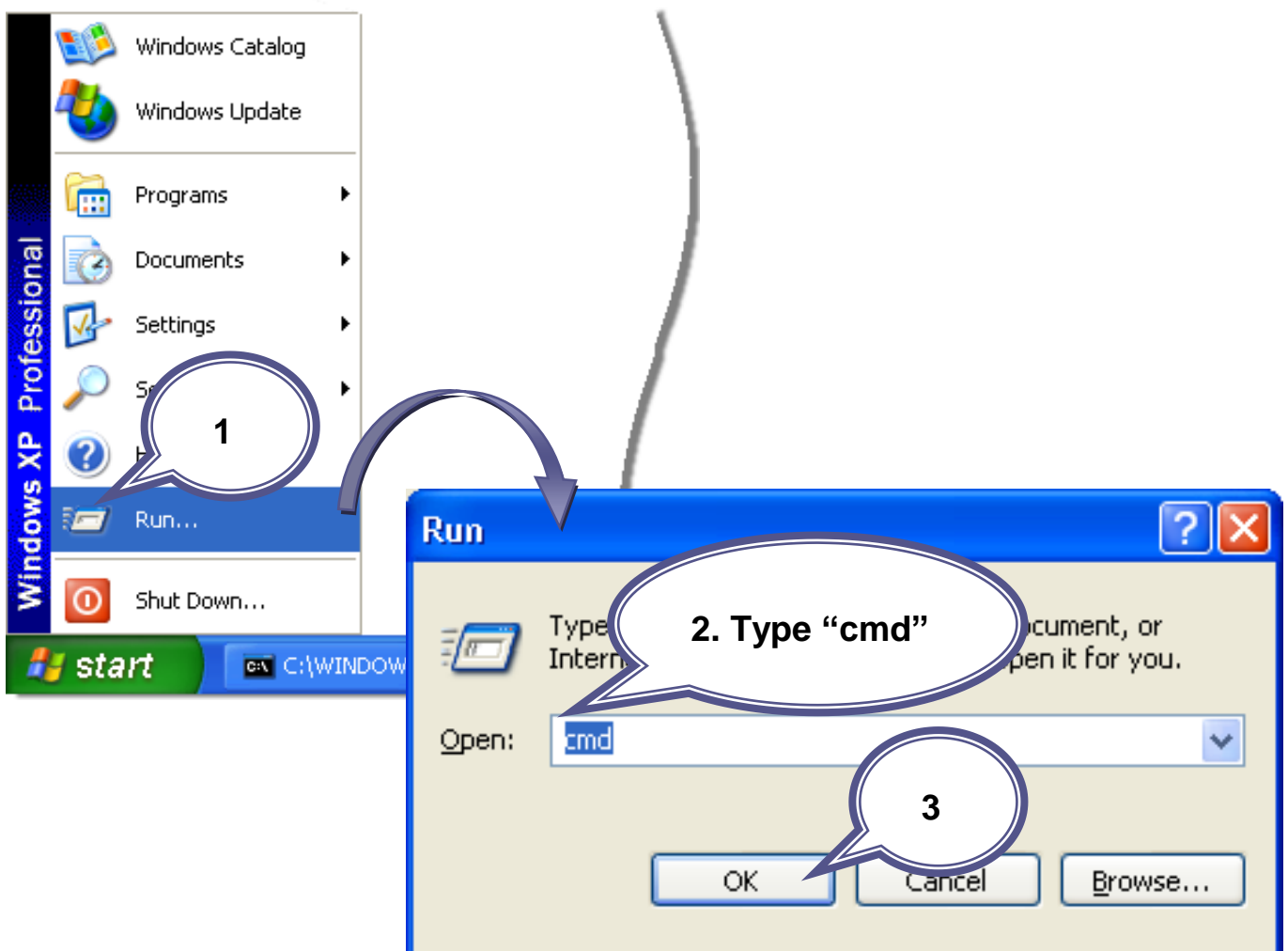
### 3.3. Build and run your first program

---

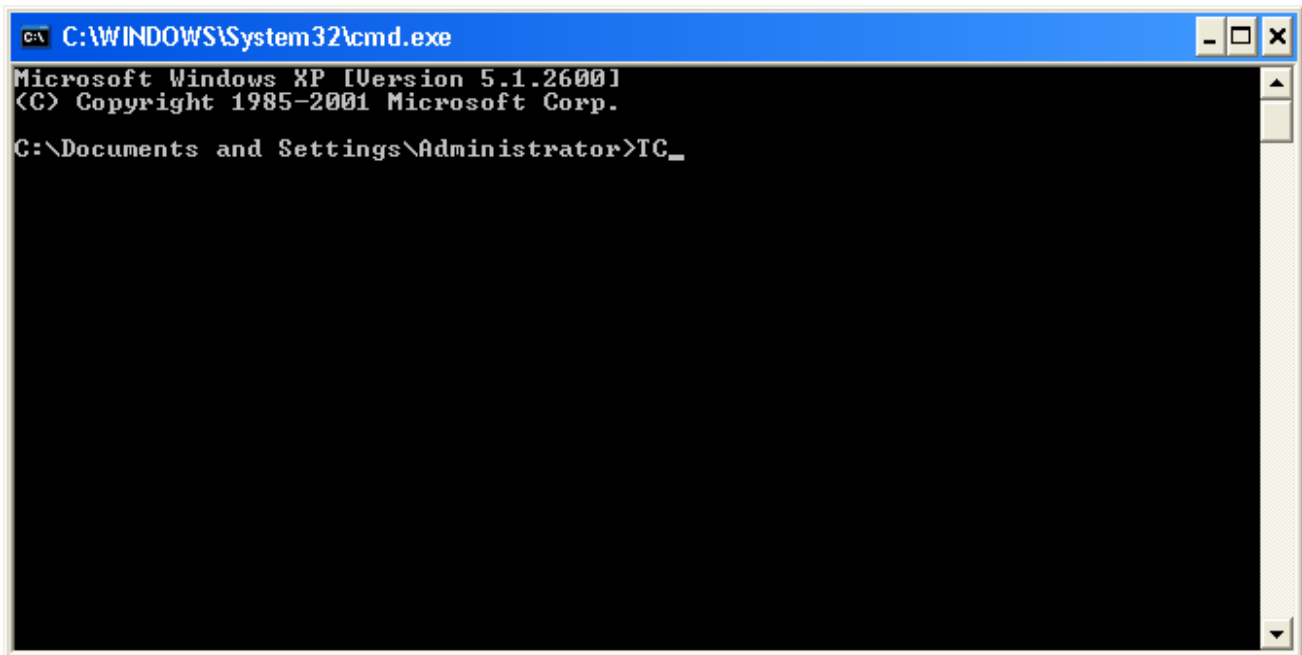
If you don't using the TC++ (Turbo C++) to write a program, please take the following steps.

#### Step 1: Open a MS-DOS command prompt

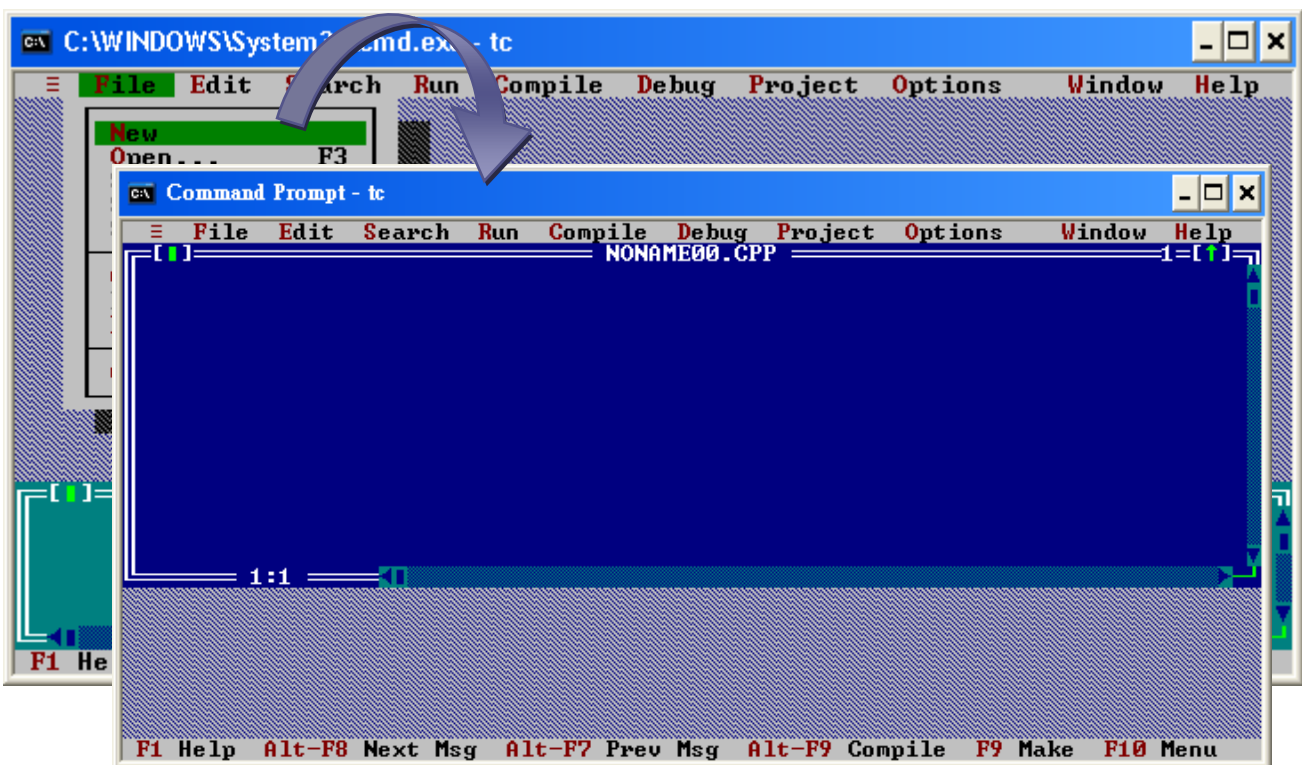
- i. Select "Run" from the "Start" menu
- ii. On the "Run" dialog box, type "cmd"
- iii. click the "OK" button



Step 2: At the command prompt, type "TC" and then press "Enter"



Step 3: Select "New" from the "File" menu to create a new source file



Step 4: Type the following code. Note that the code is case-sensitive

```
#include "7186e.h"

/* Include the header file that allows 8000e.lib functions to be used */

void main(void)
{

InitLib(); /* Initiate the 7186e library */

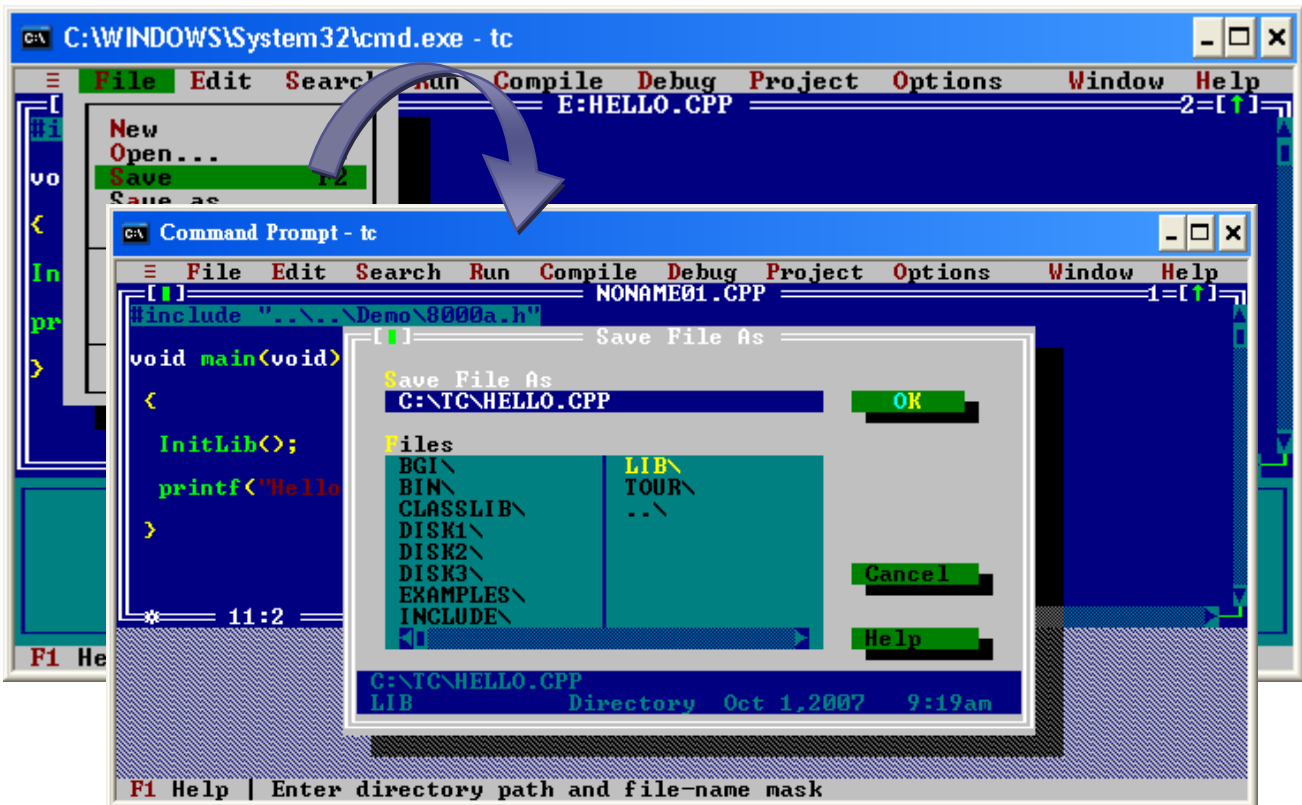
Print( "Hello world!\r\n" ); /* Print the message on the screen */

}
```



### Step 5: Save the source file

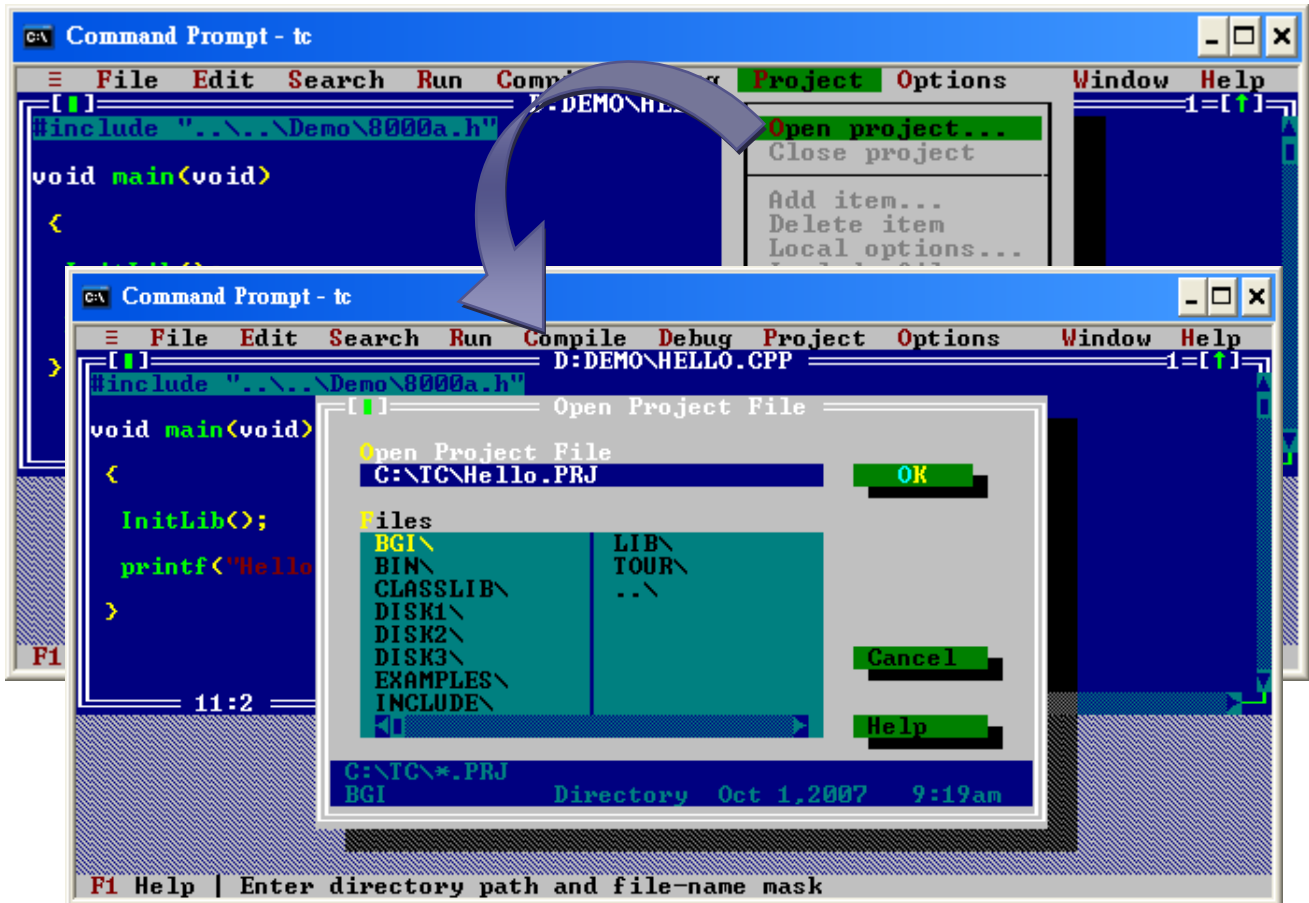
- i. Select "Save" from the "File" menu
- ii. Type the file name "Hello"
- iii. Select "OK"



If there is a text editor you are familiar with programs like Notepad you may use it to write the code as shown above, but it should be noted you must save the source code under a filename that terminates with the extension ".C" .

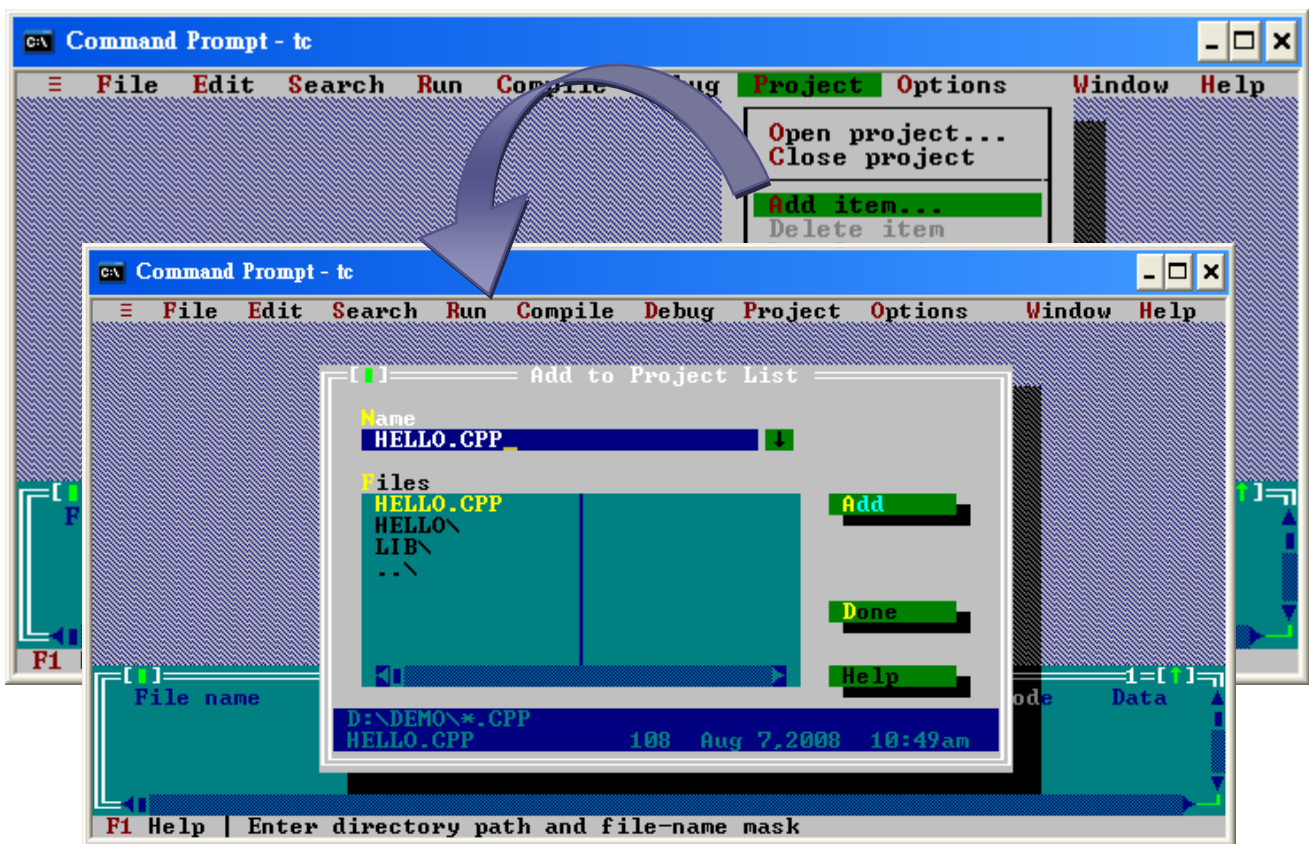
## Step 6: Create a project (\*.prj)

- i. Select "Open project..." from the "Project" menu
- ii. Type the project name "Hello"
- iii. Select "OK"



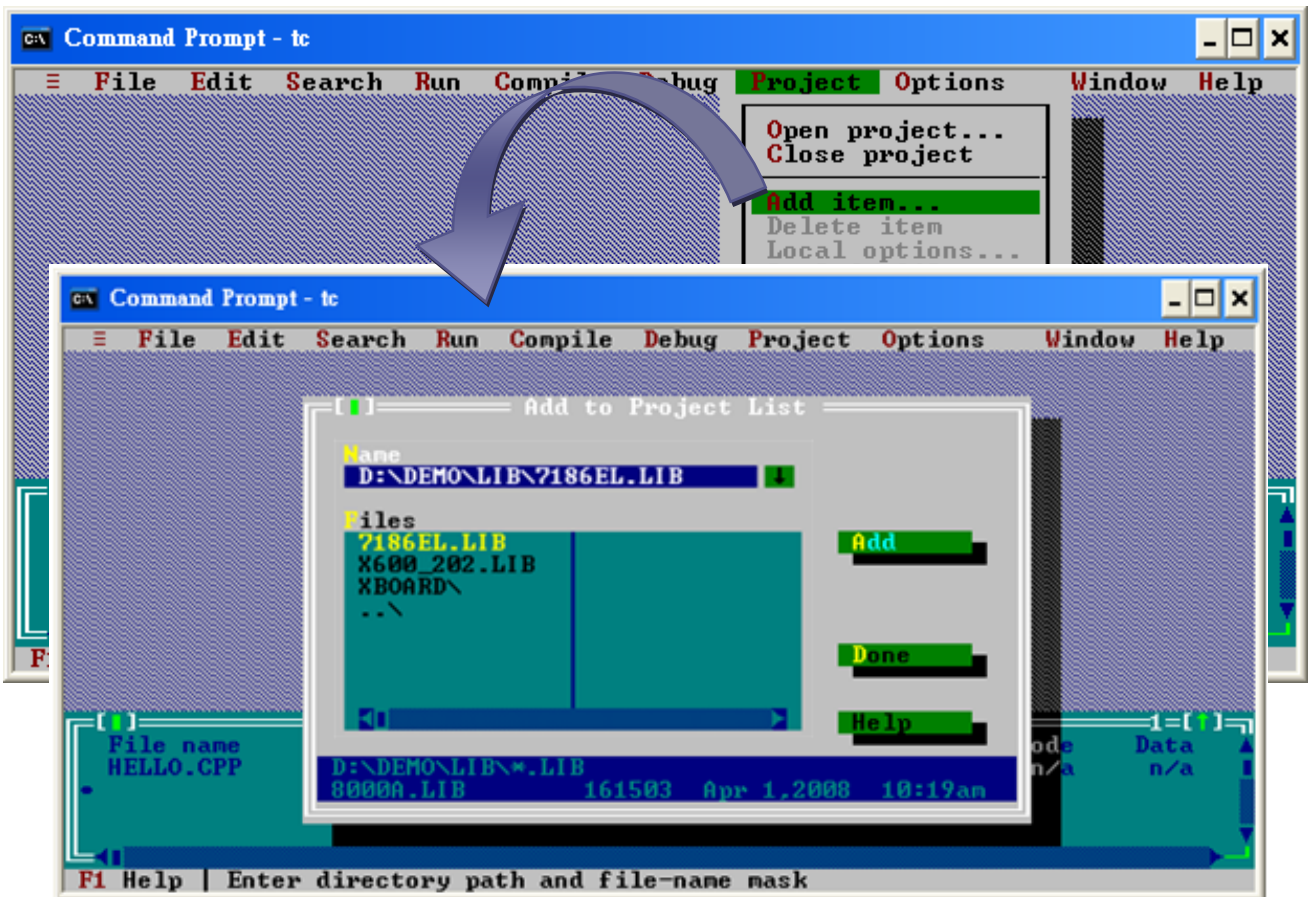
## Step 7: Add the necessary source files to the project (\*.CPP)

- i. Select "Add item..." from the "Project" menu
- ii. Type "\*.CPP" to display a list of all available source files
- iii. Choose the source files you require
- iv. Select "Add"
- v. Select "Done" to exit



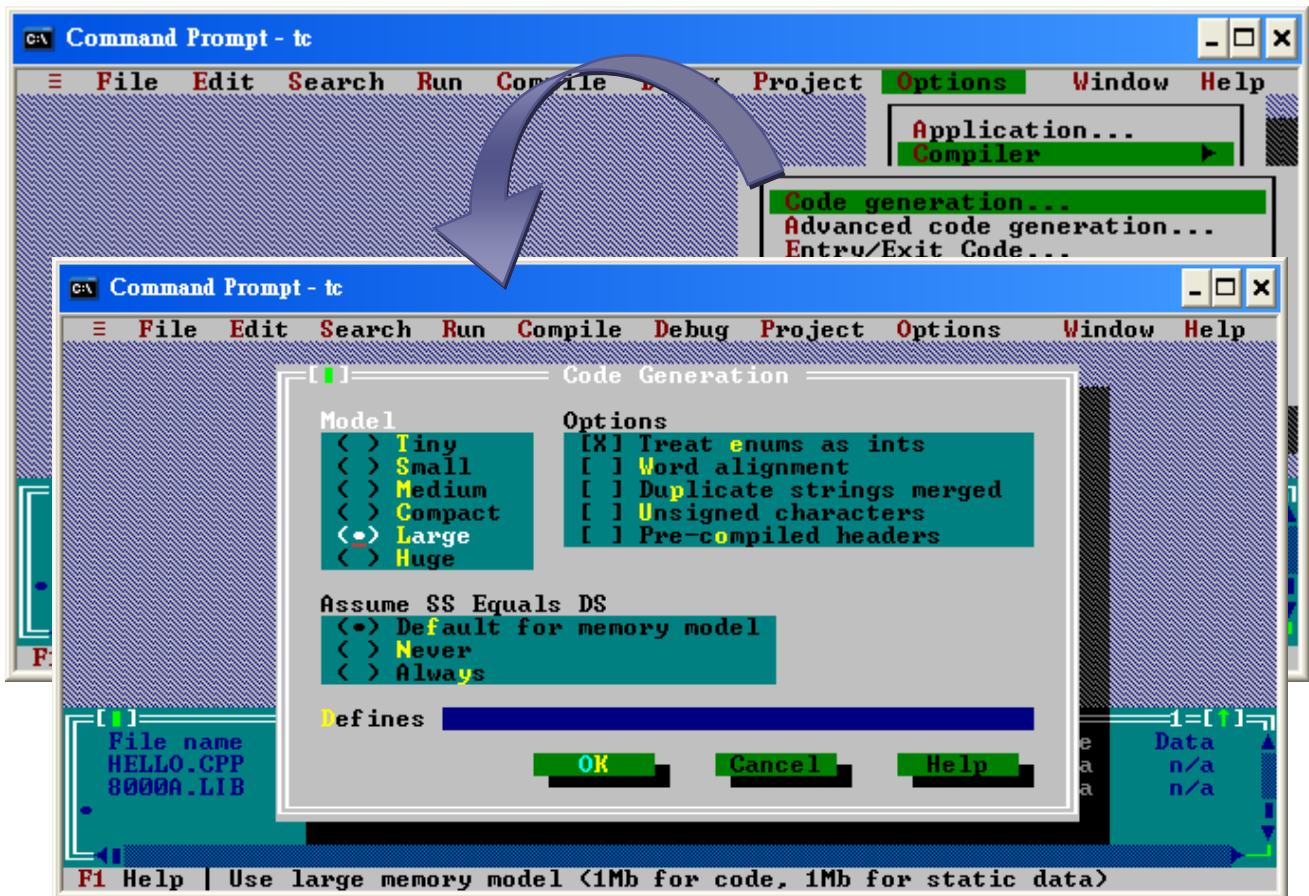
## Step 8: Add the necessary function libraries to the project (\*.lib)

- i. Select "Add item..." from the "Project" menu
- ii. Type "\*.LIB" to display a list of all available function libraries
- iii. Choose the function libraries you require
- iv. Select "Add"
- v. Select "Done" to exit



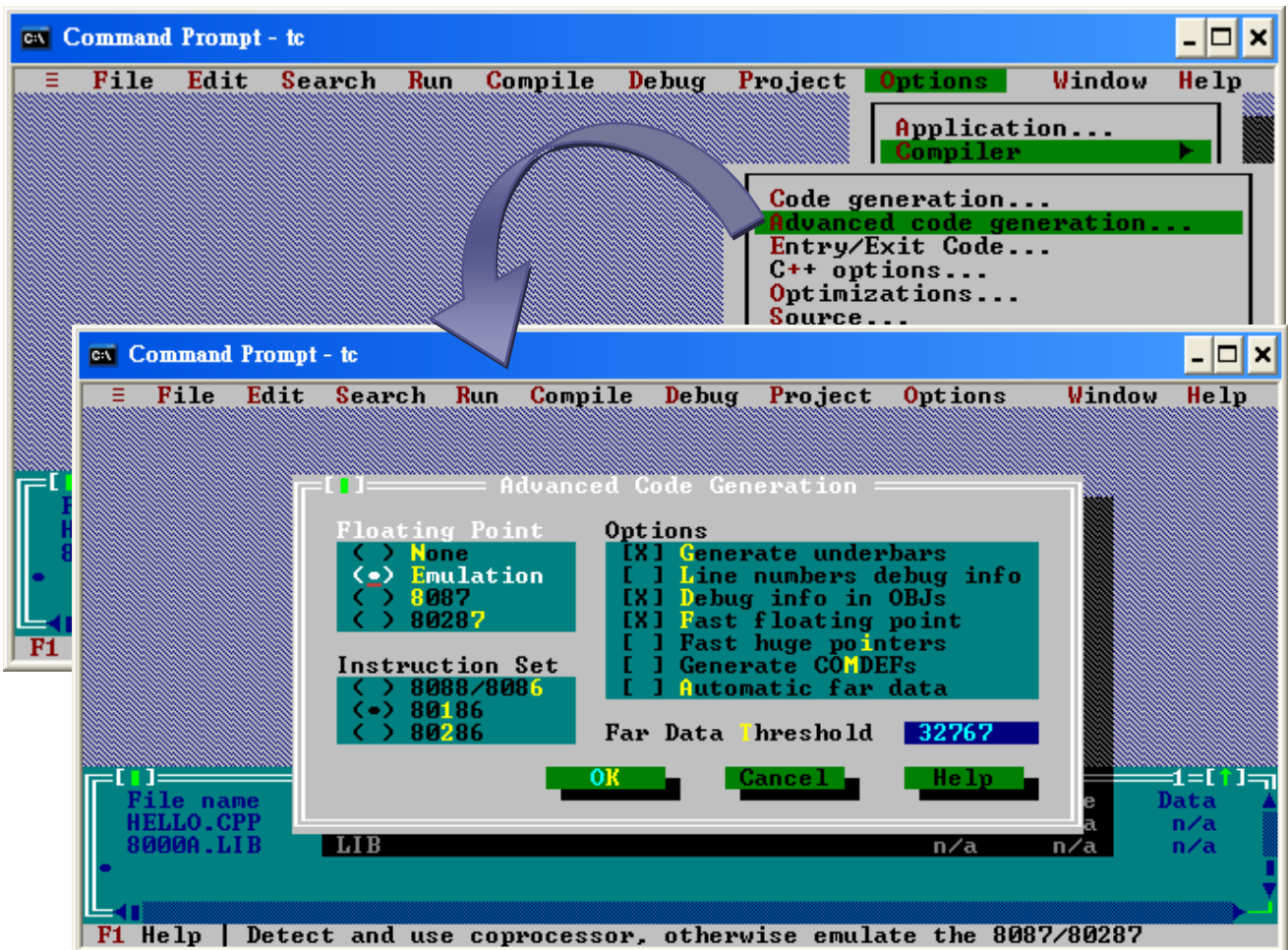
### Step 9: Set the memory model to large

- i. Select "Compiler" from the "Options" menu and then select "Code generation..."
- ii. On "Model" option, select "Large"
- iii. Select "OK"



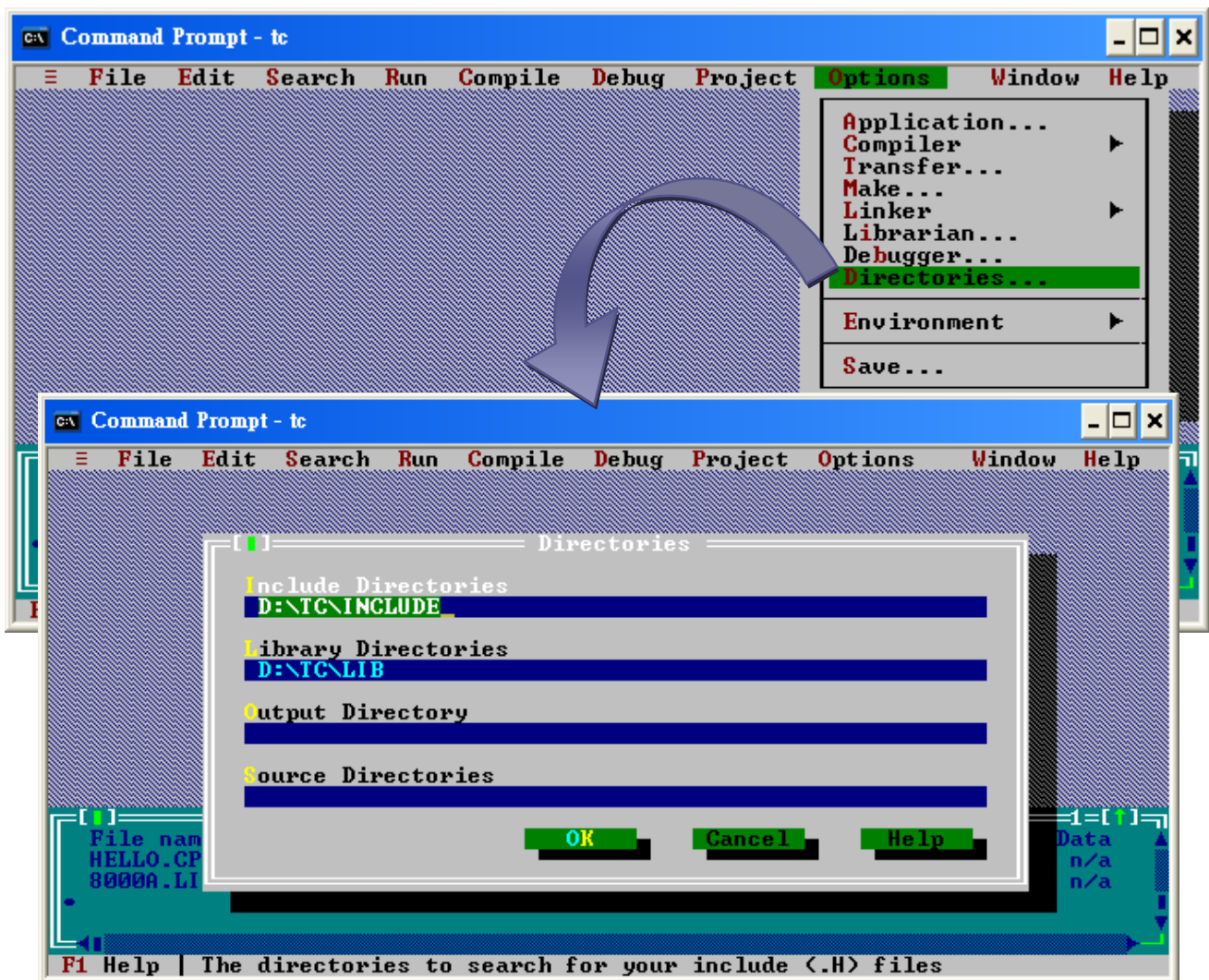
## Step 10: Set the memory model to large

- i. Select "Compiler" from the "Options" menu and then select "Advanced code generation..."
- ii. On "Floating Point" option, select "Emulation"
- iii. On "Instruction Set" option, select "80186"
- iv. Select "OK"

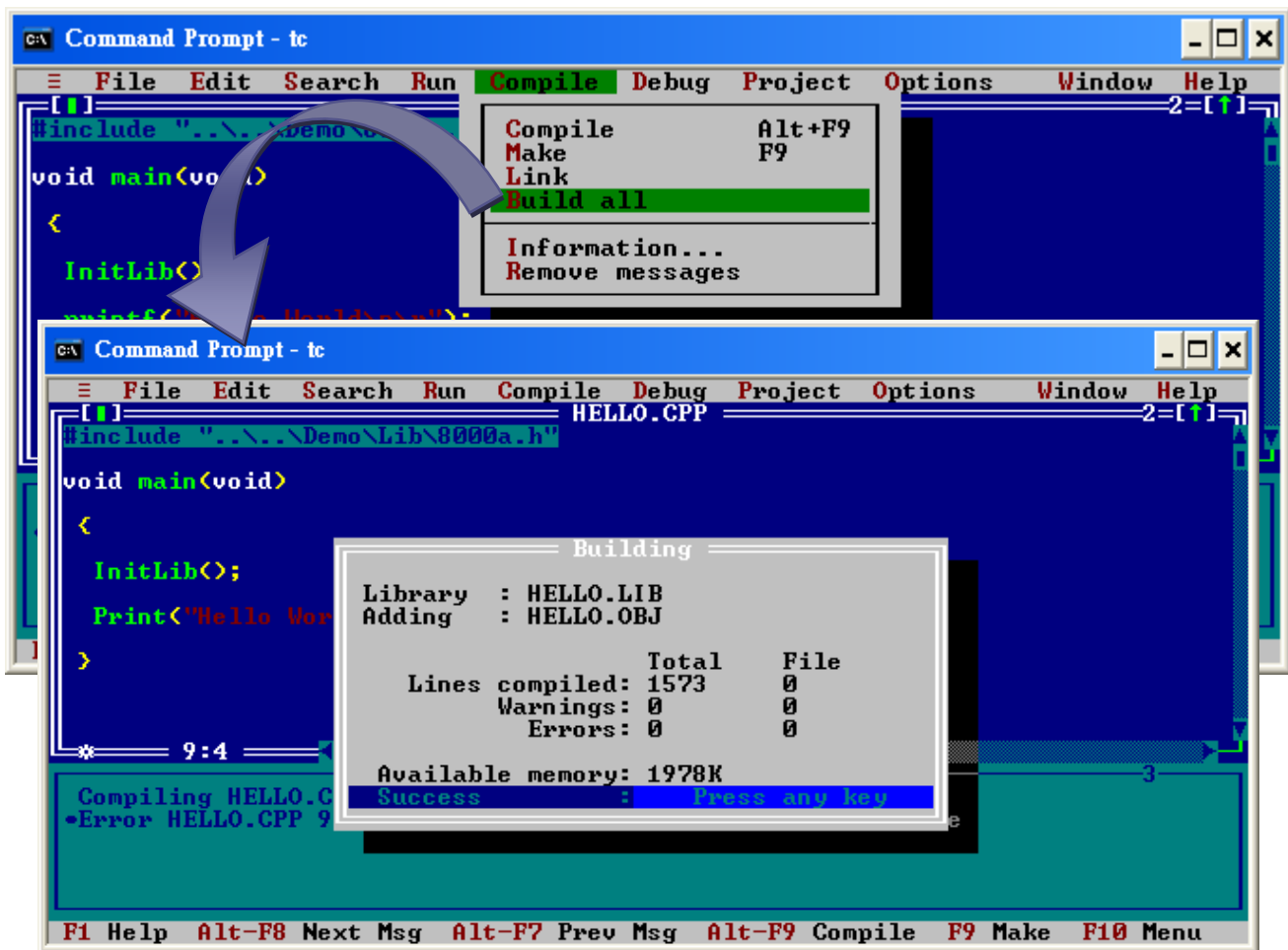


## Step 11: Set the memory model to large

- i. Select "Directories..." from the "Options" menu
- ii. On "Include Directories" option, specify the header file
- iii. On "Library Directories" option, specify the function library file
- iv. Select "OK"



Step 12: Select "Build all" from the "Compile" menu to build the project

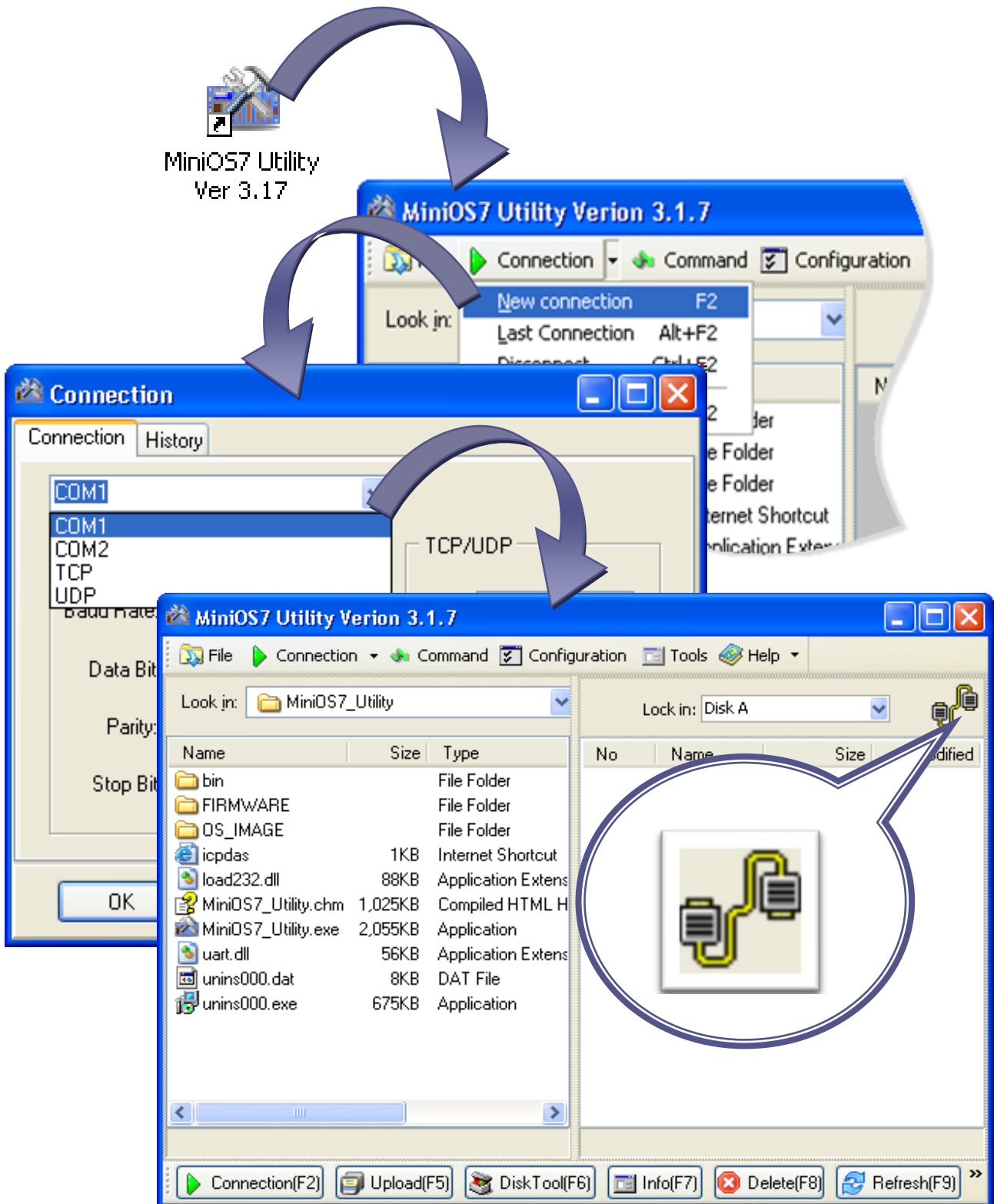




### Step 13: Use the MiniOS7 Utility to connect the $\mu$ PAC-7186EX

For more detailed information about this process, please refer to section

“2.3.1. Establishing a connection” .



## Step 14: Upload and execute files

For more detailed information about this process, please refer to section

"2.3.2. UPloading and executing programs on  $\mu$ PAC-7186EX

The image shows a sequence of screenshots from the MiniOS7 Utility software. The top screenshot shows the 'File' menu with 'Hello' selected, and a context menu with 'Run' highlighted. The middle screenshot shows a terminal window with the command 'run #0' and the output 'Hello 8000! <Flash memory is 512 K>'. The bottom screenshot shows the 'Configuration' menu with 'AutoRun' checked, and a terminal window with the command 'run #0' and the output 'Hello 8000! <Flash memory is 512 K>'. A blue callout box with the text 'Making programs start automatically' points to the 'AutoRun' checkbox. A white callout box with the text 'One is the "Hello" application file, and the other is the "autoexec.bat" batch file' points to the 'hello.exe' and 'autoexec...' files in the file list.

**Making programs start automatically**

One is the "Hello" application file, and the other is the "autoexec.bat" batch file

## 4. API and Demo Program Reference

---

There are several demo programs that have been designed for  $\mu$ PAC-7186EX. You can examine the demo source code, which includes numerous comments, to familiarize yourself with the MiniOS7 API. This will allow you to quickly develop your own applications quickly by modifying these demo programs.

### ➤ Basic

Folder	Demo	Explanation
File	Config_1_Basic	Reads information from a text file (basic).
	Config_2_Advanced	Reads a config file (text file)(advanced).
Hello	Hello_C	Reads the library version and flash memory size.
	Hello_C++	
Misc	Reset	Resets the software.
	Runprog	Illustrates how to select an item and run it.
	Serial	Illustrates how to retrieve 64-bit hardware unique serial number.
	Watchdog	Enables the WDT or bypasses the enable WatchDog function.
Memory	EEPROM	Shows how to write a value to the EEPROM.
Memory	Flash	Shows how to write and erase

Folder	Demo	Explanation
		the Flash.
LED	Led	Shows how to control the red LED display.
	Seg7led	Shows how to control the red 7-segment display.
DateTime	DateTime	Shows how to read and write the date and time from the RTC.
7K87K_ Module	7K87K_DI_for_Com	Shows how to connect and control the 7K or 87K series modules via COM2.
	7K87K_DO_for_Com	
	7K87K_AI_for_Com	
	AO_22_26_for_Com	
	AO_024_for_Com	
Com port	C_Style_IO	(1) Shows how to write a function to input data. (2) Shows how to receive a string. (3) Shows how to use a C function: sscanf or just use Scanf()
	Receive	Receives data from COM port. Slv_COM.c is in non-blocked mode Receive.c is in blocked mode.
	Slv_COM	A slave COM Port demo for (request/reply) or (command/response) applications.
	ToCom_In_Out	Illustrates how to Read/Write byte data via COM Port.
<b>7186FD</b>	<b>Utiliy</b>	<b>Utility for the MiniOS7 File System.</b>

Folder	Demo	Explanation
(for 64MB flash memory on $\mu$ PAC-7186EX-FD)		Operations Include Dir, Read, Write, etc.
	MFS_QA	Quality assurance program for the MiniOS7 File System. Including function test, read/write performance test.
	Puts	How to write a string to a file in the 64MB flash memory
	Gets	How to get a string from a file in the 64MB flash memory

For more information about these demo programs, please refer to:

CD:\ NAPDOS\7186e\ Demo\Basic\

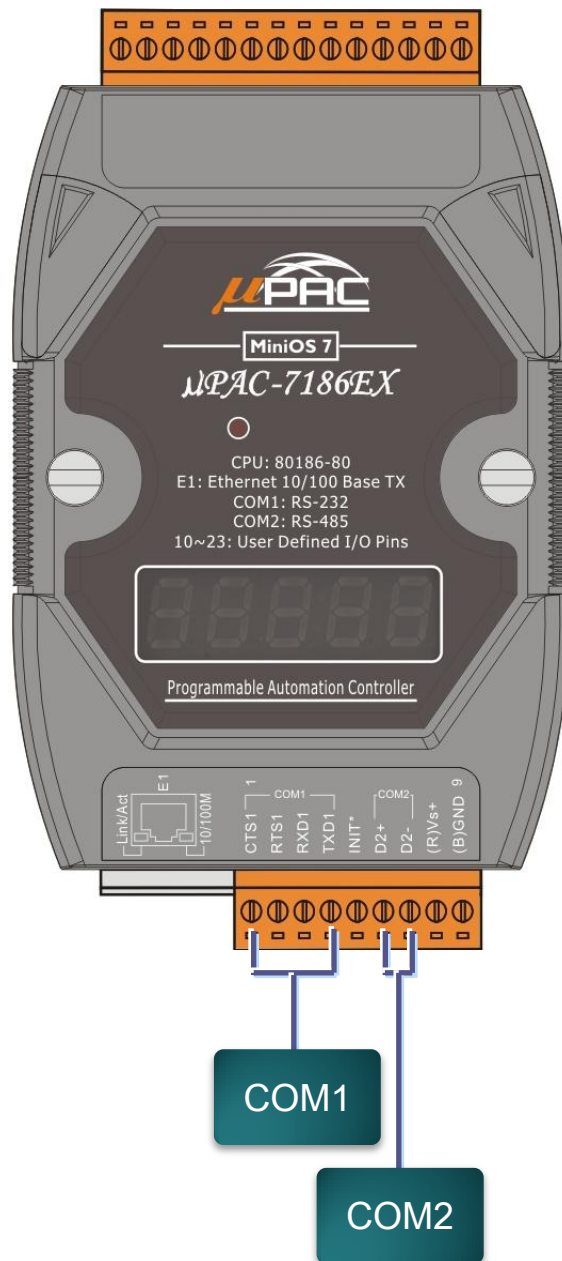
[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/)

## 4.1. API for COM port

---

➤ The  $\mu$ PAC-7186EX include two COM ports

1. MiniOS7 COM port functions
2. (C style) Standard COM port functions



### 4.1.1.Types of COM port functions

- There are two types of functions below for using COM port.

1. MiniOS7 COM port functions
2. (C style) Standard COM port functions



You have the alternative of MiniOS7 COM ports functions or (C style) Standard COM port functions. If you choose the ones, then the another can not be used.

---

- Summarize the results of the comparison between MiniOS7 COM port functions and (C style) Standard COM port functions:

Kinds of Functions	COM Port	Buffer		Functions			
		RX	TX	Check data	Send data	Read data	Show data
MiniOS7 COM port	1, 2, etc.	1 KB	1 KB	IsCom()	ToCom()	ReadCom()	printCom()
(C style) Standard COM port	1 (Note)	512 Bytes	256 Bytes	Kbhit()	Puts() Putch()	Getch()	Print()

## 4.1.2.API for MiniOS7 COM port

### API for using COM ports

---

#### 1. InstallCom()

Before any COM Port can be used, the driver must be installed by calling InstallCom().

#### 2. RestoreCom()

If the program calls InstallCom(), the RestoreCom() must be called to restore the COM Port driver.

### API for checking if there is any data in the COM port input buffer

---

#### 3. IsCom()

Before reading data from COM port, the IsCom() must be called to check whether there is any data currently in the COM port input buffer.

### API for reading data from COM ports

---

#### 4. ReadCom()

After IsCom() confirms that the input buffer contains data, the ReadCom() must be called to read the data from the COM port input buffer.



## API for sending data to COM ports

---

### 5. ToCom()

Before sending data to COM ports, the ToCom() must be called to send data to COM ports.

For example, reading and receiving data through the COM1:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int quit=0, data;

    InitLib(); /* Initiate the 7186e library */
    InstallCom(1, 115200, 8, 0, 1); /* Install the COM1 driver */

    while(! quit)
    {
        if(IsCom(1)) /* Check if there is any data in the COM port input buffer */
        {
            data=ReadCom(1); /* Read data from COM1 port */
            ToCom(1, data); /* Send data via COM1 port */
            if(data==' q' ) quit=1; /* If 'q' is received, exit the program */
        }
    }

    RestoreCom(1); /* Uninstall the COM1 driver */
}
```

## API for showing data from COM ports

---

### 6. printCom()

Functions such as printfCom() in the C library allow data to be output from COM ports.

For example, showing data from the COM1 port:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int i;

    /* Initiate the 7186e library */
    InitLib();
    InstallCom(1, 115200, 8, 0, 1); /* Install the COM1 driver */
    for (i=0;i<10;i++)
    {
        printCom(1," Test %d\n\r" , i);
    }
    Delay(10); /* Wait for all data are transmitted to COM port */
    RestoreCom(1);
}
```

► For more demo program about the COM port, please refer to:

CD:\NAPDOS\7186e\ Demo\Basic\com\_port

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/com\\_port](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/com_port)

### 4.1.3.API for standard COM port

- The standard COM port is used to download program from PC to the  $\mu$ PAC-7186EX.



The following configurations of the standard COM port are fixed:

Baudrate=115200 bps, Data format=8 bits

Parity check=none, Start bit=1, Stop bit=1

---

#### API for checking if there is any data in the input buffer

---

##### 1. Kbhit()

Before reading data from standard I/O port, the kbhit() must be called to check whether there is any data currently in the input buffer.

#### API for reading data from standard I/O port

---

##### 2. Getch()

After kbhit() confirms that the input buffer contains data, the Getch() must be called to read data from the input buffer.

## API for sending data to standard I/O port

---

### 3. Puts() – For sending a string

Before sending data to standard I/O port, the Puts() must be called to send data to COM Port.

### 4. Putch() – For sending one character

Before sending data to standard I/O port, the Putch() must be called to send data to COM Port.

## API for showing data from standard I/O port

---

### 5. Print()

Functions such as Print() in the C library allow data to be output from the COM Port.

For example, reading and receiving data through COM1:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int quit=0, data;

    InitLib(); /* Initiate the 7186e library */

    while(!quit)
    {
        if(Kbhit() /* Check if any data is in the input buffer */)
        {
            data=Getch(); /* Read data from COM1 */
            Putch(data); /* Send data to COM1 */
            if(data==' q' ) quit=1; /* If 'q' is received, exit the program */
        }
    }
}
```

For example, showing data through COM1:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int i;

    /* Initiate the 7186e library */
    InitLib();

    for(i=0;i<10;i++)
    {
        Print( "Test %d\n\r" ,i);
    }
}
```



#### 4.1.4. Comparing with MiniOS7 COM port function and Standard COM port function

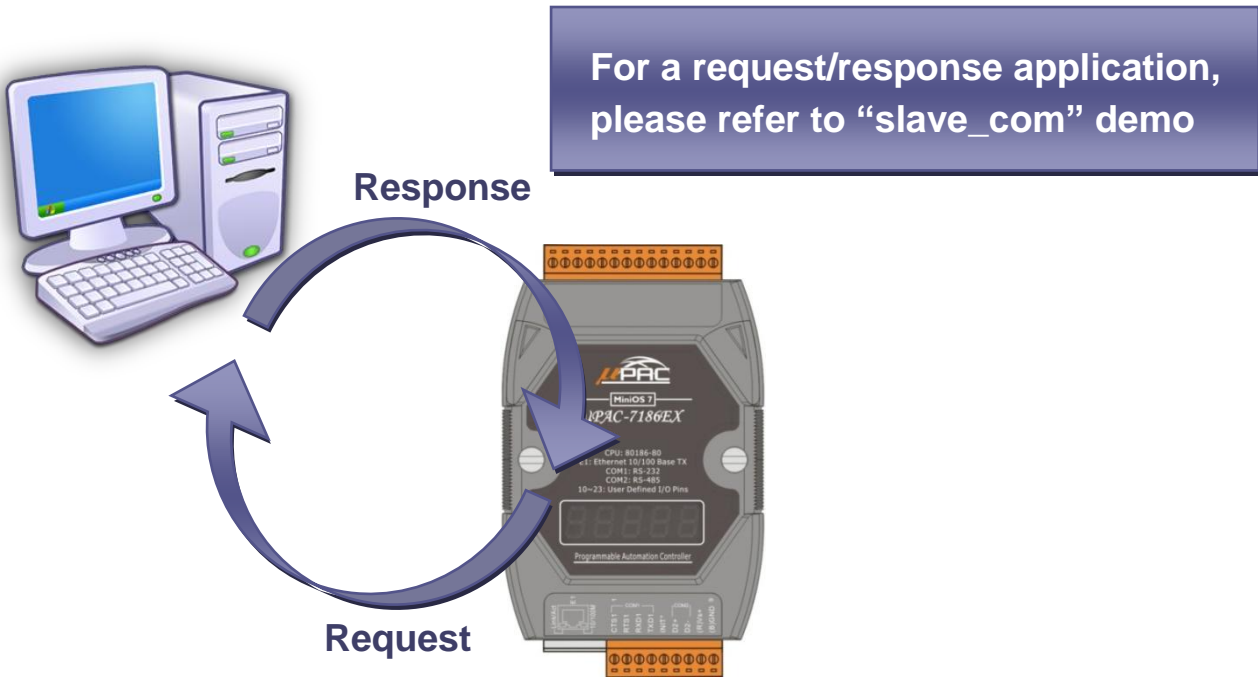
For example, learning to show the ASCII code:

MiniOS7 COM port functions	Standard COM port functions
<pre>#include&lt;stdio.h&gt; #include "7186e.h"  void main(void) {     unsigned char item;      InitLib();      InstallCom(1, 115200, 8, 0, 1);      printCom(1," Hits any key.\n" );     printCom(1," Hit the ESC to exit!\n" );      for(;;)     {         if(IsCom(1))         {             item=ReadCom(1);             if(item== ' q' )             {                 return;             }         }         else         {             printCom(1," -----\n\r" );             printCom(1," char:" );</pre>	<pre>#include&lt;stdio.h&gt; #include "7186e.h"  void main(void) {     unsigned char item;      InitLib();      Print("Hits any key.\n");     Print("Hits the ESC to exit !\n");      for(;;)     {         if(kbhit())         {             item=Getch();             if(item== ' q' )             {                 return;             }         }         else         {             Print(" -----\n\r" );             Print( "char: " );</pre>

<pre> ToCom(1,item); printCom(1,"\n\rASCII(%c)\n\r" ,item); printCom(1, "Hex(%02X)\n\r" ,item); ); } } } Delay(10); RestoreCom(1); } </pre>	<pre> Putch(item); Print("\n\rASCII(%c)\n\r" ,item); Print( "Hex(%02X)\n\r" ,item); } } } } </pre>
---	--

#### 4.1.5. Request/Response protocol define on COM port

Request/Response communication is very typical protocol architecture, if you want to design a command set of communication protocol as table below, you can refer to "slave\_com" demo.



Request	Response
c1	Debug information: Command1 Command1
c2	Debug information: Command2 Command2
Q	Debug information: Quick program
Other command	Debug information: Unknown command

For more demo program about the COM port, please refer to:

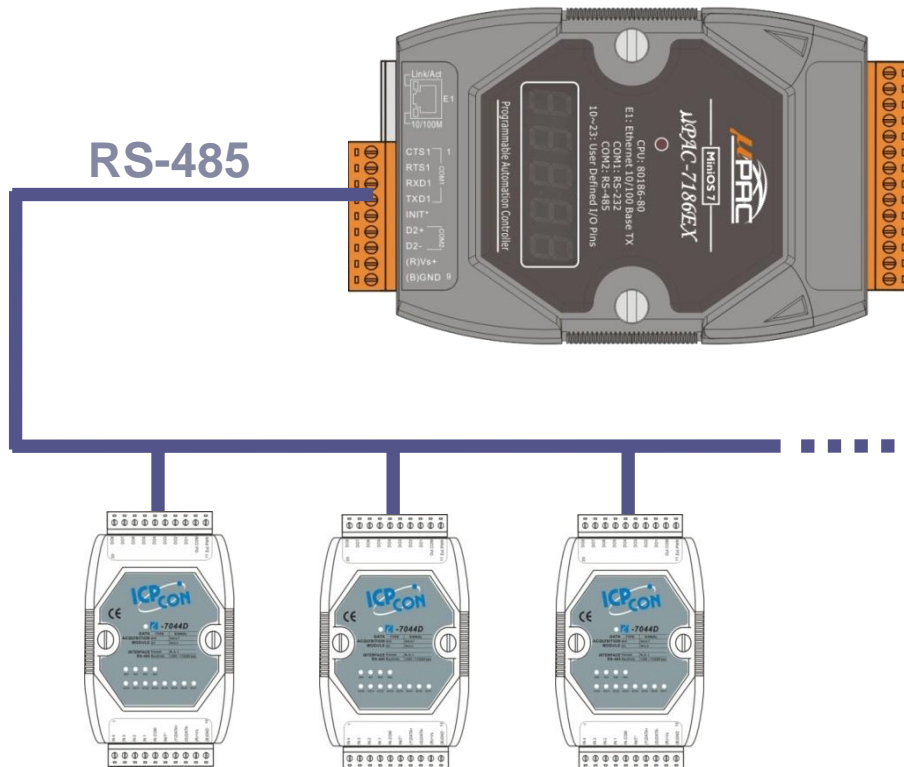
CD:\NAPDOS\7186e\ Demo\Basic\com\_port

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/com\\_port](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/com_port)

## 4.2. API for I/O modules

---

- The  $\mu$ PAC-7186EX is equipped with a RS-485 communication interface, COM2, to access the i-7K series I/O modules for a wide range of RS-485 network application, as shown below.



### Steps to communicate with i-7K series I/O modules:

- Step 1: Use `Installcom()` to install the COM port driver.
- Step 2: Use `SendCmdTo7000(2,...)` to send commands
- Step 3: Use `ReceiveResponseFrom7000_ms()` to get the response.
- Step 4: Use `RestoreCom()` to restore the COM port driver

For example, to send a command '\$01M' to i-7K I/O module for getting the module name:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    unsigned char InBuf0[60];
    InitLib(); /* Initiate the 7186e library */

    InstallCom(1,115200,8,0,1); /* Install the COM1 driver */
    InstallCom(2,115200,8,0,1); /* Install the COM2 driver */

    SendCmdTo7000(2," $01M" ,0); /* Send a command to COM2 */

    /* Timeout = 50ms, check sum disabled */
    ReceiveResponseFrom7000_ms(2,InBuf0,50,0);

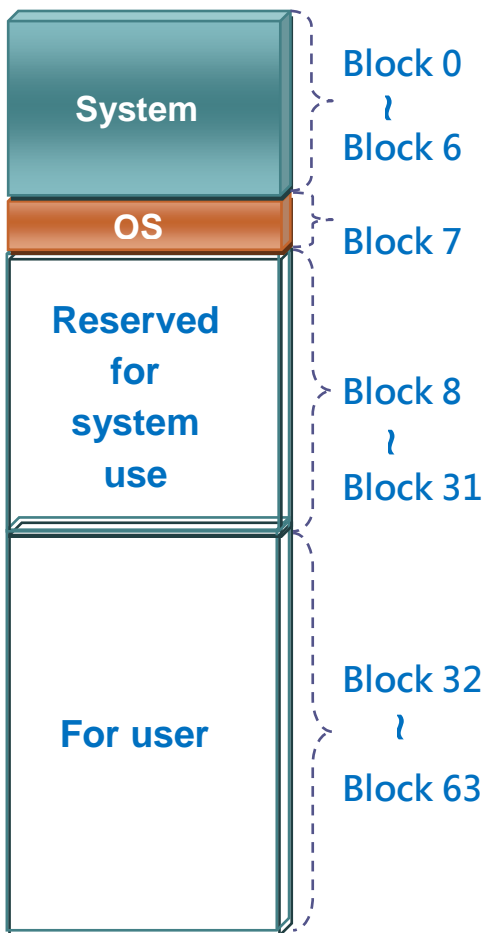
    printCom(1," Module Name = %s" , InBuf0);
    Delay(10); /* Wait for all data are transmitted to COM port */
    RestoreCom(1); /* Uninstall the COM1 driver */

    RestoreCom(2); /* Uninstall the COM2 driver */
}
```

## 4.3. API for EEPROM

---

- The EEPROM contains 64 blocks (block 0 ~ 63), and each block has 256 bytes (address 0 ~ 255), with a total size of 16,384 bytes (16K) capacity.
- The default mode for EEPROM is write-protected mode.
- The system program and OS are stored in EEPROM that are allocated as shown below.



### API for writing data to the EEPROM

#### 1. EE\_WriteEnable()

Before writing data to the EEPROM, the EE\_WriteEnable() must be called to write-enable the EEPROM.

#### 2. EE\_WriteProtect()

After the data has finished being written to the EEPROM, the EE\_WriteProtect() must be called in order to write-protect the EEPROM.

#### 3. EE\_MultiWrite()

After using the EE\_WriteEnable() to write-enable EEPROM, the EE\_MultiWrite() must be called to write the data.

## API for reading data from the EEPROM

---

### 4. EE\_MultiRead()

The EE\_WriteEnable() must be called to read data from the EEPROM no matter what the current mode is.

For example, to write data to block1, address 10 of the EEPROM:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int data=0x55, data2;

    InitLib(); /* Initiate the 7186e library */
    EE_WriteEnable();
    EE_MultiWrite(1,10,1,&data);
    EE_WriteProtect();

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

► For more demo program about the EEPROM, please refer to:

CD:\NAPDOS\7186e\ Demo\Basic\memory

∩

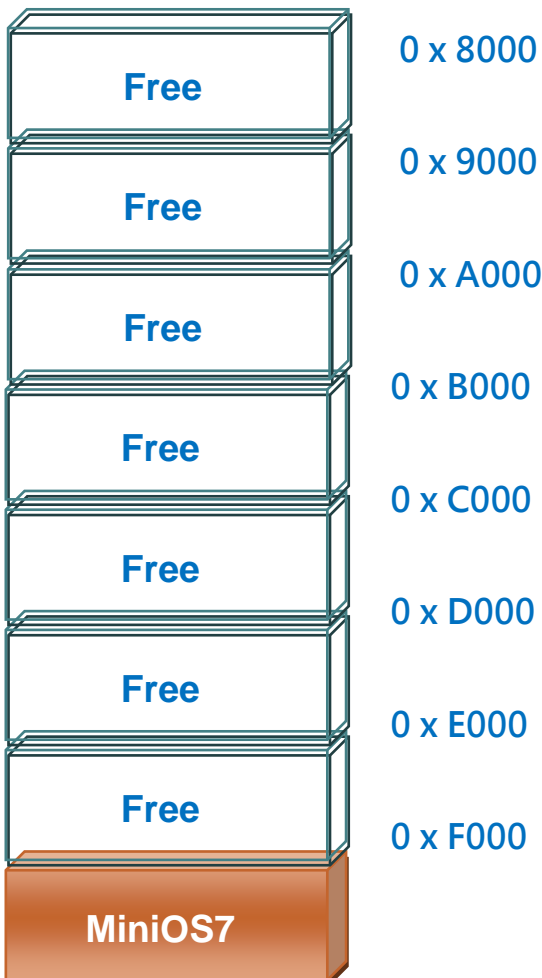
## 4.4. API for Flash Memory

---

Free: 448 K bytes

MiniOS7: 64 K bytes

Total Size: 512 K bytes



- The μPAC-7186EX module contains 512 Kbytes of Flash memory.
- MiniOS7 uses the last 64K bytes, the other parts of the memory are used to store user programs or data.
- Each bit of the Flash memory only e can be written from 1 to 0 and cannot be written from 0 to 1.
- Before any data can be written to the Flash memory, the flash must be erased , first which returns all data to 0xFF, meaning that all data bits are set to "1" . Once their is completed, new data can be written.

API for writing data to the Flash Memory

---



## 1. FlashWrite()

The FlashWrite() must be called to write data to the Flash Memory.

## API for reading data from the Flash Memory

---

## 2. FlashRead()

The FlashRead() must be called to read data from the Flash Memory.

For example, to write an integer to segment 0xD000, offset 0x1234 of the Flash memory:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int data=0xAA55, data2;
    char *dataptr;
    int *dataptr2;

    InitLib(); /* Initiate the 7186e 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;
}
```

► For more demo program about the Flash memory, please refer to:

CD:\NAPDOS\7186e\Demo\Basic\memory

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/memory](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/memory)

## 4.5. API for NVRAM and RTC

---

- The  $\mu$ PAC-7186EX is equipped with an RTC (Real Time Clock), 31 bytes of NVRAM can be used to store data.
- NVRAM is SRAM, but it uses battery to keep the data, so the data in NVRAM does not lost its information when the module is power off.
- NVRAM has no limit on the number of the re-write times. (Flash and EEPROM both have the limit on re-write times) If the leakage current is not happened, the battery can be used 10 years.

### API for writing data to the NVRAM

---

#### 1. WriteNVRAM()

The WriteNVRAM() must be called in order to write data to the NVRAM.

### API for reading data from the NVRAM

---

#### 2. ReadNVRAM()

The ReadNVRAM() must be called in order to write data to the NVRAM.

For example, use the following code to write data to the NVRAM address 0:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int data=0x55, data2;

    InitLib(); /* Initiate the 7186e library */
    WriteNVRAM(0,data);
    data2=ReadNVRAM(0); /* Now data2=data=0x55 */
}
```

For example, the following can be used to write an integer (two bytes) to NVRAM:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    int data=0xAA55, data2;
    char *dataptr=(char *)&data;

    InitLib(); /* Initiate the 7186e library */
    WriteNVRAM(0, *dataptr); /* Write the low byte */
    WriteNVRAM(1, *dataptr+1); /* Write the high byte */
}
```

```
dataptr=(char *) &data2;
*dataptr=ReadNVRAM(0); /* Read the low byte */
(*dataptr+1)=ReadNVRAM(1); /* Read the high byte */
}
```

► For more demo program about the NVRAM and RTC, please refer to:

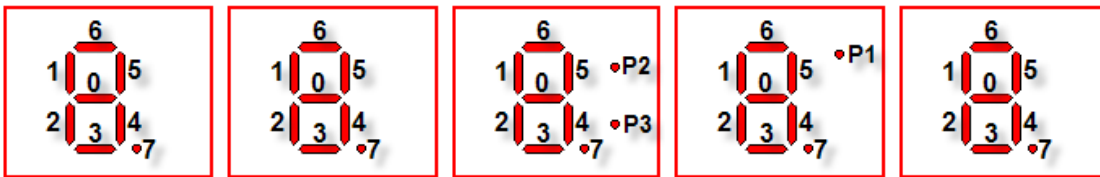
CD:\NAPDOS\7186e\Demo\Basic\memory

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/memory](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/memory)

## 4.6. API for 5-Digit LED

---

- The  $\mu$ PAC-7186EX contains a 5-Digit 7-SEG LED with a decimal point on the left-hand side of each digit, which be used to display numbers, IP addresses, time, and so on.



### API for starting the 5-Digit 7-SEG LED

---

#### 1. Init5DigitLed()

Before using any LED functions, the Init5DigitLed() must be called to initialize the 5-Digit 7-SEG LED.

### API for displaying a message on the 5-Digit 7-SEG LED

---

#### 2. Show5DigitLed()

After the Init5DigitLed() is used to initialize the 5-Digit 7-SEG LED, the Show5DigitLed() must be called to display information on the 5-Digits 7-SEG LED.

For example, use the following code to display "8000E" on the 5-Digit 7-SEG LED:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    InitLib(); /* Initiate the 7186e library */

    Init5DigitLed();

    Show5DigitLed(1,8);
    Show5DigitLed(2,0);
    Show5DigitLed(3,0);
    Show5DigitLed(4,0);
    Show5DigitLed(5,14); /* The ASCII code for the letter 'E' is 14 */
}
```

► For more demo program about the 5-digit 7-SEG LEDs, please refer to:

CD:\NAPDOS\7186e\Demo\Basic\LED\Seg7led

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/led/seg7led](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/led/seg7led)

## 4.7. API for Timer

---

- The  $\mu$ PAC-7186EX can support a single main time tick, 8 stop watch timers and 8 count down timers.
- The  $\mu$ PAC-7186EX uses a single 16-bit timer to perform these timer functions, with a timer accuracy of 1 ms..

### API for starting the Timer

---

#### 1. TimerOpen()

Before using the Timer functions, the TimerOpen() must be called at the beginning of the program.

### API for reading the Timer

---

#### 2. TimerResetValue()

Before reading the Timer, the TimerResetValue() must be called to reset the main time ticks to 0.

#### 3. TimerReadValue()

After the TimerResetValue() has reset the main time ticks to 0, the TimerReadValue() must be called to read the main time tick.



## API for stopping the Timer

---

### 4. TimerClose()

Before ending the program, the TimerClose() must be called to stop the Timer.

For example, the following code can be used to read the main time ticks from 0:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    Unsigned long time iTime;

    InitLib(); /* Initiate the 7186e library */
    TimerOpen();
    While(! quit)
    {
        If(Kbhit())
            TimerResetValue(); /* Reset the main time ticks to 0 */

        iTime=TimerReadValue(); /* Read the main time ticks from 0 */
    }
    TimerClose(); /* Stop using the 8000e timer function */
}
```

► For more demo program about the timer, please refer to:

CD:\ NAPDOS\7186e\ Demo\Basic\timer

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/timer](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/timer)

## 4.8. API for WatchDog Timer (WDT)

---

- The  $\mu$ PAC-7186EX is equipped with MiniOS7, the small-cored operating system, MiniOS7 uses the Timer 2 (A CPU internal timer) as system Timer. It is 16-bits Timer, and generate interrupt every 1 ms. So the accuracy of system is 1 ms.
- The Watch Dog Timer is always enabled, and the system Timer ISR (Interrupt Service Routine) refresh it.
- The system is reset by WatchDog. The timeout period of WatchDog is 0.8 seconds.

### API for refreshing WDT

---

#### 1. EnableWDT()

The WDT is always enabled, before user's programming to refresh it, the EnableWDT() must be called to stop refreshing WDT.

#### 2. RefreshWDT()

After EnableWDT() stop refreshing WDT, the RefreshWDT() must be called to refresh the WDT.

#### 3. DisableWDT()

After user's programming to refresh WDT, the DisableWDT() should be called to automatically refresh the WDT.

For example, to refresh the Watchdog Timer:

```
#include <stdio.h>
#include "7186e.h"

void main(void)
{
    Unsigned long time iTime;

    InitLib(); /* Initiate the 7186e library */
    Enable WDT();
    While(! quit)
    {
        RefreshWDT();
        User_function();
    }
    DisableWDT();
}
```

► For more demo program about the WatchDog Timer, please refer to:

CD:\ NAPDOS\7186e\ Demo\Basic\Misc

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/misc](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/misc)

## 4.9. API for MiniOS7 File System (For $\mu$ PAC-7186EX-FD series only)



➤ The  $\mu$ PAC-7186EX-FD series products equips an extra 64MB flash memory, the MFS is designed to read/write file from/to the 64MB flash memory.

➤ For full usage information regarding the hardware supported, application, and the specification, please refer to section

“Appendix D. What is MiniOS7 File System (MFS)” .

➤ Summarize of the MFS functions:

Function	Description
mfs_Init	Initialize the file system.
mfs_Stop	Allocated buffers are freed upon closing.
mfs_ResetFlash	Initialize the file system. All files will lose.
mfs_X600Fs_GetLibVersion	Gets the version number of function library.
mfs_GetLibDate	Gets the create date of function library.
mfs_GetFileNo	Gets the total number of files stored in the NAND Flash.
mfs_GetFreeSize	Gets the size of available space that can be used to append file.
mfs_GetBadSize	Gets the size of non-available space.
mfs_GetUsedSize	Gets the size of used space.
mfs_GetFileSize	Gets the size of file stored in the NAND Flash.
mfs_GetFileInfoByName	Uses the specified filename to retrieve file information.
mfs_GetFileInfoByNo	Uses the file number index to retrieve file information.

Function	Description
<b>mfs_DeleteAllFiles</b>	Delete all files stored in the NAND Flash.
<b>mfs_DeleteFile</b>	Delete one selected file that has been written to the NAND Flash.
<b>mfs_OpenFile</b>	1. Opens a file with a file name. 2. Creates a new file..
<b>mfs_CloseFile</b>	Closes a file with a file handle. All buffers associated with the stream are flushed before closing.
<b>mfs_ReadFile</b>	Reads a specified bytes of data from a file.
<b>mfs_WriteFile</b>	Appends a specified bytes of data to a file.
<b>mfs_Getc</b>	Gets a character from a file.
<b>mfs_Putc</b>	Outputs a character data to the file.
<b>mfs_Gets</b>	Gets a string from a file.
<b>mfs_Puts</b>	Outs a string a file.
<b>mfs_EOF</b>	Macro that tests if end-of-file has been reached on a file.
<b>mfs_Seek</b>	Repositions the file pointer of a file.
<b>mfs_Tell</b>	Returns the current file pointer.
<b>mfs_EnableWriteVerify</b>	Enable the data verification. By default, the data verification is enable.
<b>mfs_DisableWriteVerify</b>	Disable the data verification.

## API for starting 64MB flash memory

---

### 1. mfs\_Init()

Before using any MFS functions, the mfs\_Init() must be called to initialize the 64MB flash memory.

### 2. mfs\_Stop()

If the program calls the mfs\_Init() to initialize the 64MB flash memory, the mfs\_Stop() must be called to allocate buffers to free upon closing.

## API for writing/reading files from the 64MB flash memory

---

### 3. mfs\_OpenFile()

Before writing/reading data to/from the 64MB flash memory, the OpenFile() must be called to open the file.

### 4. mfs\_CloseFile()

After the data has finished being written/read to/from the 64MB flash memory, the mfs\_CloseFile() must be called to close the file with a file handle.

## API for writing data to the 64MB flash memory

---

### 5. mfs\_Puts()

After using the mfs\_OpenFile() to open the file, the FlashRead() must be called to read data from the Flash Memory.

For example, writing data to the 64MB flash memory:

```
#include <stdio.h>
#include "7186e.h"
#include "MFS.h"

#define _DISK_A 0
#define _DISK_B 1

int main(void)
{
    int iFileHandle, iRet;

    InitLib(); /* Initiate the 7186e library */
    iRet=mfs_Init();
    if(iRet<=0) return;

    iFileHandle=mfs_OpenFile(_DISK_A," Test.txt" ," w" );
    if(iFileHandle>0)
    {
        Print( "Write string to Test.txt..." );
    }
}
```



```

    mfs_Puts(iFileHandle," test mfs on 64MB flash" );
    mfs_CloseFile(iFileHandle);
    Print( "done" );
}
else
    Print( "Open file error\n\r" );
mfs_Stop();
return;
}

```

## API for reading data from the 64MB flash memory

---

### 6. mfs\_Gets()

After using the mfs\_OpenFile() to open the file, the mfs\_Gets() must be called to read data from the 64MB flash memory.

For example, reading data from the 64MB flash memory:

```

#include <stdio.h>
#include "7186e.h"
#include "MFS.h"

#define_DISK_A 0
#define_DISK_B 1

```

```

int main(void)
{
    int iFileHandle, iRet;

    InitLib(); /* Initiate the 7186e library */
    iRet=mfs_Init();
    if(iRet<=0) return;

    iFileHandle=mfs_OpenFile(_DISK_A," Test.txt" ," r" );
    if(iFileHandle>0)
    {
        Print( "Read from Test.txt...\n\r" );
        iRet=mfs_Gets(iFileHandle,Data, 80); //max length is 80 bytes.
        if(iRet>0) Print( "Data=%s\n\r" ,Data);

        mfs_CloseFile(iFileHandle);
        Print( "done" );
    }
    else
        Print( "Open file error\n\r" );
    mfs_Stop();
    return;
}

```

► For more demo program about the Flash memory, please refer to:

CD:\NAPDOS\7186e\ Demo\Basic\bc\_tc\7186fd\

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic//bc\\_tc/7186fd/](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic//bc_tc/7186fd/)

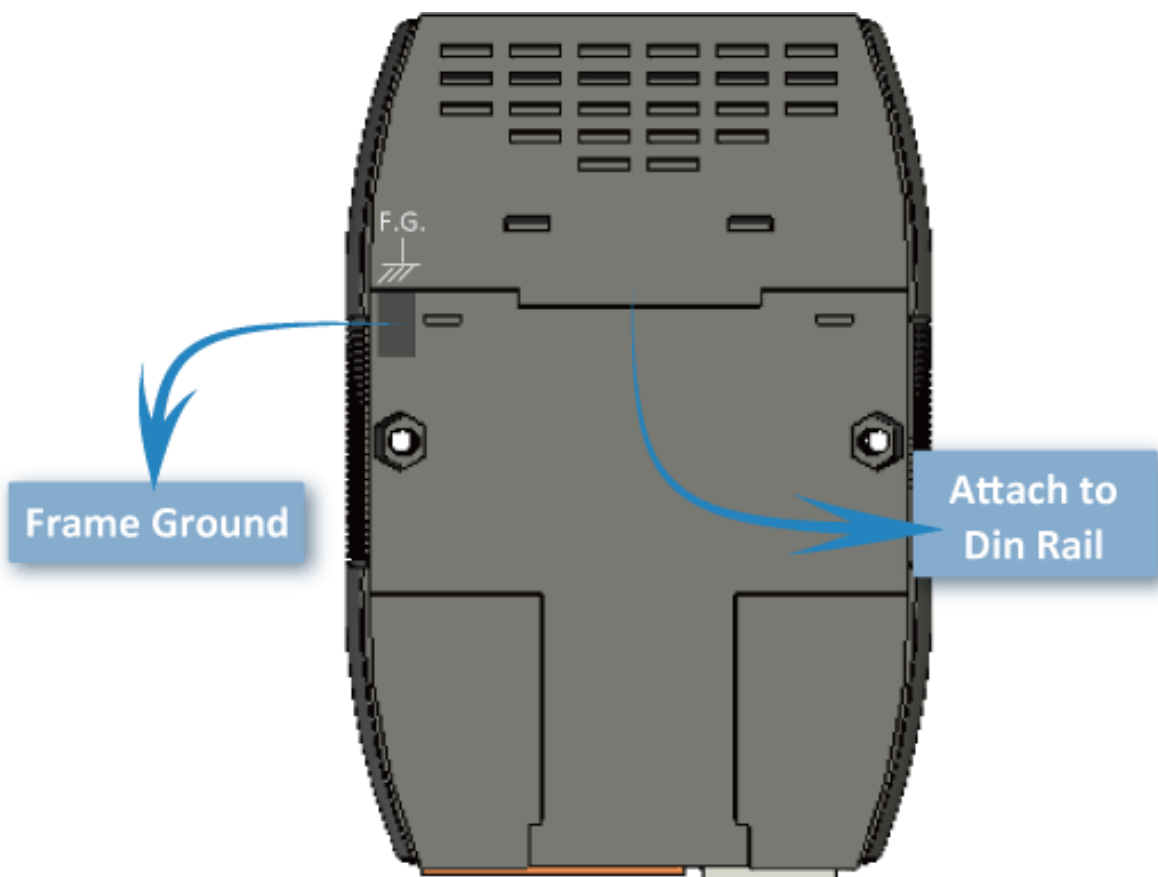
## Appendix A. Frame Ground

---

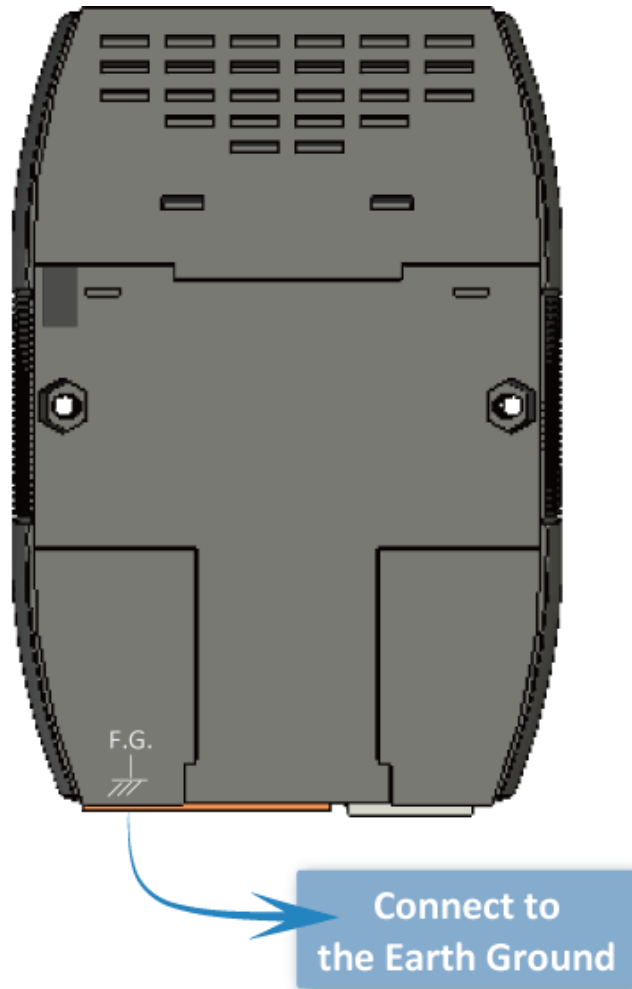
Electronic circuits are constantly vulnerable to Electro-Static Discharge (ESD), which become worse in a continental climate area. Some I-7000 ,M-7000 and I-8000 series modules feature a new design for the frame ground, which provides a path for bypassing ESD, allowing enhanced static protection (ESD) capability and ensures that the module is more reliable.

The following options will provide a better protection for the module:

The  $\mu$ PAC-7186EX controller has a metallic board attached to the back of the plastic basket as shown below.



When mounted to the DIN rail, connect the DIN rail to the earth ground because the DIN rail is in contact with the upper frame ground as shown below.



## Appendix B. What is MiniOS7

---

MiniOS7 is an embedded ROM-DOS operating system design by ICP DAS. It is functionally equivalent to other brands of DOS, and can run programs that are executable under a standard DOS.

---



DOS (whether PC-DOS, MS-DOS or ROMDOS) is a set of commands or code that tells the computer how to process information. DOS runs programs, manages files, controls information processing, directs input and output, and performs many other related functions.

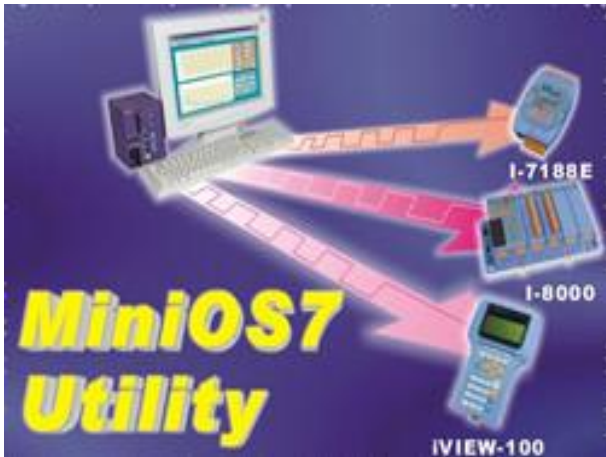
---

The following table compares the features between MiniOS7 and ROM-DOS :

Feature	MiniOS7	ROM-DOS
Power-up time	0.1 sec	4 ~ 5 sec
More compact size	< 64 K bytes	64 K bytes
Support for I/O expansion bus	Yes	No
Support for ASIC key	Yes	No
Flash ROM management	Yes	No
O.S. update (Download)	Yes	No
Built-in hardware diagnostic functions	Yes	No
Direct control of 7000 series modules	Yes	No
Customer ODM functions	Yes	No
Free of charge	Yes	No

## Appendix C. What is MiniOS7 Utility

---



MiniOS7 Utility is a tool for configuring, uploading files to all products embedded with ICPDAS MiniOS7 with easiness and quickness. Note : Since version 3.1.1, the Utility can allow users remotely access the controllers (7188E,8000E,...ect) through the Ethernet

### Functions

#### Supported connection ways

1. COM port connection (RS-232)
2. Ethernet connection (TCP & UDP)  
(Supported since version 3.1.1)

#### Maintenance

1. Upload file(s)
2. Delete file(s)
3. Update MiniOS7 image

#### Configuration

1. Date and Time
2. IP address
3. COM port
4. Disk size (Disk A, Disk B)

#### Check product information

1. CPU type
2. Flash Size
3. SRAM Size
4. COM port number

### Including Frequently Used Tools

- a. 7188XW
- b. 7188EU
- c. 7188E
- d. SendTCP
- e. Send232
- f. VxComm Utility

### PC System Requirements

1. IBM compatible PC
2. Windows 95 /98/NT/2000/XP

### Supported Products

1. 7188XA
2. 7188XB
3. 7188XC
4. 7188EX series
5. All i-8000 series
6. iView100
7.  $\mu$ PAC-7186EX
8. ET-6000 series
9. ET-7000 series

### Download location :

[http://ftp.icpdas.com.tw/pub/cd/8000cd/napdos/minios7/utility/minios7\\_utility/](http://ftp.icpdas.com.tw/pub/cd/8000cd/napdos/minios7/utility/minios7_utility/)

## Appendix D. What is MiniOS7 File System (MFS)

---

MiniOS7 file system, MFS, offers a rugged alternative to mechanical storage systems. Designed for NAND flash memory, MFS implements a reliable file system with C language API for embedded data logger applications on MiniOS7.

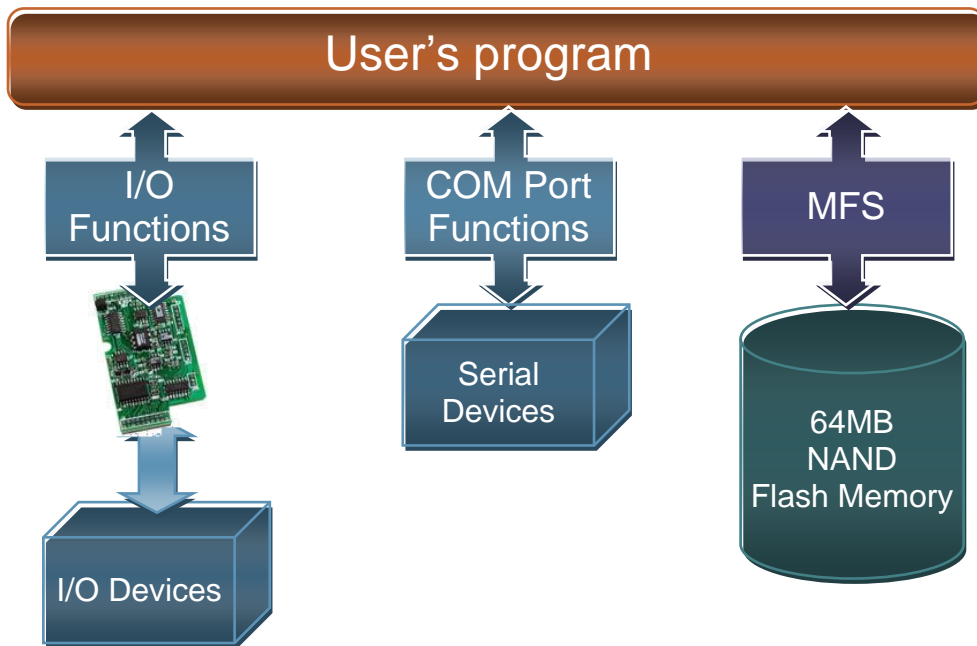


Compare to uPAC-7186EX, uPAC-7186EX-FD equips an extra 64MB flash memory. Using the MFS (MiniOS7 File System) library, you can dynamically read/write files from/to the 64MB flash memory. Based on the uPAC-7186EX-FD, many kind of applications related to data logger can be implemented. For example: log analog signal values with timestamp, log RS-232/485 communication data for analysis.

➤ **Applications.**

Log data with timestamp,

Log data and forward via the Ethernet



➤ **Hardware Supported**

μPAC-7186EX-FD (with 64MB Flash Memory)

**Note:** NVRAM:all of the 31 bytes.



➤ **MFS Specifications.**

Item	Description
Disk number	2 (disk A and B)
Disk size	1/2 size of the flash memory size
File number	456 files max. for each disk
File size	64MB max. for each file
File name	12 bytes max (case sensitive)
File operation modes	<p>Read:</p> <p>Write: Creates a new file to write data, or overwrite a file (if the file is already exist).</p> <p>Append: appends data to a file.</p>
File handle	<p>10 max. for each disk.</p> <p>For read mode: the 10 file handles can all be used for reading operation on each disk. Total 20 files can be opened for reading mode.</p> <p>For write and append mode: only 1 file handle can be used for writing operation on all disks.</p>
Writing verification	<p>Yes. Default is enabled.</p> <p>Calling <code>mfs_EnableWriteVerification</code> and <code>mfs_DisableWriteVerification</code> can change the setting.</p>
Automate file system recovery	<p>Yes.</p> <p>If an unexpected reset or power loss occurs, closed files, and files opened for reading are never at risk. Only data written since the last writing operation (<code>mfs_WriteFile</code>, ) might be lost. When the file system reboots, it restores the file system to its state at the time of the last writing operation.</p>

Item	Description
Writing speed	mfs_WriteFile : 147.5 KB/Sec (verification enabled) (default) 244.0 KB/Sec (verification disabled)  mfs_Puts: 142.1 KB/Sec (verification enabled) (default) 229.5 KB/Sec (verification disabled)
Reading speed	mfs_ReadFile: 734.7 KB/Sec mfs_Gets: 414.2 KB/Sec
Max. length of writing data	32767 bytes.
Max. length of reading data	32767 bytes.

The latest version of the MFS SDKs can be obtain from:

CD:\ NAPDOS\7186e\Demo\Basic\BC\_TC\Lib

[http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc\\_tc/lib](http://ftp.icpdas.com/pub/cd/8000cd/napdos/7186e/demo/basic/bc_tc/lib)

## Appendix E. What is VxComm Utility

---



The VxComm Driver creates COM port(s) and maps them to the Ethernet port(s) of the PDS/8000E/7188E.

The user's RS-232 client programs need only to change to the different COM port to get the access of serial devices that are allocated in the Internet or Ethernet network via the PDS/8000E/7188E.

The VxComm Driver supports Windows NT 4.0, 2000/XP/2003 and 32-bit Vista (Vista32), and is totally free for users using ICP DAS PDS/8000E/7188E... series products.

For downloading and more information, please refer to the following link:

<http://www.icpdas.com/products/Software/VxComm/vxcomm.htm>

## Appendix F. More C Compiler Settings

---

This section describes the setting of the following compilers:

- Turbo C 2.01 Compiler
- BC++ 3.1 IDE
- MSC 6.00 Compiler
- MSVC 1.50 Compiler

### F.1. Turbo C 2.01

---

You have a couple of choices here, you can :

#### **1 : Using a command line**

For more information, please refer to

```
CD:\8000\NAPDOS\8000\841x881x\Demo\hello\Hello_C\gotc.bat
```

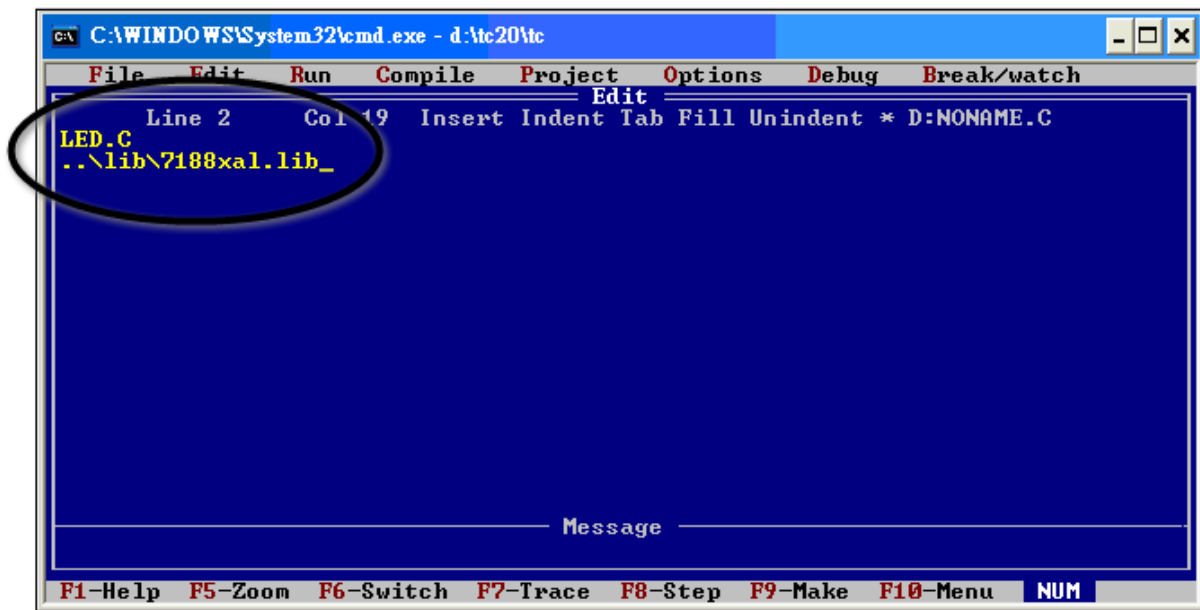
```
tcc -Ic:\tc\include -Lc:\tc\lib hello1.c ..\..\lib\8000e.lib
```

## 2 : Using the TC Integrated Environment

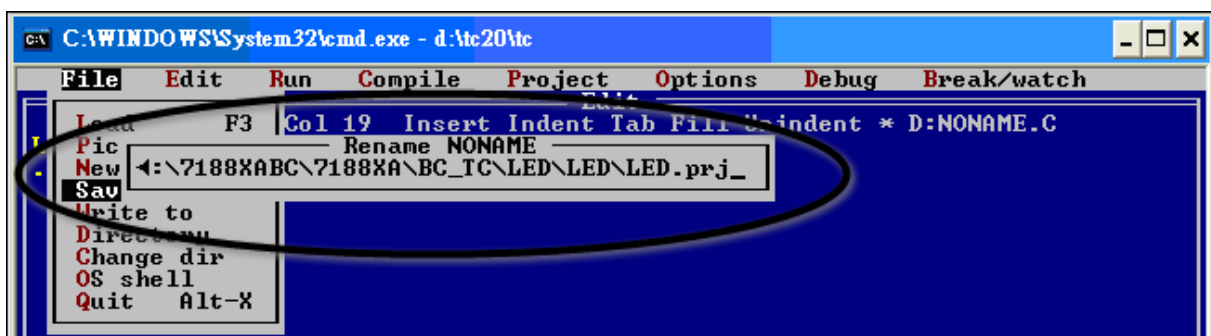
Step 1: Executing the TC 2.01

Step 2: Editing the Project file

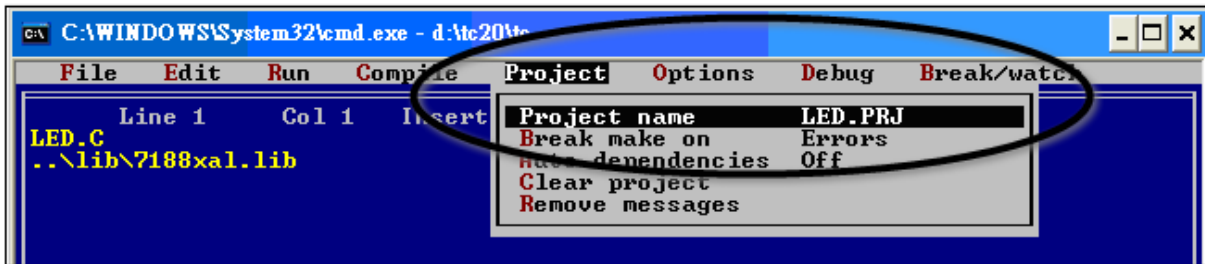
Adding the necessary library and file to the project



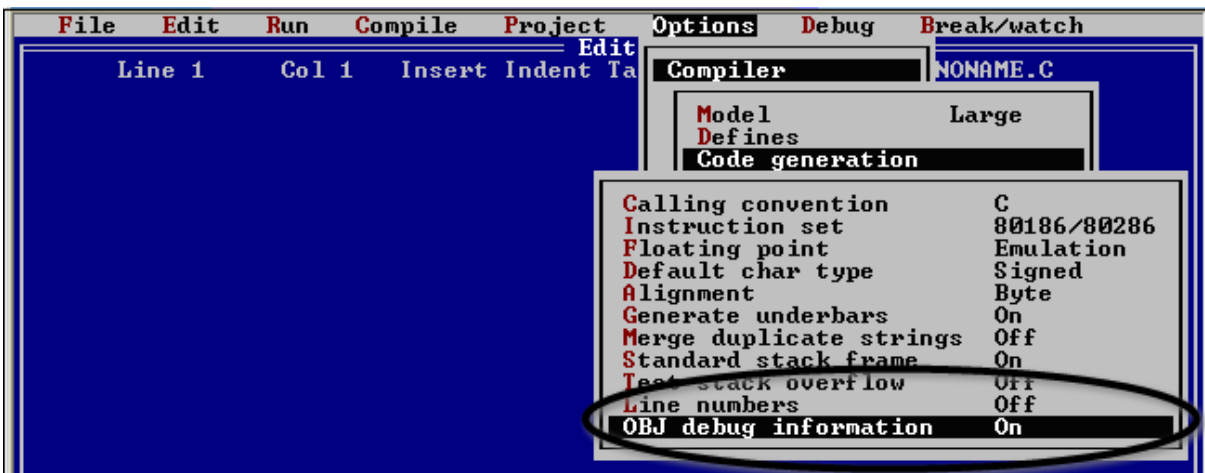
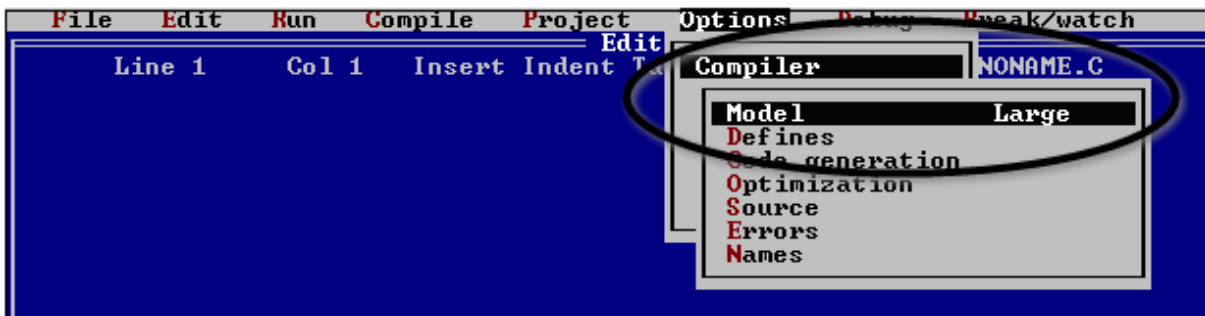
Step 3: Save the project and entering a name, such as LED.prj



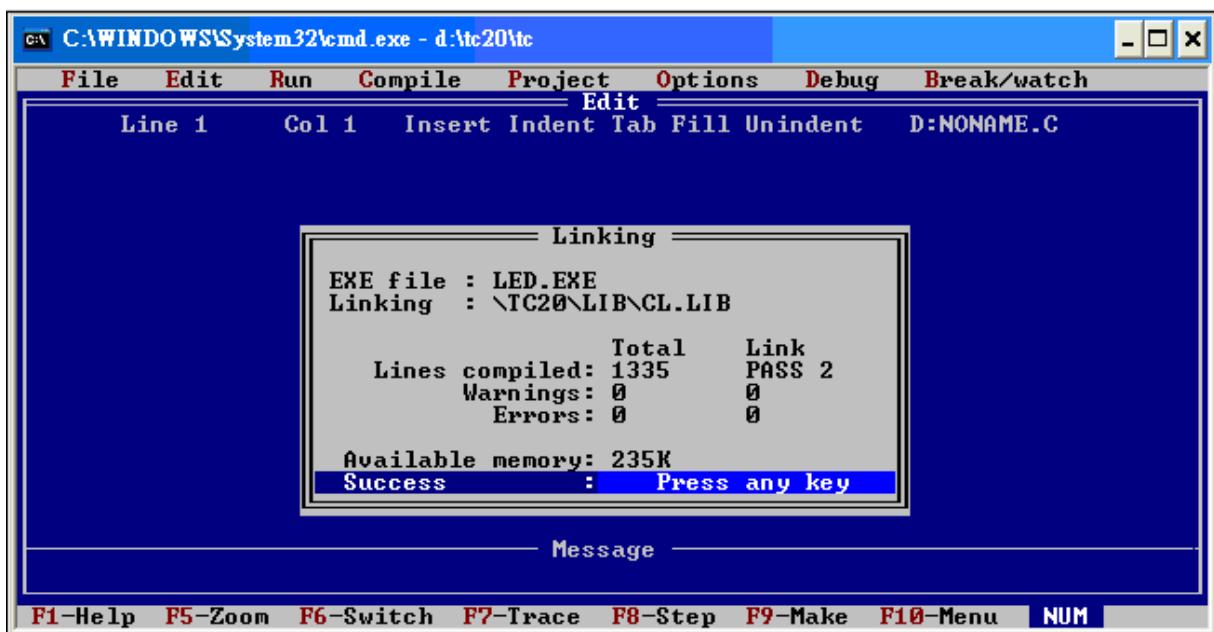
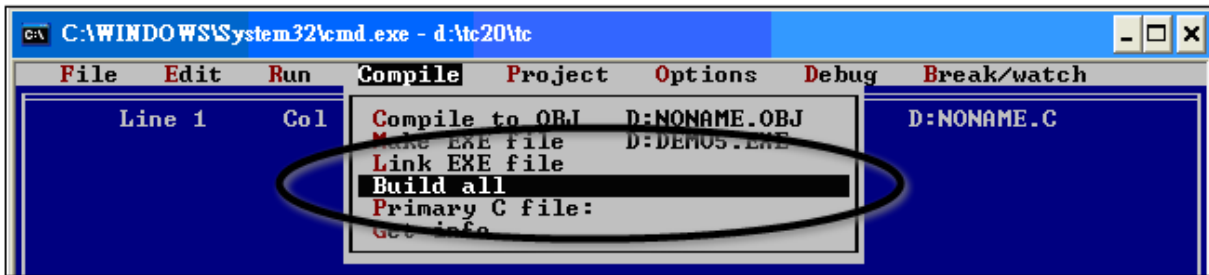
#### Step 4: Load the Project



#### Step 5: Change the Memory model (Large for 8000e.lib) and set the Code Generation to 80186/80286



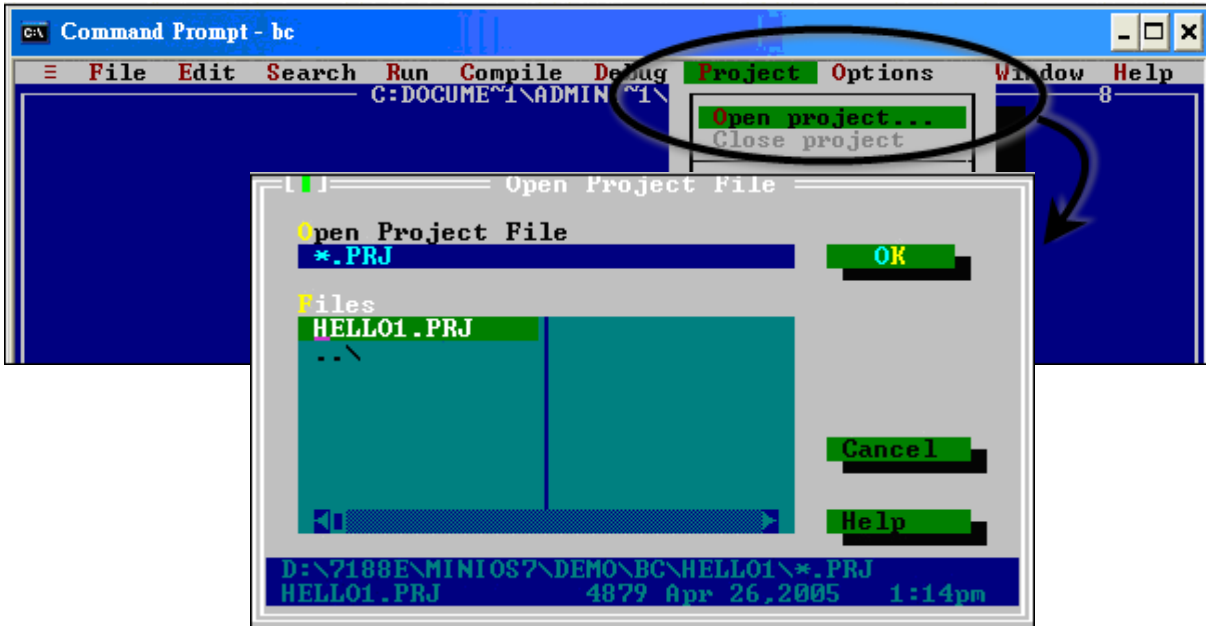
## Step 6: Building the project



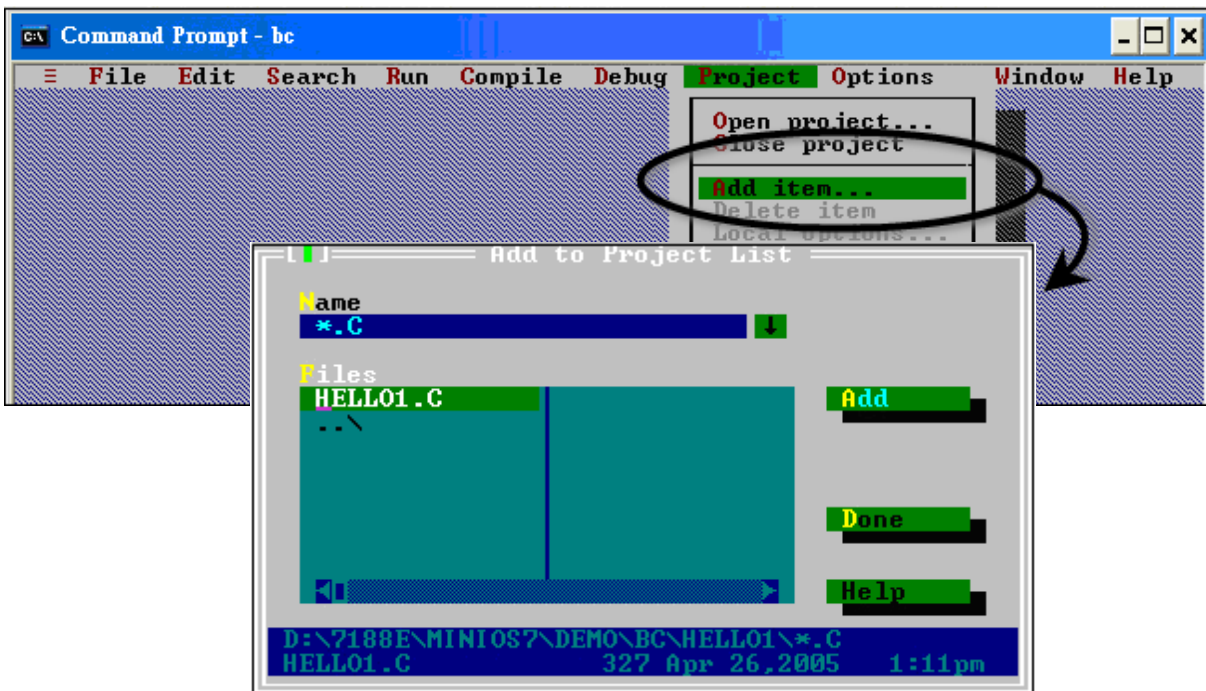
## F.2. BC++ 3.1. IDE

Step 1: Executing the Borland C++ 3.1

Step 2: Creating a new project file (\*.prj)

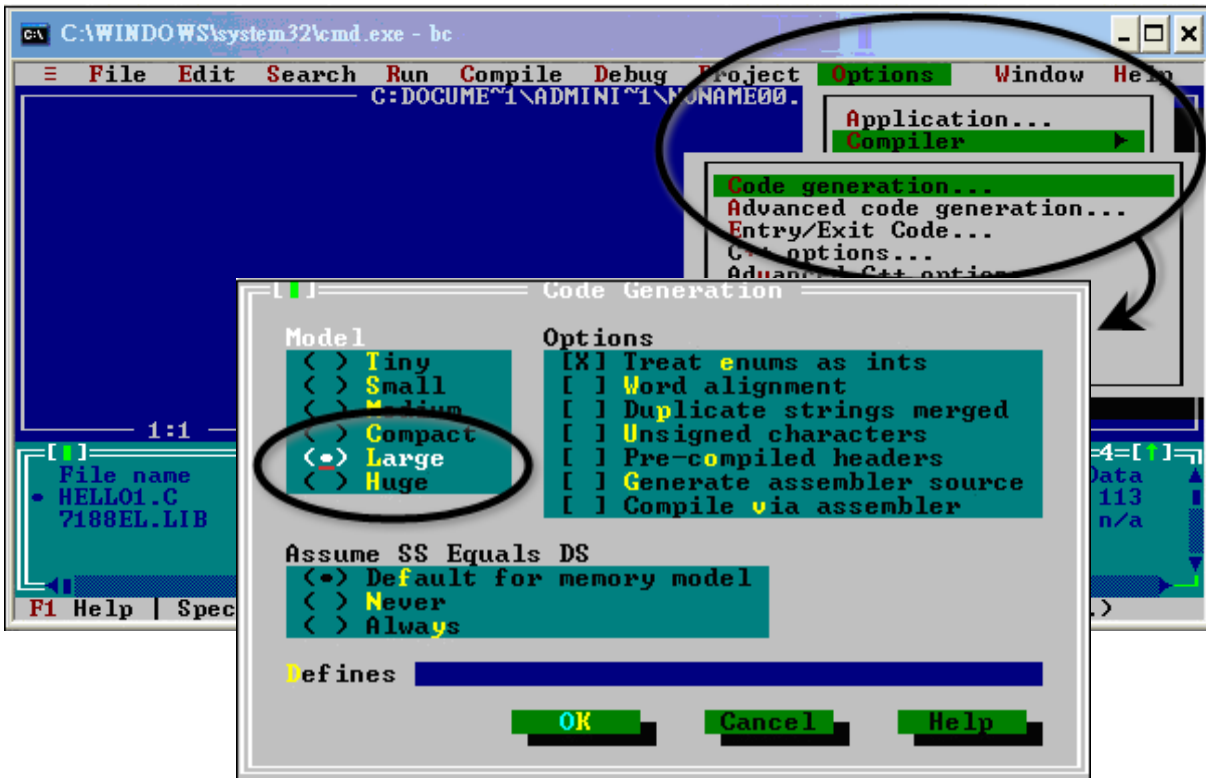


Step 3: Add all the necessary files to the project

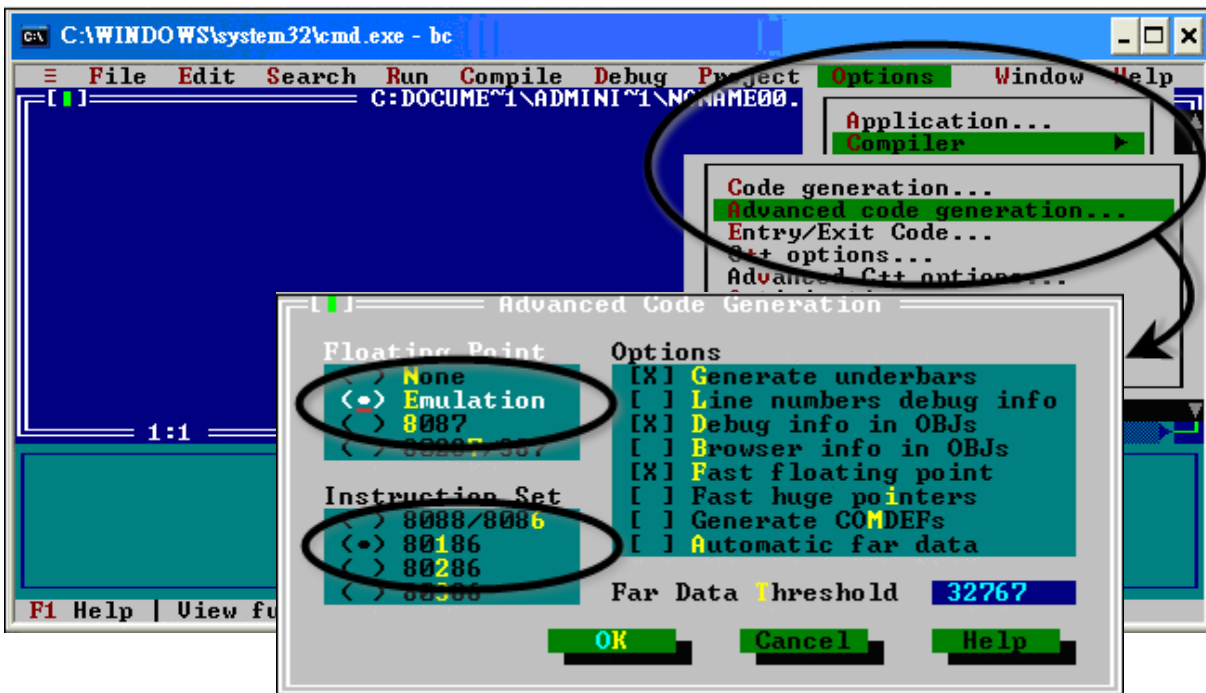




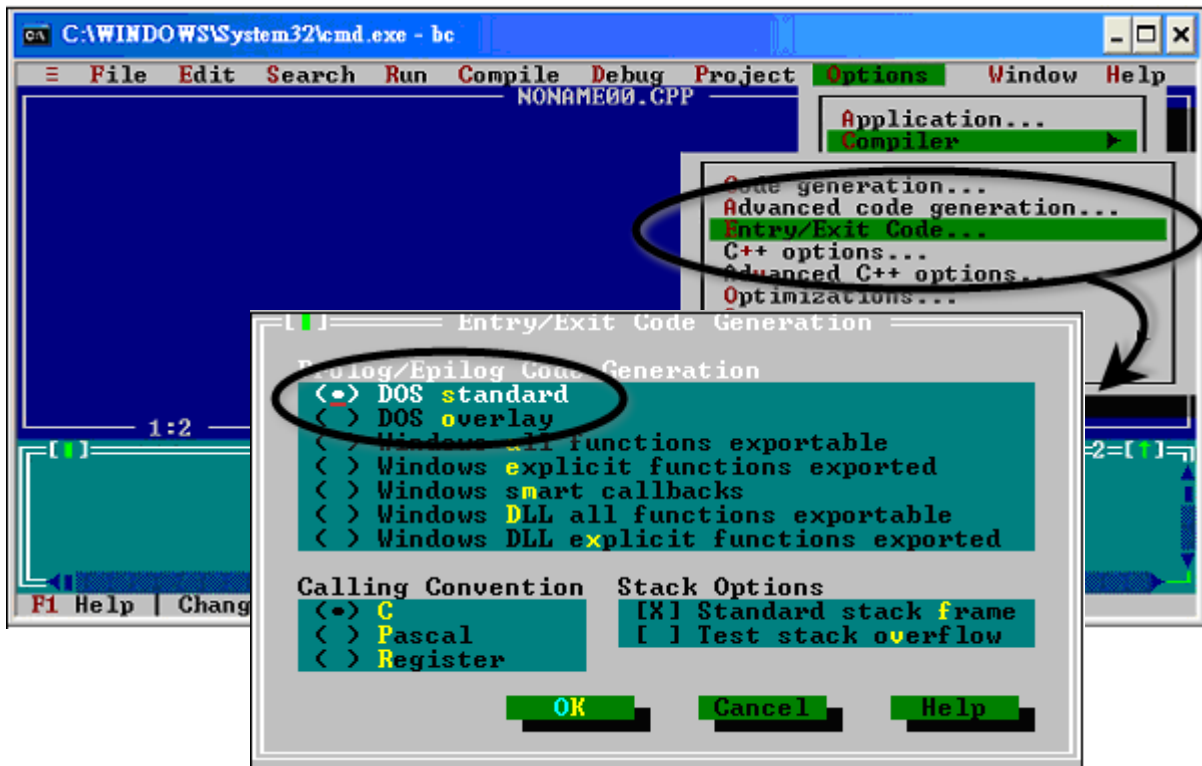
Step 4: Change the Memory model (Large for 8000e.lib)



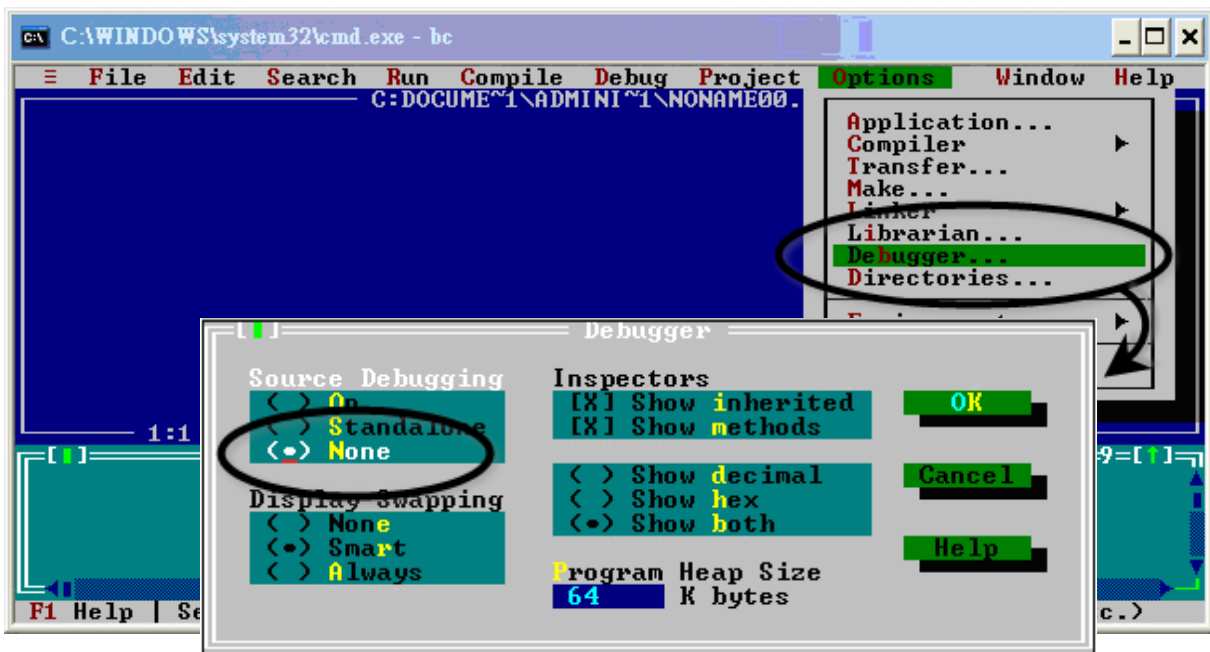
Step 5: Set the Advanced code generation options and Set the Floating Point to Emulation and the Instruction Set to 80186



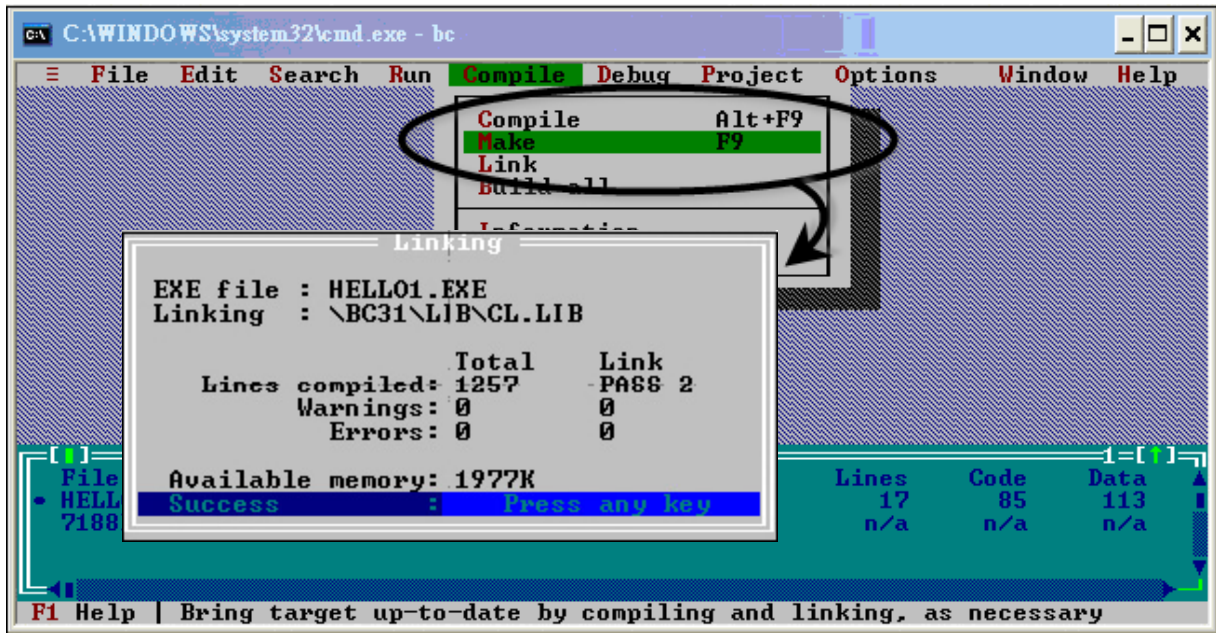
Step 6: Set the Entry/Exit Code Generation option and setting the DOS standard



Step 7: Choosing the Debugger...and set the Source Debugging to None



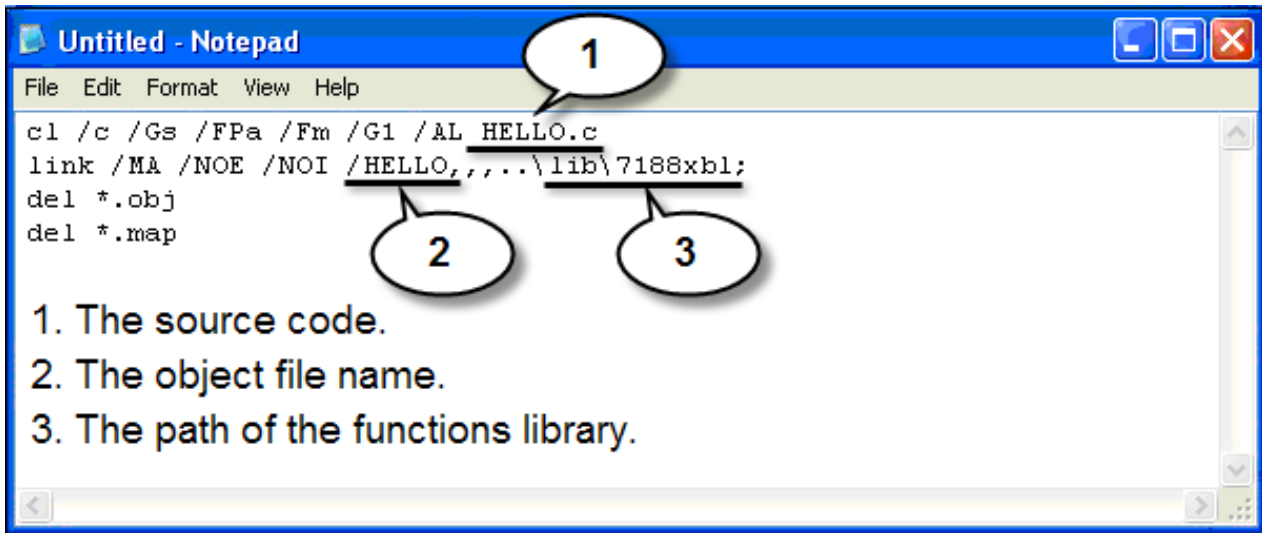
## Step 8: Make the project



## F.3. MSC 6.00

---

Step 1: In the source file folder, create a batch file called Gomsc.bat using the text editor

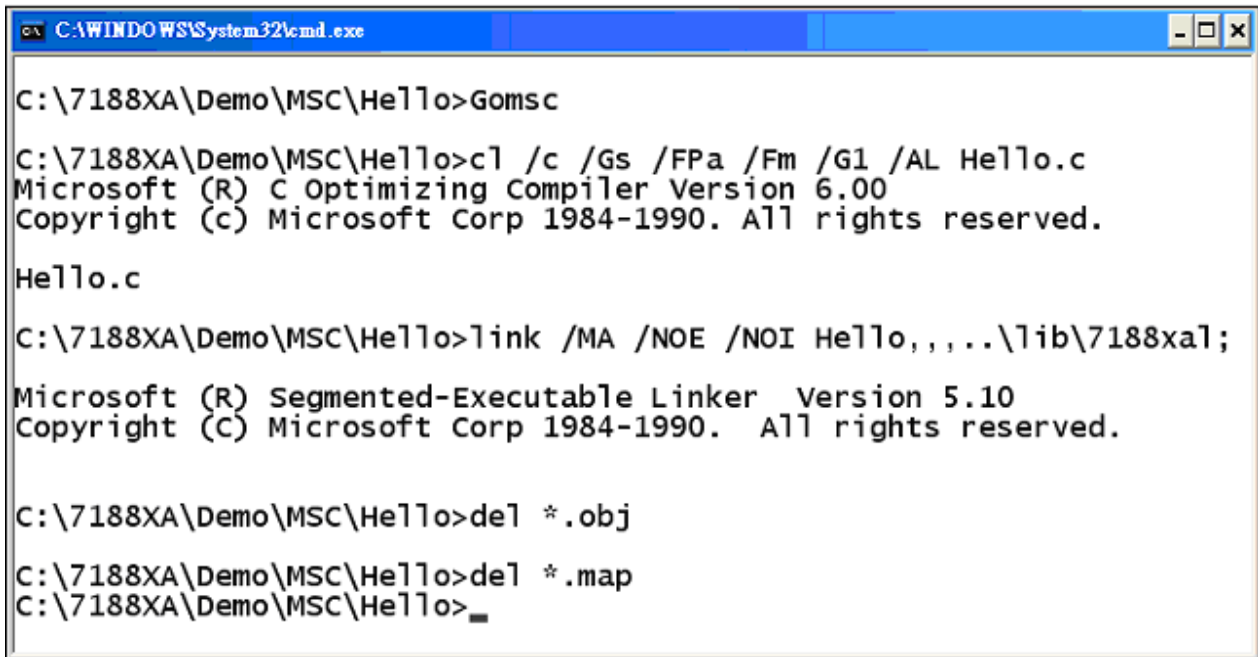


---

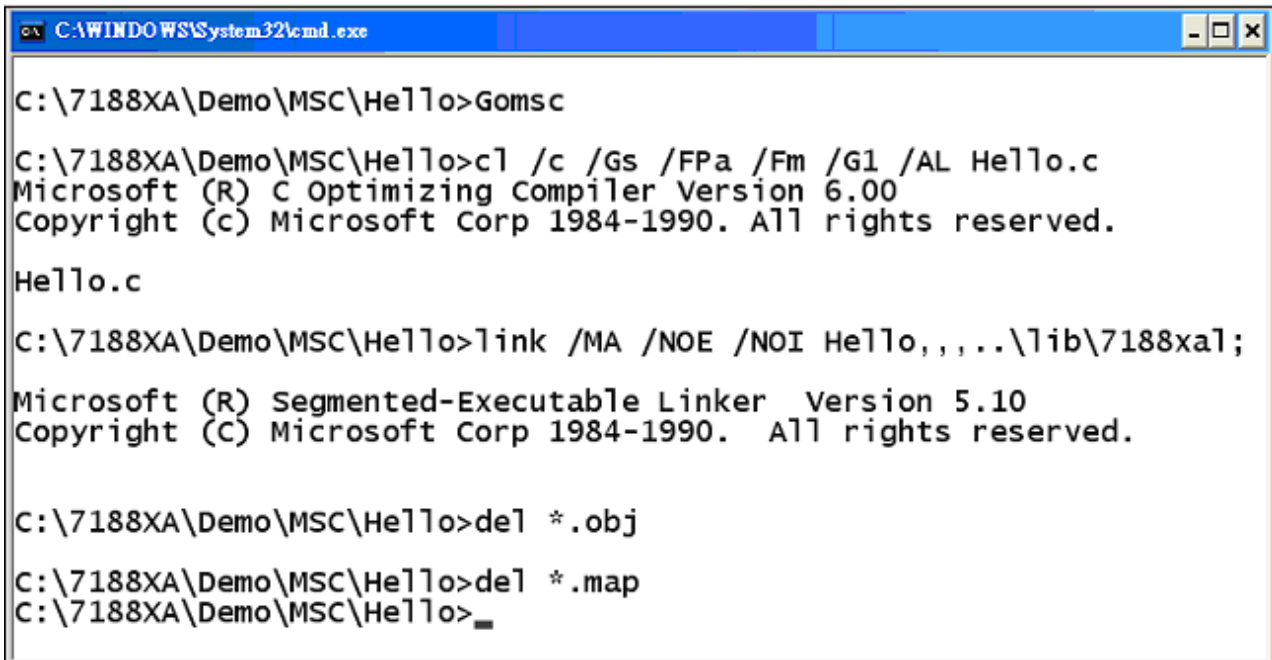
**Note:** : /C Don't strip comments      /GS No stack checking  
          /FPa : Calls with altmath        /Fm [map file]  
          /G1 : 186 instructions         /AL Large model

---

## Step 2: Run the Gomsc.bat file

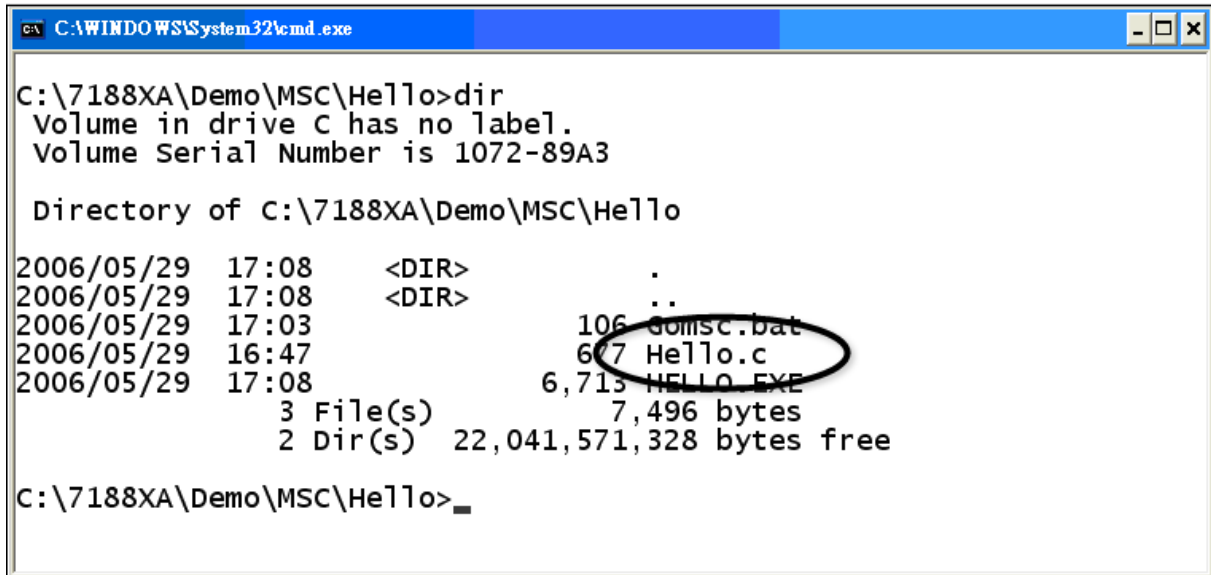


```
C:\WINDOWS\System32\cmd.exe
C:\7188XA\Demo\MSC\Hello>Gomsc
C:\7188XA\Demo\MSC\Hello>c1 /c /Gs /FPa /Fm /G1 /AL Hello.c
Microsoft (R) C Optimizing Compiler Version 6.00
Copyright (c) Microsoft Corp 1984-1990. All rights reserved.
Hello.c
C:\7188XA\Demo\MSC\Hello>link /MA /NOE /NOI Hello,,,,.\lib\7188xa1;
Microsoft (R) Segmented-Executable Linker Version 5.10
Copyright (C) Microsoft Corp 1984-1990. All rights reserved.
C:\7188XA\Demo\MSC\Hello>del *.obj
C:\7188XA\Demo\MSC\Hello>del *.map
C:\7188XA\Demo\MSC\Hello>_
```



```
C:\WINDOWS\System32\cmd.exe
C:\7188XA\Demo\MSC\Hello>Gomsc
C:\7188XA\Demo\MSC\Hello>c1 /c /Gs /FPa /Fm /G1 /AL Hello.c
Microsoft (R) C Optimizing Compiler Version 6.00
Copyright (c) Microsoft Corp 1984-1990. All rights reserved.
Hello.c
C:\7188XA\Demo\MSC\Hello>link /MA /NOE /NOI Hello,,,,.\lib\7188xa1;
Microsoft (R) Segmented-Executable Linker Version 5.10
Copyright (C) Microsoft Corp 1984-1990. All rights reserved.
C:\7188XA\Demo\MSC\Hello>del *.obj
C:\7188XA\Demo\MSC\Hello>del *.map
C:\7188XA\Demo\MSC\Hello>_
```

Step 3: A new executable file will be created if it is successfully compiled



```
C:\WINDOWS\system32\cmd.exe

C:\7188XA\Demo\MSC\Hello>dir
Volume in drive C has no label.
Volume Serial Number is 1072-89A3

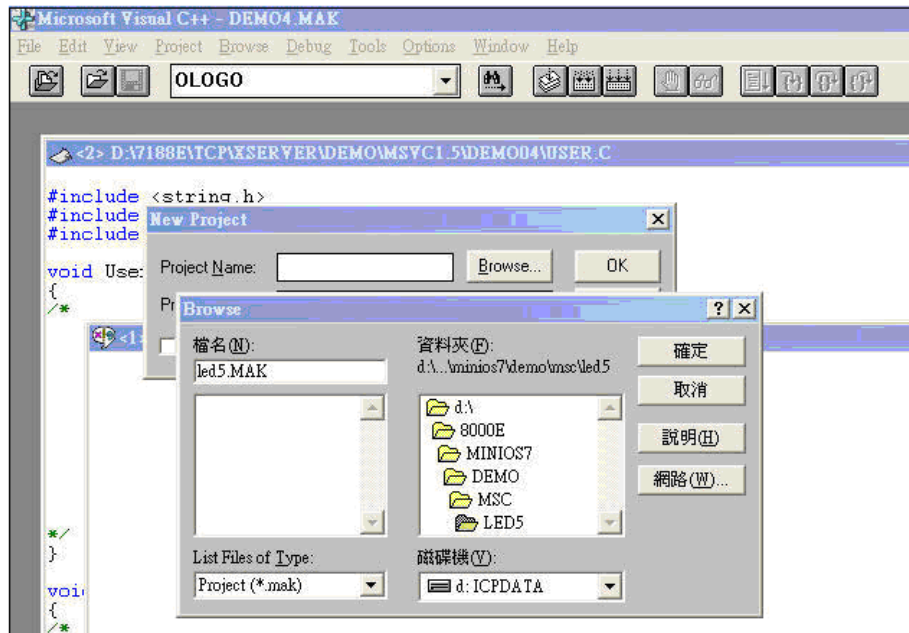
Directory of C:\7188XA\Demo\MSC\Hello

2006/05/29  17:08    <DIR>          .
2006/05/29  17:08    <DIR>          ..
2006/05/29  17:03             106 somsc.bat
2006/05/29  16:47             677 Hello.c
2006/05/29  17:08             6,713 HELLO_EXE
                3 File(s)              7,496 bytes
                2 Dir(s)  22,041,571,328 bytes free

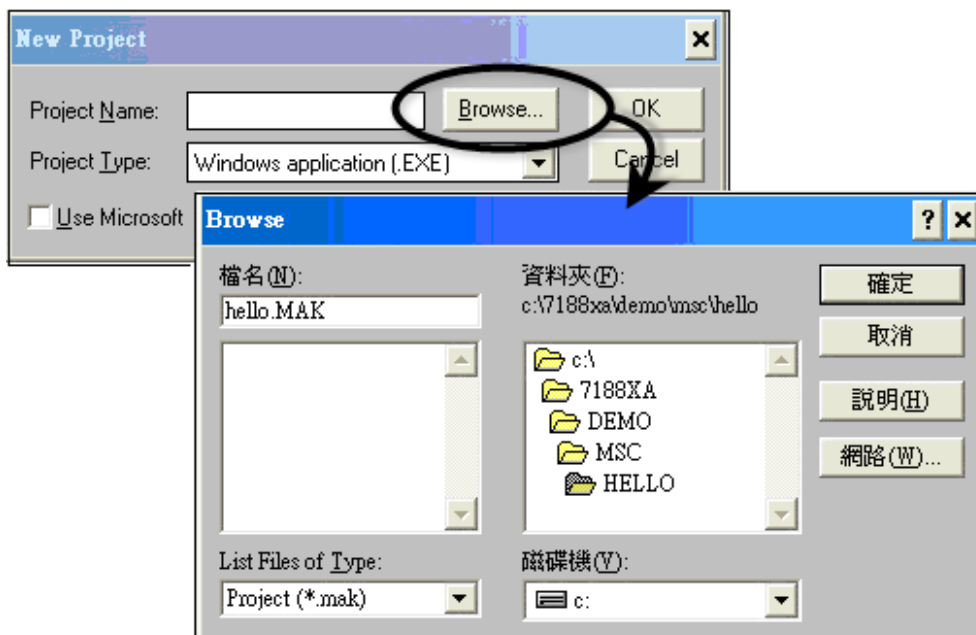
C:\7188XA\Demo\MSC\Hello>_
```

## F.4. MSVC 1.50

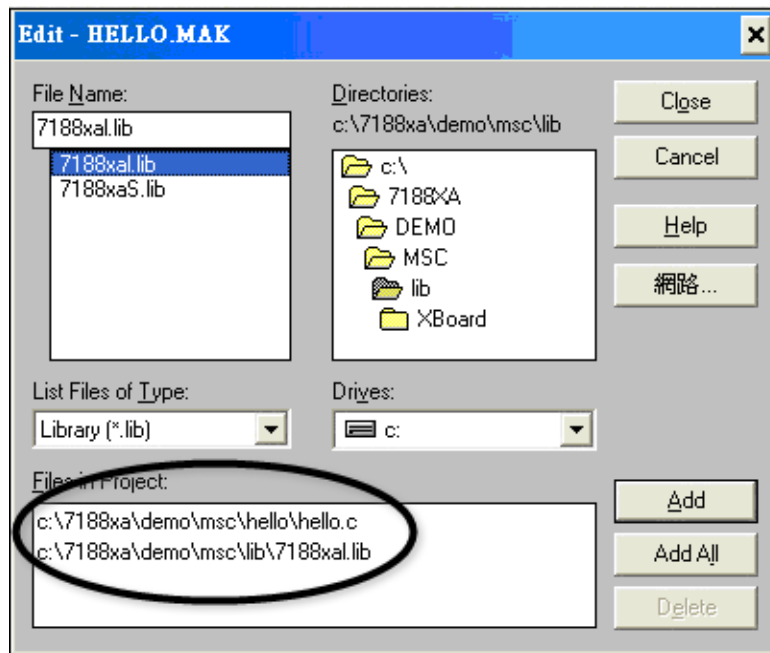
### Step 1: Run MSVC.exe



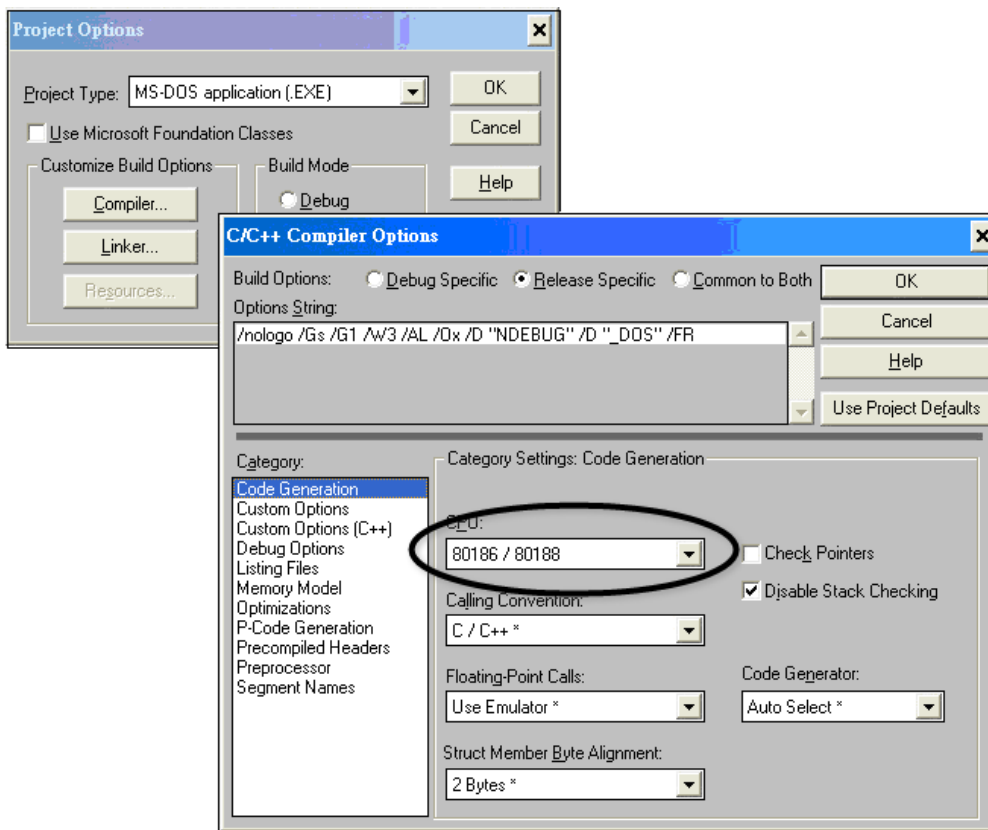
Step 2: Create a new project (\*.mak) by entering the name of the project in the Project Name field and then select MS-DOS application (EXE) as the Project type



Step 3: Add the user's program and the necessary library files to the project

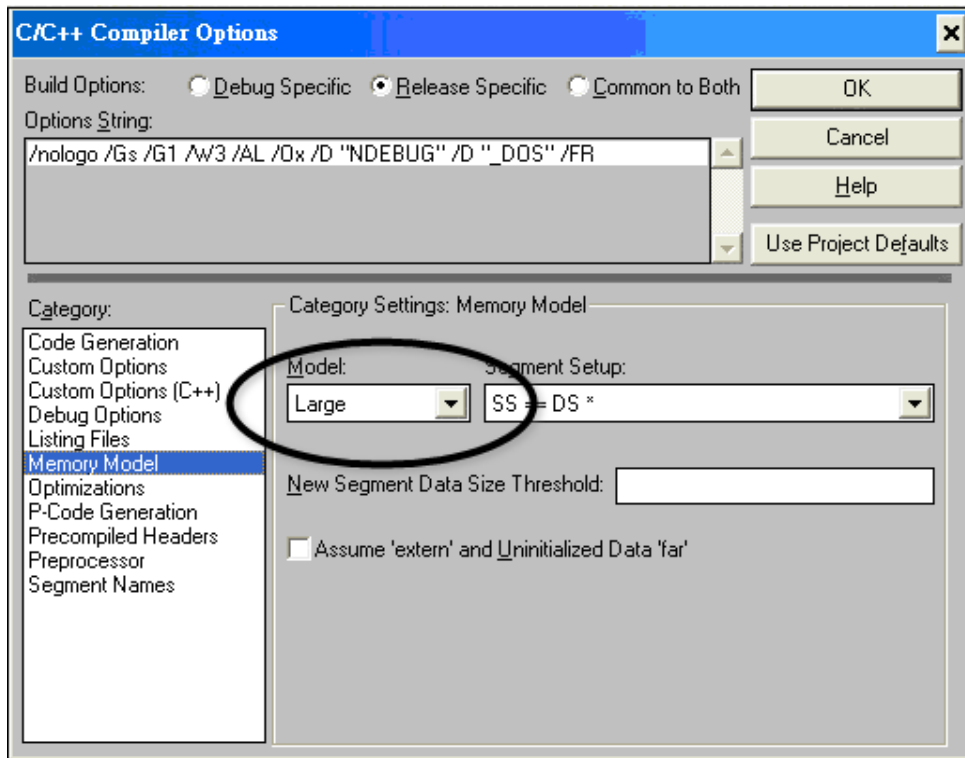


Step 4: Set the Code Generation on the Compiler.

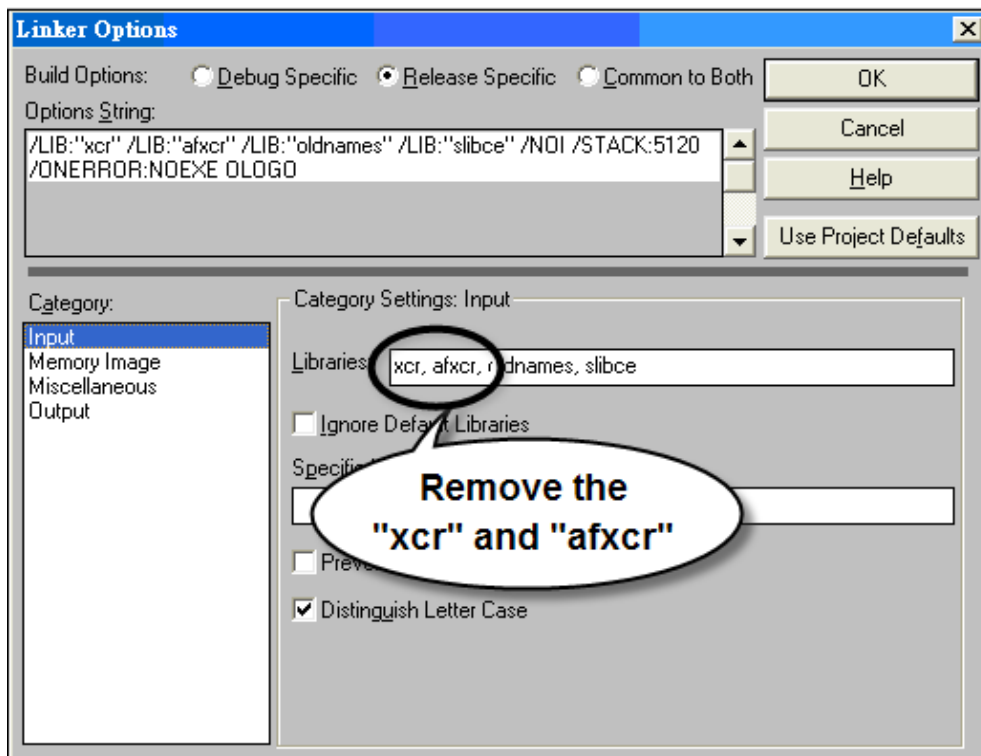




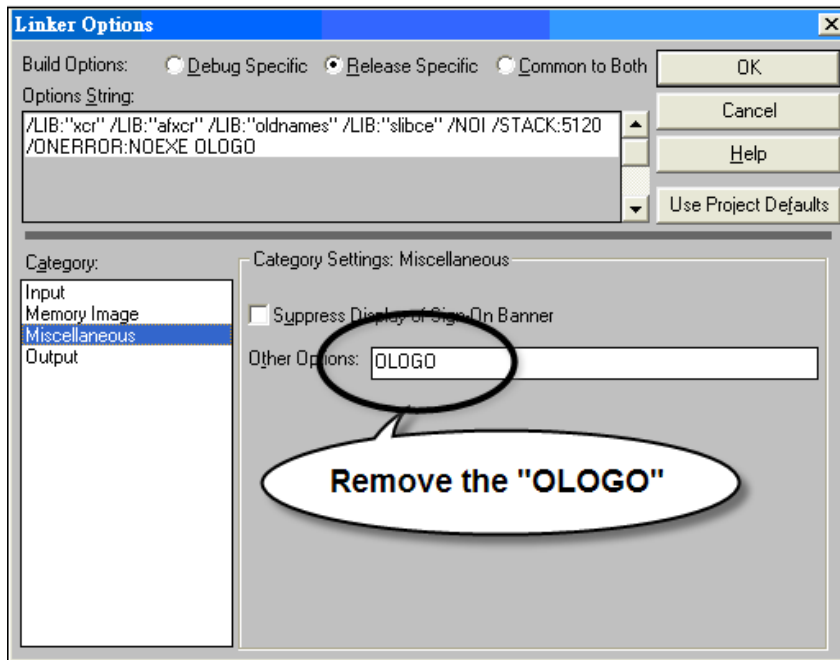
### Step 5: Change the Memory model (large for 8000e.lib)



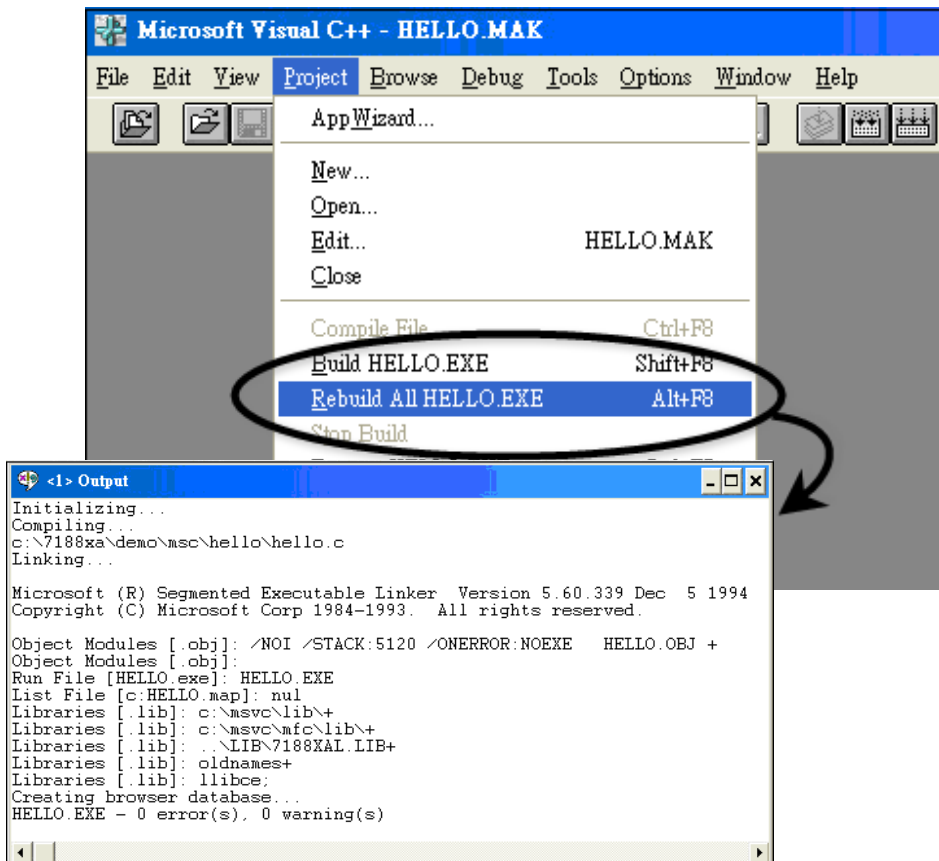
### Step 6: Remove the xcr, afxcr library from the Input Category



Step 7: Remove the OLOGO option from the miscellaneous Category.



Step 8: Rebuild the project



## Appendix G. Application of RS-485 Network

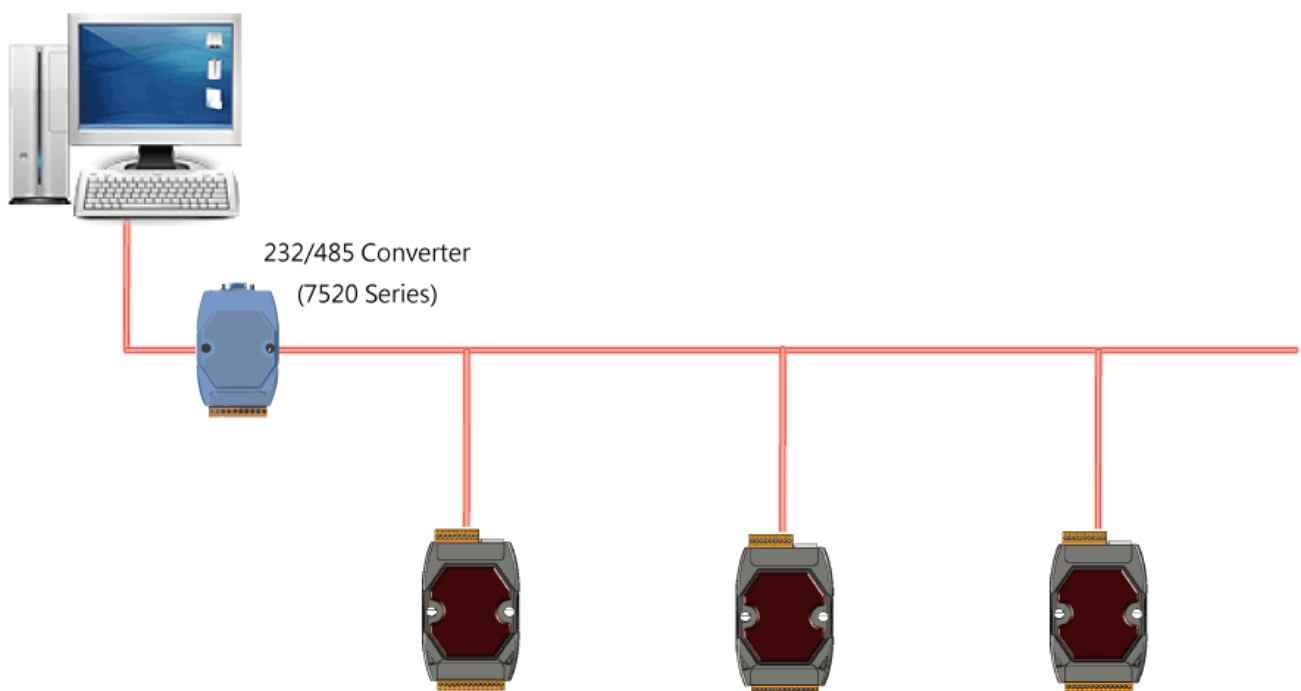
---

The RS-485 length can be up to 4000 ft or 1.2 km over a single set of twisted-pair cables, if the RS-485 network is over 4000 ft or 1.2Km, the RS-485 repeater must be added to extend the RS-485 network.

### G.1. Basic RS-485 network

---

The basic component of the RS-485 network consist of a Master Controller (or using a PC as a host controller), and some RS-485 devices.

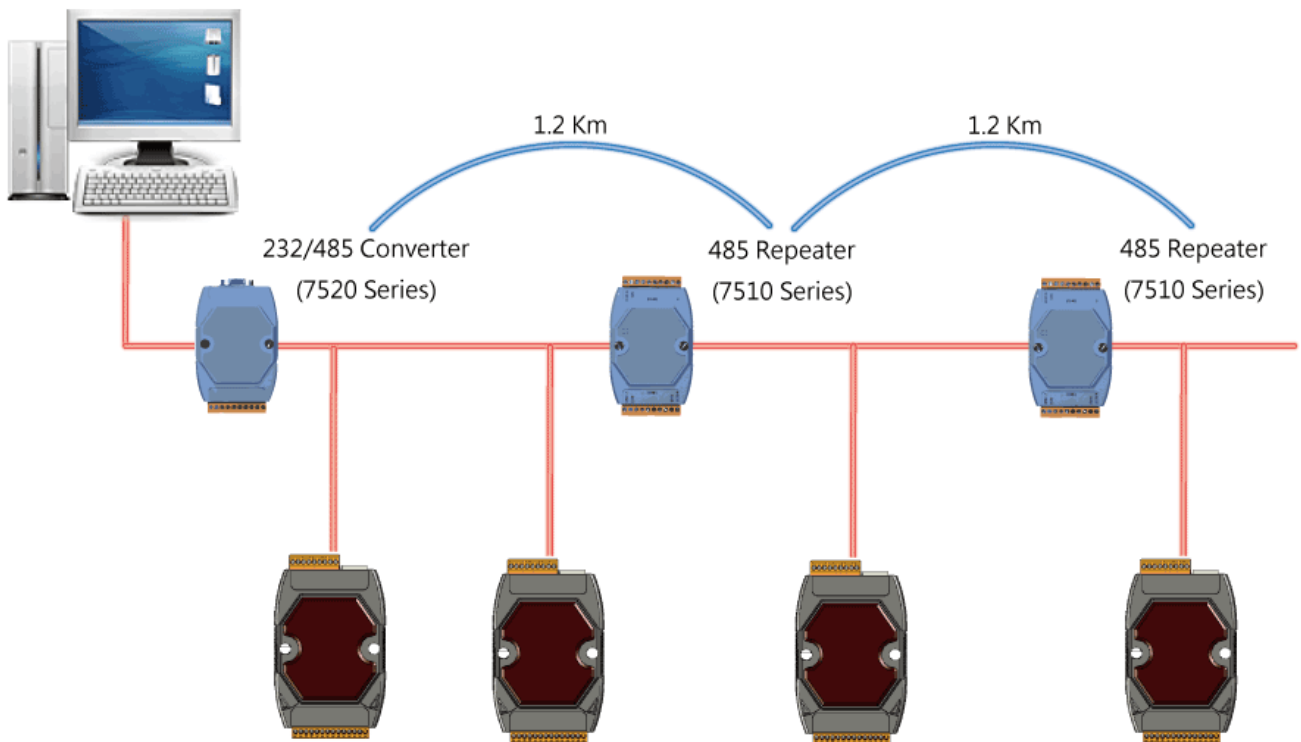


## G.2. Daisy chain RS-485 network

---

There are branches along the main network. In this case, it is better to have a repeater to isolate or filter the noise that is made by devices.

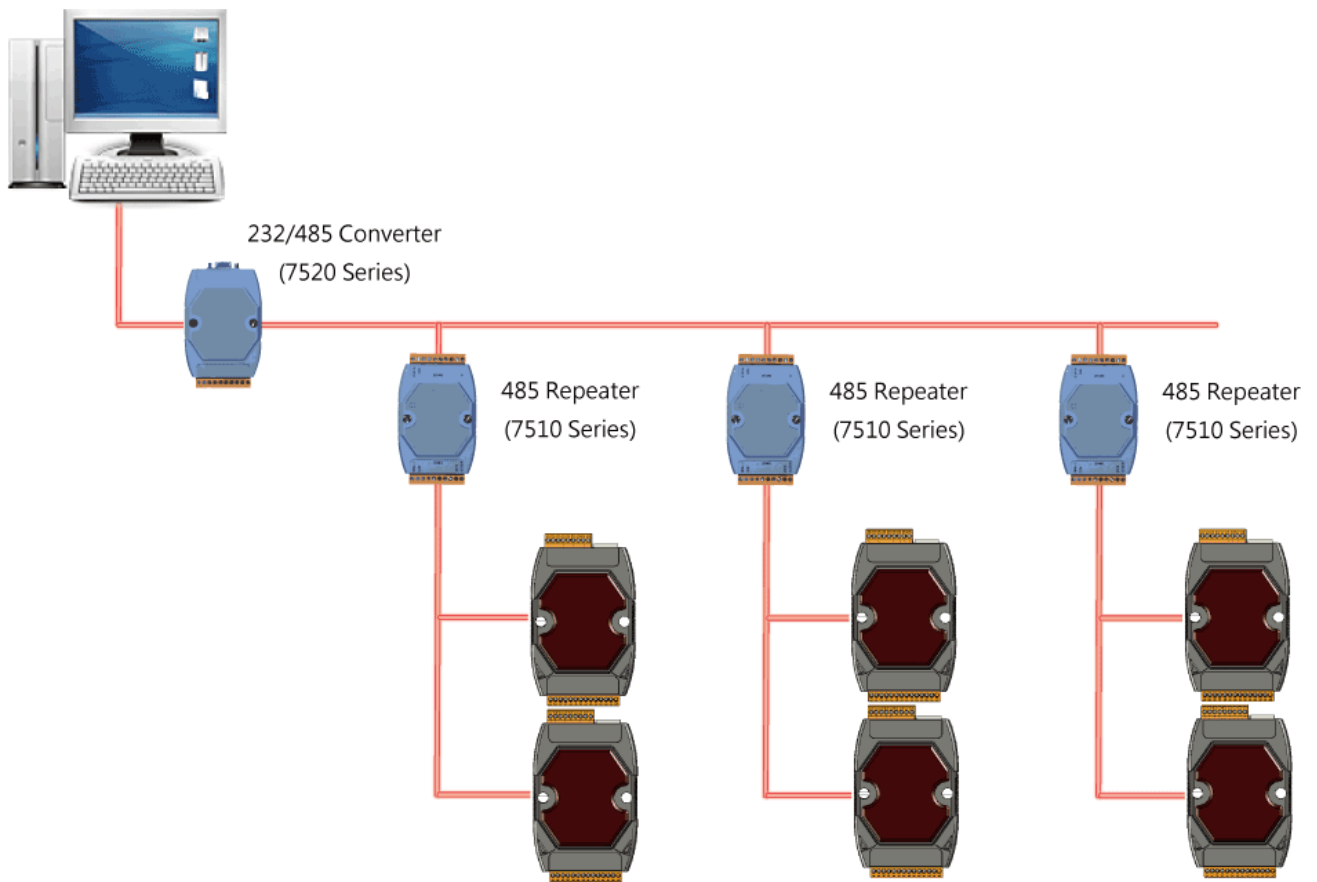
There is a better choice to use 7513 as a RS-485 hub on start type network.



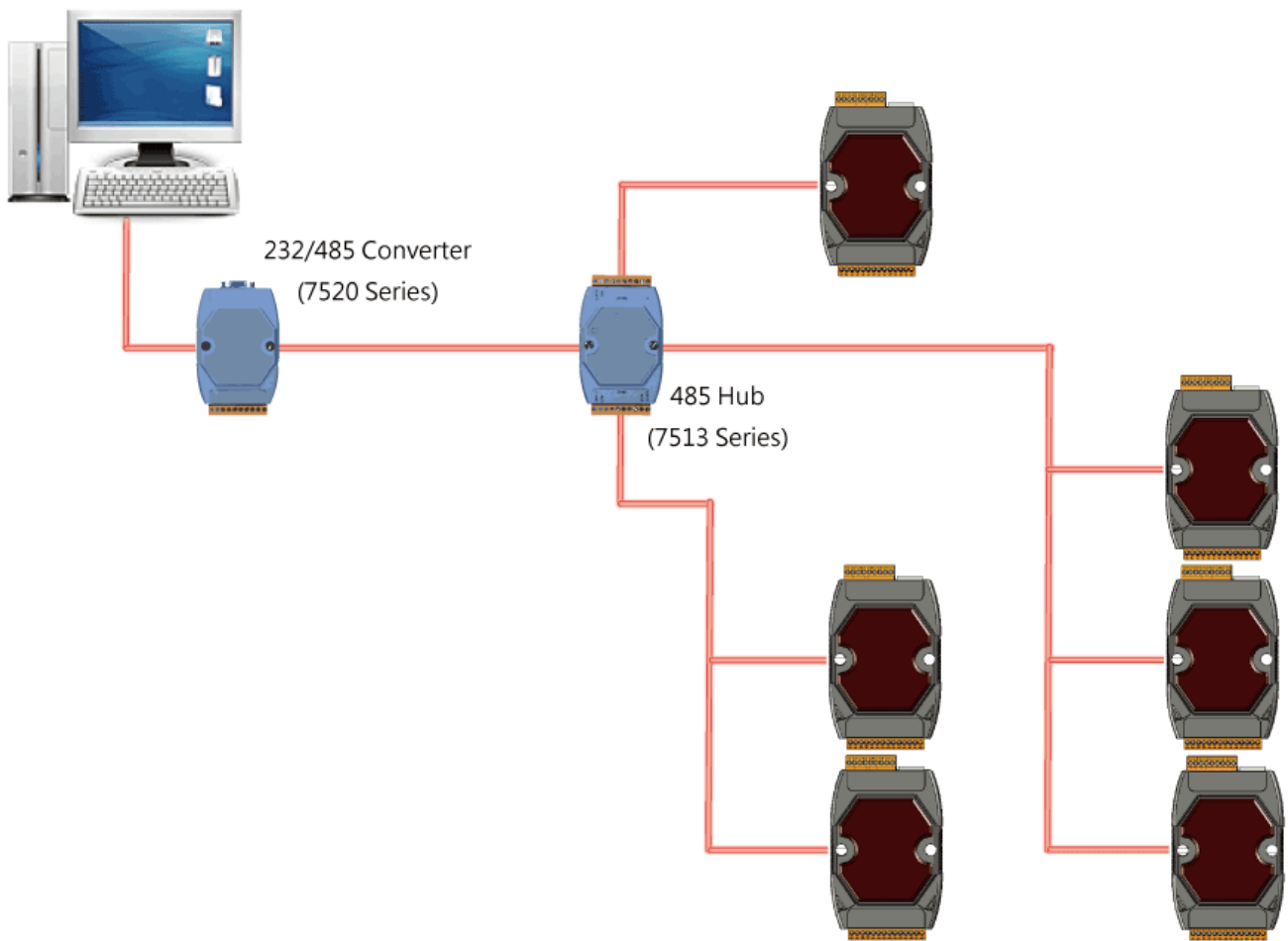
### G.3. Star type RS-485 network

---

All RS-485 devices are wired directly to the main network, If the network is up to 1.2 Km, it will need a repeater (7510 series) to extend the network length.



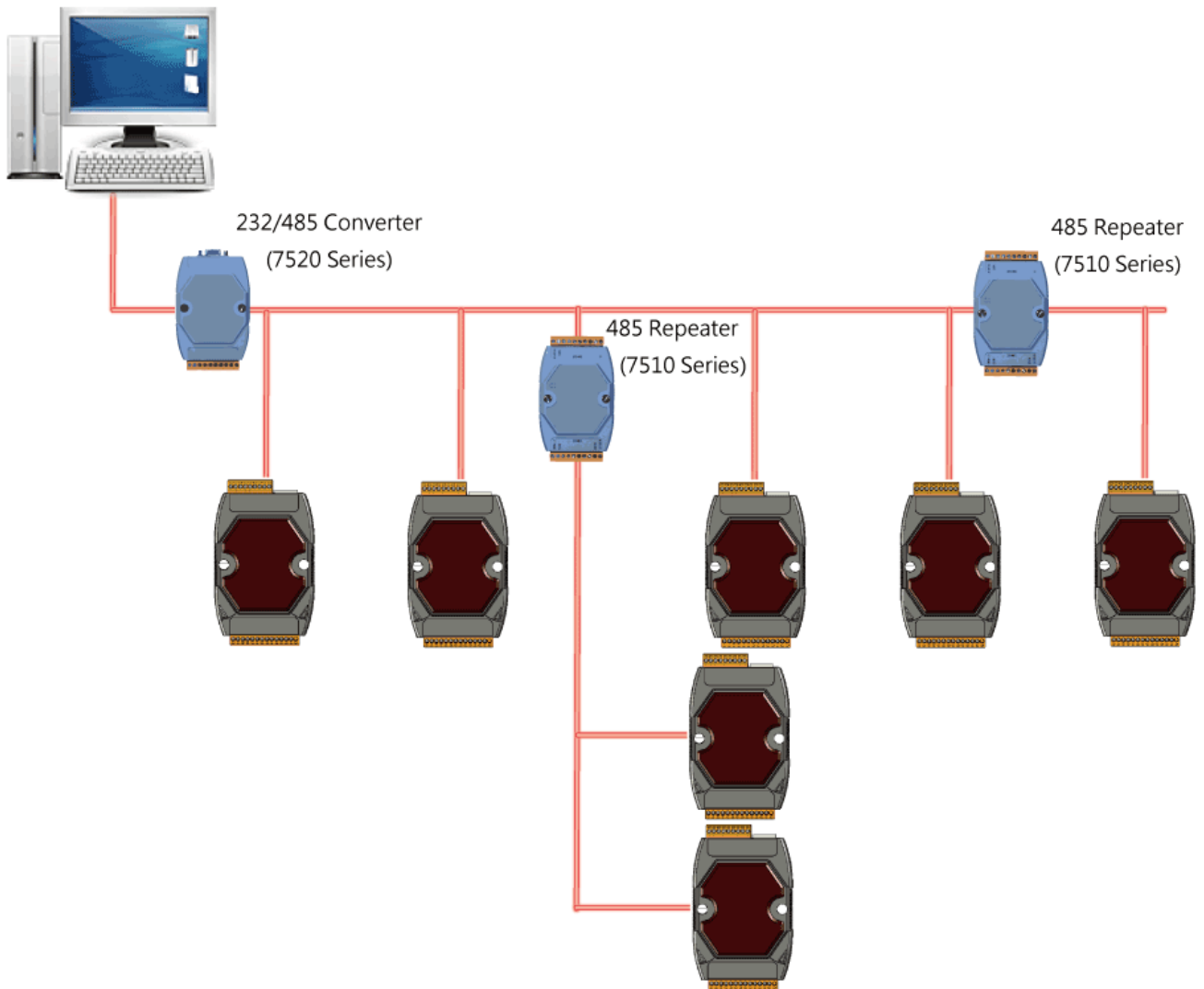
There is a better choice to use 7513 as a RS-485 hub on start type network.



## G.4. Random RS-485 network

---

There are branches along the main wire. In this case, it is better to have a repeater to isolate or filter the noise that is made by devices.

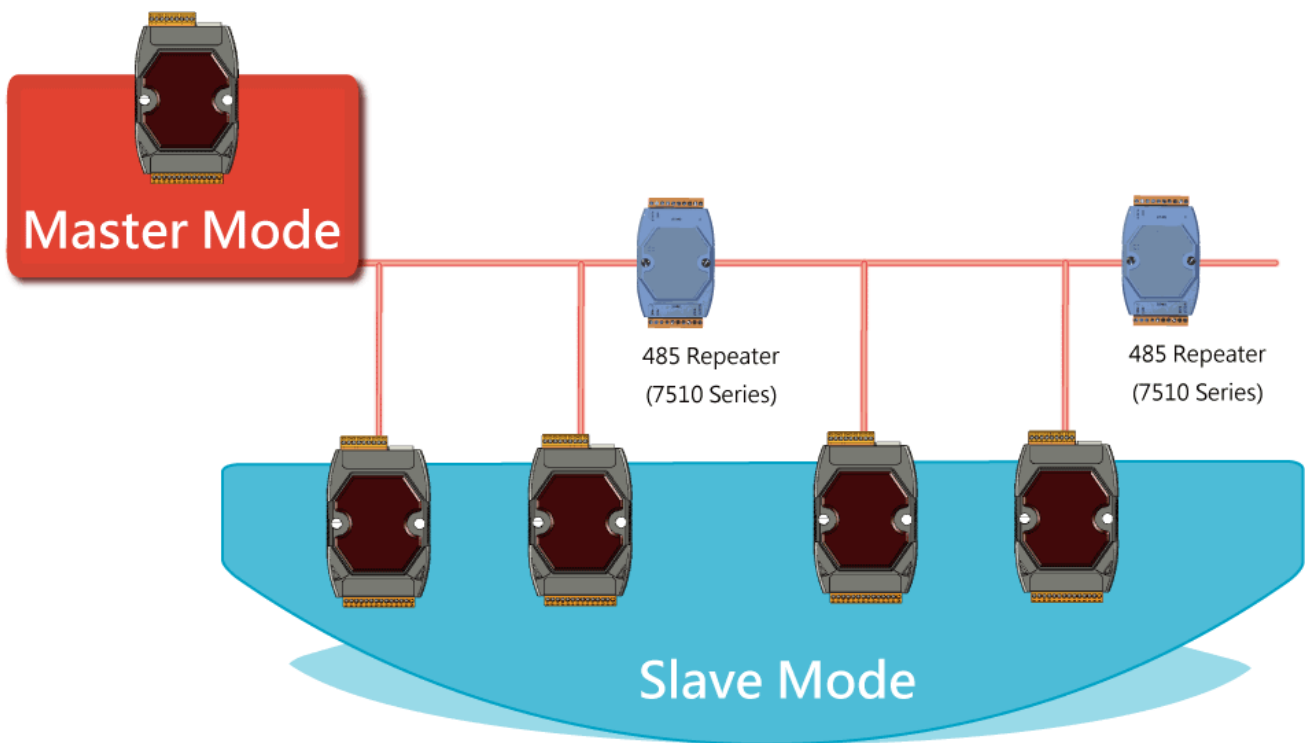


## G.5. $\mu$ PAC-7186EX Master-Slave Mode

The  $\mu$ PAC-7186EX provides two RS-485 serial port based on the master-slave architecture, all of which have a pull-high/pull-low resistor, you can set it to master mode or slave mode for implementing a RS-485 multi-drop network.

### G.5.1. $\mu$ PAC-7186EX as a Master

When one of  $\mu$ PAC-7186EX is set to master, then all the other devices on the same network must be slave mode. then the master one' s ( $\mu$ PAC-7186EX) pull-high/pull-low resistors have to adjusted to enabled. Please refer to the Figure H-1 for the jumpers' setting of the pull-high/pull-low resistors which are located at the power board of  $\mu$ PAC-7186EX.

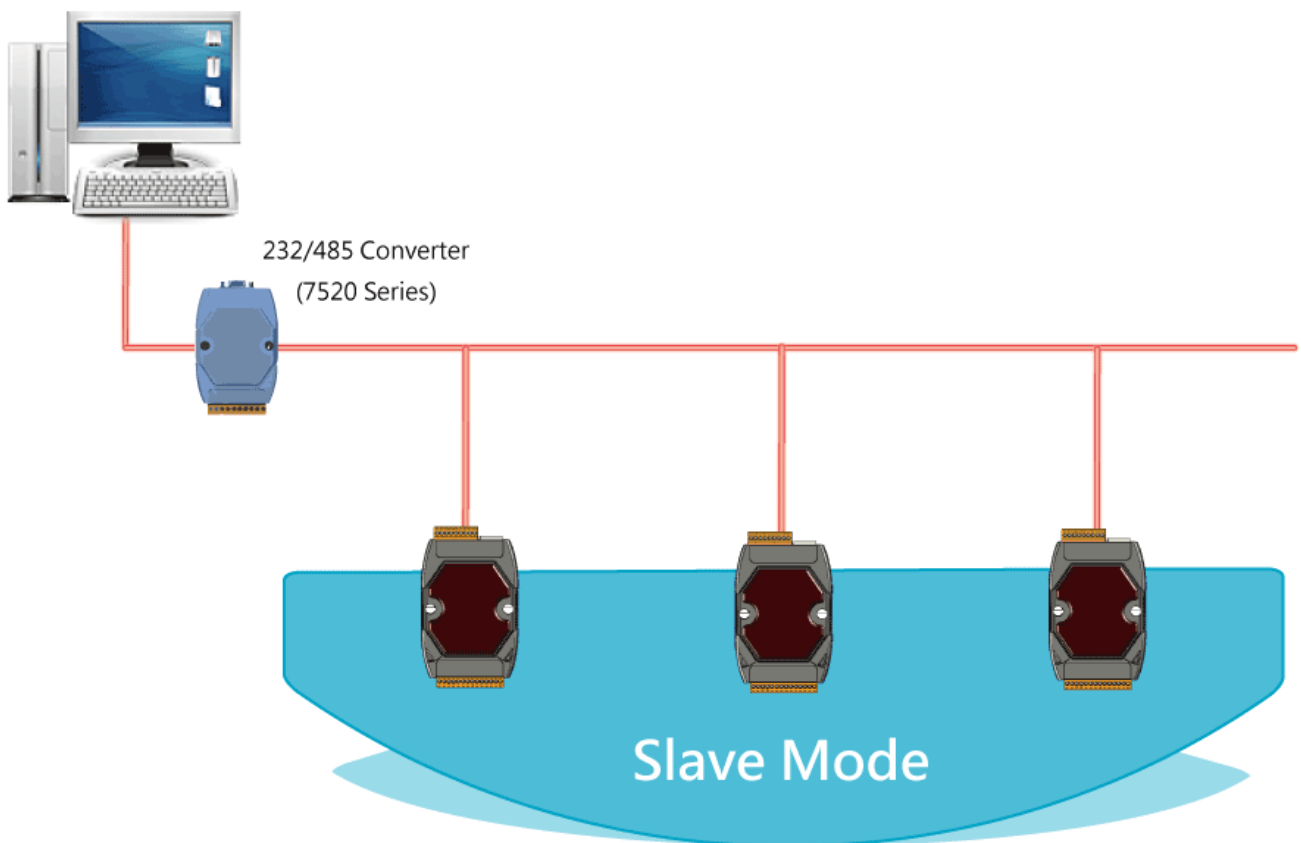




## G.5.2. $\mu$ PAC-7186EX as a slave

For most of application, when using one 7520 series as RS-232/485 converter, its pull-high/pull-low resistors are set to enabled. Then the  $\mu$ PAC-7186EX and all the other devices on this network must be slave mode (the pull-high/pull-low resistors must be disabled).

Please refer to the figure H-2 to for the jumpers' setting of the pull-high/pull-low resistors which are located at the power board of  $\mu$ PAC-7186EX.



**Figure H-2**

If there are repeaters on the RS-485 network, there will be pull-high/pull-low resistors on both sides of the repeaters (i-7510)

