iPAC-8000 User Manual
(C Language Based)

Version 1.0.1, June 2010

Service and usage information for

iP-8411

iP-8811

iP-8441/iP-8441-FD

iP-8841/iP-8841-FD

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

Trademark

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

Contact US

If you have any question, please feel free to contact us.
We will give you a quick response within 2 workdays.

Email: service@icpdas.com

Table of Contents

1. Introduction

iPAC-8000 is a compact size PAC (Programmable Automation Controller). It equips an 80186 CPU (16bits and 80MHz) running a MiniOS7 operating system, several communication interface (Ethernet, RS-232/485) and 4/8 slots to expand I/O modules.

The operating system, MiniOS7, can boot up in a very short time (0.4~0.8 seconds). It has a built-in hardware diagnostic function, and supports the full range of functions required to access all high profile I-8K and I-87K series I/O modules, such as DI, DO, DIO, AI, AO, Counter/Frequency, motion control modules, etc. And to simplify the TCP/IP software developing process, a software development template, X-Server, is provided. It implements 90% functionalities of Ethernet communication. Software engineer can easily finish the 10% remaining functionality and greatly shorten the developing time.

The iPAC-8000 is designed for applications to industrial monitoring, measurement and controlling; therefore, we made it with redundant power inputs with 1KV isolation from noise and surges, and a wide range of operating temperature (-25°C~+75°C). It is tough enough to service in harsh and rough environments.

1.1. Features

➤ Software Features

MiniOS7 embedded operating system

MiniOS7 was introduced in 1996 as an MS-DOS like operating system for embedded controller developers. The features of MiniOS7 include

A. Small kernel size (64KB)
B. Fast boot speed (0.4~0.8 second)
C. Hardware diagnostic functions
D. Simple command line operation over RS-232 or Ethernet
E. Load files via RS-232 or Ethernet

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 iPAC-8000 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.

Redundant Ethernet Communication (for iP-8441 and iP-8841 modules only)

With the dual LAN features of iPAC-8000, user's software on PCs or other controllers can implement redundant Ethernet communication. With VxComm technique, the redundant Ethernet communication is ready. One virtual COM port on PC can map to one COM port of iPAC-8000 via two IP address. When the communication is failed (or timeout), the VxComm driver can automatically and quickly switch the virtual COM port mapping to another IP address to keep the communication.

Easy-Use Software Development Template (Xserver) for TCP/IP Application

To simplify the TCP/IP software developing process, we designed a software develop template, called XServer. It is a reliable, opened, expandable, all purposed, and easily to be used library. The Xserver implements 90% functionalities of Ethernet communication. Refer the rich demo programs we provided, software engineer can easily finish the 10% remaining funtionalities and greatly shorten the developing time.

Slave I/O firmware options (for DCON or Modbus/TCP protocol)

In some simple Ethernet I/O applications, users just want to know how to send a command to the I/O to get back a response. They don’t want to develop a firmware. That is too difficult to them. Thus, we also provide two firmware for this purpose.

A. DCON firmware
   DCON firmware supports an ASCII string based command set, called DCON protocol

B. Modbus firmware
   Modbus firmware supports the standard Modbus/TCP protocol. SCADA software can easily access the I/O module plugged in the iPAC-8000.

➢ Hardware Features

80186 CPU (16bit and 80MHz) with 512KB Flash and 768KB SRAM

The 512KB flash is for storing files, and the 768KB SRAM is for running programs.

64-bit Hardware Serial Number

The 64-bit hardware serial number is unique and individual. Every serial number of iPAC-8000 is different. Users can add a checking mechanism to their AP to prevent software from pirating.

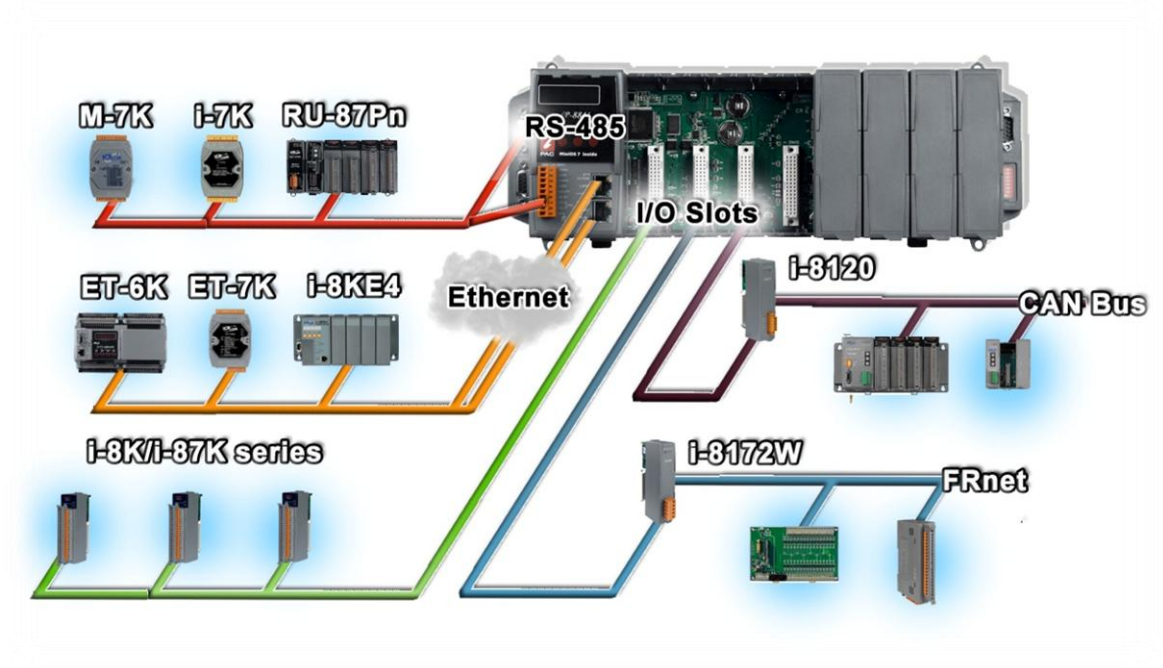
Dual Battery Backup SRAM (512KB)

To maintain important data while power off, non-volatile memory is the ideal design. The iPAC-8000 equips a 512KB SRAM with two Li-batteries to maintain data while power off. The two Li-batteries can continually supply power to the 512KB SRAM to retain the data for 5 years; and the dual-battery design can avoid data lost while replacing a new battery.

I/O Module Hot Swap Ability

The iPAC-8000 features hot swap which means that there is no need to power off the iPAC-8000 for replacing modules. And the OS provides a function sending plug-in and removing messages to user's applications. Using this feature, users can design its own plug-and-play applications.

Rich I/O Expansion Ability (RS-232/485, Ethernet, FRnet, CAN)



Beside the local I/O slots, iPAC-8000 also equips several RS-232/485 ports, two Ethernet ports to connect serial I/O and Ethernet I/O. And with FRnet and CAN communication module in local slot, FRnet I/O and CAN devices are easy to be integrated.

Dual Ethernet Ports (for iP-8441 and iP-8841 modules only)

iPAC-8000 provides two Ethernet ports. The two Ethernet ports can be used to implement redundant Ethernet communication and separate Ethernet communication (one for global Internet, one for private Ethernet).

Redundant Power Inputs

To prevent theiPAC-8000 from failing by the power loss, the power module is designed with two inputs. The iPAC-8000 can keep working even one power input fails, and mean while there is a relay output for informing the power failure.

Ventilated Housing Design Allows Operation Between -25 ~ +75 ℃

Each iPAC-8000 is housed in a plastic-based box with a column-like ventilator that can help to cool the working environment inside the box and allow the iPAC-8000 operating between -25 °C and +75 °C.

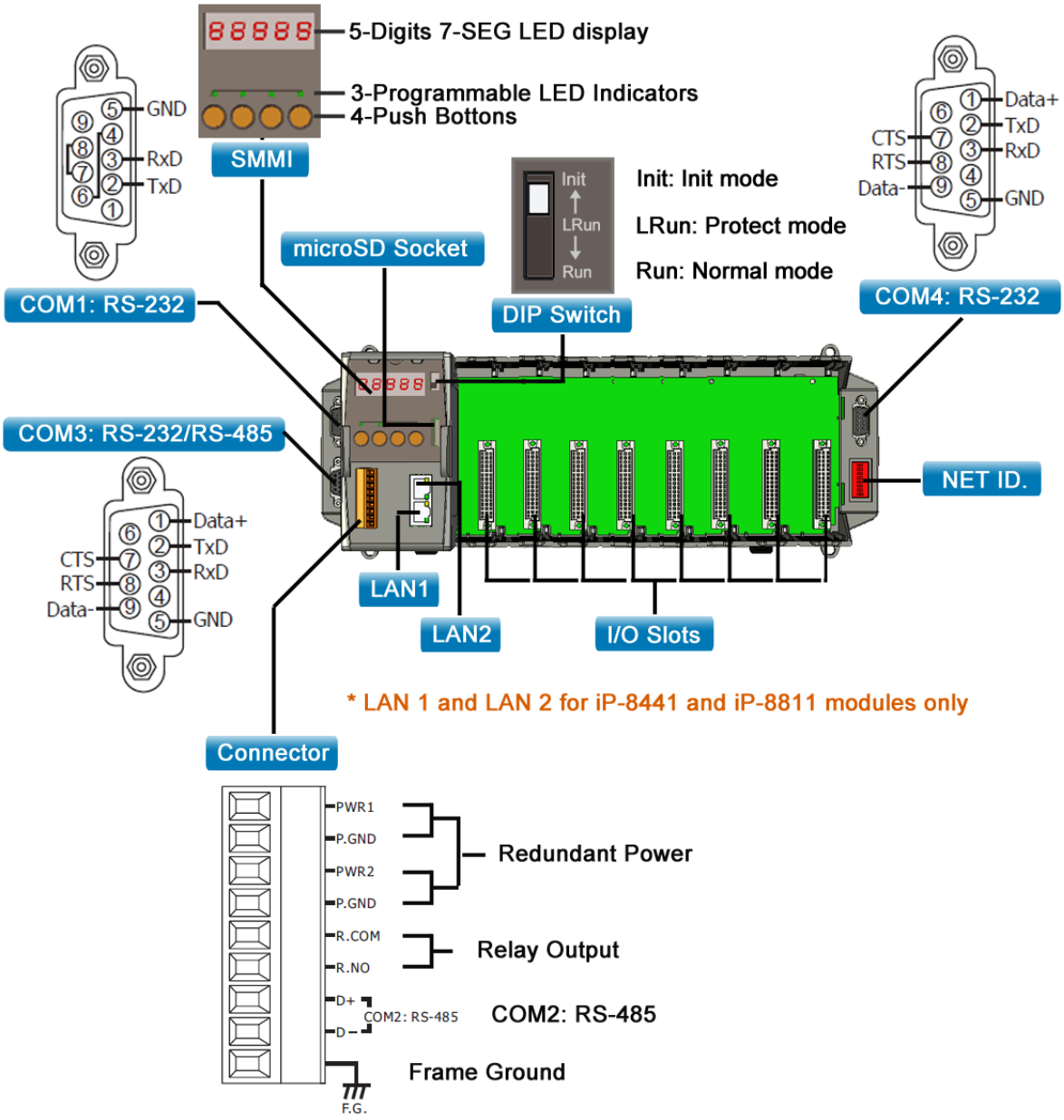
1.2. **Specification**

Models		iP-8411	iP-8441	iP-8441-FD	iP-8811	iP-8841	iP-8841-FD
System Software							
OS		MiniOS7 (DOS-like embedded operating system)					
Program Download Interface		RS-232 (COM1)	RS-232 (COM1) or Ethernet		RS-232 (COM1)	RS-232 (COM1) or Ethernet	
Programming Language		C Language					
Compilers to create.exe files		TC++ 1.01 (Freeware) TC 2.01 (Freeware) BC++ 3.1 ~ 5.2x MSC 6.0 MSVC++ (before version 1.5.2)					
CPU Module							
CPU		80186 or compatible (16-bit and 80 MHz)					
SRAM		512 KB	768 KB		512 KB	768 KB	
Flash		512 KB (100,000 erase/write cycles) with Flash protection switch					
Expansion Flash Memory		microSD socket (can support 1/2 GB microSD)					
64 MB NAND Flash Disk		-		Yes	-		Yes
Dual Battery Backup SRAM		512 KB (for 5 years data retention)					
EEPROM		16 KB; data retention: 40 years; 1,000,000 erase/write cycles					
NVRAM		31 bytes (battery backup, data valid up to 5 years)					
RTC (Real Time Clock)		Provide second, minute, hour, date, day of week, month, year)					
64-bit Hardware Serial Number		Yes					
Watchdog Timers		Yes (0.8 second)					
NET ID		8-pin DIP switch to assign NET ID as 1 ~ 255					
Communication Ports							
Ethernet		-	RJ-45 x 2, 10/100 Base-TX (Auto negotiating, auto MDI/MDI-X, LED indicators)		-	RJ-45 x 2, 10/100 Base-TX (Auto negotiating, auto MDI/MDI-X, LED indicators)	
COM0		Internal communication with the high profile I-87K series modules in slots					
COM1		RS-232 (to update firmware) (RxD, TxD and GND); non-isolated					
COM2	RS-485	D2+, D2-; self-tuner ASIC inside					

	Isolation	3000 V_{DC}			
COM3		RS-232/RS-485 (RxD, TxD, CTS, RTS and GND for RS-232, Data+ and Data- for RS-485); non-isolated			
COM4		RS-232 (RxD, TxD, CTS, RTS, DSR, DTR, CD, RI and GND); non-isolated			
SMMI					
5-Digital LED Display		Yes			
3-Programmable LED Indicators		Yes			
4-Push		Yes			
Buzzer		-	Yes	-	Yes
I/O Expansion Slots					
Slot Number		4		8	
		(For high profile I-8K and I-87K modules only)			
Hot Swap * Will be available		For high profile I-87K modules only			
Data Bus		8/16 bits			
Address Bus Range		2 K for each slot			
Mechanical					
Dimensions (W x L x H)		231 mm x 132 mm x 111 mm		355 mm x 132 mm x 111 mm	
Installation		DIN-Rail or Wall Mounting			
Operating Environment					
Operating Temperature		-25 ~ +75 °C			
Storage Temperature		-30 ~ +80 °C			
Humidity		10 ~ 90 % RH (non-condensing)			
Power					
Input Range		+10 ~ +30 V_{DC}			
Isolation		1 kV			
Redundant Power Inputs		Yes, with one power relay (1 A @ 24 V_{DC}) for alarm			
Capacity		0.85 A, 5 V supply to CPU and backplane, 5.51 A, 5 V supply to I/O expansion slots, 30 W in total		0.9 A, 5 V supply to CPU and backplane, 5.1 A, 5 V supply to I/O expansion slots, 30 W in total	
Consumption		6.7 W (0.28 A @ 24 V_{DC})			

1.3. Overview

Here is a brief overview of the components and its descriptions for module status.

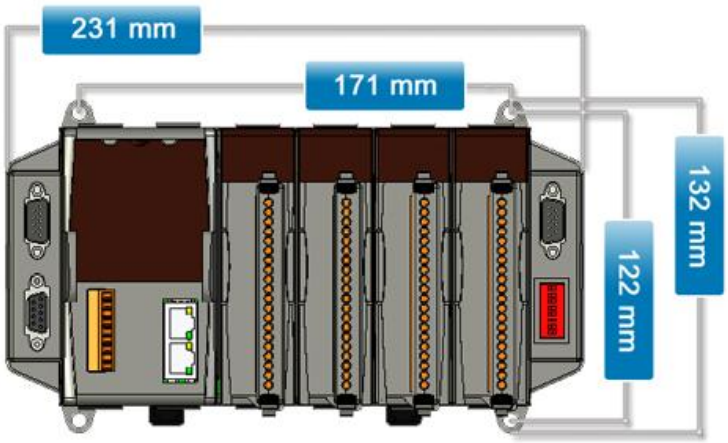


1.4. **Dimension**

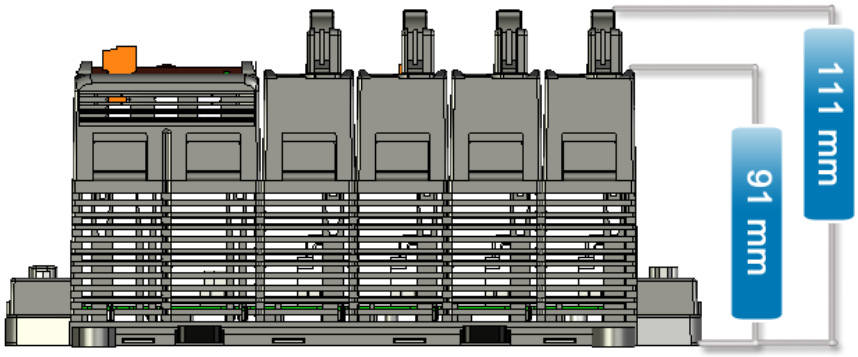
There are several series of iPAC-8000 modules whose dimensions depended on the quantity of the slot.

1.4.1. **4 Slots**

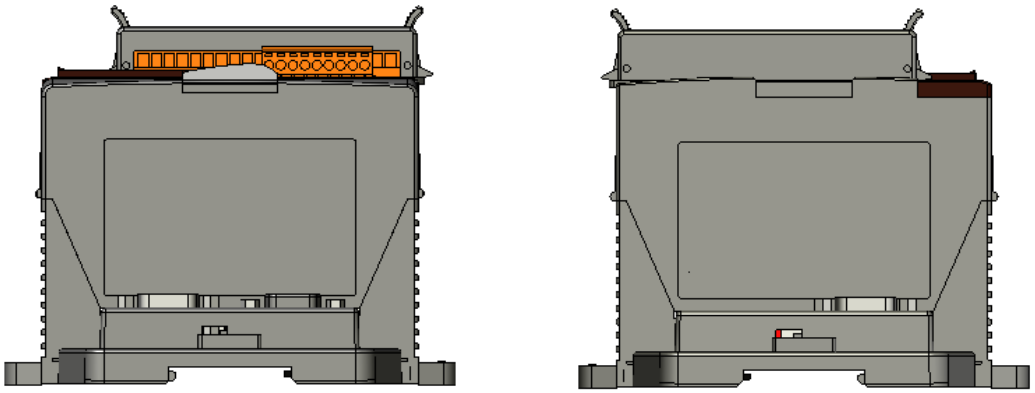
➤ Top View



➤ Front View

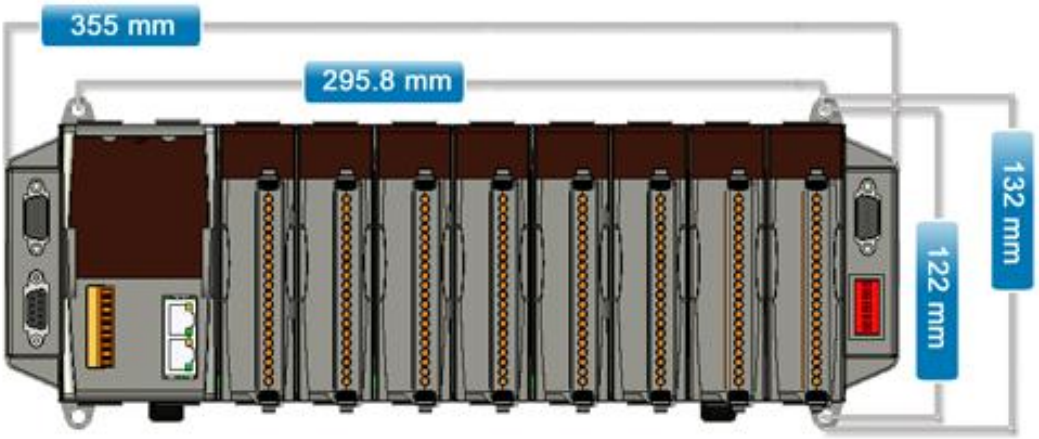


➤ Side View

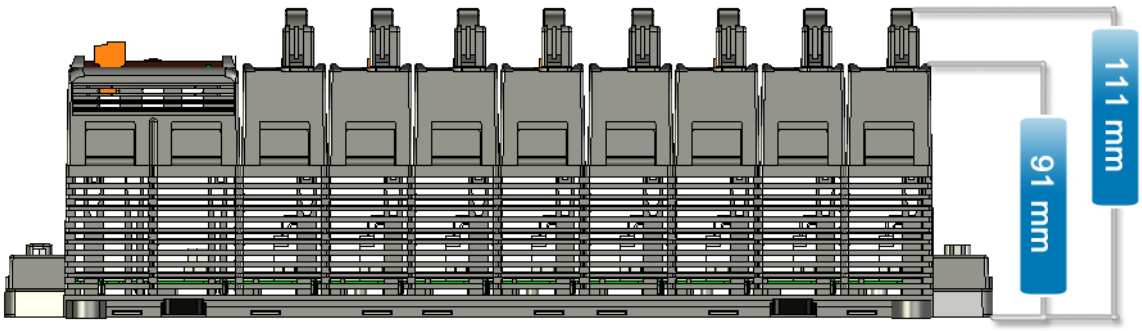


1.4.2. **8 Slots**

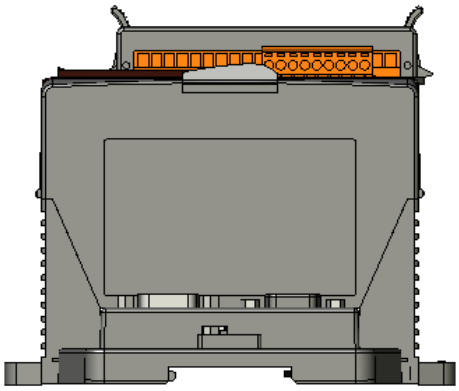
➤ Top View

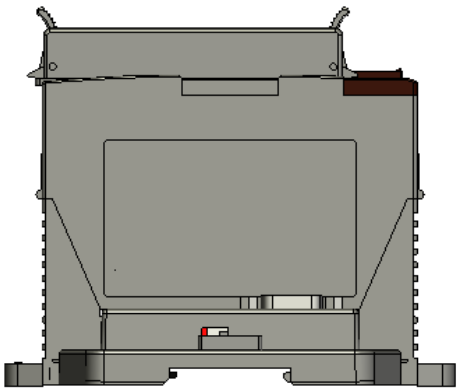


➤ Front View



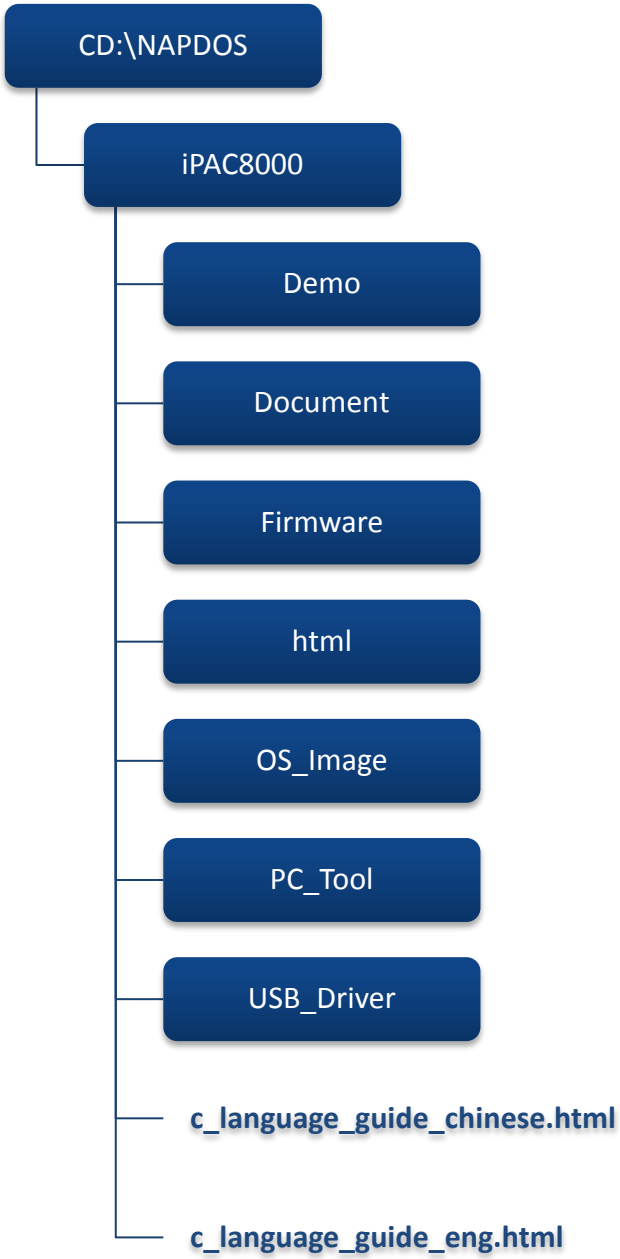
➤ Side View





1.5. Companion CD

This package comes with a CD that provides drivers, software utility, all of the required documentations…, etc. All of them are listed below.



2. Getting Started

If you are a new user, begin with this chapter, it includes a guided tour that provides a basic overview of installing, configuring and using the iPAC-8000.

Before beginning any installation, please check the package contents. If any items are damaged or missing, please contact us.

In addition to Quick Start Guide, the package includes the following items:

iPAC-8000
(IP-8xxx)

Software Utility CD

RS-232 Cable
(CA-0915)

Screw Driver
(1C016)

2.1. **Hardware Installation**

Before installing the hardware, you should have a basic understanding of hardware specification, such as the size of hard drive, the usable input-voltage range of the power supply, and the type of communication interfaces.

For complete hardware details, please refer to section “1.2. Specifications”

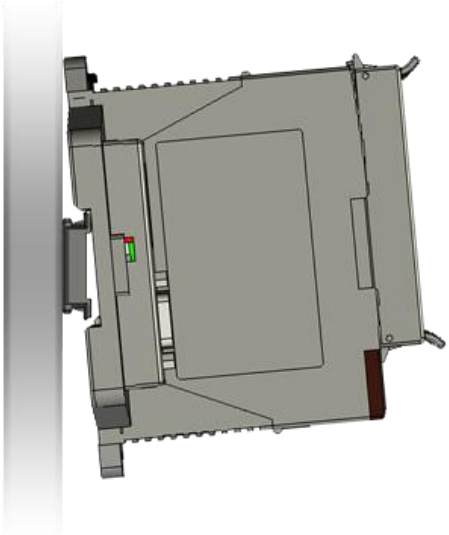
You also need to know the expansion capacities in order to choose the best expansion module for achieving maximal efficiency.
For more information about expansion module that are compatible with the iPAC-8000, please refer to

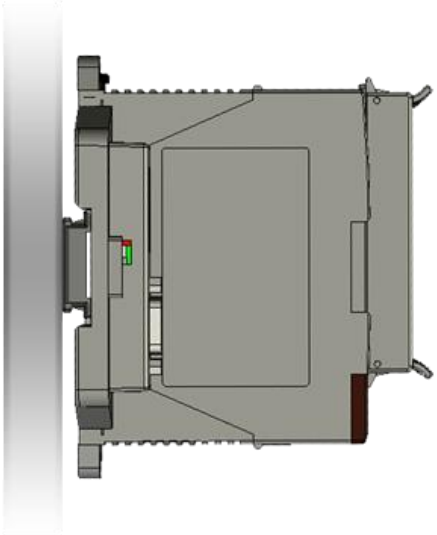
http://www.icpdas.com/products/PAC/I-8000/8000_IO_modules.htm

Below are step-by-step instructions for deploying the basic iPAC-8000 system.

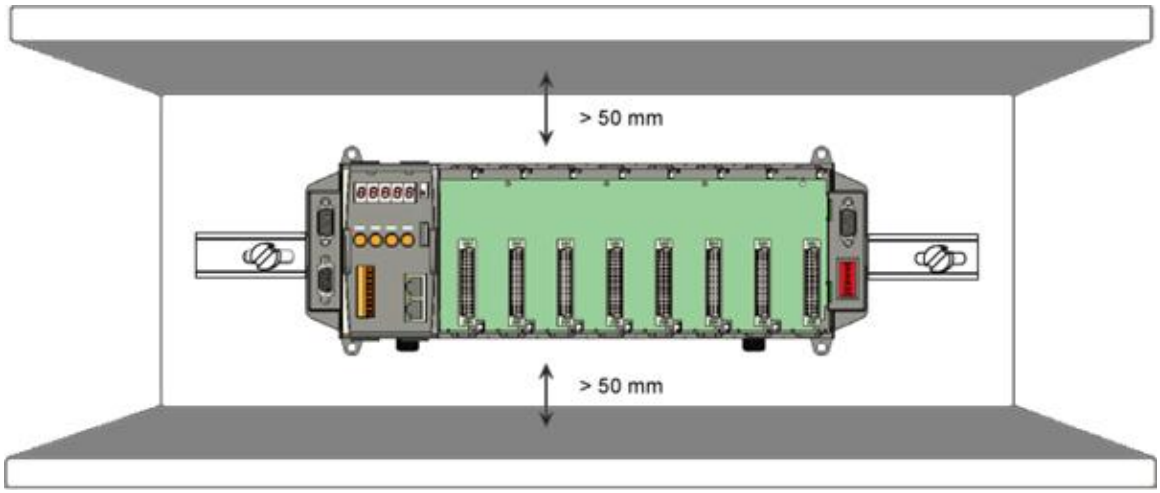
Step 1: Mount the hardware

The iPAC-8000 can be mounted with the bottom of the chassis on the DIN rail.



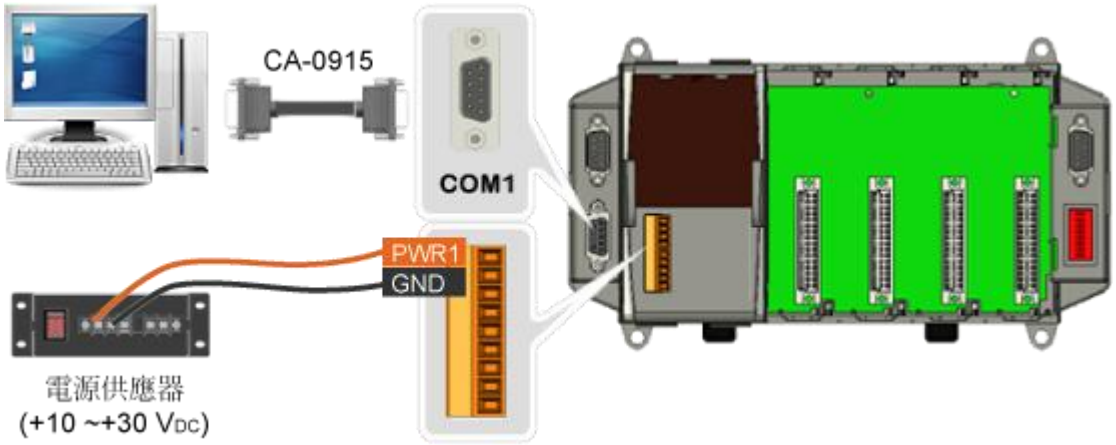


The iPAC-8000 installation must provide proper ventilation, spacing, and grounding to ensure the equipment will operate as specified. A minimum clearance of 50mm between the iPAC-8000 and the top and bottom side of the enclosure panels must be provided.



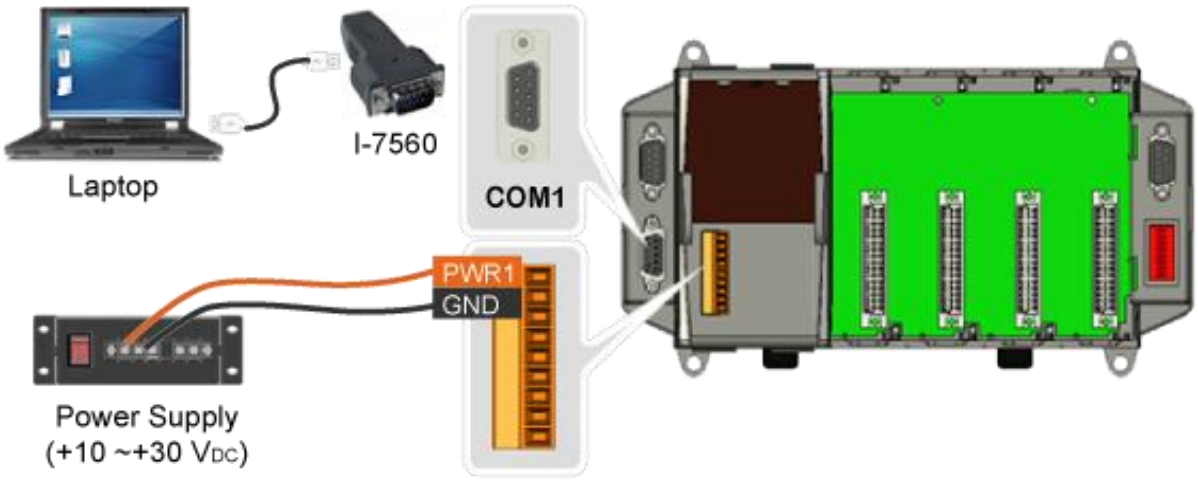
Step 2: Connect the iPAC-8000 to PC and setting up the power supply

The iPAC-8000 equips an RJ-45 Ethernet port for connection to an Ethernet hub/switch and PC, and powered by a standard 12 VDC power supply.



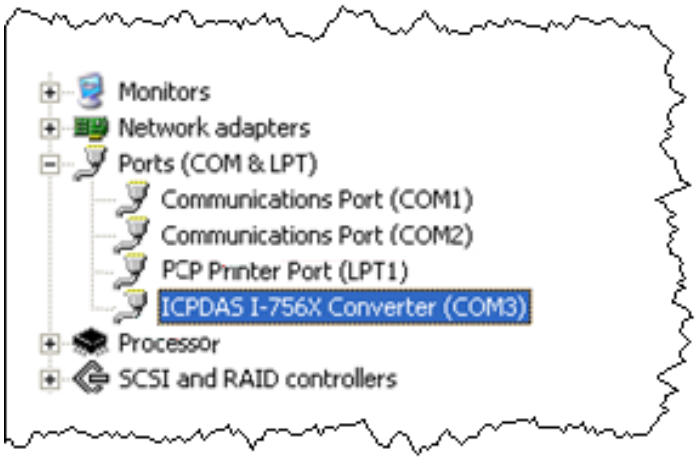
If the PC/Laptop is not fitted with a COM port, you can use the I-7560 (USB to RS-232

converter) for connection between iPAC-8000 and PC/Laptop.



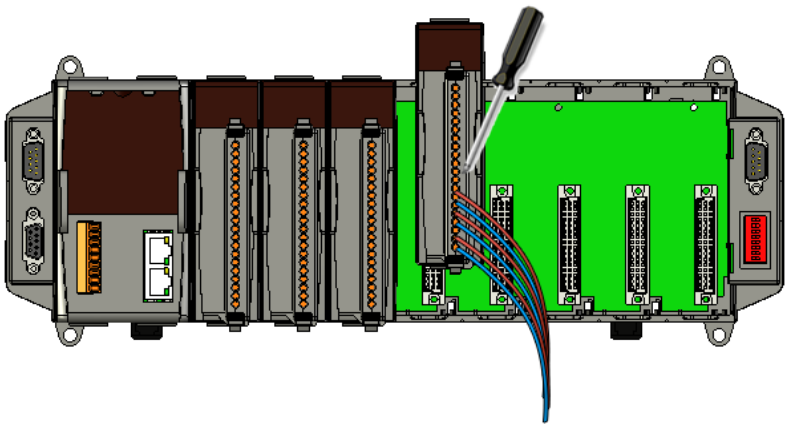
The I-7560 driver need be installed before starting to use it. You can obtain the driver from enclosed CD:
CD:\Napdos\7000\756x or FTP site:
ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/7000/756x/

After installing the USB driver, please check the “Device Manager” to make sure the driver has been installed and the COM port number which is assigned to the I-7560.



Step 3: Insert I/O modules

There are various types of I/O expansion modules for interfacing many different field devices to the iPAC-8000 system.



For more information about I/O expansion module, please refer to
http://www.icpdas.com/products/PAC/xpac/remote_io_support_list.htm

These modules have their own manuals, so if you are using them you should supplement this manual with the manual specifically designed for the special module.

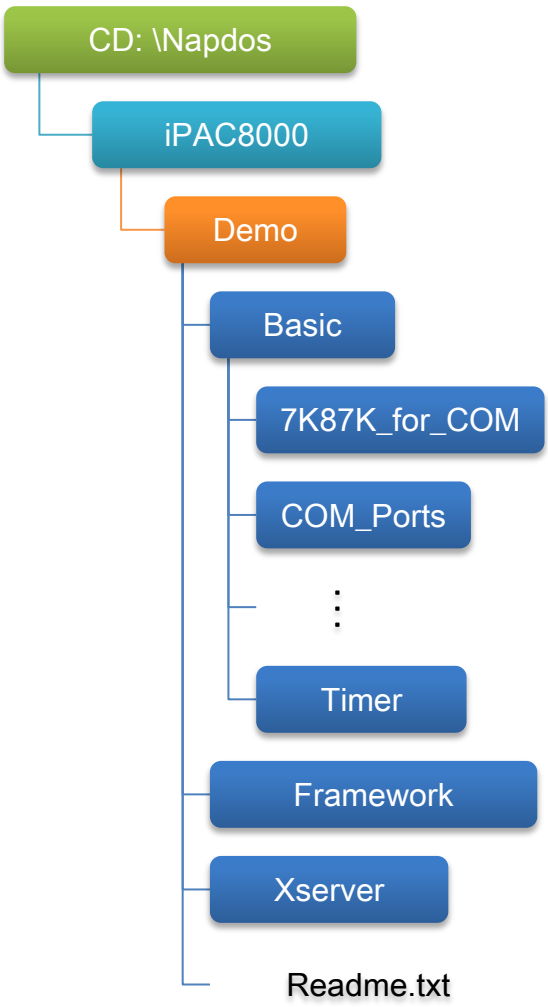
2.2. Software Installation

The Companion CD includes complete sets of APIs, demo programs and other tools for developing your own applications.

Below are step-by-step instructions for installing the iPAC-8000 APIs, demo programs and tools.

Step 1: Copy the "Demo" folder from the companion CD to PC

The folder is an essential resource for users developing your own applications which contains libraries, header files, demo programs and more information as shown below.



Step 2: Installing the MiniOS7 Utility

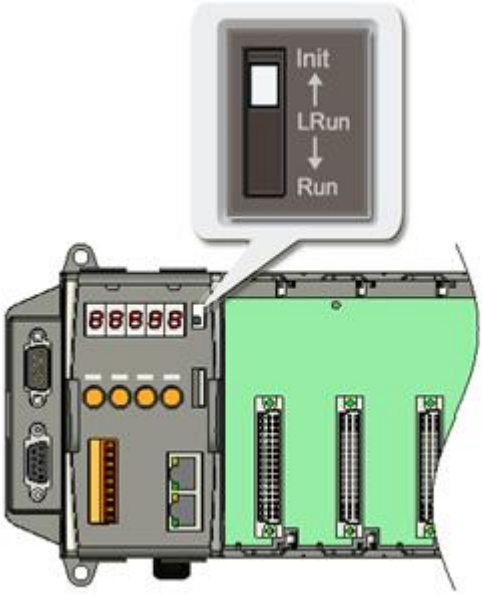
MiniOS7 Utility is a suite of tool for managing MiniOS7 devices (µPAC-5000, iPAC-8000, µPAC-7186,. etc.). It's comprised of four components – System monitor, communication manager, file manager and OS loader.

The MiniOS7 Utility can be obtained from companion CD or our FTP site:
CD:\Napdos\minios7\utility\minios7_utility\
ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/utility/minios7_utility/

2.3. **Boot Configuration**

Before you upload some programs to iPAC-8000, you need to enter the Init mode to stop the program running.

Make sure the switch of the Lock placed in the “Init” position.



2.4. **Uploading iPAC-8000 Programs**

MiniOS7 Utility is a suite of tool for managing MiniOS7 devices (μPAC-5000, iPAC-8000, μPAC-7186,. etc.). It's comprised of four components – System monitor, communication manager, file manager and OS loader.

Before you begin using the MiniOS7 Utility to upload programs, ensure that iPAC-8000 is connected to PC.

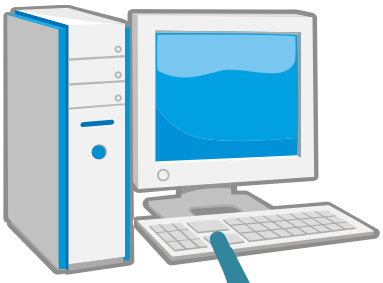
The upload process has the following main steps:

1. Establishing a connection between PC and iPAC-8000
2. Uploading and executing programs on iPAC-8000
3. Making programs start automatically

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

2.4.1. **Establishing a connection between PC and iPAC-8000**

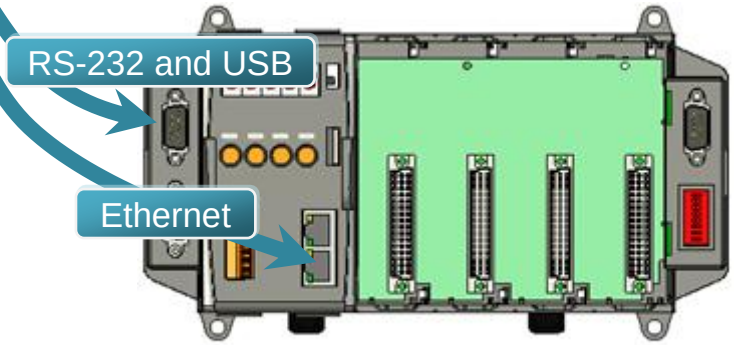
Connect the Host PC to the iPAC-8000 with the following connection types:



1. RS-232 connection
2. USB connection
3. Ethernet connection

(for iP-8441 and iP-8841 modules only)

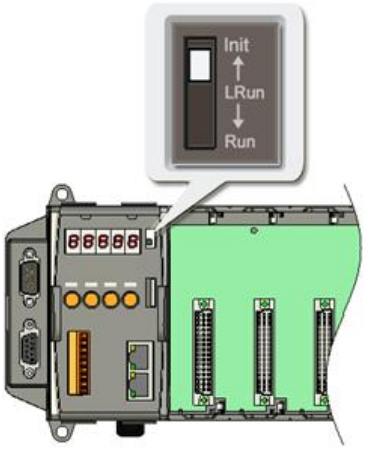
Each of the connection types will be described in detail later.



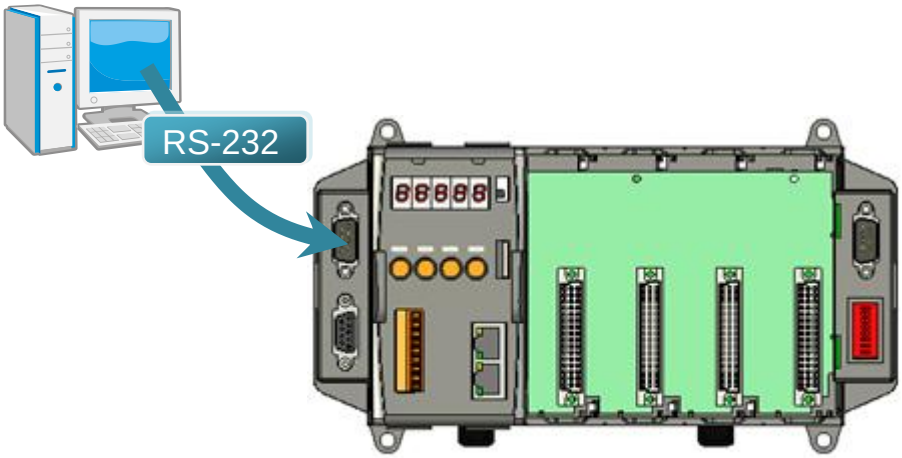
2.4.1.1. COM1 Connection

Below are step-by-step instructions on how to connect to PC using a RS-232 connection.

Step 1: Turn the switch to “Init” position

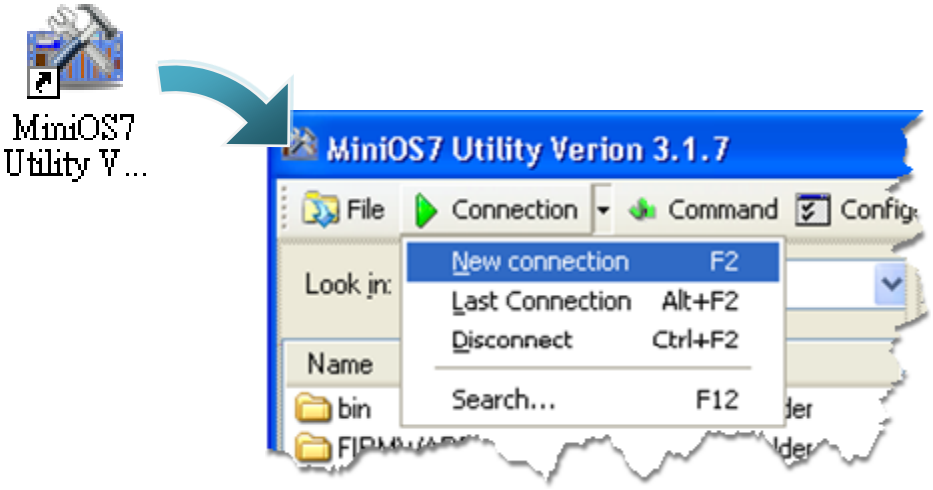


Step 2: Use the RS-232 Cable (CA-0915) to connect to PC

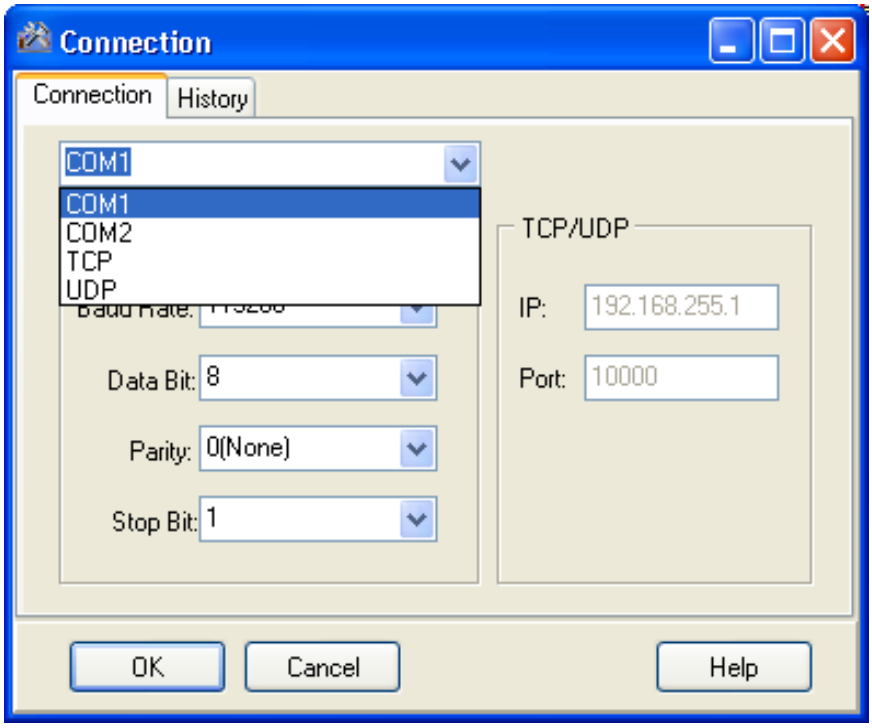


Step 3: Run the MiniOS7 Utility

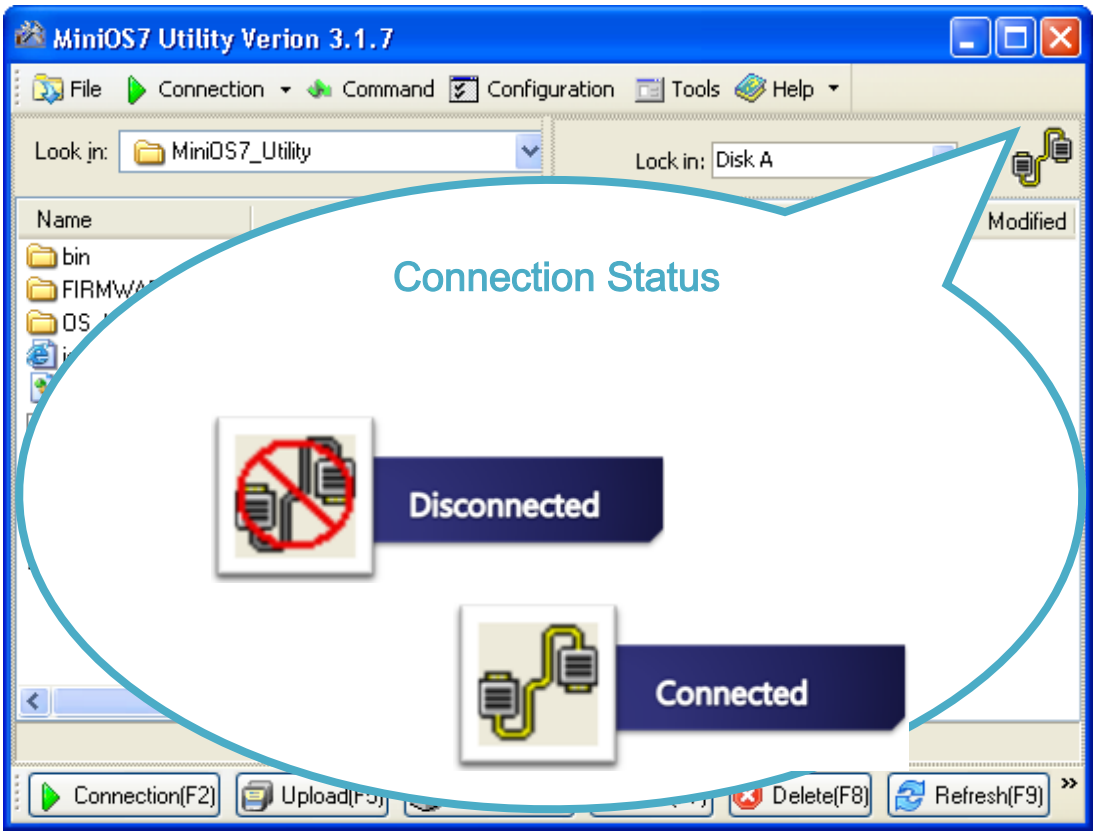
Step 4: Click the “New connection” function from the “Connection” menu



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



Step 6: The connection has already established

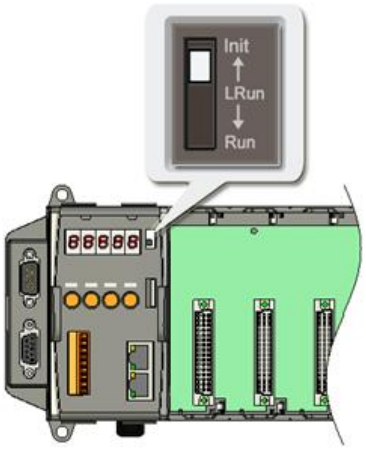


2.4.1.2. USB Connection

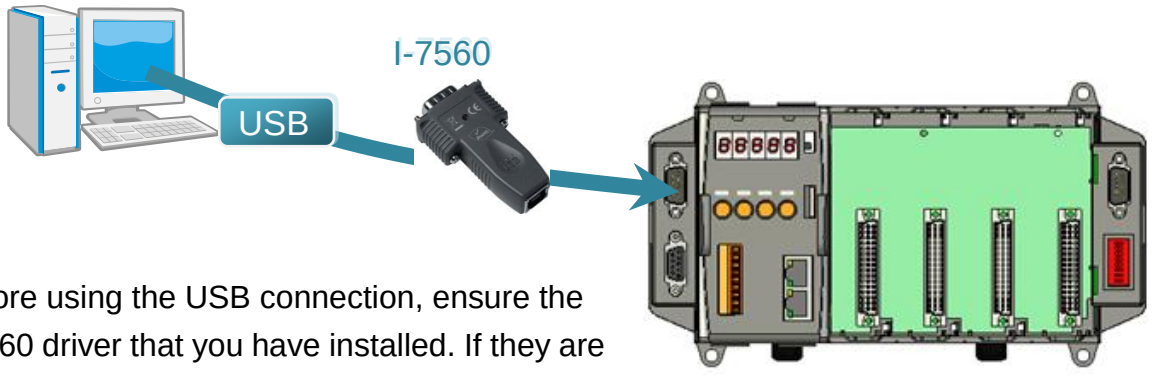
If the PC/Laptop is not fitted with a COM port, you can use the I-7560 (USB to RS-232 converter) for connection between iPAC-8000 and PC/Laptop.

Below are step-by-step instructions on how to connect to PC using a USB connection.

Step 1: Turn the switch to “Init” position



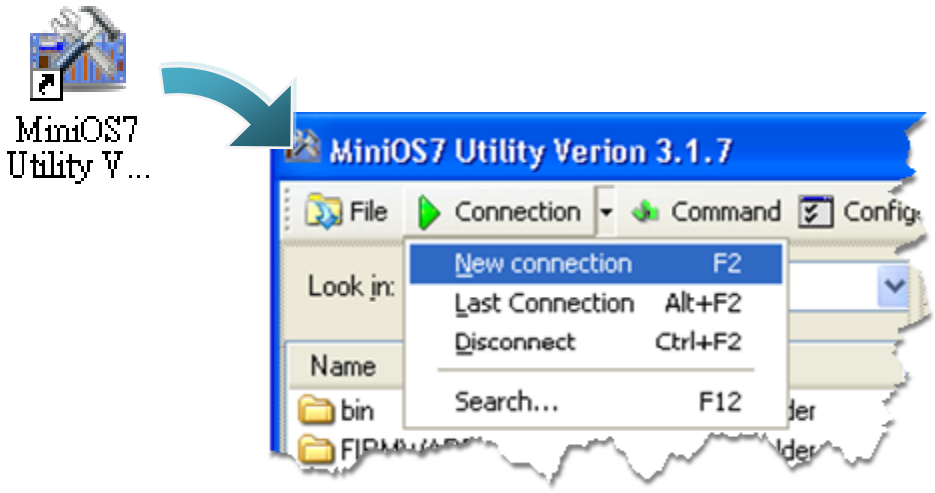
Step 2: Use the I-7560 to connect the iPAC-8000 and PC



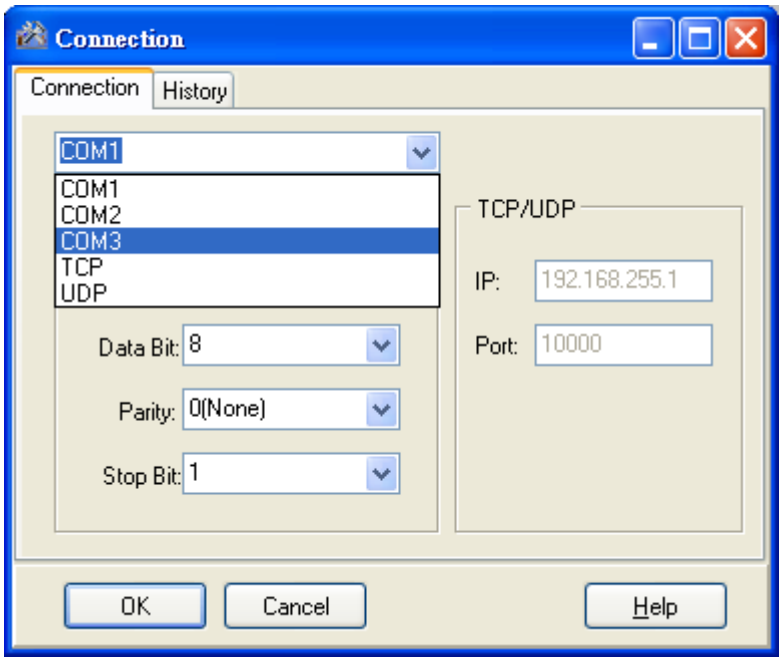
Before using the USB connection, ensure the I-7560 driver that you have installed. If they are not installed, please refer to “section 2.1. Hardware Installation”.

Step 3: Run the MiniOS7 Utility

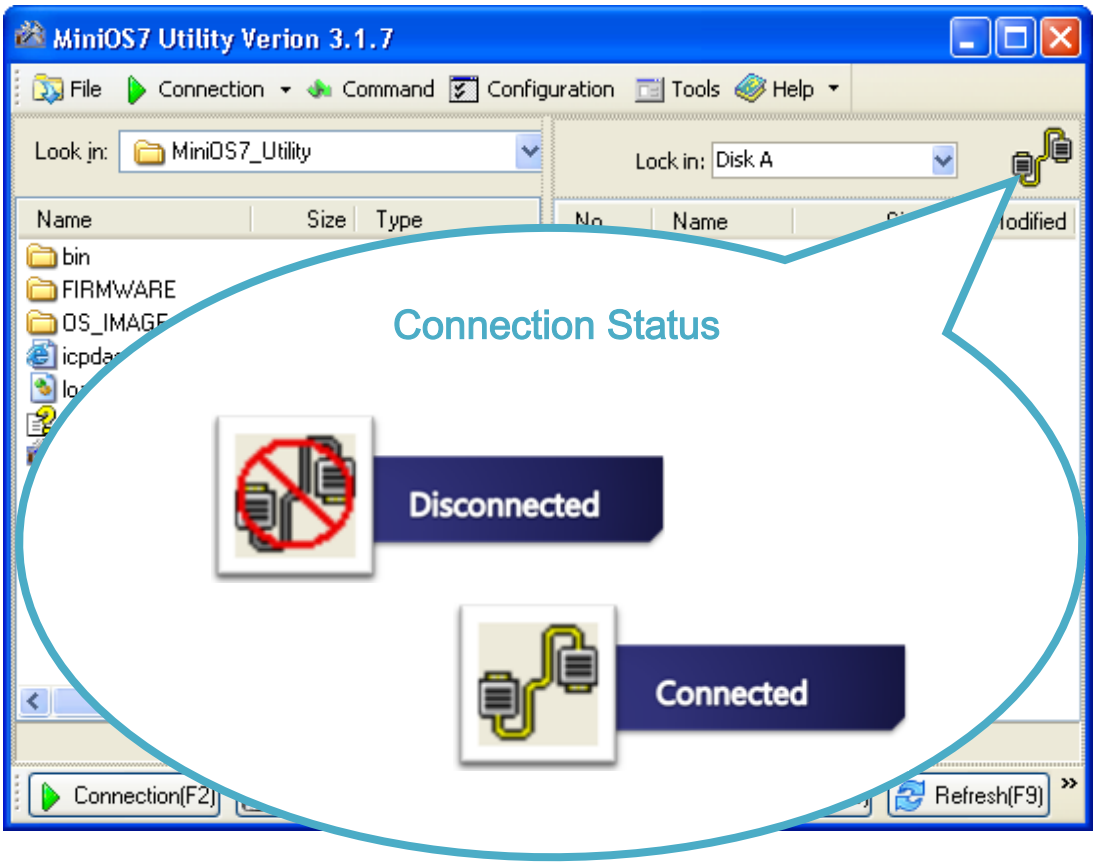
Step 4: Click the “New connection” function from the “Connection” menu



Step 5: On the “Connection” tab of the “Connection” dialog box, select “COM3” from the drop down list, and then click “OK”



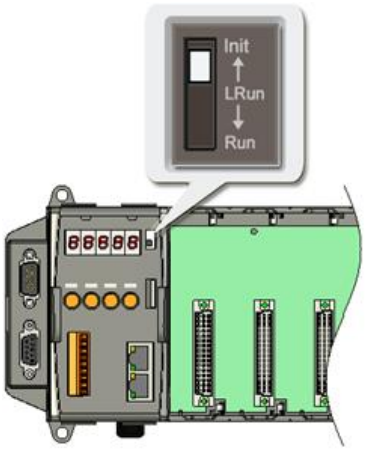
Step 6: The connection has already established



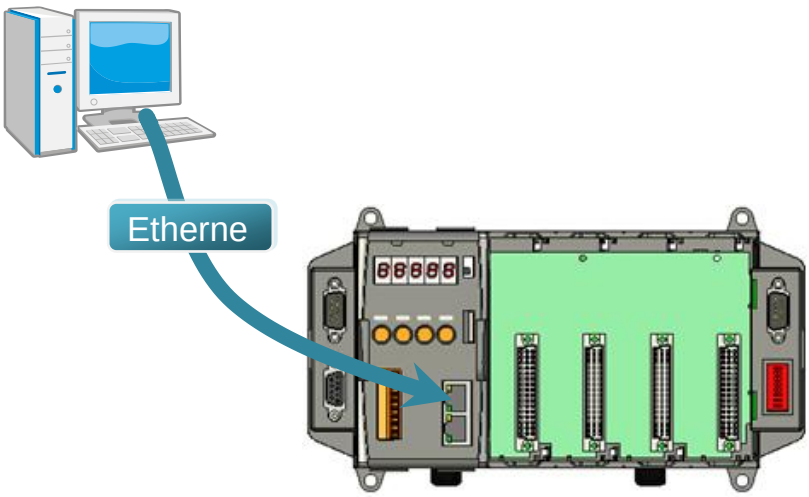
2.4.1.3. Ethernet Connection (for iP-8441 and iP-8841 modules only)

Below are step-by-step instructions on how to connect to PC using an Ethernet connection.

Step 1: Turn the switch to “Init” position

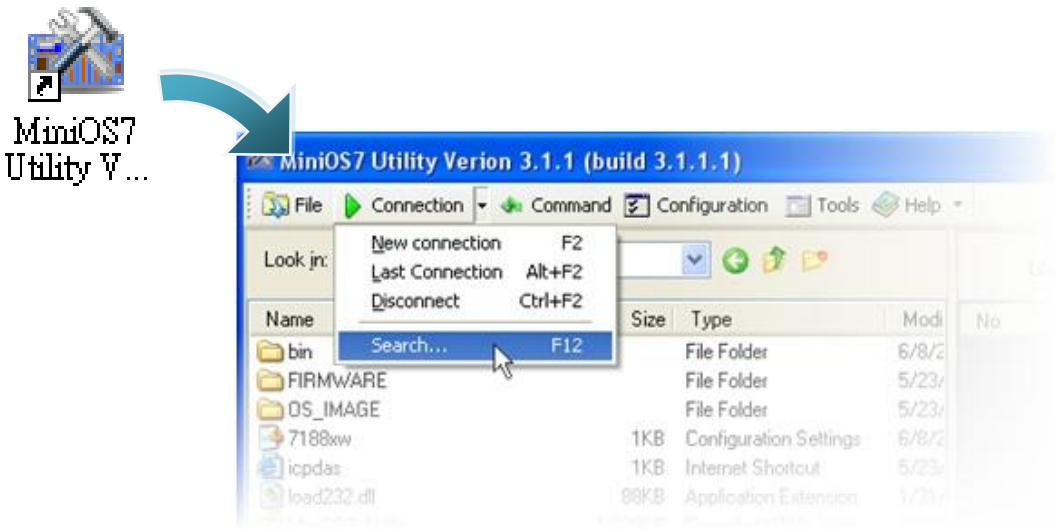


Step 2: Use the Ethernet Cable to connect to PC



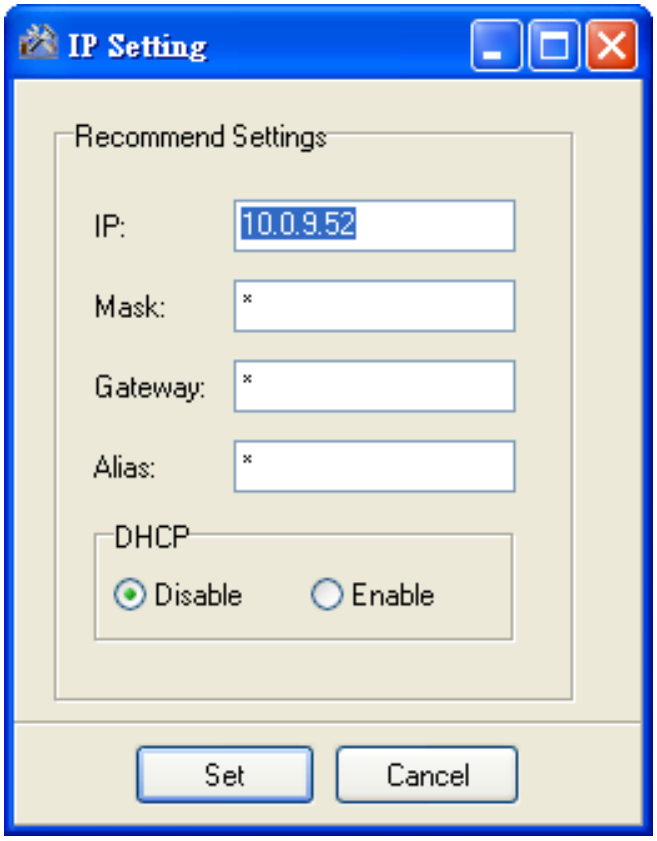
Step 3: Run the MiniOS7 Utility

Step 4: Click the “Search” function from the “Connection” menu

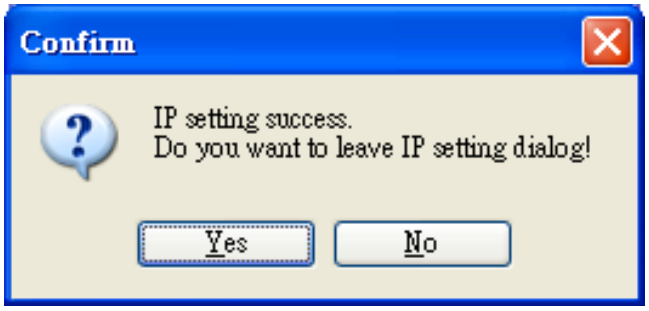


Step 5: On the “MiniOS7 Scan” dialog box, choose the module name from the list and then choose “IP setting” from the toolbar

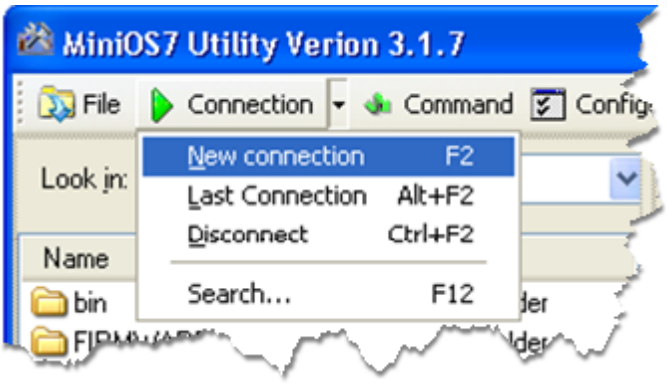
Step 6: On the “IP Setting” dialog, configure the “IP” settings and then click the “Set” button



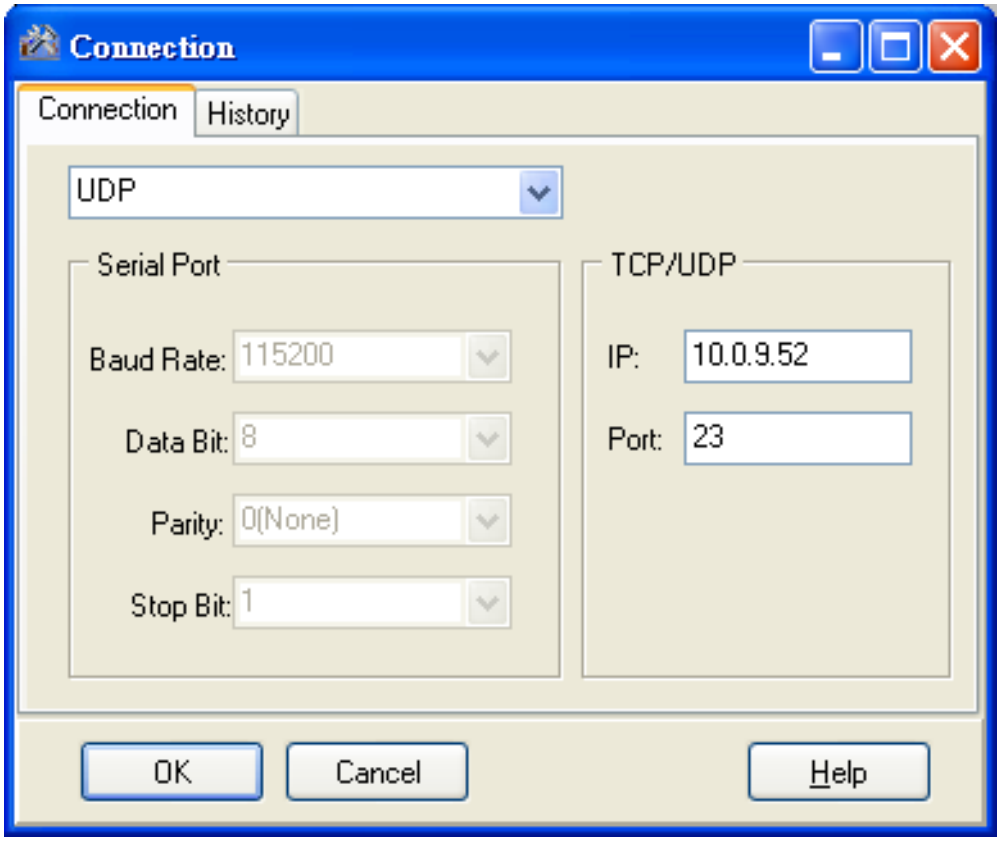
Step 7: On the “Confirm” dialog box, click “Yes”



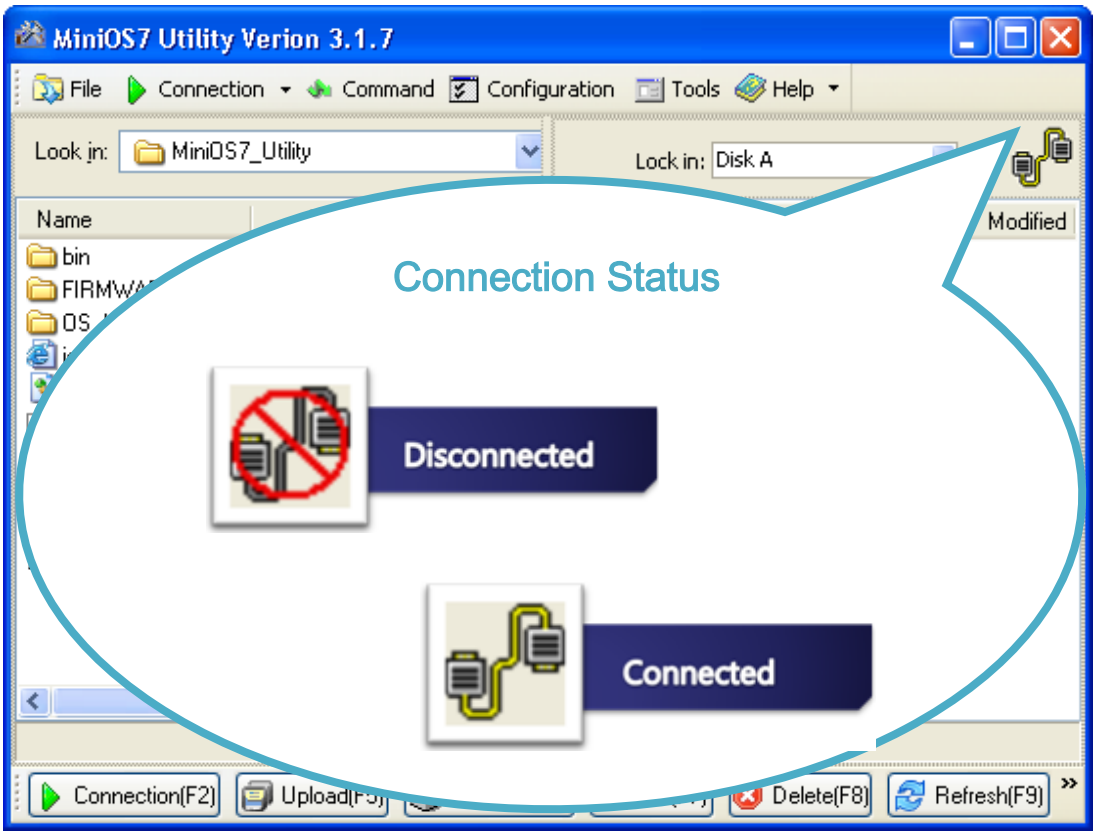
Step 8: Click the “New connection” function from the “Connection” menu



Step 9: 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”



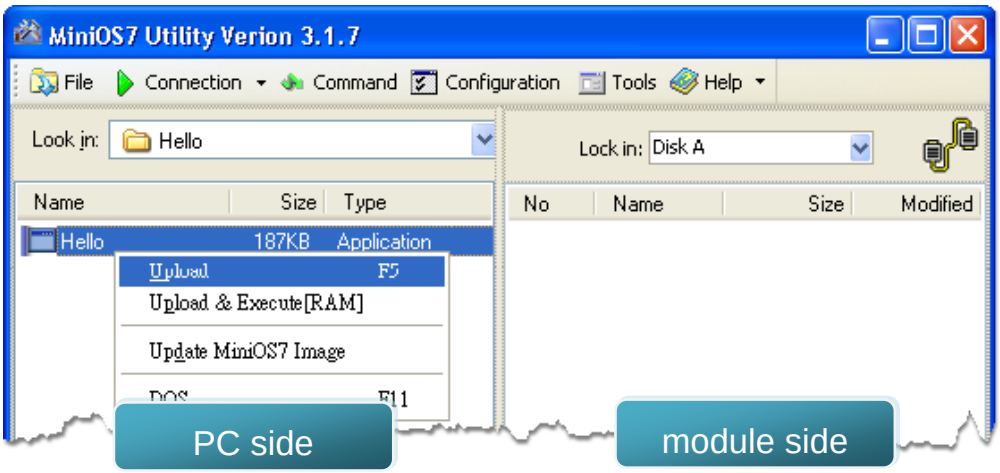
Step 10: The connection has already established



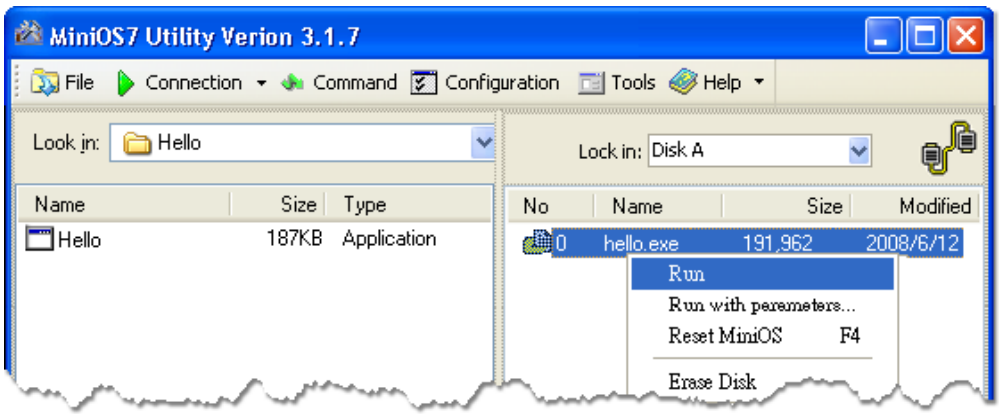
2.4.2. **Uploading and executing iPAC-8000 programs**

Before uploading and executing iPAC-8000 programs, you must firstly establish a connection between PC and iPAC-8000, for more detailed information about this process, please refer to section “2.4.1. Establishing a connection”

Step 1: On PC side, right click the file name that you wish to upload and then select the “Upload”



Step 2: On the module side, right click the file name that you wish to execute and then select the “Run”



2.4.3. **Making programs start automatically**

After upload programs on the iPAC-8000, if you need programs to start automatically after the iPAC-8000 start-up, it is easy to achieve it, to create a batch file called autoexec.bat and then upload it to the iPAC-8000, the program will start automatically in the next start-up.

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

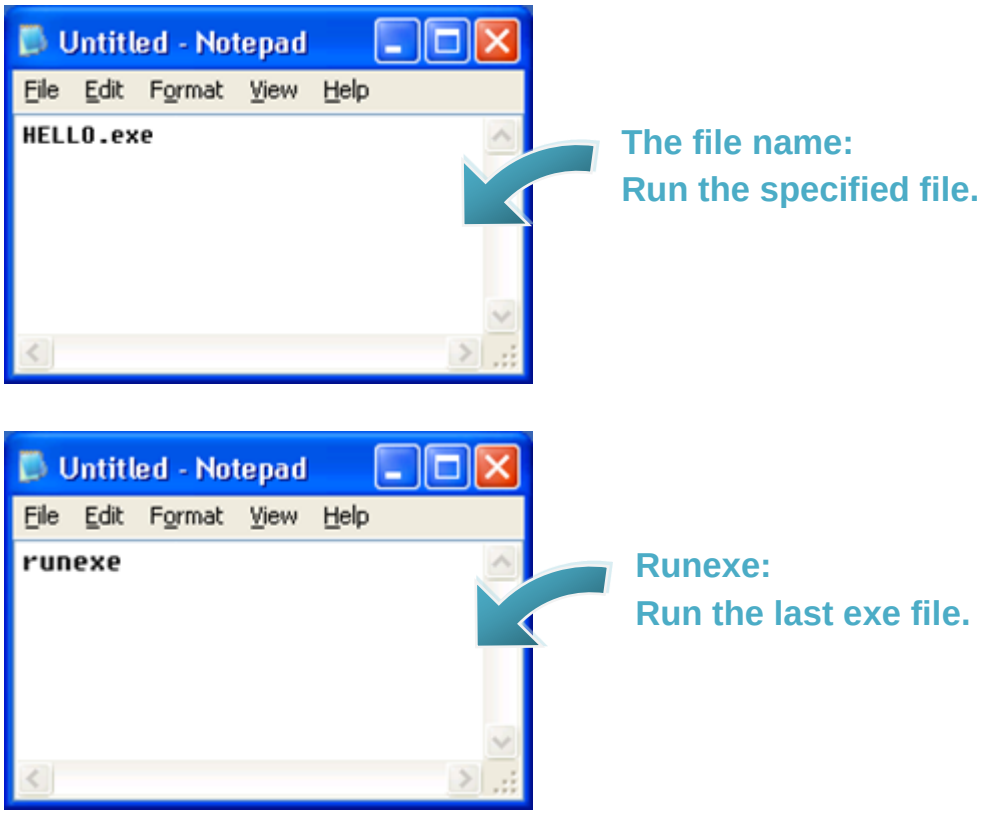
Step 1: Create an autoexec.bat file

i. Open the “Notepad”

ii. Type the command

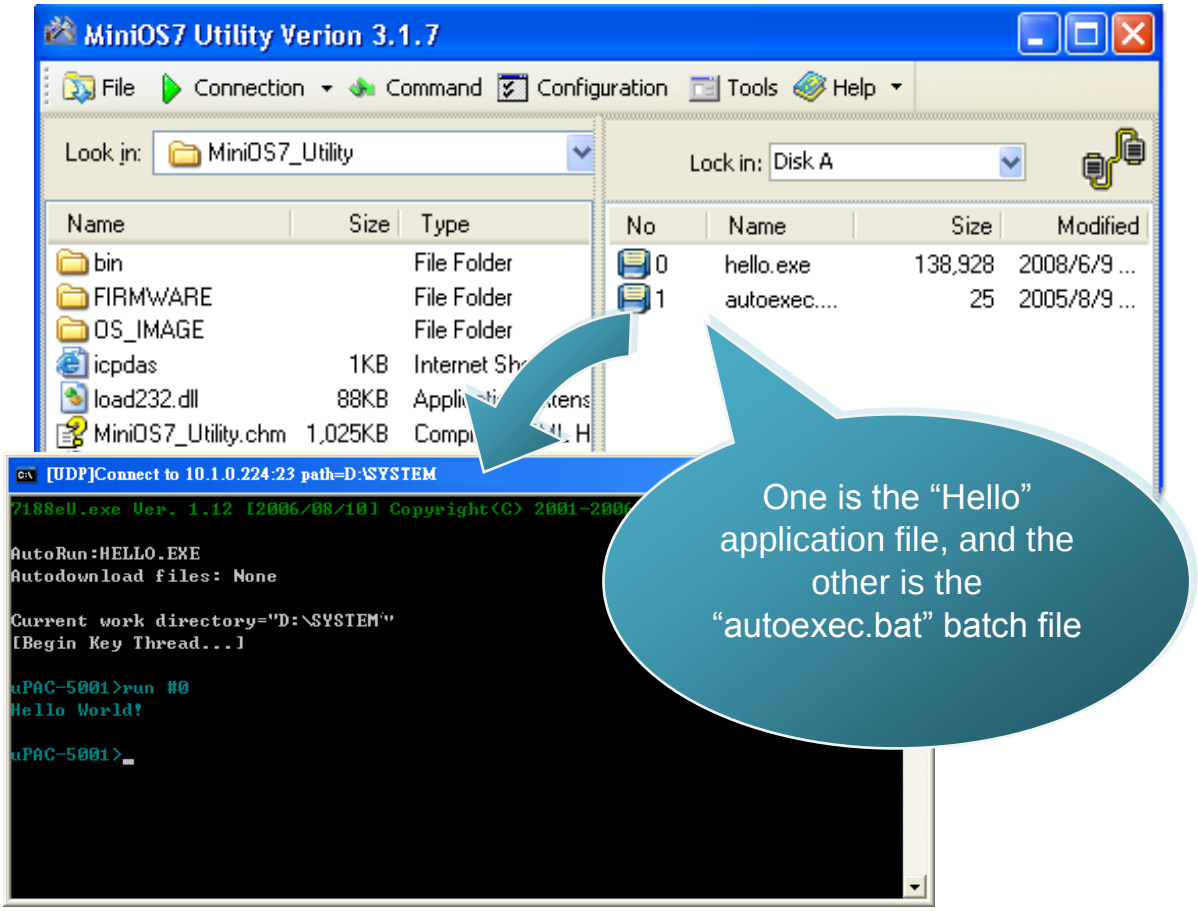
The command can be either the file name “HELLO.exe” (run the specified file) or “runexe” (run the last exe file)

iii. Save the file as autoexec.bat



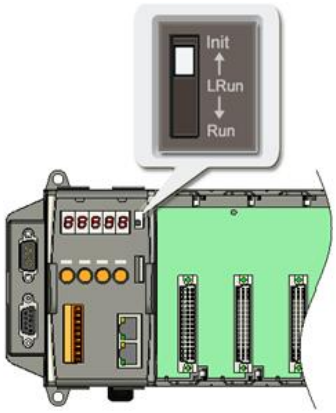
Step 2: Upload programs to iPAC-8000 using MiniOS7 Utility

For more detailed information about this process, please refer to section “2.4.1. Establishing a connection”



Tips & Warnings

Before restaring the module for settings to take effect, you must firstly turn the switch to “Init” position.



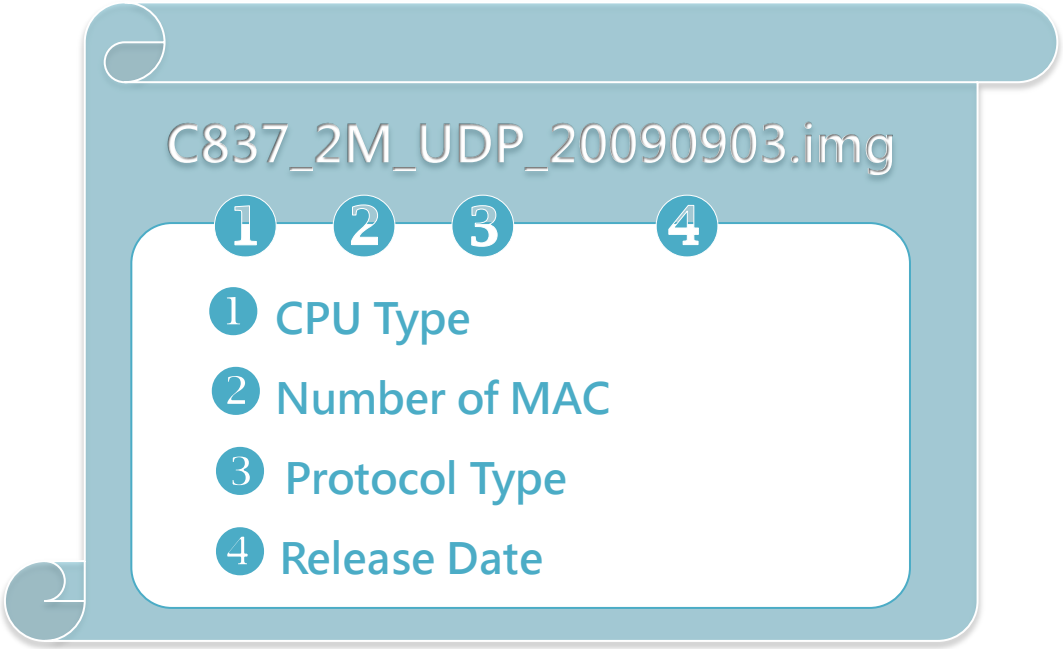
2.5. Updating iPAC-8000 OS image

ICP DAS will continue to add additional features to iPAC-8000 in the future, we advise you periodically check the ICP DAS web site for the latest update to iPAC-8000.

Step 1: Get the latest version of the iPAC-8000 OS image

The latest version of the iPAC-8000 OS image can be obtained from:

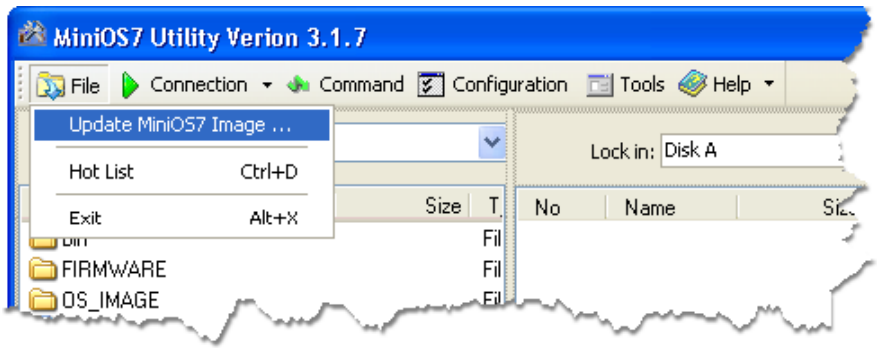
CD:\NAPDOS\iPAC8000\OS_Image
http://ftp.Icpdas.com/pub/cd/8000cd/napdos/ipac8000/os_image/



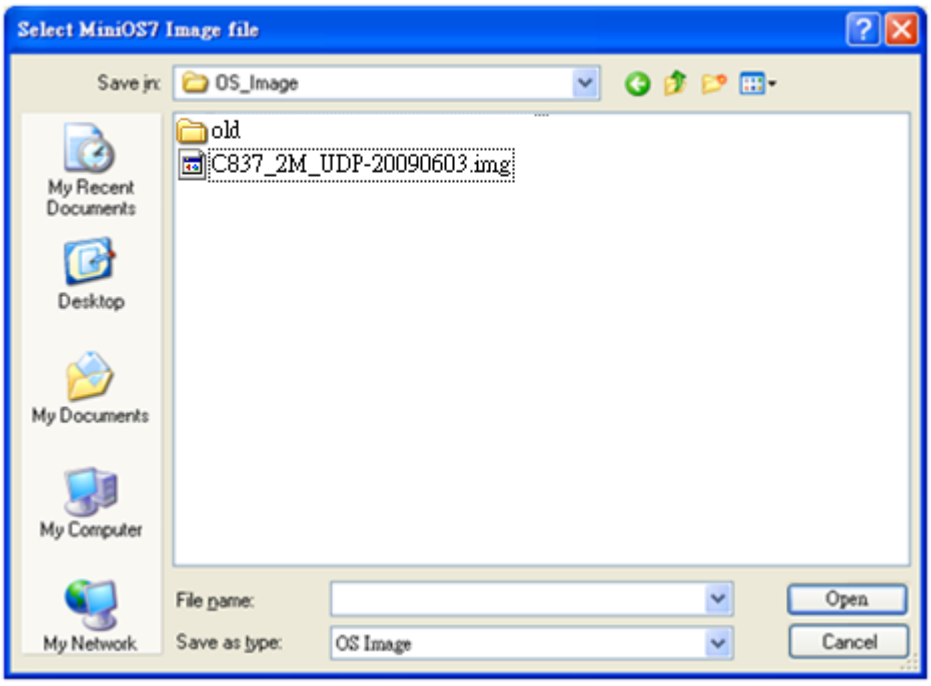
Step 2: Establish a connection

For more detailed information about this process, please refer to section “2.4.1. Establishing a connection”

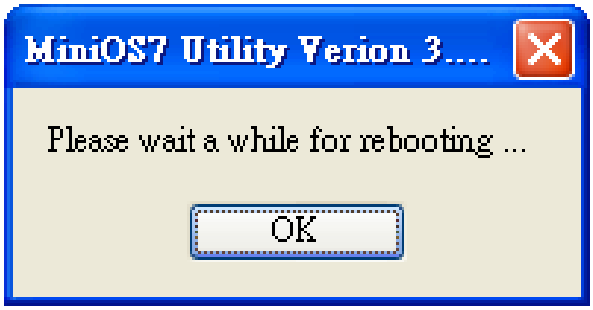
Step 3: Click the “Update MiniOS7 Image …” from the “File” menu



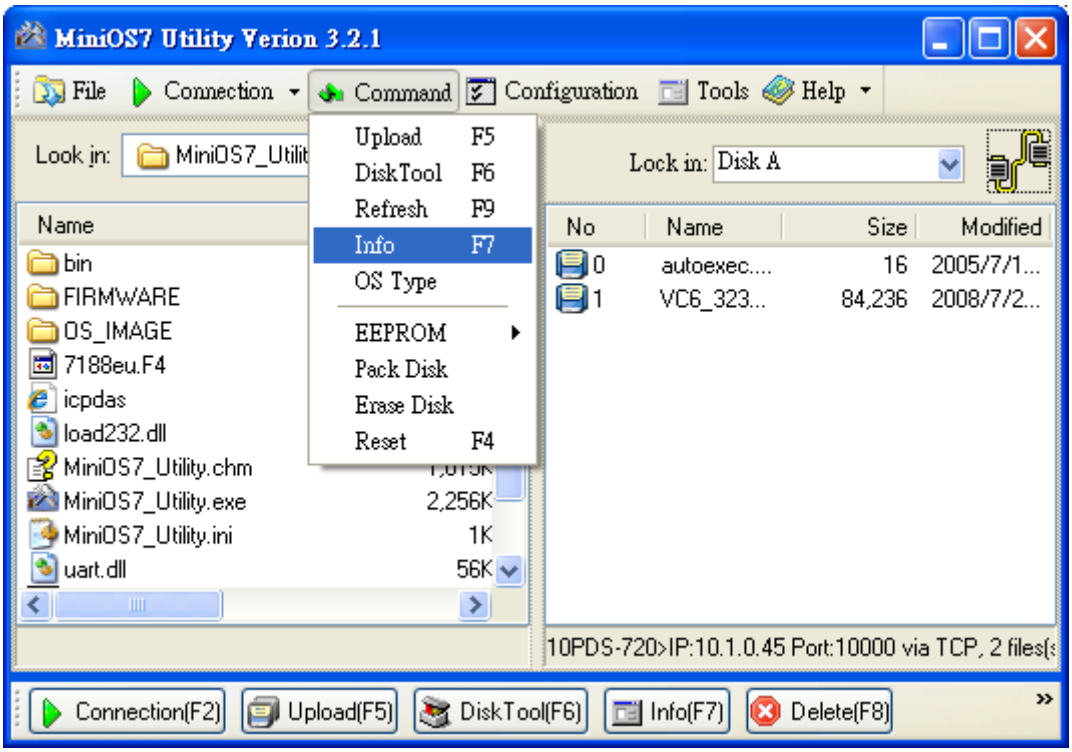
Step 4: Select the latest version of the MiniOS7 OS image



Step 5: Click the “OK”



Step 6: Click the “Info” from the “Command” menu to check the version of the OS image



3. “Hello World” - Your First Program

When you learn every computer programming language you may realize that the first program to demonstrate is "Hello World", it provides a cursory introduction to the language's syntax and output.

Below are step-by-step instructions on how to write your first iPAC-8000 program.

3.1. C Compiler Installation

C is prized for its efficiency, and is the most popular programming language for writing applications.

Before writing your first iPAC-8000 program, ensure that you have the necessary C/C++ compiler and the corresponding functions library on your system.

The following is a list of the C compilers that are commonly used in the application development services.

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

We recommend that you use Borland C++ compiler as the libraries have been created on the companion CD.

Tips & Warnings

Before compiling an application, you need to take care of the following matters.

- ➤ 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.).

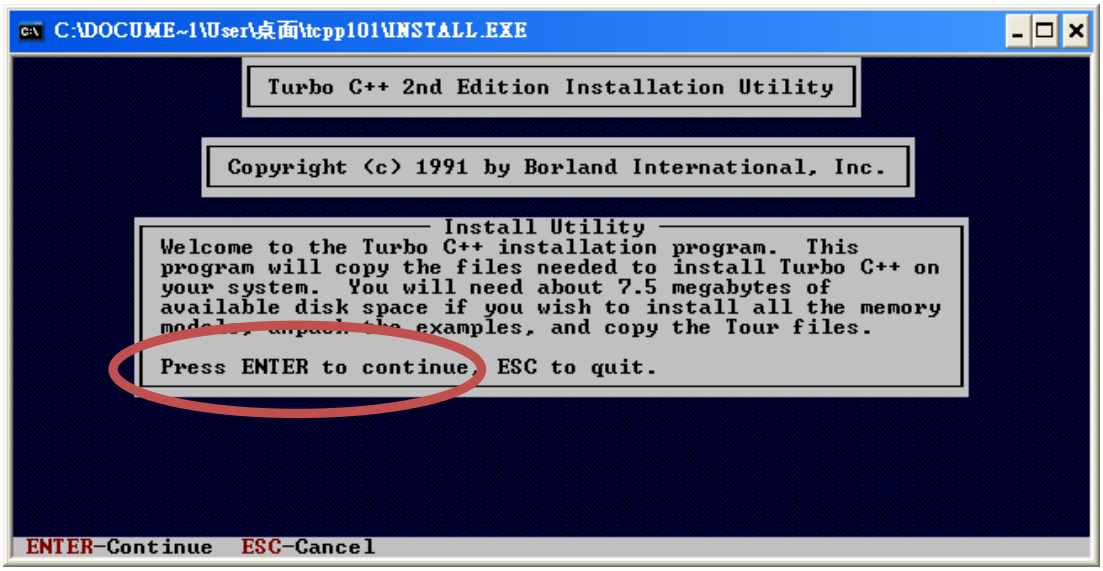
3.1.1. **Installing the C compiler**

If there is no compiler currently installed on your system, installation of the compiler should be the first step.

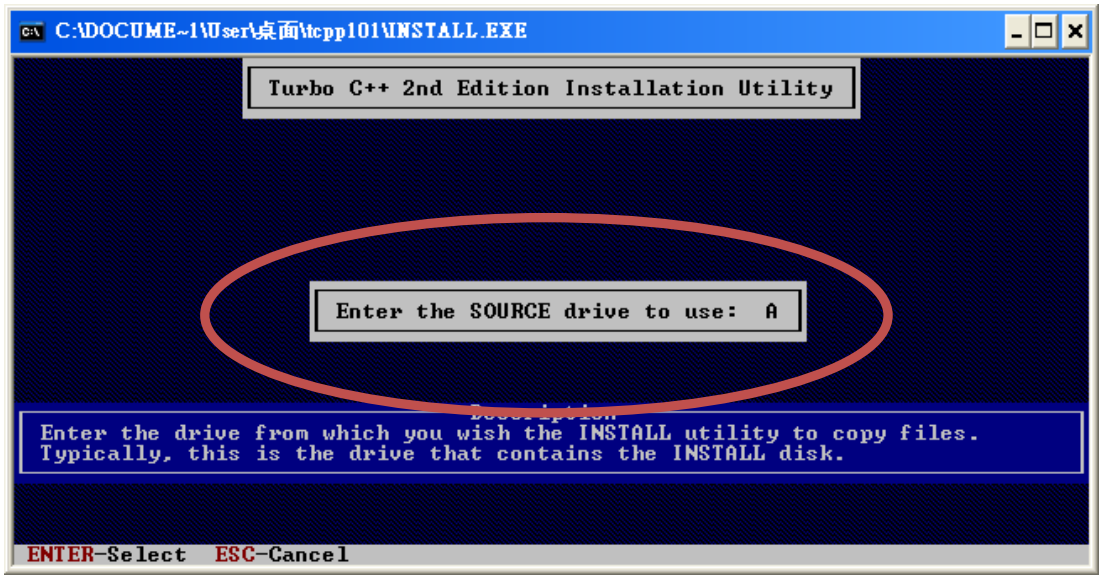
Below are step-by-step instructions for guiding you to install Turbo C++ Version 1.01 on your system.

Step 1: Double click the Turbo C++ executable file to start setup wizard

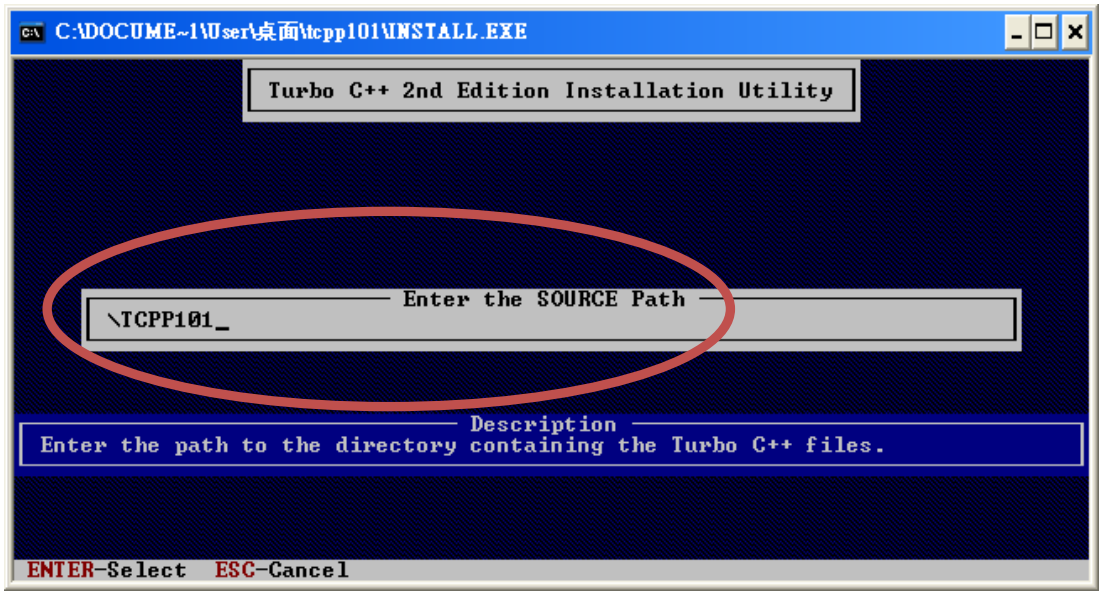
Step 2: Press “Enter” to continue



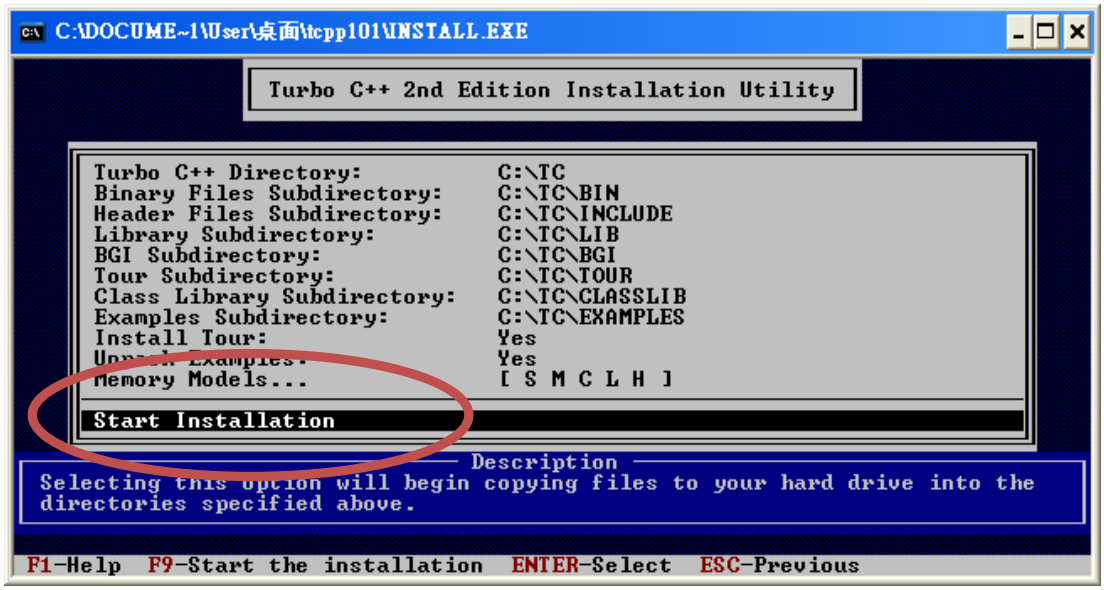
Step 3: Enter the letter of the hard drive you wish to install the software



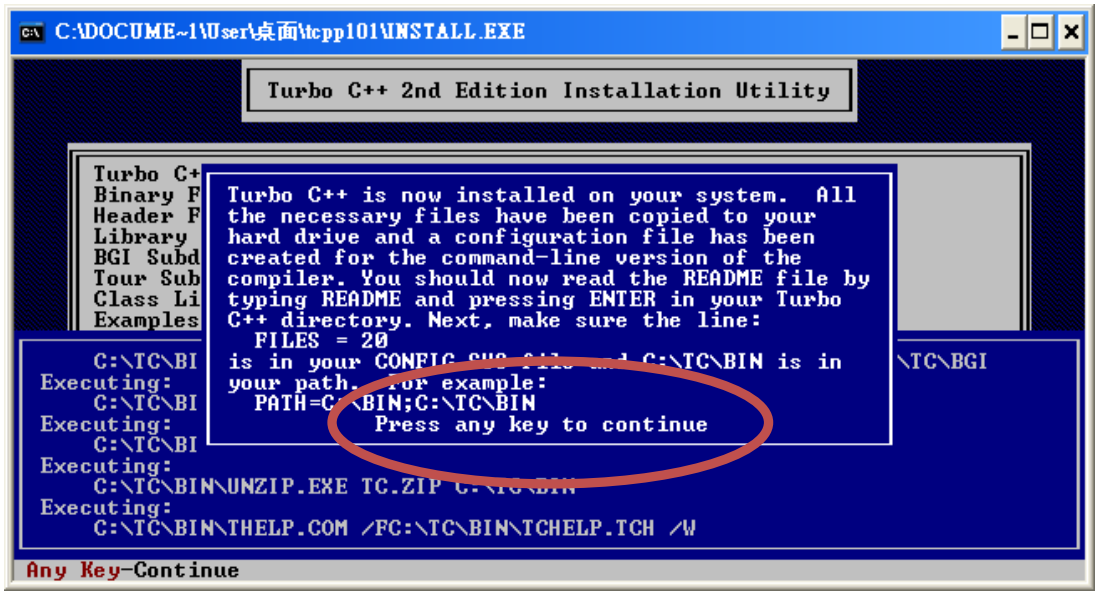
Step 4: Enter the path to the directory you wish to install files to



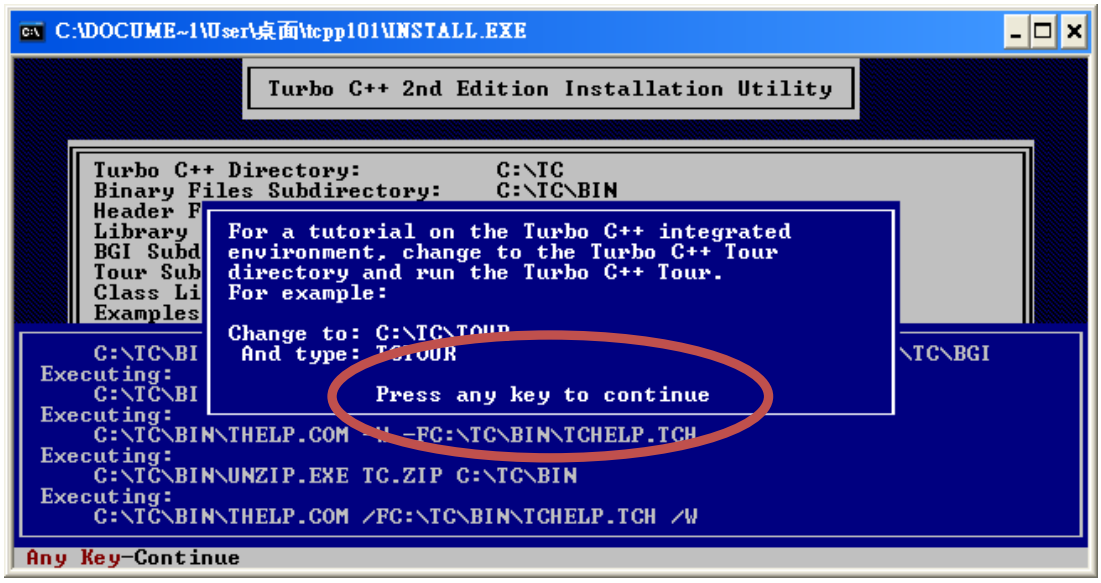
Step 5: Select “Start Installation” to begin the install process



Step 6: Press any key to continue



Step 7: Press any key to continue

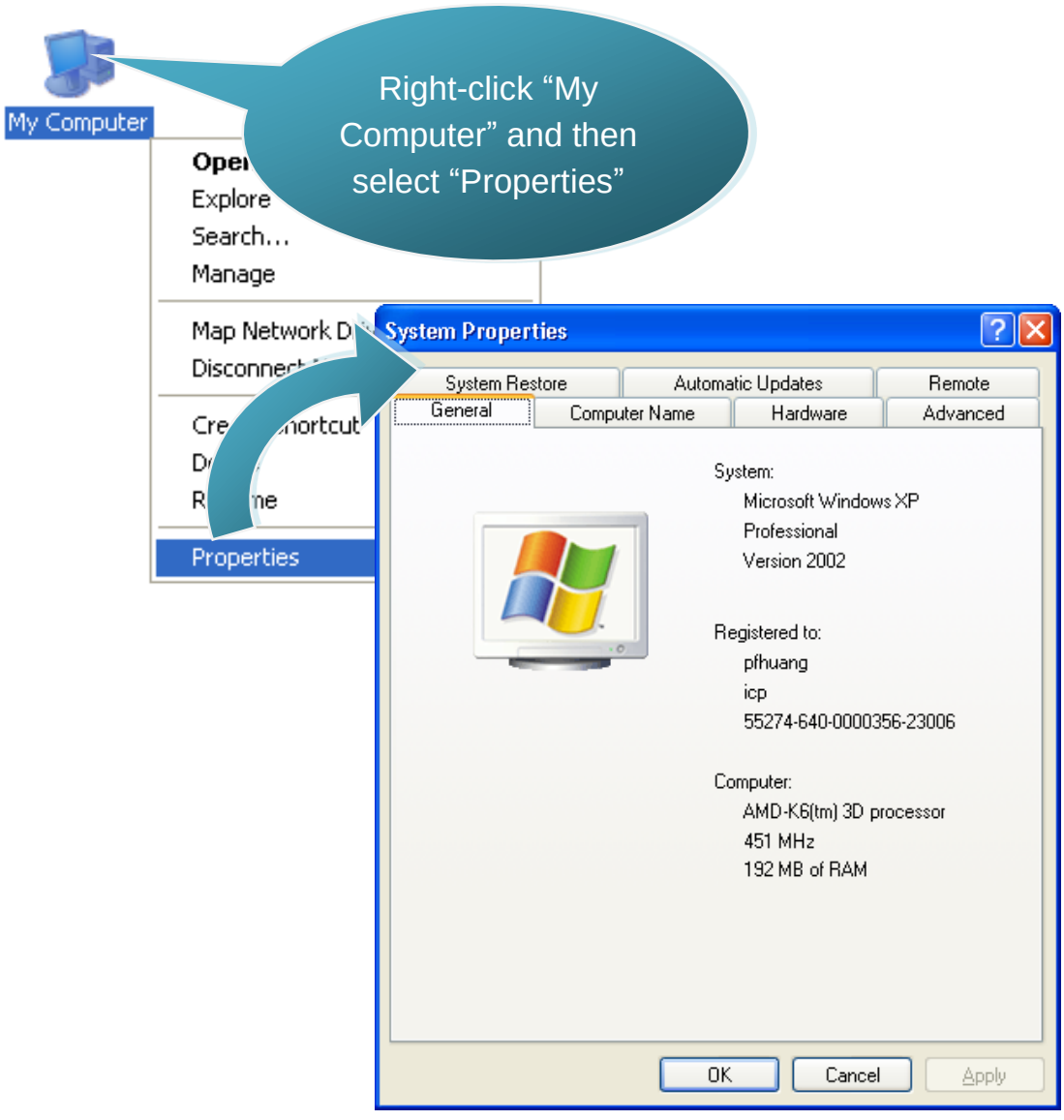


Step 8: Installation is complete

3.1.2. **Setting up the environment variables**

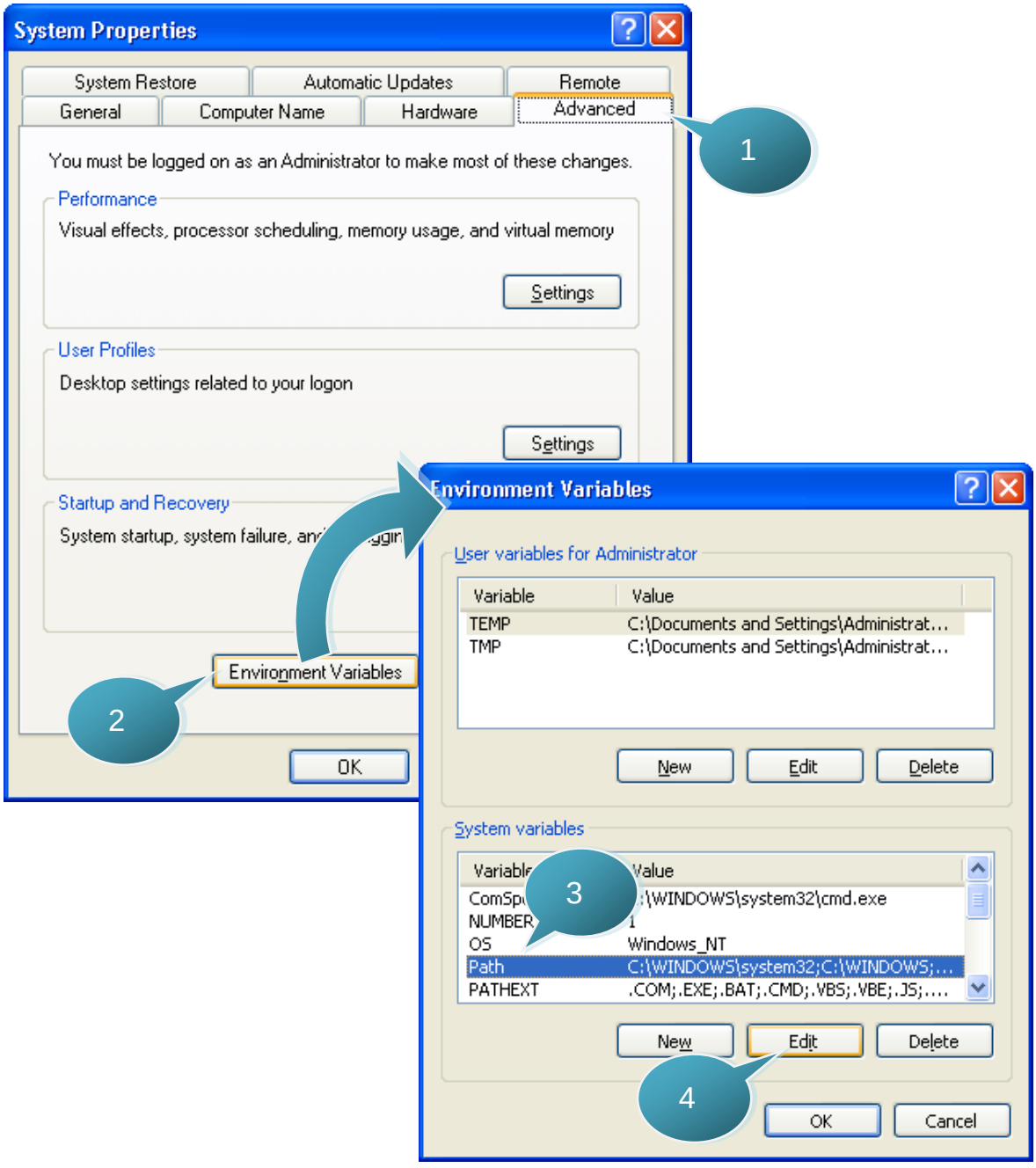
After installing the compiler, several compilers will be available from the Windows Command line. You can set the path environment variable so that you can execute this compiler on the command line by entering simple names, rather than by using their full path names.

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. **iPAC-8000 APIs**

There are several APIs for customizing the standard features and integrating with other applications, devices and services.

Before creating the application, ensure them that you have installed. If they are not installed, please refer to “section 2.2. Software Installation”.

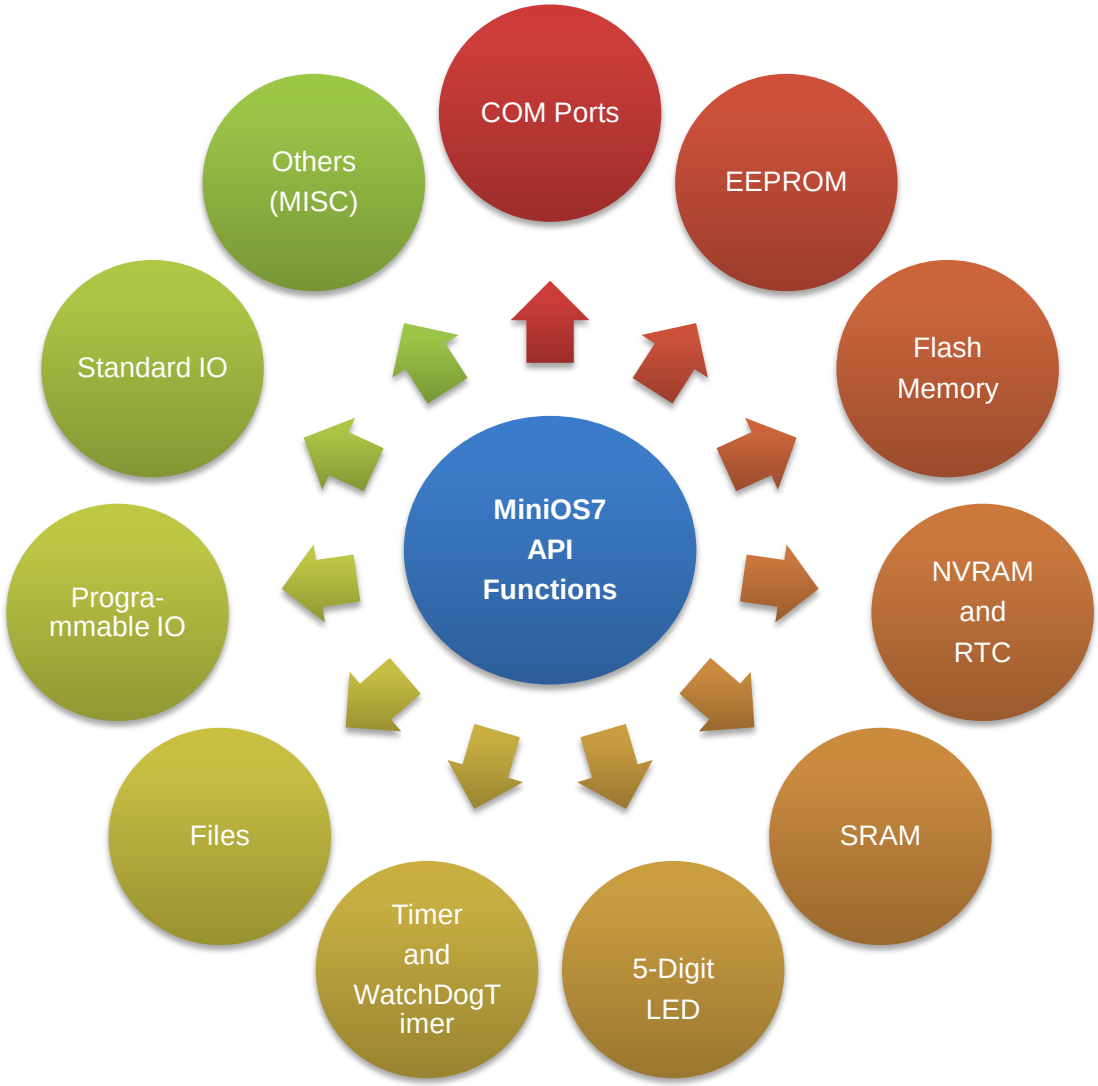
The following introduces the core API, MiniOS7 API, which is integrated into the iPAC-8000 API set.

Functions Library － 8000a.lib

This file contains the MiniOS7 API (Application Programming Interface) and has hundreds of pre-defined functions related to iPAC-8000

Header File － 8000a.h

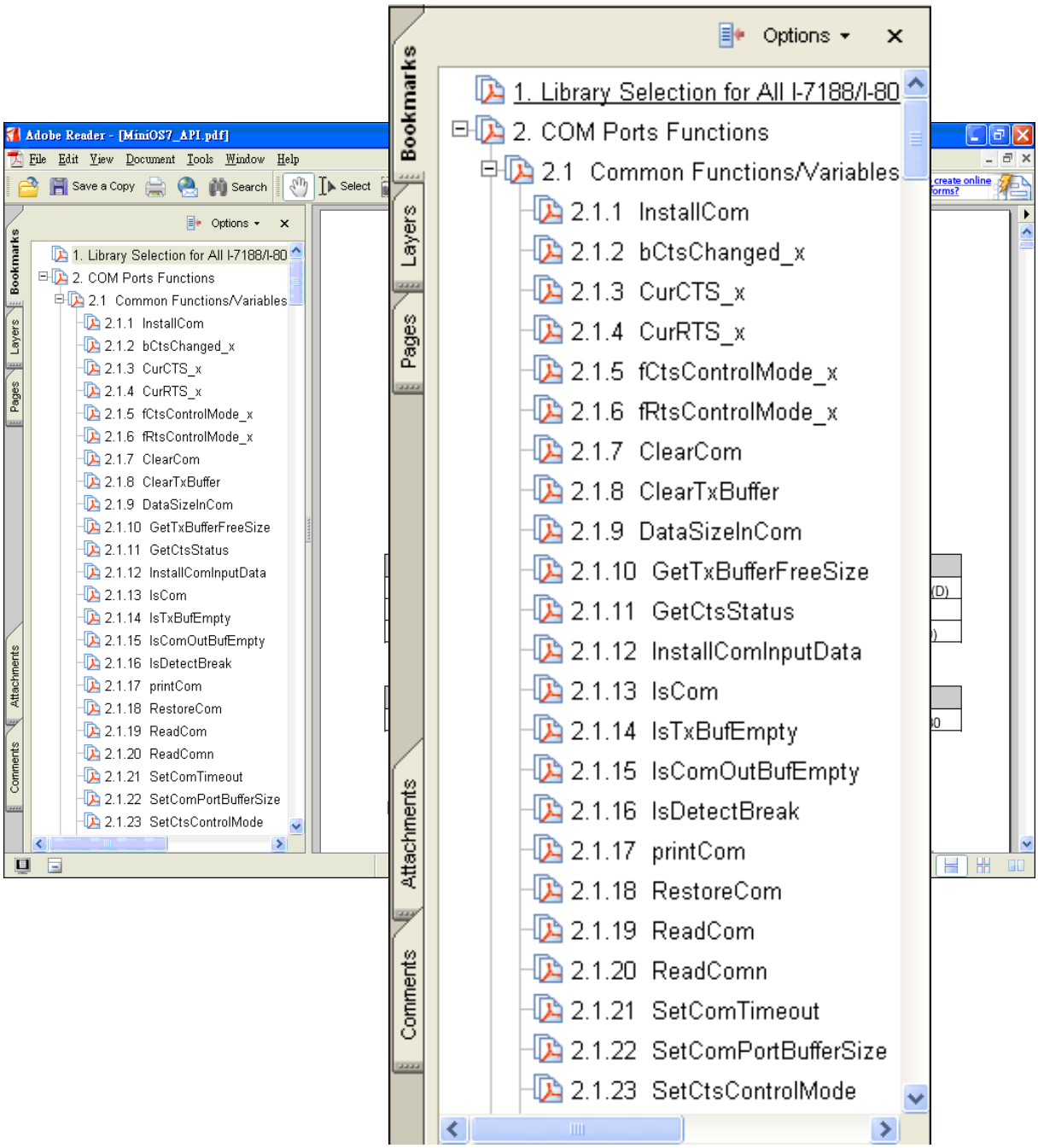
This file contains the forward declarations of subroutines, variables, and other identifiers used for the MiniOS7 API.



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/



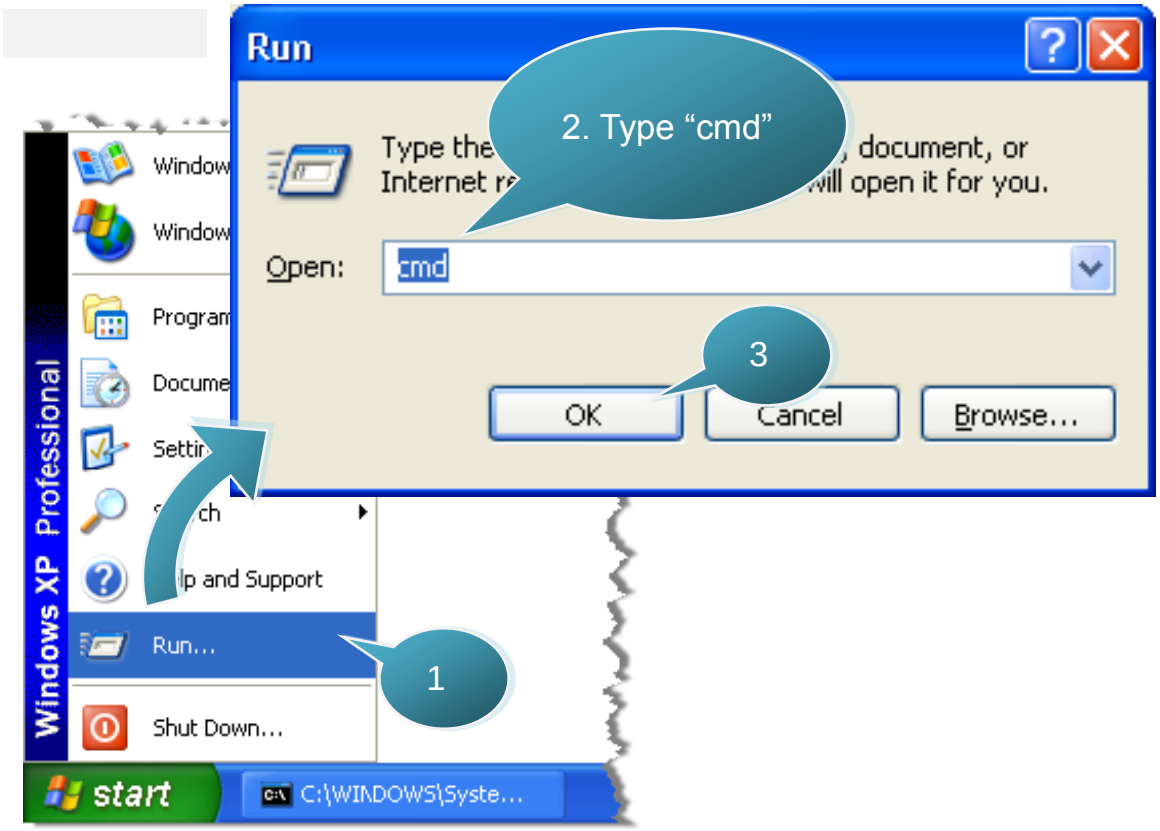
3.3. First Program in iPAC-8000

Here we assume you have installed the Turbo C++ 1.01 (as the section “3.1. C Compiler Installation”) and the iPAC-8000 APIs (as the section “2.2. Software Installation”) under the C driver root folder.

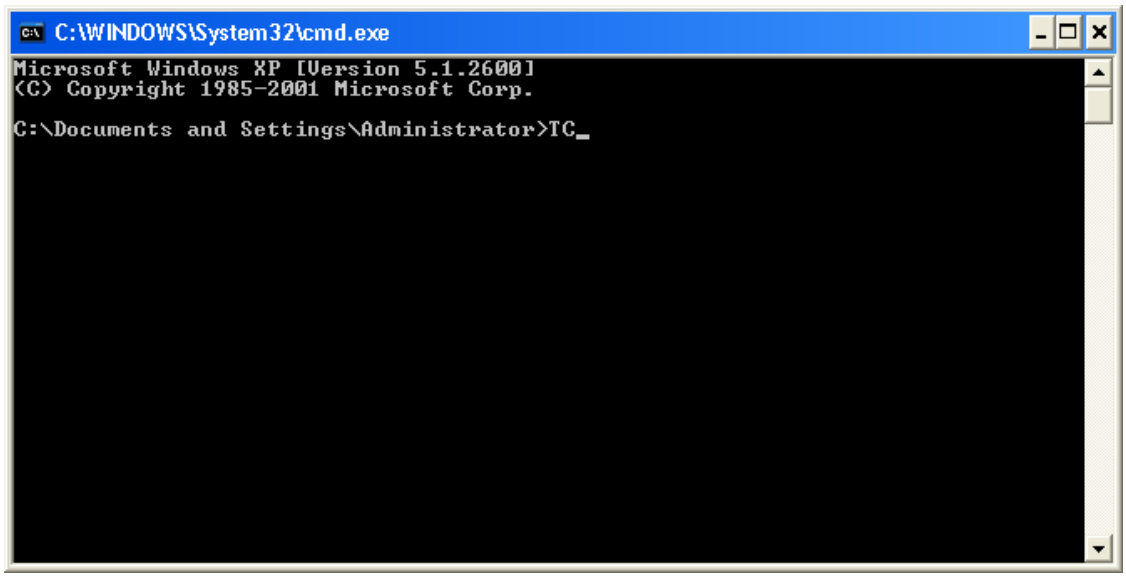
Below are step-by-step instructions for writing your first program.

Step 1: Open a MS-DOS command prompt

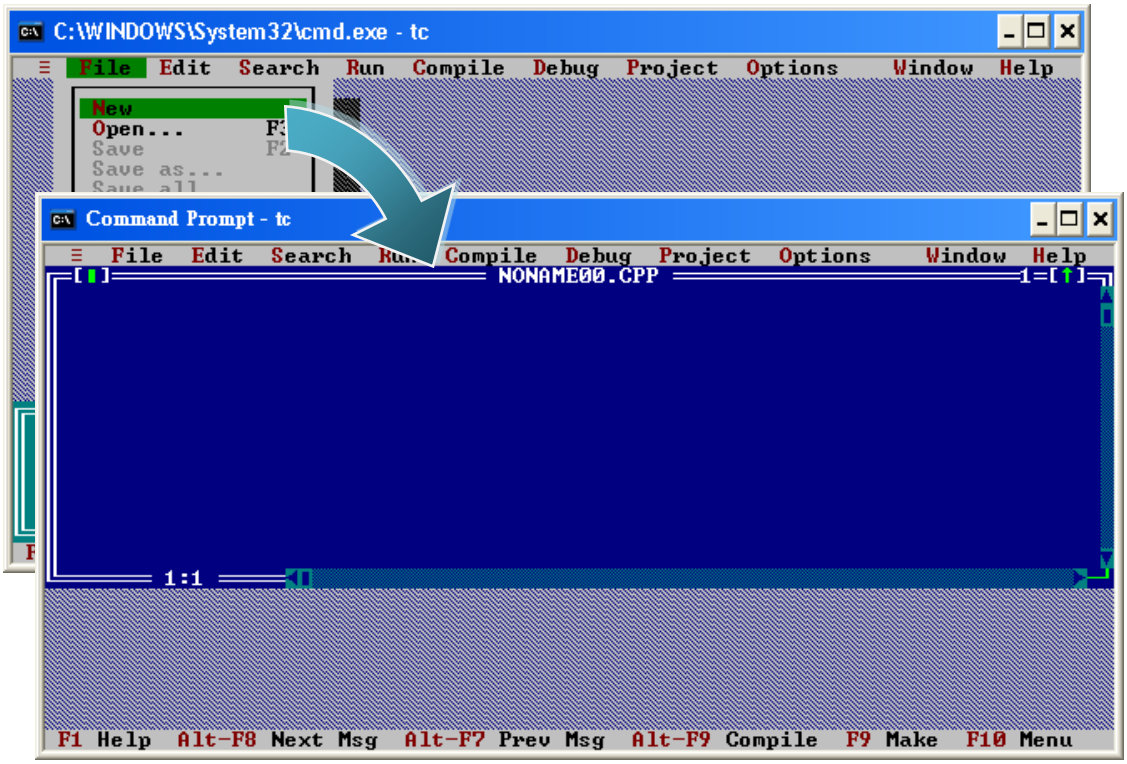
i. Select “Run” from the “Start” menu
ii. On the “Run” dialog box, type “cmd”
iii. Click the “OK” button



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

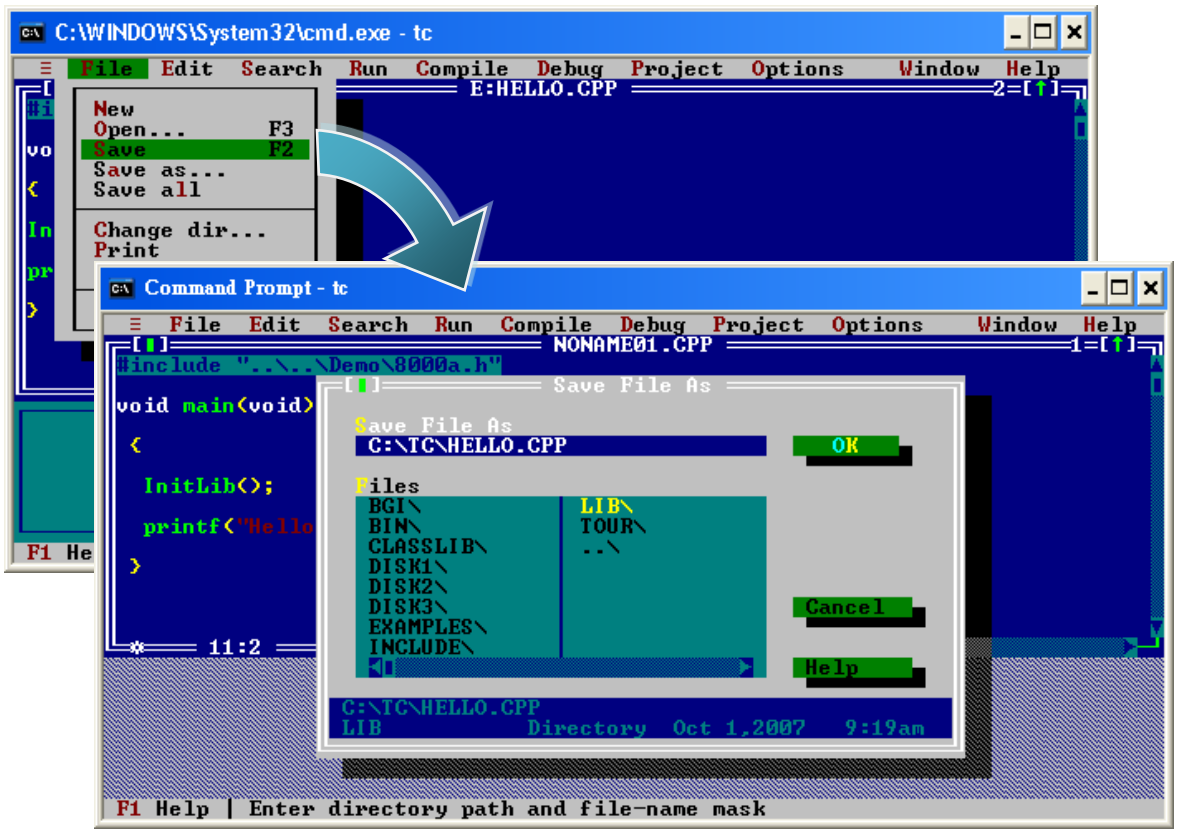


Step 4: Type the following code. Note that the code is case-sensitive

```
#include “..\..\Demo\basic\Lib\8000a.h”
/* Include the header file that allows 8000a.lib functions to be used */

void main(void)
{
  InitLib();      /* Initiate the 8000a library */
  Print(“Hello 8000!\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”



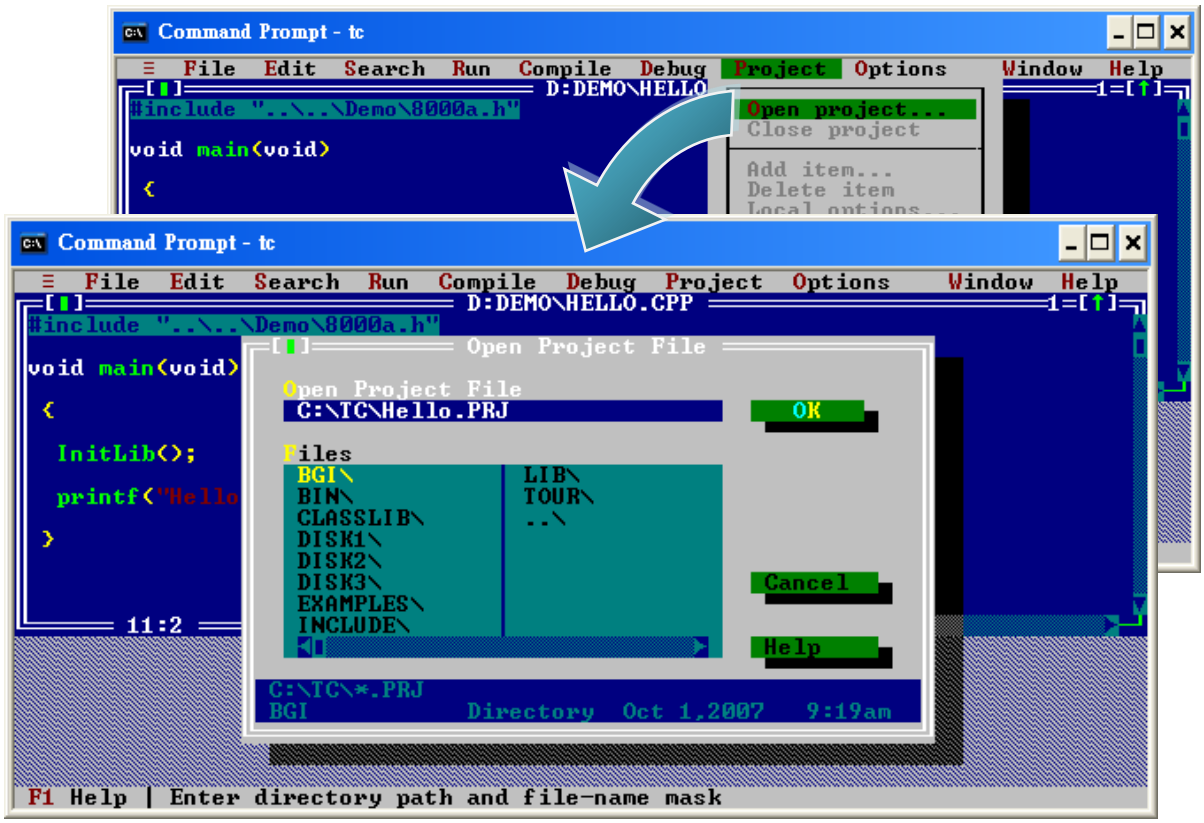
Tips & Warnings

You can write the code as shown below with your familiar text editor or other tools; please note that

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”
iv. Select “Add”
v. Select “Done” to exit



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

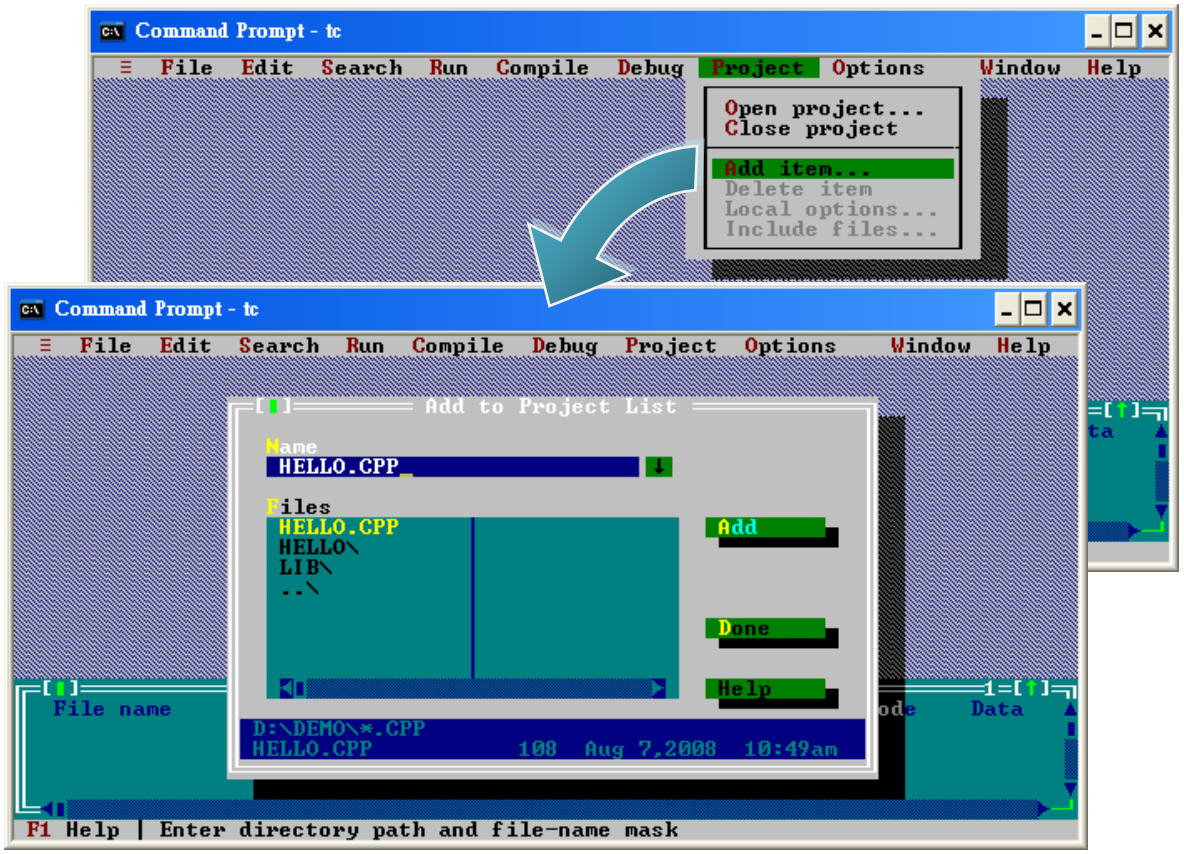
i. Select “Add item…” from the “Project” menu

ii. Type “ *.LIB ” to display a list of all available function libraries

iii. Choose the function libraries you require

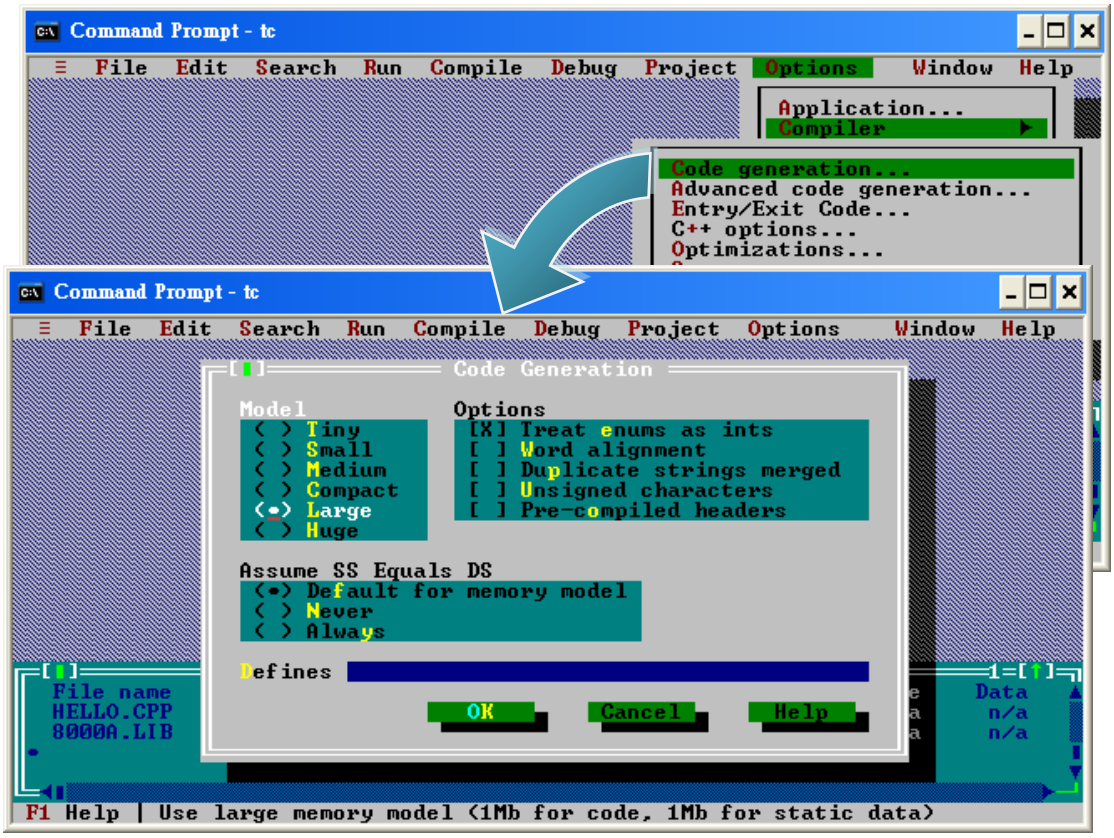
iv. Select “Add”

v. Select “Done” to exit



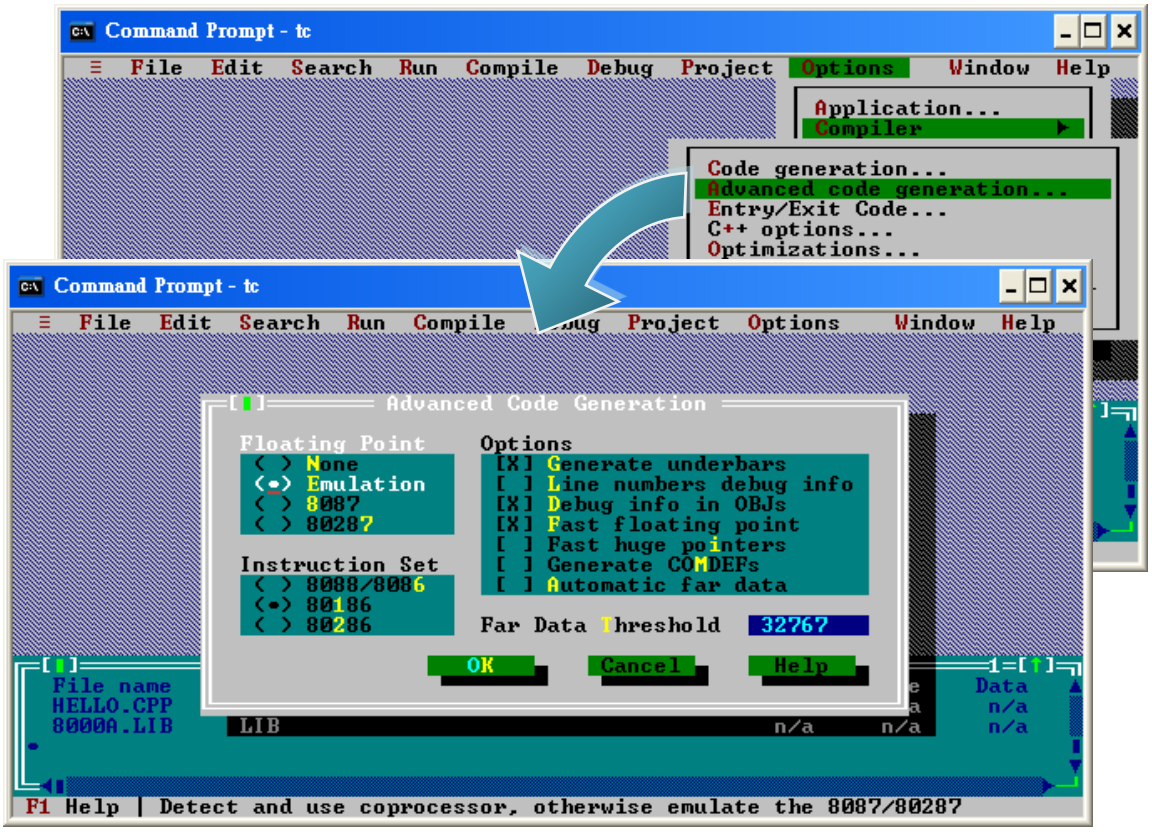
Step 9: Set the memory model to large

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



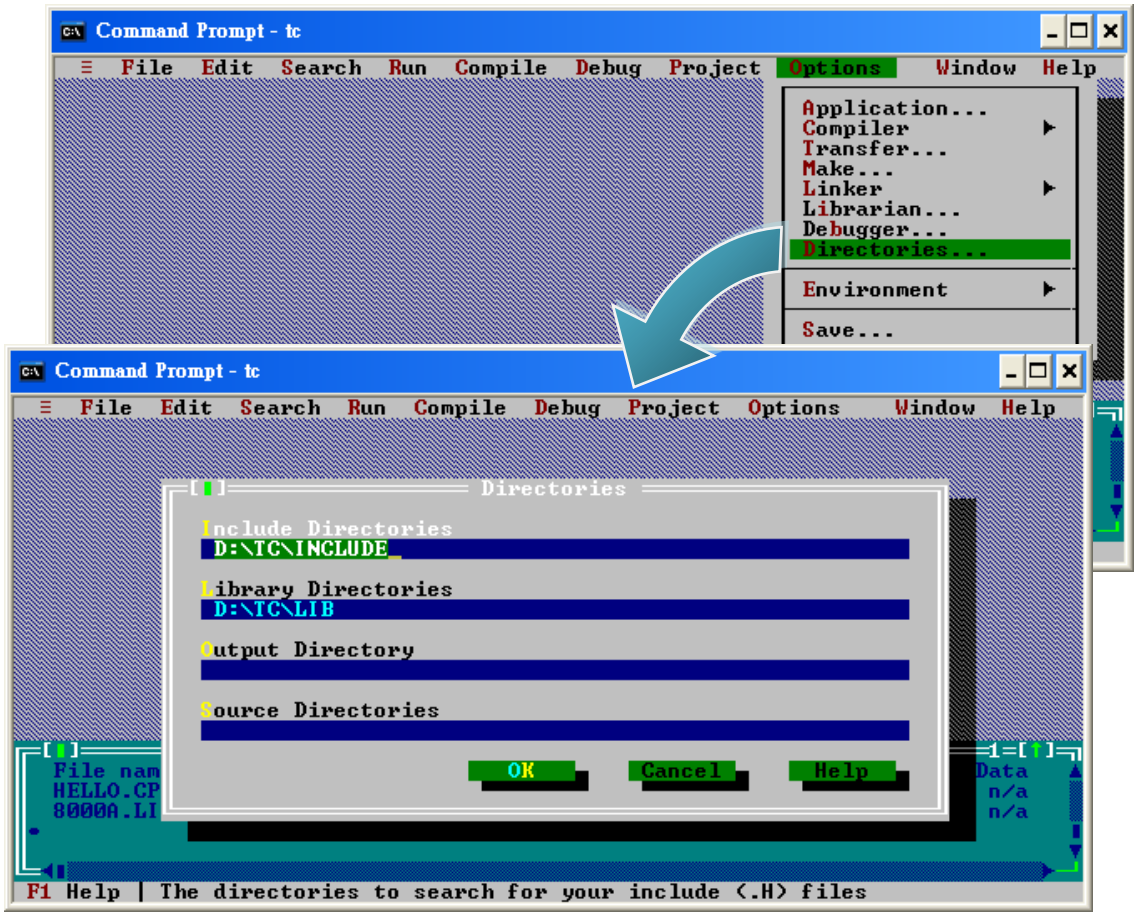
Step 10: Set the memory model to large

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”

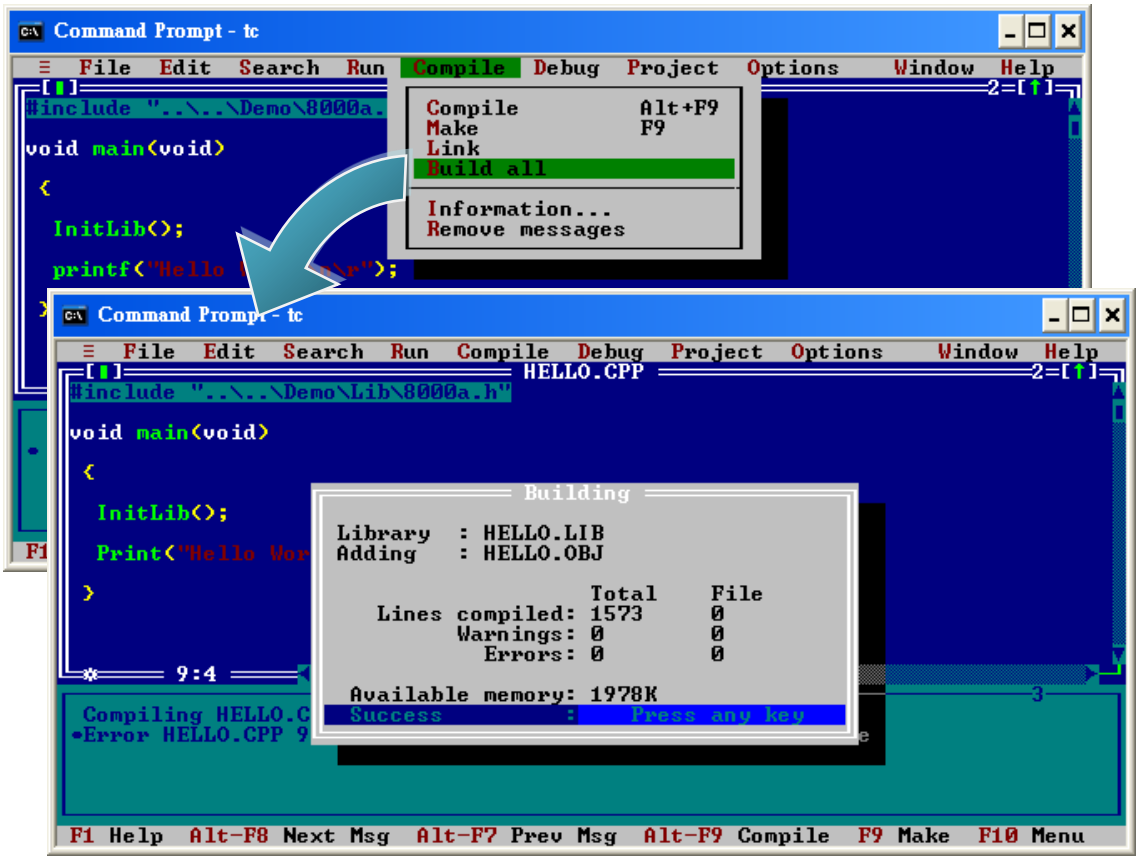


Step 11: Set the memory model to large

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”

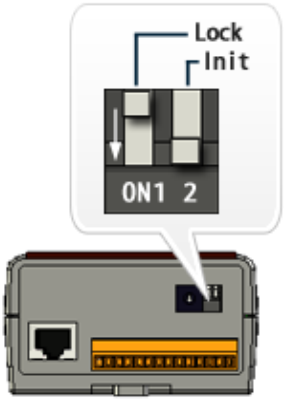


Step 12: Select “Build all” from the “Compile” menu to build the project

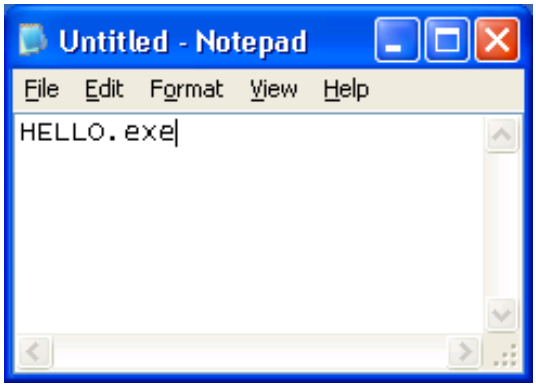


Step 13: Configure the operating mode

Make sure the switch of the Lock placed in the “OFF” position, and the switch of the Init placed in the “ON” position.



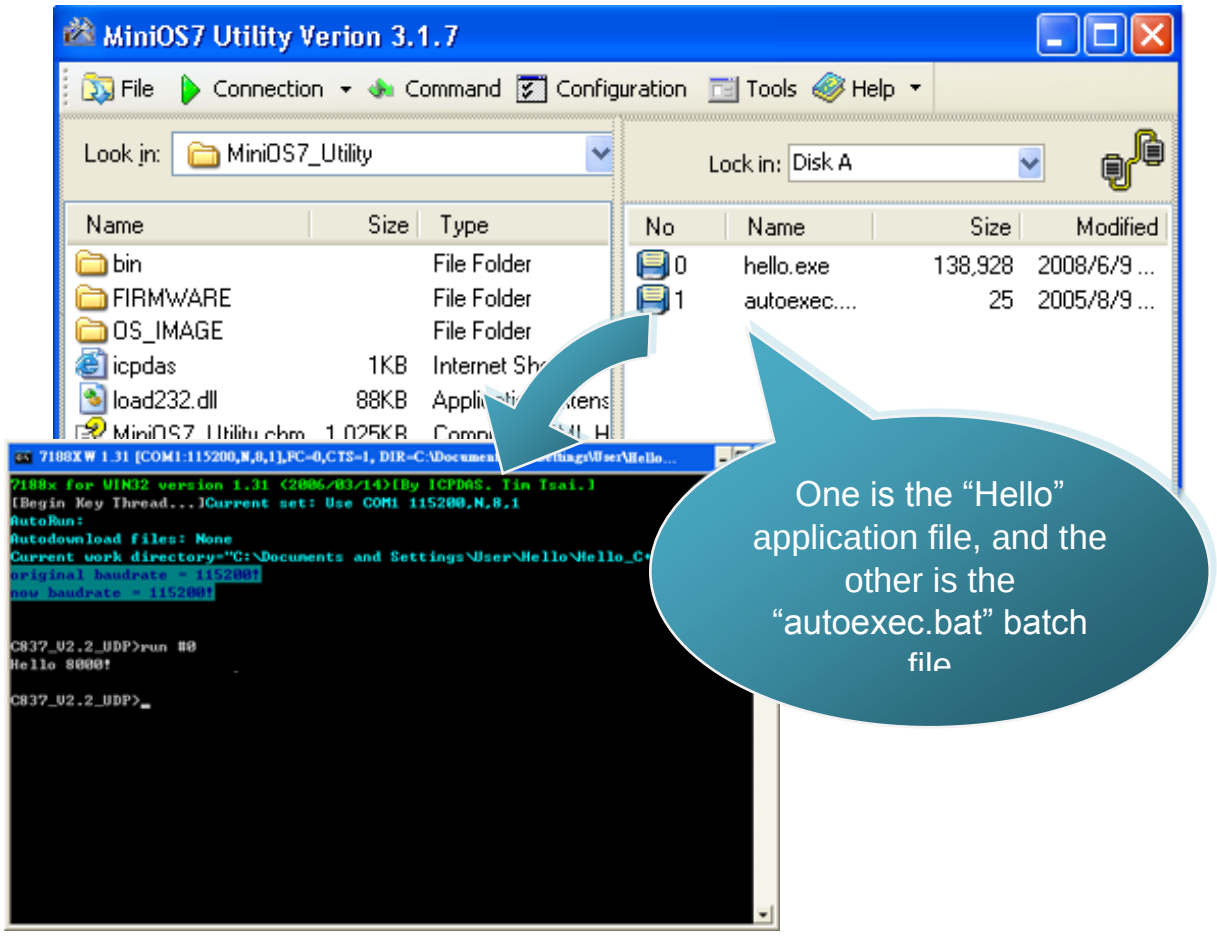
Step 14: Create an autoexec.bat file



i. Open the “Notepad”
ii. Type the “HELLO.exe”
iii. Save the file as autoexec.bat

Step 15: Upload programs to iPAC-8000 using MiniOS7 Utility

For more detailed information about this process, please refer to section “2.4.1. Establishing a connection”



4. APIs and Demo References

There are several APIs and demo programs that have been designed for iPAC-8000. You can examine the APIs and demo source code, which includes numerous functions and comments, to familiarize yourself with the MiniOS7 APIs and quickly develop your own applications quickly by modifying these demo programs.

The following table lists the APIs grouped by functional category.

API Description	Header File	Library
CPU driver	8000a.h	8000a.lib
DCON driver	DCON_FUN.h	DCON_8KL.LIB
Ethernet driver	Tcpip32.h	tcp2dm32.lib
Framework driver	MFW.h	MFW09313.LIB
microSD driver	microSD.h	SD_V100.LIB
Xserver driver	VXCOMM.H	V8a_3230.lib

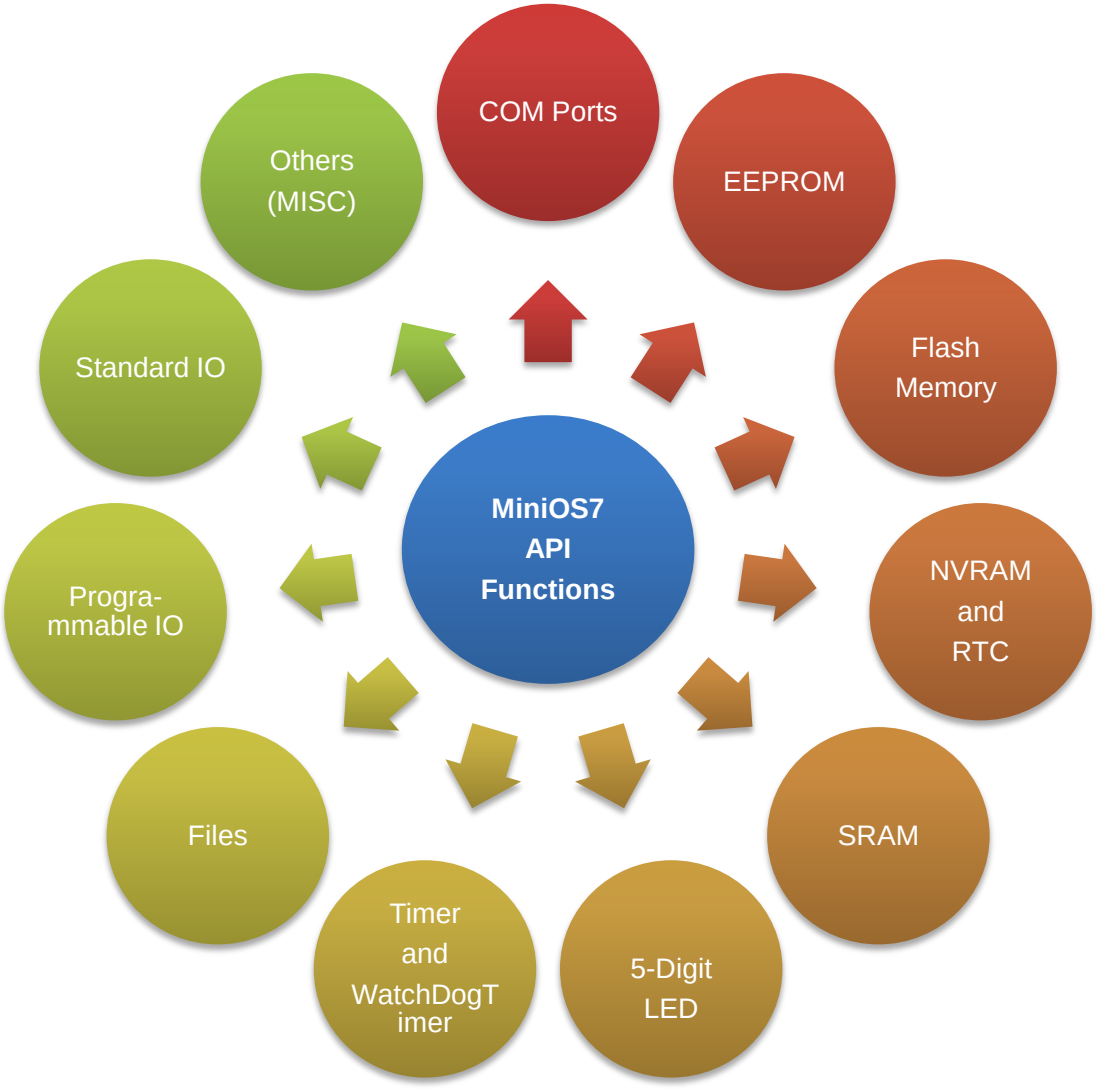
The following introduces the core API, MiniOS7 API, which is integrated into the iPAC-8000 API set.

Functions Library – 8000a.lib

This file contains the MiniOS7 API (Application Programming Interface) and has hundreds of pre-defined functions related to iPAC-8000

Header File – 8000a.h

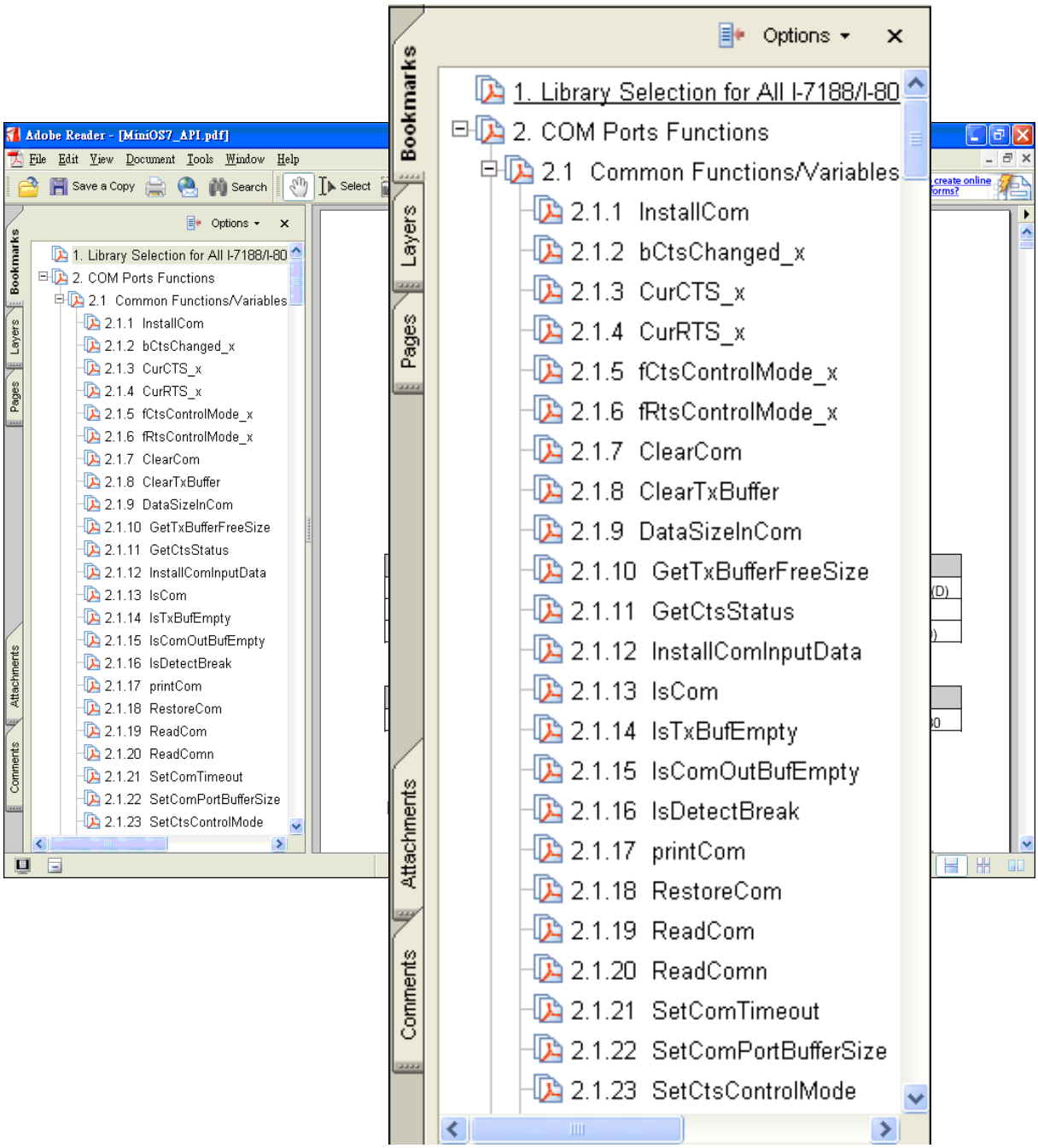
This file contains the forward declarations of subroutines, variables, and other identifiers used for the MiniOS7 API.



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

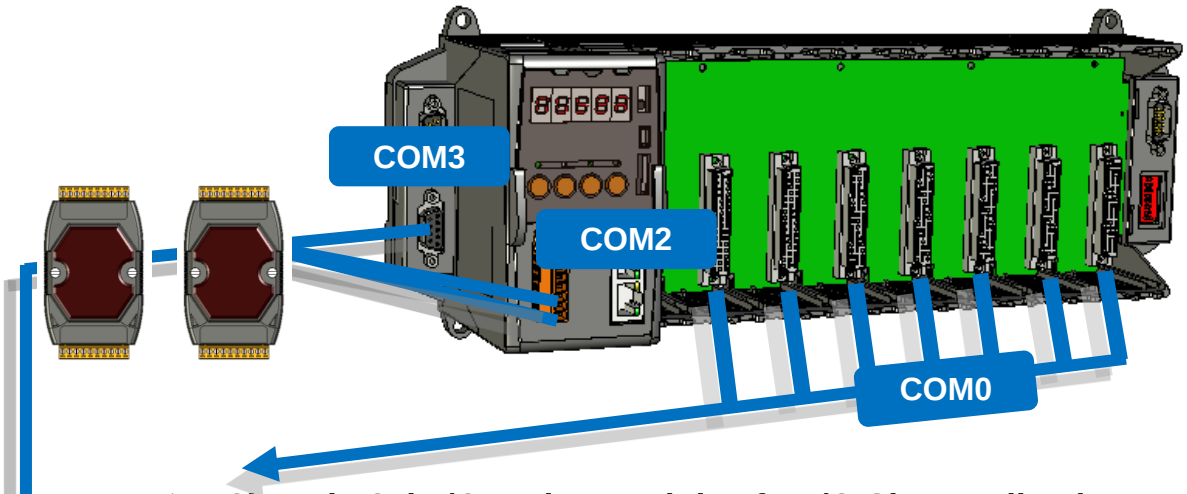
http://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/document/



The following table lists the demo programs grouped by functional category.

➤ 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.
Smmi	SystemKey	Shows how to operate the systemkey function simply and easily.
	Led	Shows how to control the red LED and 7-segment display.
Memory	Battery_Backup_SRAM	Shows how to read or write to the 256K/512K byte battery backup.
DateTime	DateTime	Shows how to read and write the date and time from the RTC.
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.
For more information about these demo programs, please refer to: **CD:\ NAPDOS\iPAC8000\ Demo\Basic** **http://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/**		



➢ I-8k and I-87k I/O series modules for I/O Slot Applications

Folder	Demo	Explanation
IO_in_Slot	8K_DI	This demo program is used by 8K series DI modules, such as 8040, 8051., etc.
	8073	This demo program is used for 8073 General Functions.
	87K_DI	This demo program is used by 87K series DI modules in Com0, such as 87040, 87051, etc.
	87024	This demo program is used by the 87024 AO module.
… more demo programs		

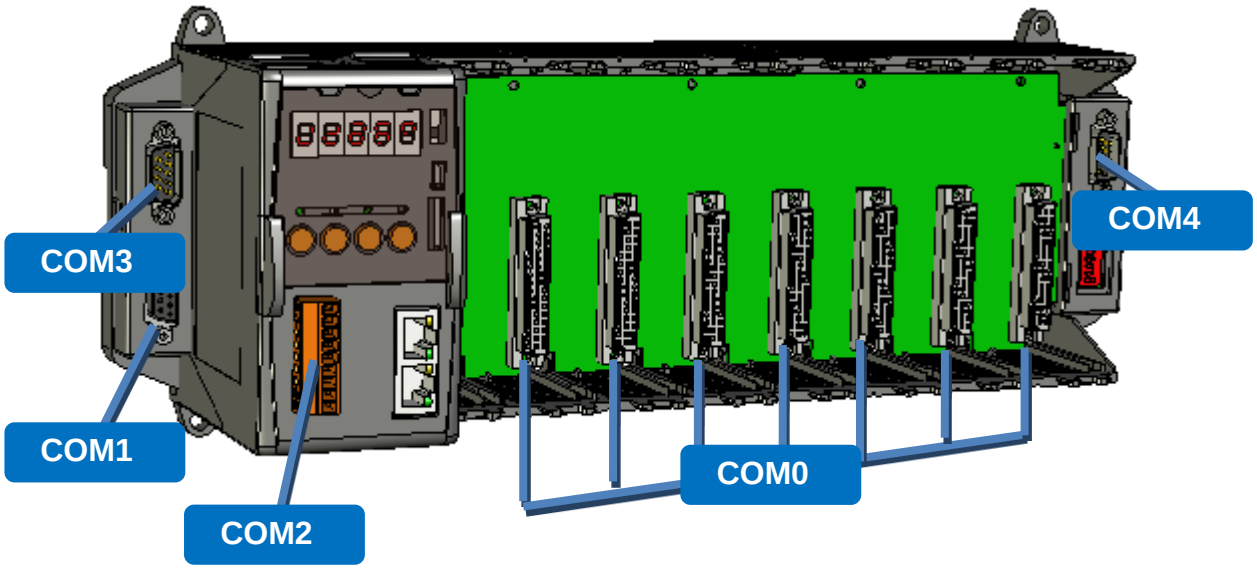
➢ I-7K and I-87k series modules for RS-485 Network Applications

Folder	Demo	Explanation
7K 87K_for_Com	7K87K_DI_for_Com	"COM Port" can be used to connect and control I-7k or I-87k series modules. ■ For iPAC-8000 module and can use, COM2, COM3. ■ For iPAC-8000 module and (CPU 40 and 80M) can use, COM3, COM4.
	7K87K_DO_for_Com	
	7K87K_AI_for_Com	
	AO_22_26_for_Com	
	AO_024_for_Com	

For more information about these demo programs, please refer to:
CD:\ NAPDOS\iPAC8000\ Demo\Basic\7K87K_for_COM
http://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/7K87K_for_COM/

4.1. API for COM Port

The iPAC-8000 provides five built-in COM ports.



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

Tips & Warnings

(C style) Standard COM port functions only can be used with the COM1, if you use the COM1 port, you'll have the alternative of MiniOS7 COM ports functions or (C style) Standard COM port functions. If you choose the ones, then another cannot be used.

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

Types of Functions	COM Port	Buffer	Functions				
MiniOS7 COM port	1, 2, etc.	1 KB	1 KB	IsCom()	ToCom()	ReadCom()	printCom()
(C style) Standard COM port	1	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 “8000a.h”

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

 InitLib();      /* Initiate the 8000a 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 */
}
```

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 “8000a.h”

void main(void)
{
 int i;

 /* Initiate the 8000a 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);
}
```

4.1.3. API for standard COM port

The standard COM port is used to upload program from PC to the iPAC-8000.

Tips & Warnings

(C style) Standard COM port functions only can be used with the COM1 port, the following configurations of the COM1 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 “8000a.h”

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

 InitLib();      /* Initiate the 8000a 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 “8000a.h”

void main(void)
{
 int i;

 /* Initiate the 8000a library */
 InitLib();
 for(i=0;i<10;i++)
 {
  Print(“Test %d\n\r”,i);
 }
}
```

4.1.4. **Port functions Comparison**

For example, learning to show the ASCII code.

MiniOS7 COM port functions	Standard COM port functions

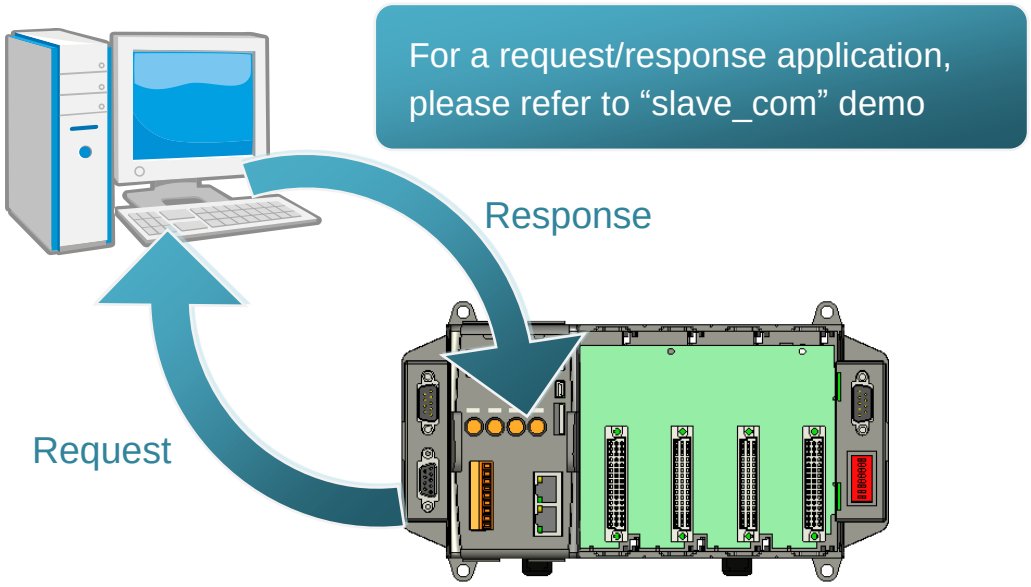
#include<stdio.h>
#include “8000a.h”
 	
void main(void)
{
unsigned char item;
 	
InitLib();
InstallCom(1, 115200, 8, 0, 1);
printCom(1,”Hits any key.\n”);
printCom(1,”Hit the ESC to exit!\n”);
 	
for(;;)
{
if(IsCom(1))
{
item=ReadCom(1);
if(item==’q’)
{
return;
}
else
{
printCom(1,”----------\n\r”);
	#include<stdio.h>
	#include “8000a.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”);

```
    printCom(1,”char:”);

    ToCom(1,item);
    printCom(1,"\n\rASCII(%c)\n\r”,item);
    printCom(1,“Hex(%02X)\n\r”,item);
   }
  }
 }
 Delay(10);
 RestoreCom(1);
}
```

```
    Print(“char:“);

    Putch(item);
    Print("\n\rASCII(%c)\n\r”,item);
    Print(“Hex(%02X)\n\r”,item);
   }
  }
 }


}
```

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

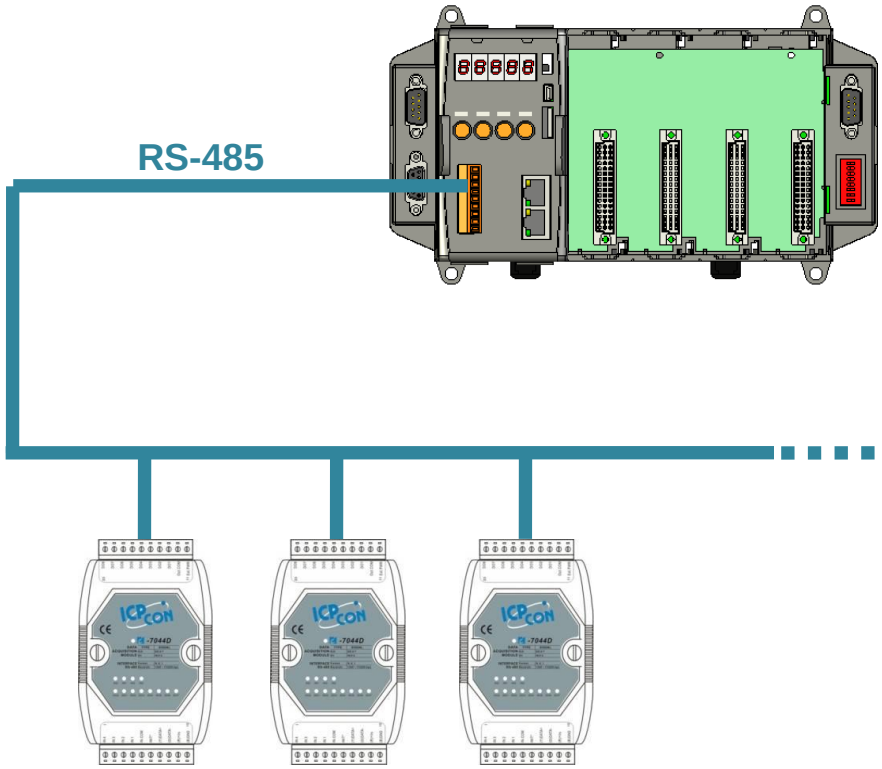
Request/Response communication is very typical protocol architecture. If you want to design a command set of communication protocol as table below, you can refer to “slave_com” demo.



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

4.2. **API for I/O Modules**

The iPAC-8000 equip a RS-485 communication interface, COM2, to access 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.

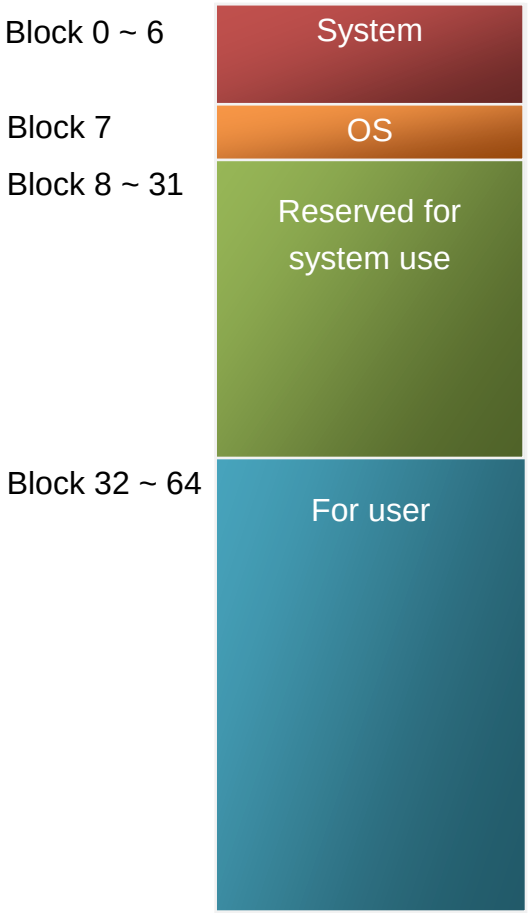
Step 4: Use RestoreCom() to restore the COM port driver

For example, to send a command ‘$01M’ to I-7K I/O module for getting the module name.

```
#include <stdio.h>
#include “8000a.h”

void main(void)
{
 unsigned char InBuf0[60];
 InitLib();      /* Initiate the 8000a 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 to 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 “8000a.h”

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

 InitLib();      /* Initiate the 8000a library */
 EE_WriteEnable();
 EE_MultiWrite(1,10,1,&data);
 EE_WriteProtect();

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

4.4. API for Flash Memory

- The iPAC-8000 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 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 there is completed, new data can be written.

Free: 448 K bytes
MiniOS7: 64 K bytes
Total Size: 512 K bytes

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 segnment 0xD000, offset 0x1234 of the Flash memory.

```
#include <stdio.h>
#include “8000a.h”

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

InitLib();      /* Initiate the 8000a library */
dataptr=(char *)&data;
FlashWrite(0xd000,0x1234, *dataptr++);
FlashWrite(0xd000,0x1235, *dataptr);

/* Read data from the Flash Memory (method 1) */
dataprt=(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;
}
```

4.5. **API for NVRAM**

- The iPAC-8000 equip 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.

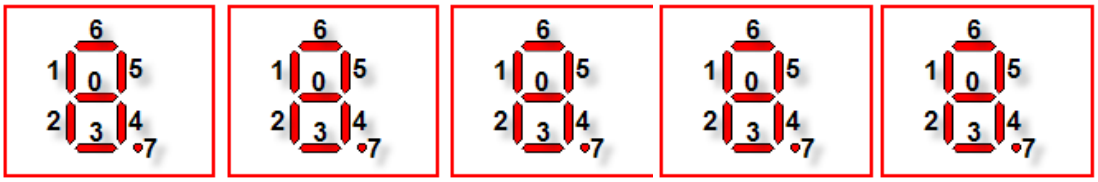
```
#include <stdio.h>
#include “8000a.h”

void main(void)
{
 int data=0x55, data2;
 InitLib();      /* Initiate the 8000a 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 “8000a.h”

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

InitLib();      /* Initiate the 8000a 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 */
}
```

4.6. **API for 5-Digital LED**

The iPAC-8000 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 “8000a.h”

void main(void)
{
 InitLib();      /* Initiate the 8000a 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 */

}
```

4.7. **API for Timer**

- The iPAC-8000 can support a single main time tick, 8 stop watch timers and 8 counts down timers.
- The iPAC-8000 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 “8000a.h”

void main(void)
{
 Unsigned long time iTime;

 InitLib();      /* Initiate the 8000a 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 iPAC-8000 timer function */
}
```

4.8. API for WatchDog Timer (WDT)

- The iPAC-8000 equips the 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) refreshes 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 “8000a.h”

void main(void)
{
 Unsigned long time iTime;

 InitLib();      /* Initiate the 8000a library */
 Enable WDT();
 While(!quit)
 {
  RefreshWDT();
  User_function();
 }
 DisableWDT();
}
```

4.9. API for MFS (For iPAC-8000-FD series only)

Required library and header files:
MFS_V212.LIB and MFS.h

The iPAC-8000-FD series products equip 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, applications, and the specification, please refer to section “Appendix C. 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.
mfs_DeleteAllFiles	Delete all files stored in the NAND Flash.
mfs_DeleteFile	Delete one selected file that has been written to the NAND Flash.
mfs_OpenFile	1. Opens a file with a file name.

Function	Description
	2. Creates a new file.
mfs_CloseFile	Closes a file with a file handle. All buffers associated with the stream are flushed before closing.
mfs_ReadFile	Reads specified bytes of data from a file.
mfs_WriteFile	Appends specified bytes of data to a file.
mfs_Getc	Gets a character from a file.
mfs_Putc	Outputs a character data to the file.
mfs_Gets	Gets a string from a file.
mfs_Puts	Outs a string a file.
mfs_EOF	Macro that tests if end-of-file has been reached on a file.
mfs_Seek	Repositions the file pointer of a file.
mfs_Tell	Returns the current file pointer.
mfs_EnableWriteVerify	Enable the data verification. By default, the data verification is enabling.
mfs_DisableWriteVerify	Disable the data verification.

4.10. API for microSD

Required library and header files:
SD_V102.LIB and microSD.h

The iPAC-8000 series can support one microSD card and the size can be 1GB or 2 GB.

- Summarize of the microSD functions:

Function	Description
pc_init	Initializes the microSD socket library
pc_open	1. Open an existing file and return a file handle 2. Creates a new file.
pc_close	Closes a file and release a file handle.
pc_read	Reads the specified file
pc_write	Writes the specified file
pc_seek	Moves the file pointer to relative offset from the current offset
pc_tell	Gets current offset of the file pointer
pc_eof	Checks whether the end-of-file is reached
pc_format	Formats the microSD card as FAT (FAT16)
pc_mkdir	Creates a directory or subdirectory
pc_rmdir	Removes an existing directory
pc_move	Renames an existing file or a directory, including the subdirectory
pc_del	Deletes the specified file
pc_deltree	Deletes the specified directory or subdirectory
pc_isdir	Checks whether the file is a directory
pc_isvol	Checks if is a volume
pc_size	Gets the size of the specified file
pc_set_cwd	Sets the current working directory
pc_get_cwd	Gets the pathname of the current working directory
pc_gfirst	Moves the pointer to the first element
pc_gnext	Moves the pointer to the next element

Function	Description
pc_gdone	Moves the pointer to the last element
pc_get_freeSize_KB	Gets the free space of the SD memory card
pc_get_usedSize_KB	Gets the used space of the SD memory card
pc_get_totalSize_KB	Gets the total size of the SD memory card
pc_get_attributes	Gets the file attributes
pc_set_attributes	Sets the file attributes
pc_get_errno	Gets the error number

API for starting microSD

1. pc_ Init()

Before using any miscroSD functions, PC_Init() must be called to initialize the microSD.

API for enabling/disabling microSD

3. pc_open()

Before writing/reading data to/from the microSD card, PC_open() must be called to open the file.

4. pc_close()

After the data has finished being written/read to/from the microSD, PC_close() must be called to close the file with a file handle.

API for writing data to the microSD

5. pc_write()

After using PC_open() to open the file, PC_write() must be called to read data from the microSD.

For example, writing data to the microSD

```
#include <string.h>
#include <stdio.h>
#include “8000a.h”
#include “microSD.h”
{
  int fd, iRet;

  InitLib();
  If(pc_init())
  {
    Print(“Init microSD ok\n\r”);
  }
  else
  {
    Print(“Init microSD failed\n\r”);
    iRet=pc_get_errno();
    switch(iRet)
    {
      case PCERR_BAD_FORMAT:      //1
          Print("Error 01: format is not FAT\n\r");
          break;
      case PCERR_NO_CARD:      //2
          Print("Error 02: no microSD card\n\r");
          break;
      default:
          Print("Error %02d: unknow error\n\r",iRet);
    }
  }
```

```
fd=pc_open(“test.txt”,(word)(PO_WRONLY|PO_CREAT|PO_APPEND),(word
)(PS_IWRITE|PS_IREAD));
  if(fd>=0)
  {
    pc_write(fd,”1234567890”,10);      //write 10 bytes
    pc_close(fd);
  }
}
```

API for reading data from the microSD

6. pc_read()

After using PC_open() to open the file, PC_read() must be called to read data from the microSD.

For example, reading data from the microSD:

```
#include <string.h>
#include <stdio.h>
#include “8000a.h”
#include “microSD.h”
{
  int fd, iRet;
  unsigned char Buffer[80];

  InitLib();
  If(pc_init())
  {
    Print(“Init microSD ok\n\r”);
  }
```

```
  else
  {
    Print(“Init microSD failed\n\r”);
    iRet=pc_get_errno();
    switch(iRet)
    {
      case PCERR_BAD_FORMAT:      //1
          Print("Error 01: format is not FAT\n\r");
          break;
      case PCERR_NO_CARD:      //2
          Print("Error 02: no microSD card\n\r");
          break;
      default:
          Print("Error %02d: unknow error\n\r",iRet);
    }
  }

 fd=pc_open(“test.txt”,(word)(PO_RDONLY),(word)(PS_IWRITE|PS_IREAD));
  if(fd>=0)
  {
    iRet=pc_read(fd,Buffer,10);      //reads 10 bytes
    Buffer[10]=0;      //adds zero end to the end of the string.
    pc_close(fd);
    Print(“%s”,Buffer);
  }
}
```

For more demo program about the microSD, please refer to:

CD:\NAPDOS\iPAC8000\Demo\Basic\microsd\

http://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/microsd/

Appendix A. What is MiniOS7?

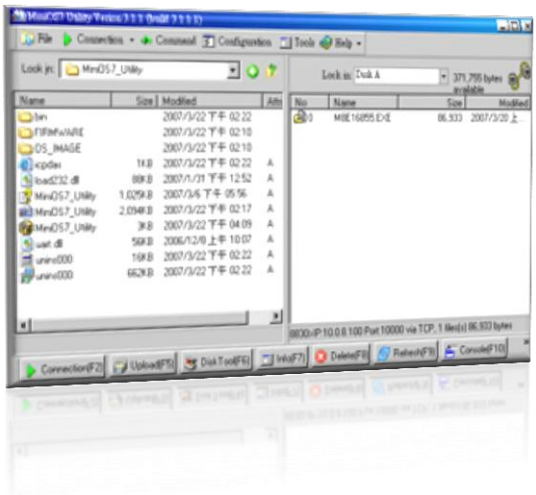
MiniOS7 is an embedded ROM-DOS operating system design by ICP DAS. It is functionally equivalent to other brands of DOS, and can run programs that are executable under a standard DOS.

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
OS update (Upload)	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 B. What is MiniOS7 Utility?



MiniOS7 Utility is a tool for configuring, uploading files to all products embedded with ICP DAS MiniOS7.

Since version 3.1.1, the Utility can allow users remotely access the controllers (7188E, 8000E..., etc) 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
     …, etc.

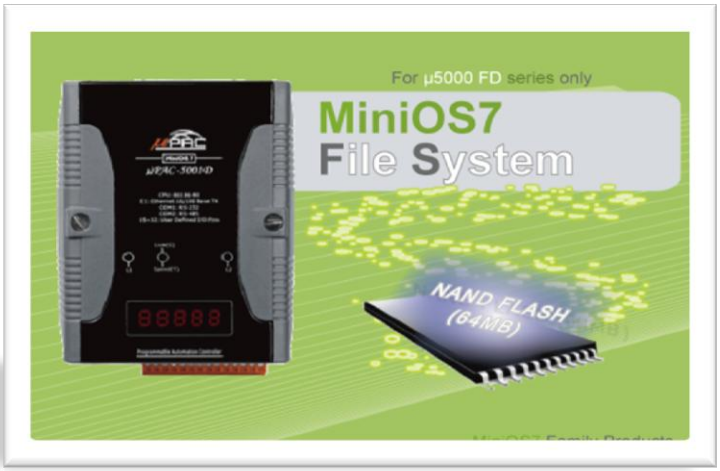
Including frequently used tools

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

Upload location:

http://ftp.Icpdas.com/pub/cd/8000cd/napdos/minios7/utility/minios7_utility/

Appendix C. What is MiniOS7 File System (MFS)?



MiniOS7 file system, MFS, offers a rugged alternative to mechanical storage systems. Designed for the 64MB NAND flash memory,

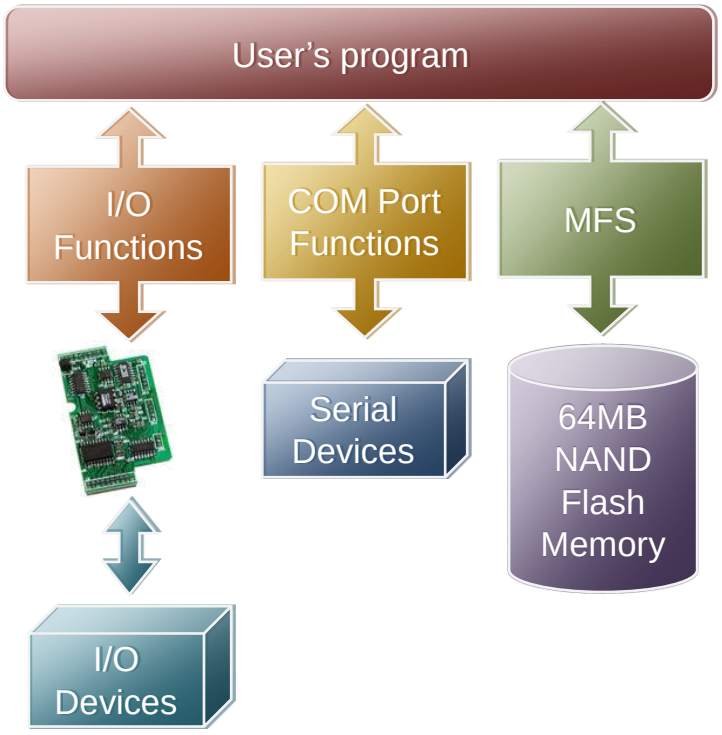
MFS implements a reliable file system with C language API for embedded data logger applications on MiniOS7.

Hardware Supported

iPAC-8000-FD (With 64MB Flash Memory), NVRAM: all of the 31 bytes.

Applications

Log data with timestamp, Log data and forward via the Ethernet



MFS Specifications

Item	Description
Disk size	1/2 size of the flash memory size
File number	456 files max. for each disk
File size	Disk Size max. for each file
File name	12 bytes max (case sensitive)
File operation modes	1. Read only 2. Write only: Creates a new file to write data, or overwrite a file (if the file is already exit). 3. Append: appends data to a file.

File handle	10 max for each disk. 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. For write and append mode: only 1 file handle can be used for writing operation on all disks.
Writing verification	Yes. Default is enabled. Calling mfs_EnableWriteVerification and mfs_DisableWriteVerification can change the setting.
Automate file system recovery	Yes. 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 (mfs_WriteFile,) might be lost. When the file system reboots, it restores the file system to its state at the time of the last writing operation.
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.

Resources upload

- MFS SDKs:
  http://ftp.Icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/lib/
- MFS Demos:
  http://ftp.Icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/64mb_flash/

Appendix D. 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

D.1. Turbo C 2.01

You have a couple of choices here, you can:

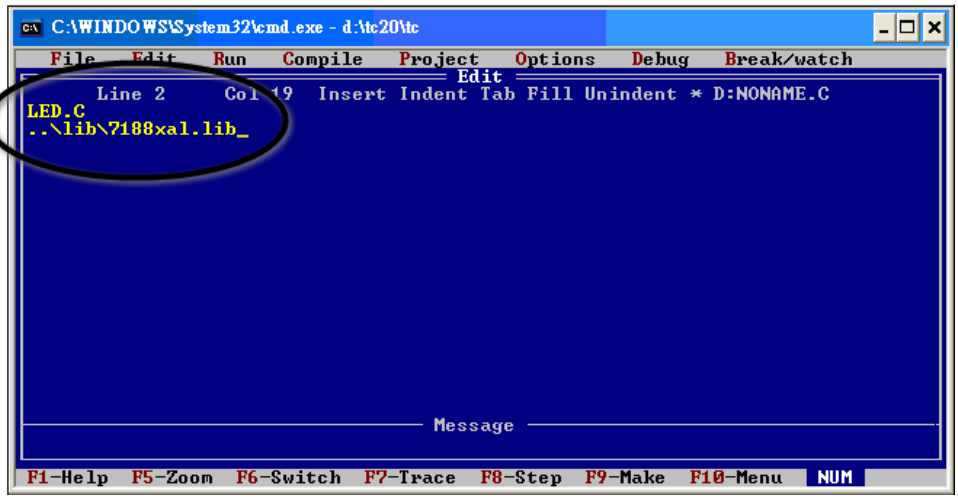
1: Using a command line

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 ..\..\Demo\basic\Lib\iPAC-8000.lib

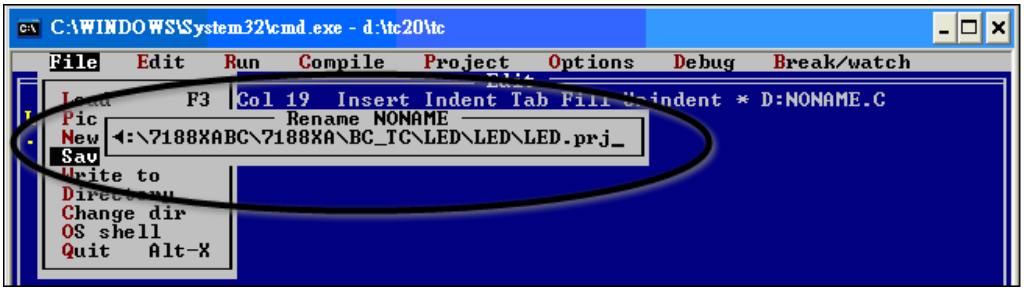
2: Using the TC Integrated Environment

Step 1: Executing the TC 2.01
Step 2: Editing the Project file

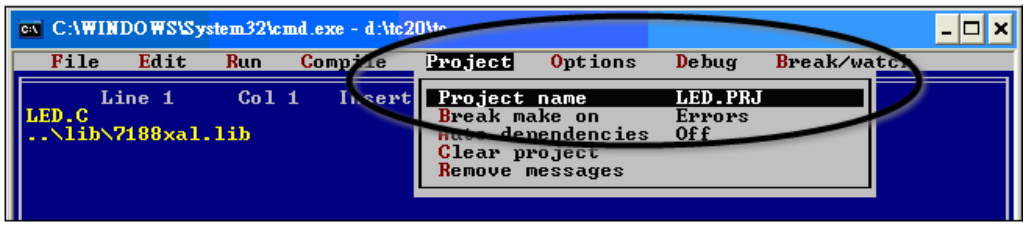
Adding the necessary library and file to the project



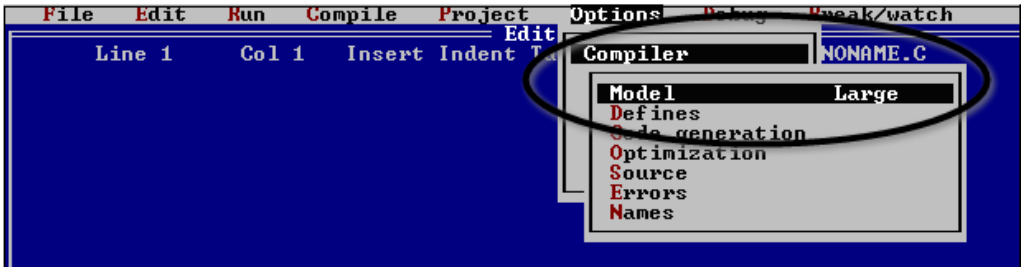
Step 3: Save the project and entering a name, such as LED.prj

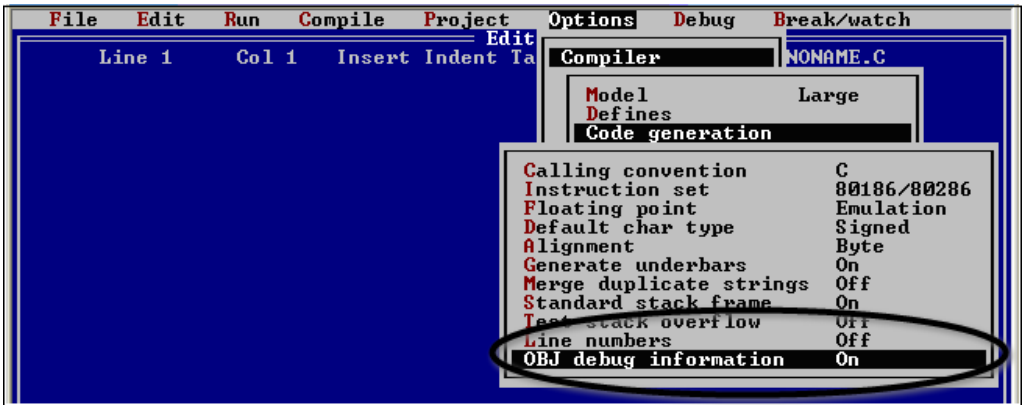


Step 4: Load the Project

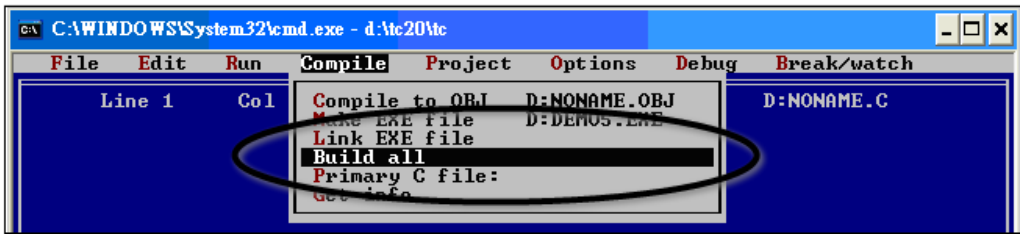


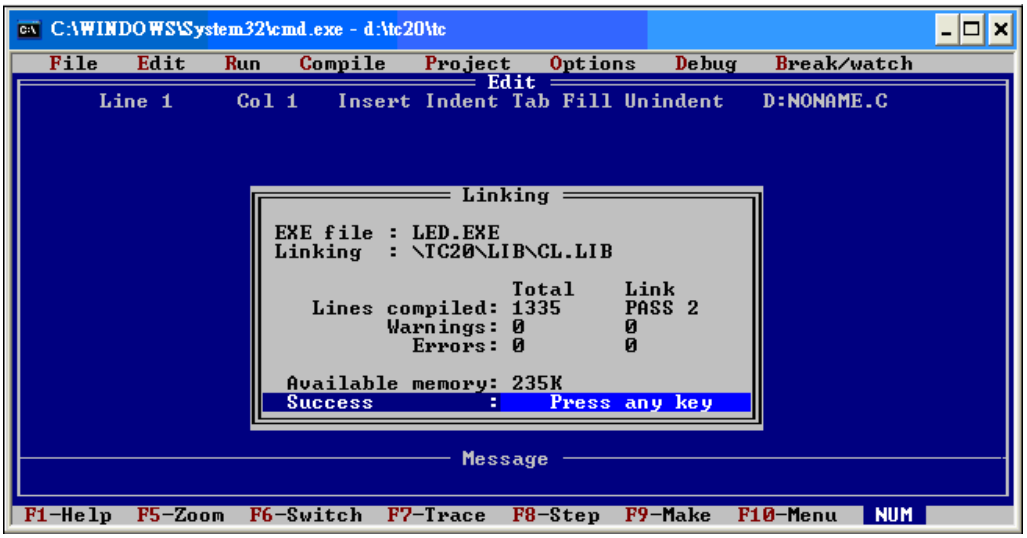
Step 5: Change the Memory model (Large for iPAC-8000.lib) and set the Code Generation to 80186/80286





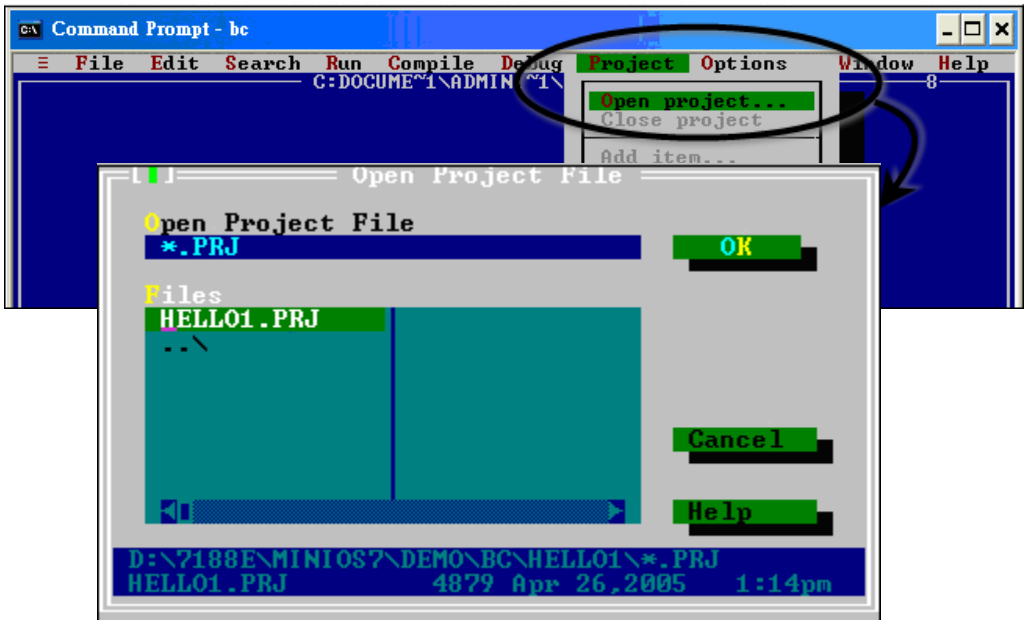
Step 6: Building the project



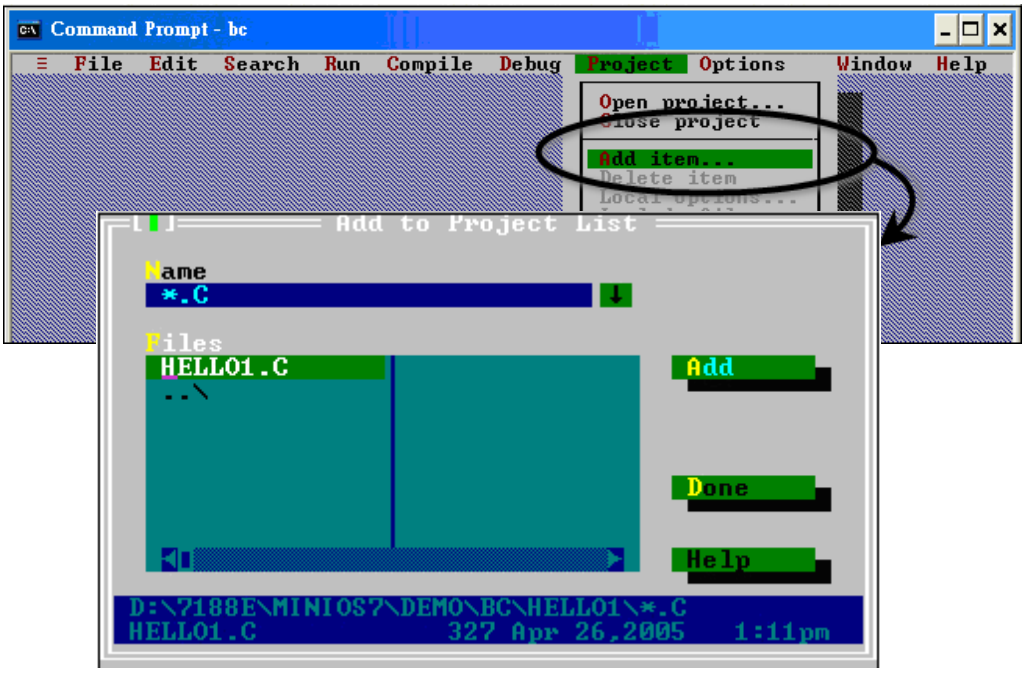


D.2. BC++ 3.1. IDE

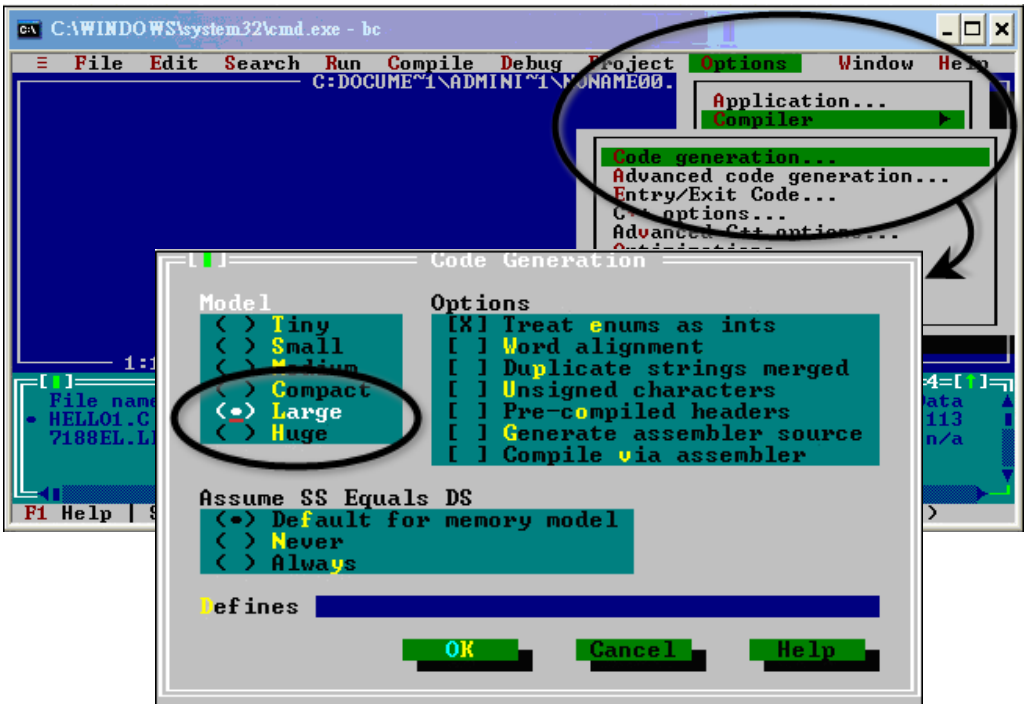
Step 1: Executing the Borland C++ 3.1
Step 2: Creating a new project file (*.prj)



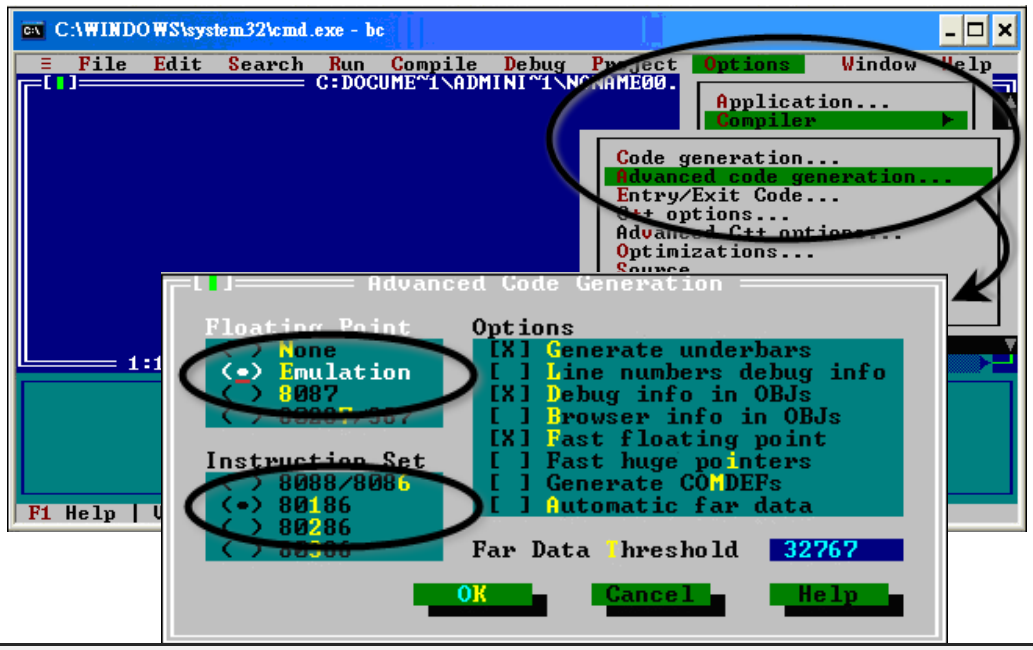
Step 3: Add all the necessary files to the project



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

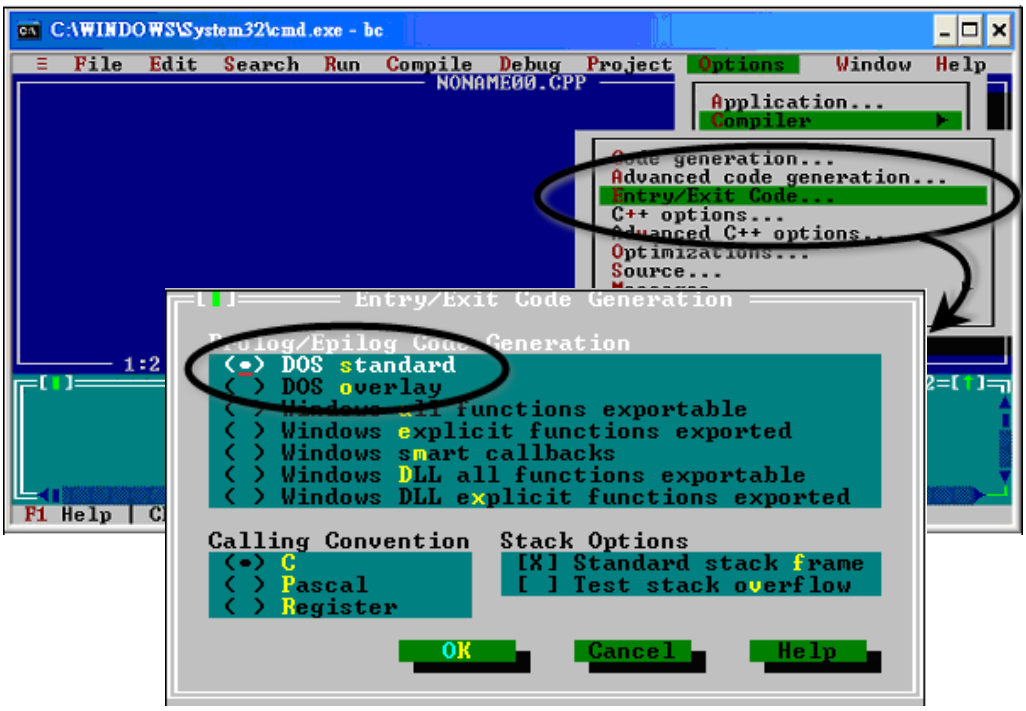


Step 5: Set the Advanced code generation options and Set the Floating Point to Emulation and the Instruction Set to 80186

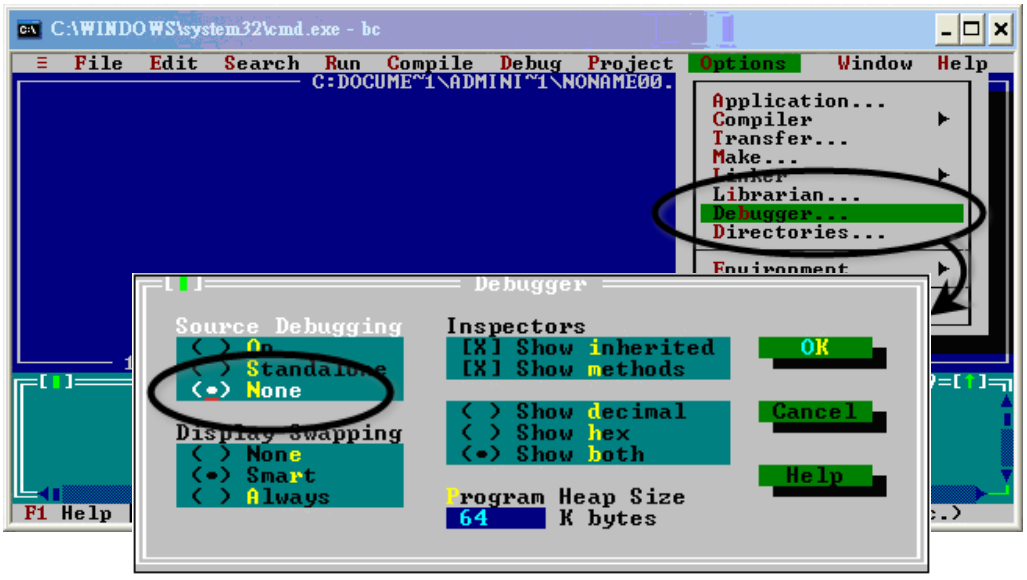


Step 6: Set the Entry/Exit Code Generation option and setting the DOS

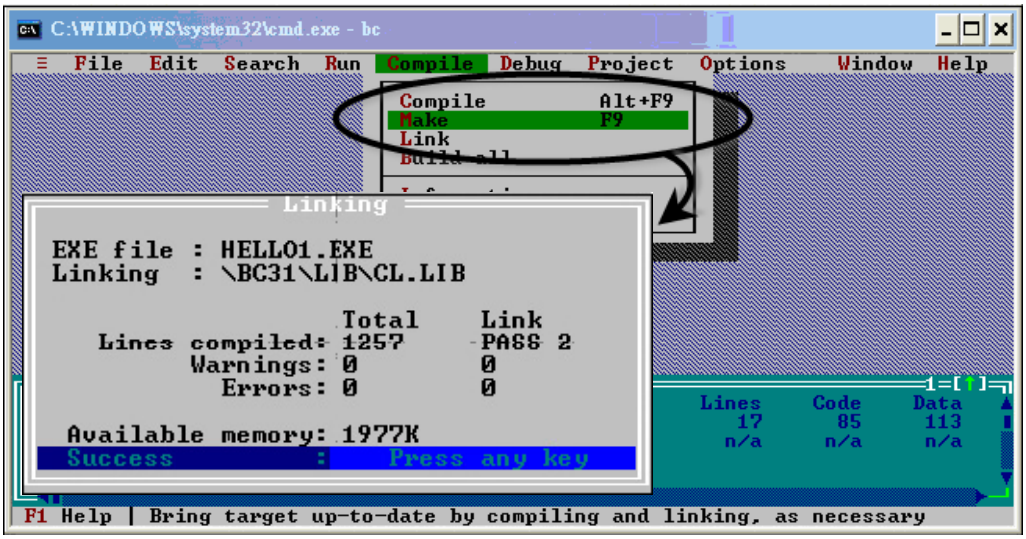
standard



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

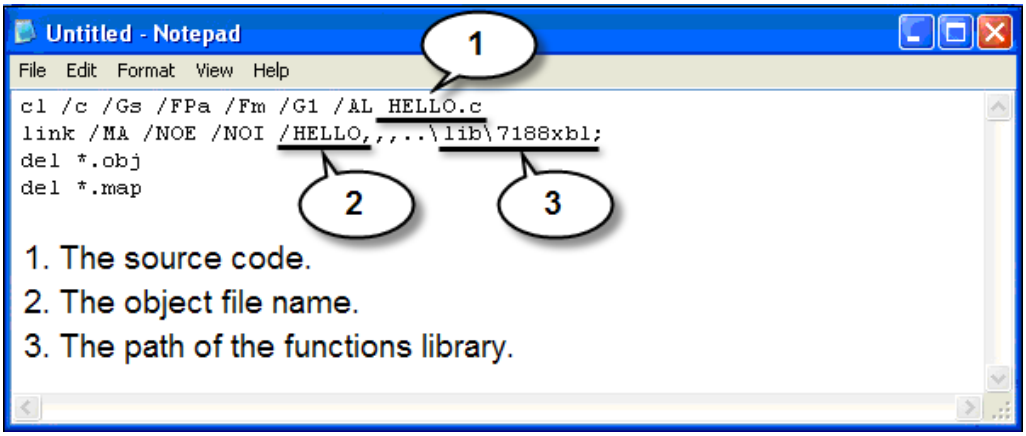


Step 8: Make the project


C:\WINDOWS\system32\cmd.exe - bc
File Edit Search Run Compile Debug Project Options Window Help
Compile Alt+F9
Make F9
Link
Linking
EXE file : HELLO1.EXE
Linking : \BC31\LIB\CL.LIB
Total Link
Lines compiled: 1257 PASS 2
Warnings: 0 0
Errors: 0 0
Available memory: 1977K
Success : Press any key
Lines Code Data
17 85 113
n/a n/a n/a
F1 Help | Bring target up-to-date by compiling and linking, as necessary

D.3. MSC 6.00

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



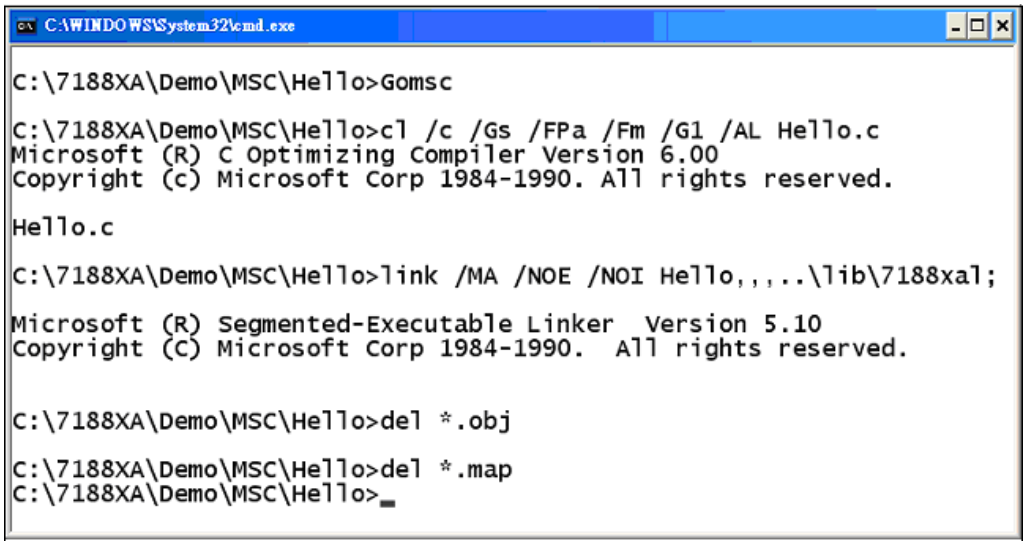
Tip & Warnings

/C Don't strip comments /GS No stack checking

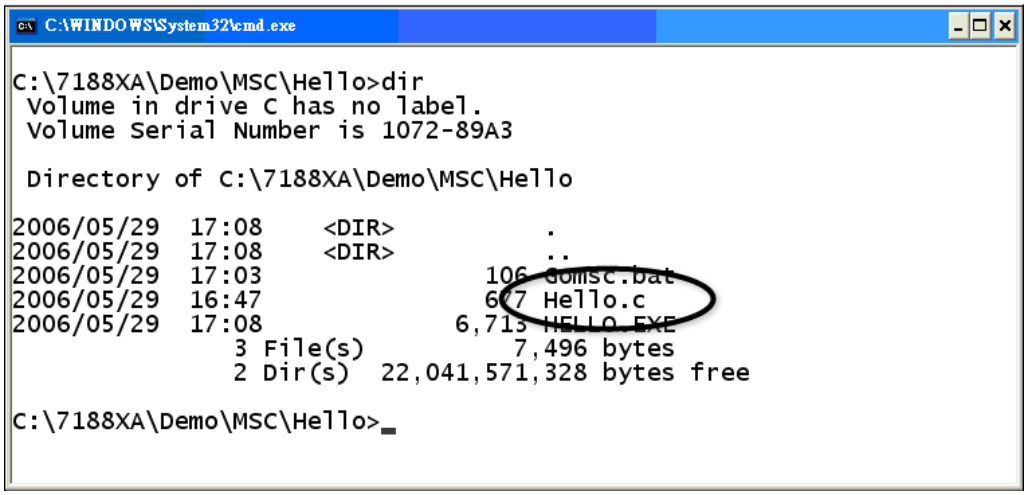
/Fpa: Calls with altmath /Fm [map file]

/G1: 186 instructions /AL Large model

Step 2: Run the Gomsc.bat file

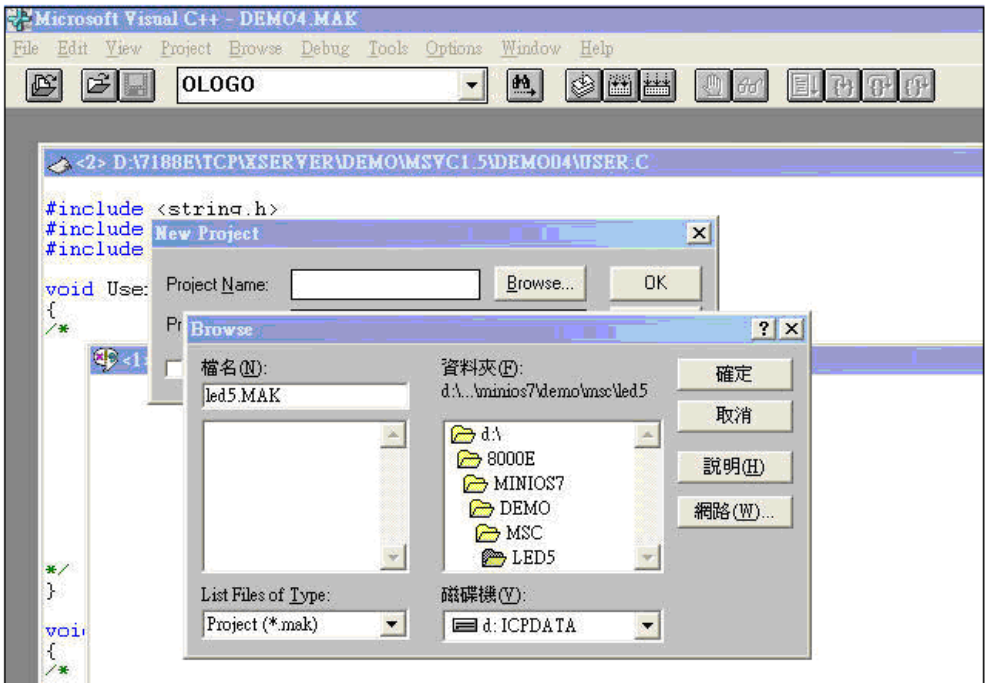


Step 3: A new executable file will be created if it is successfully compiled

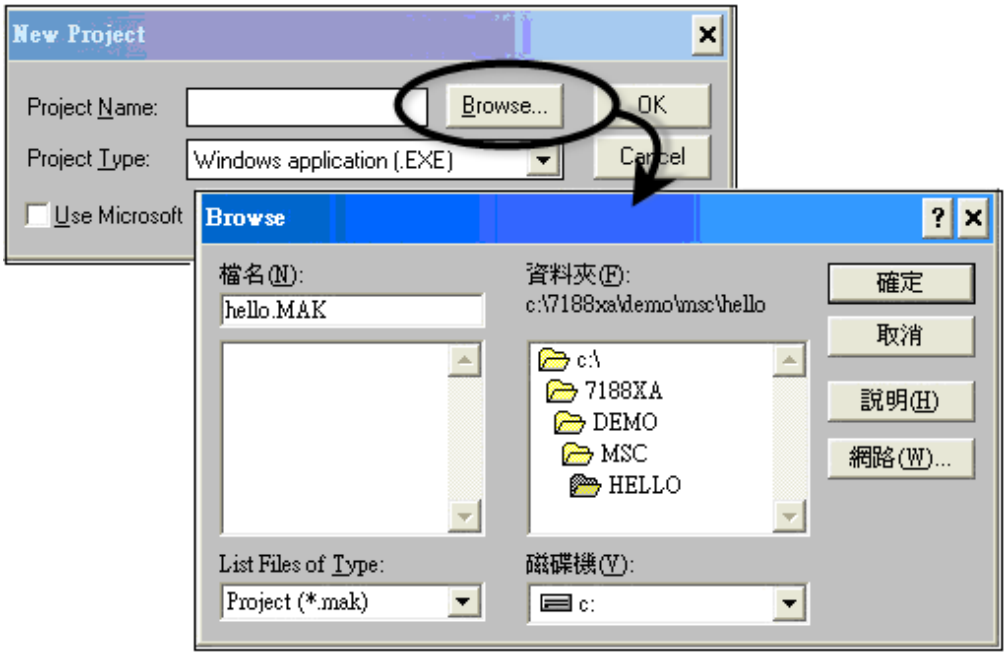


D.4. MSVC 1.50

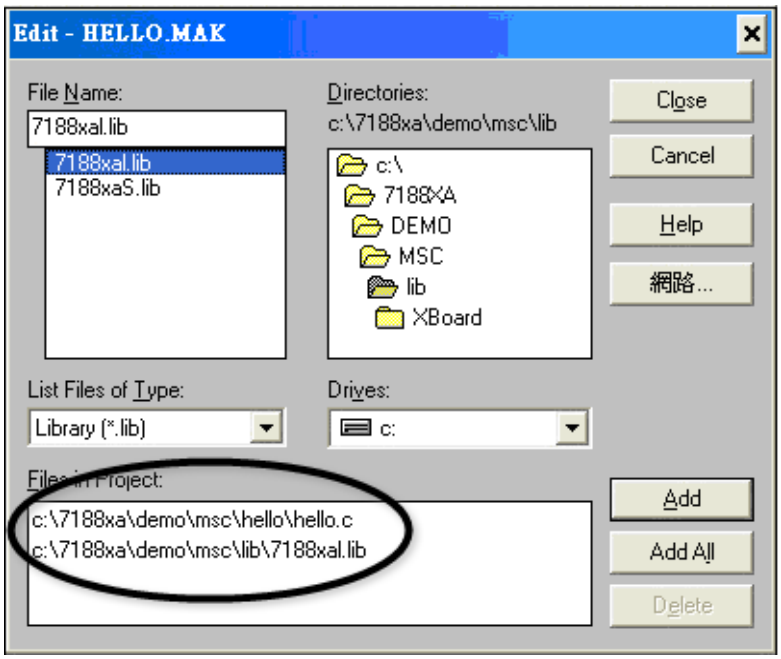
Step 1: Run MSVC.exe



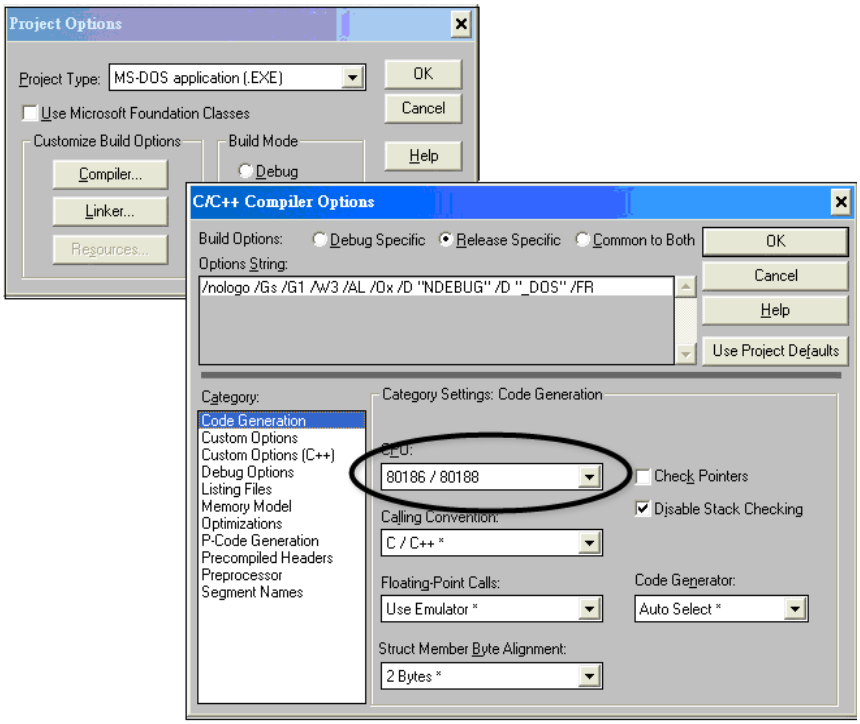
Step 2: Create a new project (*.mak) by entering the name of the project in the Project Name field and then select MS-DOS application (EXE) as the Project type



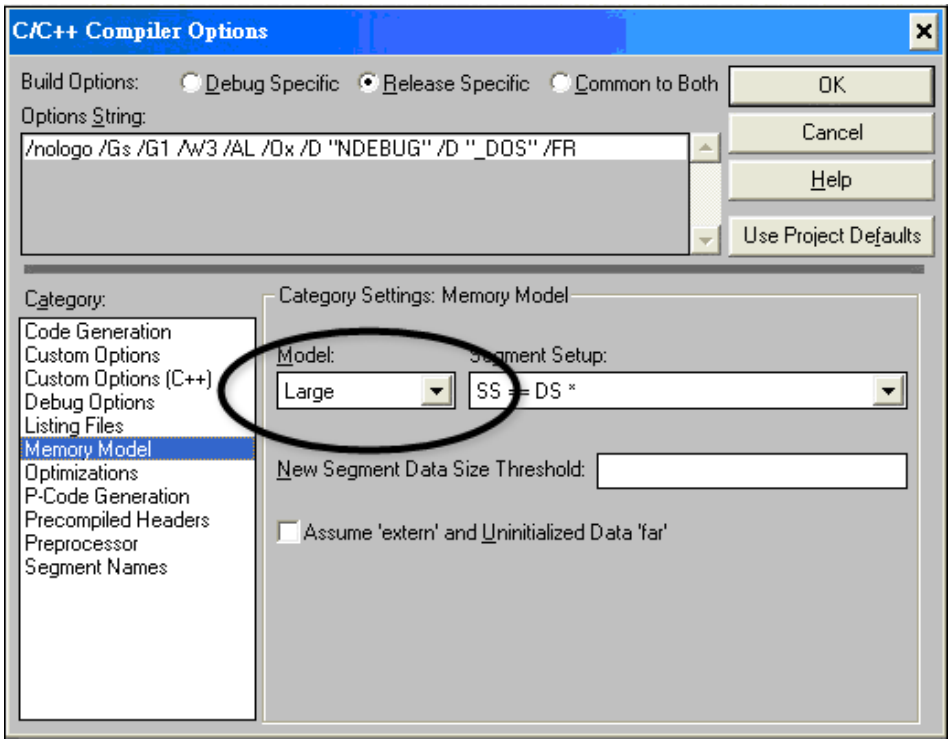
Step 3: Add the user's program and the necessary library files to the project



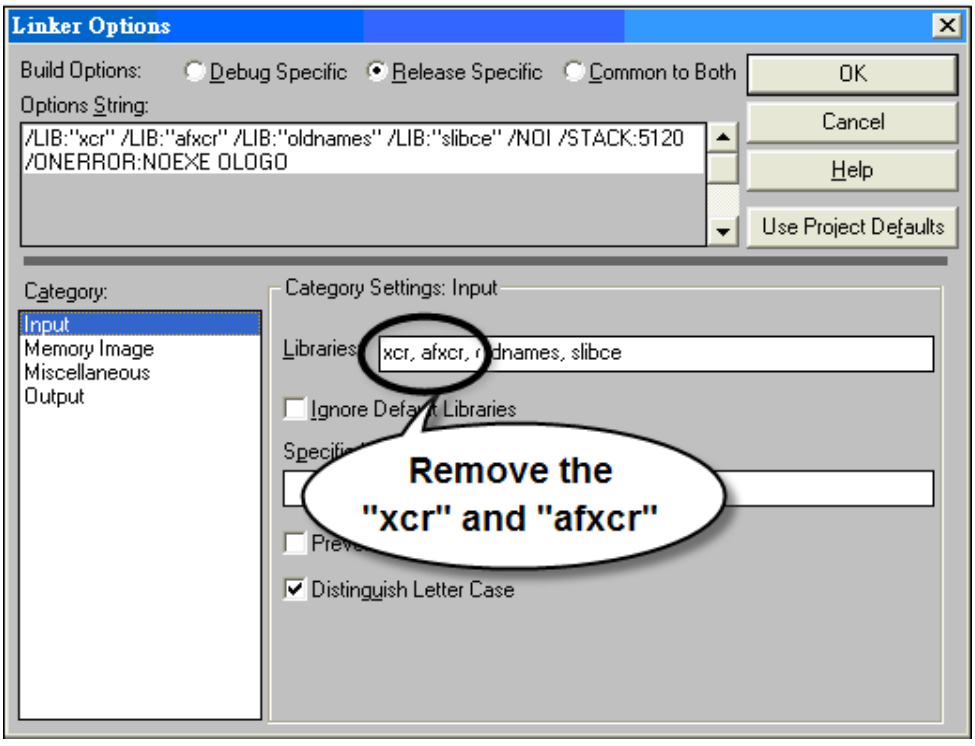
Step 4: Set the Code Generation on the Compiler.



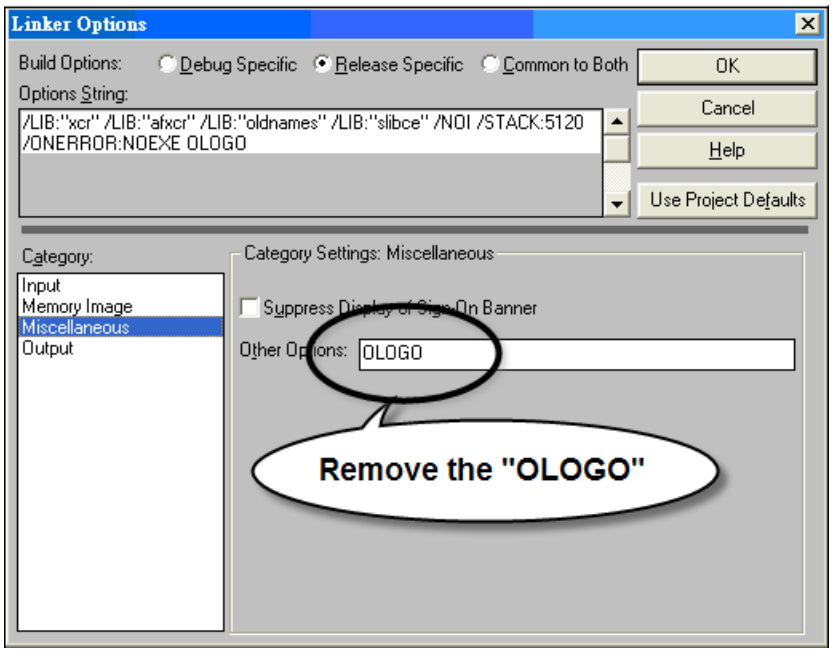
Step 5: Change the Memory model (large for iPAC-8000.lib)



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



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



Step 8: Rebuild the project