

μPAC-5000 User Manual

(C Language Based)

Version 1.0.2, Jan. 2012



Service and usage information for

μPAC-500x (D) Series

μPAC-510x (D) Series

μPAC-520x (D) Series

μPAC-530x (D) Series

μPAC-54xx (D) Series

μPAC-560x (D) Series

μPAC-570x (D) Series

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 © 2011 by ICP DAS Co., Ltd. All rights are reserved.

Trademark

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

Contact US

If you have any problem, please feel free to contact us.
You can count on us for quick response.

Email: service@icpdas.com

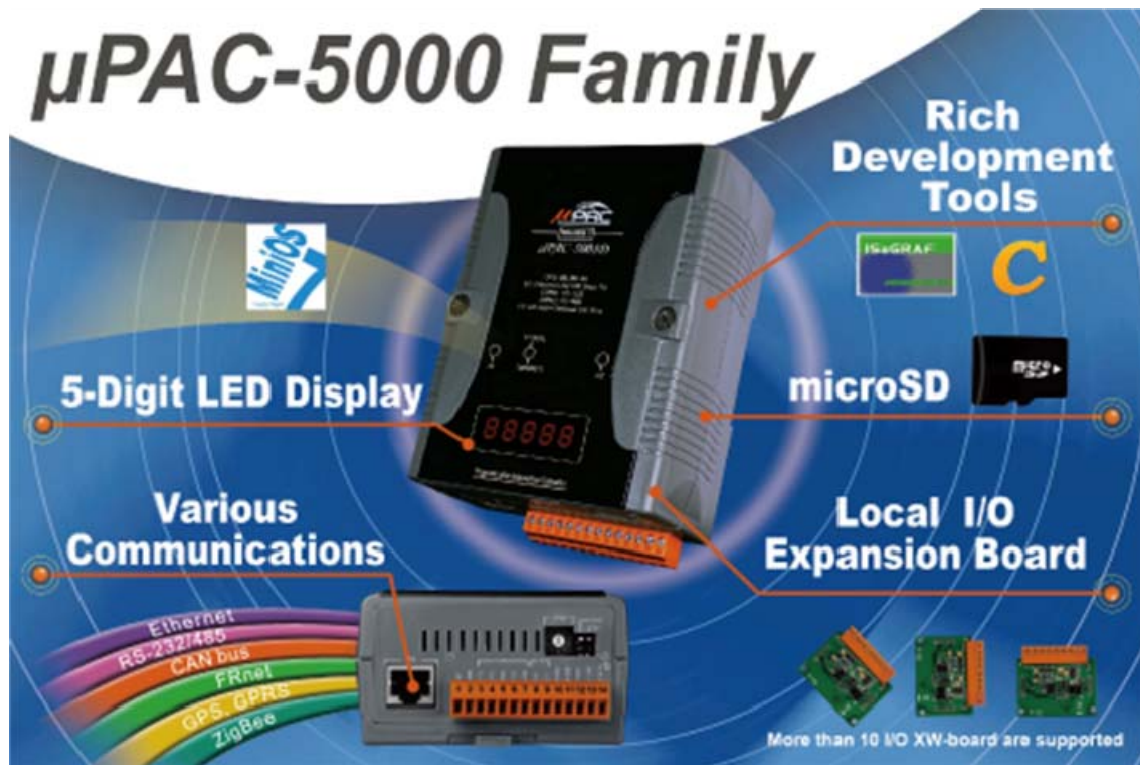
Table of Contents

Table of Contents	3
1. Introduction	6
1.1. μ PAC-5000 (C Language Based)	7
1.1.1. μ PAC-5000 Module Naming Convention	7
1.1.2. μ PAC-5000 Comparison	8
1.2. Features	16
1.3. Specifications	20
1.3.1. Common Specifications	20
1.3.2. GPS Specifications	21
1.3.3. 2G (GPRS) Specifications	22
1.3.4. 3G (WCDMA) Specifications	23
1.3.5. ZigBee Specifications	24
1.3.6. Wi-Fi Specifications	25
1.4. Overview	26
1.5. Dimensions	30
1.6. Companion CD	31
2. Getting Started	32
2.1. Hardware Installation	33
2.2. GSM Series Installation	35
2.2.1. Installing the SIM card	35
2.2.2. Installing the GSM antenna	36
2.3. Software Installation	37
2.4. Boot Configuration	39
2.5. Uploading μ PAC-5000 Programs	40
2.5.1. Establishing a connection between PC and μ PAC-5000	41

(a) RS-232 connection-----	42
(b) Ethernet Connection-----	45
2.5.2. Uploading and executing μ PAC-5000 programs-----	50
2.5.3. Making programs start automatically-----	51
2.6. Updating μ PAC-5000 OS Image-----	53
3. “Hello World” - Your First Program-----	56
3.1. C Compiler Installation-----	57
3.1.1. Installing the C compiler-----	58
3.1.2. Setting up the environment variables-----	62
3.2. μ PAC-5000 APIs-----	65
3.3. First Program in μ PAC-5000-----	66
4. APIs and Demo References-----	76
4.1. API for COM Port-----	80
4.1.1. Types of COM port functions-----	81
4.1.2. API for MiniOS7 COM port-----	82
4.1.3. API for standard COM port-----	85
4.1.4. Port functions Comparison-----	87
4.1.5. Request/Response protocol define on COM port-----	88
4.2. API for I/O Modules-----	89
4.3. API for EEPROM-----	91
4.4. API for Flash Memory-----	93
4.5. API for NVRAM-----	95
4.6. API for 5-Digital LED-----	97
4.7. API for Timer-----	98
4.8. API for WatchDog Timer (WDT)-----	100
4.9. API for MFS (For μ PAC-5000-FD series only)-----	101
4.10. API for microSD-----	105

4.11. GSM API Function-----	109
Appendix A. What is MiniOS7? -----	117
Appendix B. What is MiniOS7 Utility?-----	118
Appendix D. More C Compiler Settings -----	122
D.1. Turbo C 2.01 -----	123
D.2. Borland C++ 3.1 -----	126
D.3. MSC 6.00 -----	130
D.4. MSVC 1.50-----	132

1. Introduction



μPAC-5000 family is a palm size PAC. It can install in narrow space and survive in harsh environment. For different applications, μPAC-5000 family offers 3 series, μPAC-5000, LP-5000 and WP-5000. Customers who needs MiniOS7, Linux, or WinCE, all can find a product to fit their requirement.

μPAC-5000 family offers two CPU types (80186, PXA270), 3 OS solutions (WinCE 5.0, Linux, MiniOS7) and several software development toolkits (C, VS .NET, ISaGRAF, InduSoft) for chosen, and all of them are featured same stability and flexibility as ICP DAS's PAC family. This makes μPAC-5000 have a good potential to apply to factory automation, building automation, machine automation, manufacturing management, environment monitoring, etc.

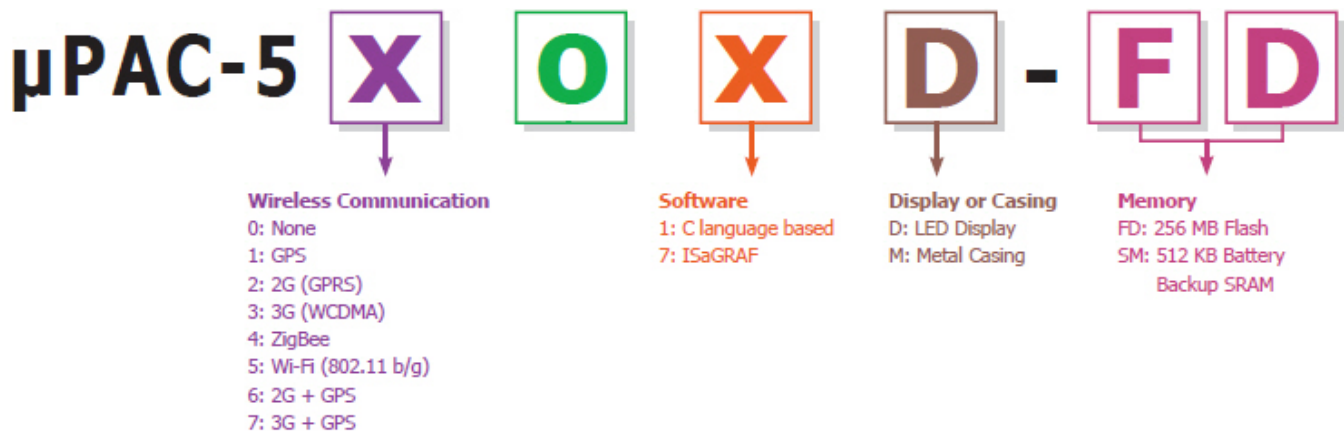
Besides, combining one of the optional expansion boards (XW-boards) for μPACs, such as DI, DO, A/D, D/A, Timer/Counter, communication interface (RS-232/422/485, CAN, FRnet), etc., customers can customize the hardware specification of their μPAC-5000 for different situations.

1.1. μ PAC-5000 (C Language Based)

1.1.1. μ PAC-5000 Module Naming Convention

As you examine this manual, you'll notice there are many different products available. Sometimes it is difficult to remember the specifications for any given product. However, if you take a few minutes to understand the module naming conventions, it may save you some time and confusion. The figure below shows how the module naming conventions work for each μ PAC-5000 product.

The μ PAC-5000 series are divided into eight series - the General, GPS, 2G (GPRS), 3G (WCDMA), Zigbee, WiFi, 2G + GPS and 3G + GPS series.



1.1.2. μPAC-5000 Comparison

μPAC-5000 can be divided into eight types, according to their features.

(1) General Series

The following table provides a comparison of μPAC-5000 General Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/ RS-485
μPAC-5001(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	-	1/1
μPAC-5001(D)-FD				microSD + 256 MB Flash			
μPAC-5001(D)-SM				MicroSD + 512 KB Battery Backup SRAM			

(2) GPS Series

The following table provides a comparison of μ PAC-5000 GPS Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/RS-485
μ PAC-5101(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	GPS	1/1
μ PAC-5101(D)-FD				microSD + 256 MB Flash			
μ PAC-5101(D)-SM				microSD + 512 KB Battery Backup SRAM			

(3) 2G (GPRS) Series

The following table provides a comparison of μ PAC-5000 2G (GPRS) Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/ RS-485
μ PAC-5201(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	2G (GPRS)	1/1
μ PAC-5201(D)-FD				microSD + 256 MB Flash			
μ PAC-5201(D)-SM				microSD + 512 KB Battery Backup SRAM			

(4) 3G (WCDMA) Series

The following table provides a comparison of μ PAC-5000 3G (WCDMA) Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/ RS-485
μ PAC-5301(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	3G (WCDMA)	1/1
μ PAC-5301(D)-FD				microSD + 256 MB Flash			
μ PAC-5301(D)-SM				microSD + 512 KB Battery Backup SRAM			

(5) ZigBee Series

The following table provides a comparison of μ PAC-5000 ZigBee Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/ RS-485
μ PAC-5401(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	ZigBee (Host)	1/1
μ PAC-5411(D)						ZigBee (Slave)	
μ PAC-5401(D)-FD				microSD + 256 MB Flash		ZigBee (Host)	
μ PAC-5411(D)-FD						ZigBee (Slave)	
μ PAC-5401(D)-SM				microSD + 512 KB Battery Backup SRAM		ZigBee (Host)	
μ PAC-5411(D)-SM						ZigBee (Slave)	

(6) Wi-Fi Series

The following table provides a comparison of μ PAC-5000 Wi-Fi Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/ RS-485
μ PAC-5501(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	Wi-Fi (802.11 b/g)	1/1
μ PAC-5501(D)-FD				microSD + 256 MB Flash			
μ PAC-5501(D)-SM				microSD + 512 KB Battery Backup SRAM			

(7) 2G (GPRS) + GPS Series

The following table provides a comparison of μ PAC-5000 2G (GPRS) + GPS Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wiress Communication	RS-232/ RS-485
μ PAC-5601(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	2G (GPRS) + GPS	1/1
μ PAC-5601(D)-FD				microSD + 256 MB Flash			
μ PAC-5601(D)-SM				microSD + 512 KB Battery Backup SRAM			

(8) 3G (WCDMA) + GPS Series

The following table provides a comparison of μ PAC-5000 3G (WCDMA) + GPS Series specifications:

Model	CPU	Flash	SRAM	Memory Expansion	Ethernet	Wireless Communication	RS-232/ RS-485
μ PAC-5701(D)	80 MHz	512 KB	512 KB	microSD	10/100 BaseTX	3G (WCDMA) + GPS	1/1
μ PAC-5701(D)-FD				microSD + 256MB Flash			
μ PAC-5701(D)-SM				microSD + 512KB Battery Backup SRAM			

1.2. Features

➤ Various CPU and OS for Choosing



MiniOS7
80186 CPU
μPAC-5000 Series

- DOS-like
- Boot up in 0.4 ~ 0.8 second
- Build-in hardware diagnostic
- Standard version for C language programming
- ISaGRAF version for IEC 61131-3 programming

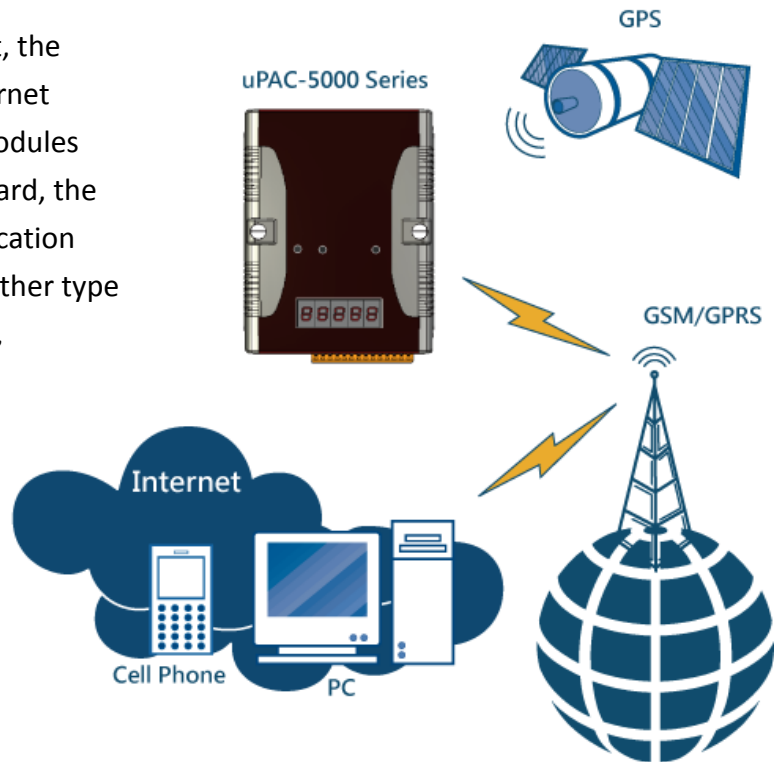
➤ Local I/O and Communication Expansion Board



The μPAC-5000 series equip an I/O expansion bus to support one optional expansion board, called XW-board. It can be used to implement various I/O functions such as DI, DO, A/D, D/A, Timer/Counter and various communication interface, such as RS-232/422/485, CAN, FRnet, etc.

➤ Remote I/O Module and Expansion Unit

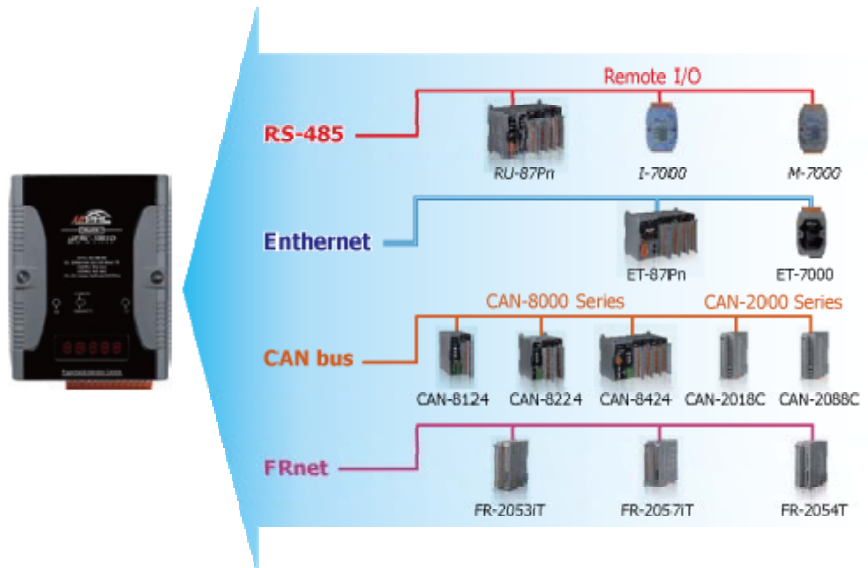
With the built-in RS-485 and Ethernet port, the μ PAC-5000 series can connect RS-485/Ethernet remote I/O Units (RU-87Pn/ET-87Pn) or modules (I-7000/M-7000/ET-7000). With an XW-board, the μ PAC-5000 series can have more communication ports or different interface to connect to other type of devices, for example: CANOpen devices, DeviceNet devices, or FRnet I/O modules.



➤ Multi-Communication Interface

There are several communication interfaces to expand I/O and connect external devices:

- Ethernet
- RS-232/485
- CAN bus
- FRnet
- GPS
- 2G/3G
- ZigBee
- Wi-Fi



➤ Various Memory Expansions

μPAC-5000 provides various memory storage options, such as EEPROM, Flash, battery-backup SRAM or microSD. Users can choose the memory based on their characteristics.



- 16 KB EEPROM: to store not frequently changed parameters.
- microSD: to implement portable data logging applications.
max. of 2 GB on the MiniOS7 platform
max. of 32 GB on Linux and WinCE platforms
- 256 MB NAND Flash: rugged data storage to resist shock and vibration.
- 512 KB battery backup SRAM: to retain data while power lost for 5 years; no write cycle limitation.



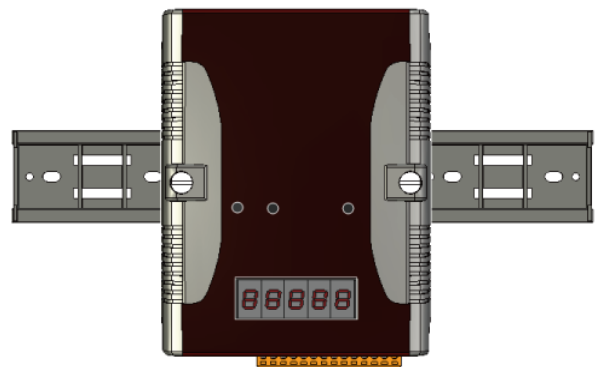
➤ Unique 64-bit Hardware Serial Number to Protect Your Program



A unique 64-bit serial number is assigned to each hardware device to protect your software against piracy.

➤ Small and Easy Installation

μPAC -5000 series have a slender shape (91 mm x 132 mm x 52 mm) to be installing in a narrow space with DIN-Rail.



➤ Plastic and Metal Housing

The default case is plastic material. Metal casing is also offered to provide extra security.

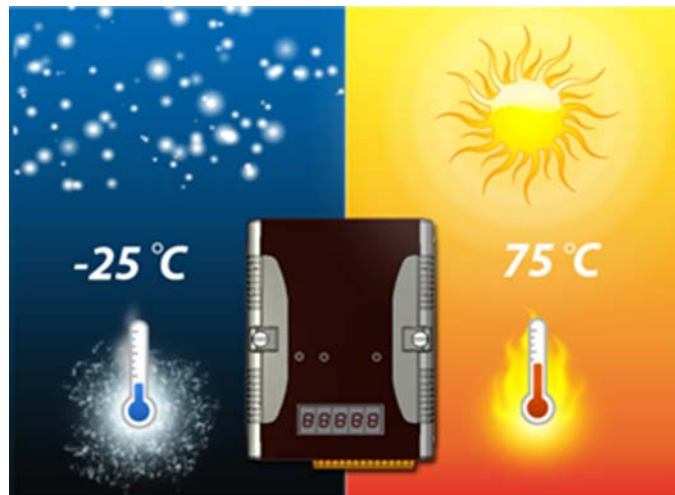
➤ Redundant Power Inputs

To prevent the μ PAC-5000 from failing by the power loss, the power module is designed with two input connectors. Once a power input fails, the power module switches to the other power input. And there is a relay output for informing the power failure.

➤ Highly Reliable Under Harsh Environment

The μ PAC-5000 operates in a wide range of temperature and humidity.

- Operating Temperature:
-25°C ~ +75 °C
- Storage Temperature:
-40°C ~ +80 °C
- Humidity:
10 ~ 95% RH (non-condensing)



1.3. Specifications

1.3.1. Common Specifications

Models	μPAC5000 Series	μPAC5000-FD Series	μPAC5000-SM Series
System Software			
OS	MiniOS7 (DOS-like embedded operating system)		
Program Upload Interface	RS-232 (COM1) or Ethernet		
Programming Language	C language		
Compilers to Create .exe Files	TC++ 1.01 TC2.01 BC++3.1 ~ 5.2x MSC 6.0 MSVC++ (Prior to version 1.5.2)		
CPU Module			
CPU	80186 or compatible (16-bit and 80MHz)		
SRAM	512 KB		
Flash	512 KB; erase unit is on sector (64K bytes); 100,000 erase/write cycles		
microSD Expansion	Yes, can support 1 or 2 GB microSD		
NAND Flash Disk	-	256 MB	-
Battery Backup SRAM	-		512 KB
EEPROM	16 KB		
NVRAM	31 Bytes (battery backup, data valid up to 5 year)		
RTC (Real Time Clock)	Provide second, minute, hour, date of week, month and year, valid from 1980 to 2079		
64-bit Hardware Serial Number	Yes, for Software Copy Protection		
Watchdog Timers	Yes (0.8 second)		
Communication Ports			
Ethernet	RJ-45 x 1, 10/100 Base-TX (Auto-negotiating, Auto MDI/MDI-X, LED indicators)		
COM1	RS-232 (Tx, Rx, RTS, CTS, GND), non-isolated, speed: 115200 bps max.		
COM2	RS-485 (D2+, D2-), self-tuner ASIC inside, non-isolated, speed: 115200 bps max.		
LED Indicator			
Programmable LED Indicators	2		
LED Display	5-digit 7-segment LED display for (D) versions		
Hardware Expansion			
I/O Expansion Bus	Yes (for only XW-Board only)		
Mechanical			
Dimension (W x H x D)	91 mm x 123 mm x 52 mm		
Installation	DIN-Rail		
Environmental			
Operating Temperature	-25 ~ + 75 °C		
Storage Temperature	-30 ~ +80 °C		
Ambient Relative Humidity	10 ~ 90 % RH (non-condensing)		
Power			
Protection	Power reverse polarity protection		
Frame Ground	Yes (for ESD protection)		
Input Range	+12 ~ +48 VDC		
Isolation	-		
Redundant Power Inputs	Yes		
Power Consumption	2 W; 2.5 W for (D) version		

1.3.2. GPS Specifications

GPS specifications of μ PAC-5100, μ PAC-5600 and μ PAC-5700 series.

GPS	
Channels	32 channels all-in-view tracking
Sensitivity	-159 dBm
Acquisition Rate	Cold start: 42 seconds; warm start: 35 seconds; reacquisition rate: 0.1 second
Accuracy	Position: 25 m CEP (S/A off); Velocity: 0.1 second (S/A off); Time: ± 1 ms
Protocol	NMEA

1.3.3. 2G (GPRS) Specifications

2G (GPRS) specifications of μ PAC-5200 and μ PAC-5600 series.

2G (GPRS)	
Band	850/900/1800/1900 MHz
GPRS Multi-slot	Class 10/8
GPRS Mobile Station	Class B
GPRS Class 10	Max. 85.6 kbps
CSD	Up to 14.4 kbps
Compliant to GSM phase 2/2+	Class 4 (2 W @ 850/900 MHz); Class 1(1W @ 1800/1900 MHz)
Coding Schemes	CS 1, CS 2, CS 3, CS 4
SMS	Text and PDU mode

1.3.4. 3G (WCDMA) Specifications

3G (WCDMA) specifications of μ PAC-5300 and μ PAC-5700 series.

3G (WCDMA)	
Band	UMTS: 2100/1900/850 MHz
Data Transfer	UMTS/HSDPA/HSUPA Upload: Max. 5.76 Mbps Download: Max. 7.2 Mbps
CPU Module	
Band	850/900/1800/1900 MHz
GPRS Multi-slot	Class 10/8
GPRS Mobile Station	Class B
GPRS Class 10	Max. 85.6 kbps
CSD	Up to 14.4 kbps
Compliant to GSM phase 2/2+	Class 4 (2 W @ 850/900 MHz); Class 1(1W @ 1800/1900 MHz)
Coding Schemes	CS 1, CS 2, CS 3, CS 4
SMS	Text and PDU mode

1.3.5. ZigBee Specifications

ZigBee specifications of μ PAC-5400 series.

	ZigBee Host	ZigBee Slave
RF channels	16	
Receive sensitivity	-102 dBm	
Data encryption	AES-CRT/AES-128	-
Transmit power	9 dBm	
Network topology support	Star, Mesh and Cluster Tree	
Antenna (2.4 GHz)	3 dBi Omni-Directional antenna	
Transmission range (LOS)	100 m	

1.3.6. Wi-Fi Specifications

Wi-Fi specifications of μ PAC-5500 series.

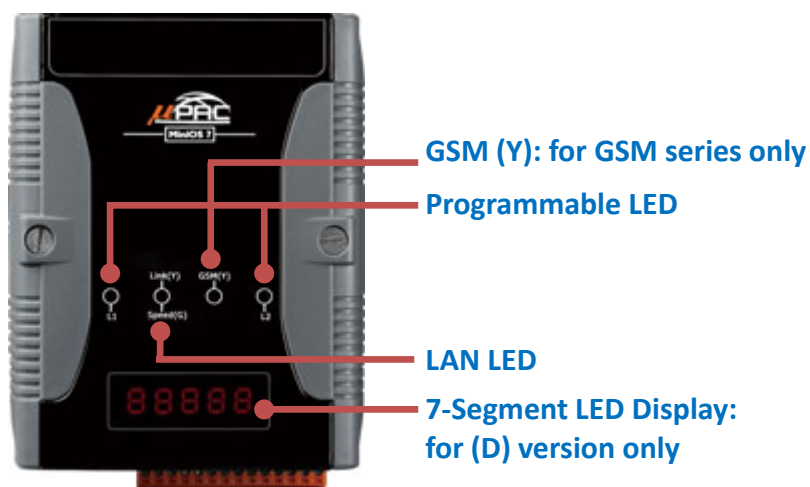
CPU Module	
Protocol	IEEE 802.11 b/g
Frequency Range	2.412GHz ~ 2.484GHz
Channel	13 channels
Security	WEP-64/ WEP-128/WPA-TKIP/WPA-AES
Receive sensitivity	-87 dBm(IEEE 802.11b) / -72 dBm (IEEE 802.11g)
Transmit Power	12 dBm(IEEE 802.11b) / 14 dBm(IEEE 802.11g)

1.4. Overview

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

Front Panel

The LED indicators and 5-digit 7-Segment LED display are on the front panel that provides a very convenient way of displaying information for faster, easier diagnostics.

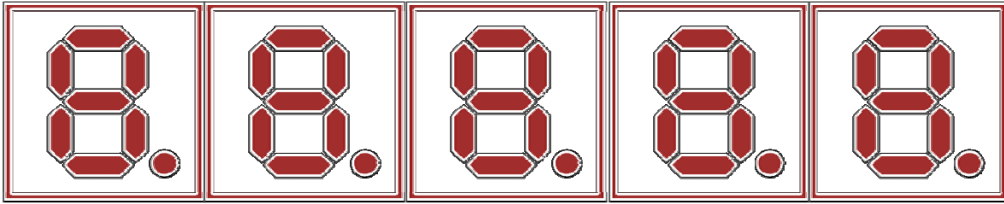


► LED Indicators

LED indicators are on the front panel of the μPAC-5000, their functions are summarized in the table below.

Indicator	State	Meaning
L1	Flashing	User programmable LED
L2	OFF	User programmable LED
Link (G)	Permanently on	Ethernet link detected
	Permanently off	No Ethernet link detected
	Flashing green	Ethernet packet received
GSM (Y) (for GSM series)	Slow blanking (Once every 3 seconds)	μPAC-5000 is registered full service
	Fast blanking (Once every 1 second)	1. μPAC-5000 is searching GSM network 2. SIM card is registering to GSM network

➤ 5-Digit 7-SEG LED display (for display version only)



μPAC-5000 display series equip 5-digit 7-SEG LED display that can be used to display decimal numbers from 0 to 9 and provided a very convenient way of displaying digital data in the form of Numbers.

Top Panel

The microSD memory socket is on the top panel that provides a simple way of expanding capacity.



➤ microSD Memory Socket

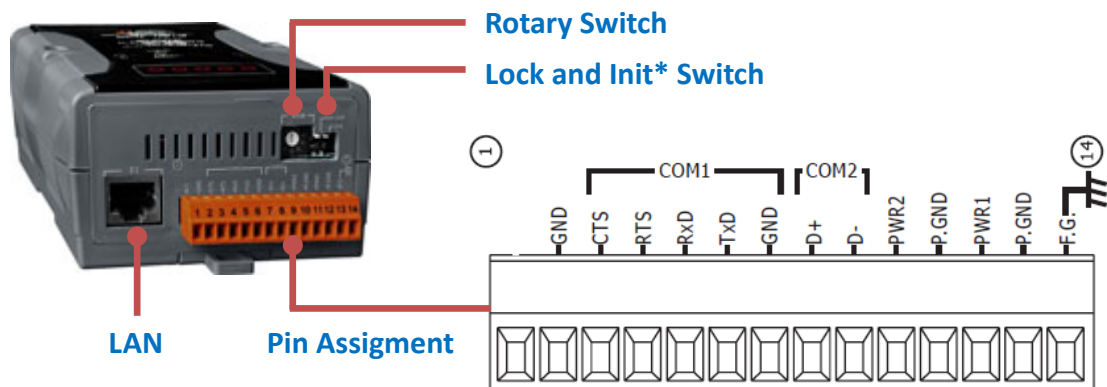
μPAC-5000 equip a microSD slot and it can support up to 2 GB microSD card.

➤ SIM Card Socket and GSM/GPRS Antenna

μPAC-5200 Series equip a SIM card slot and GSM Antenna.

Bottom Panel

The switches and interface are on the bottom panel that provides a simple way of adjusting the system and wiring the connection.



➤ Init Switch: Operating Mode Selector Switch

ON: Init mode (MiniOS7 configuration mode)

OFF: Normal mode (Firmware running mode)

In the μ PAC-5000 series, the switch is always in the OFF position. Only when updating the μ PAC-5000 firmware or OS, the switch can be moved from the OFF position to the ON position.

Move the switch to the OFF position after the update is complete.

➤ Lock Switch: Flash Memory Write Protection Switch

ON: Enable Write Protection

OFF: Disable Write Protection

μ PAC-5000 Flash memory with Write Protection can physically lock that prevents modification or erasure of valuable data on μ PAC-5000.

► LAN

An Ethernet port is an opening on μ PAC-5000 network equipment that Ethernet cables plug into. Ethernet ports accept cables with RJ-45 connectors.

► Pin Assignment

The pin assignments of the connector are as follows:

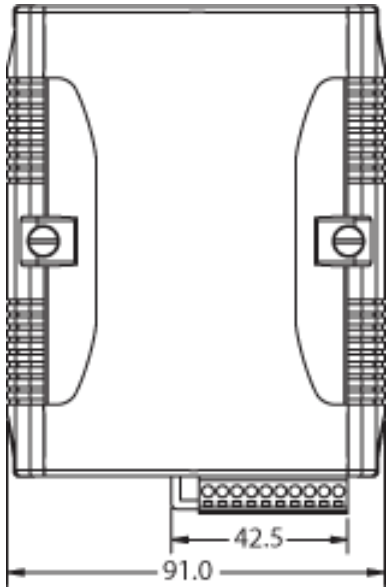


Pin	Signal	Description
1	N/A	Unassigned
2	GND	Ground
3	CTS	COM1 (RS-232)
4	RTS	
5	RxD	
6	TxD	
7	GND	
8	D+	COM2 (RS-485)
9	D-	
10	PWR2	Power Input 2
11	P.GND	
12	PWR1	Power Input 1
13	P.GND	
14	F.G.	Frame Ground

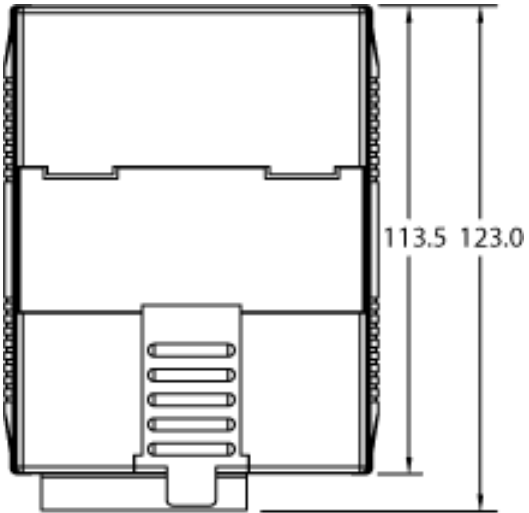
1.5. Dimensions

All dimensions are in millimeters.

Front View



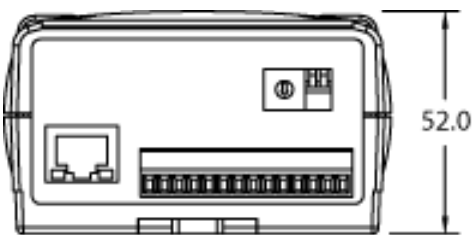
Back View



Top View

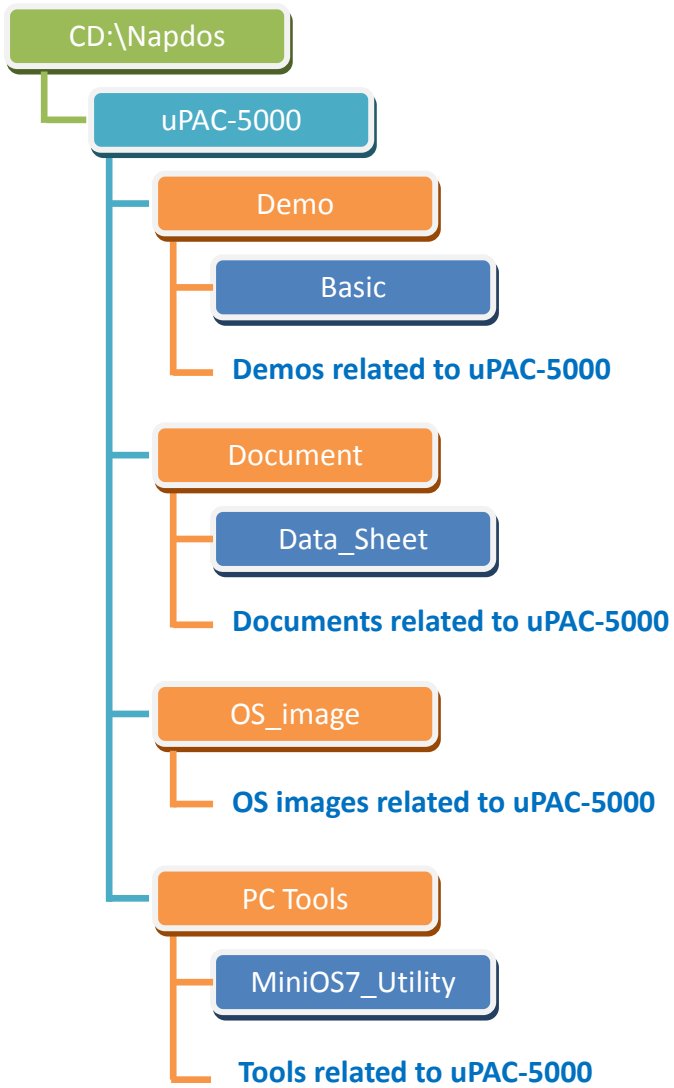


Bottom View



1.6. 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 μ PAC-5000.

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:



μ PAC-5000



Software Utility CD



RS-232 cable (CA-0910)



Screw Driver (1C016)



**GSM/GPRS Antenna (ANT-421-02)
(for GSM series only)**

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 “1.3. Specifications”

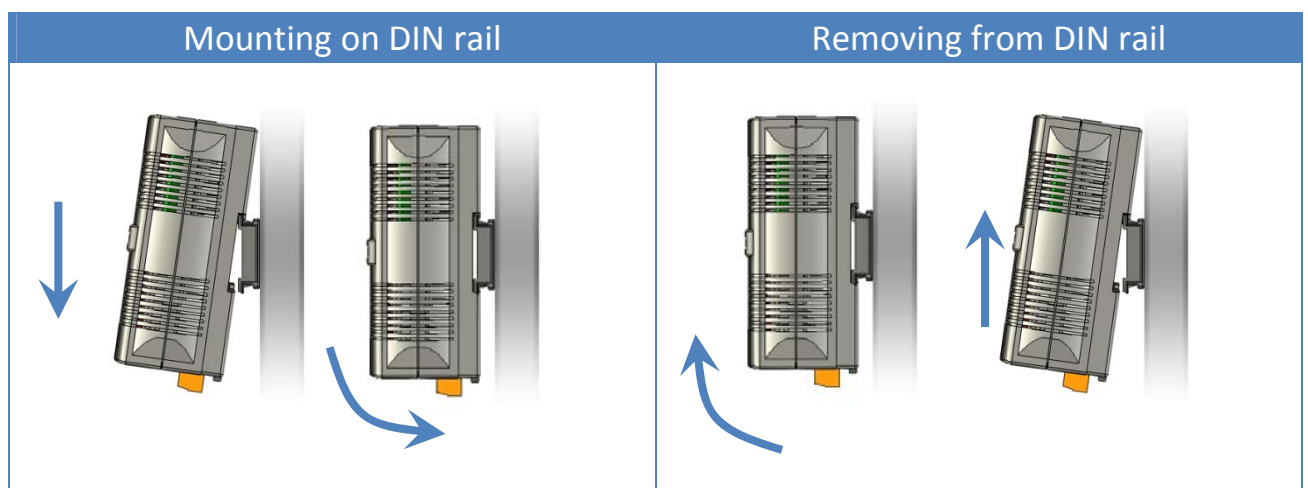
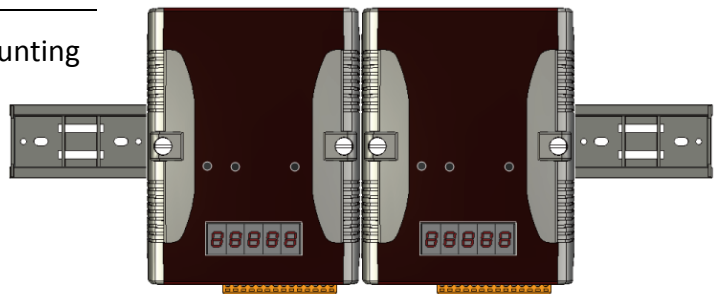
Below are step-by-step instructions for deploying the basic μ PAC-5000 system.

Step 1: Mount the hardware

The μ PAC-5000 can be mounted with the bottom of the chassis on the DIN rail or piggyback.

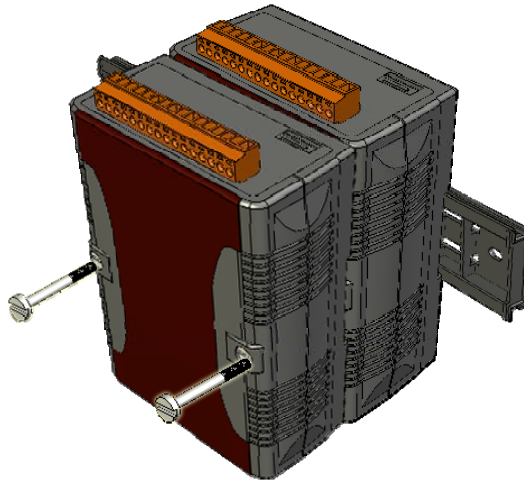
► DIN Rail mounting

The μ PAC-5000 has simple rail clips for mounting reliably on a standard 35 mm DIN rail.



➤ Piggyback mounting

The μ PAC-5000 has two holes on both sides for piggyback mounting.

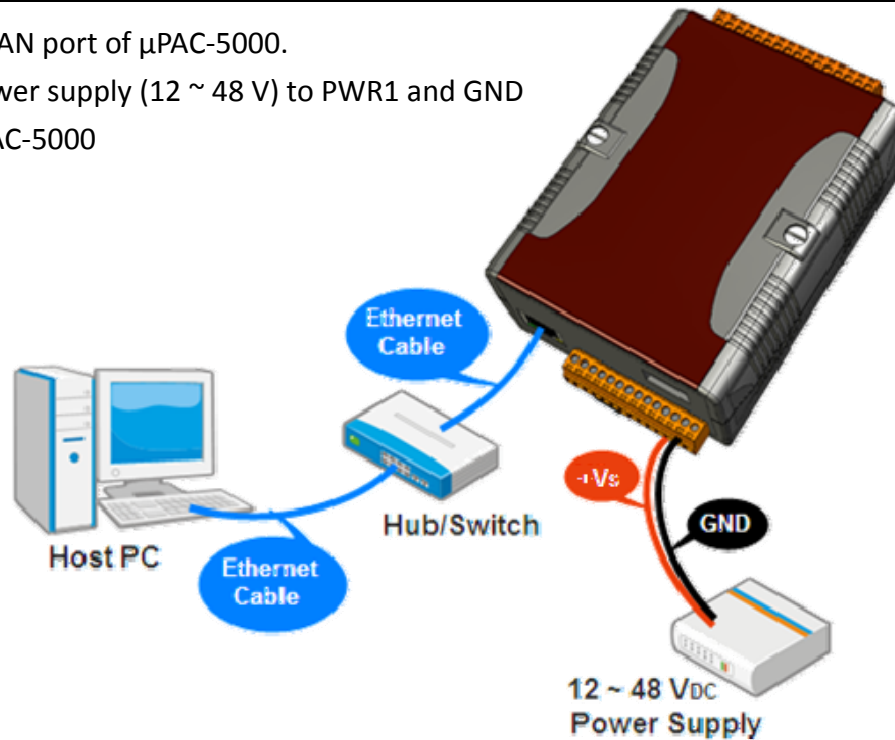


Step 2: Connect the μ PAC-5000 to PC and setting up the power supply

The μ PAC-5000 equip an RJ-45 Ethernet port for connection to an Ethernet hub/switch and PC, and powered by a standard 12 V_{DC} power supply .

➤ External power supply via a standard 12 V_{DC} power supply

- i. Connect PC to LAN port of μ PAC-5000.
- ii. Connect the power supply (12 ~ 48 V) to PWR1 and GND terminals of μ PAC-5000



2.2. GSM Series Installation

For first time users, please follow the steps below to complete the pre-installation.

2.2.1. Installing the SIM card

With the contact area face-up, insert the SIM card into the SIM card holder. Make sure the SIM card fits in the SIM card holder properly and in the correct direction. The contact area on the SIM card should be facing upward. To eject SIM card, simply, use you finger nail and apply slight pressure.



Tips & Warnings



Always power off uPAC-5000 before installing or taking out SIM card.

2.2.2. Installing the GSM antenna

You need to install the antenna before using the GSM series module. Install the GSM antenna to antenna connector on the top panel tightly. Please refer to the figures below.



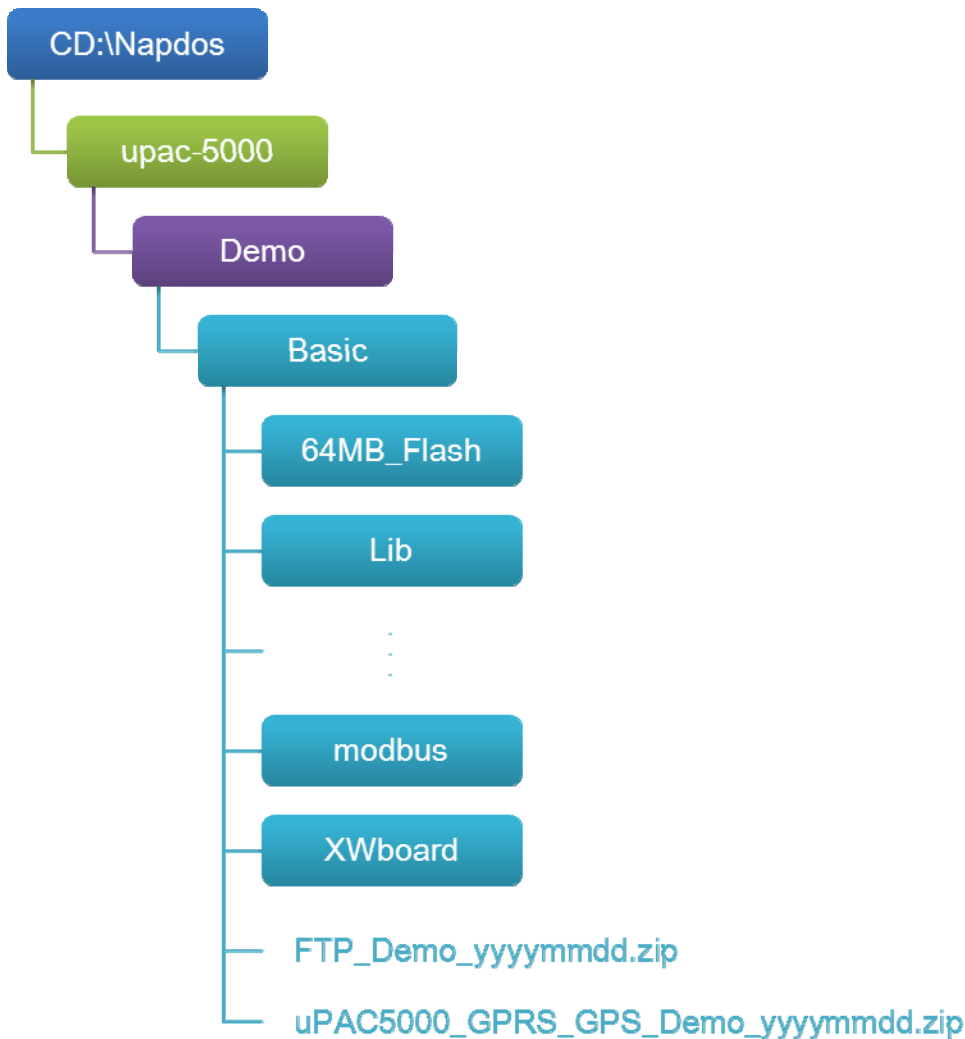
2.3. Software Installation

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 μ PAC-5000 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_V321.exe
[MiniOS7 Utility Ver 3.21] Setup

MiniOS7 Utility is a suite of tool for managing MiniOS7 devices (μ PAC-5000, iPAC-8000, μ PAC-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:

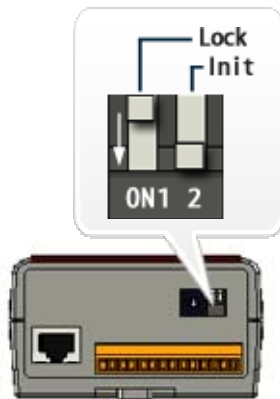
CD:\Napdos\minios7\utility\minios7_utility\

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

2.4. Boot Configuration

Before you upload some programs to μ PAC-5000, you need to enter the Init mode and disable the Write Protection.

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



2.5. Uploading μ PAC-5000 Programs

MiniOS7 Utility is a suite of tool for managing MiniOS7 devices (μ PAC-5000, iPAC-8000, μ PAC-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 μ PAC-5000 is connected to PC.

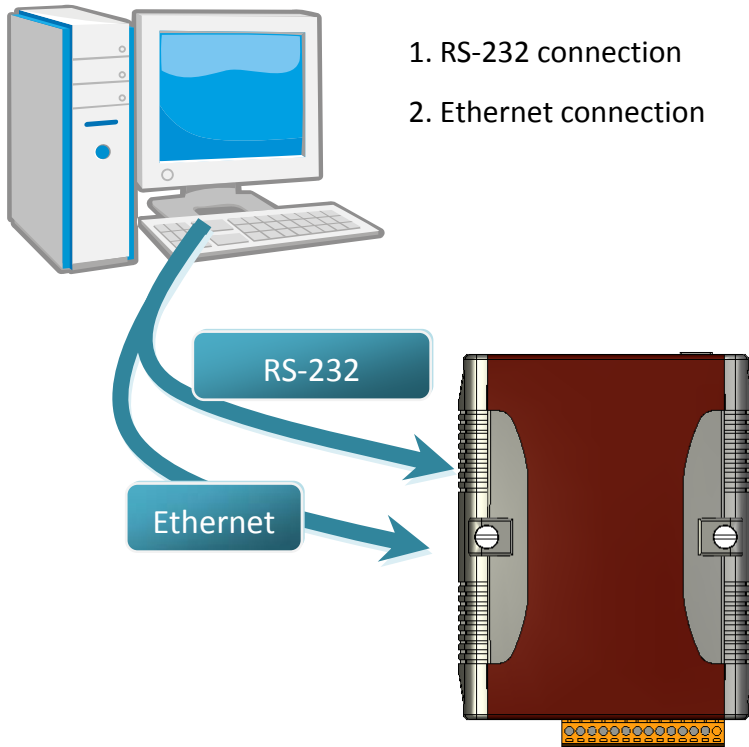
The upload process has the following main steps:

1. Establishing a connection between PC and μ PAC-5000
2. Uploading and executing programs on μ PAC-5000
3. Making programs start automatically

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

2.5.1. Establishing a connection between PC and μ PAC-5000

There are two ways to establish a connection between PC and μ PAC-5000.

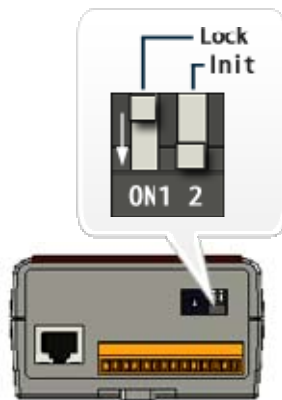


Each of the connection types will be described in detail later.

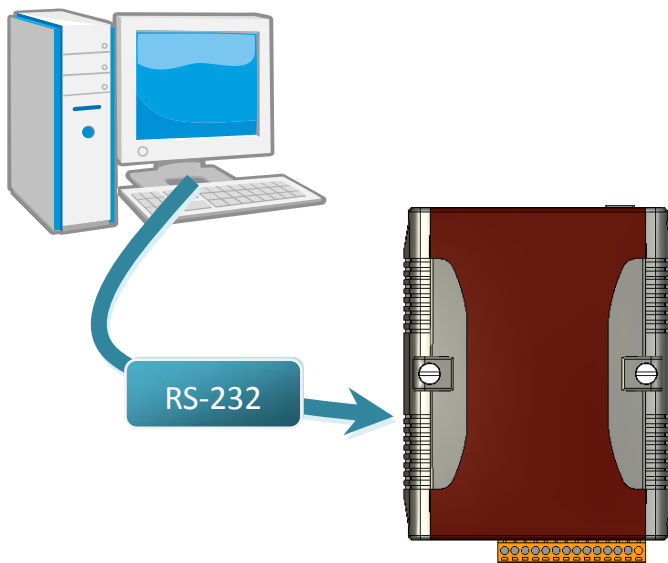
(a) RS-232 connection

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

Step 1: Ensure the switch of the Lock is in the “OFF” position, and the switch of Init is “ON” position. Then reboot the μ PAC-5000.

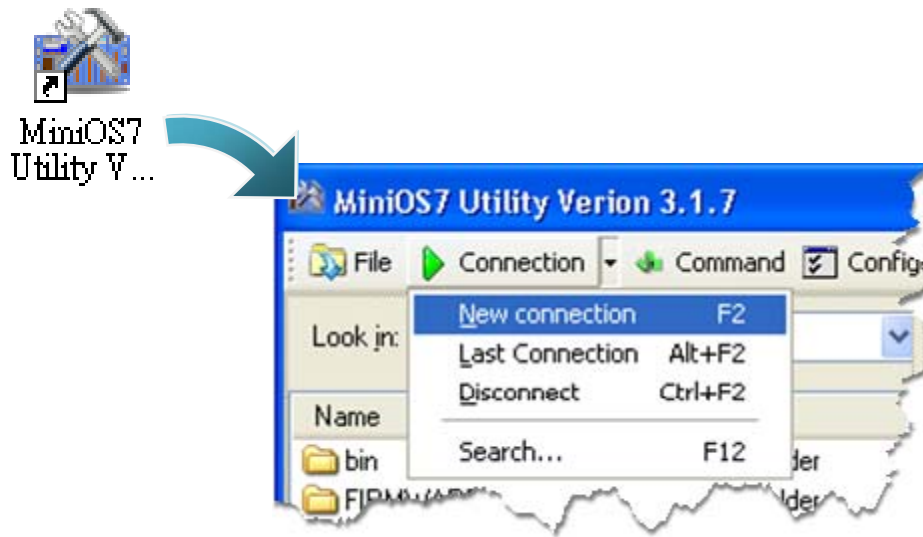


Step 2: Use the RS-232 Cable (CA-0910) to connect to PC

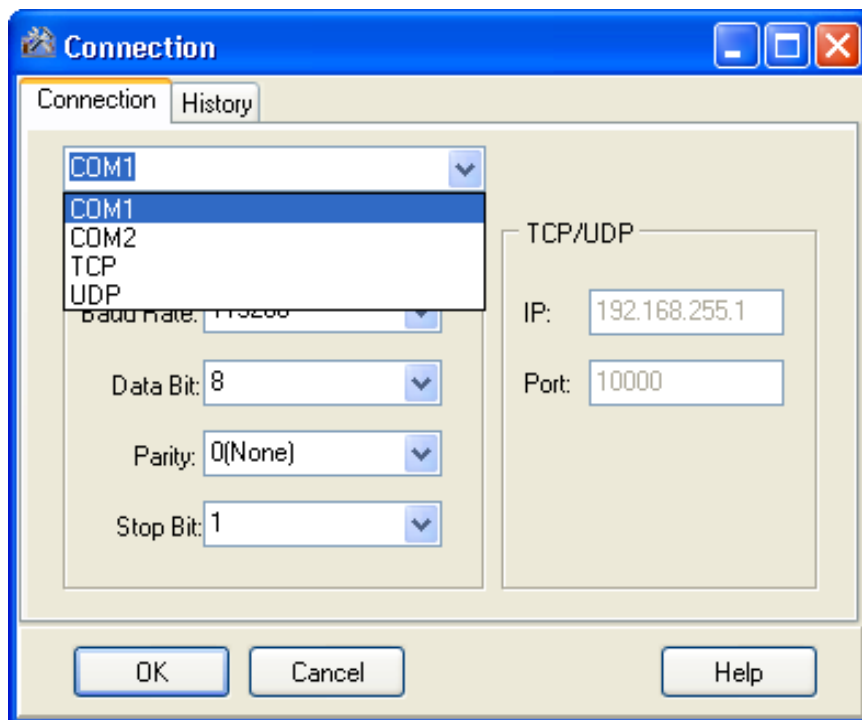


Step 3: Run the MiniOS7 Utility

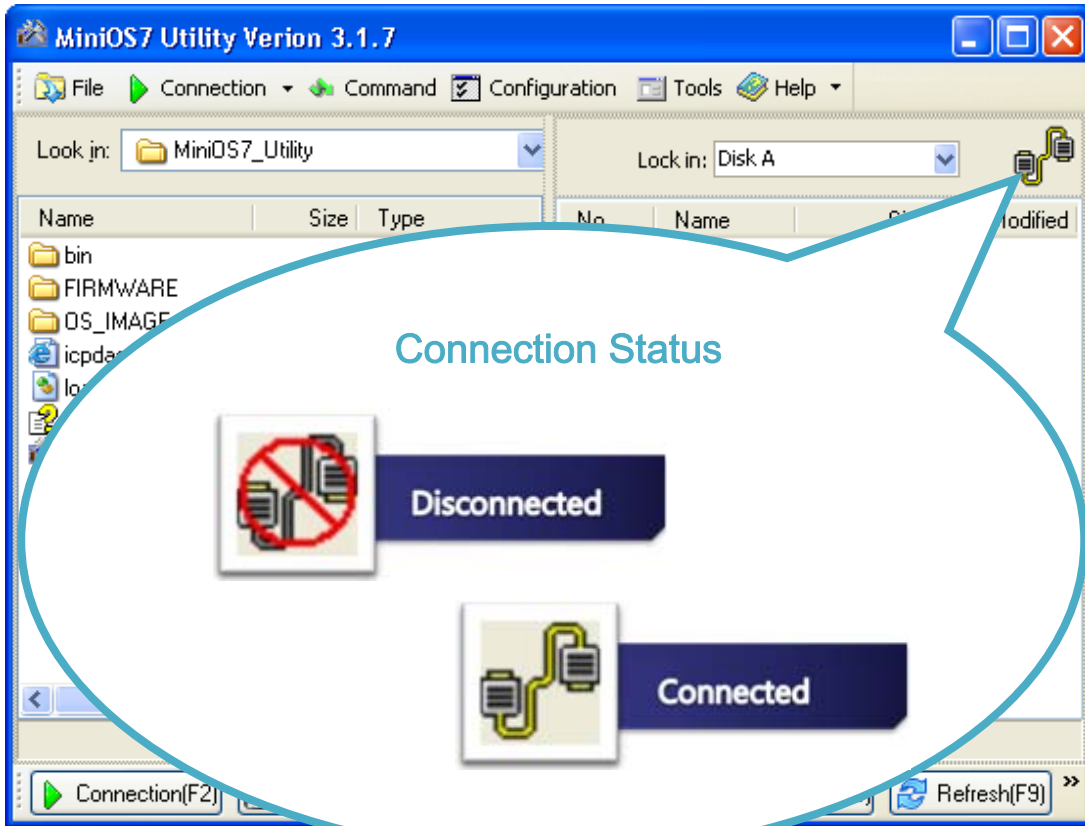
Step 4: Click the “New connection” function from the “Connection” menu



Step 5: On the “Connection” tab of the “Connection” dialog box, choose the COM port that your μ PAC-5000 is connecting to, and then click “OK”



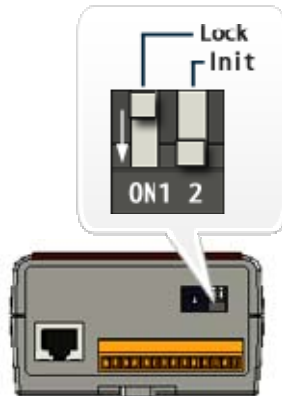
Step 6: The connection has already established



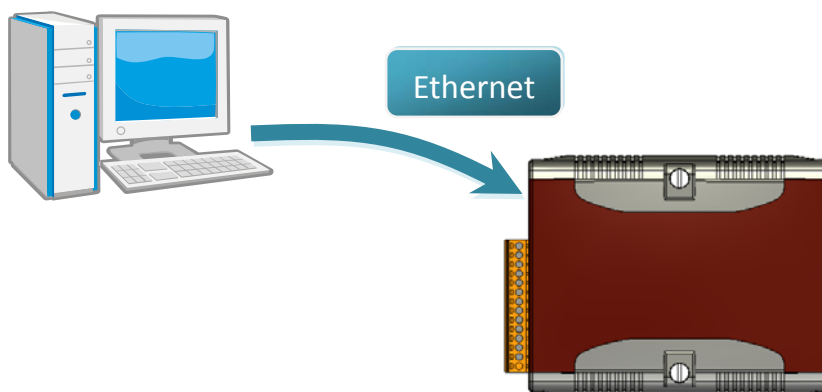
(b) Ethernet Connection

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

Step 1: Ensure the switch of the Lock is in the “OFF” position, and the switch of Init is “ON” position. Then reboot the μ PAC-5000.



Step 2: Use an Ethernet cable to connect to PC



Step 3: Run the MiniOS7 Utility

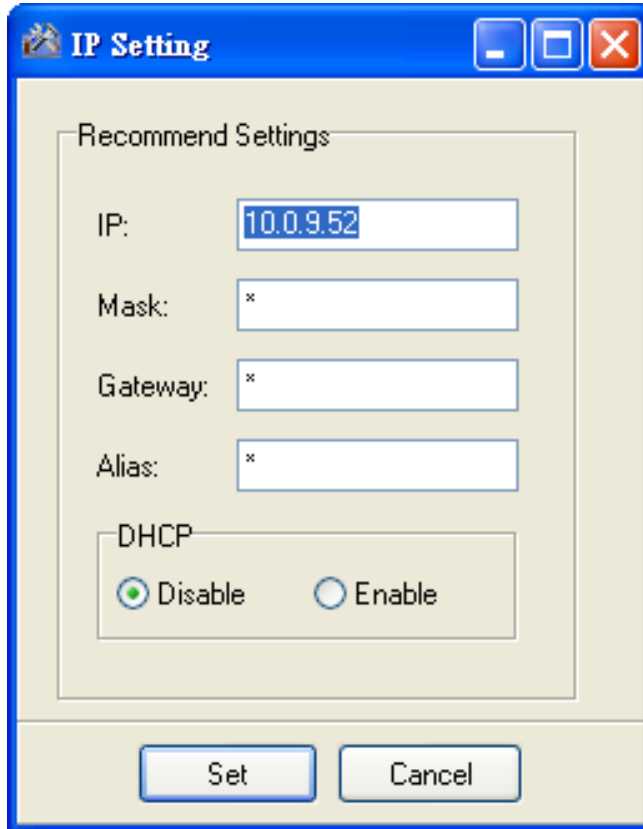
Step 4: Click the “Search” function from the “Connection” menu



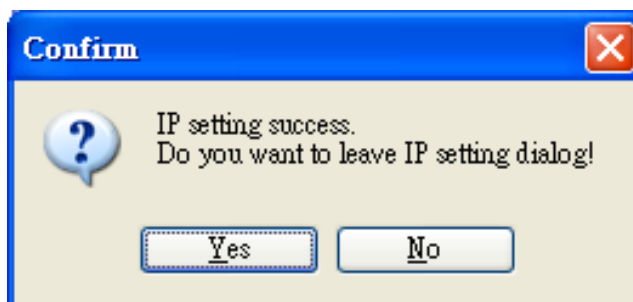
Step 5: On the “MiniOS7 Scan” dialog box, choose the module name from the list and then choose “IP setting” from the toolbar



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



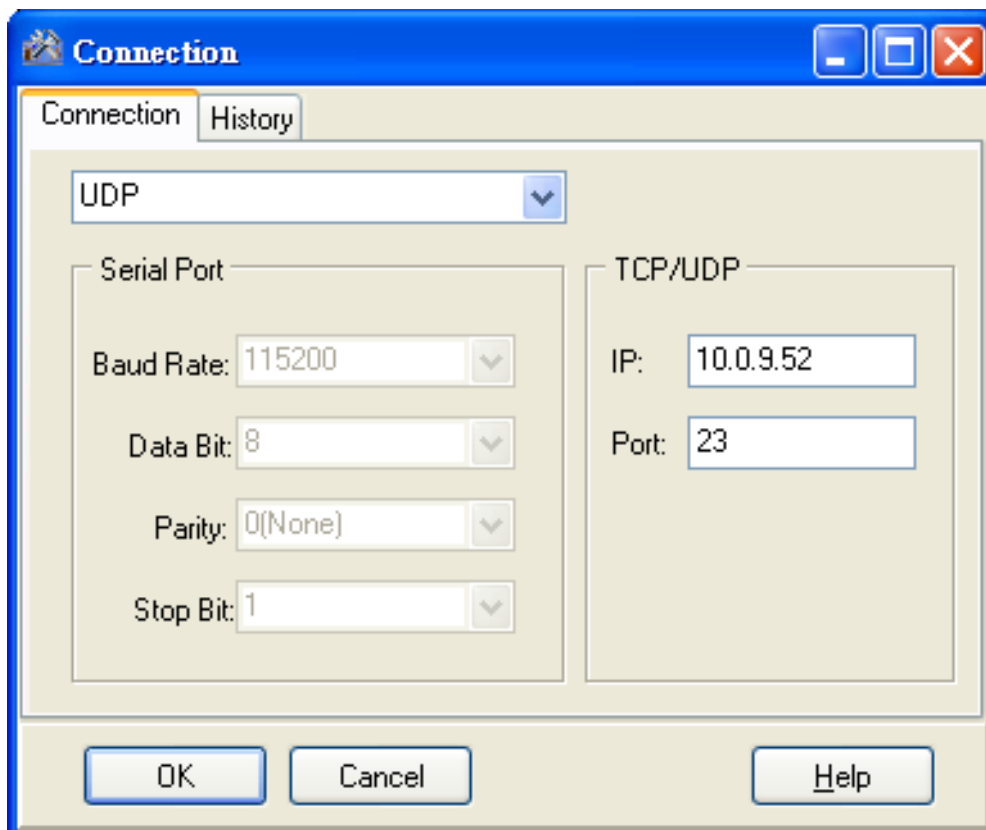
Step 7: On the “Confirm” dialog box, click “Yes”



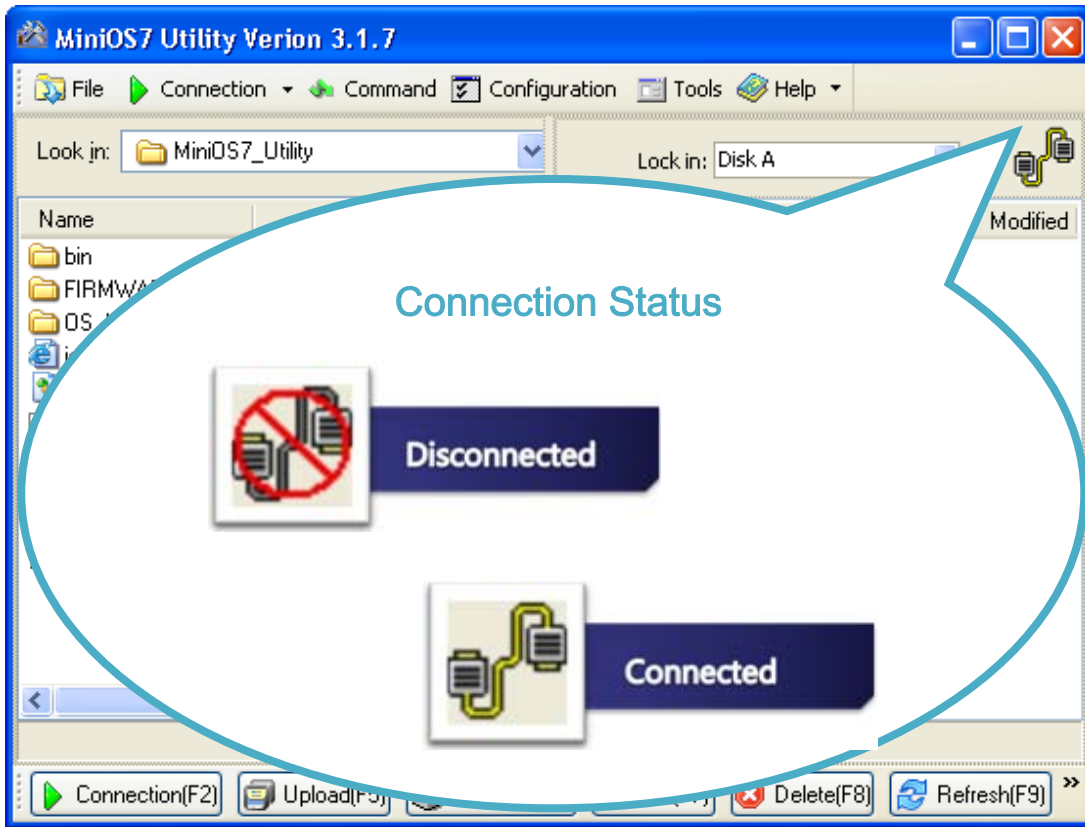
Step 8: Click the “New connection” function from the “Connection” menu



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”



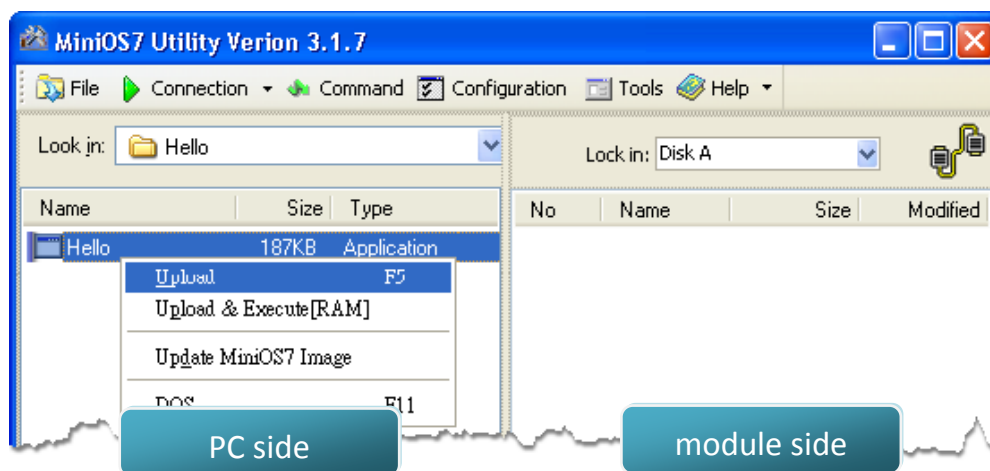
Step 10: The connection has already established



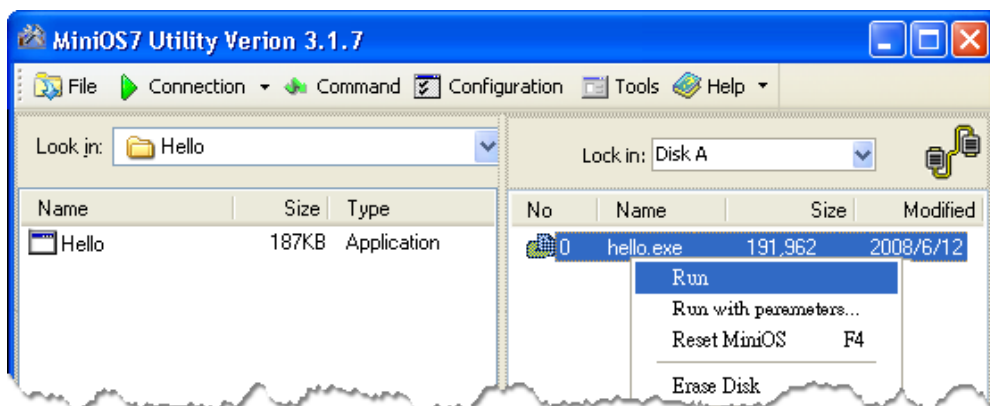
2.5.2. Uploading and executing μ PAC-5000 programs

Before uploading and executing μ PAC-5000 programs, you must firstly establish a connection between PC and μ PAC-5000, for more detailed information about this process, please refer to section “2.5.1. Establishing a connection between PC and μ PAC-5000”

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”



2.5.3. Making programs start automatically

After upload programs on the μ PAC-5000, if you need programs to start automatically after the μ PAC-5000 start-up, it is easy to achieve it, to create a batch file called autoexec.bat and then upload it to the μ PAC-5000, the program will start automatically in the next start-up.

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

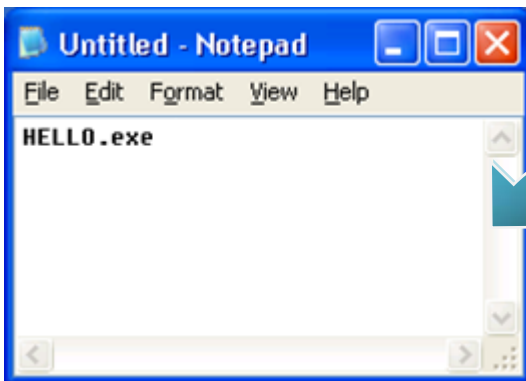
Step 1: Create an autoexec.bat file

i. Open the "Notepad"

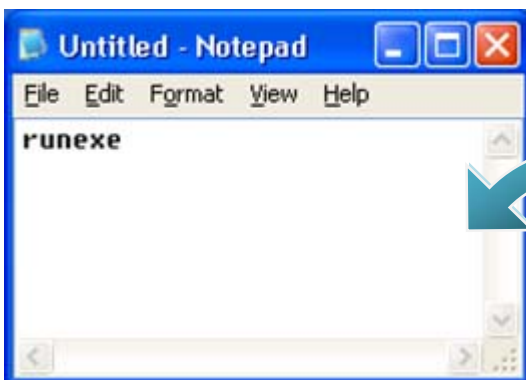
ii. Type the command

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



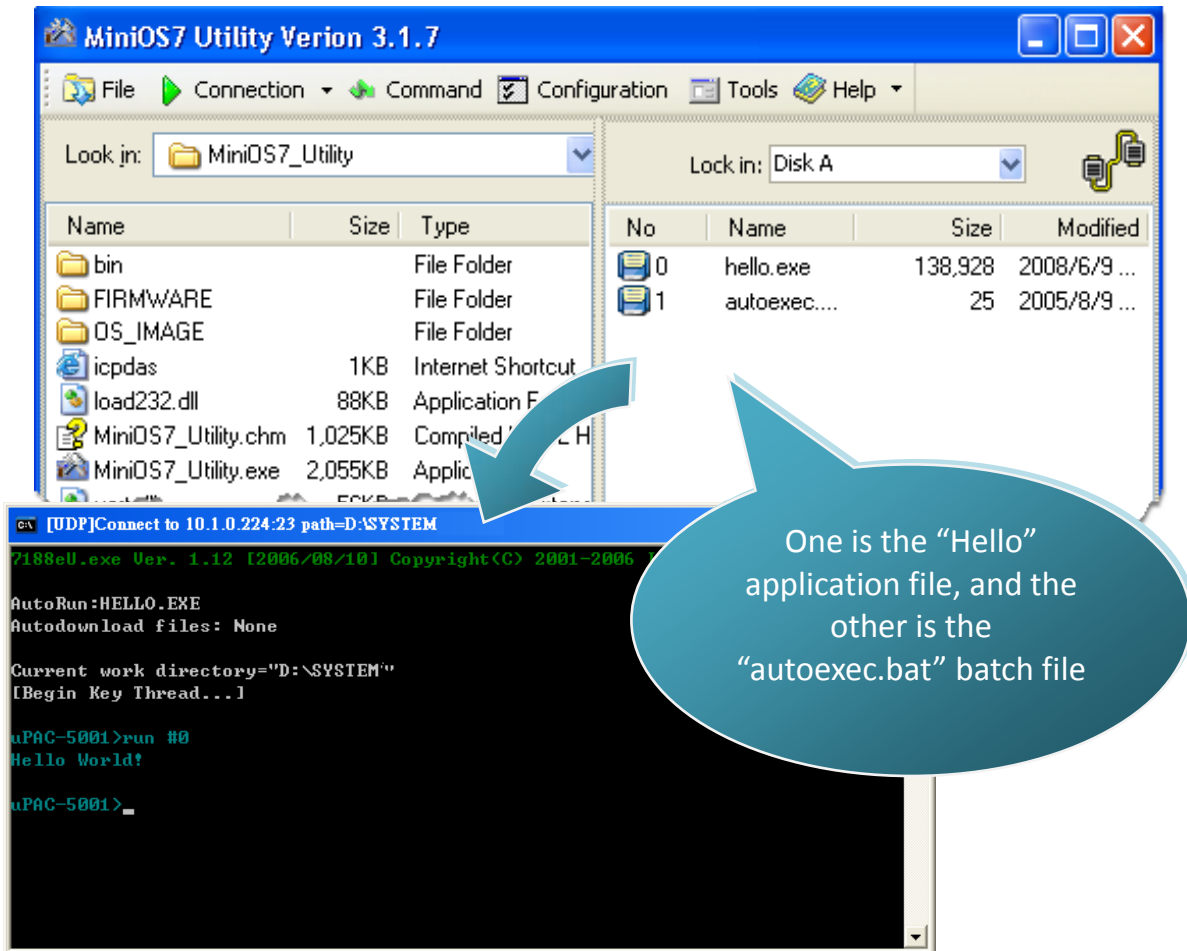
The file name:
Run the specified file.



Runexe:
Run the last execution file (.exe).

Step 2: Upload programs to μ PAC-5000 using MiniOS7 Utility

For more detailed information about this process, please refer to section “2.5.1. Establishing a connection between PC and μ PAC-5000”



Tips & Warnings



Before rebooting the module for settings to take effect, you must firstly turn the switch of Init to "OFF" position.



2.6. Updating μ PAC-5000 OS Image

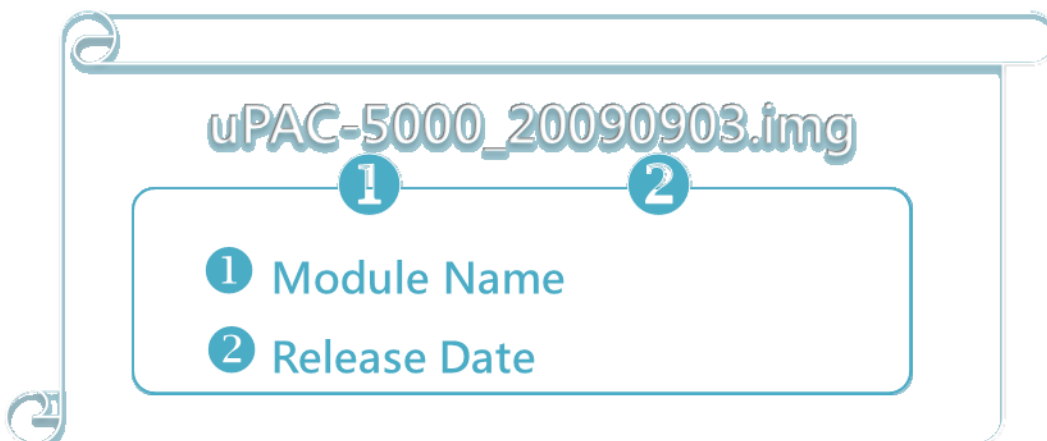
ICP DAS will continue to add additional features to μ PAC-5000 in the future, we advise you periodically check the ICP DAS web site for the latest update to μ PAC-5000.

Step 1: Obtain the latest version of the μ PAC-5000 OS image

The latest version of the μ PAC-5000 OS image can be obtained from:

CD:\NAPDOS\upac-5000\OS_image\

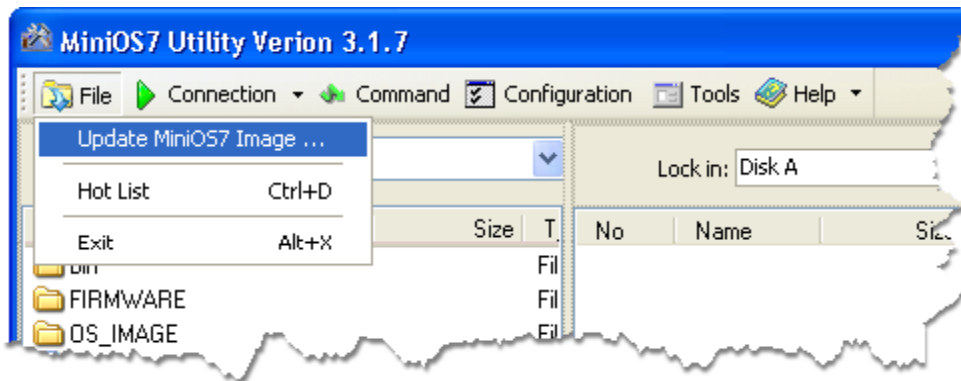
http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/os_image/



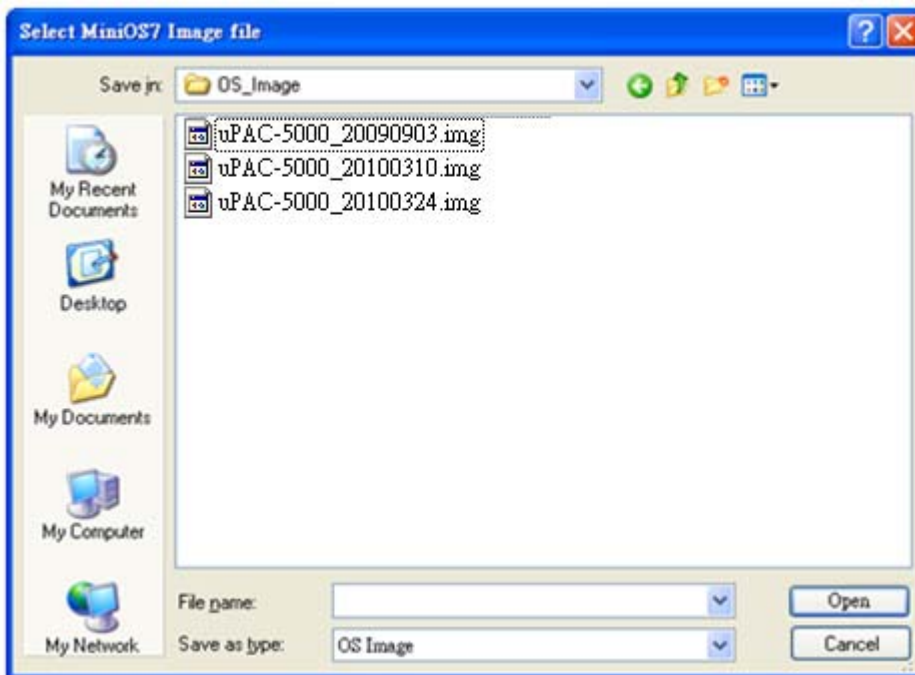
Step 2: Establish a connection

For more detailed information about this process, refer to section “2.5.1. Establishing a connection between PC and μ PAC-5000”

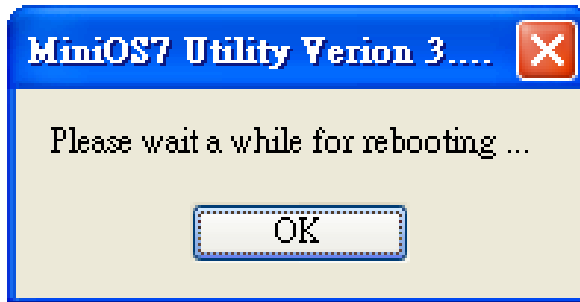
Step 3: Click the “Update MiniOS7 Image ...” from the “File” menu



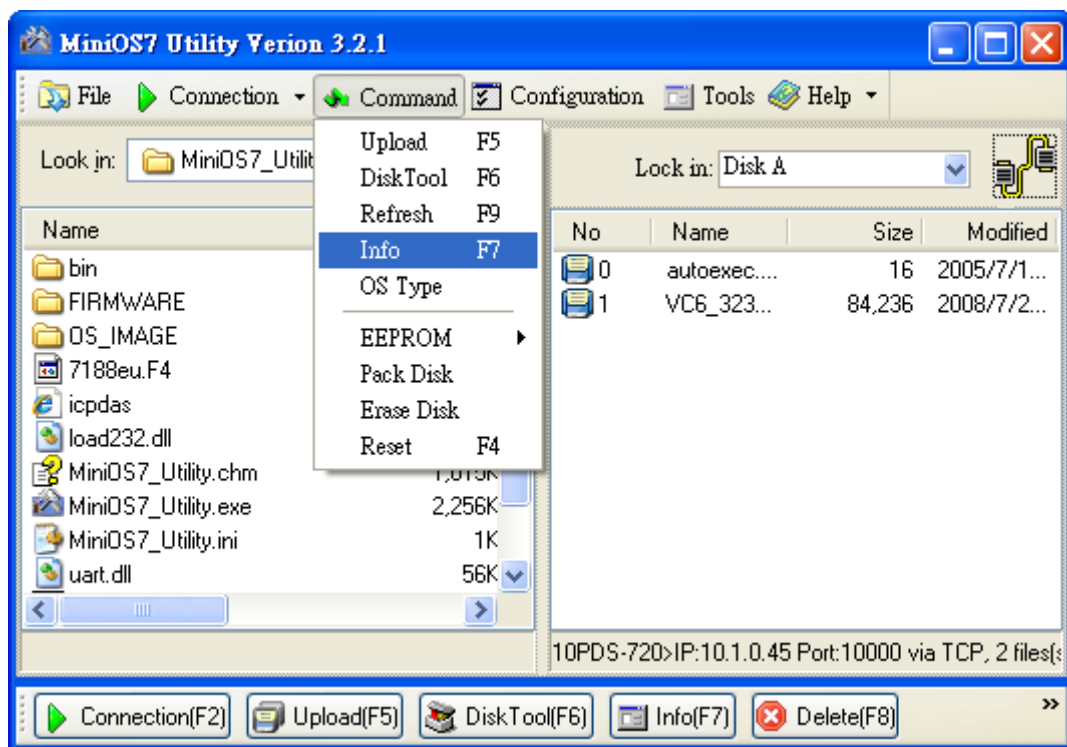
Step 4: Select the latest version of the MiniOS7 OS image



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 μ PAC-5000 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 μ PAC-5000 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++ (Prior to version 1.52)

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

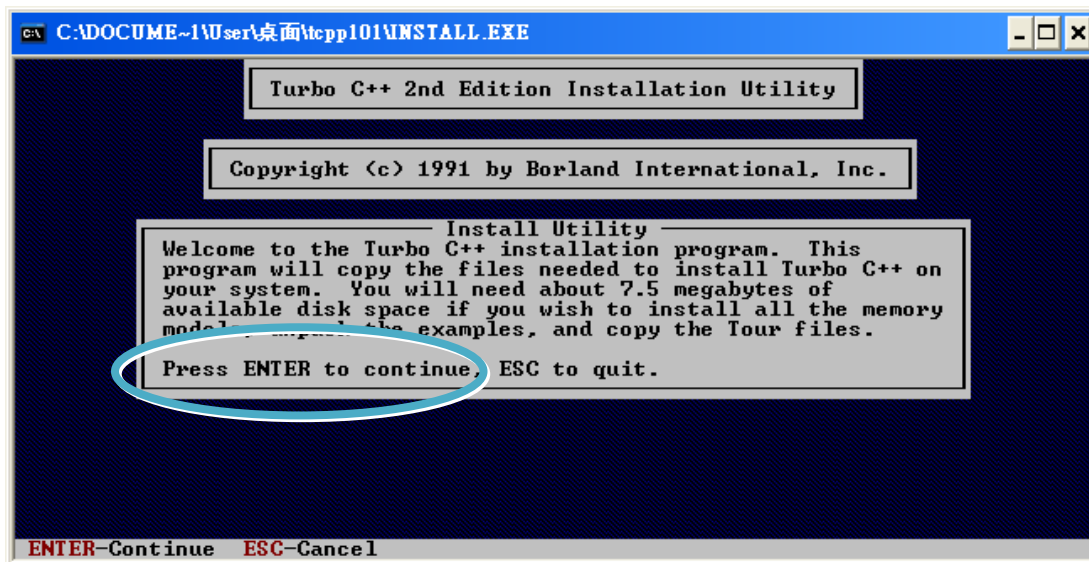
If there is no compiler currently installed on your system, installation of the compiler should be the first step.

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

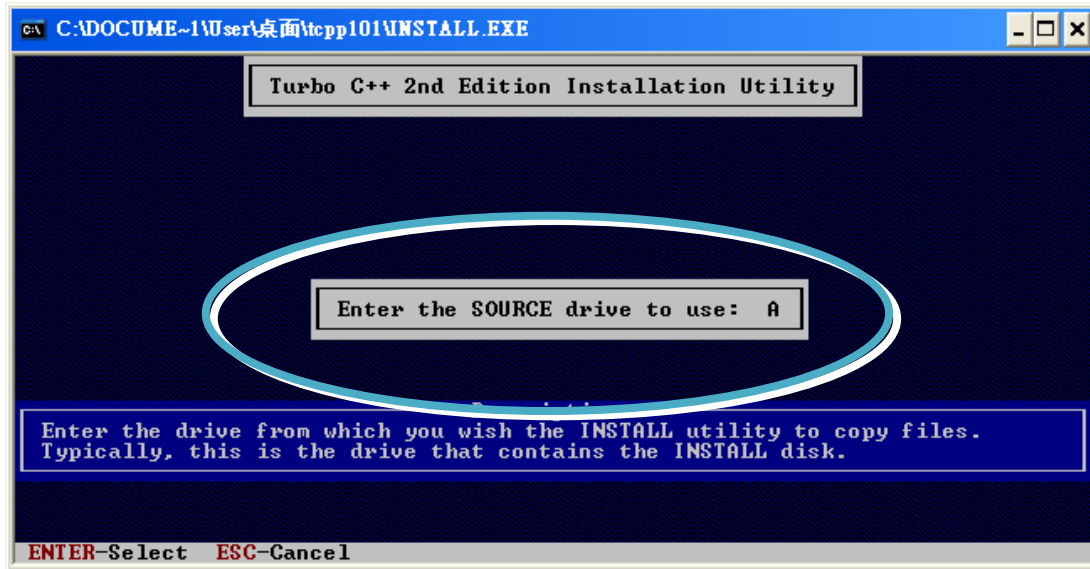
Step 1: Double click the Turbo C++ executable file to start setup wizard



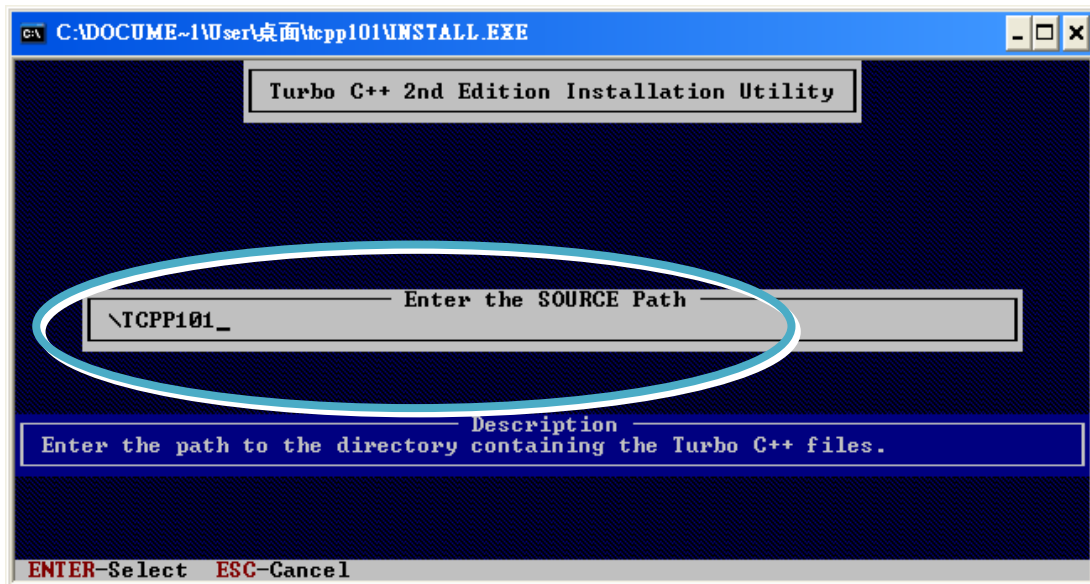
Step 2: Press "Enter" to continue



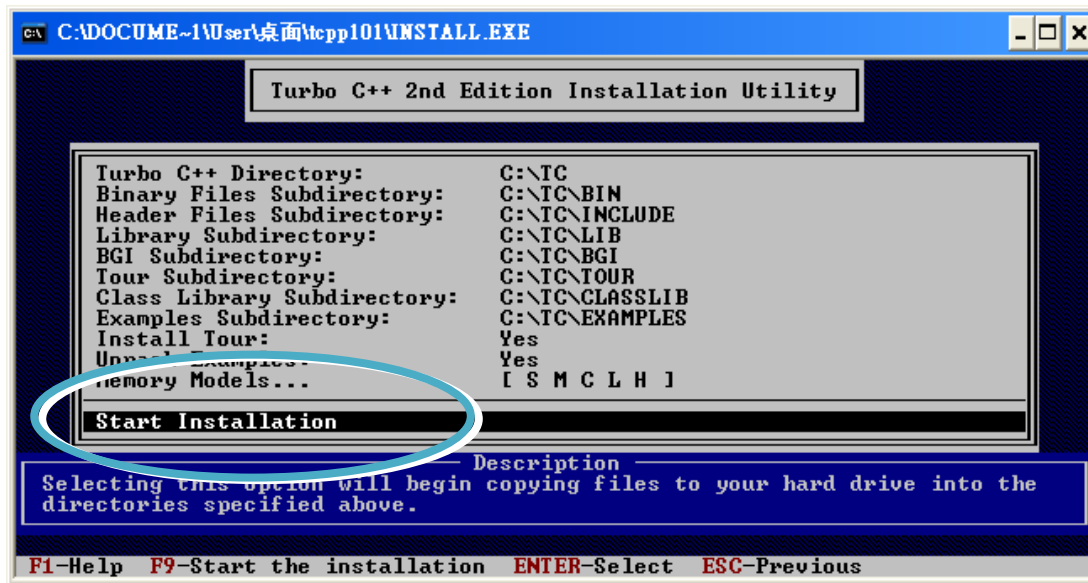
Step 3: Enter the letter of the hard drive you wish to install the software



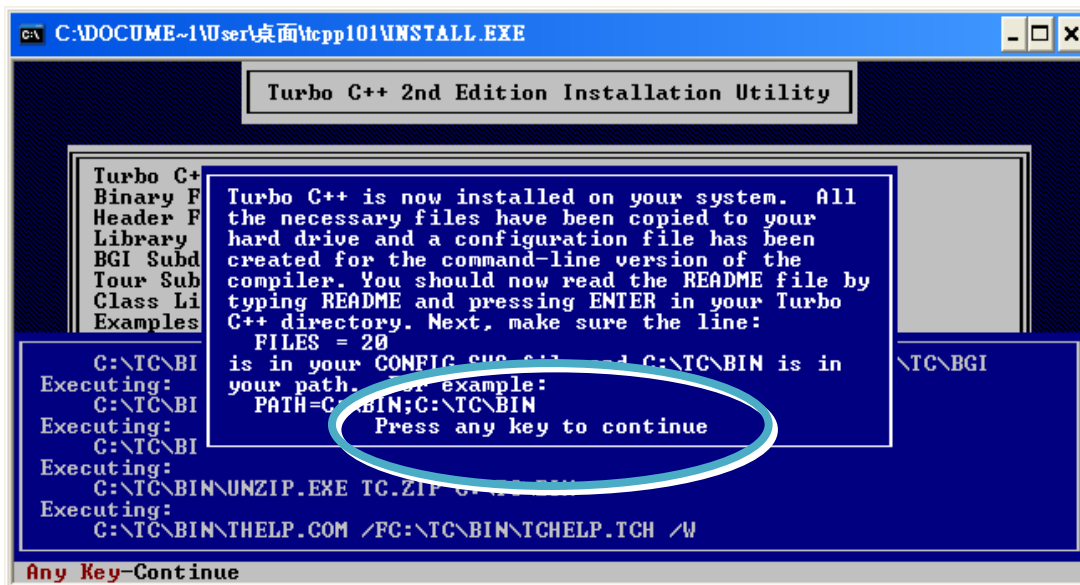
Step 4: Enter the path to the directory you wish to install files to



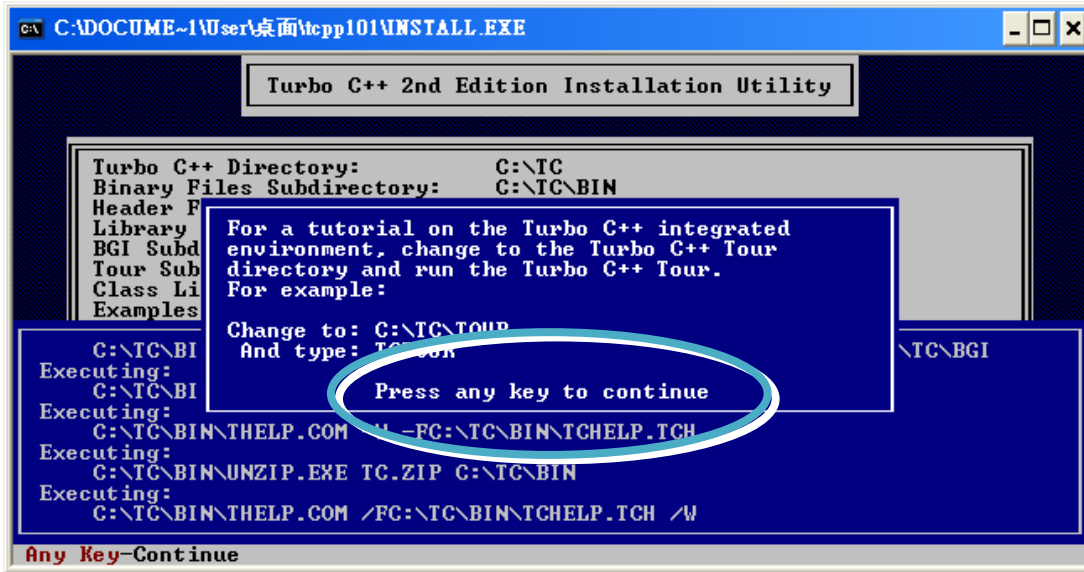
Step 5: Select "Start Installation" to begin the install process



Step 6: Press any key to continue



Step 7: Press any key to continue

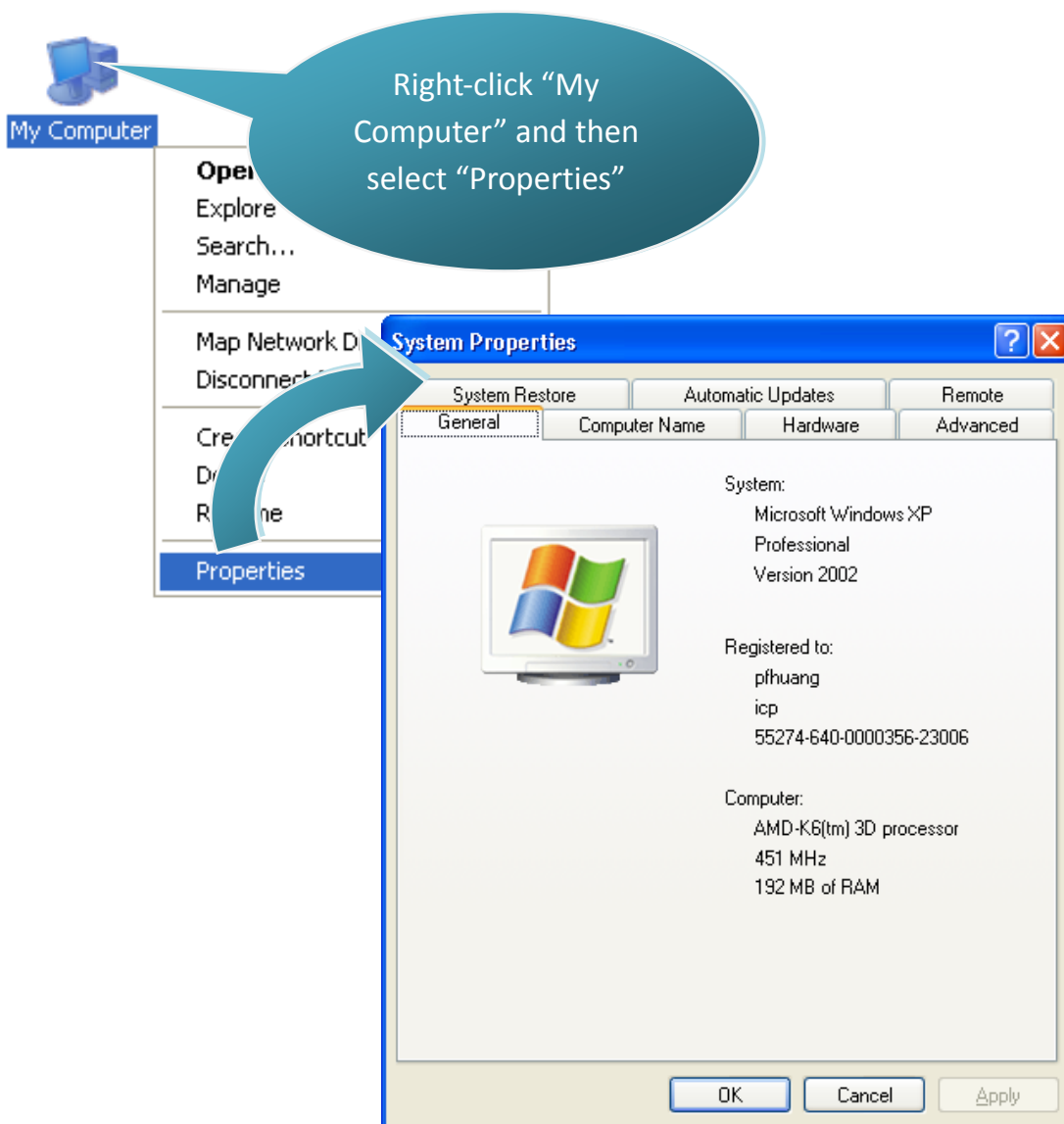


Step 8: Installation is complete

3.1.2. Setting up the environment variables

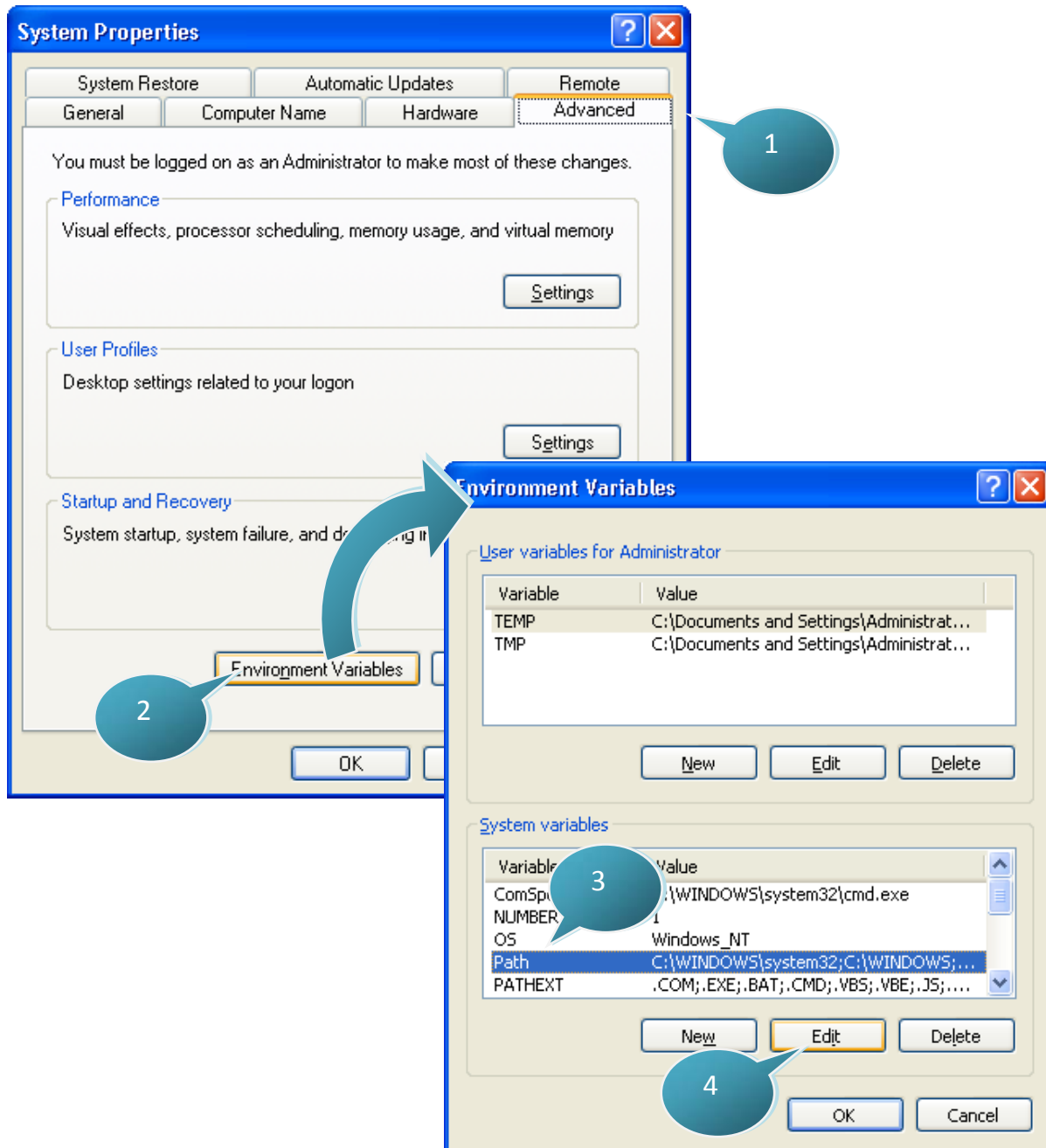
After installing the compiler, several compilers will be available from the Windows Command line. You can set the path environment variable so that you can execute this compiler on the command line by entering simple names, rather than by using their full path names.

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

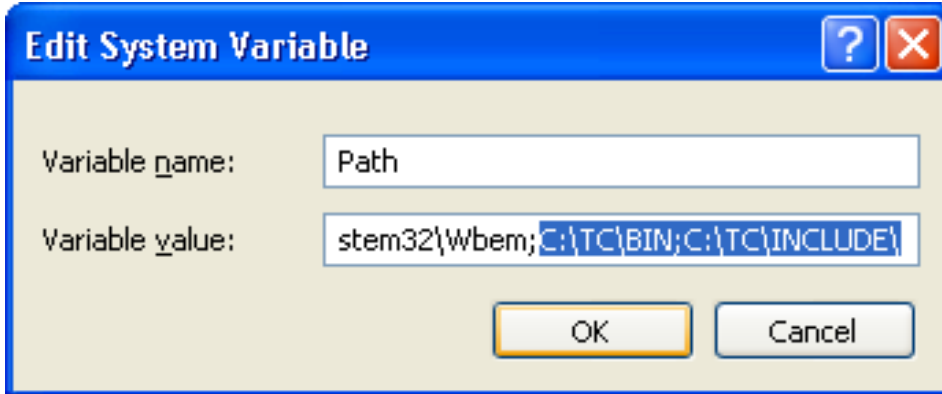
Step 3: On the “Environment Variables” dialog box, click the “Edit” button located in the “System variables” option



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. μ PAC-5000 APIs

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

For more detailed information regarding μ PAC-5000 APIs, please refer to

CD:\NAPDOS\upac-5000\Demo\basic\Lib\01_Lib_Update_History_20100507.txt

http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/lib/01_Lib_Update_History_20100507.txt

Before creating the application, ensure them that you have installed. If they are not installed, please refer to “section 2.3. Software Installation”.

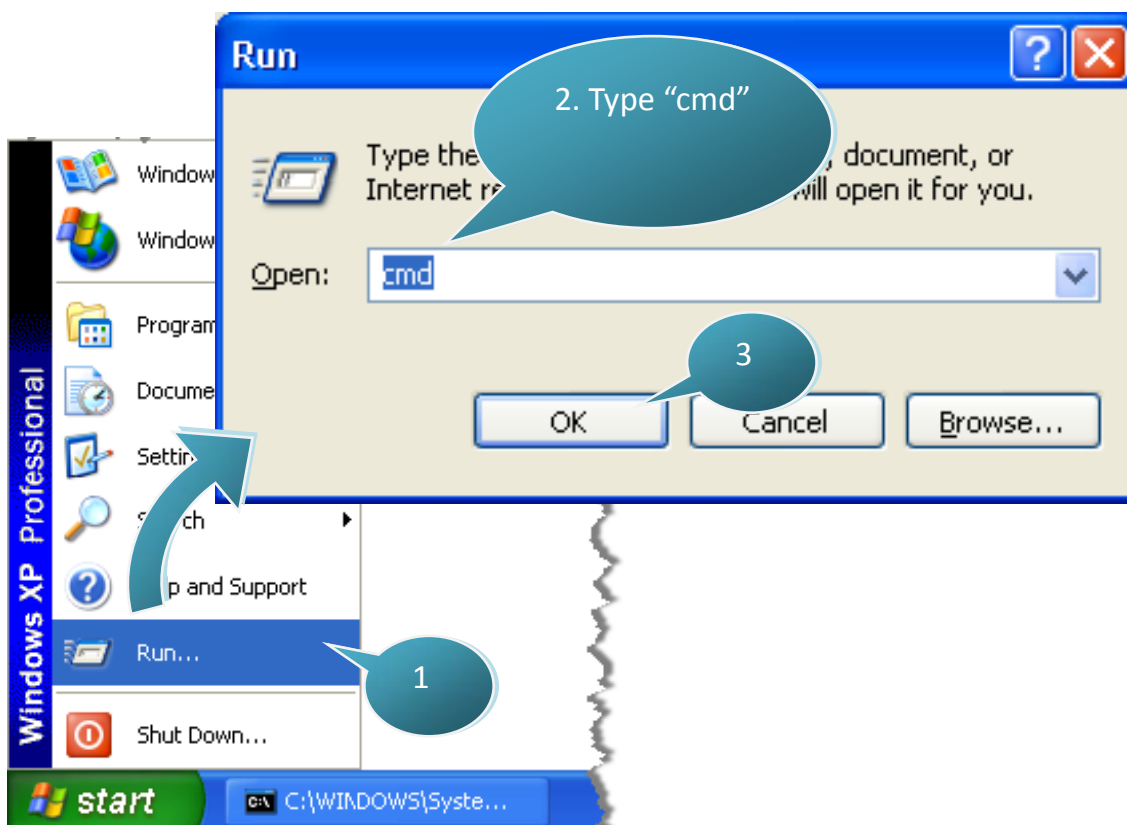
3.3. First Program in μ PAC-5000

Here we assume you have installed the Turbo C++ 1.01 (as the section “3.1. C Compiler Installation”) and the μ PAC-5000 APIs (as the section “2.3. Software Installation”) under the C driver root folder.

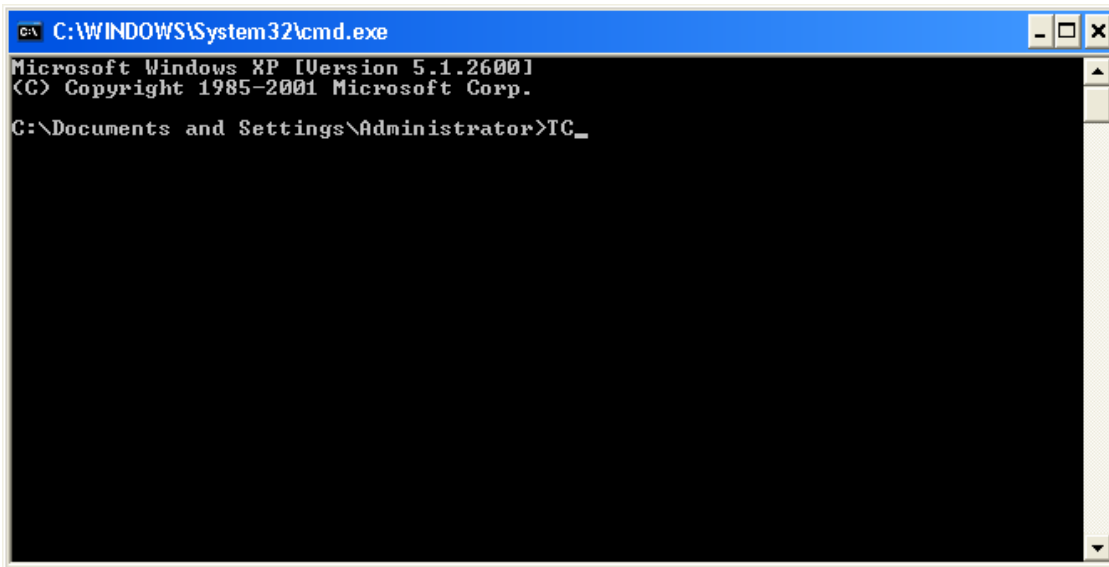
Below are step-by-step instructions for writing your first program.

Step 1: Open a MS-DOS command prompt

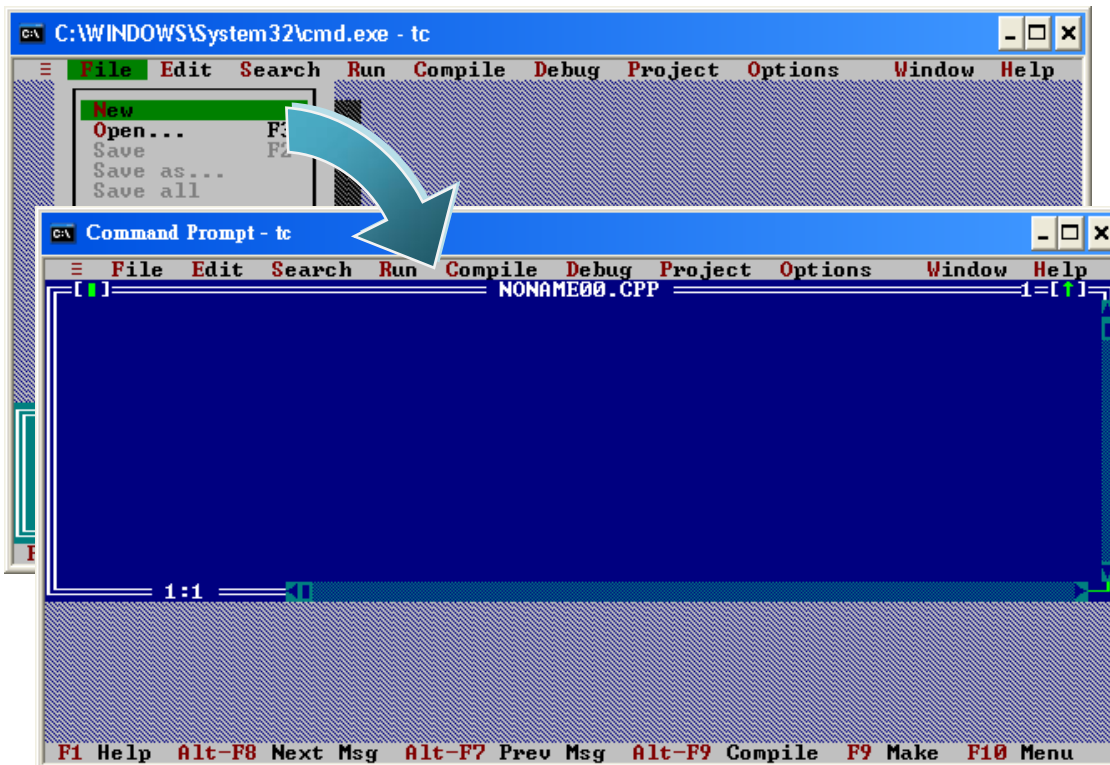
- i. Select “Run” from the “Start” menu
- ii. On the “Run” dialog box, type “cmd”
- iii. Click the “OK” button



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

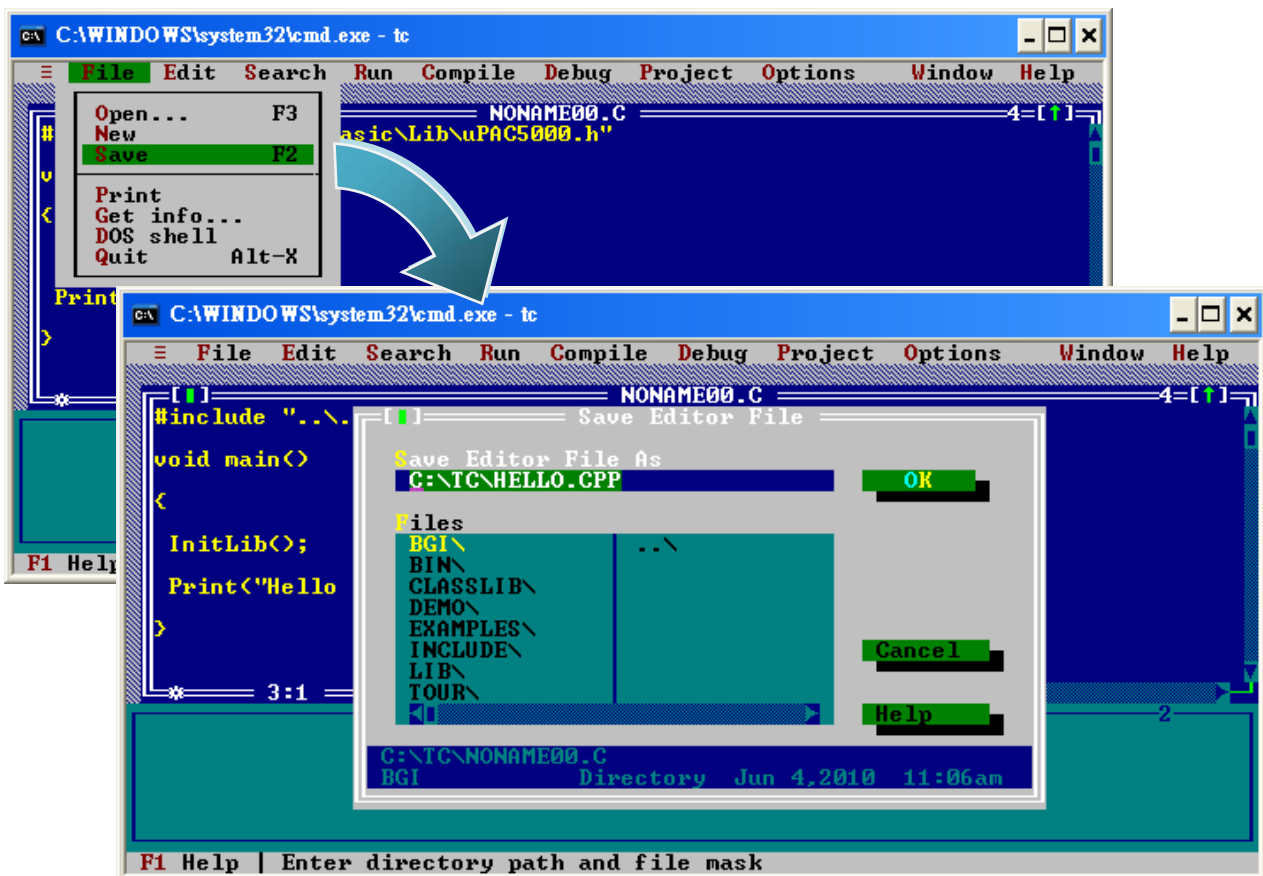
```
#include "..\..\demo\basic\lib\upac5000.h"
/* Include the header file that allows uPAC5000.lib functions to be used */

void main(void)
{
    InitLib(); /* Initiate the upac5000 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"



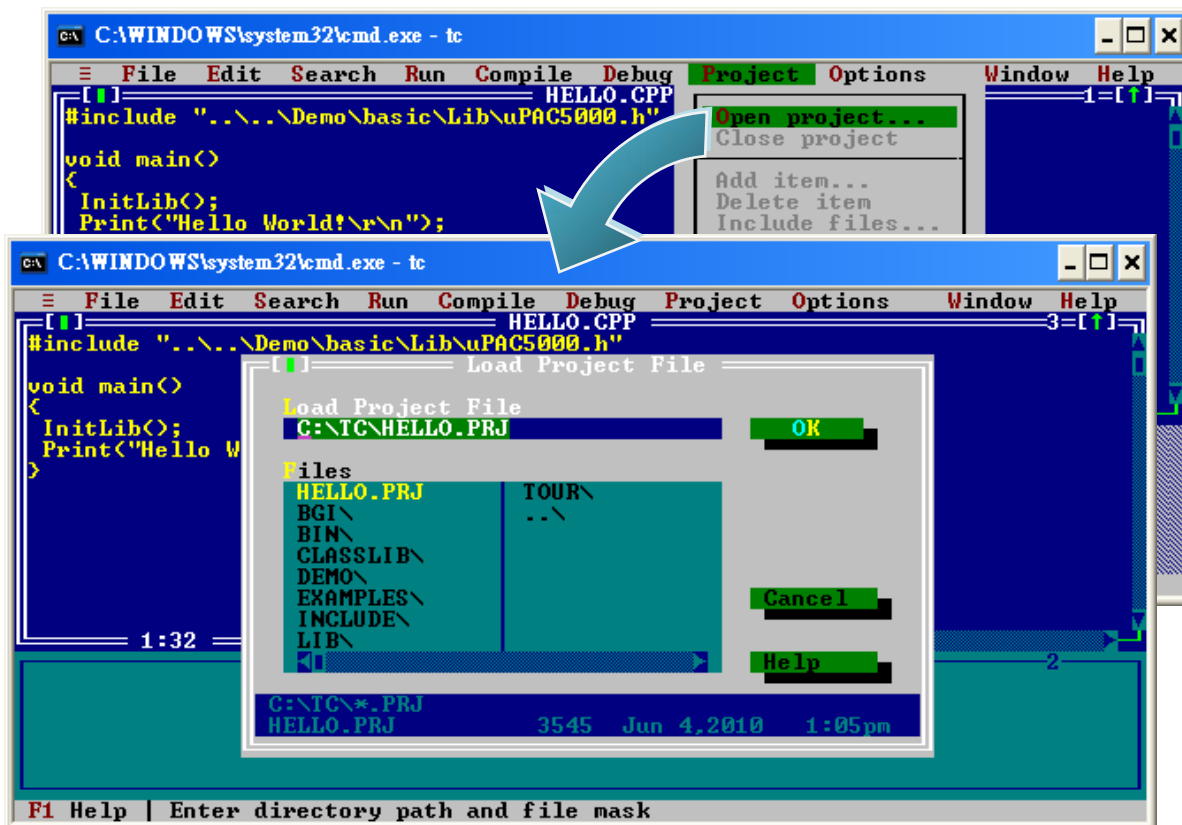
Tips & Warnings



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”.

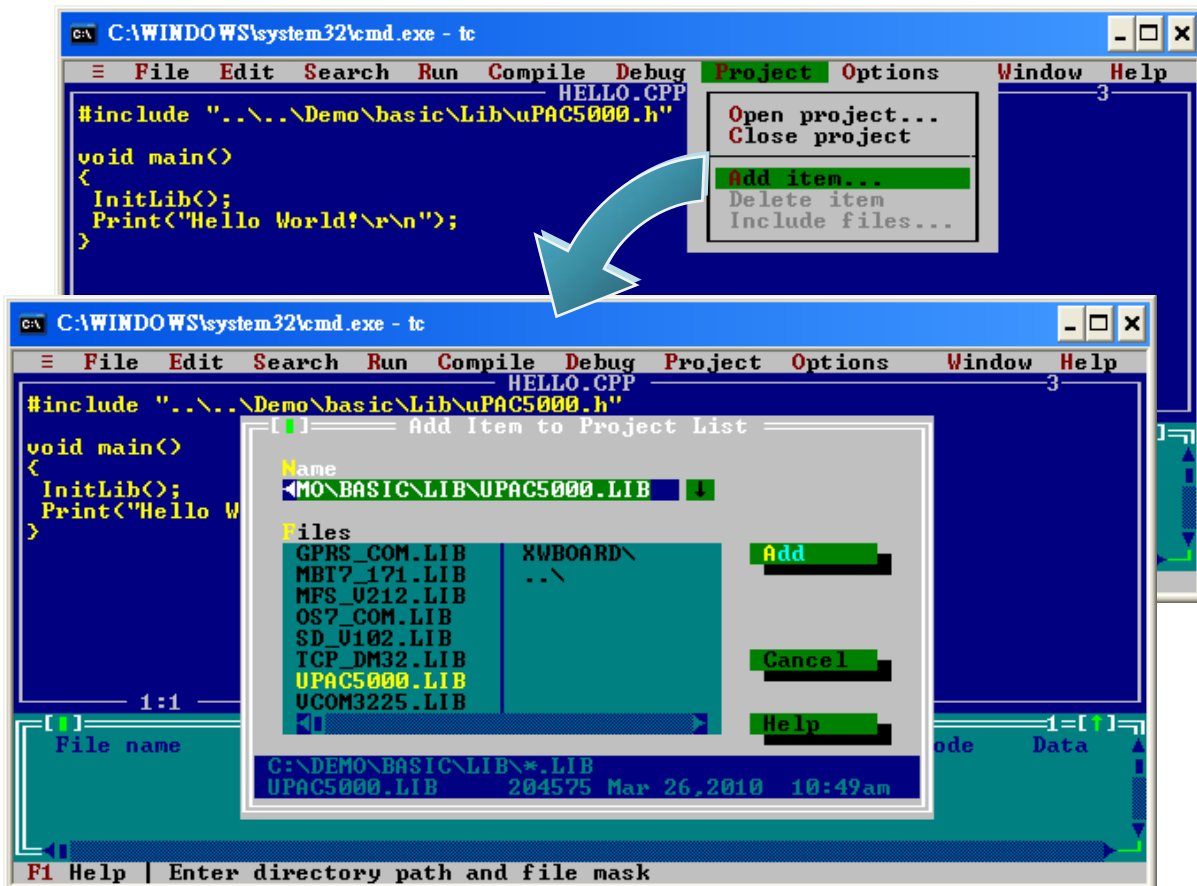
Step 6: Create a project (*.prj)

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



Step 7: Add the necessary function libraries (*.lib) to the project

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

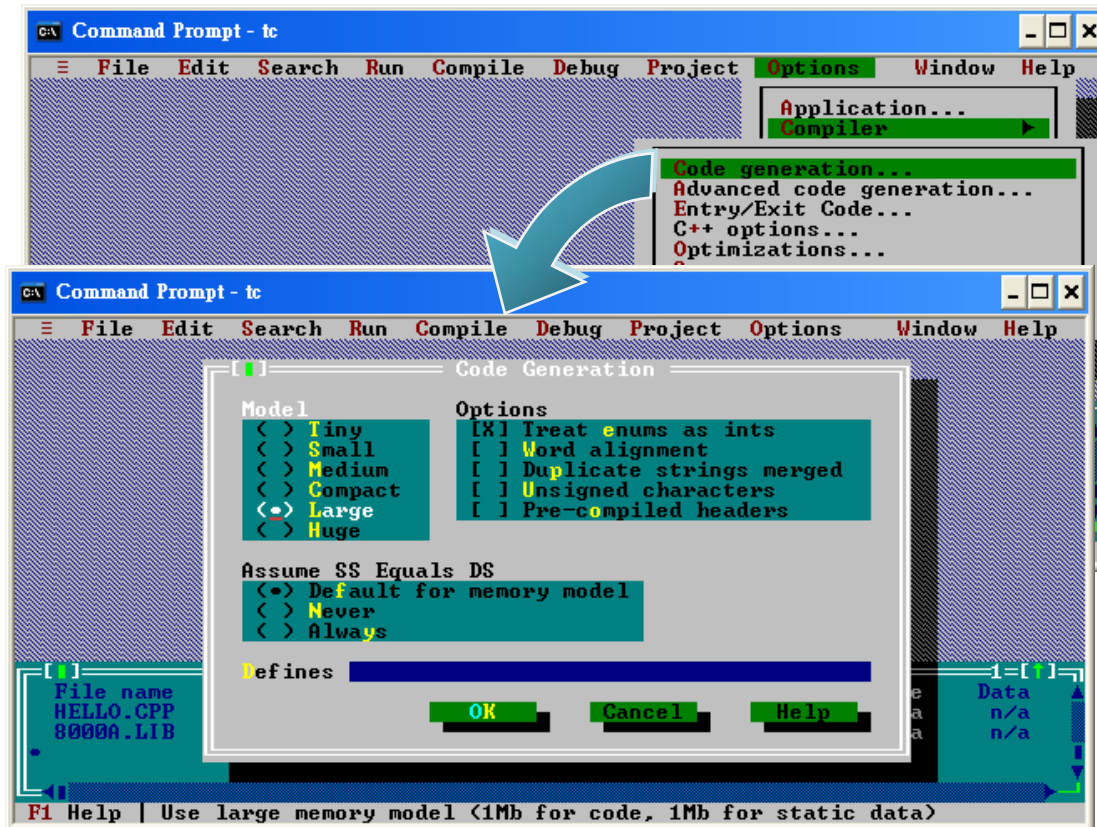


Step 8: Add the source file (*.c) to the project

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

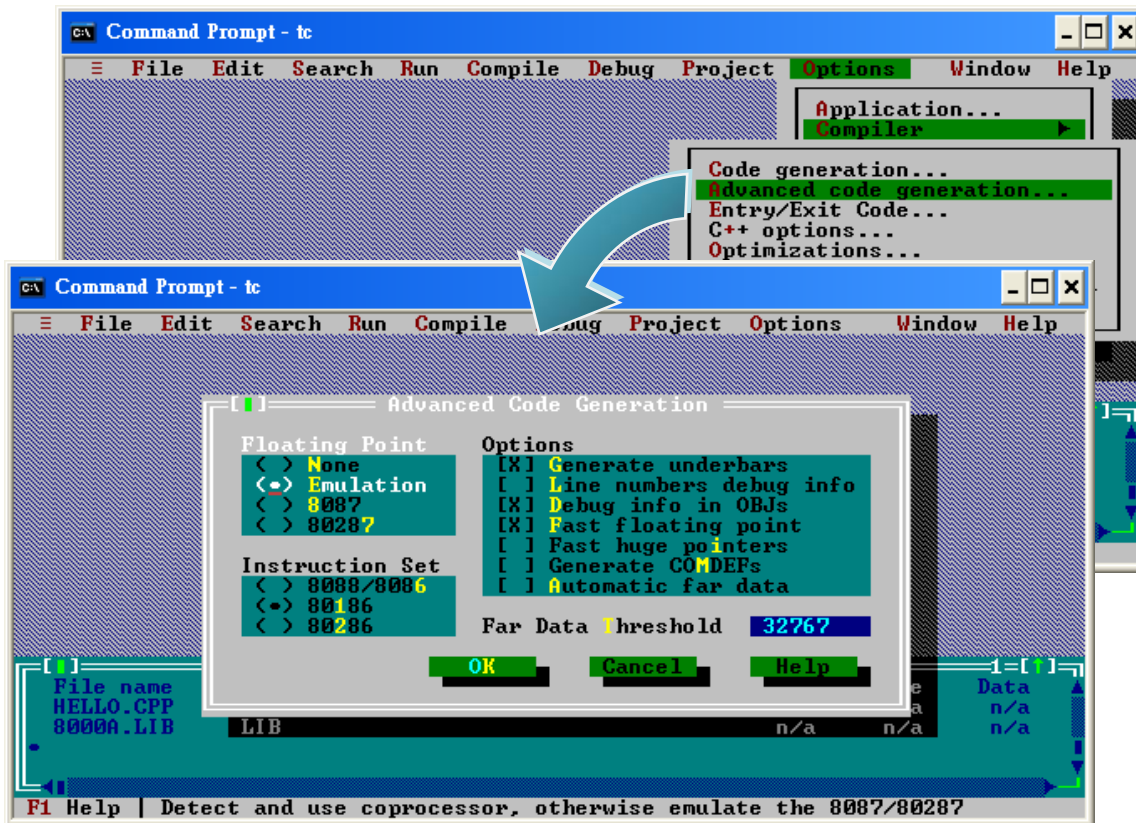
Step 9: Set the memory model to large

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



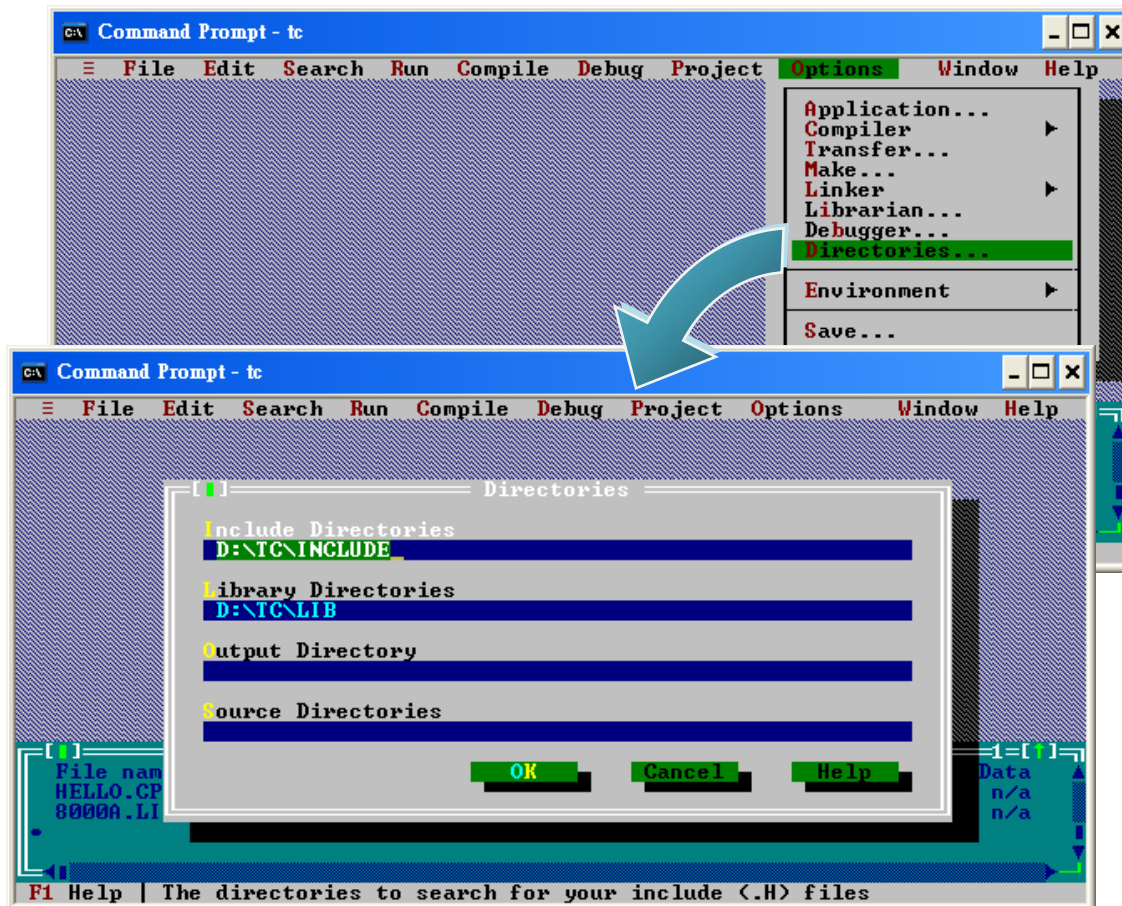
Step 10: Set the “Floating Point” to Emulation and the “Instruction Set” to 80186

- 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”

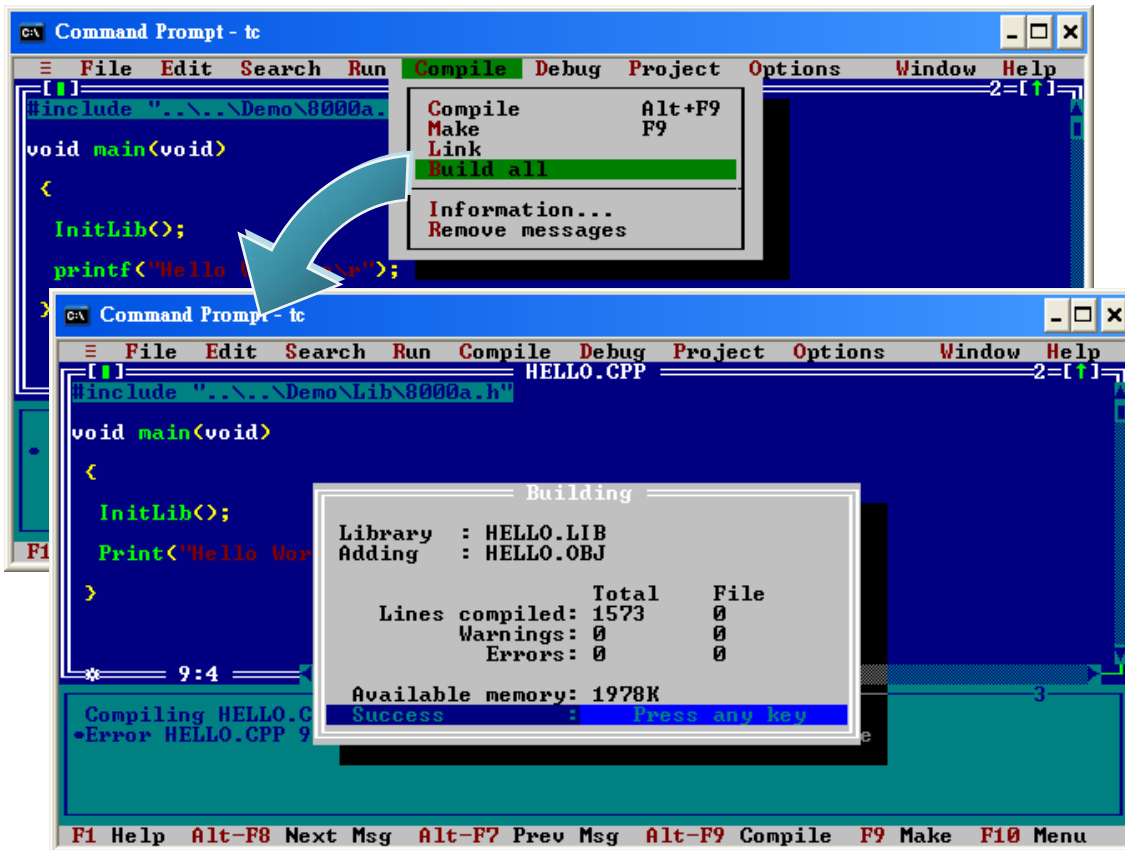


Step 11: Specify the include directories where the compiler can search for header file and libraries

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

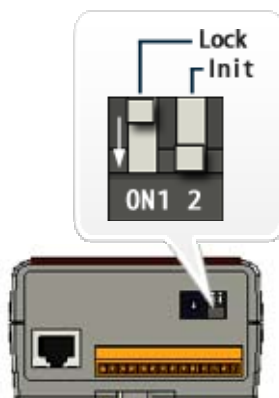


Step 12: Select "Build all" from the "Compile" menu to build the project

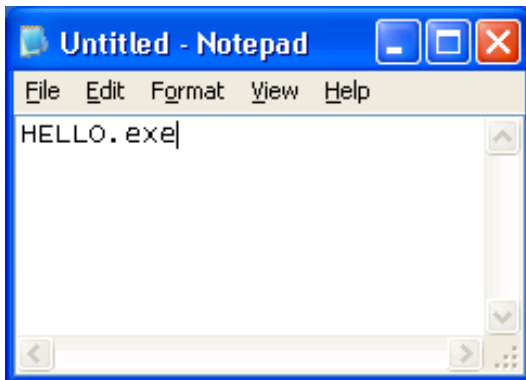


Step 13: Configure the operating mode

Make sure the switch of the Lock placed is in the "OFF" position, and the switch of the Init placed is in the "ON" position.



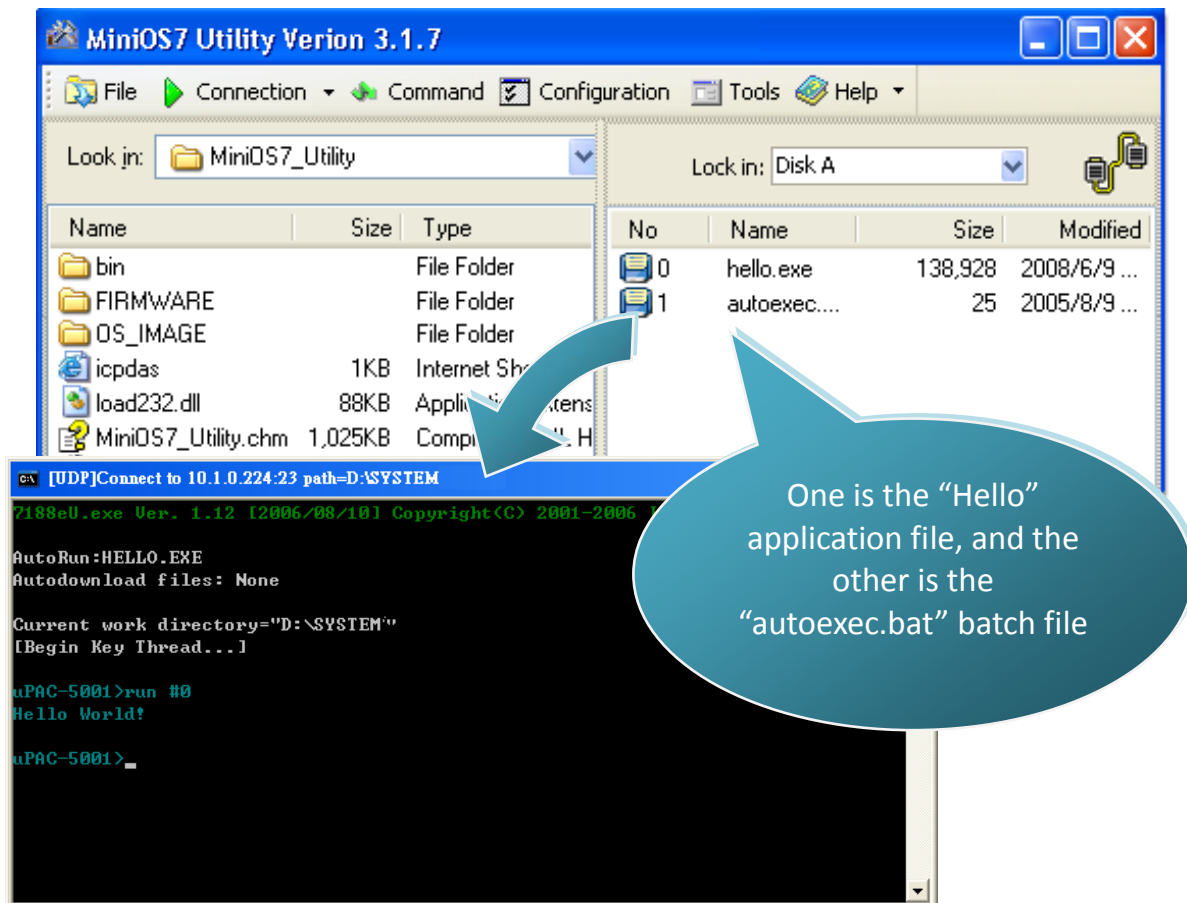
Step 14: Create an autoexec.bat file



- i. Open the "Notepad"
- ii. Type the "HELLO.exe"
- iii. Save the file as autoexec.bat

Step 15: Upload programs to μ PAC-5000 using MiniOS7 Utility

For more detailed information about this process, please refer to section "2.5.1. Establishing a connection between PC and μ PAC-5000"



4. APIs and Demo References

There are several APIs and demo programs that have been designed for μ PAC-5000.

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	uPAC5000.h	uPAC5000.lib
New version of COM port driver	OS7_COM.h	OS7_COM.lib
Ethernet driver	TCPIP32.h	Tcp_dm32.Lib
microSD driver	microSD.h	sd_V102.lib
256 MB Flash Disk driver	MFS.h	MFS@V212.lib
GSM driver	GSM.h	GSM.lib
Xserver	VxComm.h	Vcom3225.Lib
Modbus driver	MBTCP.h	MBT7_171.lib
Xboard driver	XW107.h	XW107.lib

For more detailed information regarding μ PAC-5000 APIs, please refer to

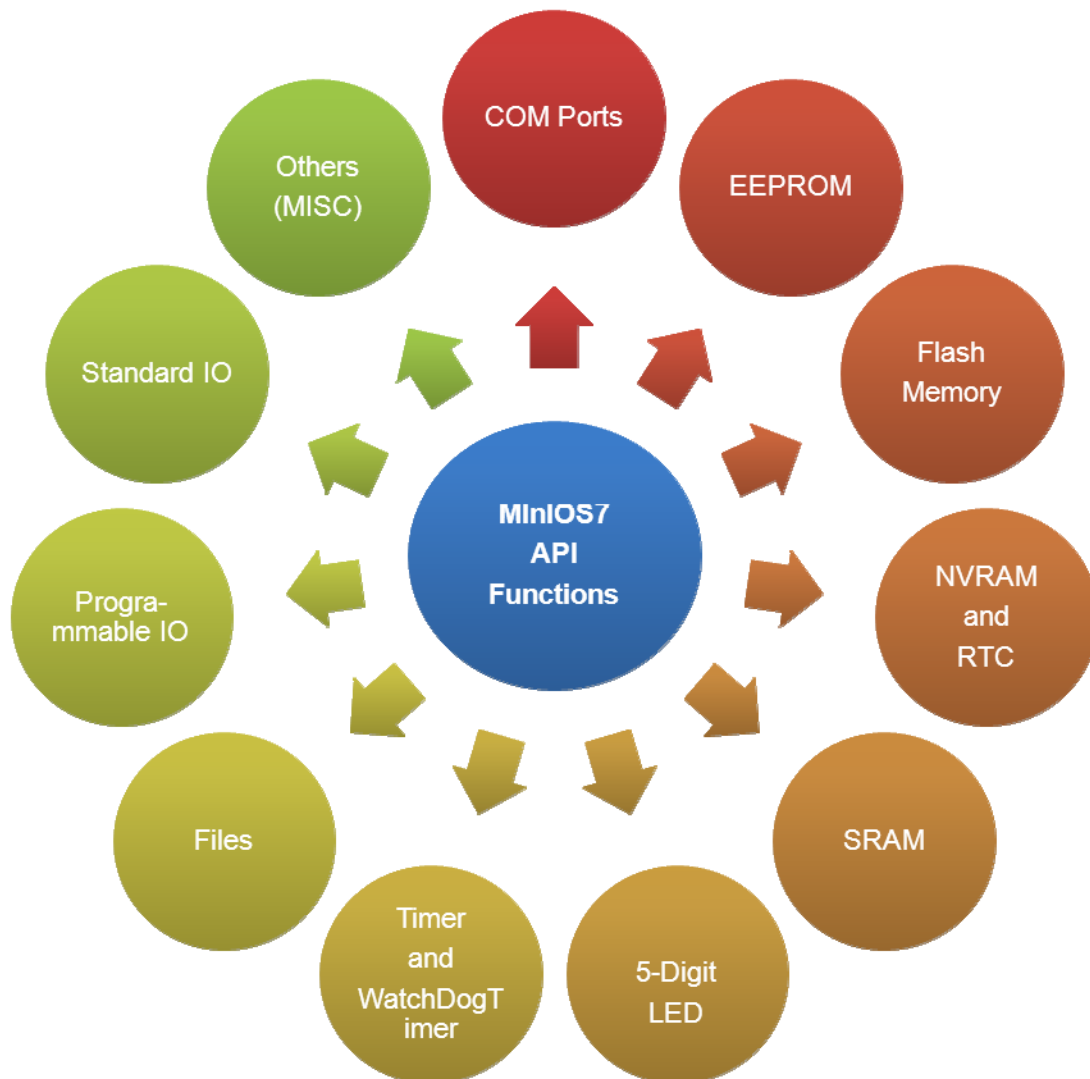
CD:\NAPDOS\upac-5000\Demo\basic\Lib\01_Lib_Update_History_20100507.txt

http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/lib/01_Lib_Update_History_20100507.txt

The following introduces the core API, MiniOS7 API, which is integrated into the uPAC5000 API set.
Functions Library — uPAC5000.lib
This file contains the MiniOS7 API (Application Programming Interface) and has hundreds of pre-defined functions related to μ PAC-5000

Header File — uPAC5000.h

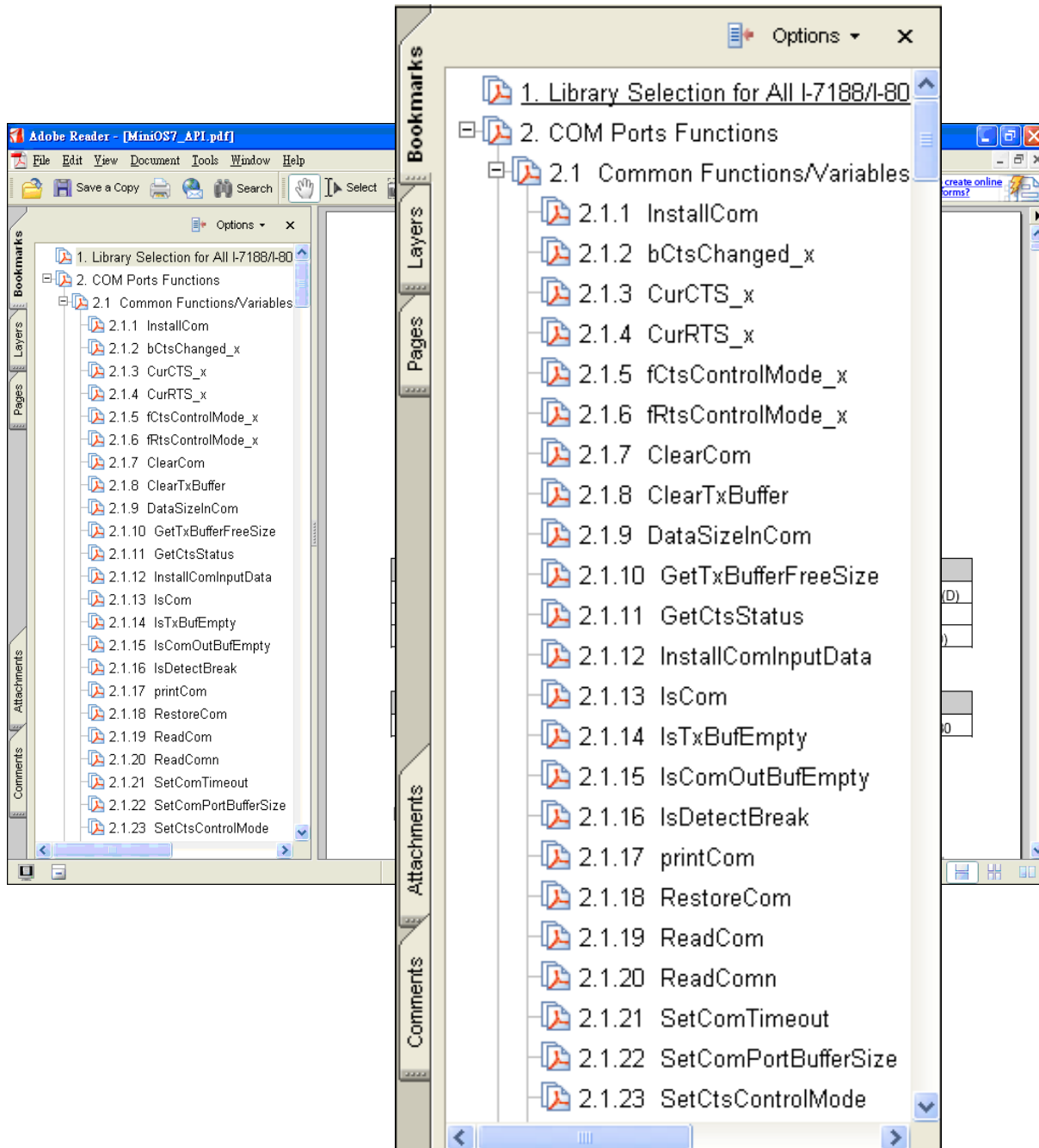
This file contains the forward declarations of subroutines, variables, and other identifiers used for the MiniOS7 API.



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.

Folder	Demo	Explanation
MISC	LED	Shows how to control the LED display.
	Rotary_Switch	Shows how to read the position of the switch.
256 MB_Flash	Gets	Shows how to get a string from a file in the 256MB flash memory
	mFS_QA	Quality assurance program for the MiniOS7 File System. Including function test, read/write performance test.
	Puts	Shows how to write a string to a file in the 256MB flash memory.
	Utility	Utility for the MiniOS7 File System. Operations Include Dir, Read, Write, etc.
microSD	sd_qa	Shows how to connect and control the microSD
	sd_read	
	sd_util	
	sd_write	
modbus	MTDemo00	Shows how to use a standard Modbus driver to retrieve data.
	MTDemo01_Link_i7000	
	MTDemo03_Link_PLC	
	MTDemo05_Modbus_TCP_Master_1	
	MTDemo05_Modbus_TCP_Master_2	
XWboard	xw107	Shows how to connect and control the XW107 board.

For more detailed information regarding μ PAC-5000 APIs, please refer to

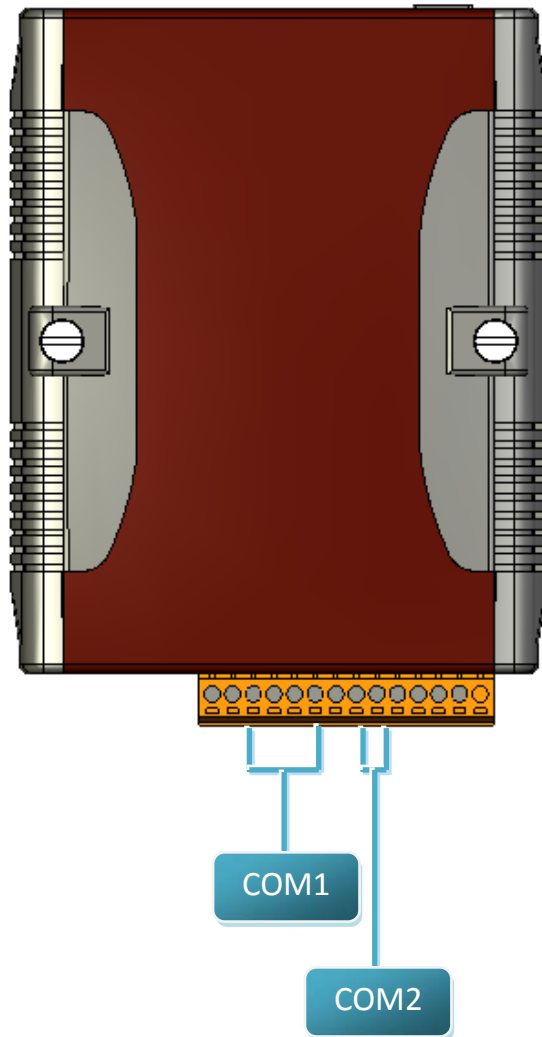
CD:\NAPDOS\upac-5000\Demo\basic\

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/>

4.1. API for COM Port

The μ PAC-5000 provides two built-in COM ports, COM1 and COM2.

- COM1 – A RS-232 port can use to connect to PC.
- COM2 – A RS-485 port in a point to point connection.



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

The software driver for the uPAC-5000 is an interrupt driven library that provides a 1K QUEUE buffer for each COM port. The software is well designed and easy to use. The MiniOS7 provides the same interface for all COM ports, so each port can be used in the same way without any difficulty.

API for using COM ports

1. InstallCom()

Before using the COM port, the COM port driver must be installed by calling InstallCom().

2. RestoreCom()

If the program calls InstallCom(), the RestoreCom() must be called to uninstall 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 port

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 "upac5000.h"

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

    InitLib(); /* Initiate the upac5000 library */
    InstallCom(1, 115200L, 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); /* Release the COM1 */
}
```

API for showing data from COM ports

6. printCom()

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

For example, showing data from the COM1 port.

```
#include <stdio.h>
#include "upac5000.h"

void main(void)
{
    int i;

    InitLib(); /* Initiate the upac5000 library */

    InstallCom(1, 115200L, 8, 0, 1); /* Install the COM1 driver */
    for(i=0; i<10; i++)
    {
        printCom(1, "Test %d\r\n", 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 μ PAC-5000.

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:

Baud rate = 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. Kbhith()

Before reading data from standard I/O port, the Kbhith() 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 Kbhith() 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

Outputs a string to the COM1 (and appends a newline character).

4. Putch() – For sending one character

Outputs a character to the COM1.

5. Print()

Functions such as printf() in the C library allow data to be output from the COM1.

For example, reading and receiving data through COM1.

```
#include <stdio.h>
#include "upac5000.h"

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

    InitLib(); /* Initiate the upac5000 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 "upac5000.h"

void main(void)
{
    int i;

    InitLib(); /* Initiate the upac5000 library */

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

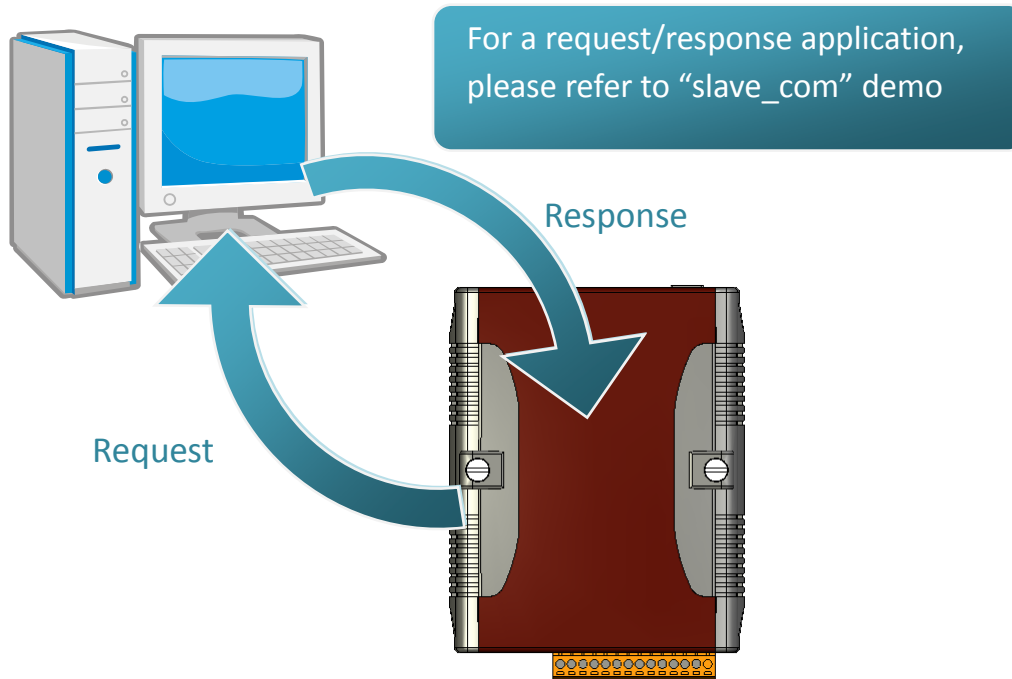
4.1.4. Port functions Comparison

For example, learning to show the ASCII code.

MiniOS7 COM port functions	Standard COM port functions
<pre>#include <stdio.h> #include "upac5000.h" void main(void) { unsigned char item; InitLib(); InstallCom(1,115200,8,0,1); printCom(1,"Press any key.\n"); printCom(1,"Press the ESC to exit!\n"); for(;;){ if(IsCom(1)){ item=ReadCom(1); if(item=='q') return; else{ printCom(1,"-----\r\n"); printCom(1,"char: "); ToCom(1,item); printCom(1,"\r\nASCII(%c)\r\n",item); printCom(1,"Hex(%02X)\r\n",item); } } } Delay(10); RestoreCom(1); }</pre>	<pre>#include <stdio.h> #include "upac5000.h" void main(void) { unsigned char item; InitLib(); Print("Press any key.\n"); Print("Press the ESC to exit!\n"); for(;;){ if(Kbhit()){ item=Getch(); if(item=='q') return; else{ Print("-----\r\n"); Print("char: "); Putch(item); Print("\r\nASCII(%c)\r\n",item); Print("Hex(%02X)\r\n",item); } } } }</pre>

4.1.5. Request/Response protocol define on COM port

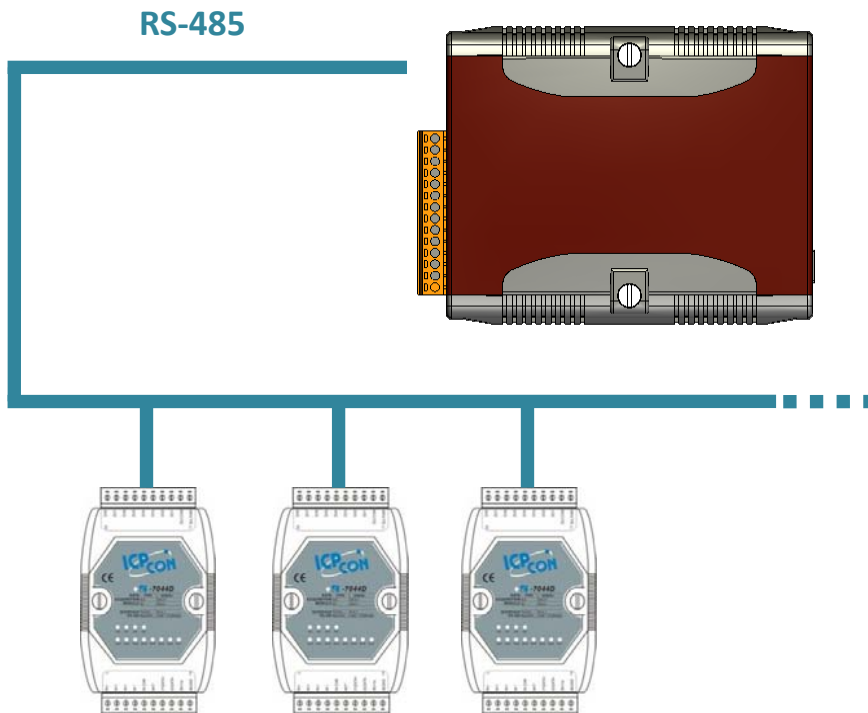
Request/Response communication is very typical protocol architecture. If you want to design a command set of communication protocol as table below, you can refer to "slave_com" demo.



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

4.2. API for I/O Modules

The μ PAC-5000 equip a RS-485 communication interface, COM2, to access I-7000 series I/O modules for a wide range of RS-485 network application, as shown below.



Steps to communicate with i-7000 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 uninstall the COM port driver

For example, to send a command '\$01M' to I-7000 I/O module for getting the module name.

```
#include <stdio.h>
#include "upac5000.h"

void main(void)
{
    unsigned char InBuf0[60];

    InitLib(); /* Initiate the upac5000 library */

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

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

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

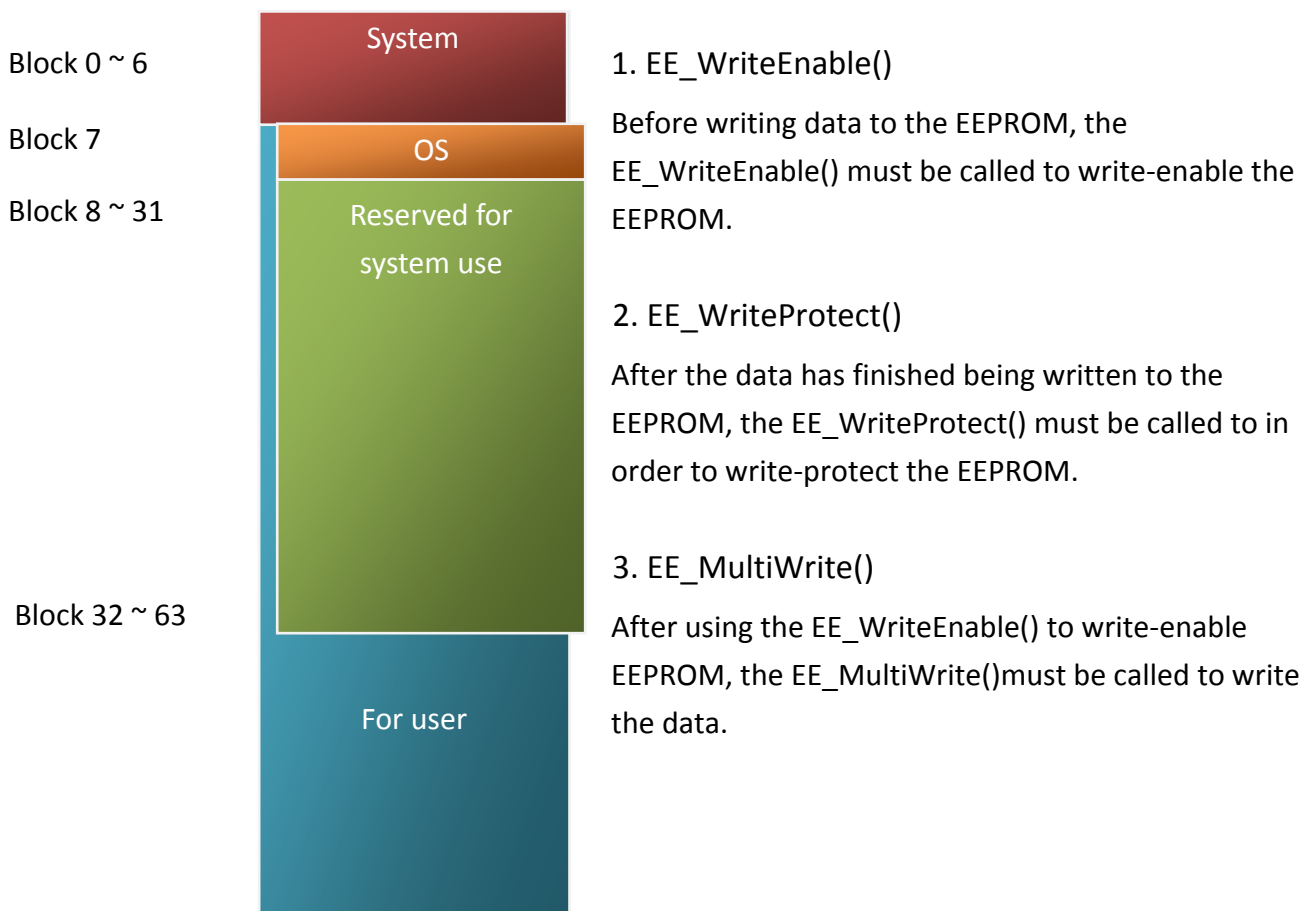
    printCom(1, "Module Name = %s", InBuf0);
    Delay(10); /* Wait for all data are transmitted to COM port */

    RestoreCom(1); /* Release the COM1 */
    RestoreCom(2); /* Release the COM2 */
}
```

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



API for reading data from the EEPROM

4. EE_MultiRead()

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 "upac5000.h"

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

    InitLib(); /* Initiate the upac5000 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 μ PAC-5000 module contains 512K bytes 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.



API for erasing data from the Flash Memory

1. EraseFlash()

The only way to change the data from 0 to 1 is to call the EraseFlash() function to erase a block from the Flash memory.

2. FlashWrite()

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

API for reading data from the Flash Memory

3. 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 "upac5000.h"

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

    InitLib(); /* Initiate the upac5000 library */

    EraseFlash(0xd000); /* Erase a block from the Flash memory */
    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;
}
```

4.5. API for NVRAM

- The μ PAC-5000 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 "upac5000.h"

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

    InitLib(); /* Initiate the upac5000 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 "upac5000.h"

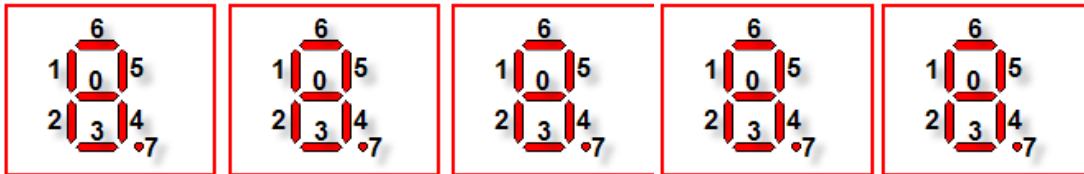
void main(void)
{
    int data=0xAA55, data2;
    char *dataptr=(char *) &data;

    InitLib(); /* Initiate the upac5000 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 μ PAC-5000 contains a 5-Digit 7-SEG LED with a decimal point on the right-hand side of each digit, which can 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-Digit 7-SEG LED.

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

```
#include <stdio.h>
#include "upac5000.h"

void main(void)
{
    InitLib(); /* Initiate the upac5000 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 μ PAC-5000 can support a single main time tick, 8 stop watch timers and 8 counts down timers.
- The μ PAC-5000 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 zero.

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 "upac5000.h"

void main(void)
{
    Unsigned long time iTime;

    InitLib(); /* Initiate the upac5000 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 uPAC5000 timer function */
}
```

4.8. API for WatchDog Timer (WDT)

- The μ PAC-5000 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 "upac5000.h"

void main(void)
{
    Unsigned long time iTime;

    InitLib(); /* Initiate the upac5000 library */

    Enable WDT();
    While(!quit) {
        RefreshWDT();
        User_function();
    }
    DisableWDT();
}
```

4.9. API for MFS (For μ PAC-5000-FD series only)



Required library and header files:

MFS@Vnnn.LIB and MFS.h

The μ PAC-5000-FD series products equip an extra 256MB flash memory, the MFS is designed to read/write file from/to the 256MB 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_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. 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.

Function	Description
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.

API for starting 256MB flash memory

1. mfs_Init()

Before using any MFS functions, the mfs_Init() must be called to initialize the 256MB flash memory.

2. mfs_Stop()

If the program calls the mfs_Init() to initialize the 256MB flash memory, the mfs_Stop() must be called to allocate buffers to free upon closing.

API for writing/reading files from the 256MB flash memory

3. mfs_OpenFile()

Before writing/reading data to/from the 256MB 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 256MB flash memory, the mfs_CloseFile() must be called to close the file with a file handle.

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 256MB flash memory:

```
#include <stdio.h>
#include "upac5000.h"
#include "MFS.h"

#define _DISK_A 0
#define _DISK_B 1

void main(void)
{
    int iFileHandle, iRet;

    InitLib(); /* Initiate the upac5000 library */

    iRet=mfs_Init();
    if(iRet!=256) return;

    iFileHandle=mfs_OpenFile(_DISK_A, "Test.txt", "w");
    if(iFileHandle>0)
    {
        Print("Write string to Test.txt...");
        mfs_Puts(iFileHandle, "test mfs on 256MB flash");
        mfs_CloseFile(iFileHandle);
        Print("done");
    }
    else
        Print("Open file error\r\n");

    mfs_Stop();
}
```

6. mfs_Gets()

After using the mfs_OpenFile() to open the file, the mfs_Gets() must be called to read data from the 256MB flash memory.

For example, reading data from the 256MB flash memory:

```
#include <stdio.h>
#include "upac5000.h"
#include "MFS.h"

#define _DISK_A 0
#define _DISK_B 1

void main(void)
{
    int iFileHandle, iRet;

    InitLib(); /* Initiate the upac5000 library */

    iRet=mfs_Init();
    if(iRet!=256) return;

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

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

    mfs_Stop();
}
```

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

CD:\NAPDOS\upAC-5000\Demo\Basic\256MB_Flash\

http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/256mb_flash/

4.10. API for microSD



Required library and header files:

SD_Vnnn.LIB and microSD.h

The μ PAC-5000 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
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

Function	Description
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 microSD 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 "upac5000.h"
#include "microSD.h"

void main(void)
{
    int fd, iRet;

    InitLib();

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

    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 "upac5000.h"
#include "microSD.h"

void main(void)
{
    int fd, iRet;
    unsigned char Buffer[80];

    InitLib();

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

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\µPAC-5000\Demo\Basic\microSD\

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/microsd/>

4.11. GSM API Function



Required library and header files:

GSM.lib and GSM.h

OS7_COM.lib and OS7_COM.h

2G/3G is a service that allows information to be sent and received across a mobile telephone network. It supports CSD (Circuit Switched Data), SMS (Short Message Service) and GPRS (General Packet Radio Service). ICP DAS provides the 2G/3G library for GSM series module. The library is an easy way to applying the 2G/3G service in the GSM series module. Therefore, there are many application architectures to apply in the system. Or users can integrate other controller system with GSM library

Summarize of the GSM functions:

Function	Description
GM_SYS_InitModem	Initialize Modem
GM_SYS_CloseModem	Close the modem
GM_SYS_CheckModemStatus	Check modem status, and suggest you check it in your loop every time
GM_SYS_CheckCmdStatus	Get the status of the command you sent
GM_SYS_CheckSignal	Check signal quality
GM_SYS_CheckReg	Check register
GM_SYS_EnableNITZ	Enable NITZ function
GM_SYS_NITZUpdateRTC	Update the RTC of the System by NITZ
GM_SYS_CheckNITZ	Check the status of NITZ
GM_SMS_SendMsg	Send a message
GM_SMS_GetNewMsg	Get a new sms message
GM_NET_SetNet	Set Net profile data
GM_NET_InstallLink	Establish TCP/UDP connection
GM_NET_CloseNet	Close Network
GM_NET_GetIP	Get local IP address
GM_NET_CloseLink	Close the established TCP/UDP connection
GM_NET_GetLinkStatus	Get the status of established TCP/UDP connection
GM_NET_Send	Send a packet
GM_NET_GetNewPacket	Get the new packet



The OS7_COM.lib library is required for using the GSM functions.

The speed of GPRS connection is less than one packet per second.

The GSM library kernel uses the timer that installed by “InstallUserTimer()”. The specific timer should be avoided when developing program using GSM library.

System Functions

1. GM_SYS_InitModem()

Before using any GSM functions, the GM_SYSInitModem() must be called to initialize the GSM modem.

2. GM_SYS_CloseModem()

If the program calls the GM_SYSInitModem() to initialize the GSM modem, the GM_SYS_CloseModem() must be called to shut down the modem before exiting program.

3. GM_SYS_CheckModemStatus()

This function can be used to retrieve the current status of GSM modem. Since the modem status can affect sending/receiving, this function should be called by your main program loop periodically, and also whenever data transmission is required.

API functions for sending/receiving SMS over GSM

4. GM_SMS_SendMsg()

Send an SMS message to the recipient phone. You need to specify the receiver phone number and the message content.

5. GM_SMS_GetNewMsg()

Receive incoming SMS message via GSM modem.

Here is an example of sending SMS messages to the recipient phone via GSM modem.

```
#include <stdlib.h>
#include <string.h>
#include "upac5000.h"
#include "OS7_COM.h"
#include "GSM.h"

void main(void)
{
    strEncode_Msg SendMsg;
    SYSProfile sysProfile;
    int iRet, sendStatus;
    char bSend, Act;

    InitLib();

    /* initialize the GSM modem */
    strcpy(sysProfile.PINCode, "0000"); /* if you are using a SIM that has a PIN, you
                                         must enter the pin or your SIM will be
                                         blocked by your mobile operator */
    sysProfile.modemPort=11; /* modem port number, uPAC-5000=11 */
    sysProfile.hardware=2; /* hardware type, uPAC-5000=2 */
    if((iRet=GM_SYS_InitModem(sysProfile))!=GM_NOERROR){
        Print("Failed to initiate the modem! Error Code %d\r\n", iRet);
        return;
    }

    /* check whether the GSM modem is ready to service */
    while(GM_SYS_CheckModemStatus()!=GM_NOERROR){
        DelayMs(1000);
    }
    Print("GSM modem registered\r\n");

    for(;;){
        if(!Kbhit()) continue;

        if((Act=Getch())=='q') break; /* press 'q' to exit the program */
        else if(Act!='s') continue; /* press 's' to send a SMS text message and any other
                                       key is ignored */

        bSend=0;

        while(!bSend){
            if(GM_SYS_CheckModemStatus()!=GM_NOERROR) continue;
            switch((sendStatus=GM_SYS_CheckCmdStatus())){
                case GM_READY:
                    /* specify the phone number that you are sending the message to */
                    strcpy(SendMsg.phoneNumber, "1234567890");
                    SendMsg.mode=GSM_7BIT;
                    strcpy(SendMsg.msg, "hello world");
                    SendMsg.dataLen=strlen(SendMsg.msg);
                    GM_SMS_SendMsg(SendMsg); /* send an ASCII message */
                    break;
            }
        }
    }
}
```

```

        case GM_NOERROR:
            Print("send success\r\n");
            bSend=1;
            break;
        case GM_BUSY:
            break;
        default:
            Print("send error, and skip this one, error code=%d\r\n", sendStatus);
            bSend=1;
            break;
    }
}
}

/* before exiting program, the modem should be shut down */
GM_SYS_CloseModem(0); /* close the modem, 0: not turn off the modem, 1: turn off
                        modem */
}

```

Here is an example of receiving incoming SMS messages via GSM modem.

```

#include <stdlib.h>
#include <string.h>
#include "upac5000.h"
#include "OS7_COM.h"
#include "GSM.h"

void main(void)
{
    strEncode_Msg RecMsg;
    SYSProfile sysProfile;
    int i, iRet;

    InitLib();

    /* initialize the GSM modem */
    strcpy(sysProfile.PINCode, "0000"); /* if you are using a SIM that has a PIN, you
                                        must enter the pin or your SIM will be
                                        blocked by your mobile operator */
    sysProfile.modemPort=11; /* modem port number, uPAC-5000=11 */
    sysProfile.hardware=2; /* hardware type, uPAC-5000=2 */
    if((iRet=GM_SYS_InitModem(sysProfile))!=GM_NOERROR){
        Print("Failed to initiate the modem! Error Code %d\r\n", iRet);
        return;
    }

    /* check whether the GSM modem is ready to service */
    while(GM_SYS_CheckModemStatus()!=GM_NOERROR){
        DelayMs(1000);
    }
    Print("GSM modem registered\r\n");
}

```



```

for(;;){
    if(!Kbhit() && Getch()=='q') break; /* press 'q' to exit the program */

    if(GM_SYS_CheckModemStatus()!=GM_NOERROR) continue;

    if(GM_SMS_GetNewMsg(&RecMsg)!=0){
        Print("\r\n-----\r\n");
        Print("phone number = %s\r\n", RecMsg.phoneNumber);
        Print("receive time = %s\r\n", RecMsg.time);
        Print("mode = %d\r\n", RecMsg.mode);
        Print("message length = %d\r\n", RecMsg.dataLen);
        if(RecMsg.mode==0){
            Print("message = %s\r\n", RecMsg.msg);
        }
        else{
            Print("message = \r\n");
            for(i=0; i<RecMsg.dataLen; i++){
                Print("%02X ", (unsigned char) RecMsg.msg[i]);
            }
        }
        Print("\r\n-----\r\n");
    }
}

/* before exiting program, the modem should be shut down */
GM_SYS_CloseModem(0); /* close the modem, 0: not turn off the modem, 1: turn off
                        modem */
}

```

The GSM series module comes with a full TCP/IP architecture to make it easy to connect the cellular network. The GSM series supports the standard connection mechanisms (TCP and UDP). Whichever mechanism is used, once a connection is established, the two sides of the connection can both send and receive data.

API functions for sending/receiving data through TCP/UDP connection

6. GM_NET_SetNet()

Initialize all network interface. Note that all interfaces only can be initialized all at once.

7. GM_NET_InstallLink()

This function performs an active open, allowing a client program to establish a connection with a remote server. The serverIP and serverPort parameters are used to specify the IP address and port number for the remote end of the connection.

8. GM_NET_GetLinkStatus()

Retrieve the status of established TCP/UDP connection.

9. GM_NET_GetIP()

Retrieve local IP address from the cellular network.

10. GM_NET_Send()

Send data packets to the recipient device using the established TCP/UDP connection.

11. GM_NET_GetNewPacket()

Receive incoming data packets from the remote device using the established TCP/UDP connection.

Here is an example of sending/receiving data packets using TCP connection.

```
#include <stdlib.h>
#include <string.h>
#include "upac5000.h"
#include "OS7_COM.h"
#include "GSM.h"

void main(void)
{
    NetProfile netProfile;
    SYSProfile sysProfile;
    GPRSData gprsData;
    int i, iRet, no=0;
    int netSendStatus;
    char myIP[16];

    InitLib();

    /* initialize the GSM modem */
    strcpy(sysProfile.PINCode, "0000"); /* if you are using a SIM that has a PIN, you
                                        must enter the pin or your SIM will be
                                        blocked by your mobile operator */

    sysProfile.modemPort=11; /* modem port number, uPAC-5000=11 */
    sysProfile.hardware=2; /* hardware type, uPAC-5000=2 */
    if((iRet=GM_SYS_InitModem(sysProfile))!=GM_NOERROR){
        Print("Failed to initiate the modem! Error Code %d\r\n", iRet);
        return;
    }

    /* check whether the GSM modem is ready to service */
    while(GM_SYS_CheckModemStatus()!=GM_NOERROR){
        DelayMs(1000);
    }
}
```

```

}
Print("GSM modem registered\r\n");

/* configure the network settings */
strcpy(netProfile.APN, "INTERNET"); /* APN for network provided by your cellular
                                     provider */
strcpy(netProfile.pw, ""); /* username for network provided by your cellular
                             provider */
strcpy(netProfile.user, ""); /* password for network provided by your cellular
                              provider */
strcpy(netProfile.DnsServerIP, ""); /* if NULL is used as the default parameter
                                     value */
GM_NET_SetNet(netProfile);

/*specify the IP and the port number of remote server that you are connecting to*/
GM_NET_InstallLink(0, 1, "192.168.255.2", 10000);

for(;;){
    if(!Kbhit() && Getch()=='q') break; /* press 'q' to exit the program */

    if(GM_SYS_CheckModemStatus()!=GM_NOERROR) continue;

    if(GM_NET_GetLinkStatus(0)!=1) continue;
    else GM_NET_GetIP(myIP);

    switch((netSendStatus=GM_SYS_CheckCmdStatus())){
        case GM_READY:
            /* send data to the remote server */
            gprsData.link=0;
            sprintf(gprsData.data, "hello world");
            gprsData.dataLen=strlen(gprsData.data);
            if(GM_NET_Send(gprsData.link,gprsData.data,gprsData.dataLen)!=GM_NOERROR)
                Print("Failed to send data\r\n");
            break;
        case GM_NOERROR:
            Print("send success\r\n");
            break;
        case GM_BUSY:
            break;
        default:
            Print("Failed to send data, Error Code %d\r\n", netSendStatus);
            break;
    }

    /* receive incoming data packets via TCP/IP */
    if(GM_NET_GetNewPacket(&gprsData)!=NULL){
        no++;
        Print("----new GPRS Data (%d)----\r\n", no);
        Print("IP:Port=%s:%d, from link=%d\r\n", gprsData.IP, gprsData.port,
            gprsData.link);
        Print("Data length = %d\r\n", gprsData.dataLen);
        Print("Data=\r\n");
        for(i=0; i<gprsData.dataLen; i++)
            Putch(gprsData.data[i]);
    }
}

```

```
}  
  
GM_NET_CloseLink(0); /* close the established connection */  
GM_NET_CloseNet(); /* shut down networking */  
  
/* before exiting program, the modem should be shut down */  
GM_SYS_CloseModem(0); /* close the modem, 0: not turn off the modem, 1: turn off  
modem */  
}
```

For more demo programs about the GSM features, please refer to:

CD:\NAPDOS\uPAC-5000\Demo\Basic\GSM\

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/gsm>

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.

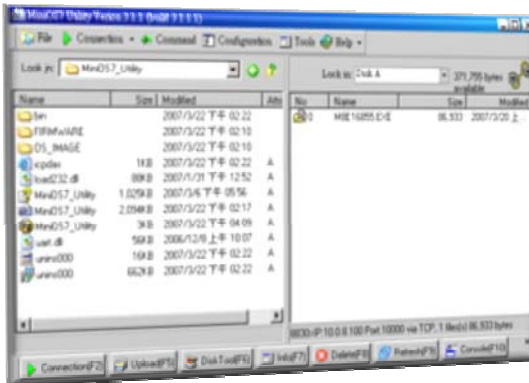


DOS (whether PC-DOS, MS-DOS or ROMDOS) is a set of commands or code that tells the computer how to process information. DOS runs programs, manages files, controls information processing, directs input and output, and performs many other related functions.

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 and 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. Send232

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 256MB NAND flash memory,

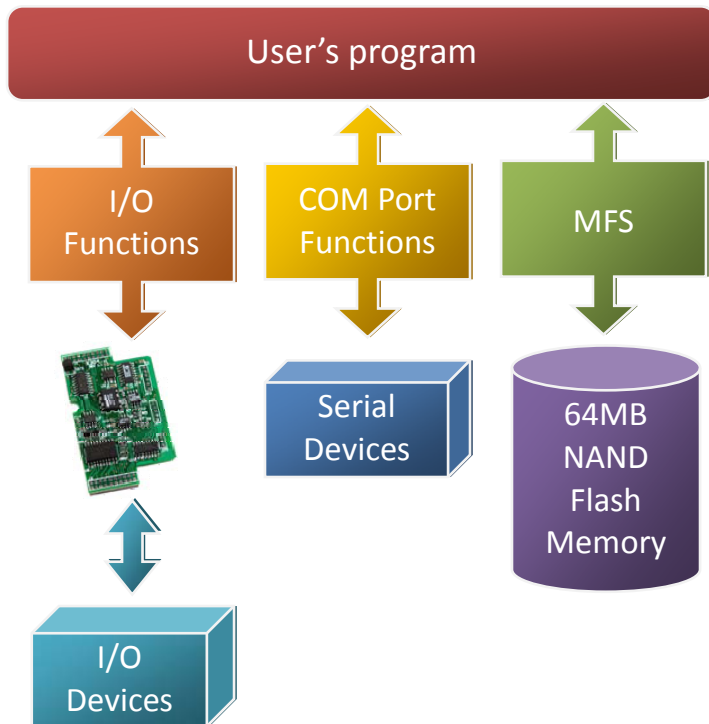
MFS implements a reliable file system with C language API for embedded data logger applications on MiniOS7.

Hardware Supported

μPAC-5000-FD (With 256MB 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	1023 files max. for each disk
File size	Disk Size max. for each file
File name	12 bytes max (case sensitive)
File operation modes	<ol style="list-style-type: none"> 1. Read only 2. Write only: Creates a new file to write data, or overwrite a file (if the file is already exist). 3. Append: appends data to a file.
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 one file handle can be used for writing operation on all disks.</p>

Writing verification	Default is enabled. Calling <code>mfs_EnableWriteVerification</code> and <code>mfs_DisableWriteVerification</code> can change the setting.
Automate file system recovery	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.
Writing speed	<code>mfs_WriteFile</code> : 147.5 KB/Sec (verification enabled) (default) 244.0 KB/Sec (verification disabled) <code>mfs_Puts</code> : 142.1 KB/Sec (verification enabled) (default) 229.5 KB/Sec (verification disabled)
Reading speed	<code>mfs_ReadFile</code> : 734.7 KB/Sec <code>mfs_Gets</code> : 414.2 KB/Sec
Max. length of writing data	32767 bytes.
Max. length of reading data	32767 bytes.

Resources upload

- MFS SDKs:
<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/lib/>
- MFS Demos:
http://ftp.lcpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/256mb_flash/

Appendix D. More C Compiler Settings

This section describes the setting of the following compilers:

- Turbo C 2.01
- Borland C++ 3.1
- MSC 6.00
- MSVC 1.50 (Prior to version 1.52)

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

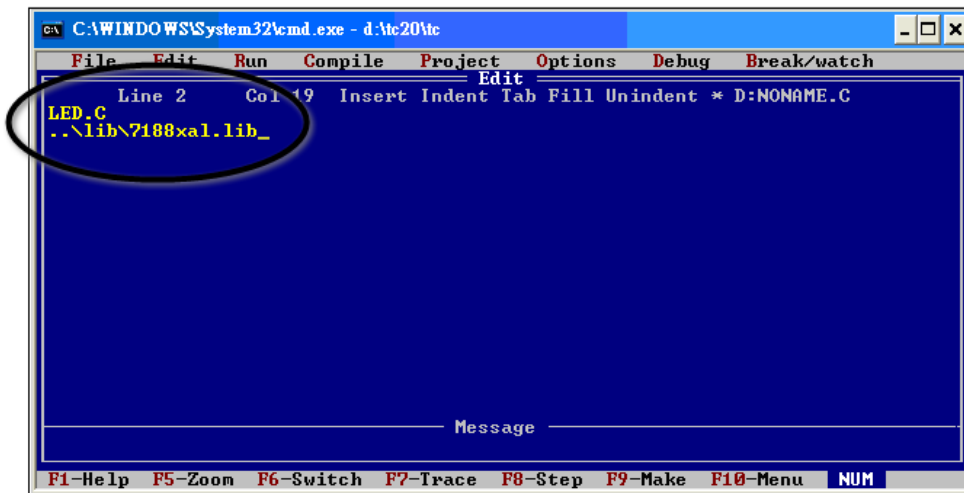
tcc -lc:\tc\include -Lc:\tc\lib hello1.c ..\..\Demo\basic\Lib\µPAC5000.lib

2: Using the TC Integrated Environment

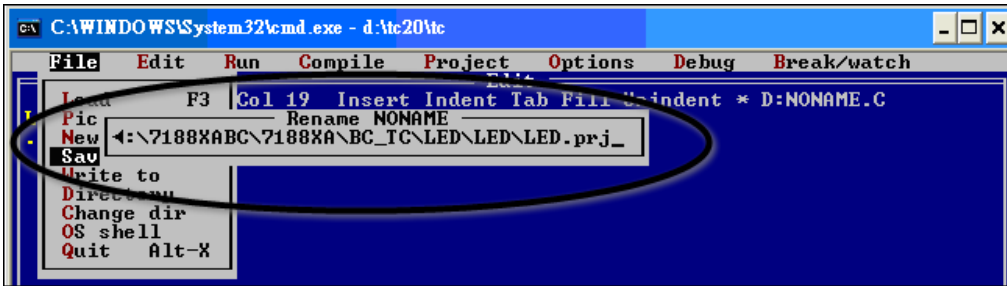
Step 1: Executing the TC 2.01

Step 2: Editing the Project file

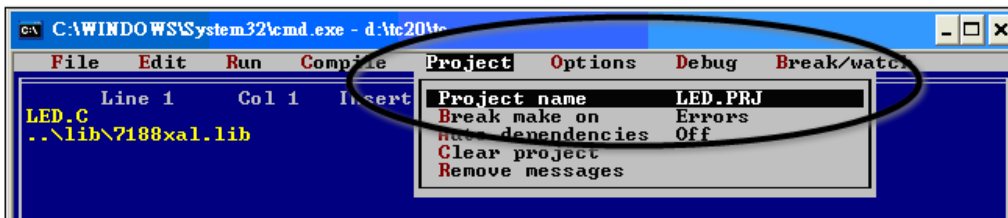
Adding the necessary library and file to the project



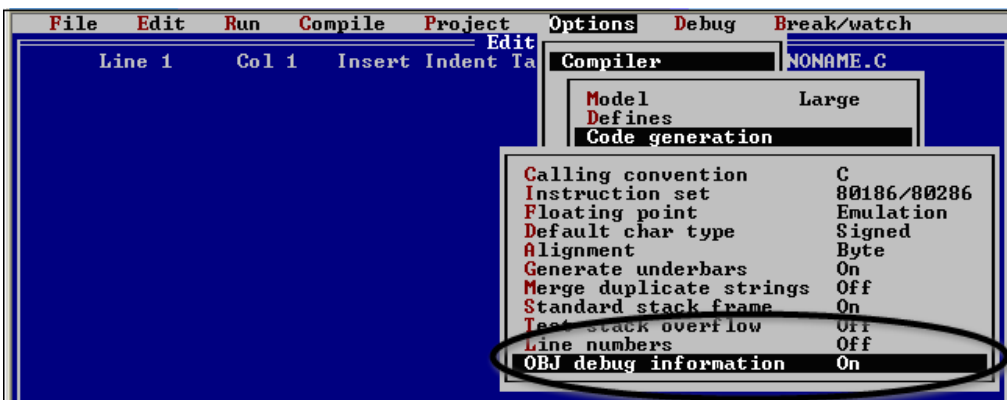
Step 3: Save the project and entering a name, such as LED.prj



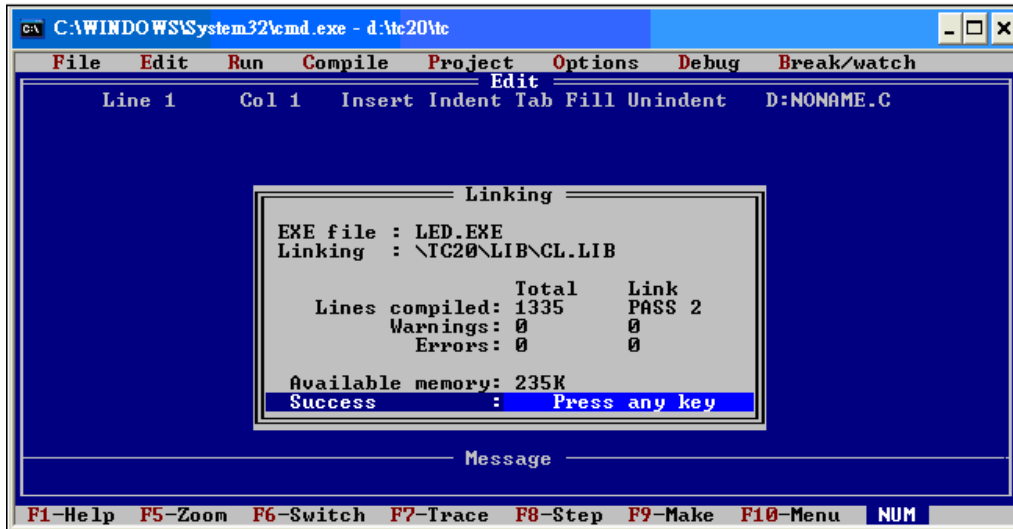
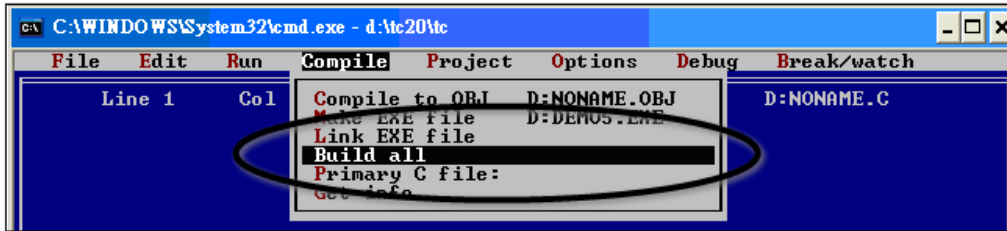
Step 4: Load the Project



Step 5: Change the Memory model (Large for uPAC5000.lib) and set the Code Generation to 80186/80286



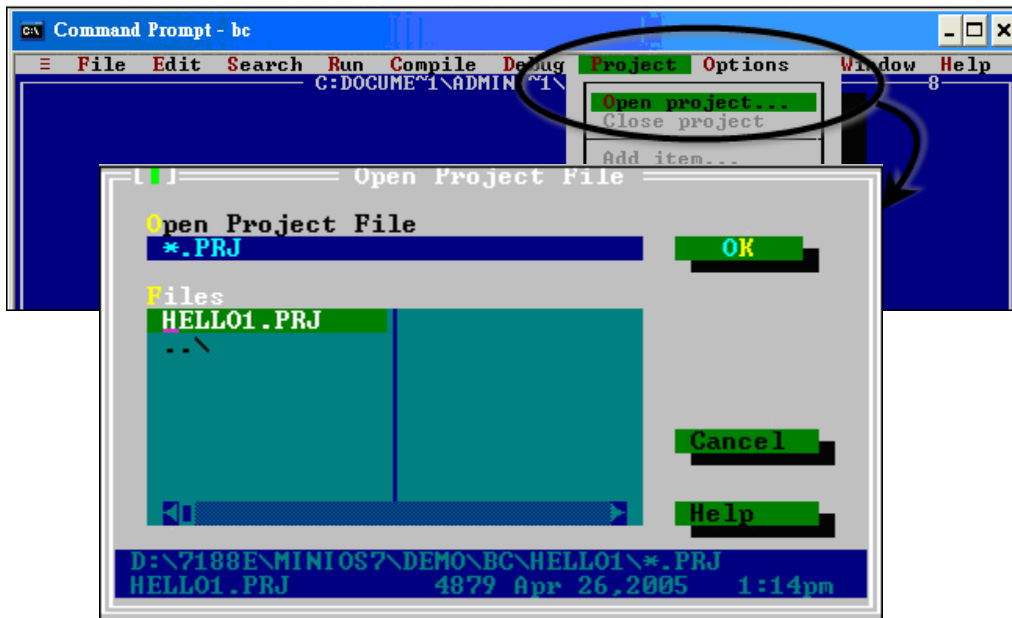
Step 6: Building the project



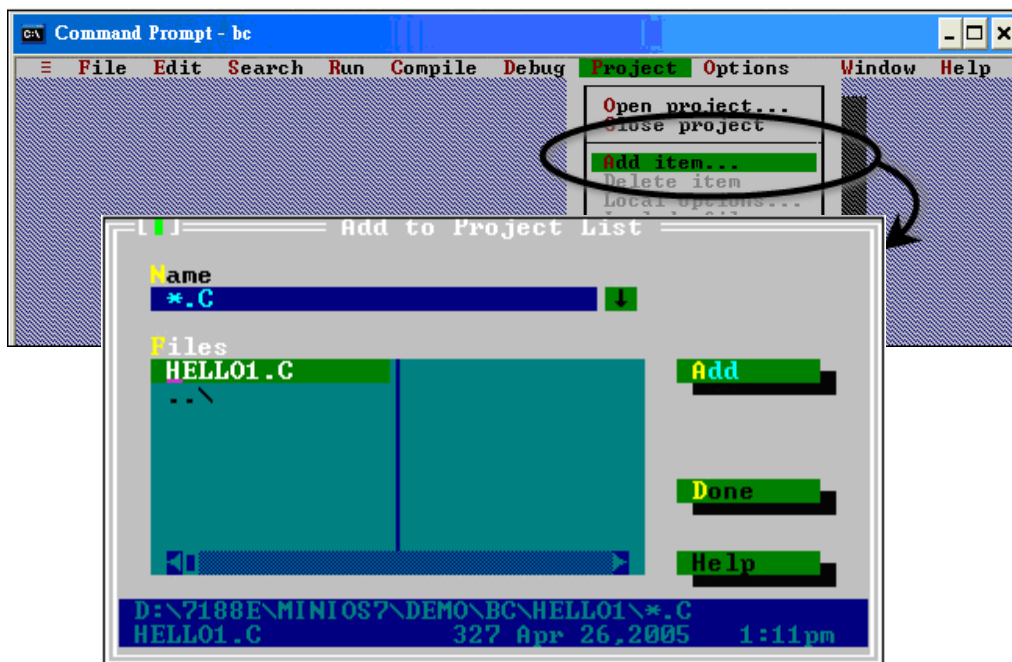
D.2. Borland C++ 3.1

Step 1: Executing the Borland C++ 3.1

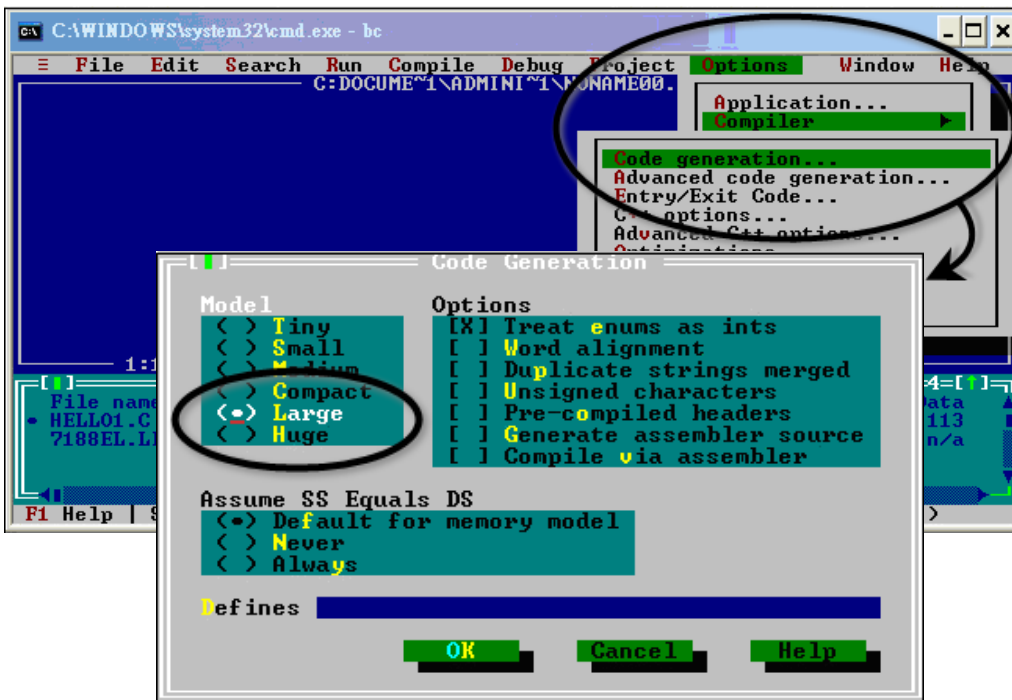
Step 2: Creating a new project file (*.prj)



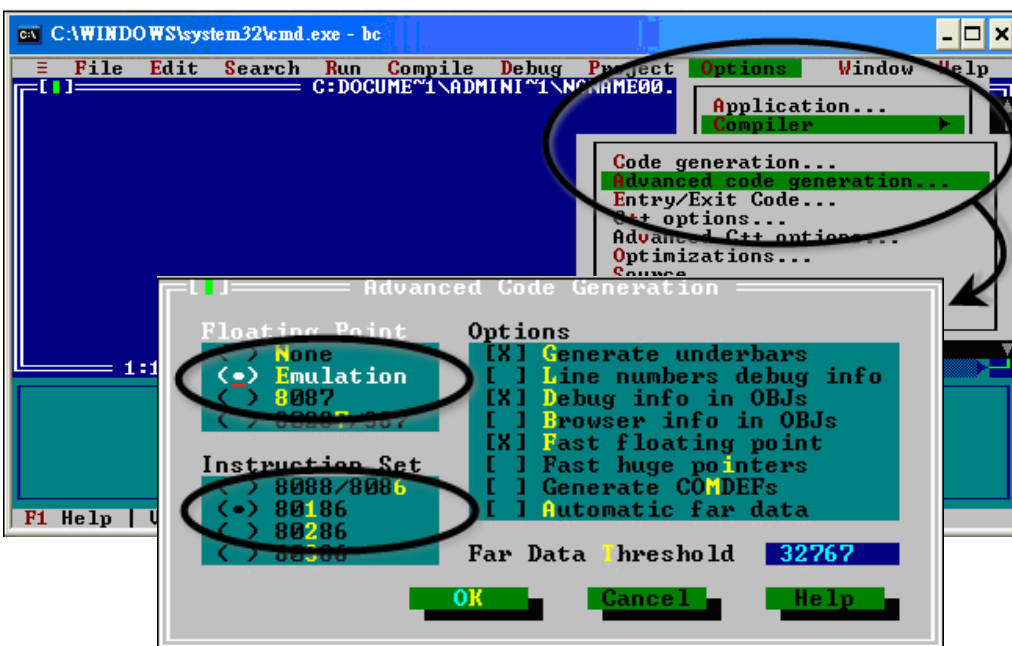
Step 3: Add all the necessary files to the project



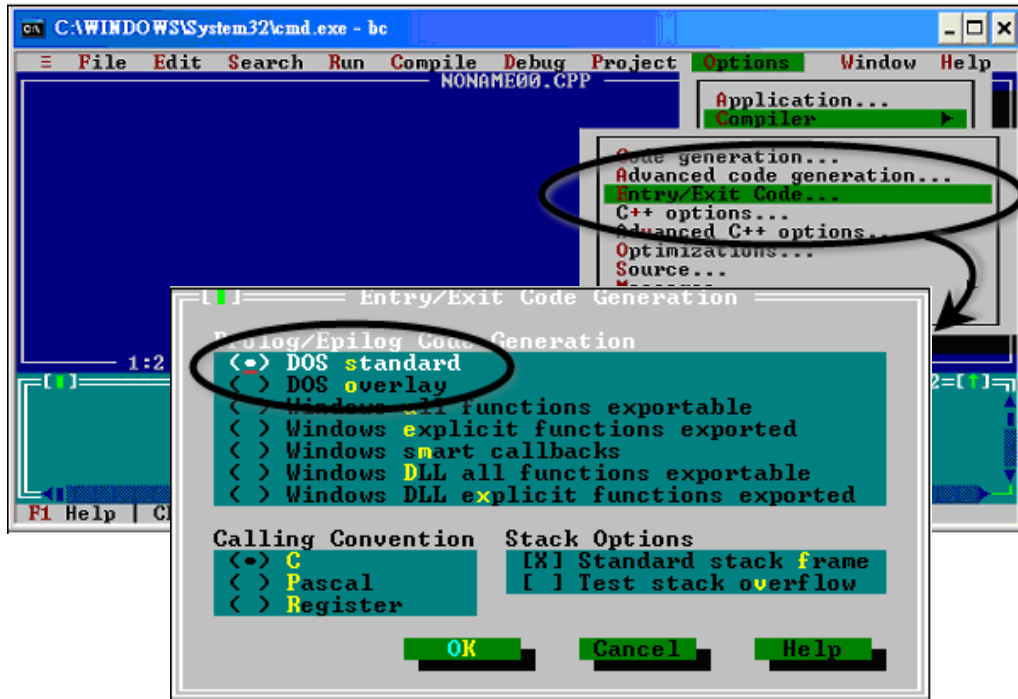
Step 4: Change the Memory model (Large for uPAC5000.lib)



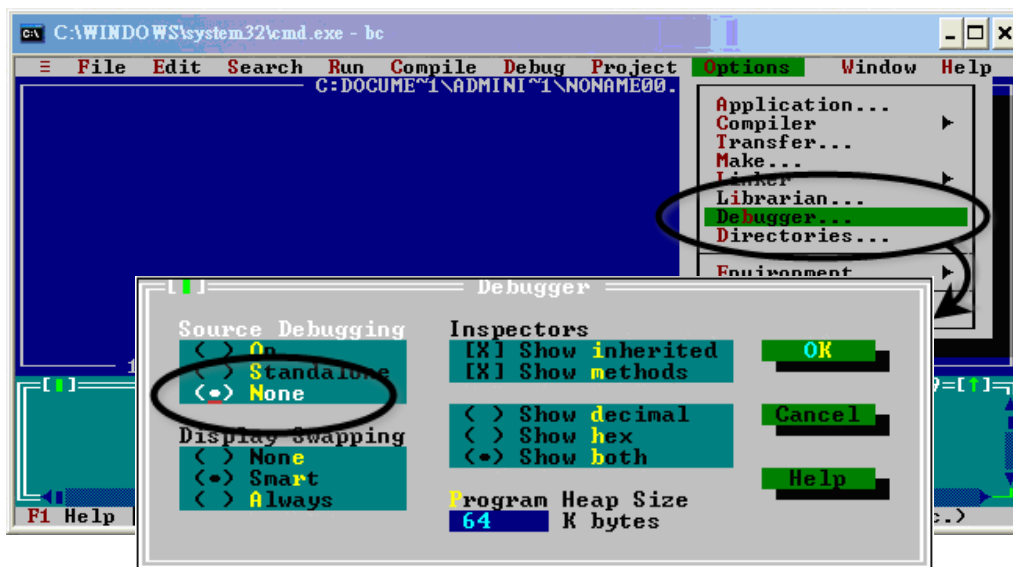
Step 5: Set the Advanced code generation options and Set the Floating Point to Emulation and the Instruction Set to 80186



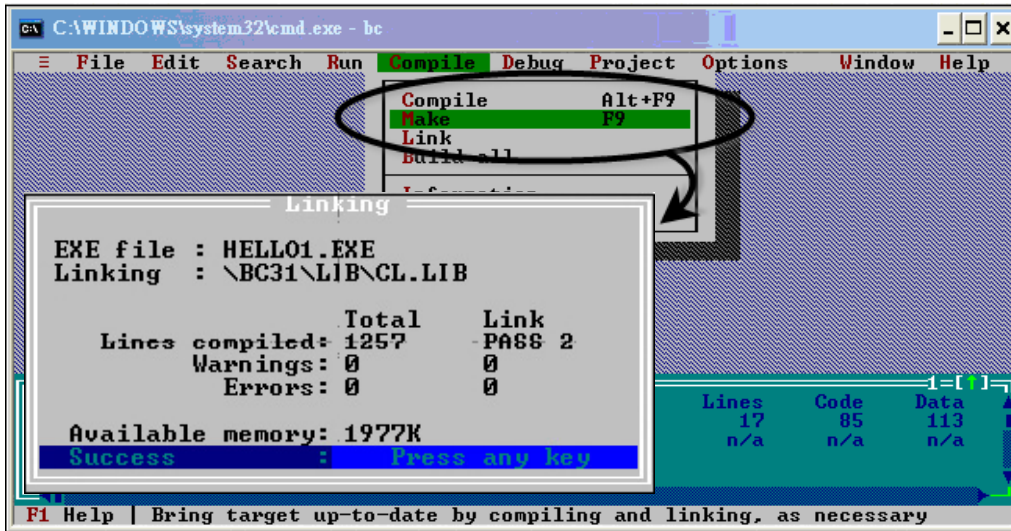
Step 6: Set the Entry/Exit Code Generation option and setting the DOS standard



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

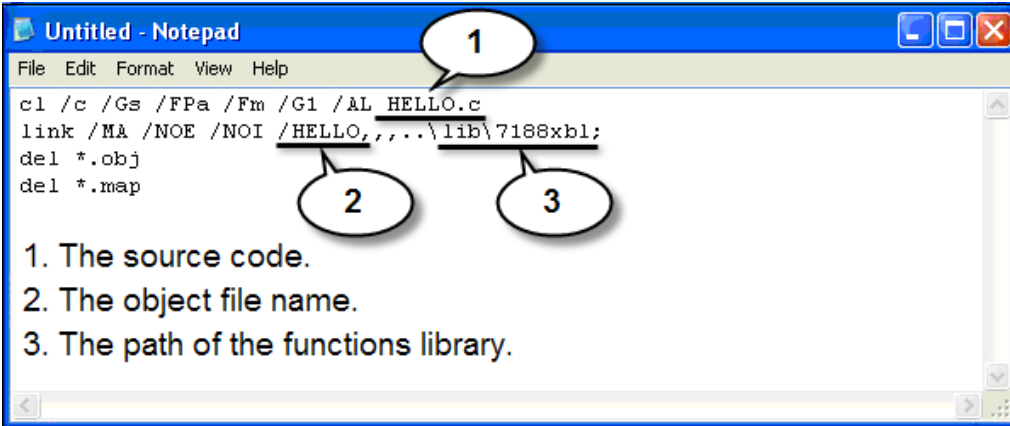


Step 8: Make the project



D.3. MSC 6.00

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



```
File Edit Format View Help
c1 /c /Gs /FPa /Fm /G1 /AL HELLO.c
link /MA /NOE /NOI /HELLO,....\lib\7188xbl;
del *.obj
del *.map
```

1. The source code.
2. The object file name.
3. The path of the functions library.

Tip & Warnings

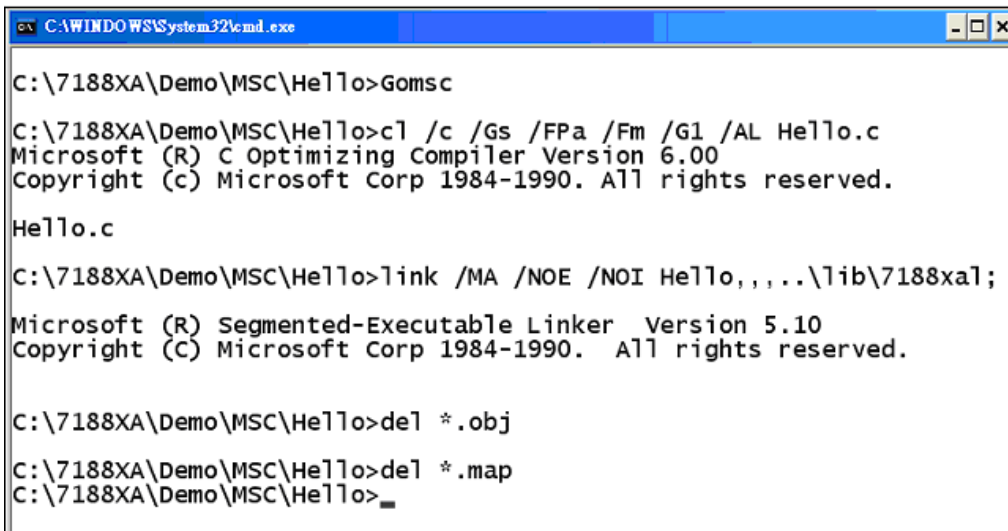


/C: Don't strip comments /GS: No stack checking

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

/G1: 186 instructions /AL: Large model

Step 2: Run the Gomsc.bat file



```
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\7188xal;
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

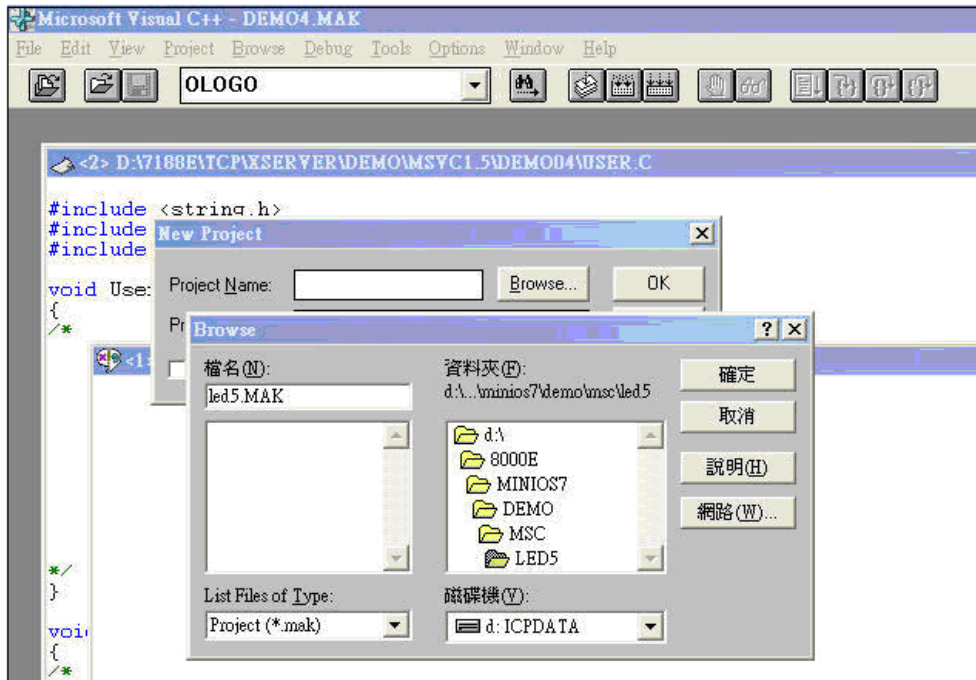
Directory of C:\7188XA\Demo\MSC\Hello

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             607 Hello.c
2006/05/29  17:08             6,715 HELLO.EXE
               3 File(s)              7,496 bytes
               2 Dir(s)  22,041,571,328 bytes free

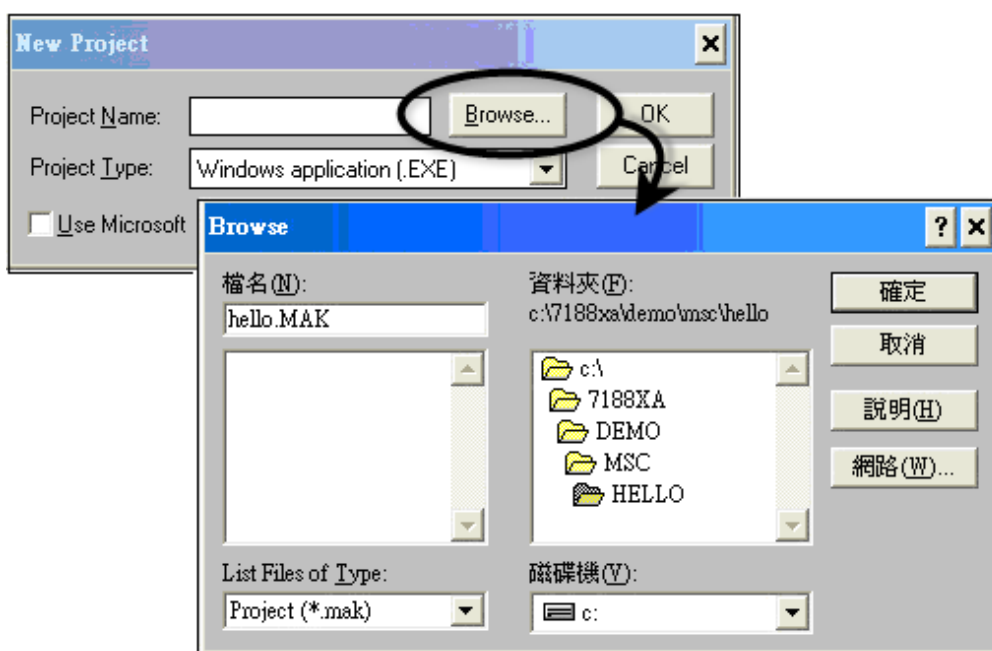
C:\7188XA\Demo\MSC\Hello>
```

D.4. MSVC 1.50

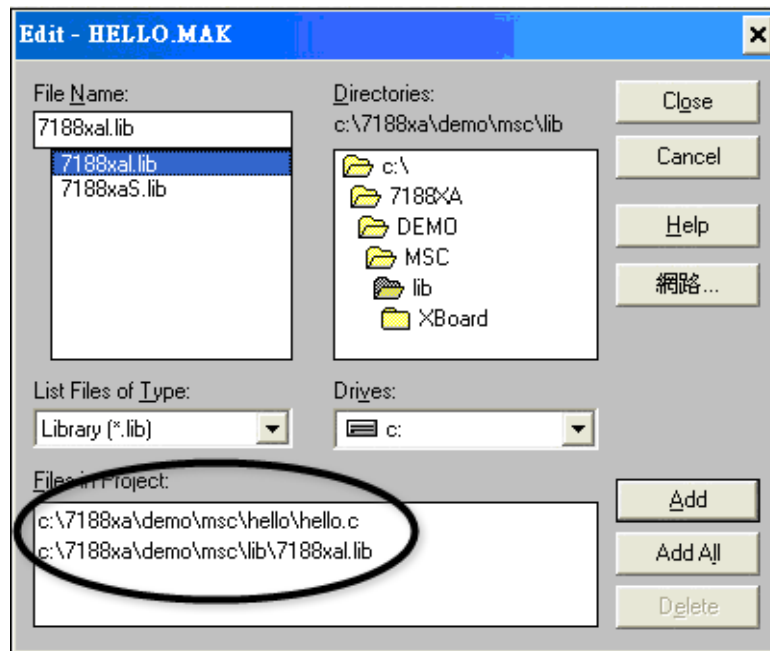
Step 1: Run MSVC.exe



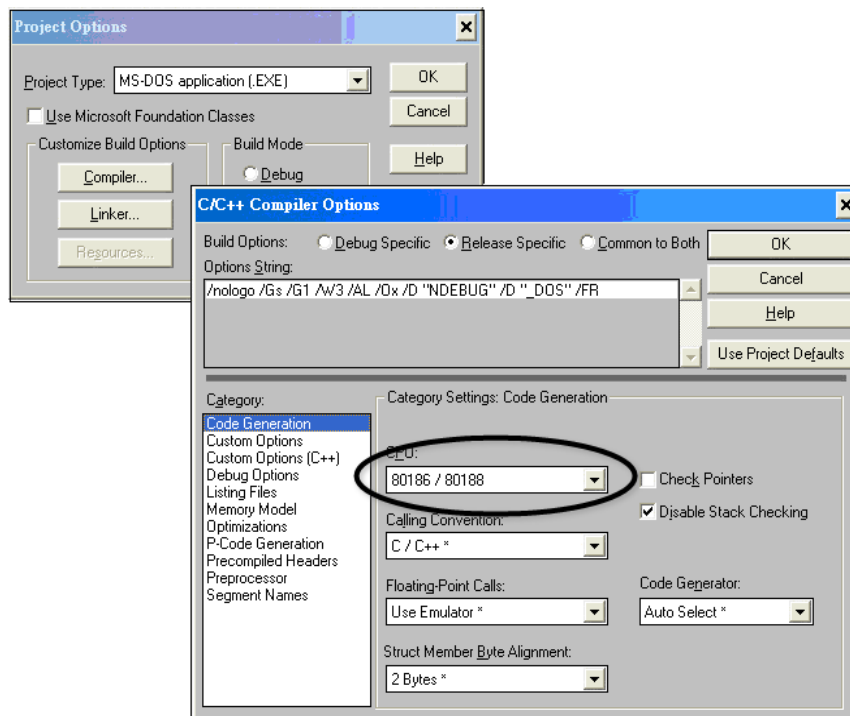
Step 2: Create a new project (*.mak) by entering the name of the project in the Project Name field and then select MS-DOS application (EXE) as the Project type



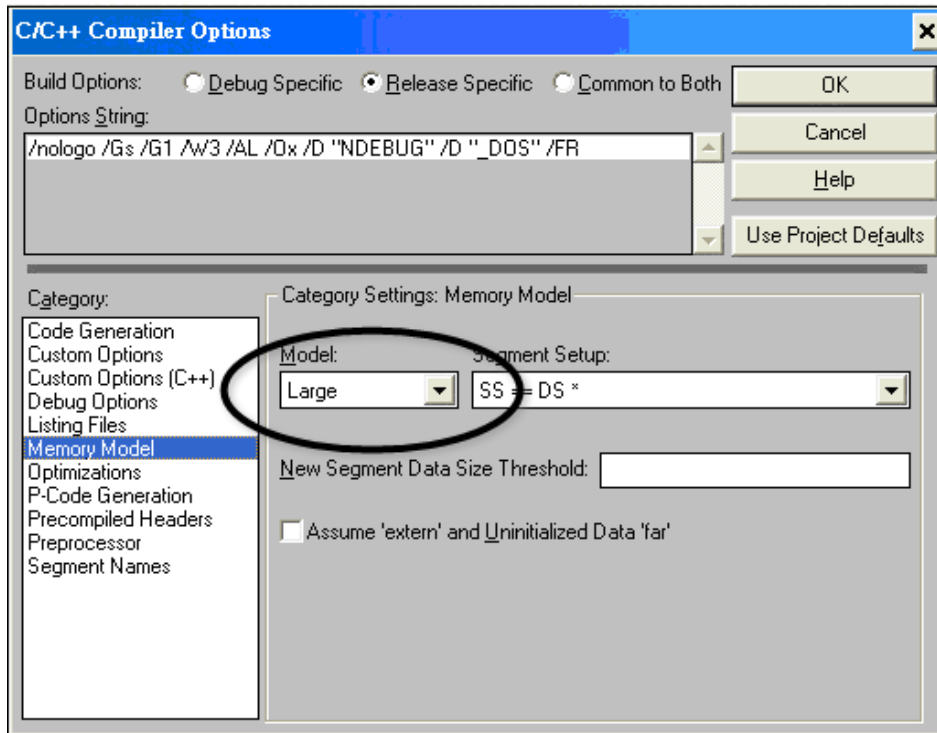
Step 3: Add the user's program and the necessary library files to the project



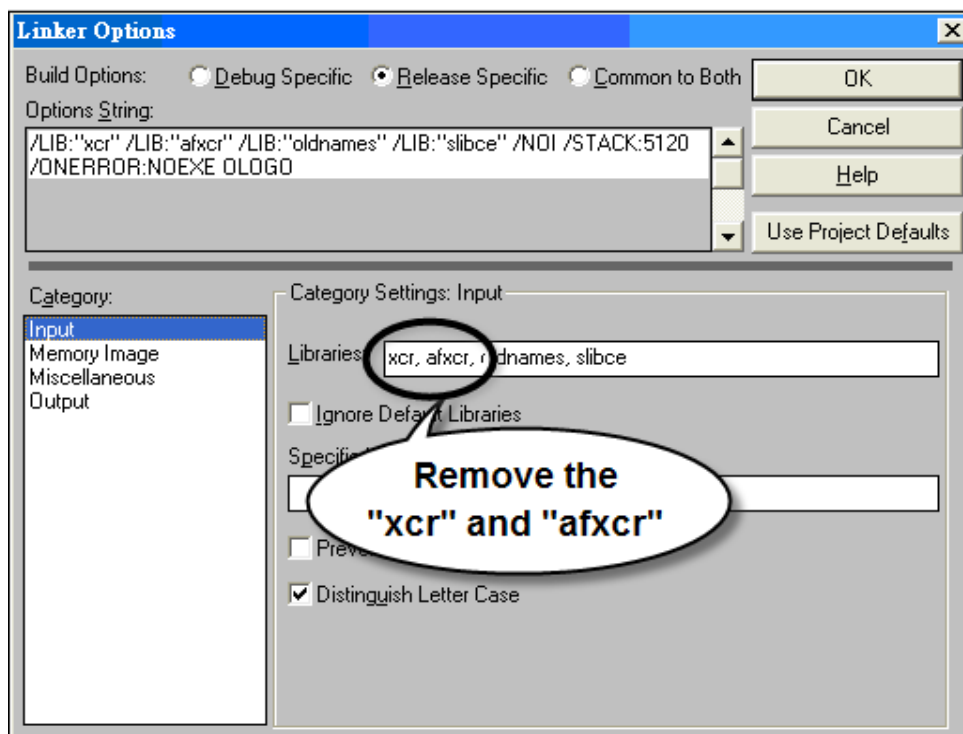
Step 4: Set the Code Generation on the Compiler.



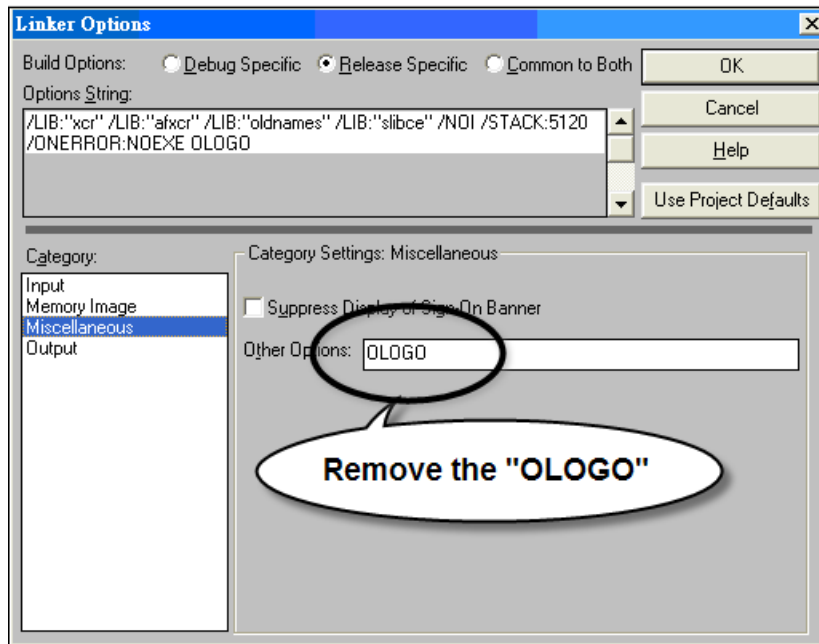
Step 5: Change the Memory model (large for uPAC5000.lib)



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

