# µPAC-5001D-CAN2 User Manual
# (C Language Based)

**Version 1.1, June. 2016**




## Service and usage information for

µPAC-5001D-CAN2

Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

## 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 @ 2016 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



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# Table of Contents


Table of Contents ----------3

1. Introduction ----------6

1.1. Features ---------- 7

1.2. µPAC-5001D-CAN2 Specifications ---------- 10

1.2.1. µPAC-5001D-CAN2 ---------- 10

1.3. Overview ---------- 11

1.4. Dimensions ---------- 16

1.5. Companion CD ---------- 17

2. Getting Started ---------- 18

2.1. Hardware Installation ---------- 19

2.2. Software Installation ---------- 21

2.3. Boot Configuration ---------- 23

2.4. Uploading µPAC-5001D-CAN2 Programs ---------- 24

2.4.1. Establishing a connection between PC and µPAC-5001D-CAN2 ---------- 25

(a) RS-232 connection ---------- 26

(b) Ethernet Connection ---------- 29

2.4.2. Uploading and executing µPAC-5001D-CAN2 programs ---------- 34

2.4.3. Making programs start automatically ---------- 35

2.5. Updating µPAC-5001D-CAN2 OS Image ---------- 37

3. “Hello World” - Your First Program ---------- 40

3.1. C Compiler Installation ---------- 41

3.1.1. Installing the C compiler ---------- 42

3.1.2. Setting up the environment variables ---------- 46

3.2. µPAC-5001D-CAN2’s APIs ---------- 49


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

3.3. First Program in µPAC-5001D-CAN2 ---------- 50

4. APIs and Demo References ---------- 60

4.1. API for COM Port ---------- 65

4.1.1. Types of COM port functions ---------- 66

4.1.2. API for MiniOS7 COM port ---------- 67

4.1.3. API for standard COM port ---------- 70

4.1.4. Port functions Comparison ---------- 72

4.1.5. Request/Response protocol define on COM port ---------- 73

4.2. API for I/O Modules ---------- 74

4.3. API for EEPROM ---------- 76

4.4. API for Flash Memory ---------- 78

4.5. API for NVRAM ---------- 80

4.6. API for 5-Digital LED ---------- 82

4.7. API for Timer ---------- 83

4.8. API for WatchDog Timer (WDT) ---------- 85

4.9. API for microSD ---------- 86

4.10. API for CAN bus ---------- 90

4.10.1. API for CAN Initialization ---------- 91

4.10.2. API for CAN Interrupt ---------- 96

4.10.3. API for Transmitting CAN Messages ---------- 99

4.10.4. API for Receiving CAN Messages ---------- 103

4.10.5. API for LED Indicator ---------- 108

4.10.6. API for User-Defined Interrupt ---------- 110

4.10.7. API for Detecting CAN Bus Baud Rate ---------- 113

4.10.8. Return Code ---------- 116

Appendix A. What is MiniOS7? ---------- 117

Appendix B. What is MiniOS7 Utility? ---------- 118

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

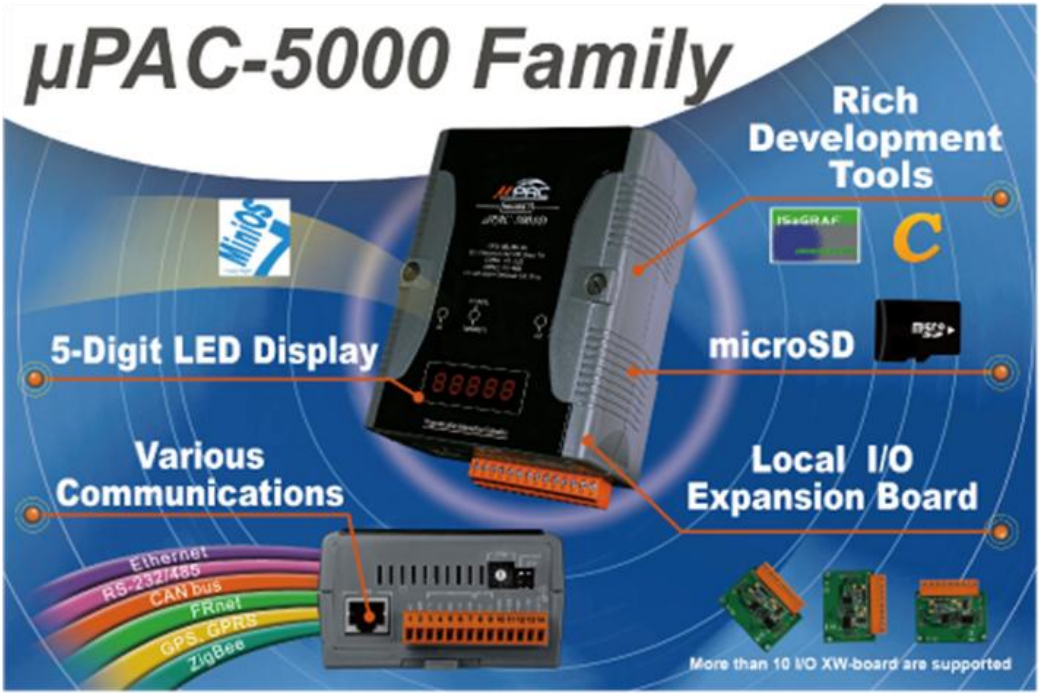
Appendix C. More C Compiler Settings ---------- 119
C.1. Turbo C 2.01 ---------- 120
C.2. Borland C++ 3.1 ---------- 123
C.3. MSC 6.00 ---------- 127
C.4. MSVC 1.50 ---------- 129
Appendix D. Core's application and wiring ---------- 133

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 1. Introduction




The µPAC-5001D-CAN2 is palm size PACs (**P**rogrammable **A**utomation **C**ontroller). With abundant and various peripherals and communication ports, the µPAC-5001D-CAN2 can integrate different communication interface, like CAN bus, RS-232, RS-485, Ethernet and so on. In order to increase the modules openness and applications flexibility, the µPAC-5001D-CAN2 provides DOS-like real-time single-task operation system for adapting to all kinds of needs. Users can develop application programs via C/C++ compiler. In respect of application development, the µPAC-5001D-CAN2 provides various libraries and demo programs about the peripheral components. Users may be confused how to use those abundant libraries. For that purpose, they also provide demo programs about libraries. When users need to use various peripheral like communication ports, watch dog, real-time clock (RTC), seven-segment display, reading/accessing into flash, EEPROM, MircoSD, even using 64-bit hardware unique series number to protect the development of application software, they can be completed easily and quickly to meet all kinds of needs for application program development by revising and combining these sample programs. In order to work under harsh environment, the µPAC-5001D-CAN2 was designed with low power consumption and fanless products. Besides, they add all manner of anti-jamming protection components on circuit design and meet requirements of the wide operating temperature and wide operating voltage. Basis of all features which are described previously, the µPAC-5001D-CAN2 are ideal for data collection applied to integrate all kinds of communication interface or as a data processing center. They are also designed to convert different protocol network such as gateway or bridge devices.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 1.1. Features

## ➢ Various CPU and OS for Choosing

MiniOS7
80186 CPU
µPAC-5000 Series

- DOS-like OS
- Boot up in 0.4 ~ 0.8 second
- Build-in hardware diagnostic
- Standard version for C language programming

## ➢ Remote I/O Module and Expansion Unit

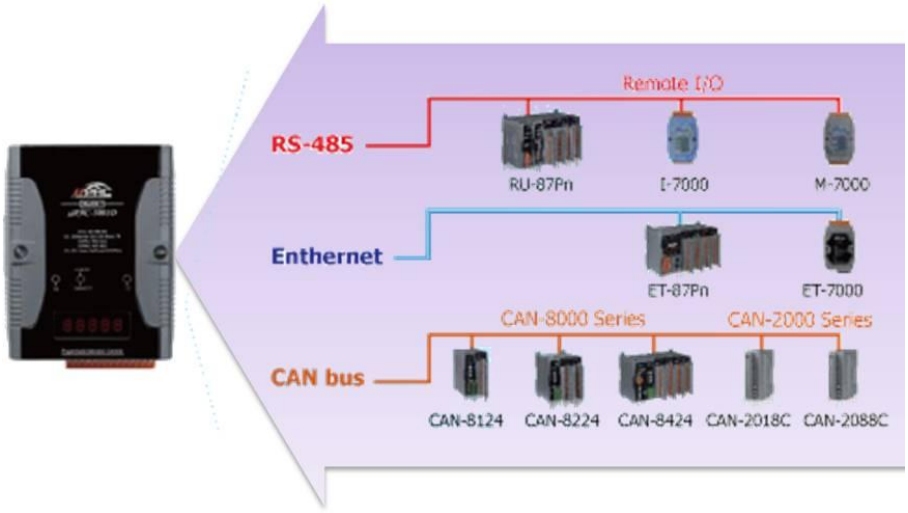
With the built-in CAN, RS-232/485 and Ethernet port, the µPAC-5001D-CAN2 can connect CAN/RS-485/Ethernet remote I/O Units (CAN-8x23/CAN-8x24/RU-87Pn/ET-87Pn) or modules (CAN-2000D/CAN-2000C/I-7000/ M-7000/ET-7000).




## ➢ Multi-Communication Interface

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

- Ethernet
- RS-232/485
- CAN bus






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

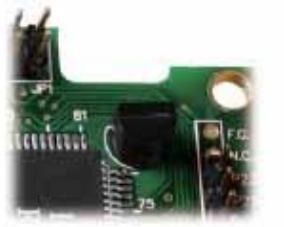
## ➢ Various Memory Expansions




µPAC-5001D-CAN2 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 2 GB on the Linux and WinCE platform

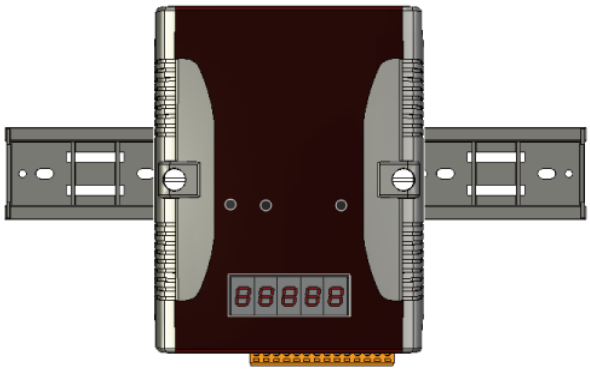
## ➢ Unique 64-bit Hardware Serial Number to Protect Your Program



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

## ➢ Small and Easy Installation

µPAC-5001D-CAN2 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 µPAC-5001D-CAN2 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.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

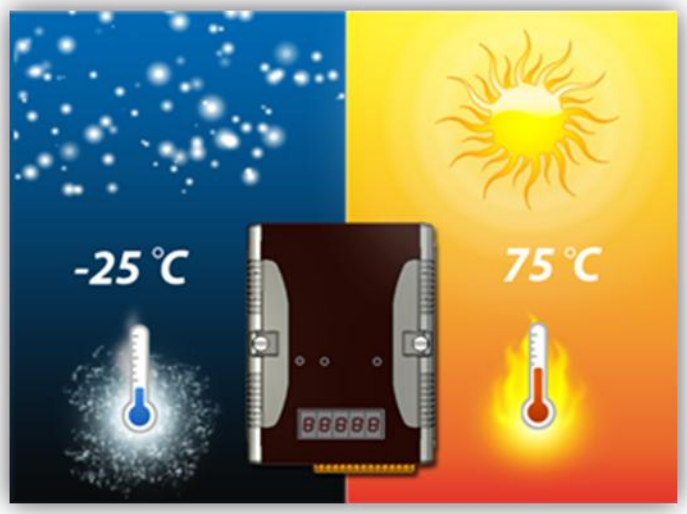


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## ➢ Highly Reliable Under Harsh Environment

The µPAC-5001D-CAN2 operates in a wide range of temperature and humidity.

- Operating Temperature:
  -25℃ ~ +75 ℃
- Storage Temperature:
  -30℃ ~ +80 ℃
- Humidity:
  10 ~ 95% RH (non-condensing)




LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 1.2. µPAC-5001D-CAN2 Specifications

## 1.2.1. µPAC-5001D-CAN2

| Models | µPAC-5001D-CAN2 |
|---|---|
| **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 |
| 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) |
| **Ethernet Interface** | |
| Controller | 10/100 Base-TX (Auto-negotiating, Auto MDI/MDI-X, LED indicators) |
| **UART Interface** | |
| COM1 | RS-232 |
| COM2 | RS-485 (D2+, D2-), self-tuner ASIC inside,. |
| **LED Indicator** | |
| Programmable LED | 5 |
| LED Display | 5-digit 7-segnment LED display |
| **CAN Interface** | |
| Controller | NXP SJA1000T with 16MHz clock Frequency |
| Transceiver | NXP TJA1042 |
| Channel Number | 2 |
| Connector | 18-pin screwed terminal block (CAN_GND, CAN_L, CAN_GND) |
| Transmission Speed(bps) | 5 k ~ 1 M selected by user defined |
| Terminator Resistor | Jumper for the 120 Ω terminator resistor |
| Specification | ISO 11898-2, CAN 2.0A and CAN 2.0B |
| **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 | 3 W |



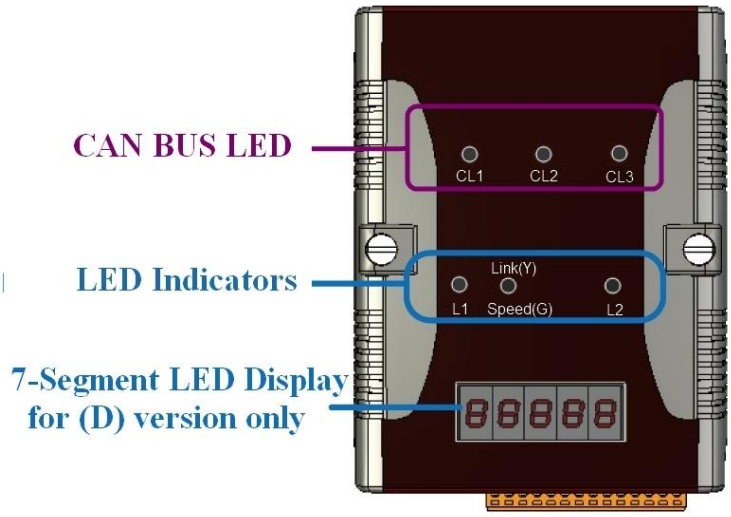
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

# 1.3. Overview

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

## Front Panel




The LED indicators and 5-digit 7-Segnment 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-5001D-CAN2, their functions are summarized in the table below.

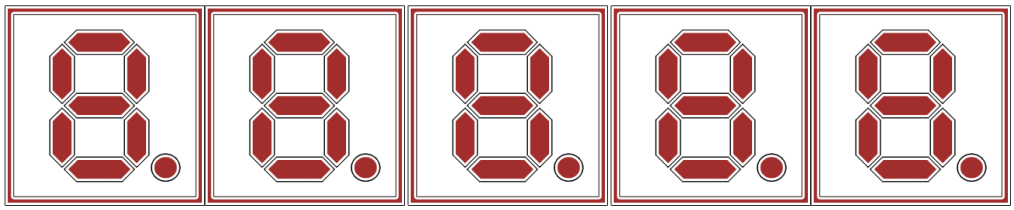
| Indicator | State | Meaning |
|---|---|---|
| L1 | Flashing | User programmable LED |
| L2 | OFF | User programmable LED |
| Link (G) | Permanently on | Ethernet link detected |
| | Permanently off | No Ethernet link detected |
| | Flashing green | Ethernet packet received |
| CL1 | OFF | User programmable LED |
| CL2 | OFF | User programmable LED |
| CL3 | OFF | User programmable LED |

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

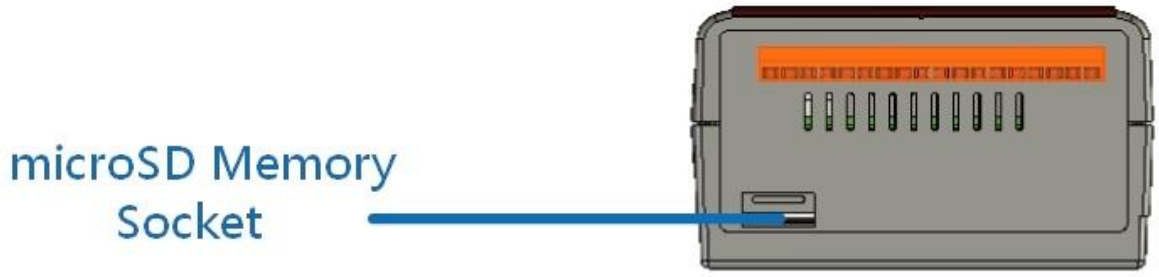
➤ 5-Digit 7-SEG LED display (for display version only)



µPAC-5001D-CAN2 equips with 5-digital 7-SEG LED display that can be used to display decimal numbers from 0 to 9 and provide 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

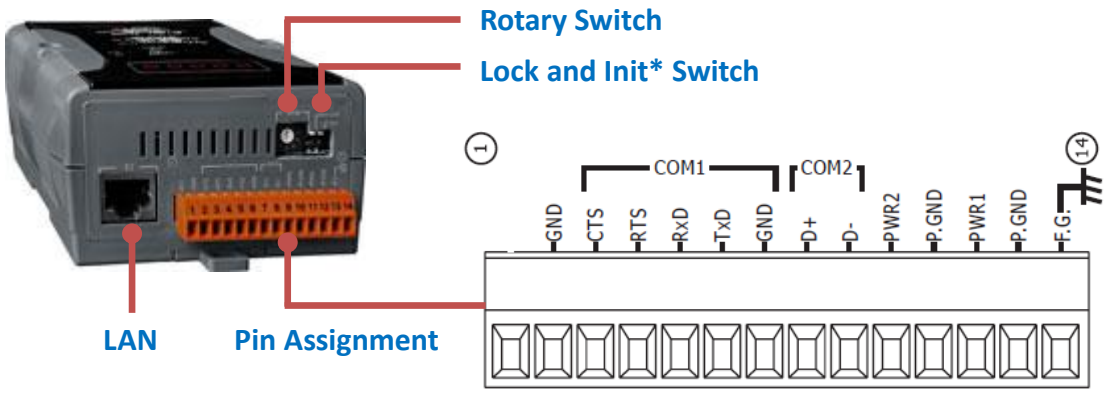
µPAC-5001D-CAN2 equips a microSD slot and it can support up to 2 GB microSD card.



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 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 µPAC-5001D-CAN2, the switch is always in the OFF position. Only when updating the µPAC-5001D-CAN2 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

µPAC-5001D-CAN2 Flash memory with Write Protection can physically lock that prevents modification or erasure of valuable data onµPAC-5001D-CAN2.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

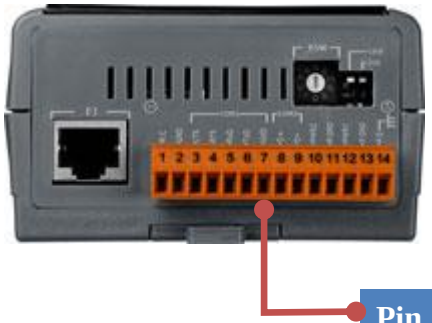
## ➤ LAN

µPAC-5001D-CAN2 includes an Ethernet port for network equipment use, and it supports RJ-45 connectors.

An Ethernet port is an opening on µPAC-5001D-CAN2 network equipment that Ethernet cables plug into. Ethernet ports accept cables with RJ-45 connectors.

## ➤ Pin Assignment

The pin assignments of the connector is as follows:



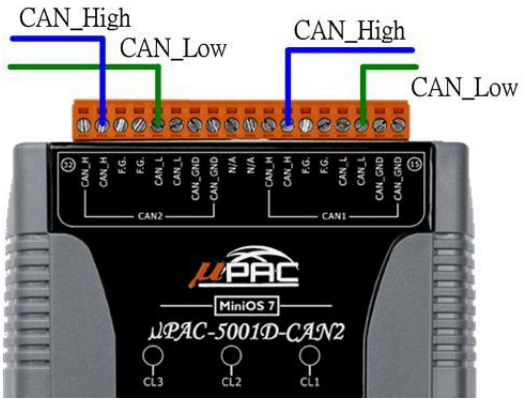

| Pin | Signal | Description |
|---|---|---|
| 1 | N/A | Unassigned |
| 2 | GND | Ground |
| 3 | CTS | COM1 (RS-232) |
| 4 | RTS | |
| 5 | RxD | |
| 6 | TxD | |
| 7 | GND | |
| 8 | D+ | COM2 (RS-485) |
| 9 | D- | |
| 10 | PWR2 | Power Input 2 |
| 11 | P.GND | |
| 12 | PWR1 | Power Input 1 |
| 13 | P.GND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

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

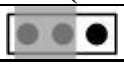
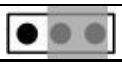


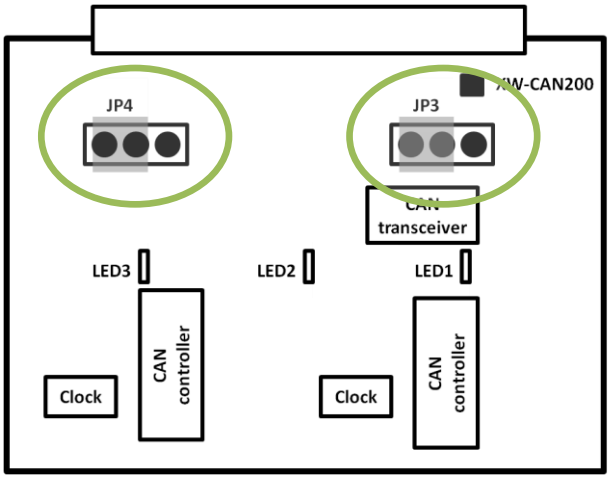
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## ➤ CAN BUS Pin Assignment




## ➤ Terminal Resistor Jumper Selection

| Apply the termination resistor(120Ω) | Don't apply the termination resistor |
|---|---|
|  |  |





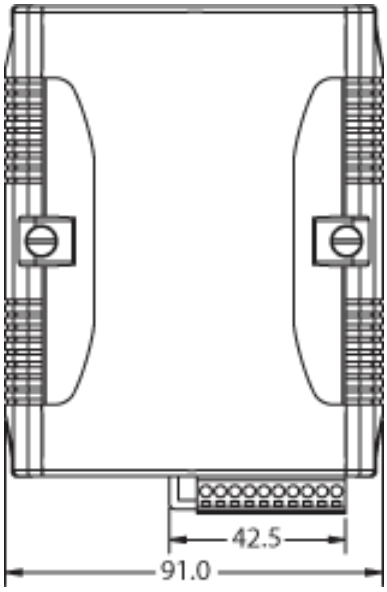


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

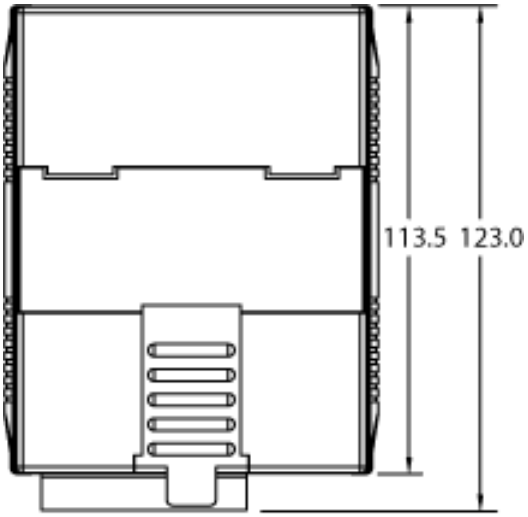
# 1.4. Dimensions

All dimensions are in millimeters.

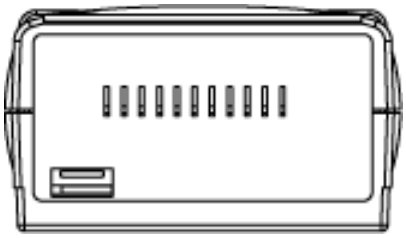
Front View



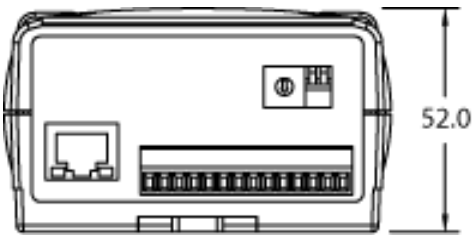

Back View




Top View



Bottom View



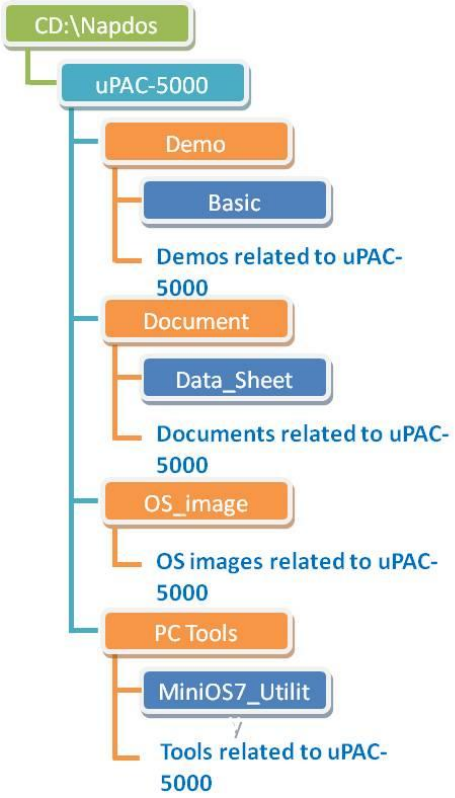



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

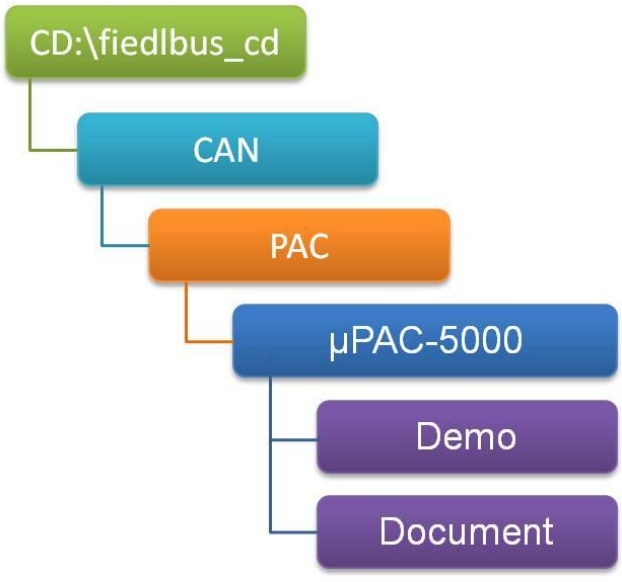
# 1.5. Companion CD

This package comes with two CDs. One of them provides standard drivers, software utility and the basic documentations. The other provides CAN bus demo and documentations.All of the CD content is listed below.

### CD:\Napdos




### CD:\fieldbus_cd






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

# 2. Getting Started

If you are a new user, please begin with this chapter. It includes a guided tour that provides a basic overview of installing, configuring and using for the µPAC-5001D-CAN2.

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:



**µPAC-5001D-CAN2**




**Software Utility CD**



**RS-232 cable (CA-0910)**



**Screw Driver (1C016)**



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 2.1. Hardware Installation

Before installing the hardware, you should have a basic understanding of hardware specification, such as the hard disc units, 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”

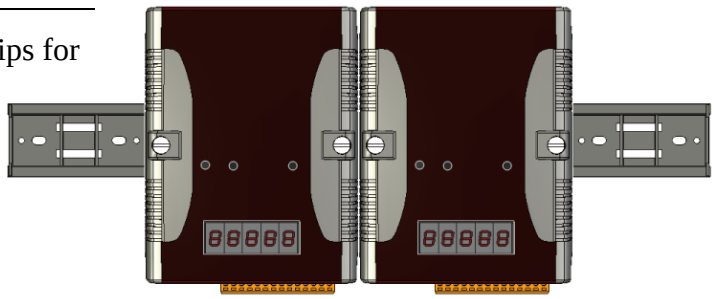
Below are step-by-step instructions for deploying the basic µPAC-5001D-CAN2 system.

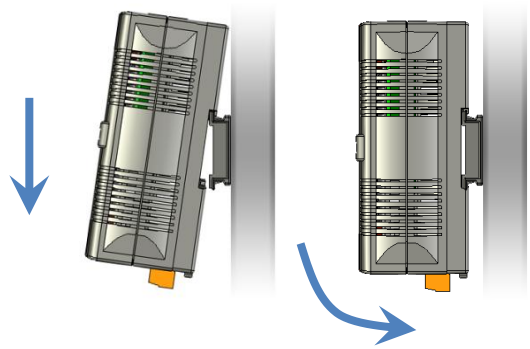
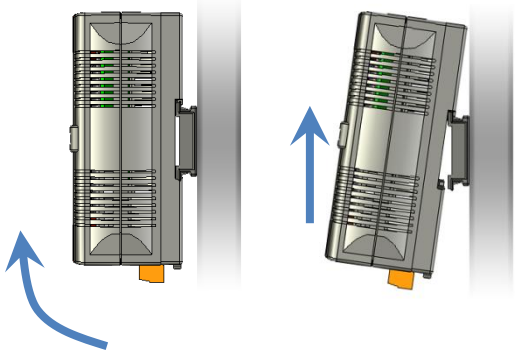
Step 1: Mount the hardware

The µPAC-5001D-CAN2 can be mounted with the bottom of the chassis on the DIN rail or piggyback.

### ➤ DIN Rail mounting

The µPAC-5001D-CAN2 has simple rail clips for mounting reliably on a standard 35 mm DIN rail.



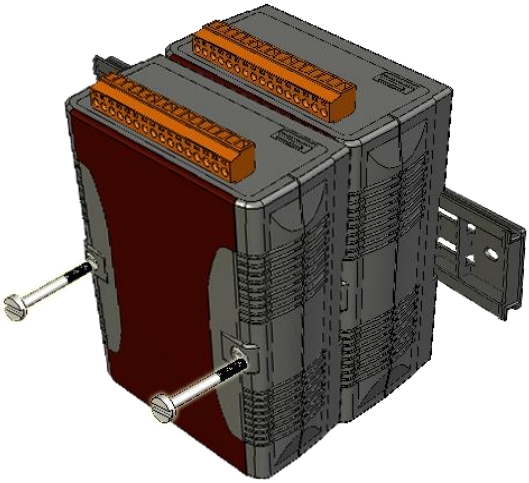
| Mounting on DIN rail | Removing from DIN rail |
|---|---|
|  |  |



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## ➢ Piggyback mounting

The µPAC-5001D-CAN2 has two holes on both sides for piggyback mounting.

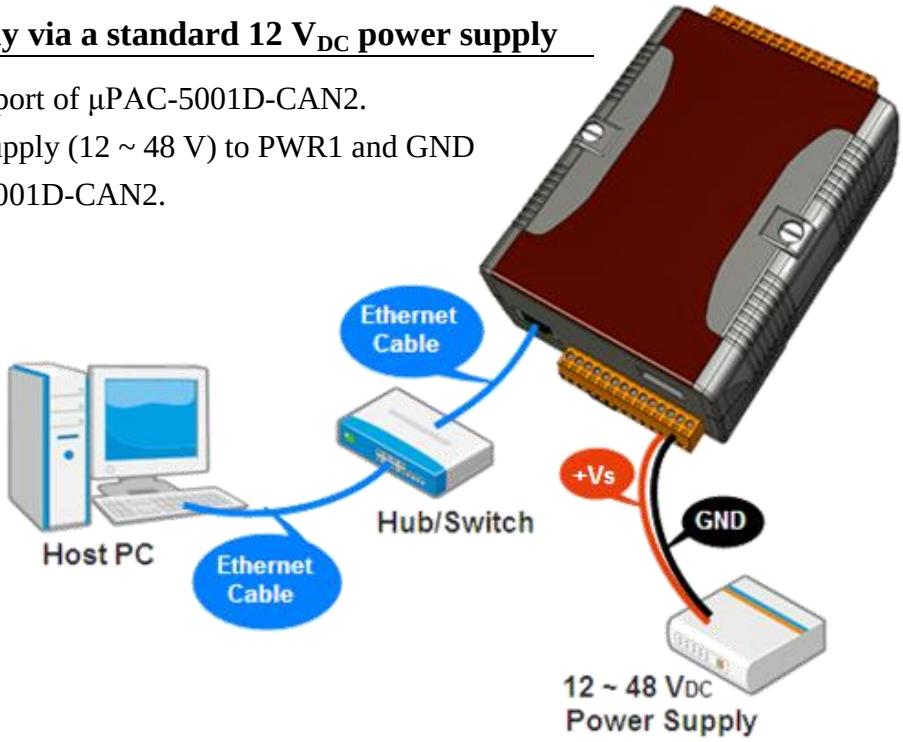


Step 2: Connect the µPAC-5001D-CAN2 to PC and setting up the power supply

The µPAC-5001D-CAN2 equips 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 µPAC-5001D-CAN2.
ii. Connect the power supply (12 ~ 48 V) to PWR1 and GND terminals of µPAC-5001D-CAN2.




LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



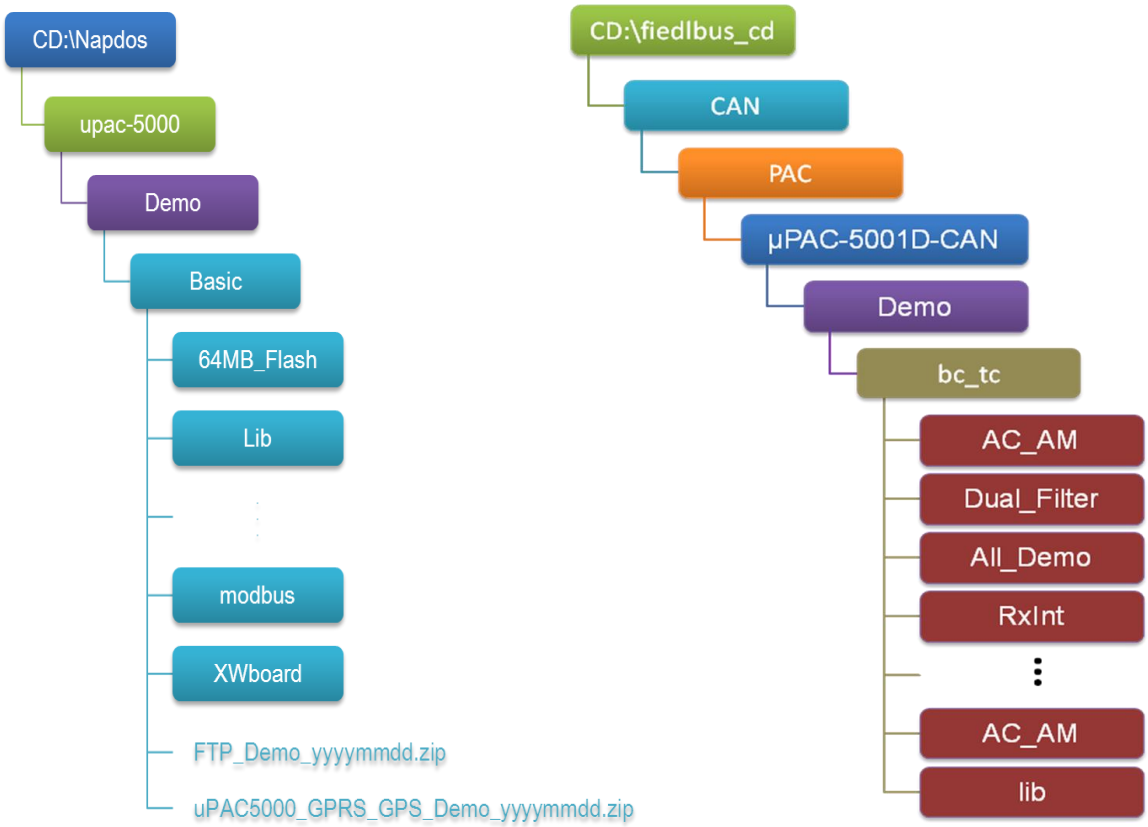
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 2.2. Software Installation

The Companion CDs includes APIs, demo programs and other tools for µPAC-5001D-CAN2. Below is a step-by-step instruction for installing the 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.






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Step 2: Installing the MiniOS7 Utility




MiniOS7 Utility is a suite of tool for managing MiniOS7 devices (µPAC-5001D-CAN2). It comprises four components – System monitor, communication manager, file manager and OS loader.

You can obtain the MiniOS7 Utility 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/

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

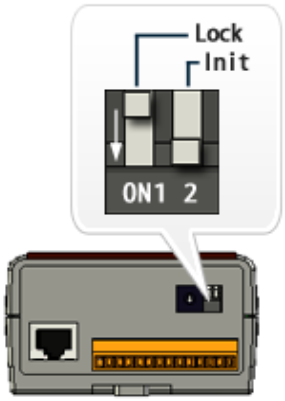


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 2.3. Boot Configuration

Before you upload some programs to µPAC-5001D-CAN2, 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.




LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 2.4. Uploading µPAC-5001D-CAN2 Programs

MiniOS7 Utility is a suite of tool for managing MiniOS7 devices (µPAC-5001D-CAN2). It comprises four components – System monitor, communication manager, file manager and OS loader.

Before you begin using the MiniOS7 Utility to upload programs, please check that µPAC-5001D-CAN2 is connected to PC.

The upload process has the following main steps:

1. Establishing a connection between PC and µPAC-5001D-CAN2
2. Uploading and executing programs on µPAC-5001D-CAN2
3. Making programs start automatically

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

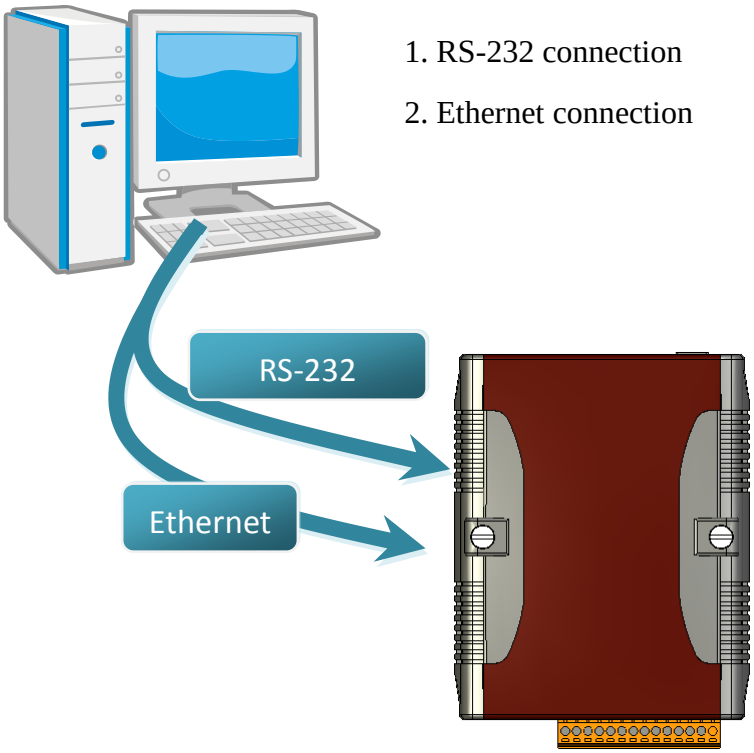


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

### 2.4.1. Establishing a connection between PC and µPAC-5001D-CAN2

There are two ways to establish a connection between PC and µPAC-5001D-CAN2.




1. RS-232 connection
2. Ethernet connection

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



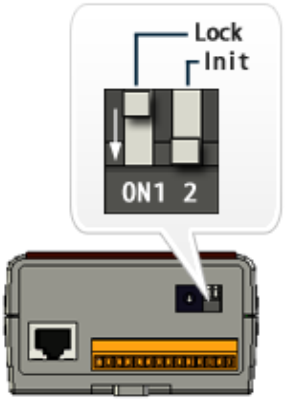
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

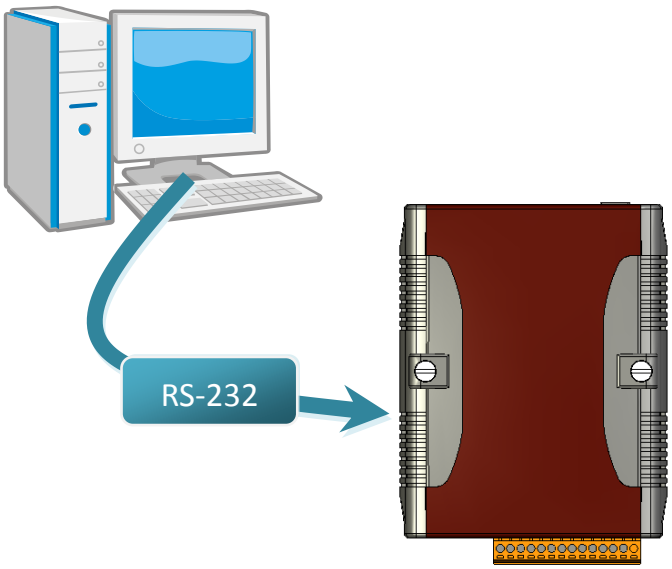
## (a) RS-232 connection

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 µPAC-5001D-CAN2.




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



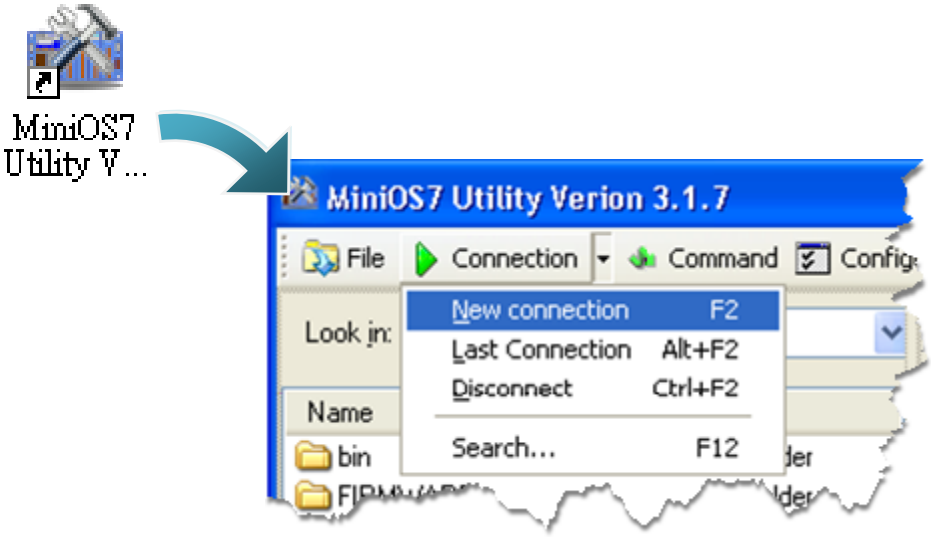



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

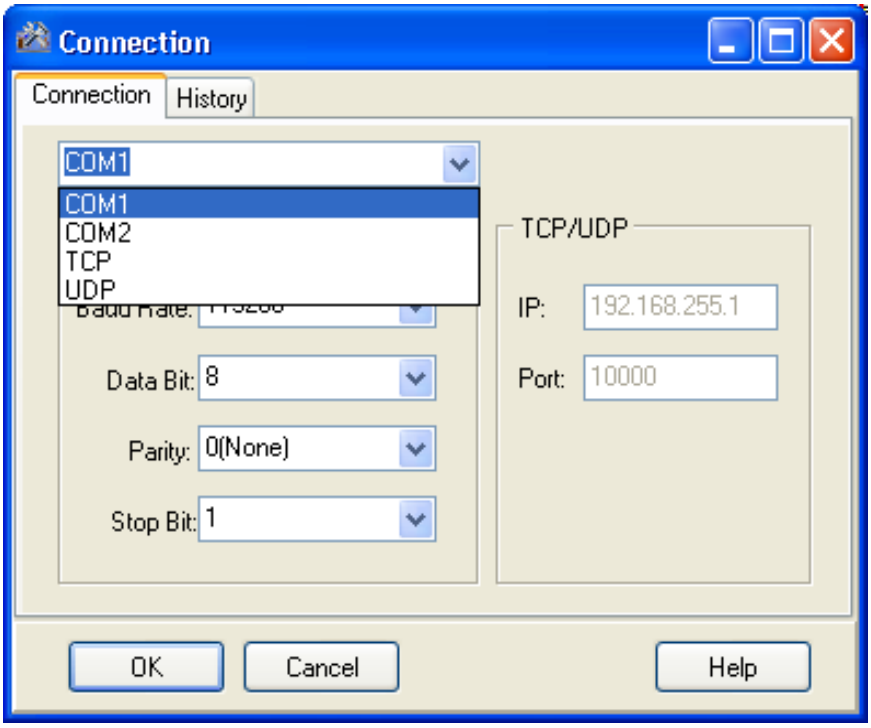
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Step 3: Run the MiniOS7 Utility

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



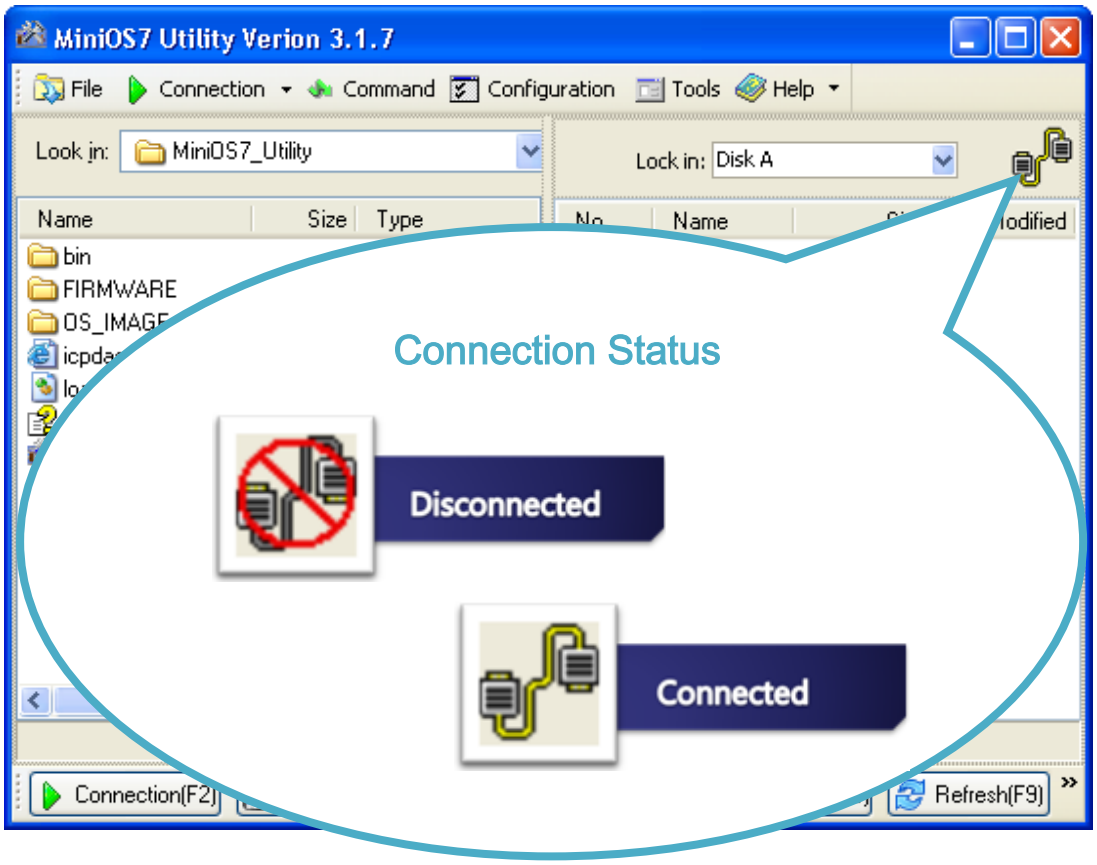

Step 5: On the “Connection” tab of the “Connection” dialog box, choose the COM port that your µPAC-5001D-CAN2 is connecting to, and then click “OK”






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Step 6: The connection has already established





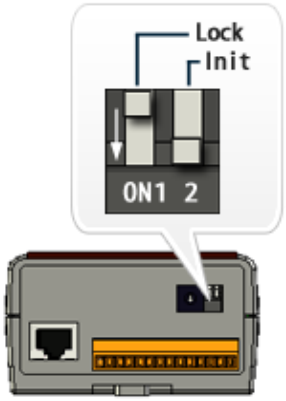


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

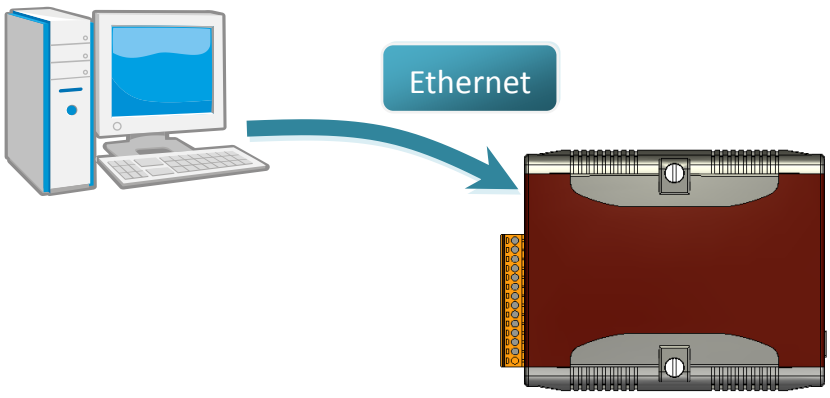
## (b) Ethernet Connection

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 µPAC-5001D-CAN2.




Step 2: Use an Ethernet cable to connect to PC

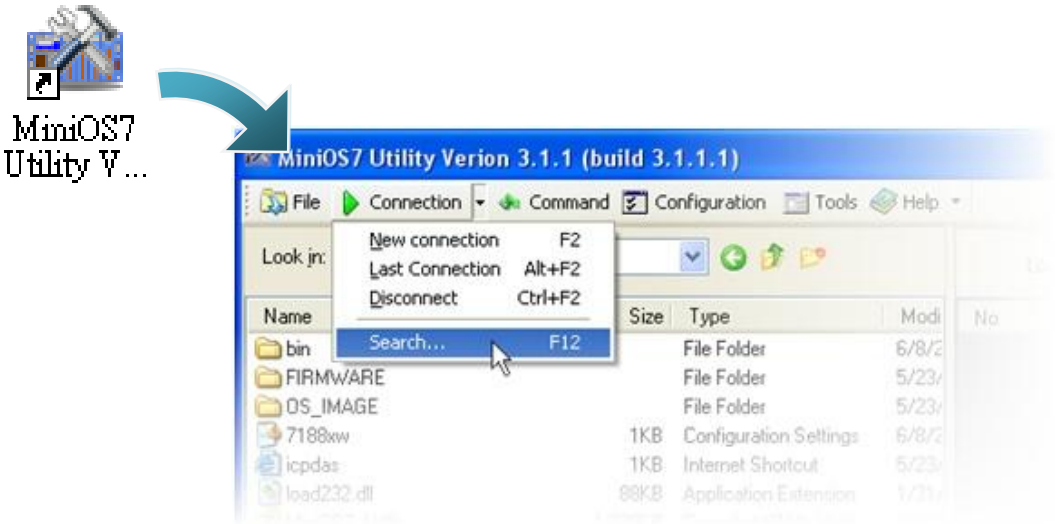





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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

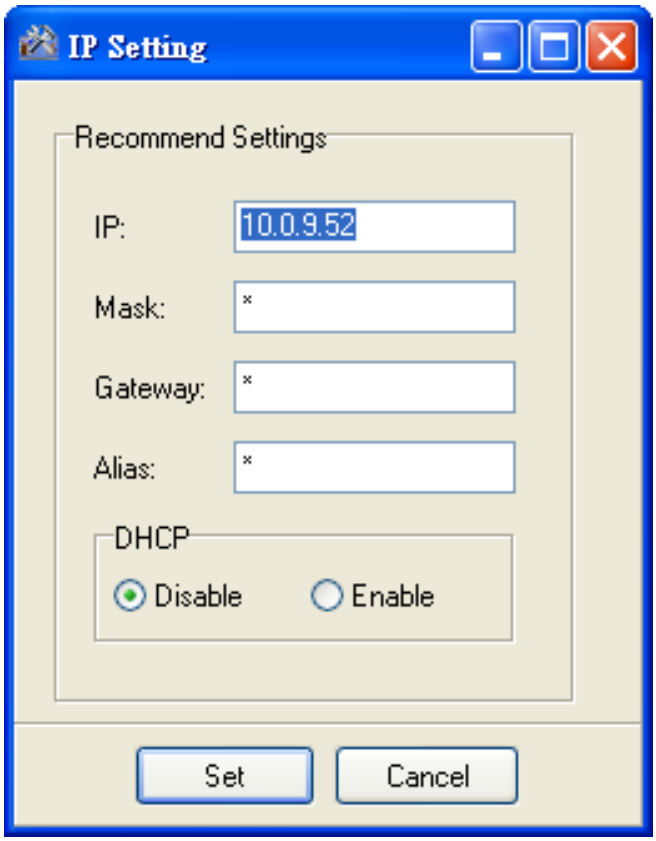




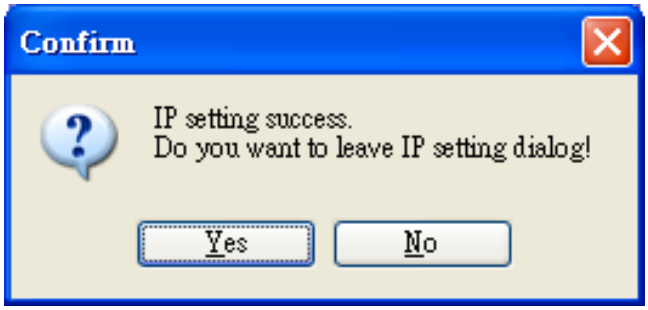


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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



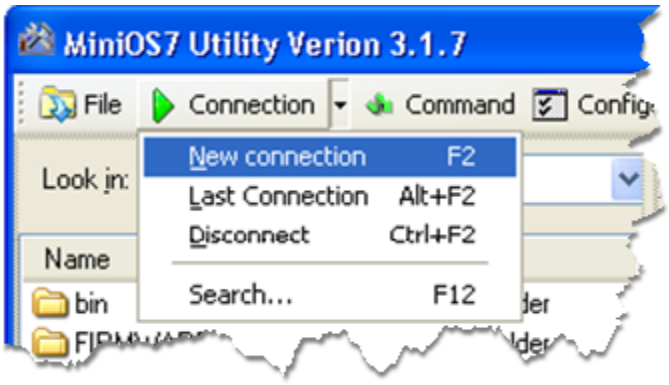

Step 7: On the “Confirm” dialog box, click “Yes”



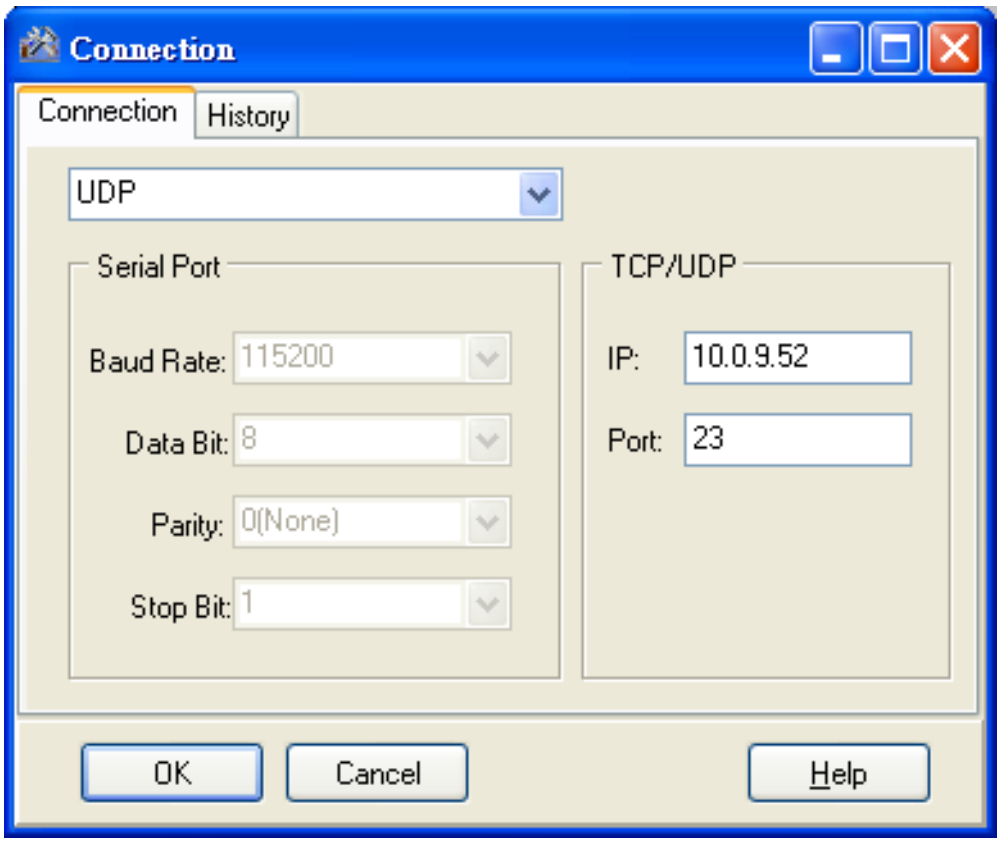


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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”

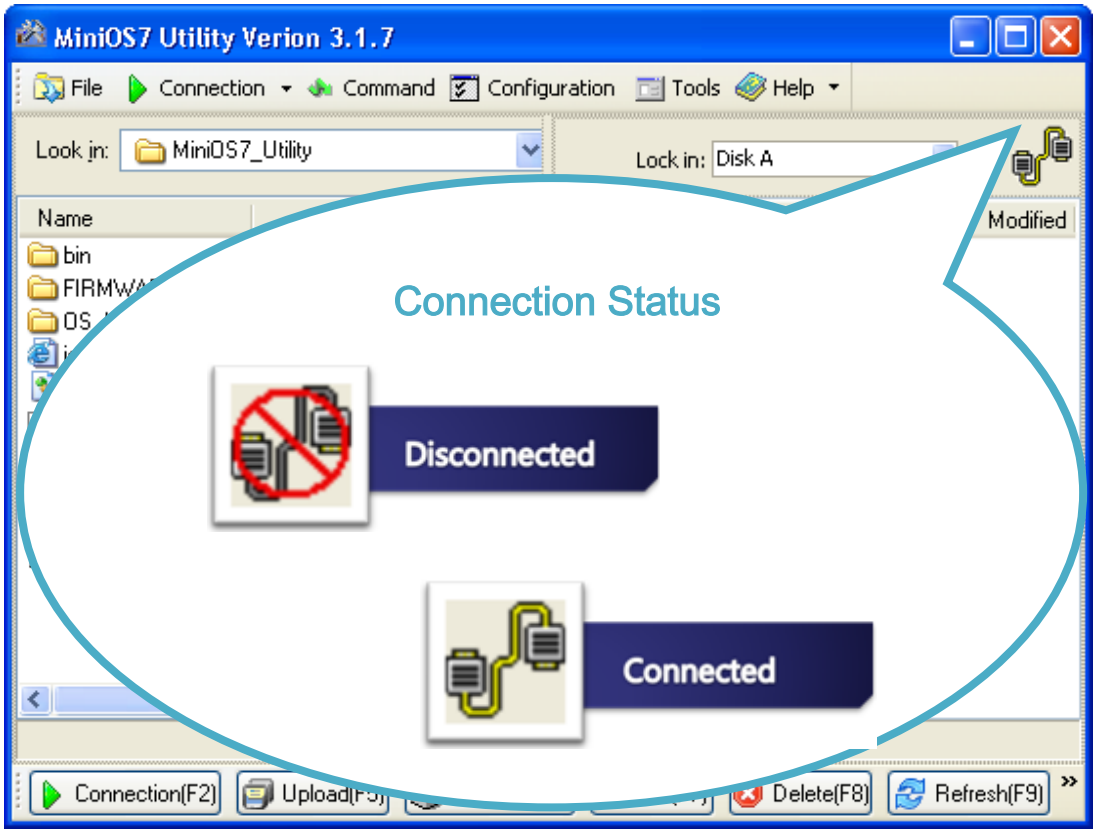



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Step 10: The connection has already established



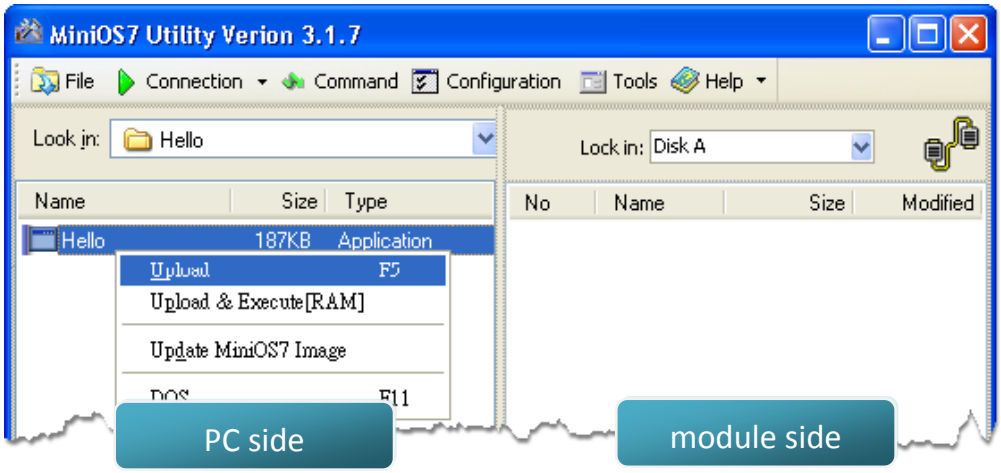



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

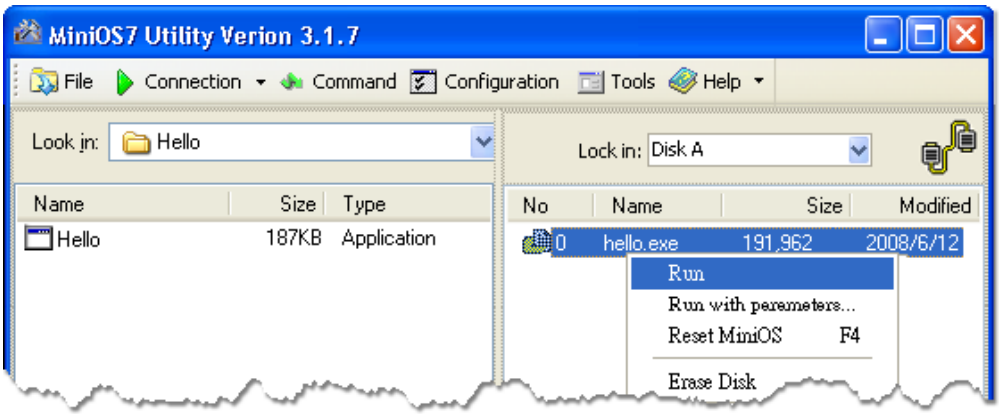
## 2.4.2. Uploading and executing µPAC-5001D-CAN2 programs

Before uploading and executing µPAC-5001D-CAN2 programs, you must firstly establish a connection between PC and µPAC-5001D-CAN2, for more detailed information about this process, please refer to section “2.4.1. Establishing a connection between PC and µPAC-5001D-CAN2”

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”






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

## 2.4.3. Making programs start automatically

After upload programs on the µPAC-5001D-CAN2, if you need programs to start automatically after the µPAC-5001D-CAN2 start-up, it is easy to achieve it to create a batch file called autoexec.bat and then upload it to the µPAC-5001D-CAN2, the program will start automatically in the next start-up.

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

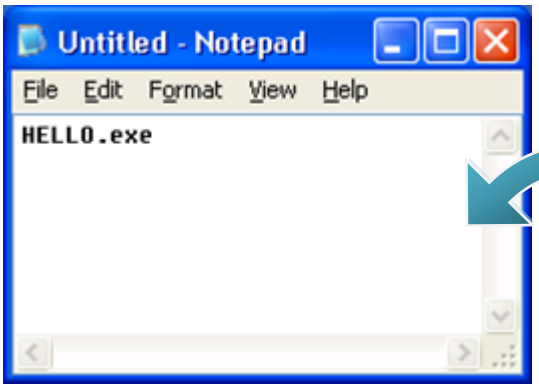
| Step 1: Create an autoexec.bat file |
|---|

i. Open the “Notepad”

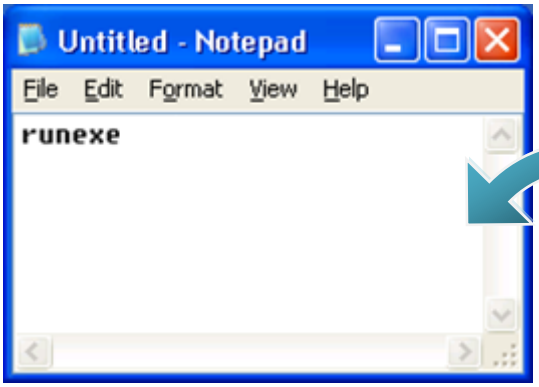
ii. Type the command

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 execuntion file (.exe).**

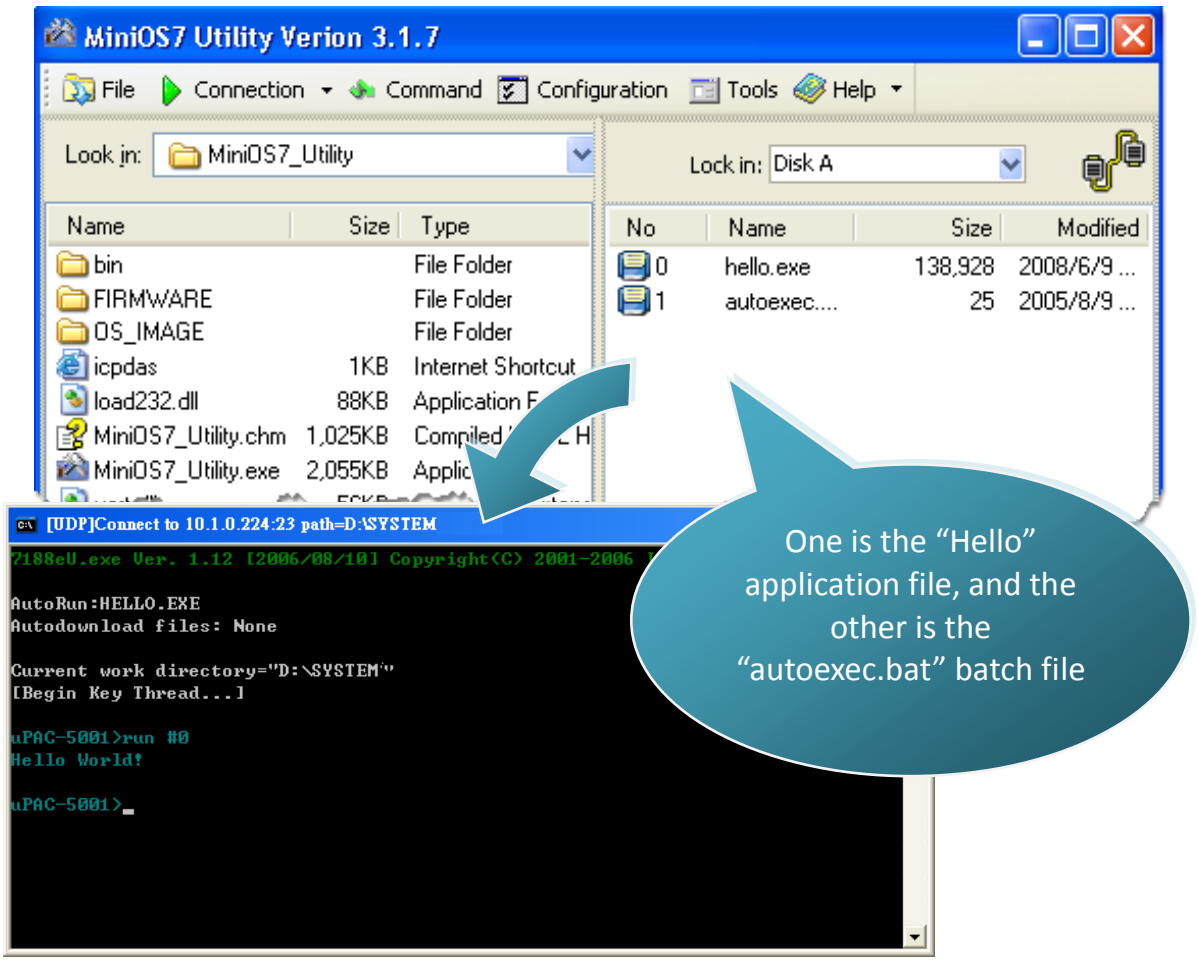
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

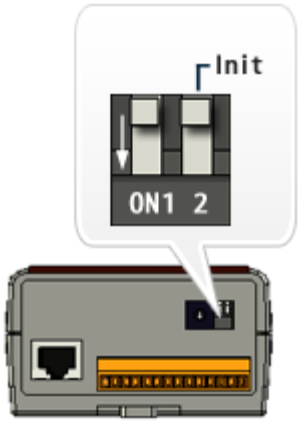
Step 2: Upload programs to µPAC-5001D-CAN2 using MiniOS7 Utility

For more detailed information about this process, please refer to section “2.4.1. Establishing a connection between PC and µPAC-5001D-CAN2”




## Tips & Warnings

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





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 2.5. Updating µPAC-5001D-CAN2 OS Image

ICP DAS will continue to add additional features to µPAC-5001D-CAN2 in the future, we advise you periodically check the ICP DAS web site for the latest update to µPAC-5001D-CAN2.

Step 1: Obtain the latest version of the µPAC-5001D-CAN2 OS image

The latest version of the µPAC-5001D-CAN2 OS image can be obtained from:
CD:\NAPDOS\upac-5000\OS_image\
http://ftp.Icpdas.com/pub/cd/8000cd/napdos/upac-5000/os_image/

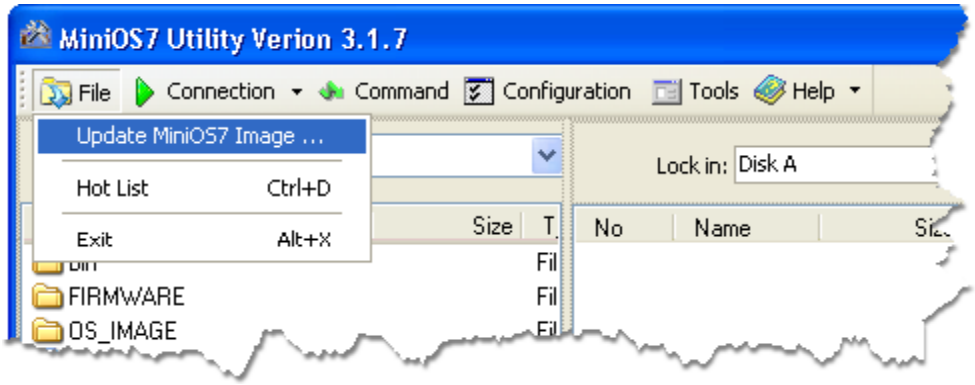



Step 2: Establish a connection

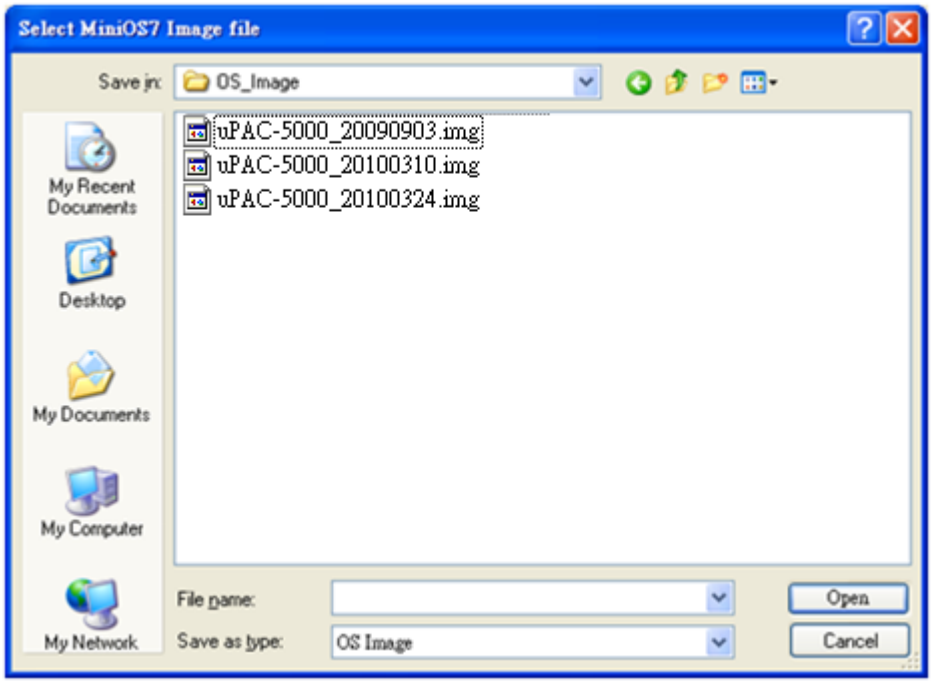
For more detailed information about this process, refer to section “2.4.1. Establishing a connection between PC and µPAC-5001D-CAN2”


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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



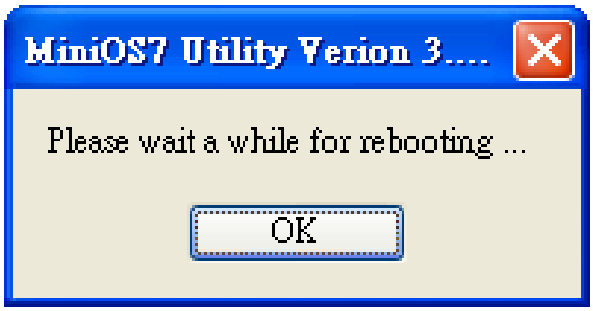

Step 4: Select the latest version of the MiniOS7 OS image



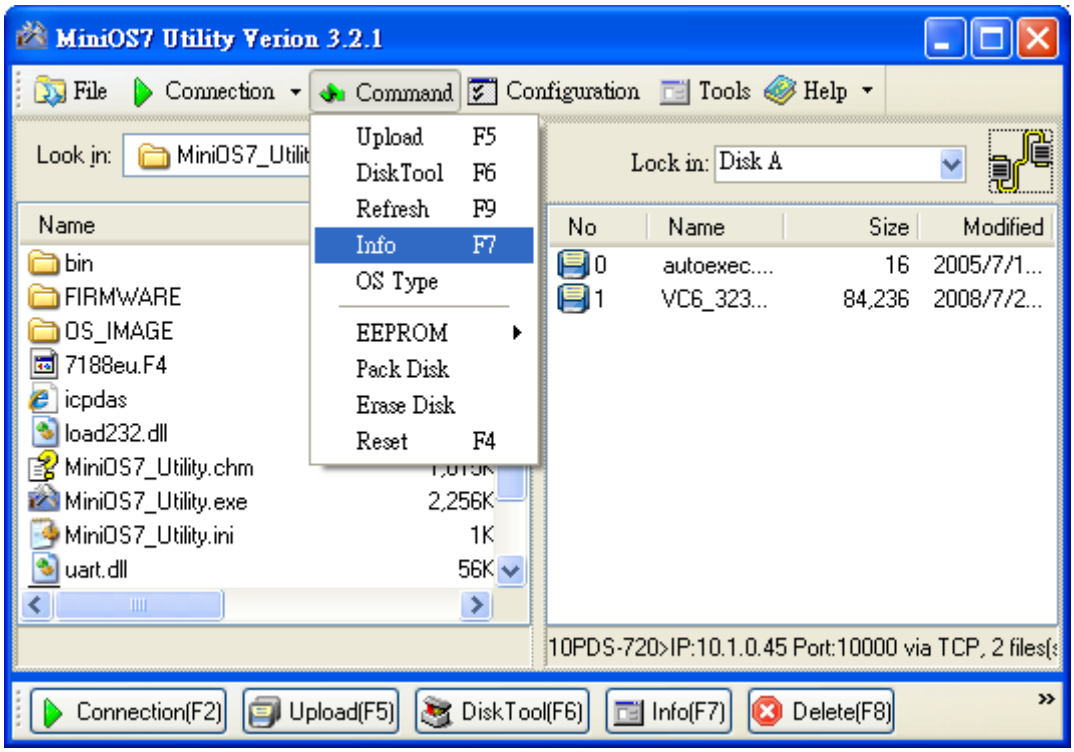



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Step 5: Click the “OK”




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






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 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 µPAC-5001D-CAN2’s program.



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 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 µPAC-5001D-CAN2's 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.).

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## 3.1.1. Installing the C compiler

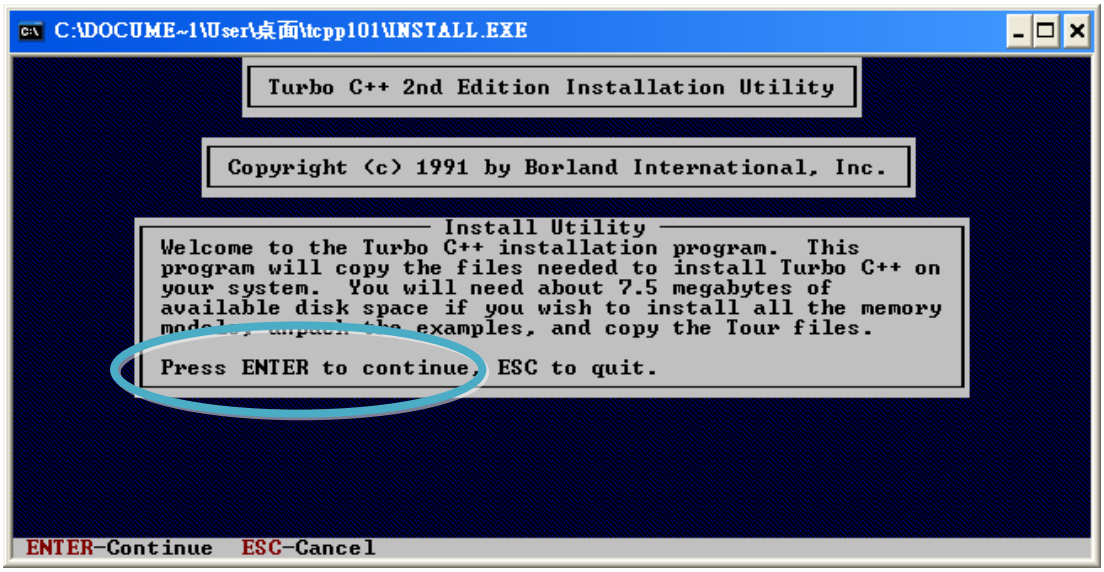
If there is no compiler currently installed on your system, installation of the compiler should be the first step.

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

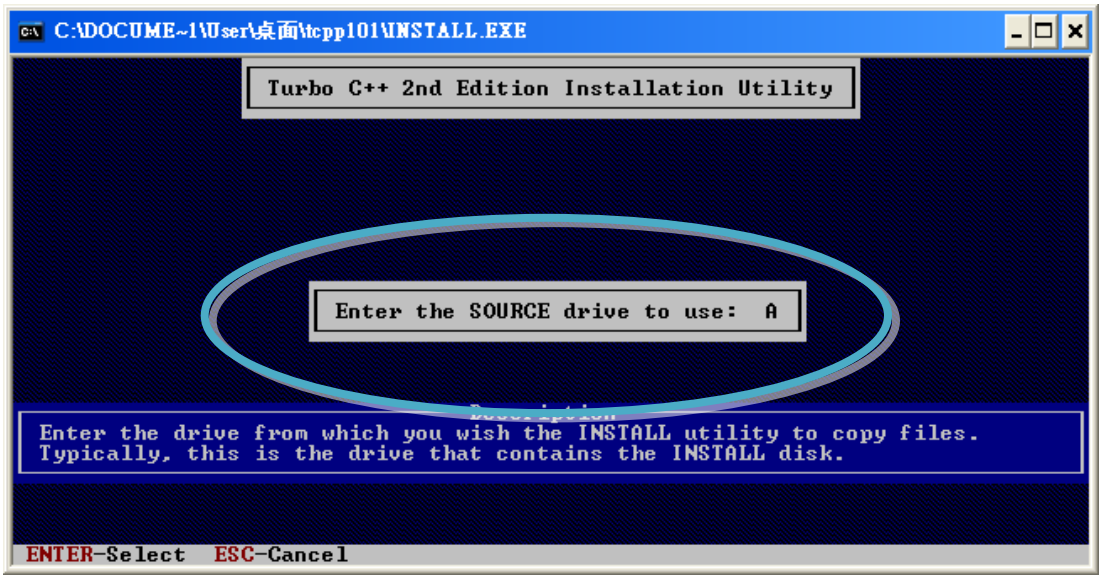




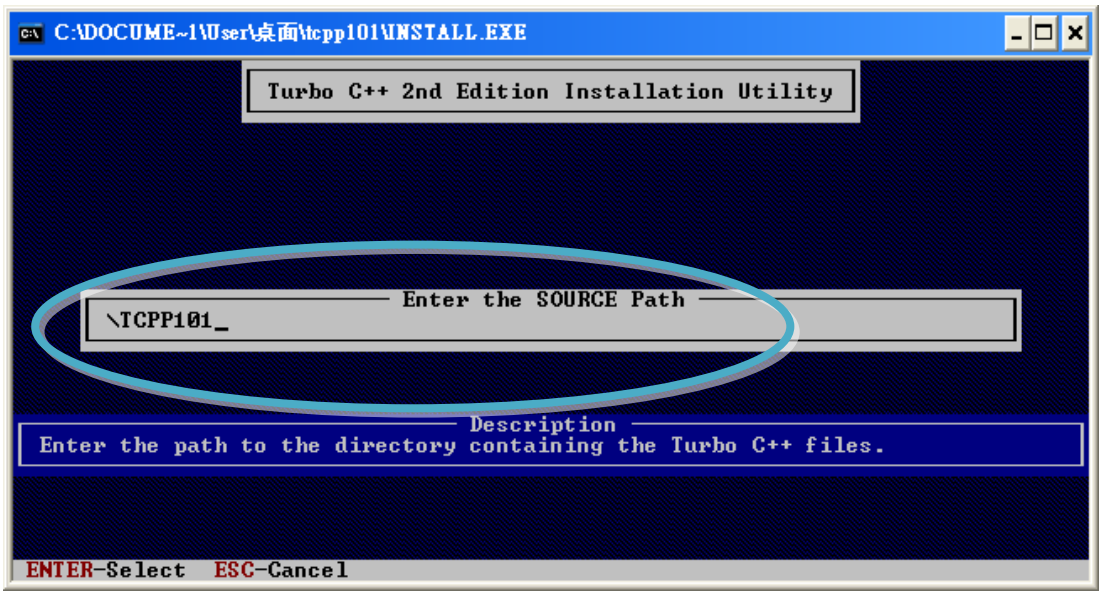

Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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

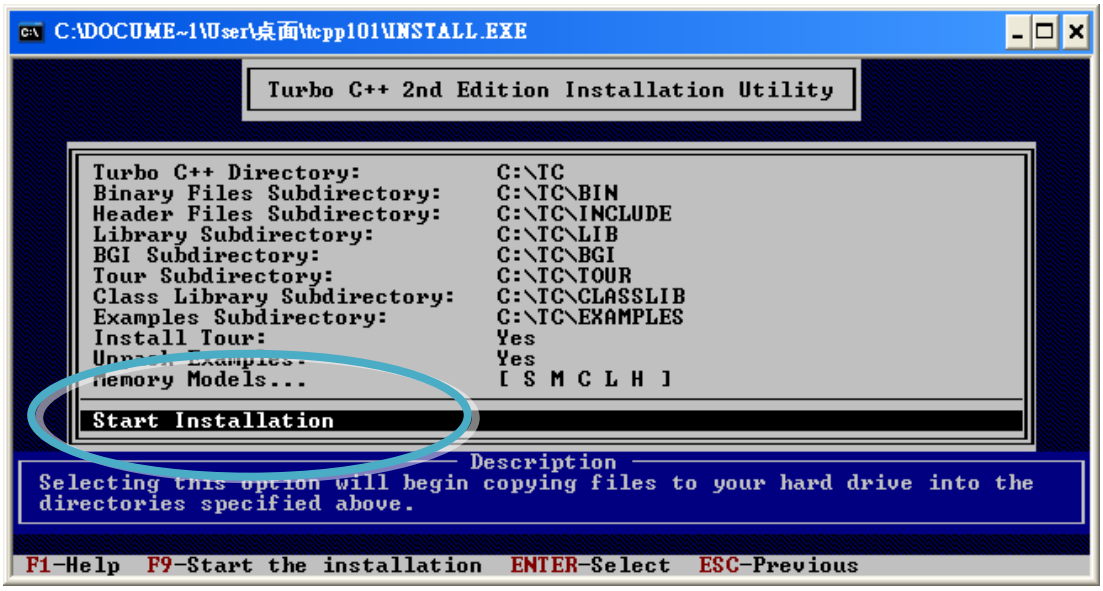




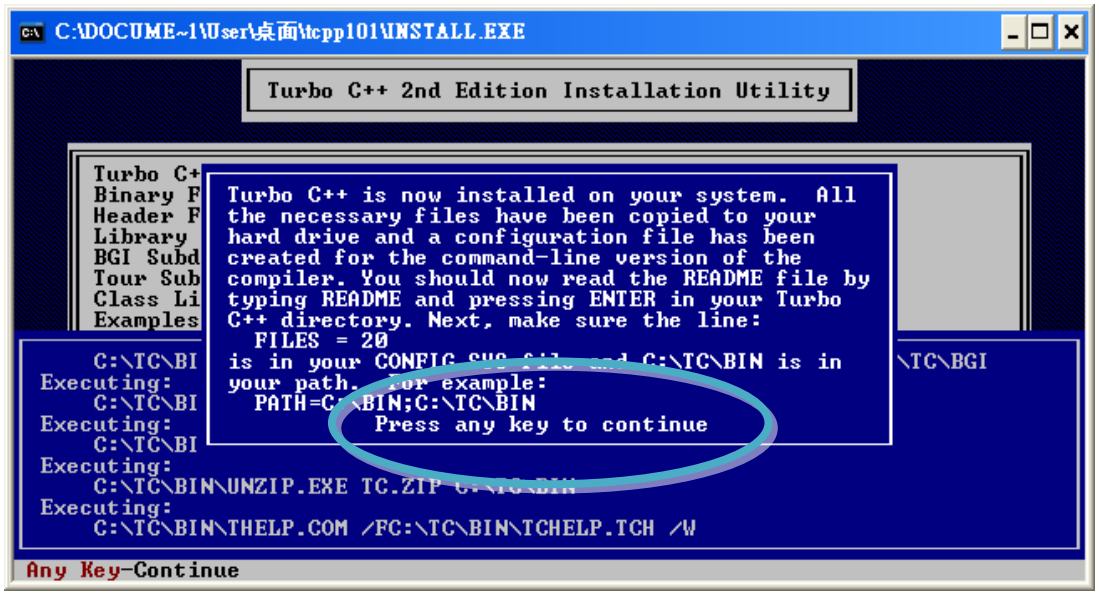


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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



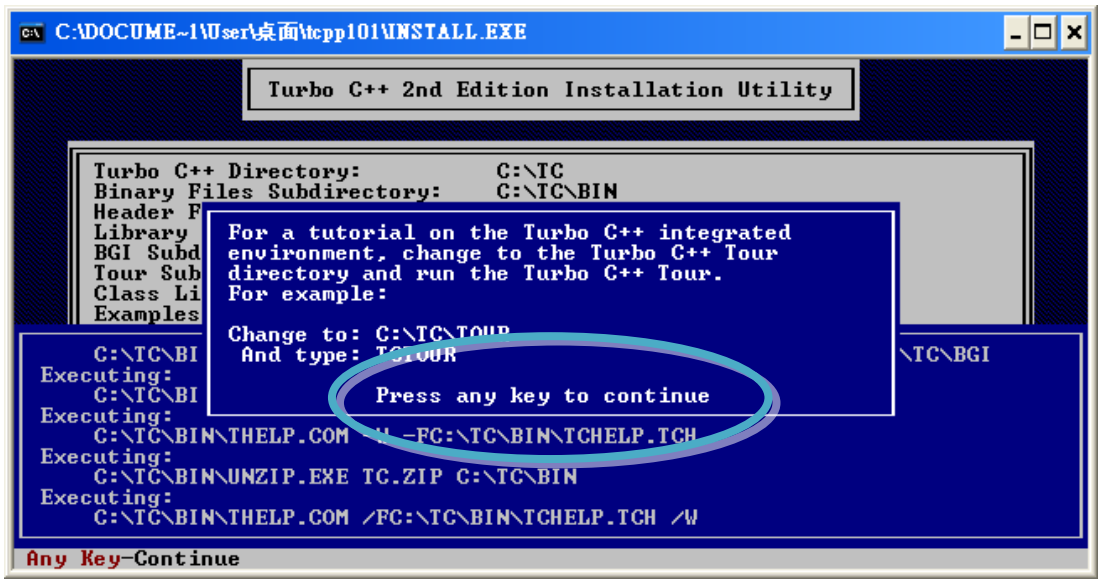

Step 6: Press any key to continue






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Step 7: Press any key to continue




Step 8: Installation is complete

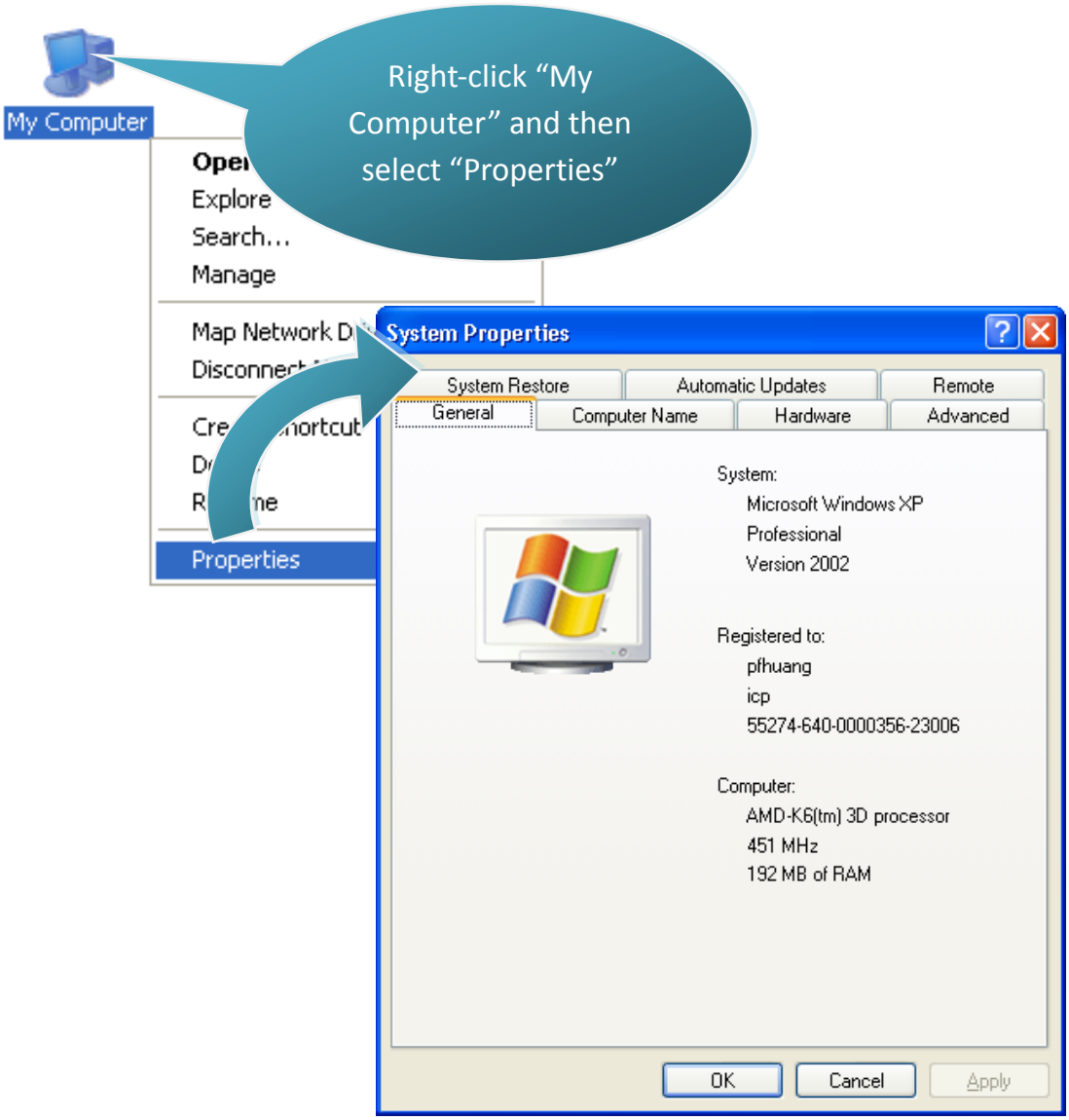



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## 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



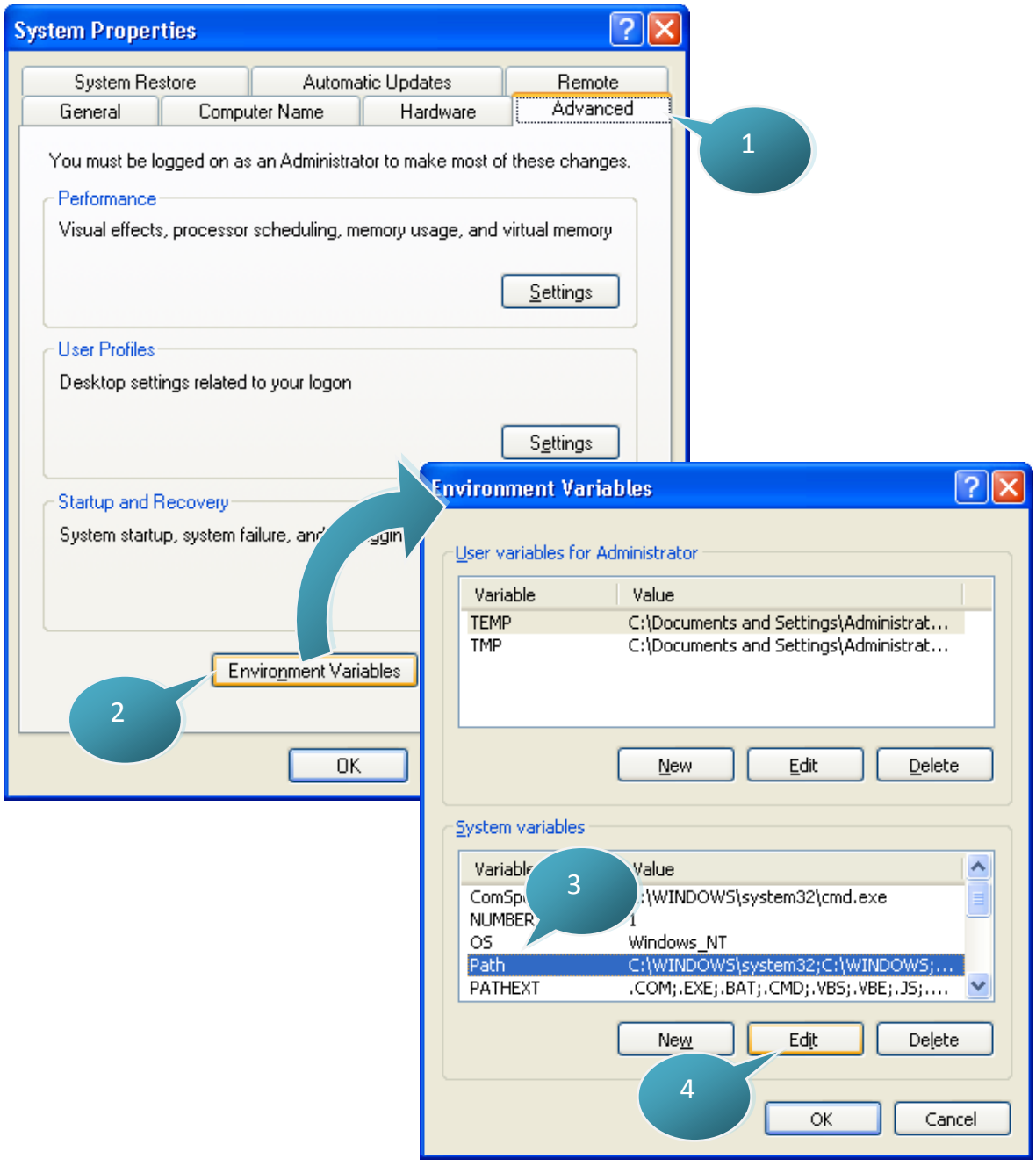



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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







Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## 3.2. µPAC-5001D-CAN2's APIs

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

For more detailed information regarding µPAC-5000 series standard APIs, please refer to following path about updating history files
CD:\NAPDOS\upac-5000\Demo\basic\Lib\
http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/lib/

For more detailed information regarding µPAC-5001D-CAN2 CAN bus APIs, please refer to
CD:\ fieldbus_cd\can\pac\upac-5001D-CAN\demo\bc_tc\lib\
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pac/upac-5001D-CAN/demo/bc_tc \lib\

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


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

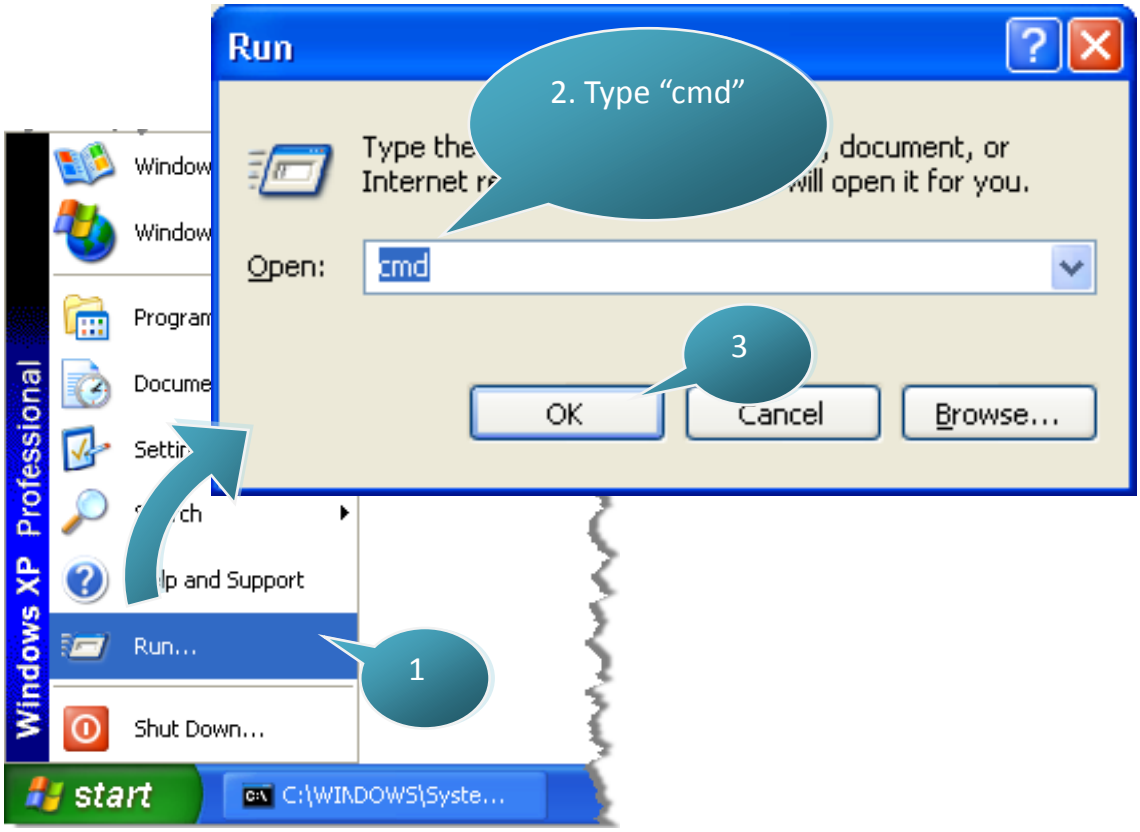
# 3.3. First Program in µPAC-5001D-CAN2

Here we assume you have installed the Turbo C++ 1.01 (as the section “3.1. C Compiler Installation”) and the µPAC-5001D-CAN2's 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

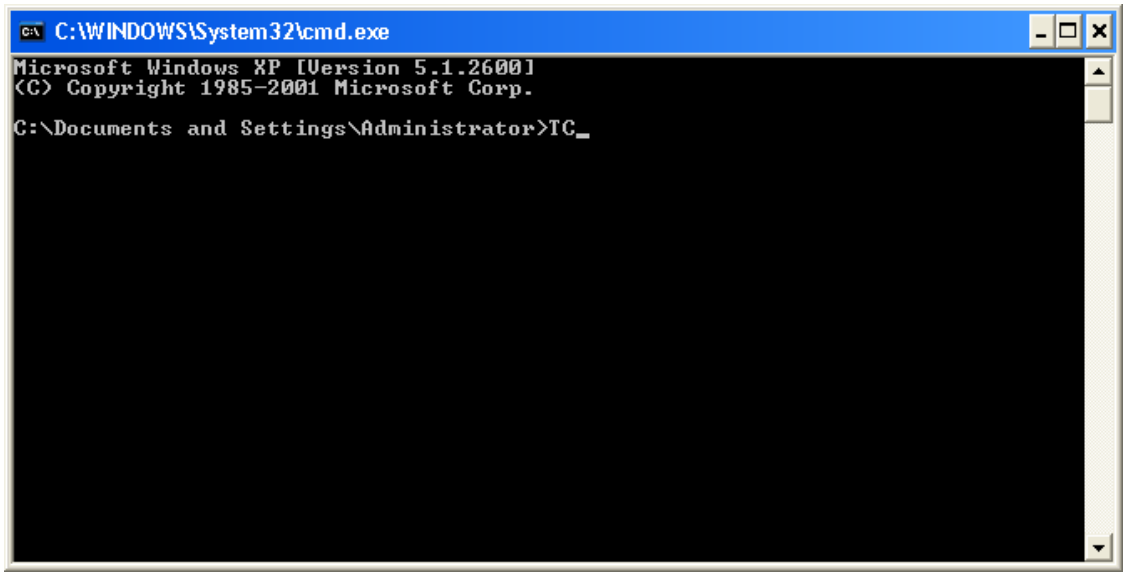




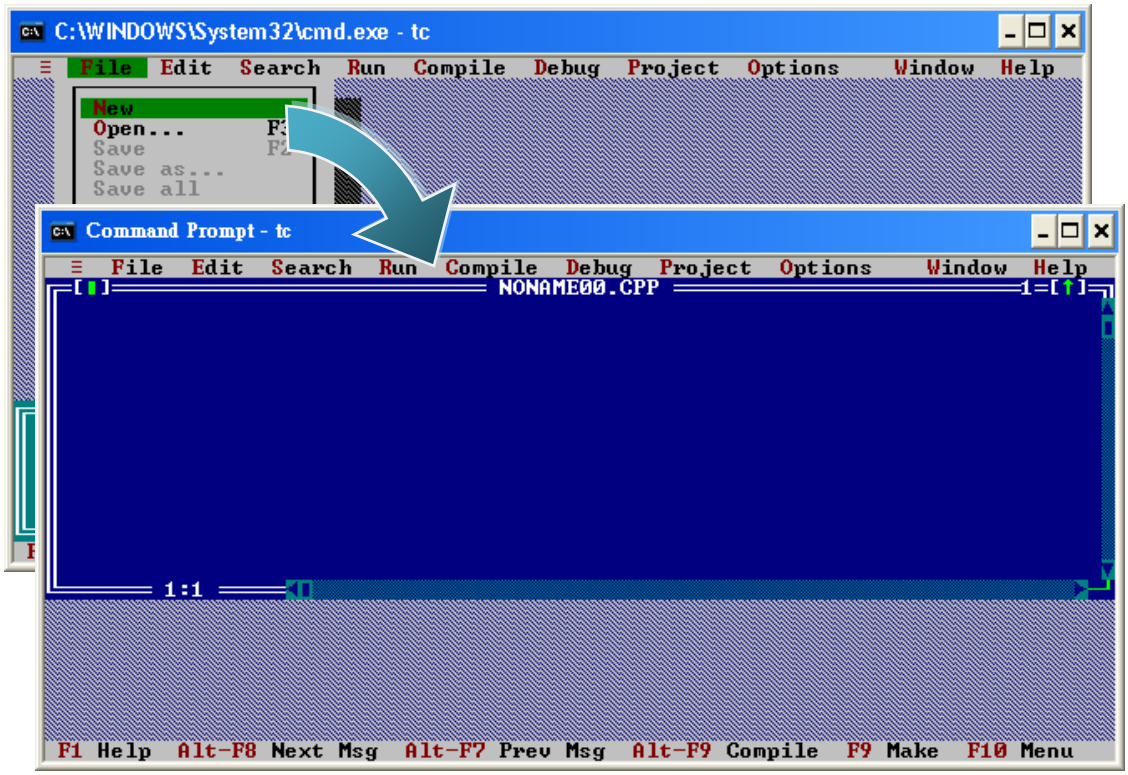

Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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 */

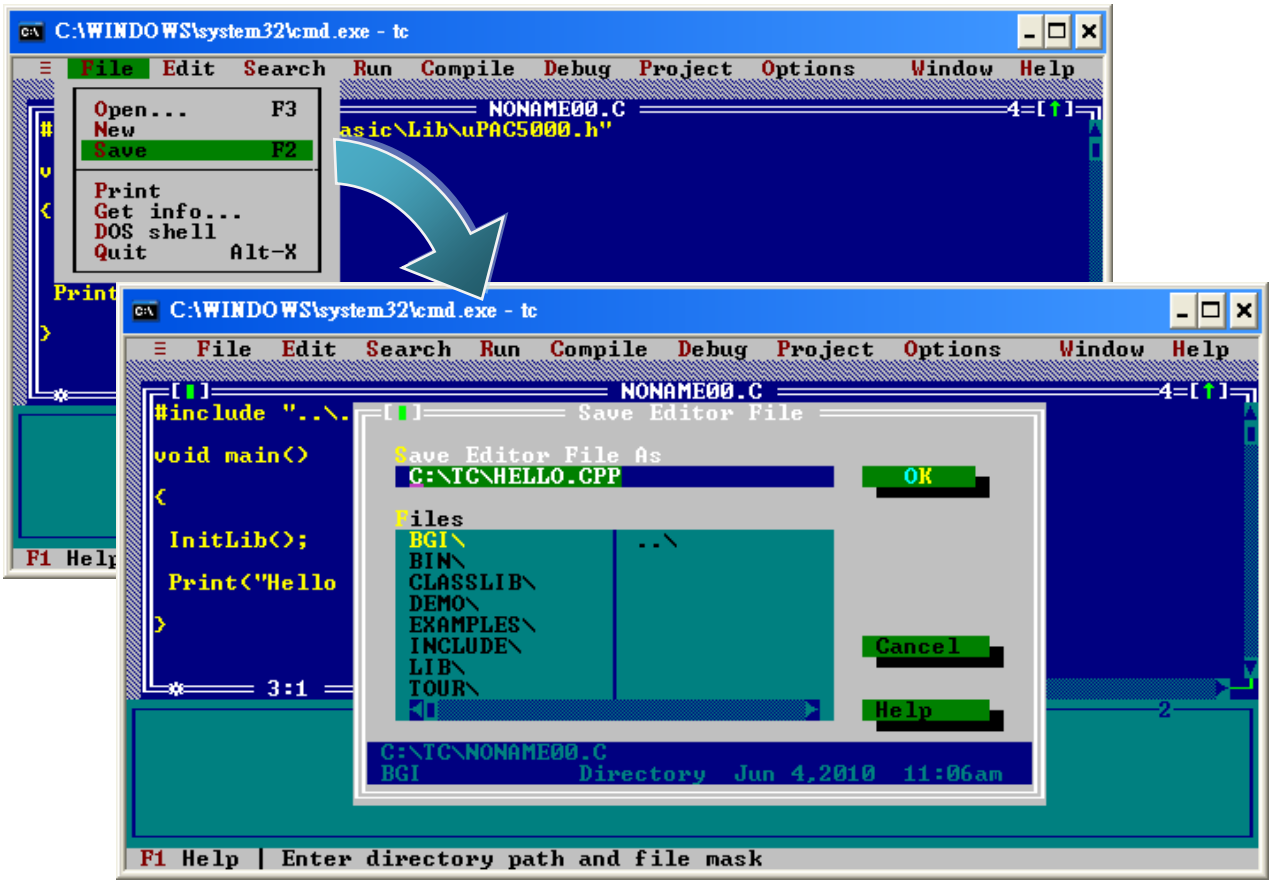
   Print("Hello world\r\n"); /* Print the message on the screen */
}
```

Step 5: Save the source file

i. Select “Save” from the “File” menu

ii. Type the file name “Hello”

iii. Select “OK”







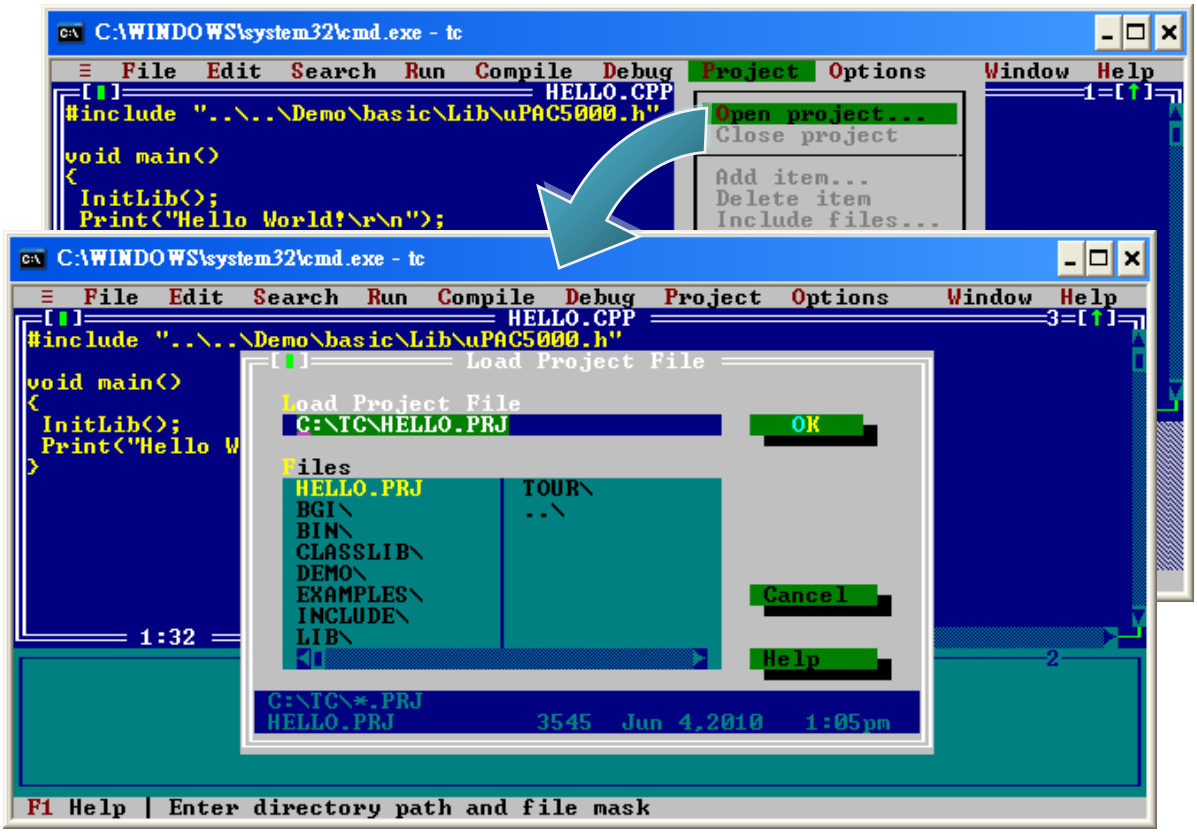
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## 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”

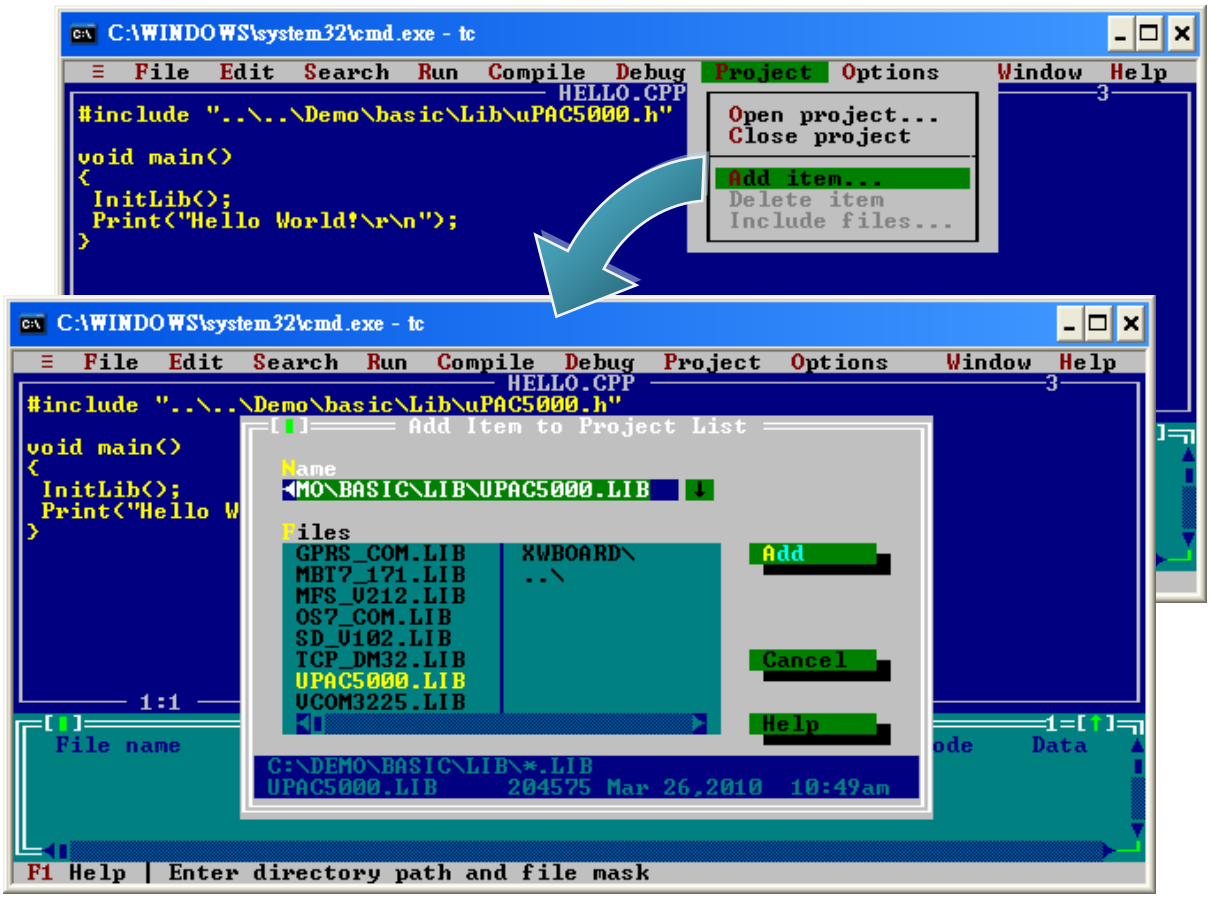





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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. Slelect “Done” to exit



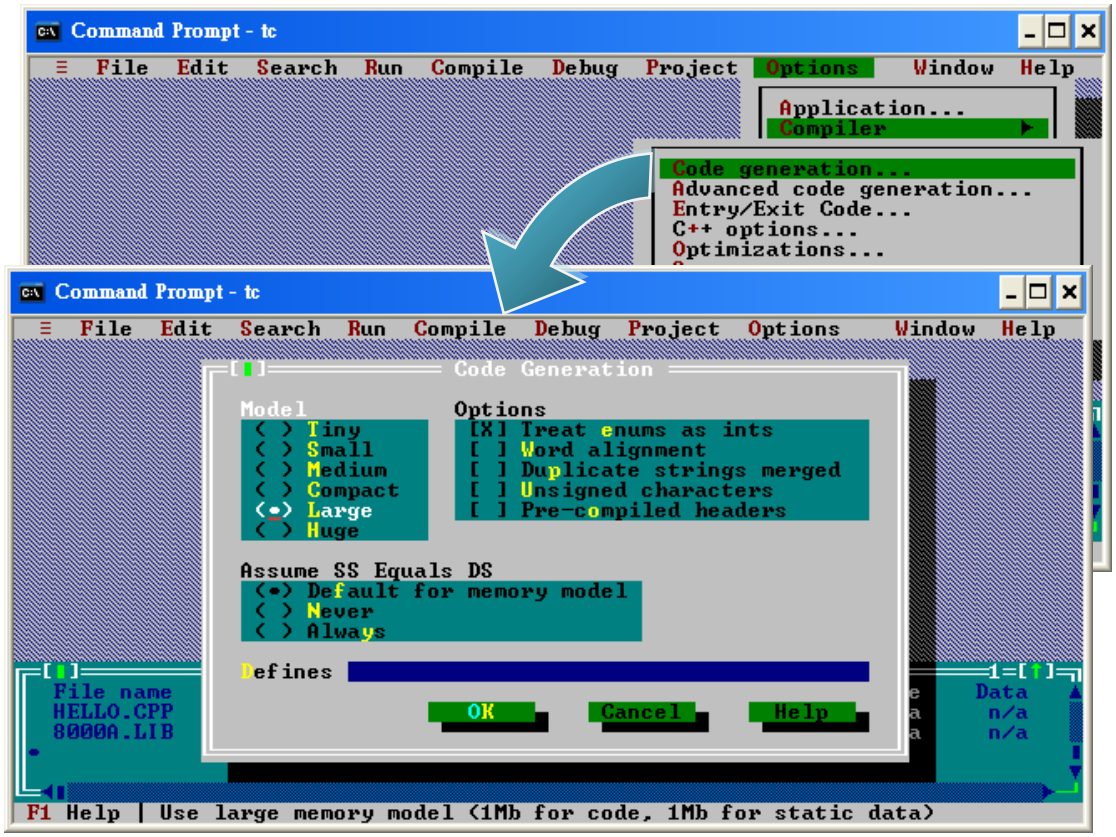

Step 8: Add the source file (*.c) to the project

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. Slelect “Done” to exit


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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”

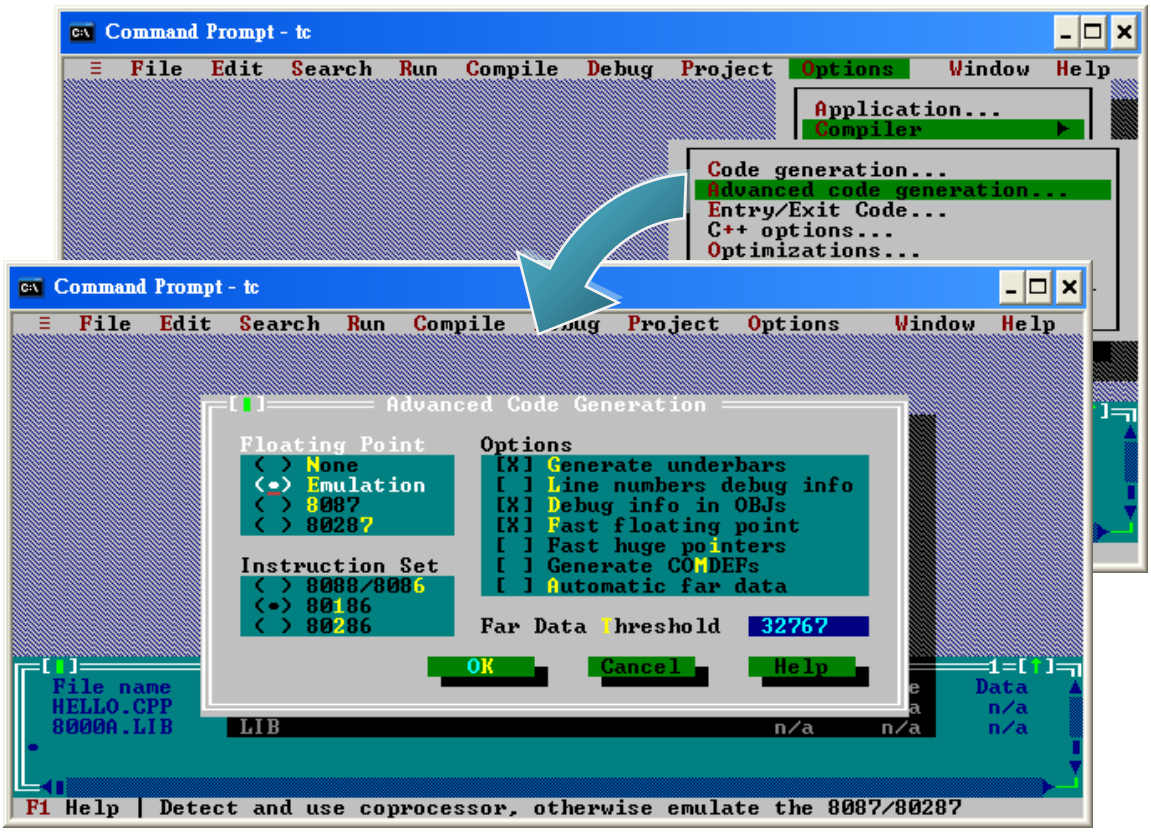





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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”

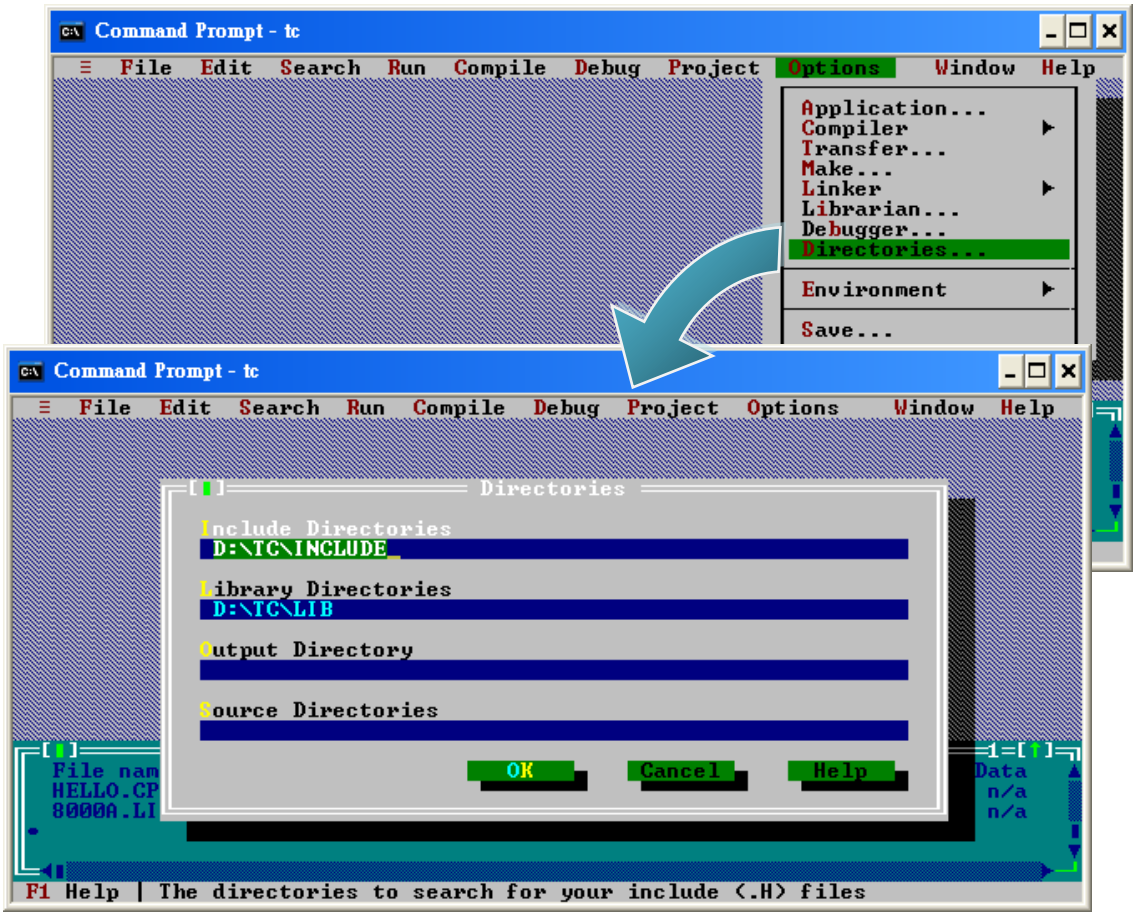





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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

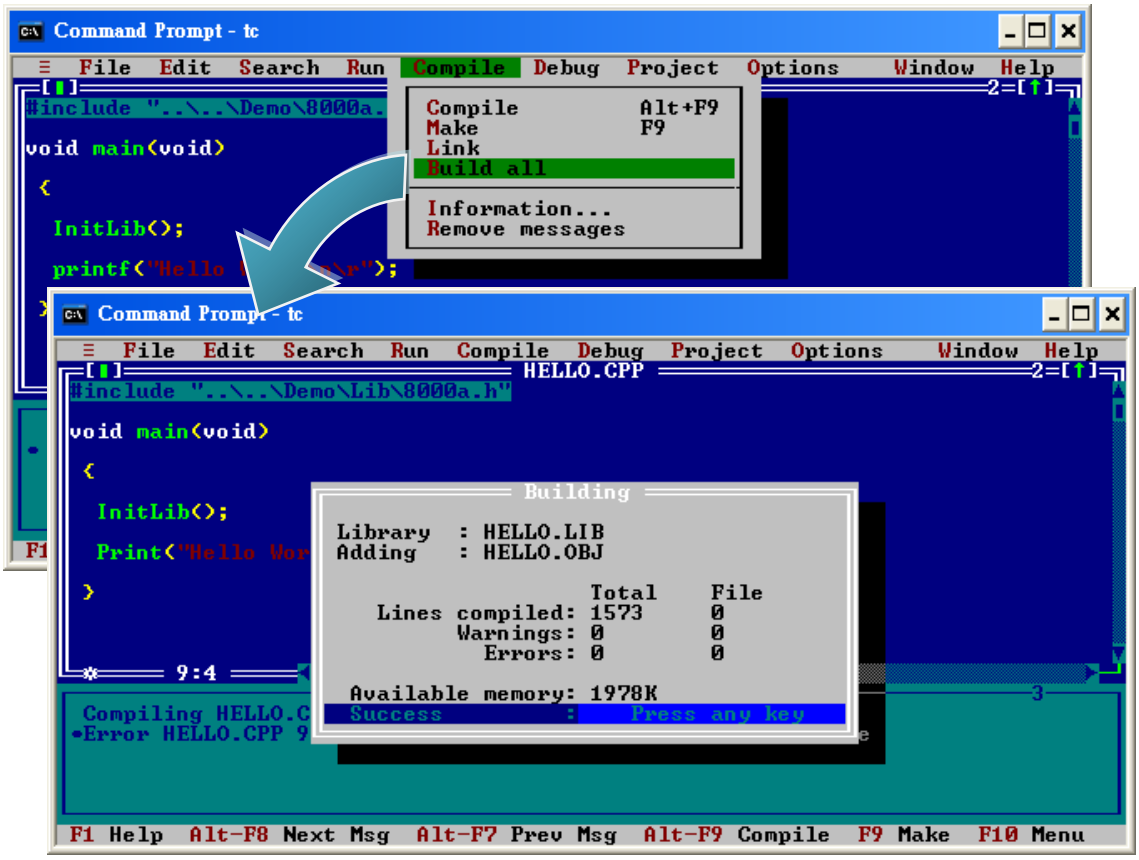
i. Select “Directories…” from the “Options” menu
ii. On “Include Directories” option, specify the header file directory
iii. On “Library Directories” option, specify the function library directory
iv. Select “OK”





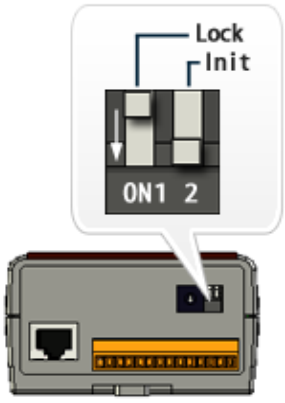
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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.

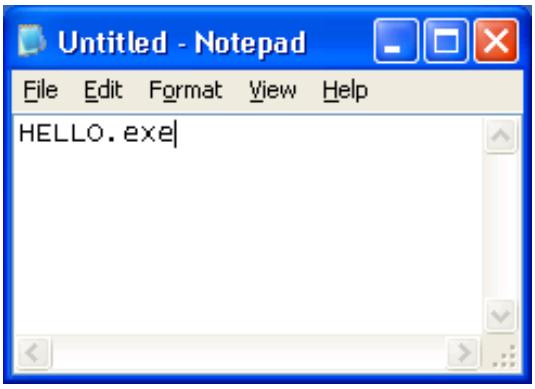



Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

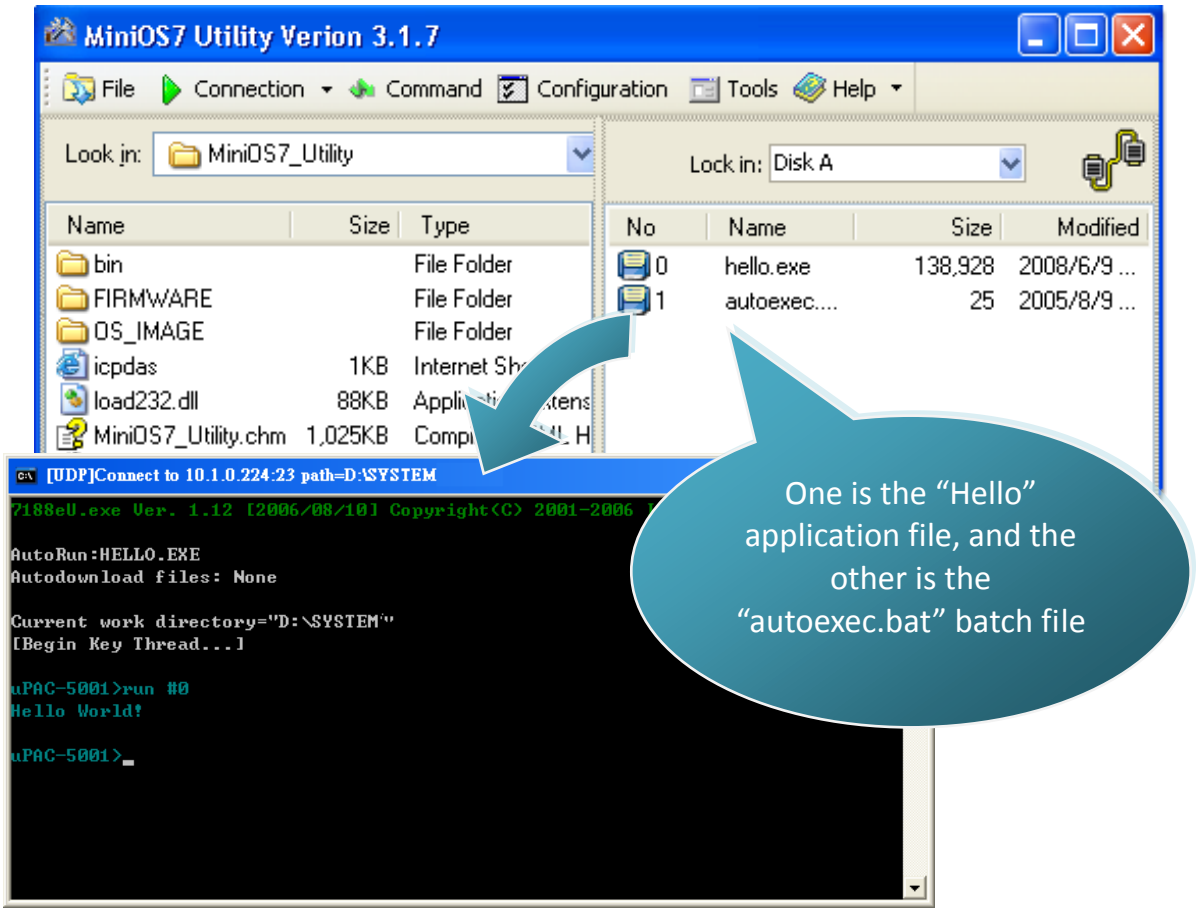
Step 14: Create an autoexec.bat file




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

Step 15: Upload programs to µPAC-5001D-CAN2 using MiniOS7 Utility

For more detailed information about this process, please refer to section “2.4.1. Establishing a connection between PC and µPAC-5001D-CAN2”





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4. APIs and Demo References

There are several APIs and demo programs that have been designed for µPAC-5001D-CAN2.
You can examine the APIs and demo source code, which includes numerous functions and comments, to familiarize yourself with the MiniOS7 APIs and 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 |
| Xserver | VxComm.h | Vcom3225.Lib |
| Modbus driver | MBTCP.h | MBT7_171.lib |
| Xboard Driver for CAN BUS | XWCAN.h | XWCAN.lib |

For more detailed information regarding µPAC-5000 standard APIs, please refer to following path about updating history file:
CD:\NAPDOS\upac-5000\Demo\basic\Lib\
http://ftp.icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/lib/

For more detailed information regarding µPAC-5001D-CAN2 CAN bus APIs, please refer to:
CD:\fieldbus_cd\can\pac\upac-5001D-CAN\demo\bc_tc\lib\
ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pac/upac-5001-CAN/demo/bc_tc/lib/

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

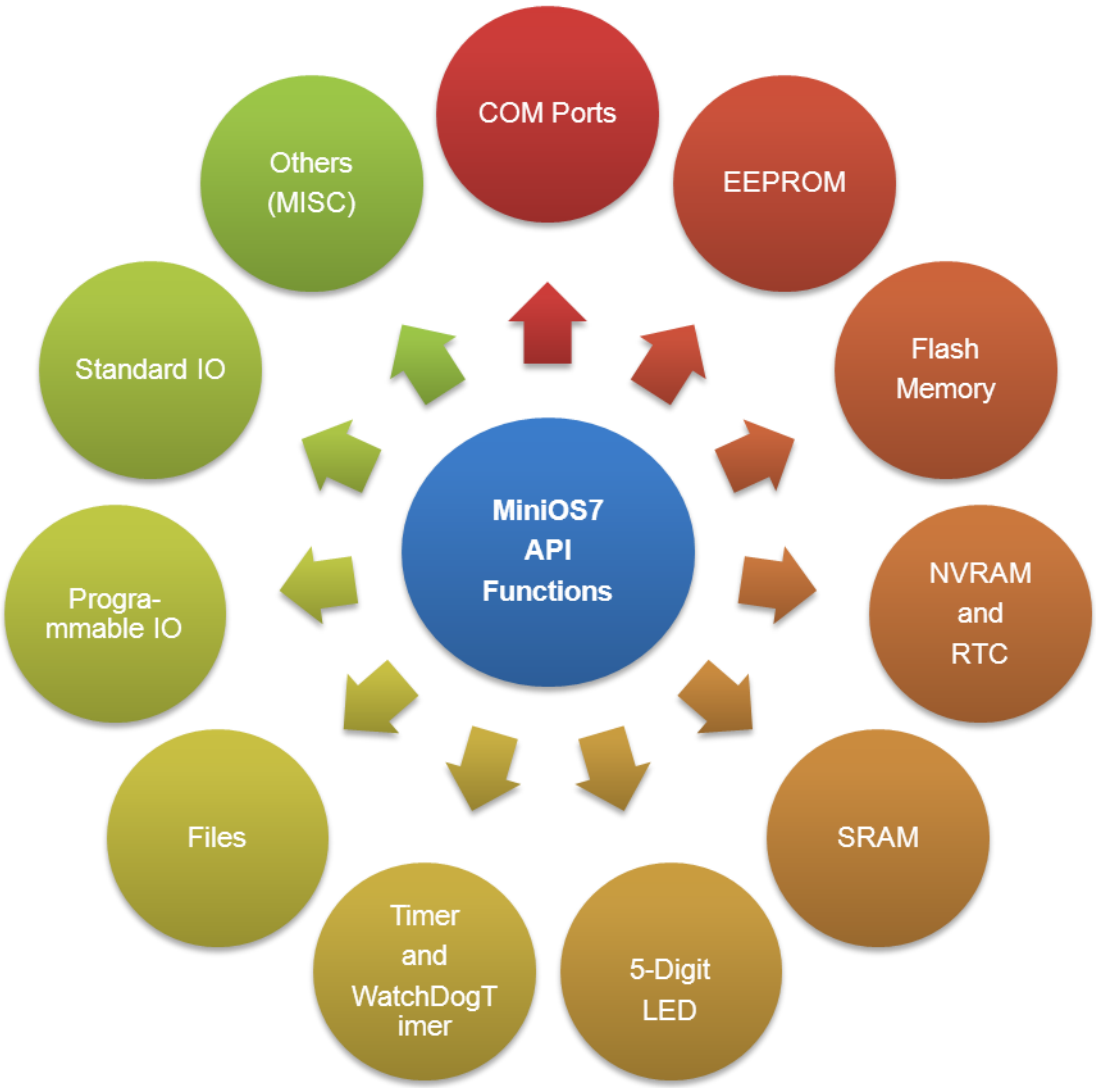
The following introduces the core API, MiniOS7 API, which is integrated into the µPAC-5001D-CAN2 API set.

Functions Library ─ uPAC5000.lib

This file contains the MiniOS7 API (Application Programming Interface) and has hundreds of pre-defined functions related to µPAC-5000.

Header File ─ uPAC5000.h

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



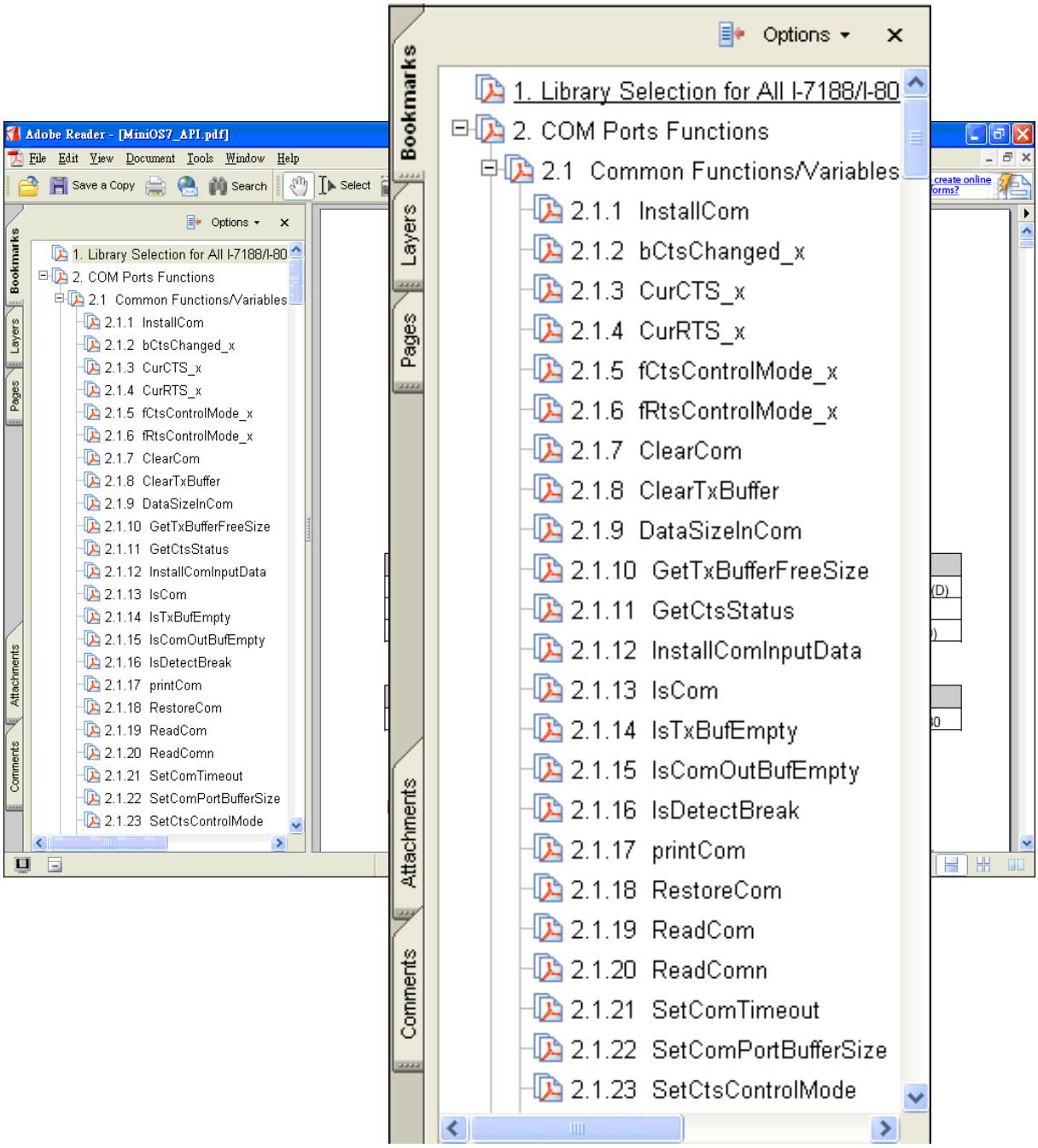



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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/




LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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

| Folder | Demo | Explanation |
|---|---|---|
| LED | LED | Shows how to control the LED display. |
| | Seg7led | Shows how to control the 5-digit 7-segment LED display. |
| MISC | Rotary_Switch | Shows how to read the position of the switch. |
| Memory | EEPROM | Shows how to read/write EEPROM. |
| | Flash | Shows how to access Flash |
| | Nvram-r,Nvram-w | Shows how to read/write Nvram. |
| microSD | sd_qa | Shows how to connect and control the microSD |
| | sd_read | |
| | sd_util | |
| | sd_write | |
| Timer | Demo90,demo91,…,demo99 | Shows how to use timer function. |
| 7k87k_Module | 7k87k_ai_for_com,<br>7k87k_demo_for_com,<br>…<br>Ao_24_for_com | Shows how to control 7k and 87k module. |

For more detailed information regarding µPAC-5000 APIs, please refer to:
CD:\NAPDOS\upac-5000\Demo\basic\
http://ftp.Icpdas.com/pub/cd/8000cd/napdos/upac-5000/demo/basic/


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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

| Folder | Demo | Explanation |
|---|---|---|
| **Demo** | AC_AM | Use the AccCode and AccMask |
| | Dual_Filter | Use the AccCode and AccMask to set dual filter |
| | All_Demo | Demo the total functions provided by the XWCAN.lib. |
| | L1_L2_L3 | Use the CL1, CL2, and CL3 LEDs. |
| | RxInt | Receive the CAN messages by interrupt mode. |
| | RxPoll | Receive the CAN messages by polling mode. |
| | TxInt | Send the CAN messages to the CAN network by interrupt mode. |
| | TxPoll | Send the CAN messages to the CAN network by polling mode. |
| | UserInt | Use the UserCANInt function to apply the users' CAN interrupt service routine. |
| | SCH_Baud | Demo for search the CAN bus baud rate. |

For more detailed information regarding µPAC-5001D-CAN2 CAN bus APIs, please refer to:

CD:\fieldbus_cd\can\pac\upac-5001D-CAN\demo\bc_tc\lib\

ftp://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pac/upac-5001D-CAN/demo/bc_tc/lib/

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

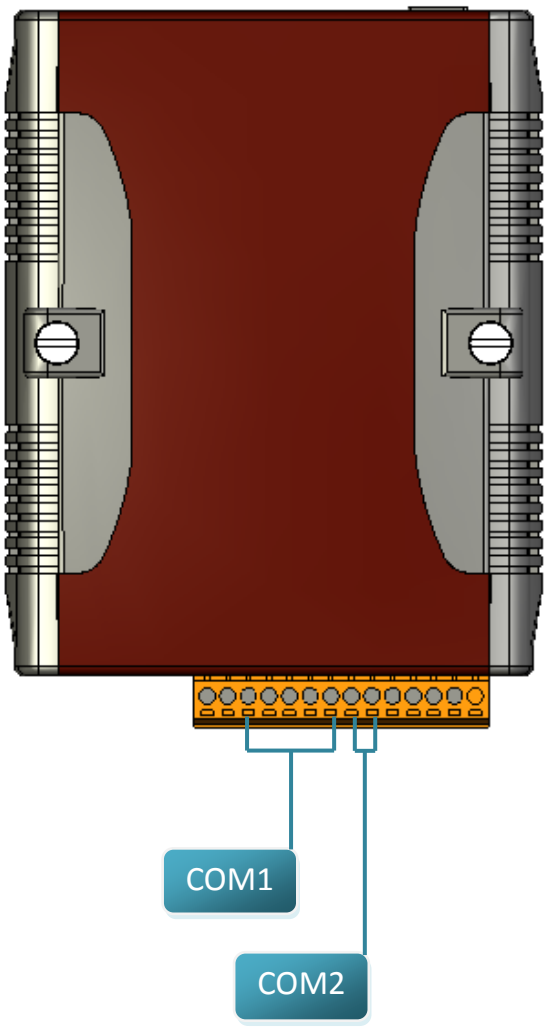


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.1. API for COM Port

The µPAC-5001D-CAN2 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.




LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 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 | | | | |
|---|---|---|---|---|---|---|---|
| Mini OS7 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()<br>Putch() | Getch() | Print() |



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

## 4.1.2. API for MiniOS7 COM port

The software driver for the uPAC-5001D-CAN2 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()



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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 upac-5000 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 */
}
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

### 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 upac-5000 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);
}
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

## 4.1.3. API for standard COM port

The standard COM port is used to upload program from PC to the µPAC-5001D-CAN2.

### 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. Kbhit()

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

API for reading data from standard I/O port

2. Getch()

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

API for sending data to standard I/O port

3. Puts() – For sending a string

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

4. Putch( ) – For sending one character

Outputs a character to the COM1.



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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 upac-5000 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 upac-5000 library */

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

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## 4.1.4. Port functions Comparison

For example, learning to show the ASCII code.

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

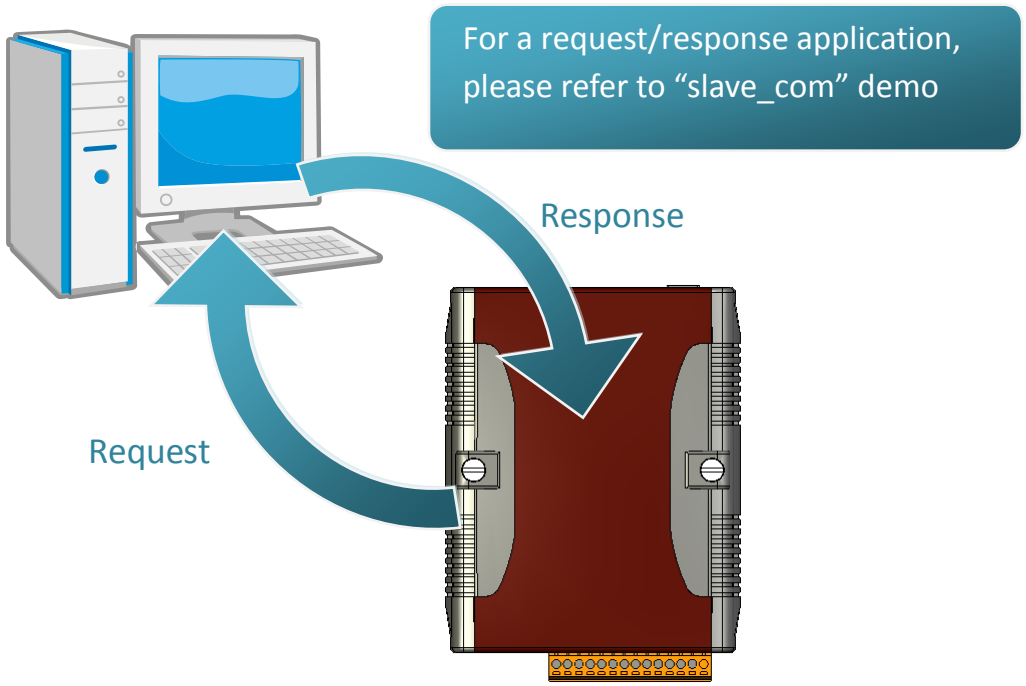


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

## 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<br>Command1 |
| c2 | Debug information: Command2<br>Command2 |
| Q | Debug information: Quick program |
| Other command | Debug information: Unknown command |

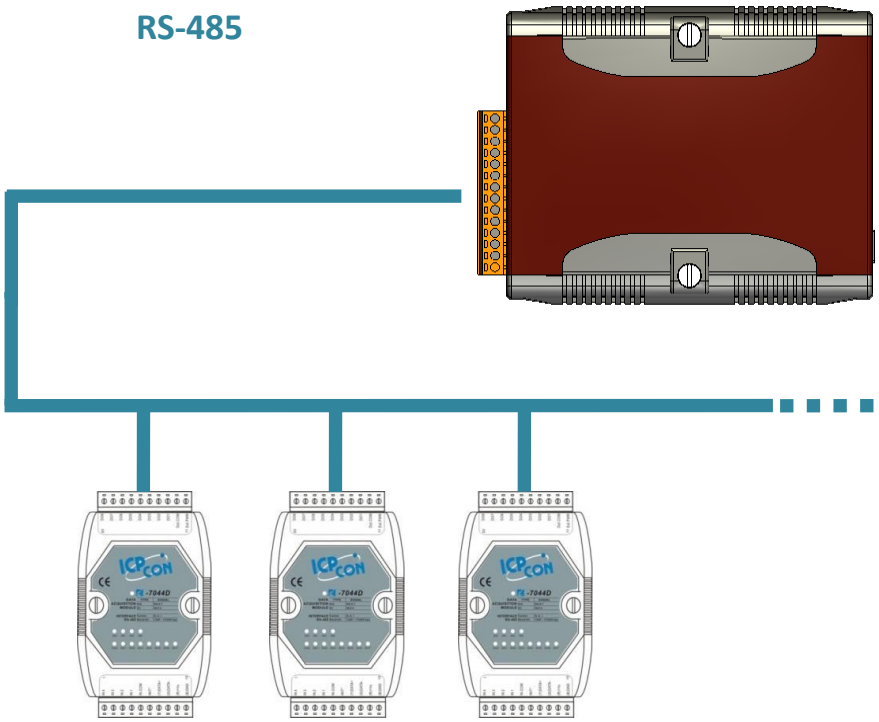


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

# 4.2. API for I/O Modules

The µPAC-5001D-CAN2 equips 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

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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 upac-5000 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 */
}
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

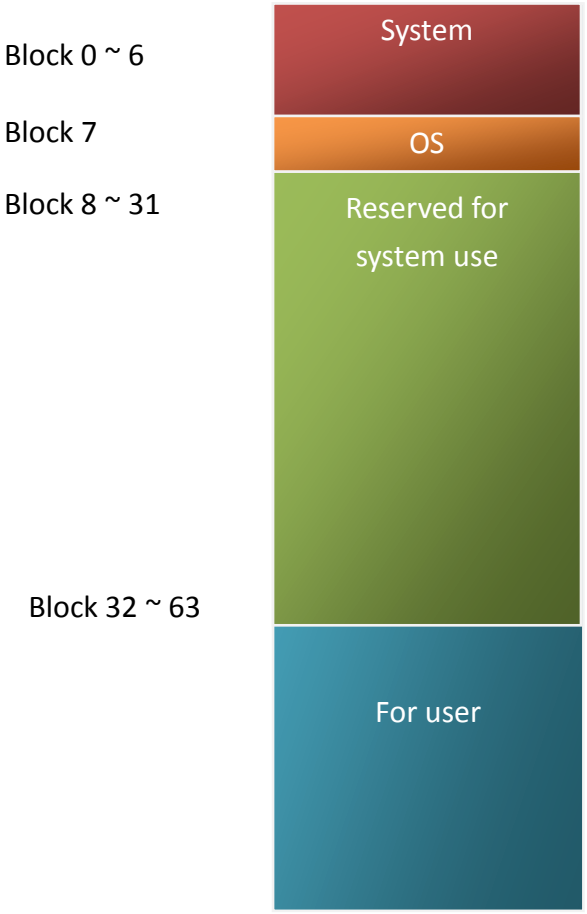


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.3. API for EEPROM

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

API for writing data to the EEPROM




1. EE_WriteEnable()

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

2. EE_WriteProtect()

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

3. EE_MultiWrite()

After using the EE_WriteEnable() to write-enable EEPROM, the EE_MultiWrite()must be called to write the data.

API for reading data from the EEPROM

4. EE_MultiRead()

Read data from the EEPROM no matter what the current mode is.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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 upac-5000 library */

   EE_WriteEnable();
   EE_MultiWrite(1, 10, 1, &data);
   EE_WriteProtect();

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



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.4. API for Flash Memory

- The µPAC-5001D-CAN2 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.


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## API for writing data to 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 segnment 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 upac-5000 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) */
   dataprt=(char *) &data2;
   *dataptr=FlashRead(0xd000, 0x1234);
   *(dataptr+1)=FlashRead(0xd000, 0x1235);

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

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.5. API for NVRAM

- The µPAC-5001D-CAN2 equips 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 upac-5000 library */

   WriteNVRAM(0, data);
   data2=ReadNVRAM(0); /* Now data2=data=0x55 */
}
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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 upac-5000 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 */
}
```

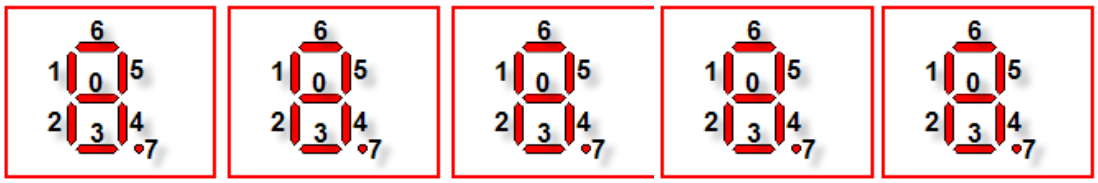
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.6. API for 5-Digital LED

The µPAC-5001D-CAN2 contains a 5-Digit 7-SEG LED with a decimal point on the right-hand side of each digit, which be used to display numbers, IP addresses, time, and so on.




API for starting the 5-Digit 7-SEG LED

1. Init5DigitLed()

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

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

2. Show5DigitLed()

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

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

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

void main(void)
{
   InitLib(); /* Initiate the upac-5000 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 */
}
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.7. API for Timer

- The µPAC-5001D-CAN2 can support a single main time tick, 8 stop watch timers and 8 counts down timers.
- The µPAC-5001D-CAN2 use 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.



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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 upac-5000 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 */
}
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.8. API for WatchDog Timer (WDT)

- The µPAC-5001D-CAN2 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 upac-5000 library */

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

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.9. API for microSD



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

The µPAC-5001D-CAN2 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<br>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 |



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

| 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 miscroSD functions, PC_Init() must be called to initialize the microSD.

## API for enabling/disabling microSD

2. pc_open()

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

3. 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

4. pc_write()

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



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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

### 5. 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:



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
#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\uPAC-5000\Demo\Basic\microSD\

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



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10. API for CAN bus

| Function | Description |
|---|---|
| CAN_Reset | CAN controller hardware reset |
| XWCANInit | Initialize the CAN hardware |
| SetCANBaud | Change CAN baud |
| SetCANMask | Change CAN message filter |
| CAN_InstallIrq | Enable the embedded controller interrupt |
| CAN_RemoveIrq | Disable the embedded controller interrupt |
| CAN_Restore | Release the resource and disable the embed controller interrupt |
| CAN_CreateBuffer | Change the reception and transmission buffer sizes |
| SendCANMsg | Send a CAN message to the CAN network |
| GetCANMsg | Receive a CAN message |
| GetStatus | Obtain the CAN controller status and reception/transmission buffer status |
| ClearStatus | Reset the reception and transmission buffer status |
| CL1Off | Turn LED (CL1) off |
| CL2Off | Turn LED (CL2) off |
| CL3Off | Turn LED (CL3) off |
| CL1On | Turn LED (CL1) on |
| CL2On | Turn LED (CL2) on |
| CL3On | Turn LED (CL3) on |
| UserCANInt | Design user-defined interrupt routine |
| CAN_SearchBaud | Search the necessary CAN Bus baud rate |

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.1. API for CAN Initialization

API for CAN Reset

**1. CAN_Reset( int CANPort )**

Reset the CAN controller by hardware circuit. After running this function, the CAN controller will be set to initial state. For more information about this, please refer to the SJA1000 data sheet on the web site.

http://www.semiconductors.philips.com/pip/SJA1000.html#datasheet

Parameter:

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

API for CAN Initialization

**2. XWCANInit( int CANPort,char IntMode,unsigned long CANBaud,**
**char BT0, char BT1,unsigned long AccCode,unsigned long AccMask )**

Initialize the software buffer and XW-CAN 200 hardware, which includes CAN controller, LED1, LED2 and LED3.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

IntMode: CAN controller interrupt mode. Each bit of IntMode parameter indicates different function shown as follows.

| Interrupt Type | Value of IntMode |
|---|---|
| Receive Interrupt Enable | 0x01 |
| Transmit Interrupt Enable | 0x02 |
| Error Warning Interrupt Enable | 0x04 |
| Data Overrun Interrupt Enable | 0x08 |
| Wake-up Interrupt Enable | 0x10 |
| Error Passive Interrupt Enable | 0x20 |
| Arbitration Lost Interrupt Enable | 0x40 |
| Bus Error Interrupt Enable | 0x80 |

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

| Interrupt Type | Meaning |
|---|---|
| Receive Interrupt | When a message has been received without errors, the receive interrupt will be triggered. |
| Transmit Interrupt | When a message has been successfully transmitted or the transmit buffer is accessible again, the transmit interrupt will be triggered. |
| Error Warning Interrupt | If the error or bus status is set or clear, the error interrupt will be triggered. |
| Data Overrun Interrupt | If a message was lost because there was not enough space for that message in the FIFO (FIFO has 64 bytes), the overrun interrupt will be triggered. |
| Wake-up Interrupt | When the CAN controller is sleeping and bus activity is detected. The Wake-up interrupt will be triggered. |
| Error Passive Interrupt | If CAN controller has at least one error counter exceeds the protocol-defined level of 127 or if the CAN controller is in the error passive status, the Error Passive Interrupt will be triggered. |
| Arbitration Lost Interrupt | When the CAN controller lost the arbitration and becomes a receiver. The Arbitration Lost Interrupt will be triggered. |
| Bus Error Interrupt | When the CAN controller detects an error on the CAN bus, the Bus Error Interrupt will be triggered. |

Use one-byte value to implement the interrupt. For example, if receive and overrun interrupt are needed in the BasicCAN(CAN 2.0A) mode. Set the IntMode value to 0x09(That is 0x01+0x08.).

CANBaud: Use a long int to set this parameter. For example, if users want to set CAN baud to 125K bps. Use the value 125000UL.

BT0, BT1: Set the special user-defined baud rate. Users can set arbitrary baud with these parameters. But users need to have the background of SJA1000 CAN controller and TJA1042 CAN transceiver, and calculate the values of BT0 and BT1 by themselves (The clock frequency of CAN controller is 16MHz.).

AccCode, AccMask: The AccCode is used for deciding what kind of ID the CAN controller will be accepted. The AccMask is used for deciding which bit of ID will need to check with AccCode. If the bit of AccMask is set to 0, it means that the bit in the same position of ID need to be checked, and the bit value ID need to match the bit of AccCode in the same position.

For 11-bit ID Message:

| Register | bits of register | Filter Target |
|---|---|---|
| AccCode[0] and AccMask[0] | bit7~bit0 | bit10 ~ bit3 of ID |
| AccCode[1] and AccMask[1] | bit7~bit5 | bit2 ~ bit0 of ID |
| AccCode[1] and AccMask[1] | bit4 | RTR |
| AccCode[1] and AccMask[1] | bit3~bit0 | no use |



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

| | | |
|---|---|---|
| AccCode[2] and AccMask[2] | bit7~bit0 | bit7 ~ bit0 of 1st byte data |
| AccCode[3] and AccMask[3] | bit7~bit0 | bit7 ~ bit0 of 2nd byte data |

For 29-bit ID Message:

| Register | bits of register | Filter Target |
|---|---|---|
| AccCode[0] and AccMask[0] | bit7~bit0 | bit28 ~ bit21 of ID |
| AccCode[1] and AccMask[1] | bit7~bit0 | bit20 ~ bit13 of ID |
| AccCode[2] and AccMask[2] | bit7~bit0 | bit12 ~ bit5 of ID |
| AccCode[3] and AccMask[3] | bit7~bit3 | bit4 ~ bit0 of ID |
| AccCode[3] and AccMask[3] | bit2 | RTR |
| AccCode[3] and AccMask[3] | bit1~bit0 | no use |

Note: 1. AccCode[0] means the most significant byte of AccCode and AccCode[3] means the least significant byte of AccCode.

2. AccMask[0] means the most significant byte of AccMask and AccMask[3] means the least significant byte of AccMask.

3. Bit10 is most significant bit and Bit0 is least significant bit

For example (In 29 bit ID message):

AccCode : 00h 00h 00h A0h
AccMask : FFh FFh FFh 1Fh
ID Value : ?? ?? ?? Ah and Bh will be accepted. (??: don't care)

(Note: The mark "h" behind the value means hex format.)

## Tip change Baud Rate and change Filter



After calling XWCANInit, users can change Buad Rate via API **SetCANBaud().**

**SetCANBaud (int CANPort,unsigned long CANBaud, char BT0, char BT1)**

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

CANBaud, BT0, BT1: Please refer to the parameters description in the XWCANInit function.

After calling XWCANInit, users can change CAN message Filter via API **SetCANFilter().**

**SetCANFilter (int CANPort,unsigned long AccCode, unsigned long AccMask)**

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

AccCode, AccMask: Please refer to the parameters description in the XWCANInit function.



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

**3. CAN_Restore(int CANPort)**

Set the interrupt function disable, release all software buffer, and reset CAN chip. This function must be called to release resource before the program is terminated.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

For example, CAN Initilization setting.

```
#include <stdlib.h>
#include "..\lib\UPAC5000.h"
#include "..\lib\XWCAN.h"

/*If the UserCANInt() function is not used, please don't remove it.*/
void UserCANInt(int CANPort,char CANInt)
{
   ...
}

void main(void)
{
      int ret;
      InitLib();     //Initial uPAC-5000 LIB
      /*********************************************************/
      /*          initialize and configure the CAN controller                    */
      /*********************************************************/
      ret=XWCANInit(0,0,125000UL,0,0,0x00000000UL,0xffffffffUL);
      /*
          XCANInit function Parameter descriptions
          0                  : for CAN Port 1
          0                  : for IntMode useless
          125000UL           : for CAN baud
          0                  : for BT0 of user defined baud
          0                  : for BT1 of user defined baud
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
        0x00000000UL : for AccCode of CAN message filter
        0xffffffffUL : for AccMask of CAN message filter
    */
    switch (ret)
    {
        /*Return Code,Check if configuration is OK*/
        case CAN_ResetError:
                Print("Reset Error!\n\r");
                return;
        case CAN_SetACRError:
                Print("\n\rSet ACR Error!");
                return;
        case CAN_SetAMRError:
                Print("\n\rSet AMR Error!");
                return;
        case CAN_SetBaudRateError:
                Print("\n\rSet Baud Rate Error!");
                return;
        case CAN_BaudNotSupport:
                Print("\n\rBaud Rate Not Support!");
                return;
        case CAN_ConfigError:
                Print("\n\rConfiguration Failure!");
                return;
            case CAN_SetPortError:
                 Print("\n\rSet CAN Port Failure!");
                return;
      }
            CAN_Restore(0); //release resource
    /*CAN_Restore function parameter descirptions:
        0               :for CAN Port 1
    */
}
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.2. API for CAN Interrupt

## API for installing CAN Interrupt

**1. CAN_InstallIrq(int CANPort)**

Set the interrupt function enable. Afterwards, the CPU of µPAC-5001D-CAN2 can receive the interrupt signal from CAN controller.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

## API for removing CAN Interrupt

**2. CAN_RemoveIrq(int CANPort)**

Disable the interrupt function. Afterwards, the CPU of µPAC-5001D-CAN2 can't receive the interrupt signal from CAN controller.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

## API for creating CAN transmission/reception Interrupt buffer

**3. CAN_CreateBuffer(int CANPort, int BufMode, unsigned int BufferSize)**

Call this function for changing the reception and transmission software buffer sizes.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

BufMode: 0 for changing reception software buffer size.
Others for changing transmission software buffer size.

BufferSize: the new buffer sizes for software buffer. (maimum 3120 CAN messages)

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Note: If users don't use this function after enabling CAN interrupt, the default reception/transmission software buffer sizes will be created and its size is 256 records.

For example, CAN interrupt setting.

```
#include <stdlib.h>
#include "..\lib\UPAC5000.h"
#include "..\lib\XWCAN.h"

/*If the UserCANInt() function is not used, please don't remove it.*/
void UserCANInt(int CANPort,char CANInt)
{
   ...
}
void main(void)
{
      int ret;
      InitLib();     //Initial uPAC-5000 LIB
      /*********************************************************/
      /*          initialize and configure the CAN controller                    */
      /*********************************************************/
      ret=XWCANInit(0,1,125000UL,0,0,0x00000000UL,0xffffffffUL);
      /*
          XCANInit function Parameter descriptions
          0                  : for CAN Port 1
          1                  : for reception interrupt
          125000UL           : for CAN baud
          0                  : for BT0 of user defined baud
          0                  : for BT1 of user defined baud
          0x00000000UL : for AccCode of CAN message filter
          0xffffffffUL : for AccMask of CAN message filter
      */
      switch (ret)
      {
          /*Return Code,Check if configuration is OK*/
          case CAN_ResetError:
                Print("Reset Error!\n\r");
                return;
          …
          …
        }
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
    CAN_CreateBuffer(0,0,1000);
    /*CAN_CreateBuffer function parameter descirptions:
        0           :for CAN Port 1
        0           :for CAN reception buffer
        1000        :for 1000 CAN messages
    */
    CAN_InstallIrq(0); //Install Port0 IRQ
    /*CAN_InstallIrq function parameter descirptions:
        0           :for CAN Port 1
    */
    CAN_Restore(0);
}
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.3. API for Transmitting CAN Messages

API for Transmitting CAN Messages

1. **SendCANMsg(int CANPort,unsigned char Mode,unsigned long MsgID, unsigned char RTR, unsigned char DataLen, unsigned char *Data)**

   If the transmit buffer is disable, this function will send a message to the CAN network. However, if the transmit buffer is enable, this function will send all the messages stored in the transmit buffer to the CAN network.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

Mode: This parameter is used for CAN ID type.

| Mode value | Meaning |
|---|---|
| 0 | Send a 11-bit ID CAN message |
| others | Send a 29-bit ID CAN message |

MsgID: The ID of this CAN message. The ID may be a 11-bit value or 29-bit value.

RTR: Remote transmits request byte.

| RTR value | Meaning |
|---|---|
| 0 | This CAN message is not a remote transmit request message. |
| 1 | This CAN message is a remote transmit request message. |

DataLen: The pure data length of a CAN messages. The range of this value is 0~8.

*Data: Store the data of CAN message. The numbers of data bytes need to match with the "DataLen".

For example, CAN data transmitssion.

```
#include <stdlib.h>
#include "..\..\lib\UPAC5000.h"
#include "..\..\lib\XWCAN.h"
```

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
/*If the UserCANInt() function is not used, please don't remove it.*/
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
void UserCANInt(int CANPort,char CANInt)
{
   ...
}
void main(void)
{
      int ret;
      unsigned char MsgMode,MsgRTR,MsgDataLen,MsgData[8];
      unsigned long MsgID,MsgUpperTimeStamp,MsgLowerTimeStamp;

      InitLib();     //Initial uPAC-5000 LIB
      /*********************************************************/
      /*          initialize and configure the CAN controller                  */
      /*********************************************************/
      ret=XWCANInit(0,0,125000UL,0,0,0x00000000UL,0xffffffffUL);
      /*
          XCANInit function Parameter descriptions
          0                  : for CAN Port 1
          0                  : for IntMode useless
          125000UL           : for CAN baud
          0                  : for BT0 of user defined baud
          0                  : for BT1 of user defined baud
          0x00000000UL : for AccCode of CAN message filter
          0xffffffffUL : for AccMask of CAN message filter
      */
      switch (ret)
      {
          /*Return Code,Check if configuration is OK*/
          case CAN_ResetError:
                Print("Reset Error!\n\r");
                return;
          …
          …
        }

        while(1){
          /*********************************************************/
          /*                  Send CAN messages to CAN bus                 */
          /*********************************************************/
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
ret=SendCANMsg(0,1,0x12345678UL,0,8,MsgData);
/*      SendCANMsg function parameter descirptions:
        0                   : for CAN Port 1
        1                   : for Send message with 29-bit ID
        0x12345678UL : for CAN message ID
        0                   : for CAN message RTR
        8                   : for CAN message data length
        MsgData          : for CAN message output data
*/
if (ret)
{
    switch(ret)
     { /*Return Code,Check if configuration is OK*/
        case CAN_DataLengthError:
                Print("Transmission Data Length Error!\n");
                break;
        case CAN_TransmitIncomplete:
                Print("Transmission is incomplete!\n");
                break;
        case CAN_TransmitBufferLocked:
                Print("CAN controller transmit Buffer is locked!\n");
                break;
        }
        break;
}
DelayMs(500);
ret=SendCANMsg(0,0,0x123UL,0,8,MsgData);
 /*     SendCANMsg function parameter descirptions:
           0                   : for CAN Port 1
           0                   : for Send message with 11-bit ID
           0x123UL         : for CAN message ID
           0                   : for CAN message RTR
           8                   : for CAN message data length
           MsgData          : for CAN message output data
*/
if (ret)
{
    switch(ret)
     { /*Return Code,Check if configuration is OK*/
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
                case CAN_DataLengthError:
                        Print("Transmission Data Length Error!\n");
                        break;
                case CAN_TransmitIncomplete:
                        Print("Transmission is incomplete!\n");
                        break;
                case CAN_TransmitBufferLocked:
                        Print("CAN controller transmit Buffer is locked!\n");
                        break;
              }
              break;
            }
            DelayMs(500);
            if (Kbhit()){ /*if press any key, exit the program*/
                Print("Exit this program!\n");
                break;
            }
        }
        CAN_Restore(0);
    }
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.4. API for Receiving CAN Messages

API for Transmitting CAN Messages

1. **GetCANMsg(int CANPort,unsigned char *Mode, unsigned long *MsgID**
**, unsigned char *RTR, unsigned char *DataLen, unsigned char *Data**
**, unsigned long *UpperTime , unsigned long *LowerTime)**

Receive CAN messages from receive buffer or from CAN bus directly. If the receive interrupt is set to enable in IntMode parameter of XWCANInit function. This function will read back the CAN message stored in the software receive buffer. If the receive interrupt is disable, this function uses the polling method to check if there is any CAN message in CAN chip buffer. If yes, return the CAN message.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

*Mode: This parameter is used for get the ID type (11-bit or 29-bit ID) of a CAN message.

*MsgID: This is for obtaining the ID of a CAN message.

*RTR: This is for obtaining the RTR of a CAN message.

| RTR value | Meaning |
|---|---|
| 0 | This CAN message is not a remote transmit request message. |
| 1 | This CAN message is a remote transmit request message. |

*DataLen: This is for obtaining the data length of a CAN message.

*Data: This is for obtaining the Data of a CAN message. The Data buffer size must be 8 bytes.

*UpperTime: Get the time stamp of a CAN message. The time stamp unit is us (micro second), This parameter only show the upper part of time stamp.

**Real time stamp = upper part * 0x1000000UL+lower part**

*LowerTime: Get the lower part of time stamp of a CAN message.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## Tip get CAN message status



Users can Read the CAN controller status and software buffer overflow flag message via API GetStatus().

**GetStatus (int CANPort,unsigned char *CANReg, unsigned char *OverflowFlag)**

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

*CANReg: CAN controller status. Please refer to the following table.

| Bit NO. | Description |
|---|---|
| 7 (MSB) | Bus status. 1 for bus off, 0 for bus on. |
| 6 | Error status. 1 for at least one error, 0 for OK. |
| 5 | Transmit status. 1 for transmitting, 0 for idle. |
| 4 | Receive status. 1 for receiving, 0 for idle. |
| 3 | Transmit complete status. 1 for complete, 0 for incomplete. |
| 2 | Transmit buffer status. 1 for released, 0 for locked |
| 1 | Data overrun status. 1 for reception buffer overrun, 0 for OK. |
| 0 (LSB) | Receive buffer status. 1 for at least one message stored in the reception buffer, 0 for empty. |

*OverflowFlag: CAN reception and transmission buffer overflow flag. Please refer to the following table.

| Bit NO. | Description |
|---|---|
| Others | Reserved |
| 1 | 1 for reception software buffer overflow. 0 for normal. |
| 0 (LSB) | 1 for transmission software buffer overflow. 0 for normal. |

Users can clean the CAN reception or transmission software buffer overflow flag via API ClearStatus().

**ClearStatus(int CANPort)**

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

When one of these two buffers is full, the corresponding overflow flag will be set to 1.
In this case, users need to use this function to clear the overflow flag to acknowledge the error information.

---

For example, CAN data reception.

```
#include <stdlib.h>
#include "..\lib\UPAC5000.h"
#include "..\lib\XWCAN.h"

/*If the UserCANInt() function is not used, please don't remove it.*/
void UserCANInt(int CANPort,char CANInt)
{
   ...
}
void main(void)
{
    int ret;
    unsigned char MsgMode,MsgRTR,MsgDataLen,MsgData[8];
    unsigned long MsgID,MsgUpperTimeStamp,MsgLowerTimeStamp;

    InitLib();    //Initial uPAC-5000 LIB
    /*********************************************************/
    /*          initialize and configure the CAN controller                    */
    /*********************************************************/
    ret=XWCANInit(0,0,125000UL,0,0,0x00000000UL,0xffffffffUL);
    /*
        XCANInit function Parameter descriptions
        0                  : for CAN Port 1
        0                  : for IntMode useless
        125000UL           : for CAN baud
        0                  : for BT0 of user defined baud
        0                  : for BT1 of user defined baud
        0x00000000UL : for AccCode of CAN message filter
        0xffffffffUL       : for AccMask of CAN message filter
    */
    switch (ret)
    {
        /*Return Code,Check if configuration is OK*/
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
        case CAN_ResetError:
                Print("Reset Error!\n\r");
                return;
        …
        …
    }

    while(1)
    {
        /*********************************************************/
        /*                Receive CAN messages from CAN bus      */
        /*********************************************************/
        ret=GetCANMsg(0,&MsgMode,&MsgID,&MsgRTR,&MsgDataLen,
                        MsgData,&MsgUpperTimeStamp,&MsgLowerTimeStamp);
        /* GetCANMsg function Parameter descriptions
              0              : CAN Port 1
              &Mode          : Get CAN ID is 2.0A or 2.0B
              &MsgID         : Get CAN Message ID
              &RTR           : Get CAN Message RTR
              &DataLen       : Get CAN Message Data Length
              &Data          : Get CAN Message Data
              &UpperTime : Get the time stamp of a CAN message
              &LowerTime : Get the lower part of time stamp of a CAN message
        */
        if (!ret)
        {
           Print("%lu(%lus-%luus):Mode=%d,ID=%lx,RTR=%d,Len=%d",MsgCnt,MsgUpperTimeStamp,
           MsgLowerTimeStamp,MsgMode,MsgID,MsgRTR,MsgDataLen);
           if (MsgDataLen && !MsgRTR)
           {
                Print(",Data=");
                for (i=0;i<MsgDataLen;i++){
                    Print("%x,",MsgData[i]);
                }
                Print("\n\r");
           }
            else
           {
              switch(ret)
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
                {
                    case CAN_DataLengthError:
                        Print("Reception Data Length Error!\n");
                        break;
                    case CAN_DataOverrun:
                        Print("Software Reception Data Buffer Overrun!\n");
                        break;
                    case CAN_ReceiveError:
                        Print("Receive data Error!\n");
                        break;
                    case CAN_ReceiveBufferEmpty:
                        /*No message in receive buffer*/
                        break;
                }
            }
        }
    }
    CAN_Restore(0);
}
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.5. API for LED Indicator

### API for LED Indicator

1. CL1Off(), CL2Off(), CL3Off()
   Turn CL1, CL2, CL3 off.

2. CL1On(), CL2On(), CL3On()
   Turn CL1, CL2, CL3 on.

For example. Control LED indicator.

```
#include <stdlib.h>
#include "..\..\lib\UPAC5000.h"
#include "..\..\lib\XWCAN.h"

/*If the UserCANInt() function is not used, please don't remove it.*/
void UserCANInt(int CANPort,char CANInt)
{
   ...
}
void main(void)
{
      int ret,i=0;
      InitLib();
      /*************************************************************/
      /*          initialize and configure the CAN controller      */
      /*************************************************************/
      ret=XWCANInit(0,0,125000UL,0,0,0x00000000UL,0xffffffffUL);
      switch (ret)
      {
          /*Return Code,Check if configuration is OK*/
          case CAN_ResetError:
                Print("Reset Error!\n\r");
                return;
          …
        }

      /*************************************************************/
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
    /*                          Initial all LEDs                                */
    /************************************************************/
    Print("Press any key to exit the program.\n\r");
    while(1){
        switch(i)
        {
            case 0:
                CL1On();
                CL2Off();
                CL3Off();
                break;
            case 1:
                CL1Off();
                CL2On();
                CL3Off();
                break;
            case 2:
                CL1Off();
                CL2Off();
                CL3On();
        }
        if (++i==3) i=0;
        DelayMs(500);
    }
    CL1Off();
    CL2Off();
    CL3Off();
    CAN_Restore(0);
}
```



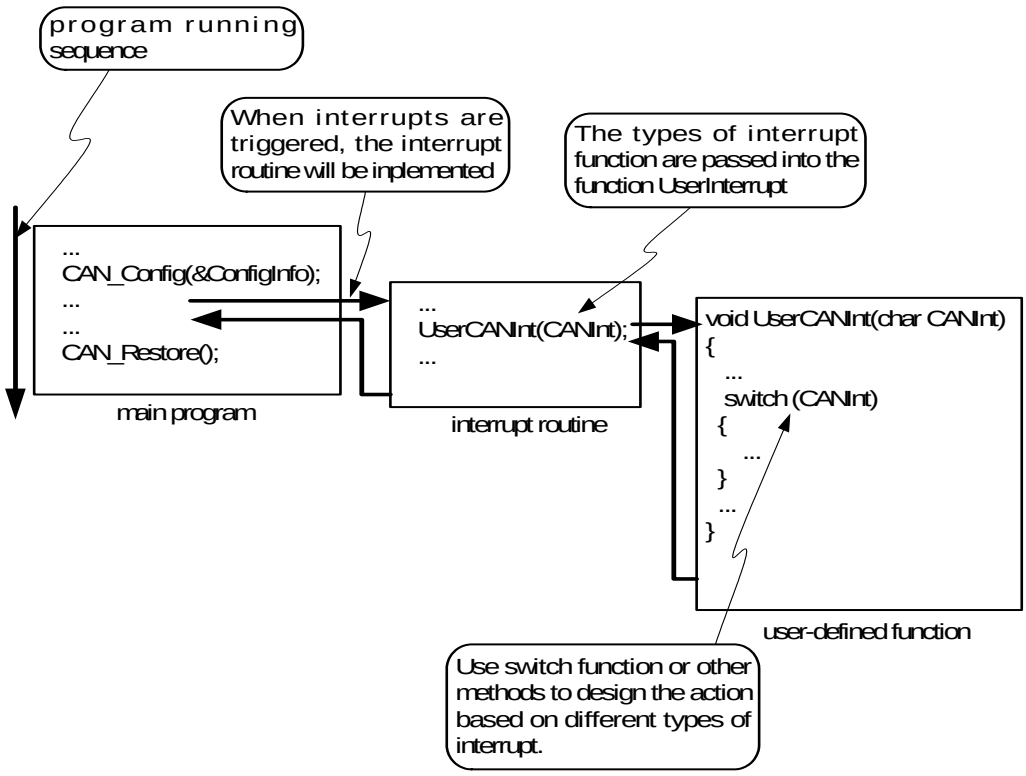
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.6. API for User-Defined Interrupt

## API for User-Defined Interrupt

**1. UserCANInt(int CANPort,char CANInt)**

This function is created by users and is used to program the CAN interrupt service routine by users. The parameter CANINT is passed automatically when the interrupt functions are triggered. It indicates what kinds of CAN controller interrupt are active. Therefore, users only need to design their interrupt routine according to dealing with different interrupt functions. If it is not used, please reverse this function in the users' .C file for avoiding the complier error. The following figure is the general concept of the function UserCANInt.




**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

CANInt: The interrupt service routine will bypass the CANInt parameter to users to indicate what interrupt is triggered. For the meanings of CANInt parameters, please refer to the following table.

| CANIntMode Value (Hex) | Meaning |
|---|---|
| 0x01 | Receive a message successfully |



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

| | |
|---|---|
| 0x02 | Transmit a message successfully |
| 0x04 | Error warring |
| 0x08 | Data Overrun |
| 0x10 | CAN controller wake-up |
| 0x20 | Bus Passive |
| 0x40 | Arbitration Lost |
| 0x80 | Bus Error |

For example, user-defined interrupt.

```
#include <stdlib.h>
#include "..\lib\UPAC5000.h"
#include "..\lib\XWCAN.h"

unsigned long MsgCount=0;    //Create a global variable to check the    number of RX Interrupt

/*If the UserCANInt() function is not used, please don't remove it.*/
void UserCANInt(int CANPort,char CANInt)
{ /*check CAN Port 1 receive interrupt*/
   if(CANPort ==0x0)
   {
       if (CANInt==0x01)
       {
           MsgCount++;
       }
    }
}

void main(void)
{
       int ret,i=0;
       unsigned long tmpMsgCount=0;
      InitLib();
      /*********************************************************/
      /*          initialize and configure the CAN controller                  */
      /*********************************************************/
      ret=XWCANInit(0,1,125000UL,0,0,0x00000000UL,0xffffffffUL);
       /*
          XCANInit function Parameter descriptions
          0                  : for CAN Port 1
          1                  : for reception interrupt
          125000UL           : for CAN baud
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
        0                   : for BT0 of user defined baud
        0                   : for BT1 of user defined baud
        0x00000000UL : for AccCode of CAN message filter
        0xffffffffUL : for AccMask of CAN message filter
    */
    switch (ret)
    {
        /*Return Code,Check if configuration is OK*/
        case CAN_ResetError:
            Print("Reset Error!\n\r");
            return;
        …
    }

    /*************************************************************/
    /*                          Enable Irq                       */
    /*************************************************************/
    CAN_CreateBuffer(0,0,1000);
    /* CAN_CreateBuffer function parameter descirptions:
        0           :for CAN Port 1
        0           :for CAN reception buffer
        1000        :for 1000 CAN messages
    */
    CAN_InstallIrq(0); //Install Port0 IRQ
    /* CAN_InstallIrq function parameter descirptions:
        1           :for CAN Port 1
    */

    while(1)
    {
      if(MsgCount != tmpMsgCount)
      {
        Print("%lu CAN message%c %s received.\n\r",
        MsgCount,(MsgCount<2)?' ':'s',(MsgCount<2)?"is":"are");
       }
    }

    CAN_Restore(0);
}
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 4.10.7. API for Detecting CAN Bus Baud Rate

## API for detecting CAN bus baud rate

**1. CAN_SearchBaud(int CANPort,unsigned long CANBaud, char BT0, char BT1, unsigned int Timeout)**

Enter “Listen Only Mode” and enable receive and error interrupt to detect the right bit-rate of the CAN bus. Upon successful reception of a message, the “CAN_NoError” message will be return. Otherwise, the “CAN_AutoBaudTimeout” message will be return.

**Parameter:**

CANPort：Selected CAN Port.

| value | CAN Port |
|---|---|
| 0 | For CAN Port 1 |
| 1 | For CAN Port 2 |

CANBaud: Use a long int to set this parameter. For example, if users want to set CAN baud to 125K bps. Use the value 125000UL.

BT0, BT1: Set the special user-defined baud rate. Users can set arbitrary baud with these parameters. But users need to have the background of SJA1000 CAN controller and TJA1042 CAN transceiver, and calculate the values of BT0 and BT1 by themselves (The clock frequency of CAN controller is 16MHz.)

Timeout: Set the timer for search a necessary CAN bus baud rate.

For example, detecting CAN bus baud rate and send/receive CAN message via the detected baud rate .

```
#include <stdlib.h>
#include "..\lib\UPAC5000.h"
#include "..\lib\XWCAN.h"

/*If the UserCANInt() function is not used, please don't remove it.*/
void UserCANInt(int CANPort,char CANInt)
{
   ...
}
void main(void)
{
      int ret,i;
      unsigned char MsgMode,MsgRTR,MsgDataLen,MsgData[8];
      unsigned long MsgID,MsgUpperTimeStamp,MsgLowerTimeStamp;
```



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
unsigned long CANBaud[]={1000000UL,800000UL,500000UL,250000UL,
                         200000UL,125000UL,100000UL,50000UL,
                         25000UL,20000UL,10000UL,5000UL};
int FoundBaud=0;
InitLib();
//
/*************************************************************/
/*              Start to detect CAN bus baudrate             */
/*************************************************************/
Print("Start to detect CAN bus baudrate\r\n");
for(i=0;i<=11;i++)
{
   FoundBaud=CAN_SearchBaud(0,CANBaud[i],0,0,500);
   if((FoundBaud == 0) || (FoundBaud != CAN_AutoBaudTimeout))
    break;
}
if(FoundBaud == CAN_NoError)
    Print("Find CAN Baudrate: %lu\r\n",CANBaud[i]);
else if(FoundBaud == CAN_AutoBaudTimeout)
    Print("Timeout, CAN Baudrate Not Find, use default setting.\r\n");
else    Print("CAN_SearchBaud Error, %d\r\n",FoundBaud);
 /*************************************************************/
/*          initialize and configure the CAN controller      */
/*************************************************************/
if(FoundBaud == CAN_NoError)
   ret=XWCANInit(0,3,CANBaud[i],0,0,0x00000000UL,0xffffffffUL);
else
   ret=XWCANInit(0,3,125000UL,0,0,0x00000000UL,0xffffffffUL);
 /* XCANInit function Parameter descriptions
            0                 : for CAN port selection
            3                 : for Receive and transmission interrupt
            125000UL          : for CAN baud
            0                 : for BT0 of user defined baud
            0                 : for BT1 of user defined baud
            0x00000000UL : for AccCode of CAN message filter
            0xffffffffUL : for AccMask of CAN message filter
 */
switch (ret){    /*Check if configuration is OK*/
     case CAN_ResetError:
```

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

```
            Print("Reset Error!\n\r");
            return;
        case CAN_SetACRError:
            Print("\n\rSet ACR Error!");
            return;
        case CAN_SetAMRError:
            Print("\n\rSet AMR Error!");
            return;
        case CAN_SetBaudRateError:
            Print("\n\rSet Baud Rate Error!");
            return;
        case CAN_BaudNotSupport:
            Print("\n\rBaud Rate Not Support!");
            return;
        case CAN_ConfigError:
            Print("\n\rConfiguration Failure!");
            return;
        }
    }

    while(1)
    {
        /*************************************************************/
        /*                Receive CAN messages from CAN bus          */
        /*************************************************************/
        …
        …
        /*************************************************************/
        /*                Send CAN messages to CAN bus               */
        /*************************************************************/
        …
        DelayMs(100);
        if (Kbhit()){ /*if press any key, exit the program*/
            Print("Exit this program!\n");
            break;
        }
    }
    CAN_Restore(0);
}
```

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

## 4.10.8. Return Code

| Return Code | Error ID | Comment |
|---|---|---|
| 0 | CAN_NoError | OK |
| 5 | CAN_ResetError | Enter reset mode error |
| 8 | CAN_ConfigError | CAN chip configure error |
| 9 | CAN_SetACRError | Set to Acceptance Code Register error |
| 10 | CAN_SetAMRError | Set to Acceptance Mask Register error |
| 11 | CAN_SetBaudRateError | Set Baud Rate error |
| 14 | CAN_InstallIrqFailure | Enable interrupt functions failure |
| 15 | CAN_RemoveIrqFailure | Disable interrupt functions failure |
| 16 | CAN_TransmitIncomplete | Data can't be transmitted successfully |
| 17 | CAN_TransmitBufferLocked | Previously transmission is not completed yet |
| 18 | CAN_ReceiveBufferEmpty | No message is stored in the receive buffer now |
| 19 | CAN_DataOverrun | Data was lost because there was not enough space in software receive buffer |
| 20 | CAN_ReceiveError | Receive data is not completed |
| 21 | CAN_SoftBufferIsFull | Software transmit buffer is full |
| 22 | CAN_SoftBufferIsEmpty | There is no message stored in the user-declared software buffer |
| 23 | CAN_BaudNotSupport | This Baud Rate is not supported |
| 24 | CAN_DataLengthError | Data length doesn't match the total data bytes |
| 25 | CAN_NotEnoughMemory | There is not enough memory space to create the reception or transmission software buffer. |
| 26 | CAN_TypeOf7188Error | The type of 7188 is not defined by this library |
| 50 | CAN_AutoBaudTimeout | CAN bus baud rate not found |
| 51 | CAN_SendParamError | Set "CANport" parameter on SendCANMsg() API error |
| 52 | CAN_ReceiveParamError | Set "CANport" parameter on GetCANMsg () API error |

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 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:

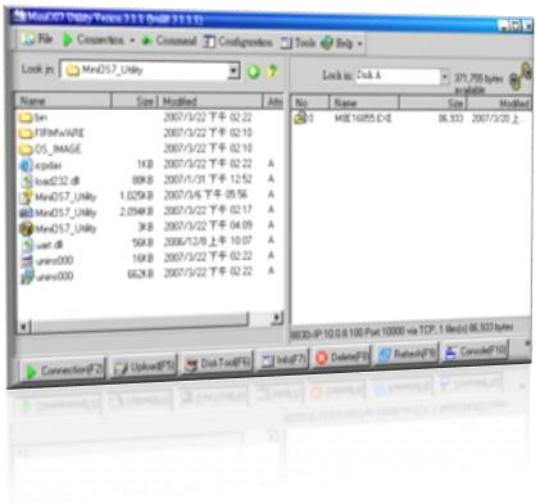
| Feature | MiniOS7 | ROM-DOS |
|---|---|---|
| Power-up time | 0.1 sec | 4 ~ 5 sec |
| More compact size | < 64 K bytes | 64 K bytes |
| Support for I/O expansion bus | Yes | No |
| Support for ASIC key | Yes | No |
| Flash ROM management | Yes | No |
| OS update (Upload) | Yes | No |
| Built-in hardware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

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# 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/

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# Appendix C. 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)

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# C.1. Turbo C 2.01

You have a couple of choices here, you can:

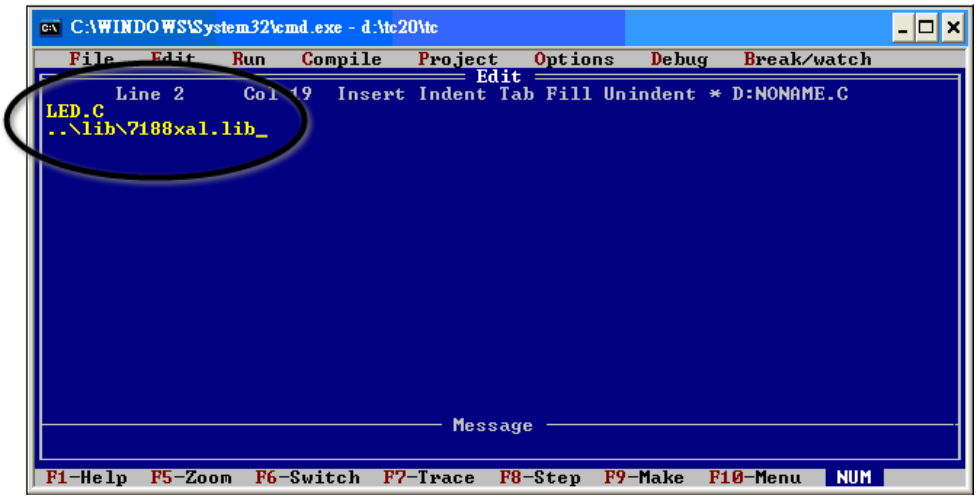
1: Using a command line

For more information, please refer to
CD:\8000\NAPDOS\8000\841x881x\Demo\hello\Hello_C\gotc.bat
tcc -Ic:\tc\include -Lc:\tc\lib hello1.c ..\..\Demo\basic\Lib\uPAC5000.lib

2: Using the TC Integrated Environment

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

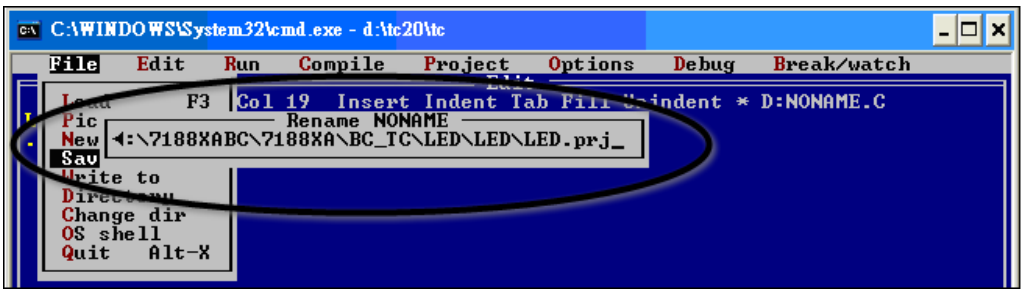
Adding the necessary library and file to the project



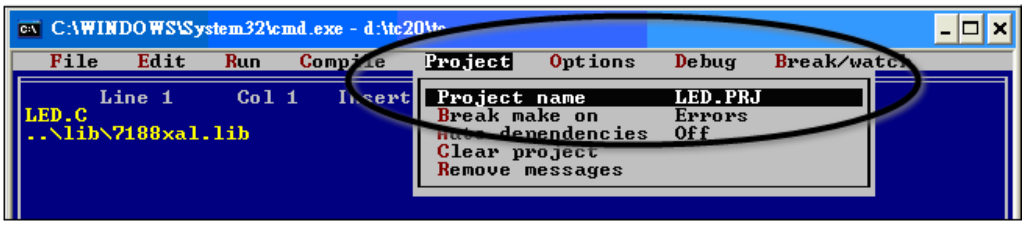



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

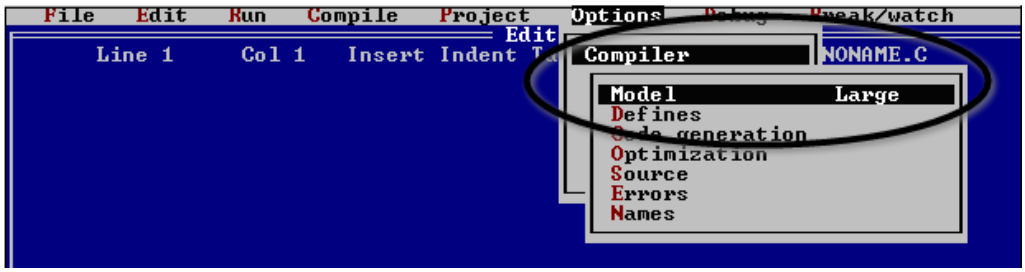
Step 3: Save the project and entering a name, such as LED.prj

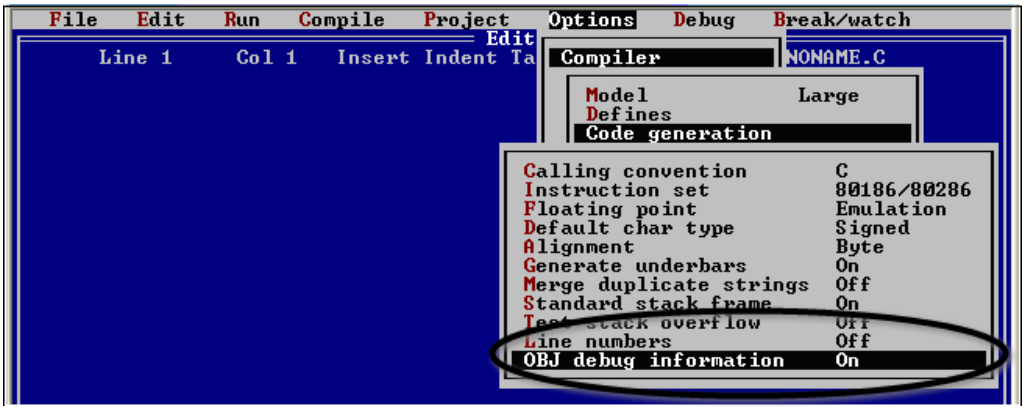



Step 4: Load the Project




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



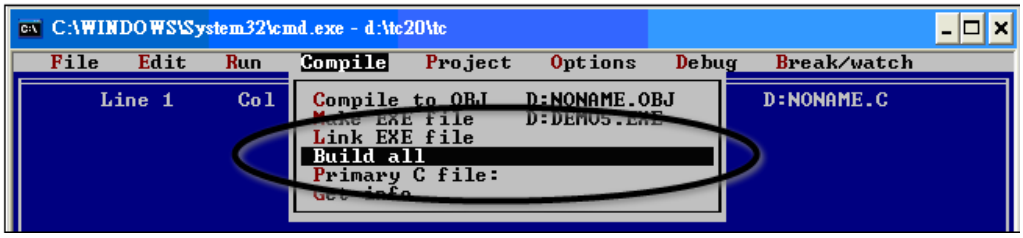



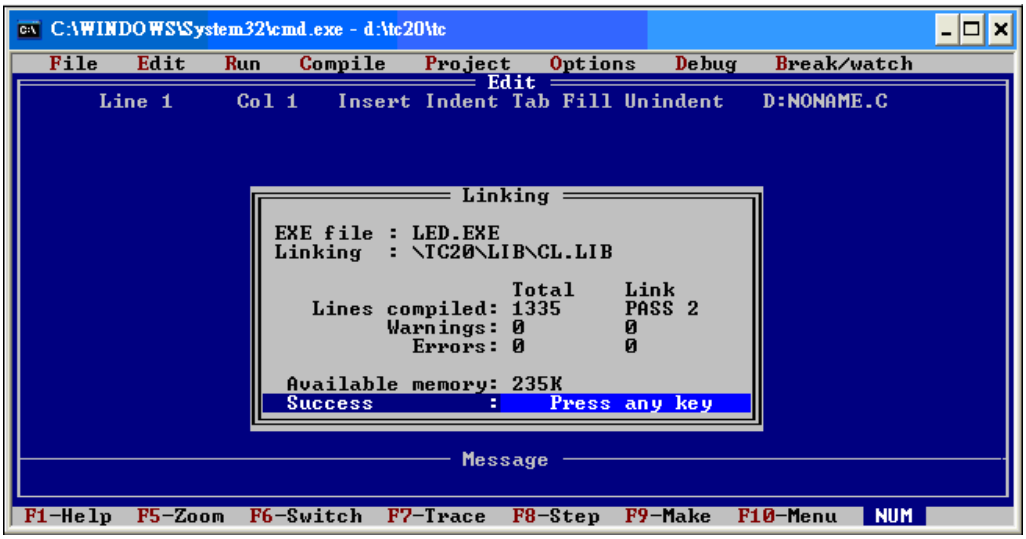

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Step 6: Building the project



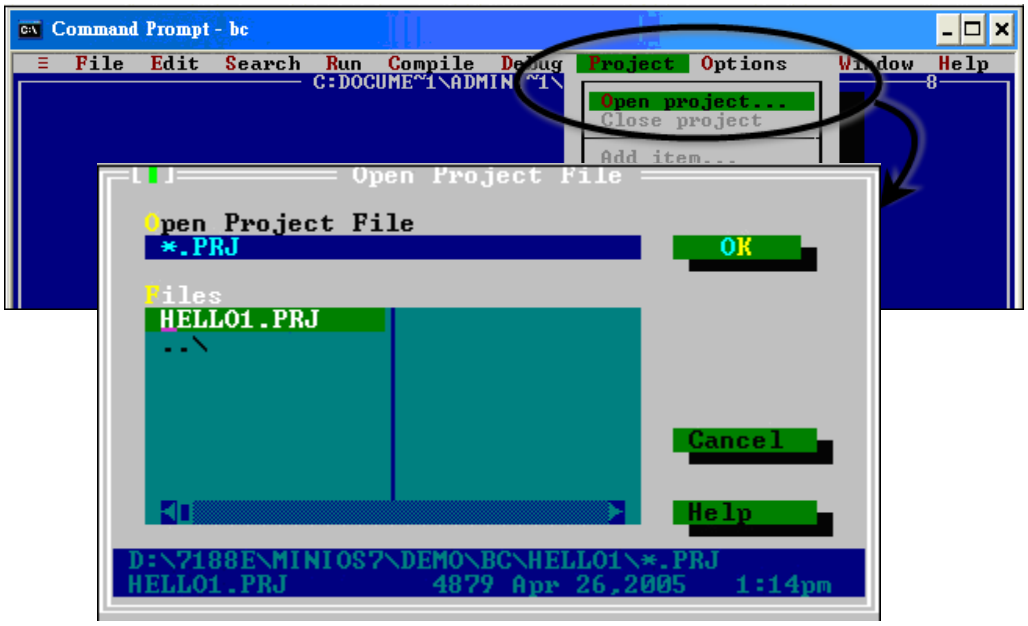



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

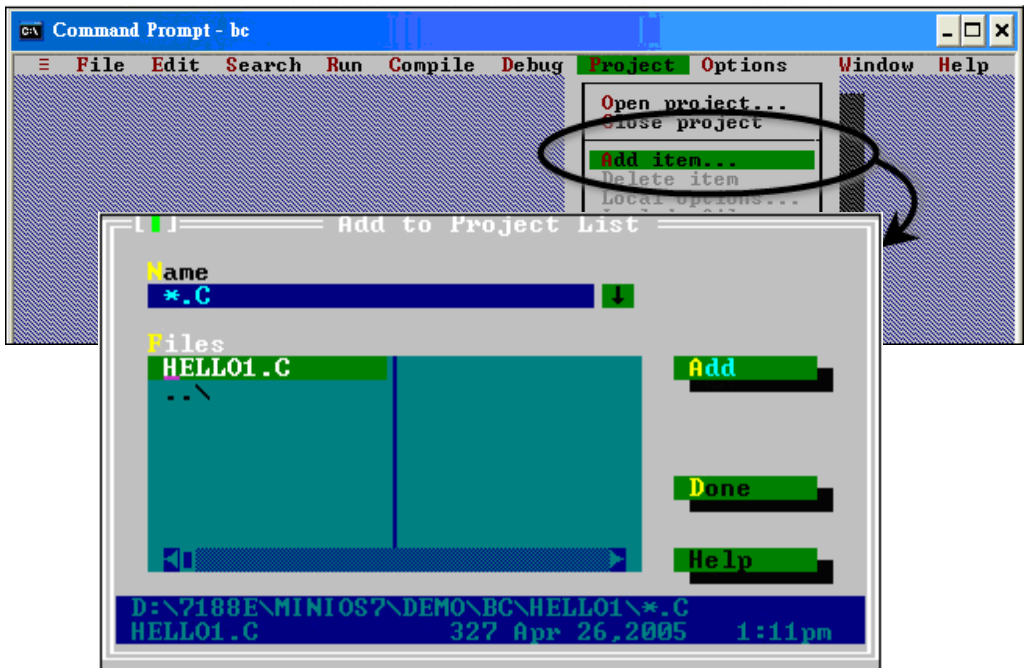
# C.2. Borland C++ 3.1

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

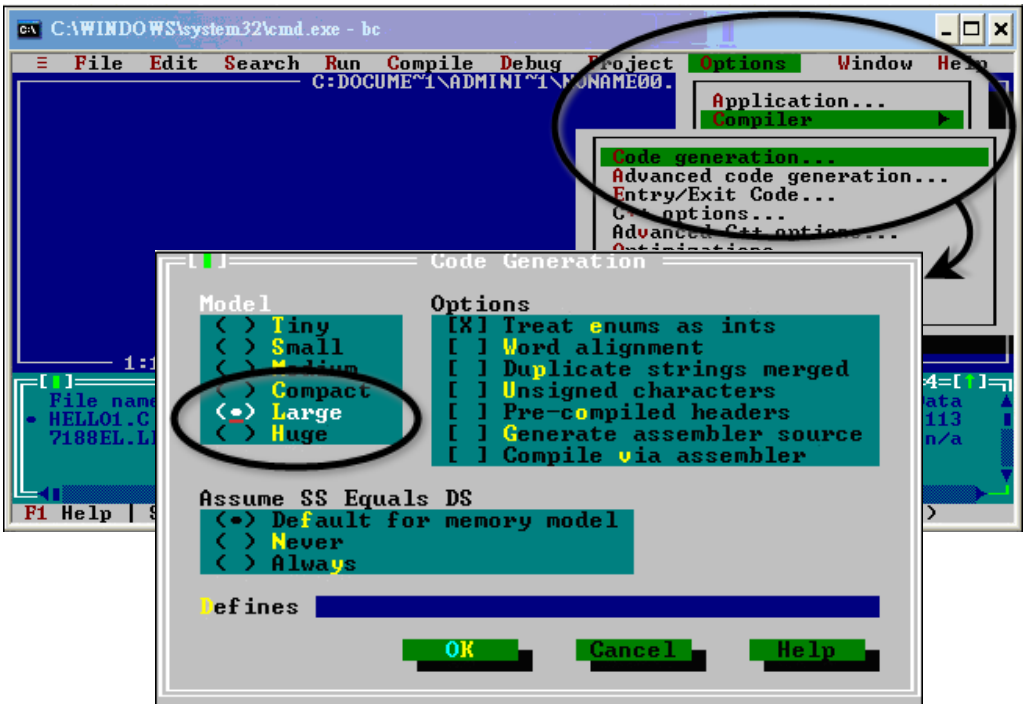



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

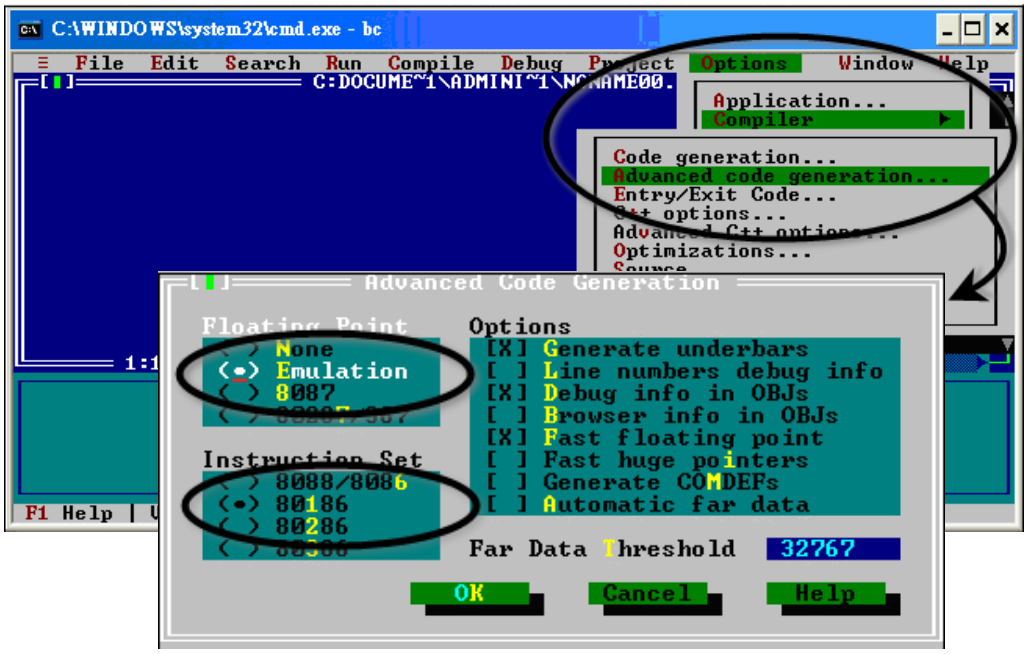


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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



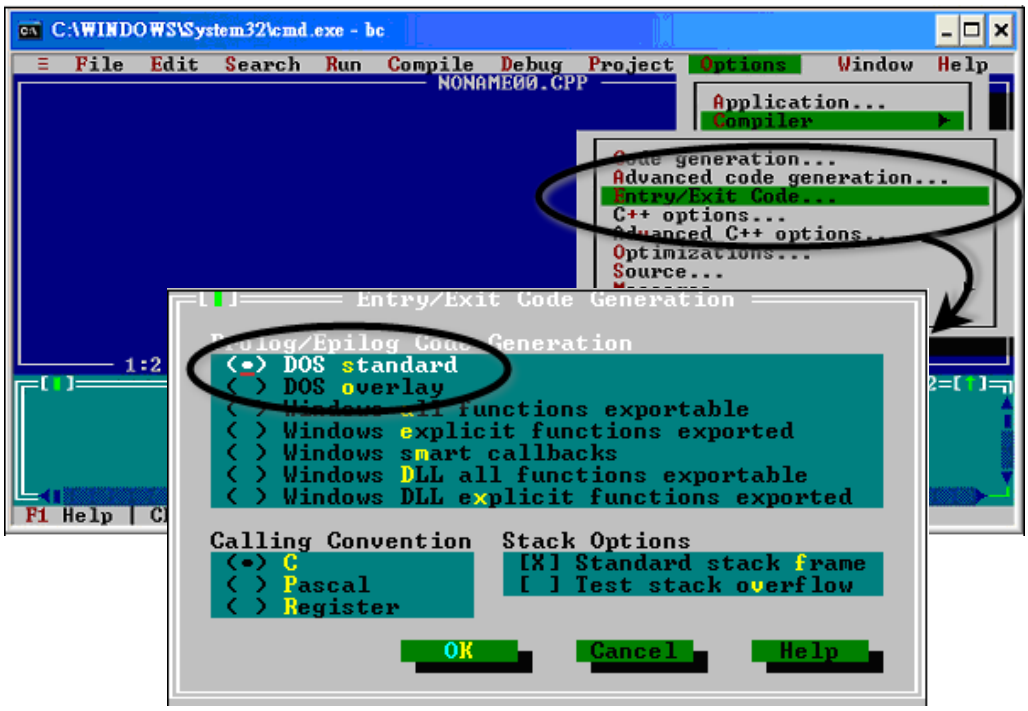

Step 5: Set the Advanced code generation options and Set the Floating Point to Emulation and the Instruction Set to 80186



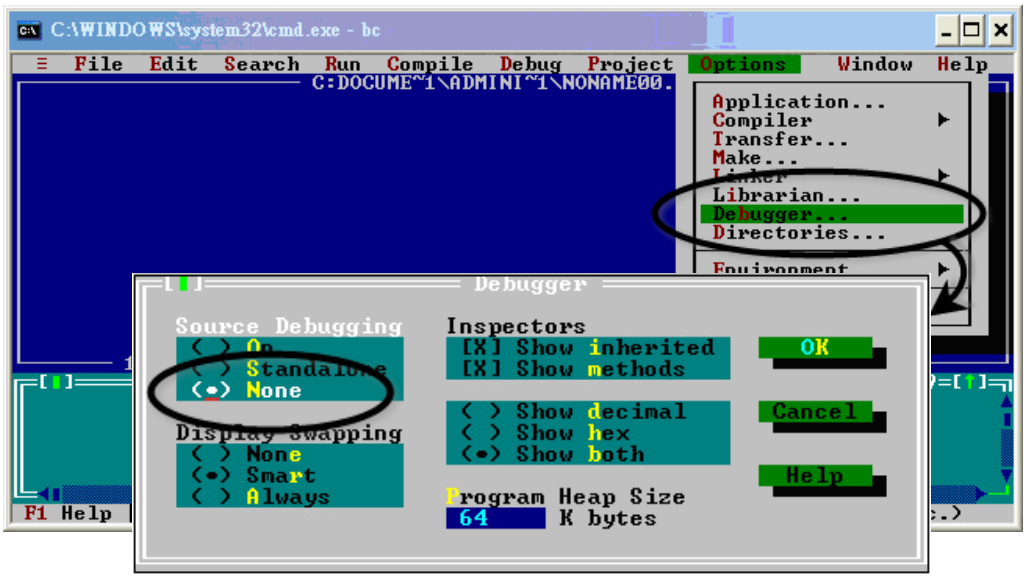



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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



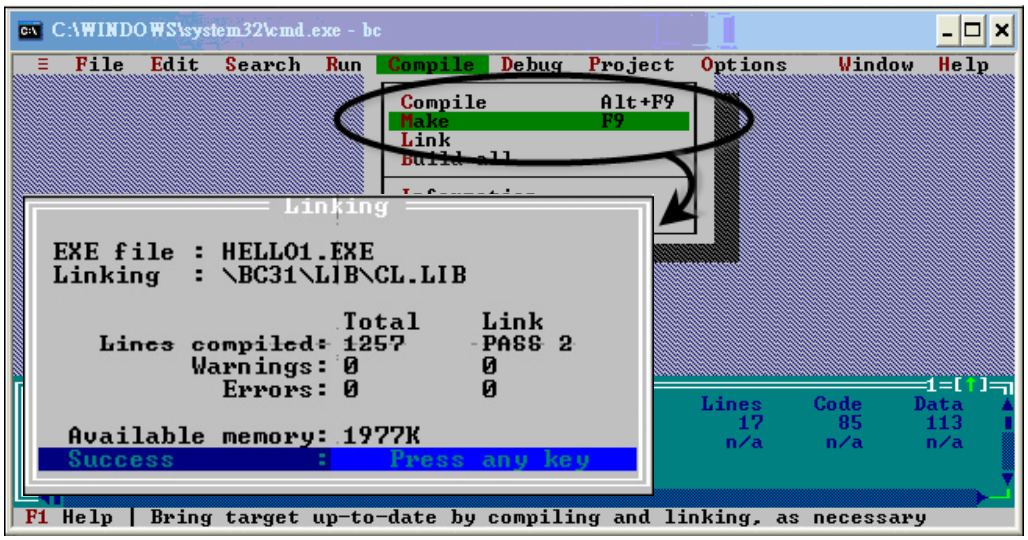

Step 7: Choosing the Debugger…and set the Source Debugging to None






Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Step 8: Make the project

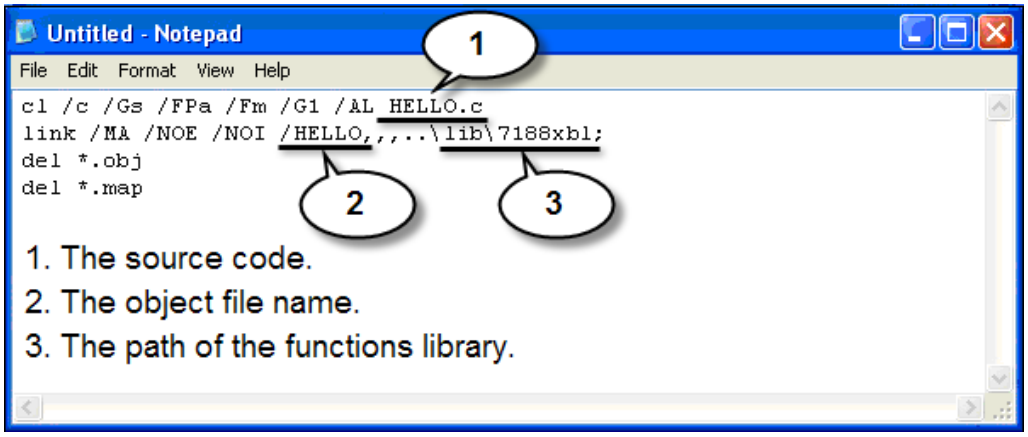





Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# C.3. MSC 6.00

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



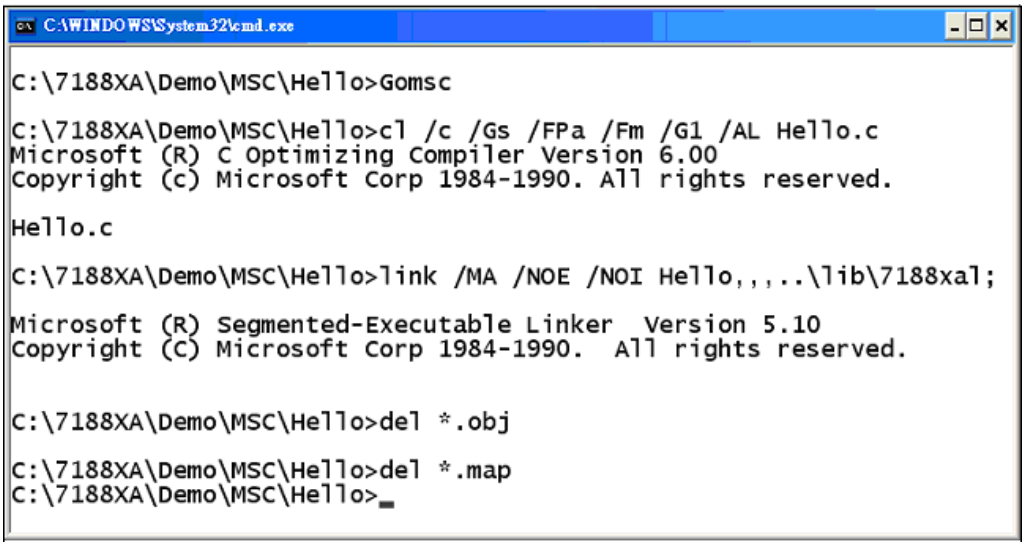

## Tip & Warnings

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

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

/G1: 186 instructions /AL: Large model

Step 2: Run the Gomsc.bat file

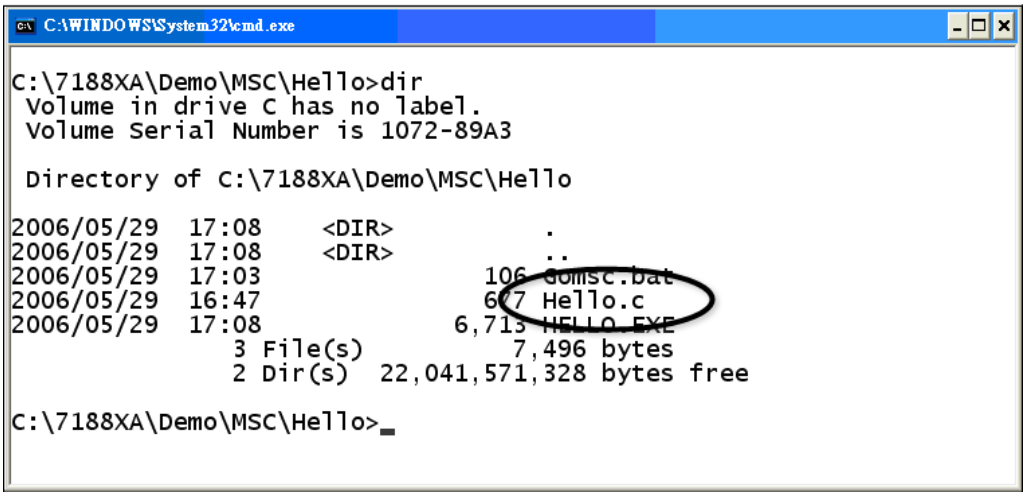



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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

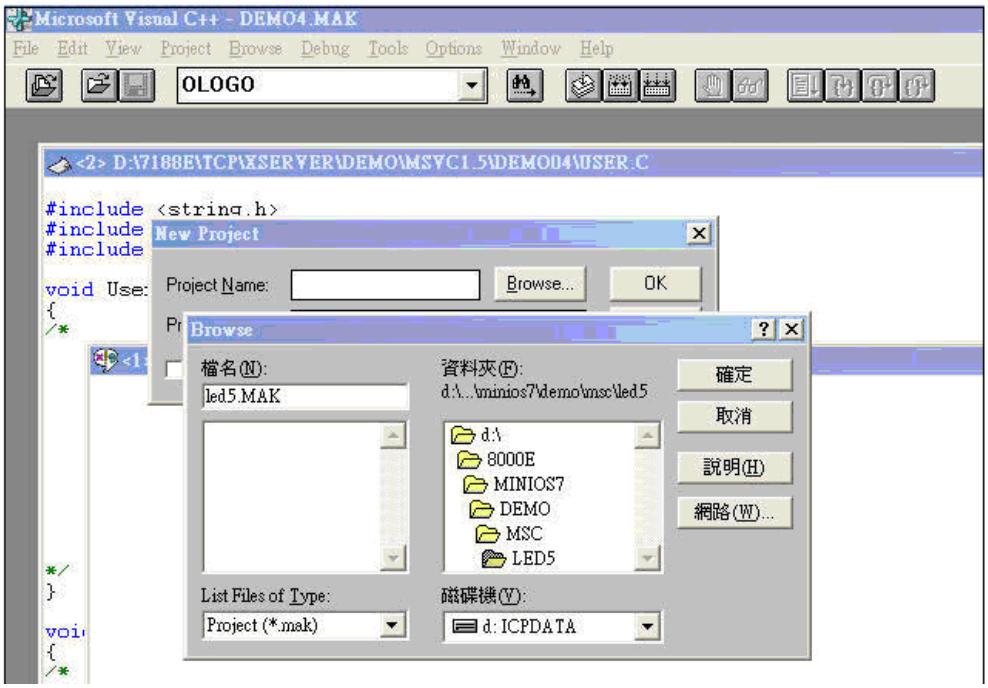




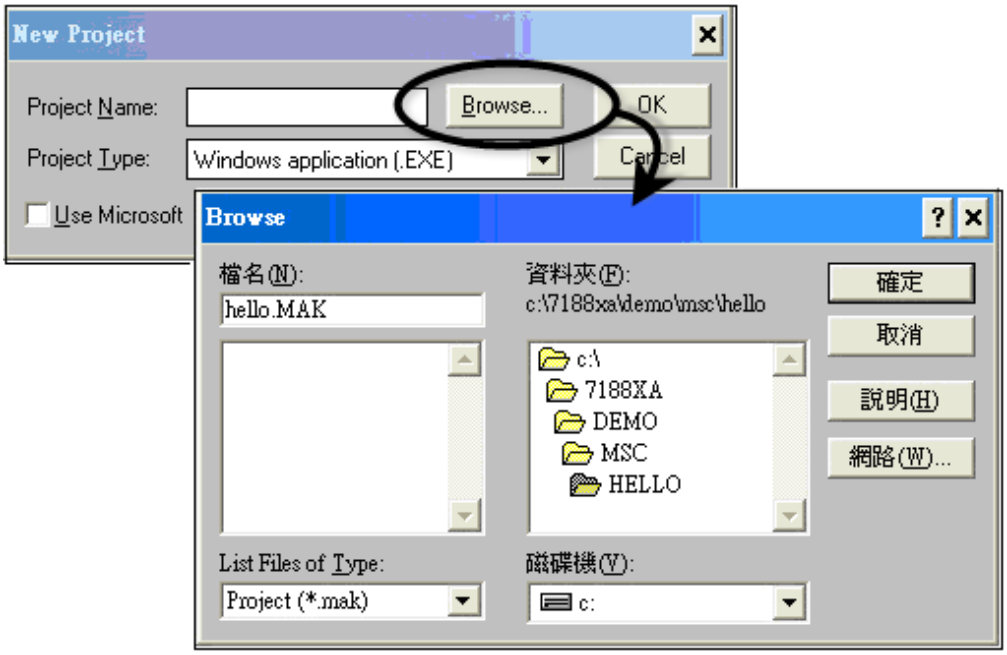
Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# C.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

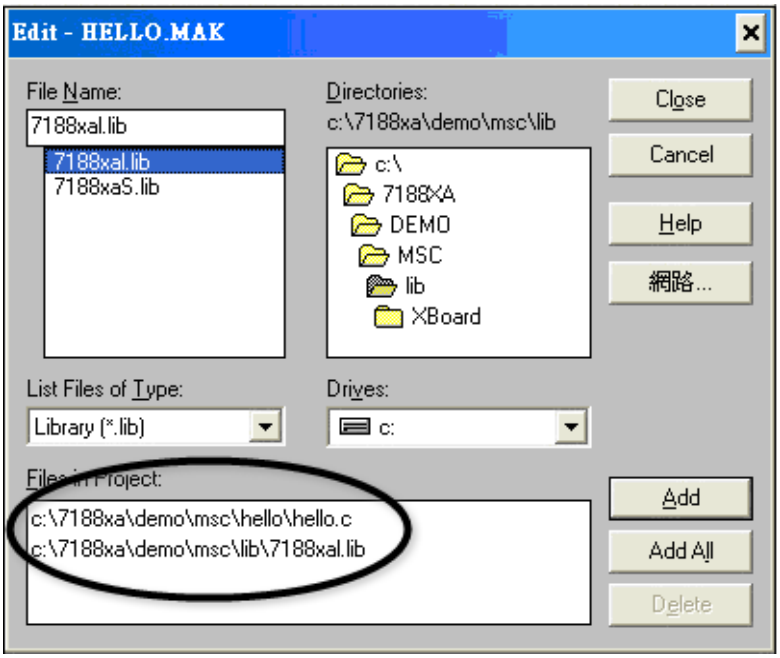



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

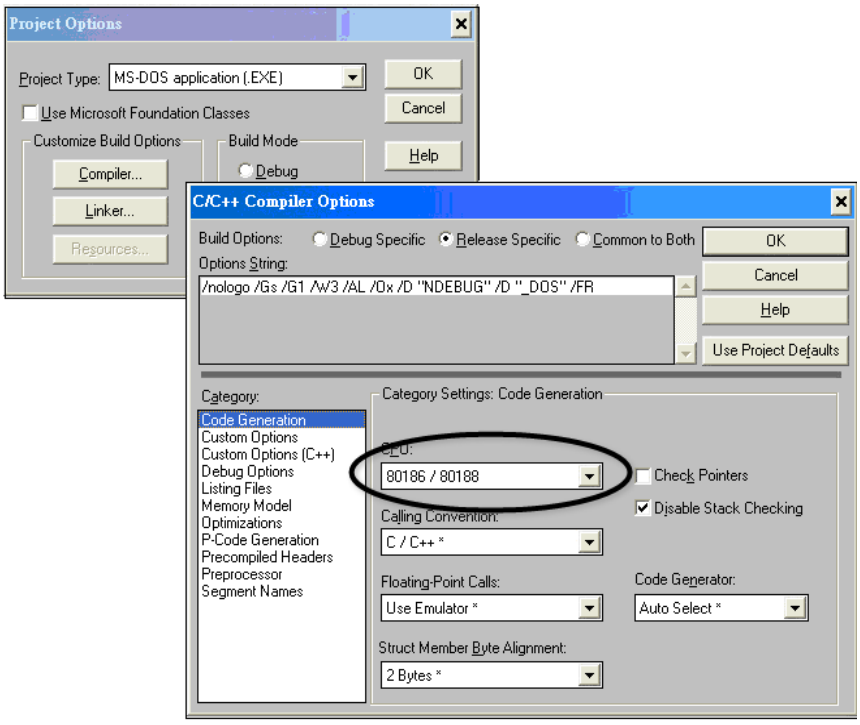


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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




Step 4: Set the Code Generation on the Compiler.

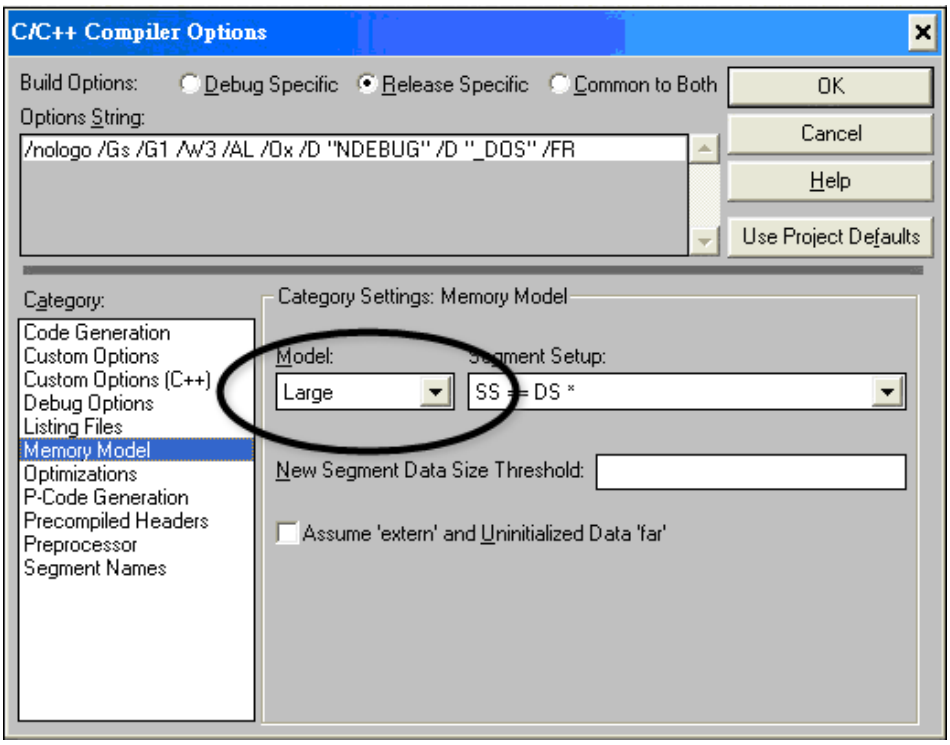



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

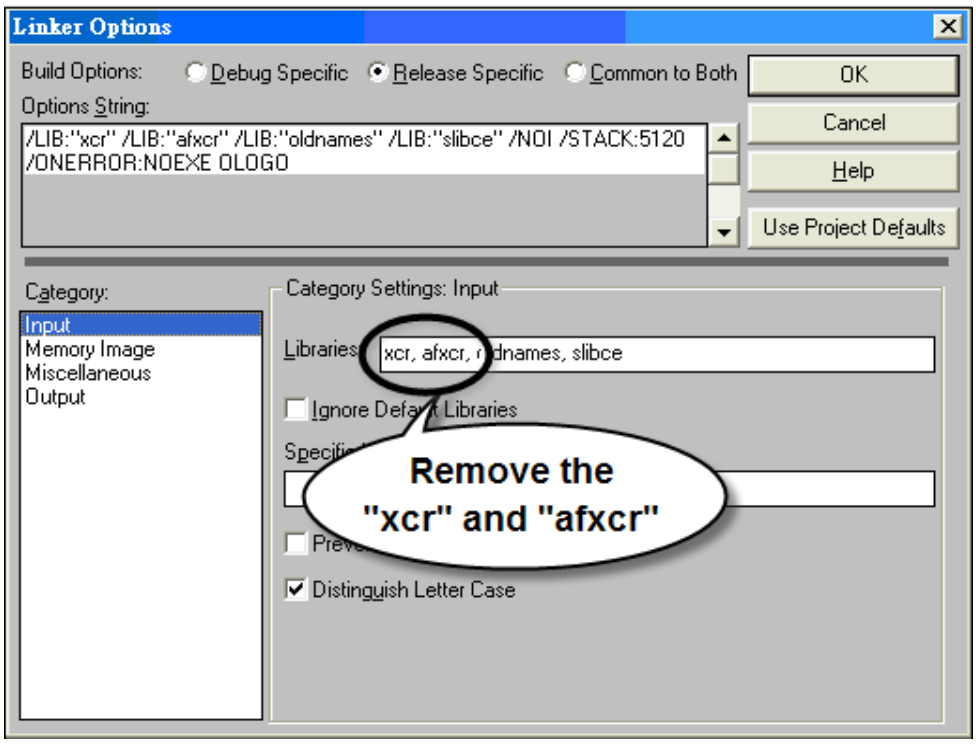


Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

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




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

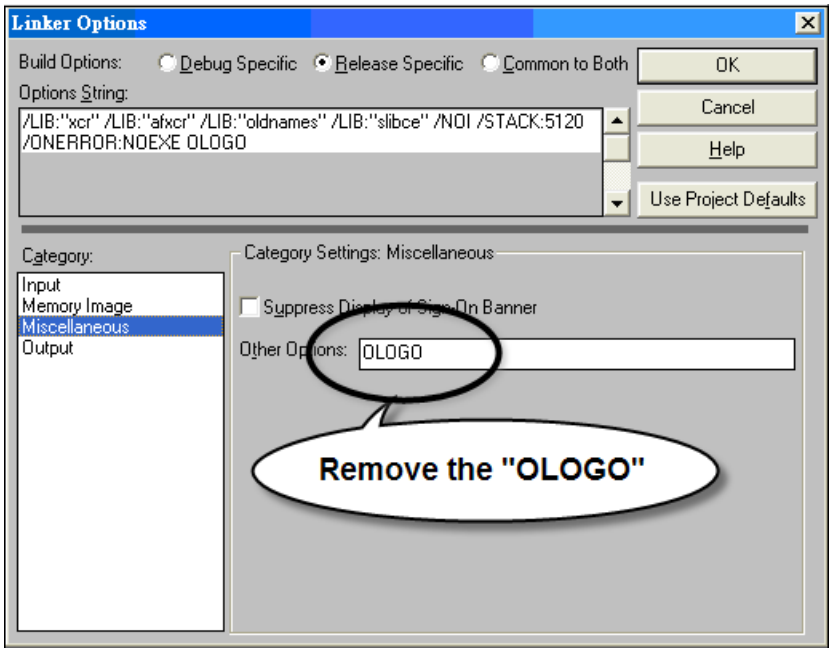




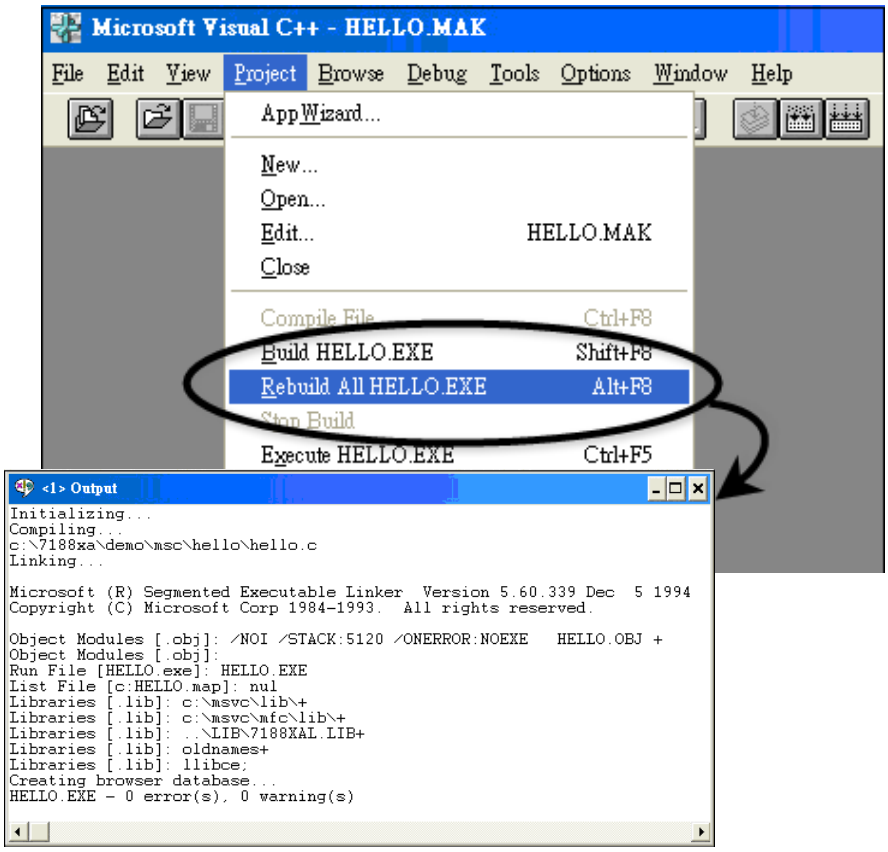

Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

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




Step 8: Rebuild the project



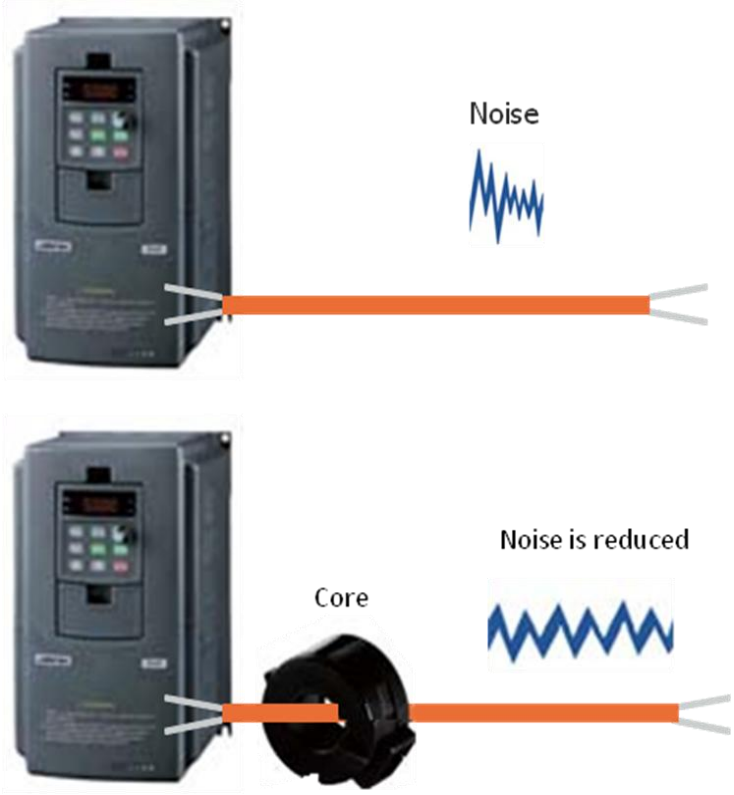

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com

# Appendix D. Core's application and wiring

Core(Ferrite) is useful to reduce ElectroMagnetic Interference(EMI) and anti-noise, it mainly uses for communication interface like RS-232, RS-422, RS-485, CAN Bus, FRNET, PROFIBUS, Ethernet,etc. And it also uses for the cable of power supply side.
The below photo will illustrate how to reduce noise.




The below photos is the wiring of the core on CAN Bus side.




Note: When the communication works normally, using many cores will make the communication error.




Copyright © 2016 ICP DAS Co., Ltd. All Rights Reserved. E-mail: service@icpdas.com