

VP-2000/VH-2000 Series (C Language Based, MiniOS7 Inside) User Manual

Version 1.0.2, August 2010



Service and usage information for

VP-2111

VH-2110

VH-2111P

VH-2211

VH-2311

Warranty

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

Warning

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

Copyright

Copyright © 2009 by ICP DAS Co., Ltd. All rights are reserved.

Trademark

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

Contact US

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

Email: service@icpdas.com

Table of Contents

1. Introduction	6
1.1. ViewPAC Family (MiniOS7 Inside)	7
1.1.1. ViewPAC Module Naming Convention	7
1.1.2. ViewPAC Comparison	8
1.2. Features	9
1.3. Specification	10
1.4. Overview	12
1.5. Dimension	15
1.6. Companion CD	18
2. Getting Started	19
2.1. Hardware Installation	20
2.1.1. Mounting the Hardware	21
2.1.2. Mounting the IP-65 Waterproof connector	27
2.2. Software Installation	29
2.3. Boot Configuration	31
2.4. Uploading ViewPAC Programs	32
2.4.1. Establishing a connection between PC and ViewPAC	33
2.4.1.1. RS-232 connection	34
2.4.1.2. Ethernet Connection	37
2.4.2. Uploading and executing ViewPAC programs	42
2.4.3. Making programs start automatically	43
1.1. Updating the ViewPAC OS image	45

3. “Hello World” – Your First Program	48
3.1. C Compiler Installation	49
3.1.1. Installing the Compiler	50
3.1.2. Setting up the Environment Variables	54
3.2. ViewPAC APIs	57
3.3. First Program in ViewPAC	58
4. APIs and Demo References	69
4.1. API for COM Port	78
4.1.1. Types of COM port functions	80
4.1.2. API for MiniOS7 COM port	81
4.1.3. API for standard COM port	84
4.1.4. Port functions Comparison	87
4.1.5. Request/Response protocol define on COM port	89
4.2. API for I/O Modules (For VP-2111 module only)	90
4.2.1. Steps to use I-8K series I/O modules in slots	92
4.2.2. Steps to use I-87K series I/O modules in slots	93
4.2.3. Steps to use I-7K and I-87K series I/O modules that are connected with COM ports	95
4.3. API for EEPROM	97
4.4. API for Flash Memory	99
4.5. API for NVRAM	101
4.6. API for Timer	103
4.7. API for WatchDog Timer (WDT)	105
4.8. API for MFS (For VP-2111/VH-2111P/VH-2211/VH-2311 modules only)	107

Appendix A. What is MiniOS7	113
Appendix B. What is MiniOS7 Utility	114
Appendix C. What is MiniOS7 File System (MFS)	115
Appendix D. I-8K and I-87K serial Modules (For VP-2111 module only)	119
Appendix E. Application of RS-485 Network	120
E.1. Basic RS-485 Network	121
E.2. Daisy Chain RS-485 Network	122
E.3. Star Type RS-485 Network	123
E.4. Random RS-485 Network	125
E.5. Master/Slaves Settings	126
E.5.1. ViewPAC as a Master (default):	126
E.5.2. ViewPAC as a Slave:	128

1. Introduction

ViewPAC combines iPAC, graphic display and keypad in one unit. It equips an 80186 CPU (16-bit and 80MHz) running a MiniOS7 operating system, several communication interface (Ethernet, RS-232/485), 3 slots to expand I/O modules, STN LCD and a rubber keypad.



Its operating system, MiniOS7, can boot up in a very short time (0.4~0.8 seconds). It has a built-in hardware diagnostic function, and supports the full range of functions required to access all high profile I-8K and I-87K series I/O modules, such as DI, DO, DIO, AI, AO, Counter/Frequency, motion control modules, etc.

Compared with traditional HMI + PLC solutions, ViewPAC reduces overall system cost, space and gives you all the best features of HMIs and PLCs.

1.1. ViewPAC Family (MiniOS7 Inside)

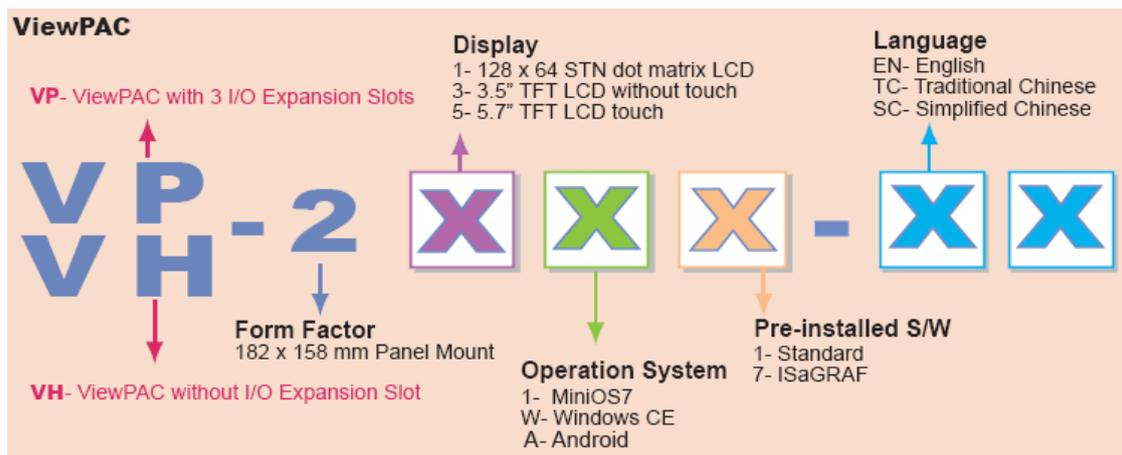
ViewPAC can be divided into five types, according to their features.

- VP-2111
- VH-2110
- VH-2111P (Coming soon)
- VH-2211 (Coming soon)
- VH-2311 (Coming soon)

For their comparison, please refer to section “1.1.2. General Series”

1.1.1. ViewPAC Module Naming Convention

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



1.1.2. ViewPAC Comparison

The following table provides a specification comparison of ViewPAC model.

	VP-2111	VH-2110	VH-211P	VH-2211	VH-2311
OS	MiniOS7				
CPU	80 MHz				
Flash	512 KB				
RAM	768 KB	512 KB	768 KB		
Dual Battery Backup SRAM	512 KB	-	512 KB		
Flash Disk	64 MB	-	64 MB		
STN LCD Resolution	128 x 64				
Ethernet	1		2		
GPS/GPRS	-			GPRS	GPS/GPRS
RS-232/RS-485	3				
I/O Slot	3	-			

1.2. Features

Main features of hardware and software.

Software Features

- MiniOS7 Embedded Operating System (DOS-like)
- C language Based Software Development Toolkit
- Modbus Library Provided
- Hardware Diagnostic Functions
- Load Files via RS-232 or Ethernet

Hardware Features

- 80186, 80 MHz CPU (16-bit)
- IP65 Compliant Front Panel
- STN LCD with Chinese Font
- Rubber Keypad with 24 Keys
- One 10/100M Ethernet Port
- 64-bit Hardware Serial Number for Software Protection
- Operating Temperature: -15 ~ +55 °C

1.3. Specification

	VP-2111	VH-2110	VH-2111P	VH-2211	VH-2311
System Software					
OS	MiniOS7 (DOS-like embedded operating system)				
Program Download Interface	RS-232 (COM1) or Ethernet				
Programming Language	C language				
Compilers to create.exe Files	TC++1.01 (Freeware); TC 2.01 (Freeware); BC++3.1 ~ 5.2x; MSC 6.0;MSC++ (before version 1.5.2)				
CPU Module					
CPU	80186 or compatible (16-bit and 80 MHz)				
SRAM	768 KB	512 KB	768 KB		
Dual Battery Backup SRAM	512 KB (for 5 years retention)	-	512 KB (for 5 years retention)		
Flash	512 KB (100,000 erase/write cycles)				
Flash Disk	64 MB NAND Flash (100,000 erase/write cycles)	-	64 MB NAND Flash (100,000 erase/write cycles)		
EEPROM	16 KB; Data Retention: 40 years; 1,000,000 erase/write cycles				
NVRAM	31 bytes (battery backup, data valid up to 5 year)				
RTC (Real Time Clock)	Provide second, minute, hour, date, day of week, month, year				
64-bit Hardware Serial Number	Yes				
Watchdog Timers	Yes (0.8 second)				
Communication Ports					
Ethernet	RJ-45 x 1, 10/100 Base-TX (Auto-negotiating, Auto MDI/MDI-X, LED indicators)		RS-45 x 2, 10/100 Base-TX (Auto-negotiating, Auto MDI/MDI-X, LED indicators)		
COM0	Internal communication with the high profile I-87K series modules in slots				
COM1	RS-232 (to update firmware) (RXD, TXD and GND); non-isolated				
COM2	RS-485	D2+, D2-; self-tuner ASIC inside			
	Isolation	2500 V _{DC}	-	2500 V _{DC}	
COM3	RS-232/RS-485 (Rx/D, Tx/D, CTS, RTS and GND for RS-232, Data+ and Data- for RS-485); non-isolated	RS-232 (Rx/D, Tx/D, CTS, RTS and GND for RS-232); non-isolated	RS-232/RS-485 (Rx/D, Tx/D, CTS, RTS and GND for RS-232, Data+ and Data- for RS-485); non-isolated		
MMI (Man-Machine Interface)					
LCD	STN, 128 x 64 Dot Matrix LCD				
Display Mode	Text + Graphics				
Text Font	English + Simplified Chinese/Traditional Chinese				
Rubber Keypad	24 keys				
Buzzer	Yes				
LED Indicators	3 Dual-Color LEDs (PWR, RUN, LAN1, L1, L2, L3; L1 ~ L3 for user programmable)	2 Dual-Color LEDs (RUN, LAN1, L1, L2; L1 ~ L2 for user programmable)	3 Dual-Color LEDs (PWR, RUN, LAN1, L1, L2, L3; L1 ~ L3 for user programmable)		
I/O Expansion Slots					
Slot Number	3 (For High Profile I-8K and I-87K Modules Only)	-			
Hot Swap * will be available	For High Profile I-87K Modules Only	-			
Data Bus	8/16 bits	-			
Address Bus Range	2 K for each slot	-			
GSM/GPRS					
Band	-	850/900/1800/1900 MHz			
GPRS Multi-slot	-	Class 10/8			
GPRS Mobile Station	-	Class B			

GPRS Class 10	-	Max. 85.6 kbps			
CSD	-	Up to 14.4 kbps			
Compliant to GSM phase 2/2+	-	Class 4 (2W @ 850/900 MHz); Class 1 (1W @ 1800/1900 MHz)			
Coding Schemes	-	CS 1, CS 2, CS 3, CS 4			
SMS	-	Text and PDU mode			
GPS					
Channels	-	16 channels all-in-view tracking			
Sensitivity	-	-159 dBm			
Acquisition Rate	-	Cold start: 42 seconds; warm start: 35 seconds; reacquisition rate: 0.1 second			
Accuracy	-	Position: 25 m CEP (S/A off); Velocity: 0.1 second (S/A off); Time: ± 1 ms			
Protocol	-	NMEA			
Mechanical					
Dimensions (W x H x D)	182 mm x 158 mm x 125 mm				
Installation	Panel mounting				
Ingress Protection	Front panel: IP 65				
Environmental					
Operating Temperature	-15 ~ +55 °C				
Storage Temperature	-30 ~ +80 °C				
Ambient Relative Humidity	10 ~ 90 % RH (non-condensing)				
Power					
Input	PoE	-	IEEE 802.3af, Class 3	-	
Range	Terminator Block	+10 ~ +30 V _{DC}	+12 ~ +48 V _{DC}	+18 ~ +55 V _{DC}	+10 ~ +30 V _{DC}
Isolation		1 kV	-	1 kV	
Capacity		3 A, 5 V supply to I/O expansion slots			
Consumption		6 W (0.25 A @ 24 V)	3.6 W (0.15 A @ 24 V)	6 W (0.25 A @ 24 V)	9.6 W (0.4 A @ 24 V) 10.8 W (0.45 A @ 24 V)

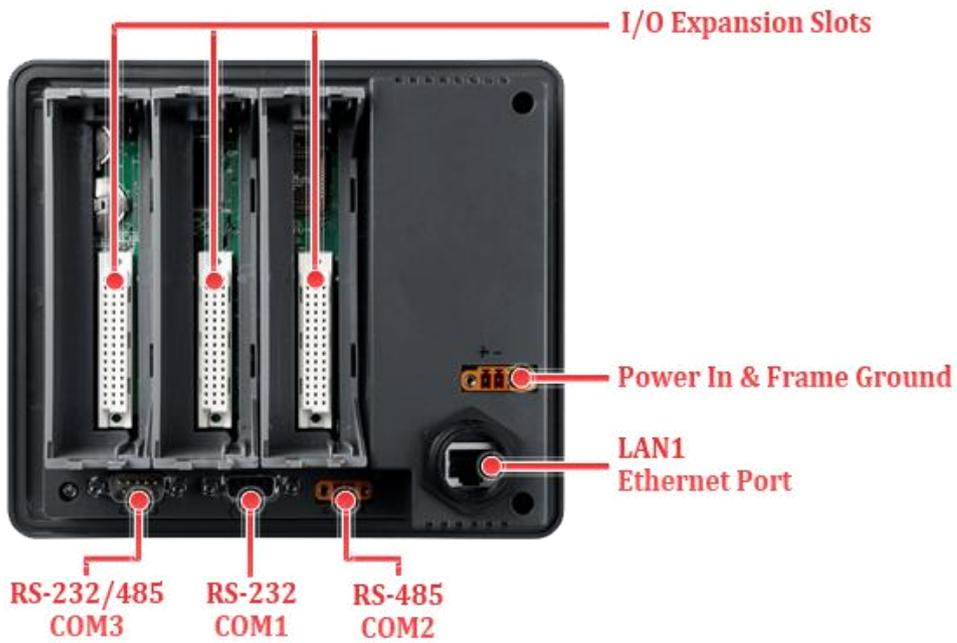
1.4. Overview

Here is a brief overview of the components.

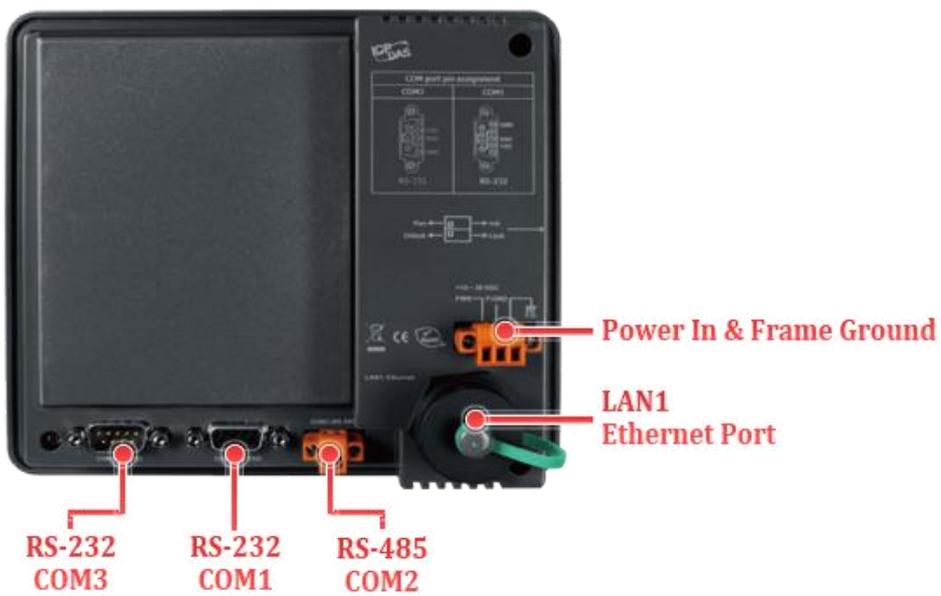
► VP-2000/VH-2000 Series



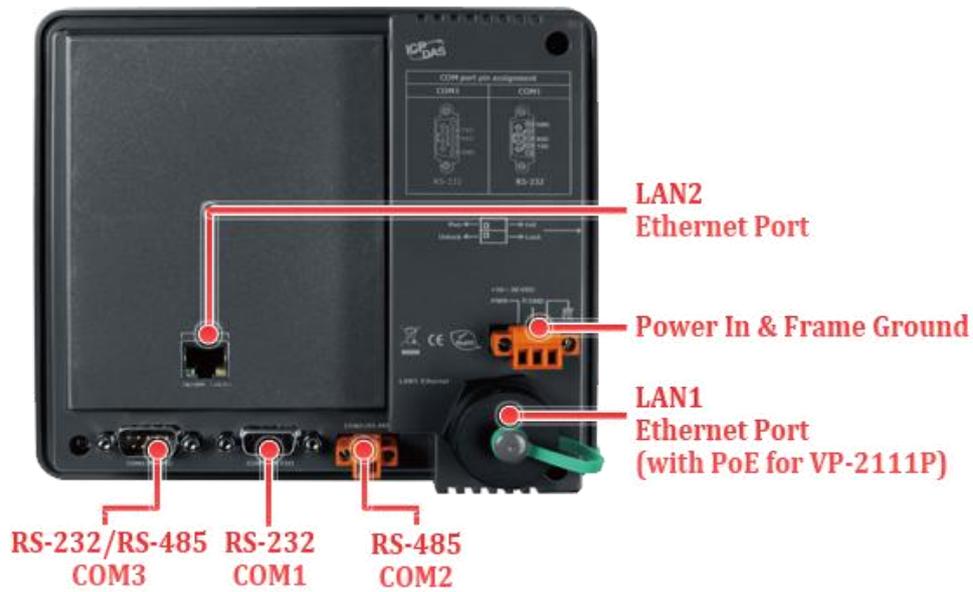
► VP-2111



► VH-2110



➤ **VH-2111P/VH-2211/VH-2311**

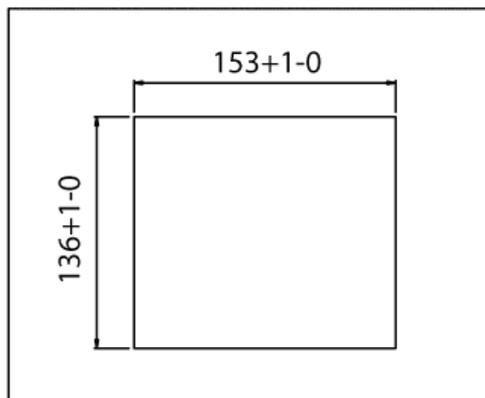


1.5. Dimension

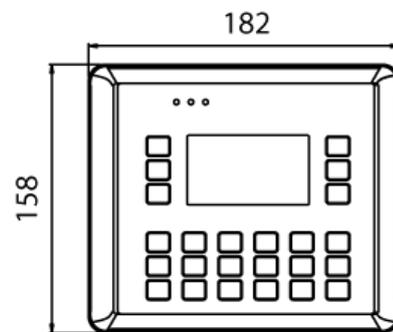
All dimensions are in millimeters.

► VP-2000/VH-2000 Series

Recommended Panel Cut-Out

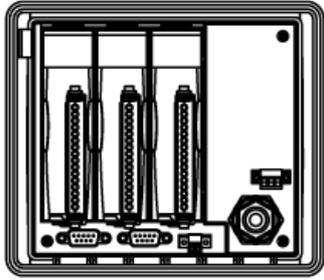


Front View

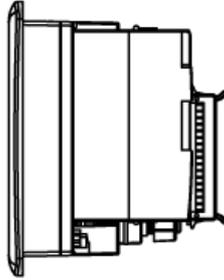


► VP-2000

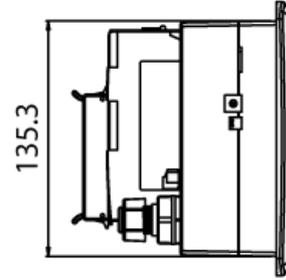
Back View



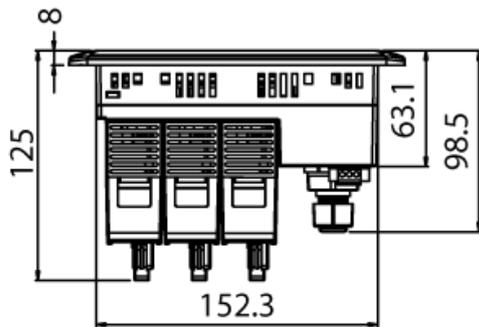
Left Side View



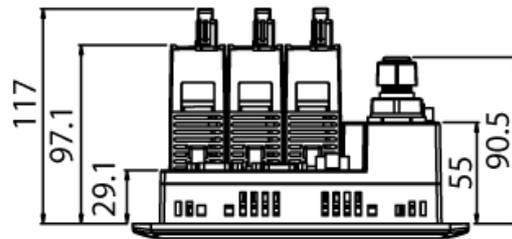
Right Side View



Top View

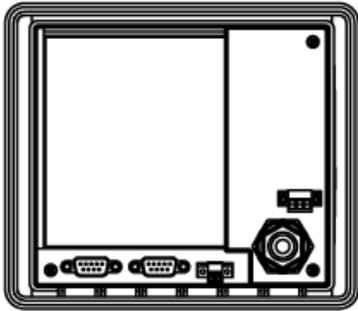


Bottom View

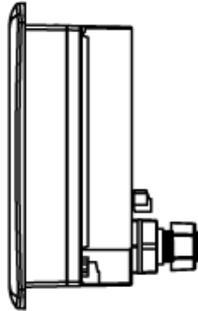


► VH-2000

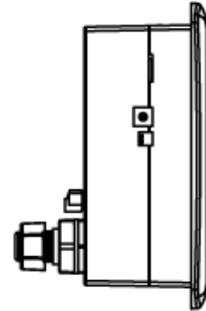
Back View



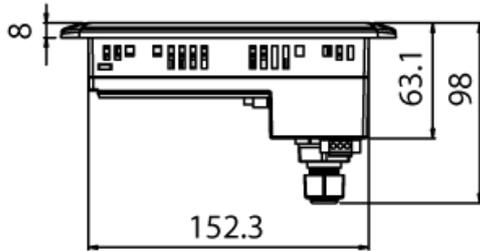
Left Side View



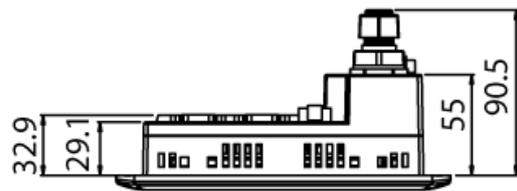
Right Side View



Top View

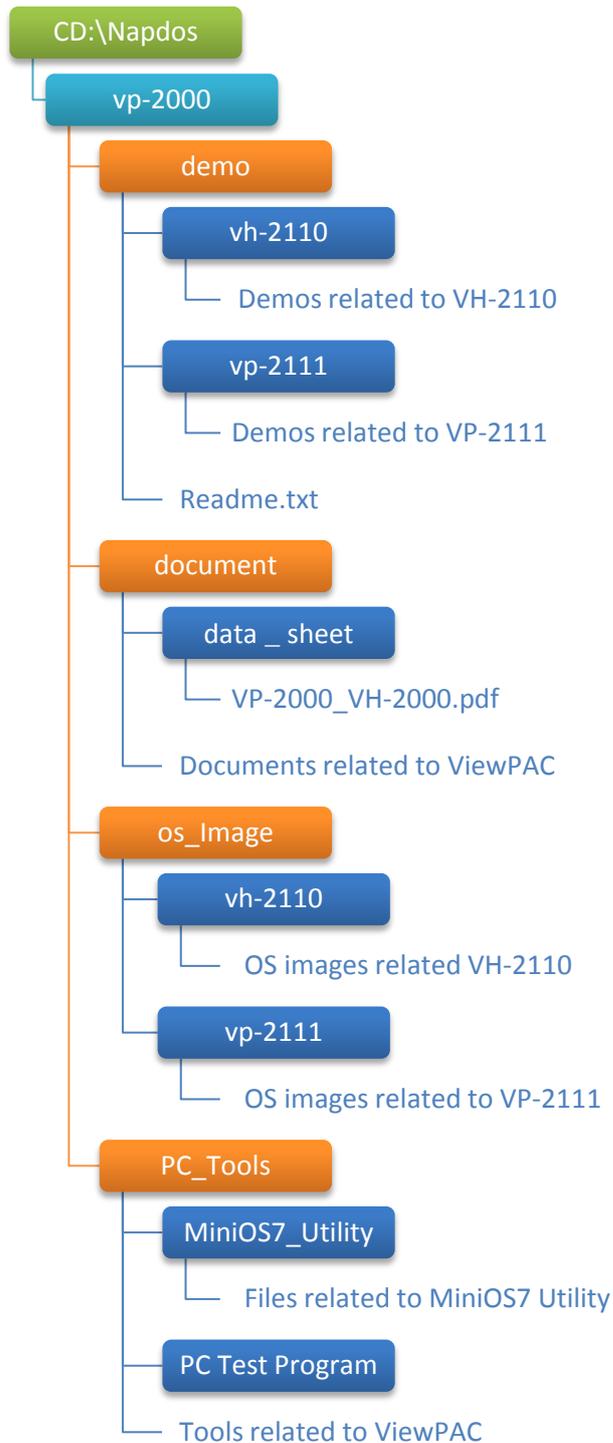


Bottom View



1.6. Companion CD

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



2. Getting Started

If you are a new user, begin with this chapter, it includes a guided tour that provides a basic overview of installation and configuration.

In addition to Quick Start Guide, the package includes the following items, if any items are damaged or missing, please contact us.



**VP-2111/VP-2111-TC
VH-2110/VH-2110-TC**



RJ-45 Waterproofing Kit



Software Utility CD



**CA-0915
RS-232 Cable**



Panel Clips *5



Screw Driver



Screw *5

2.1. Hardware Installation

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

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

You also need to know the expansion capacities in order to choose the best expansion module for achieving maximal efficiency.

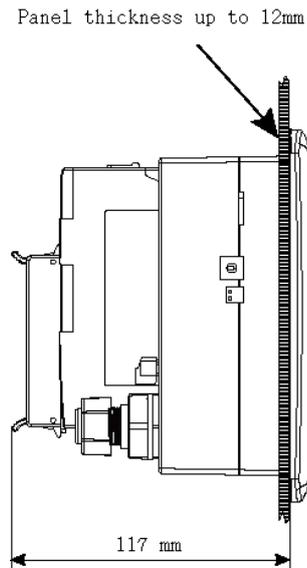
For more information about expansion module that are compatible with the unit, please refer to

http://www.icpdas.com/products/PAC/viewpac/IO_Expansion.htm

2.1.1. Mounting the Hardware

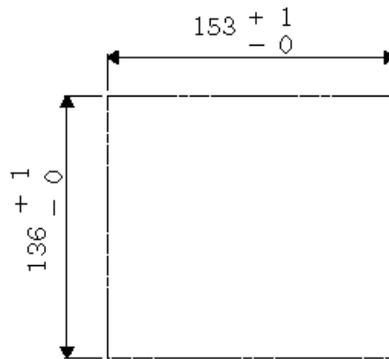
ViewPAC can be mounted in a panel of maximum thickness 12mm.

Adequate access space can be available at the rear of the instrument panel for wiring and servicing purposes.



Below are step-by-step instructions for mounting the ViewPAC hardware.

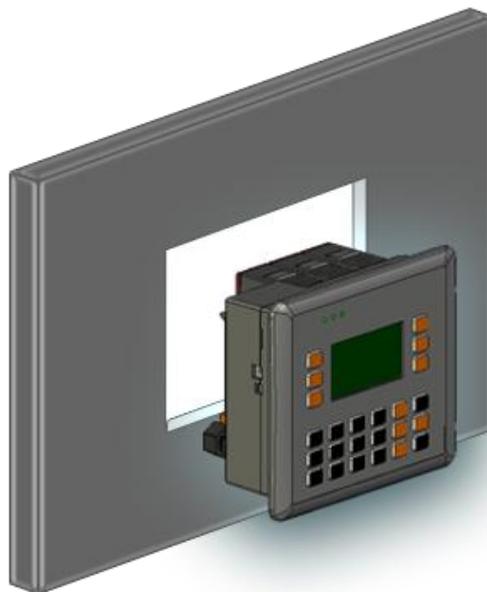
Step 1: Prepare the panel cut-out to the size as below shown



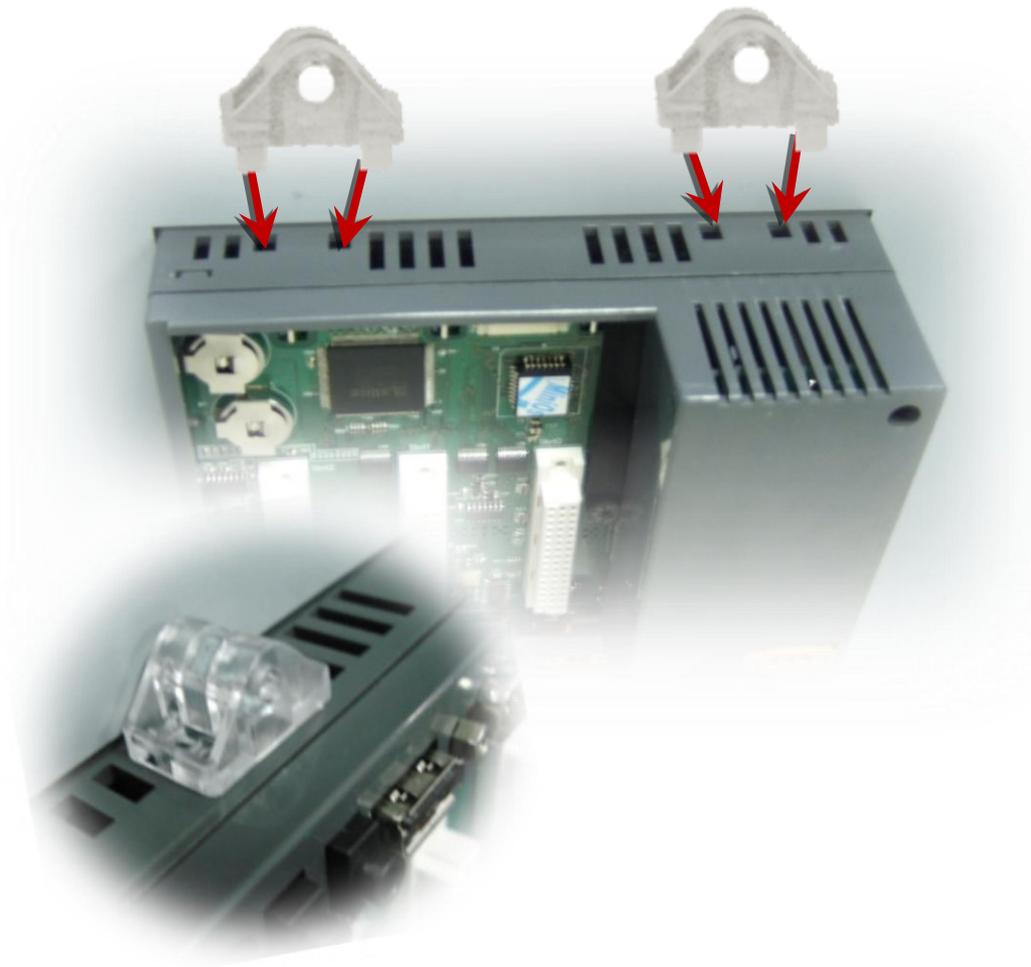
Recommended
Panel Cut-Out

Take care not to cover ventilation holes in the top, bottom and sides of the instrument.

Step 2: Insert the ViewPAC through the panel cut-out

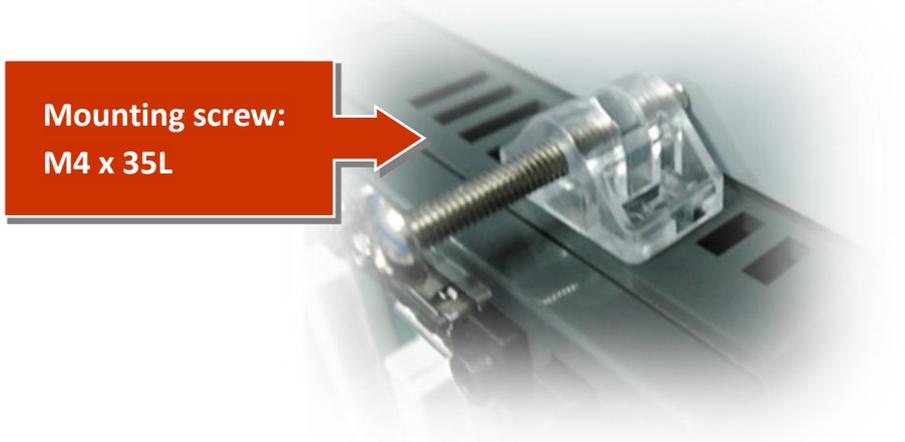


Step 3: Install the panel mounting clips in the View PAC of the upper and lower panel surface



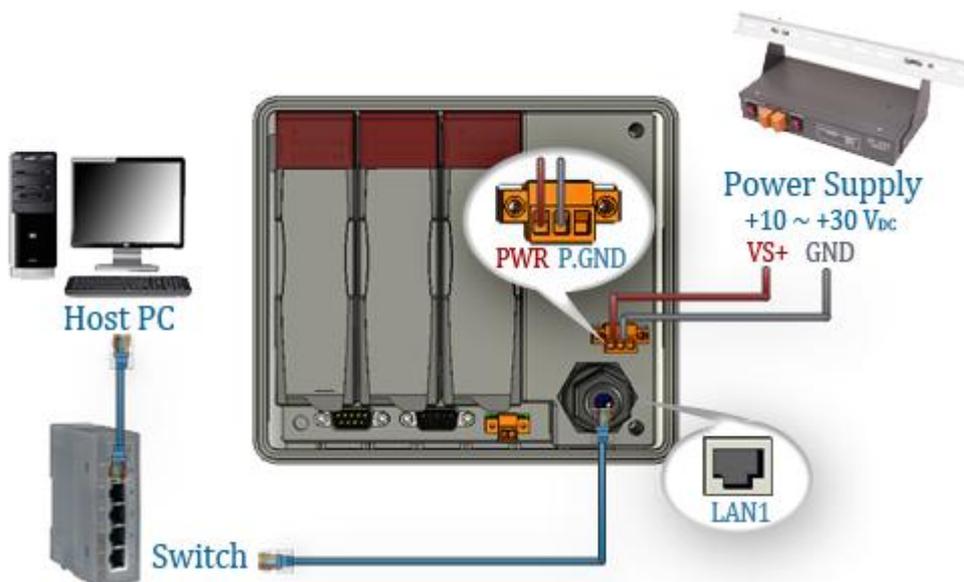
Step 4: Screw the panel mounting clips to the panel

**Mounting screw:
M4 x 35L**

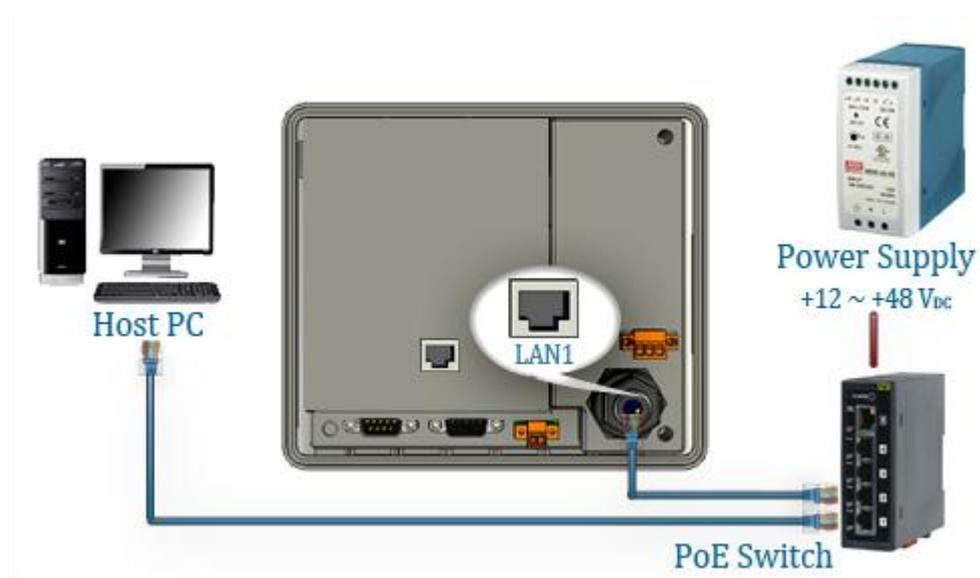


Step 5: Connect the ViewPAC to PC and setting up the power supply

- i. Connect PC to LAN port of ViewPAC.
- ii. Connect the power supply (10 ~ 30 V) to PWR1 and GND terminals of ViewPAC

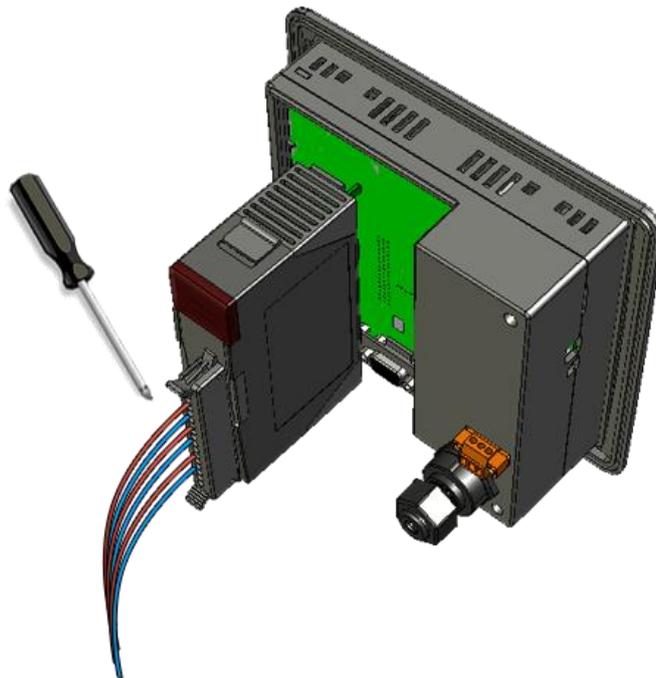


VP-2111P has a PoE Ethernet port. The power can be received from a PoE switch.



Step 6: Inserting the I/O modules (for VP-2111 module only)

It is recommended that the power to the VP-2111 is switched off when wiring the I/O module which are plugging in the VP-2111 slots.



For more information about expansion module that are compatible with the ViewPAC, please refer to

http://www.icpdas.com/products/PAC/viewpac/IO_Expansion.htm

Tips & Warnings



By I-8K and I-87K series expansion modules, support is provided only in High Profile series.

2.1.2. Mounting the IP-65 Waterproof connector

The ViewPAC provides an IP-65 waterproof connector which consists of the following components plugged in RJ-45 cable.

Below are step-by-step instructions for mounting the IP-65 Waterproof connector.





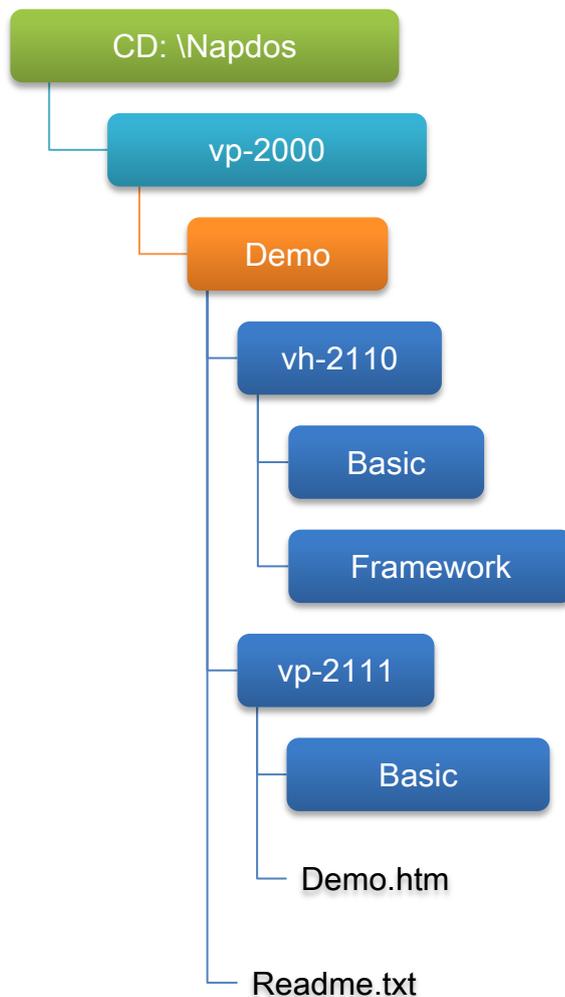
2.2. Software Installation

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

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

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

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



Step 2: Install the MiniOS7 Utility



MiniOS7_Utility_V321.exe
[MiniOS7 Utility Ver 3.21] Setup

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

The MiniOS7 Utility can be obtained from companion CD or our FTP site:

CD:\Napdos\minios7\utility\minios7_utility\

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

2.3. Boot Configuration

Before you upload some programs to ViewPAC, you need to enter the Init mode and disable the Write Protection.

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



2.4. Uploading ViewPAC Programs

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

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

The upload process has the following main steps:

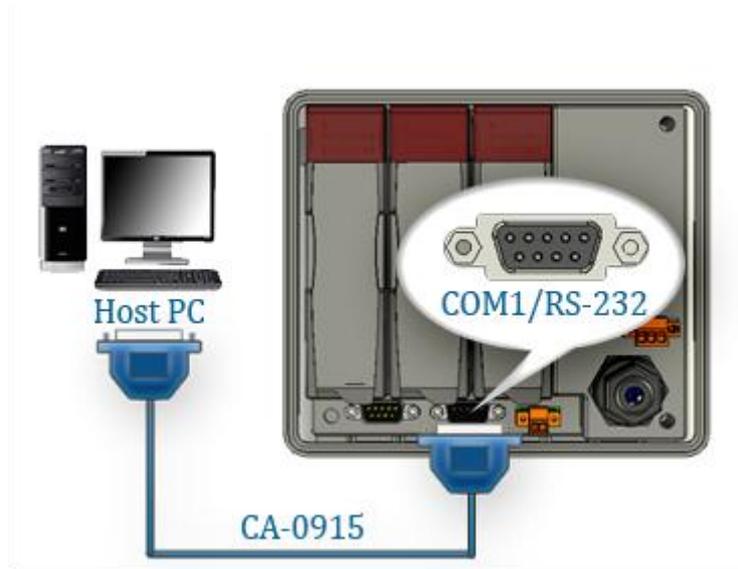
1. Establishing a connection between PC and ViewPAC
2. Uploading and executing programs on ViewPAC
3. Making programs start automatically

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

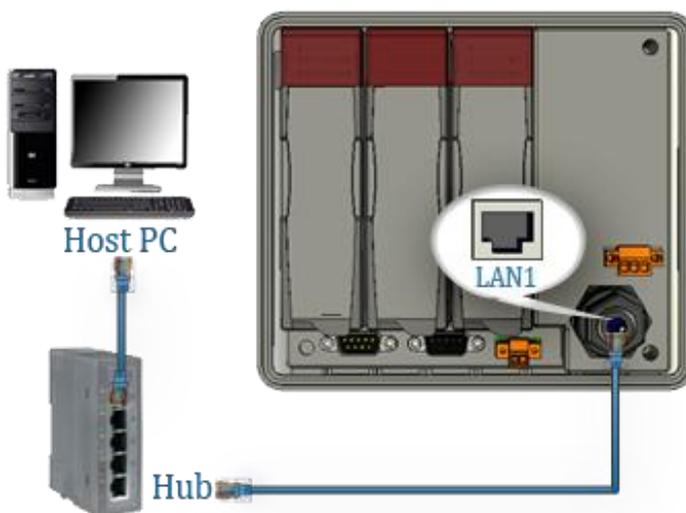
2.4.1. Establishing a connection between PC and ViewPAC

There are two ways to establish a connection between PC and ViewPAC.

► RS-232 Connection



► Ethernet Connection



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

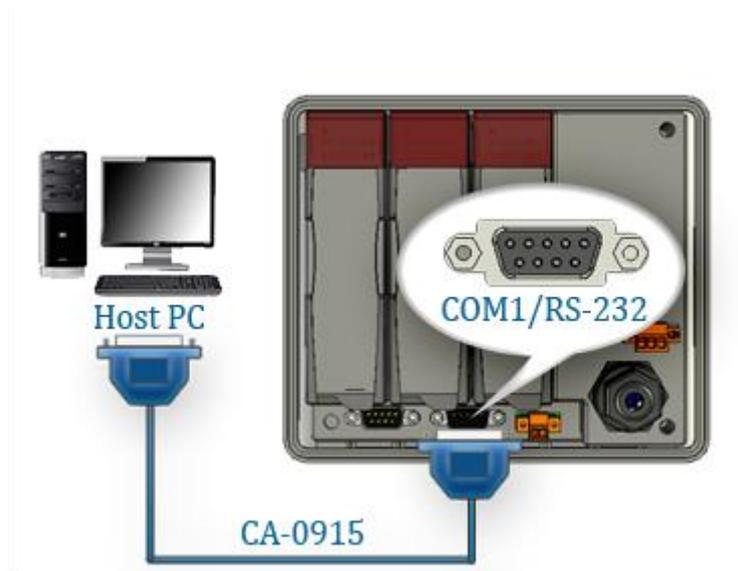
2.4.1.1. RS-232 connection

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

Step 1: Turn the switch of the Lock to “ON” position, and the switch of Init to “ON” position

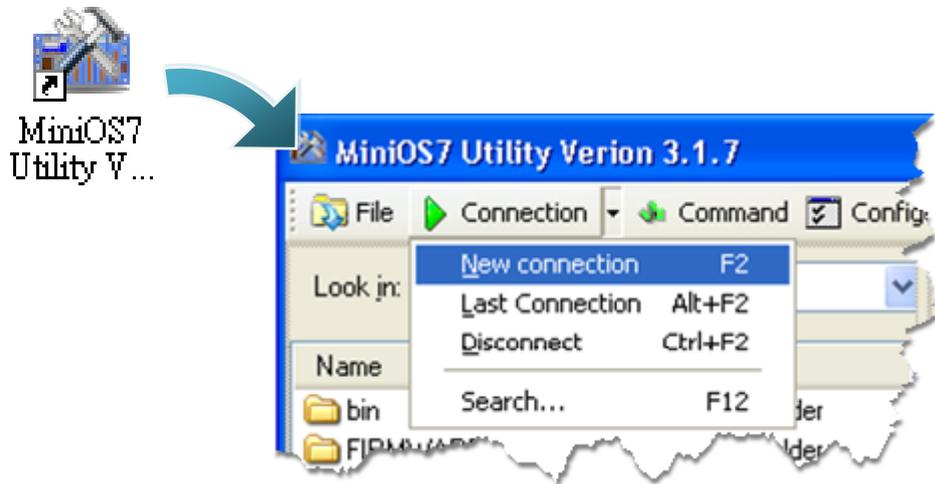


Step 2: Use the RS-232 Cable (CA-0915) to connect to PC

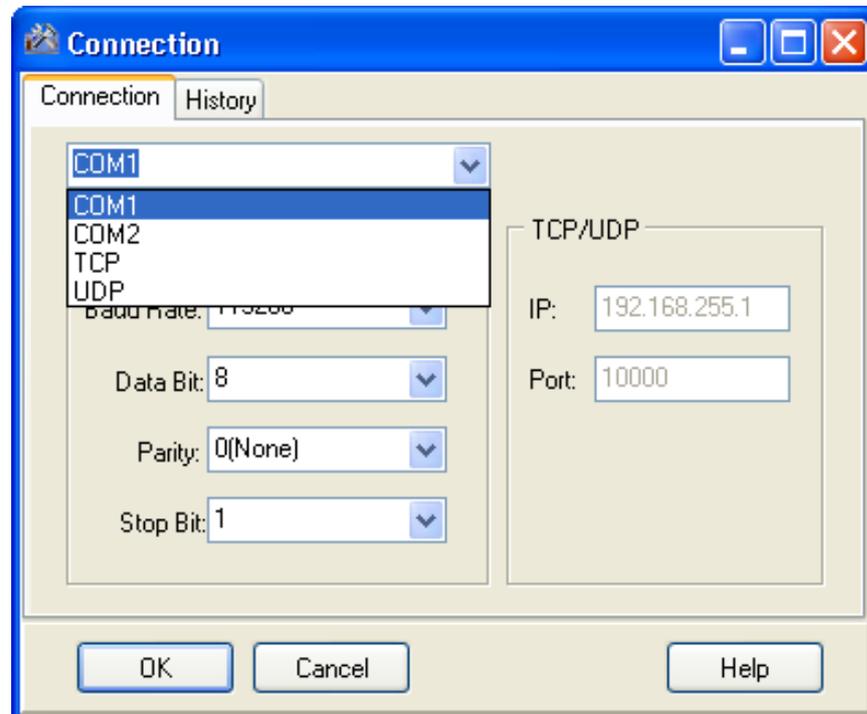


Step 3: Run the MiniOS7 Utility

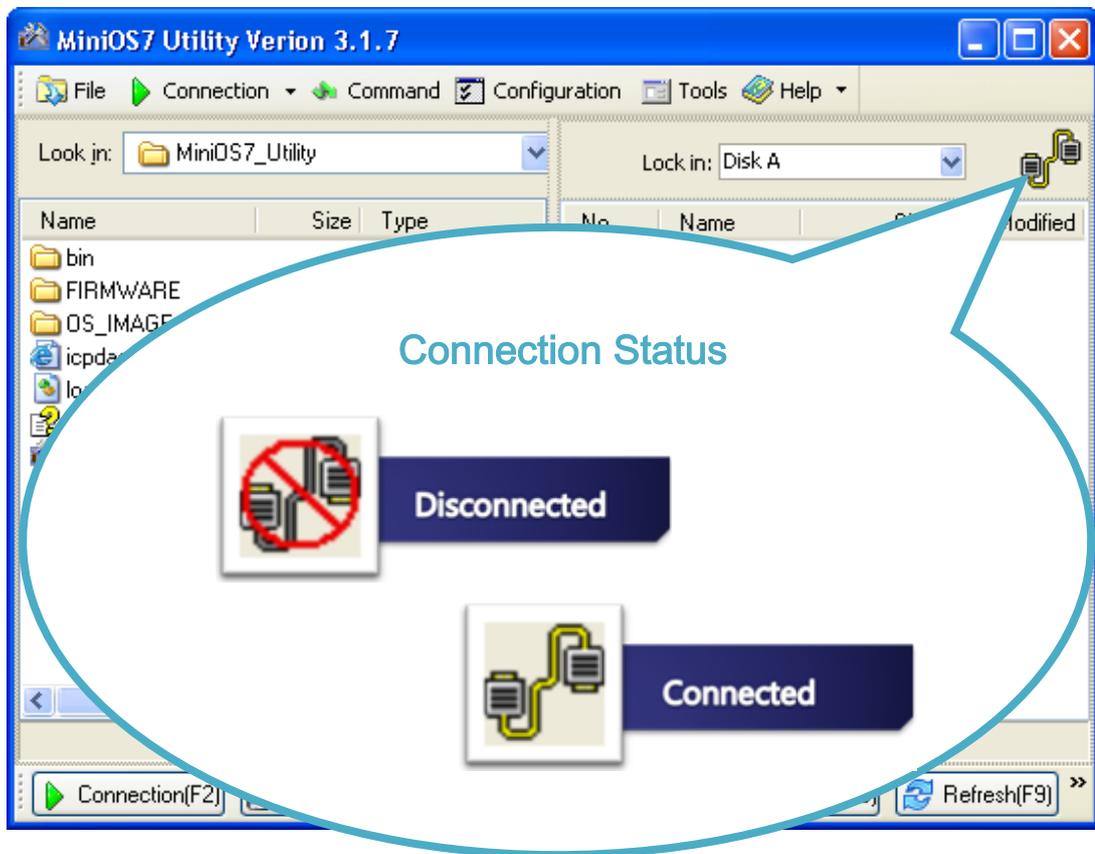
Step 4: Click the “New connection” function from the “Connection” menu



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



Step 6: The connection has already established



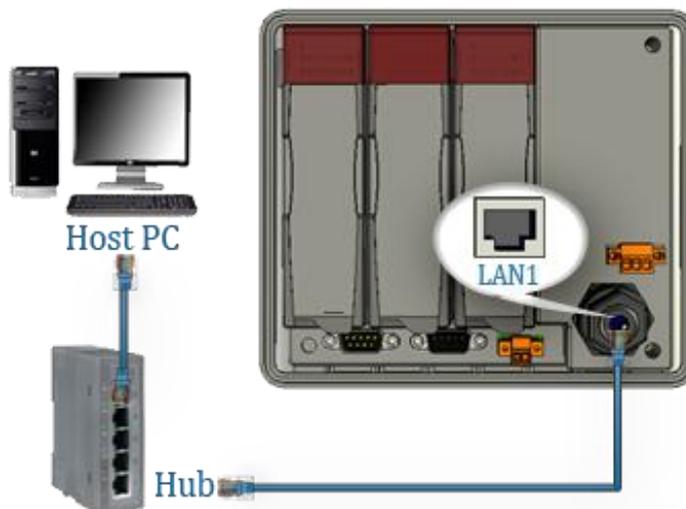
2.4.1.2. Ethernet Connection

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

Step 1: Turn the switch of the Unlock to “ON” position, and the switch of Init to “ON” position



Step 2: Use an Ethernet cable to connect to PC



Step 3: Run the MiniOS7 Utility

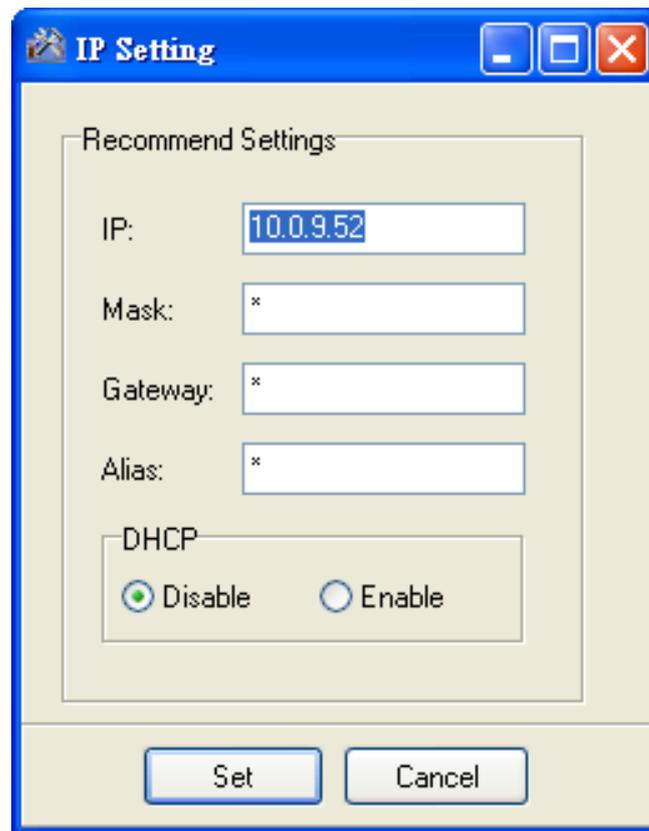
Step 4: Click the “Search” function from the “Connection” menu



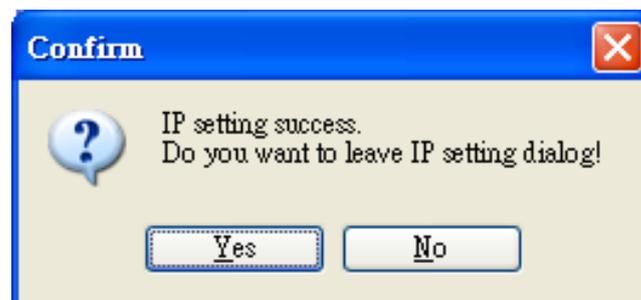
Step 5: On the “MiniOS7 Scan” dialog box, choose the module name from the list and then choose “IP setting” from the toolbar



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



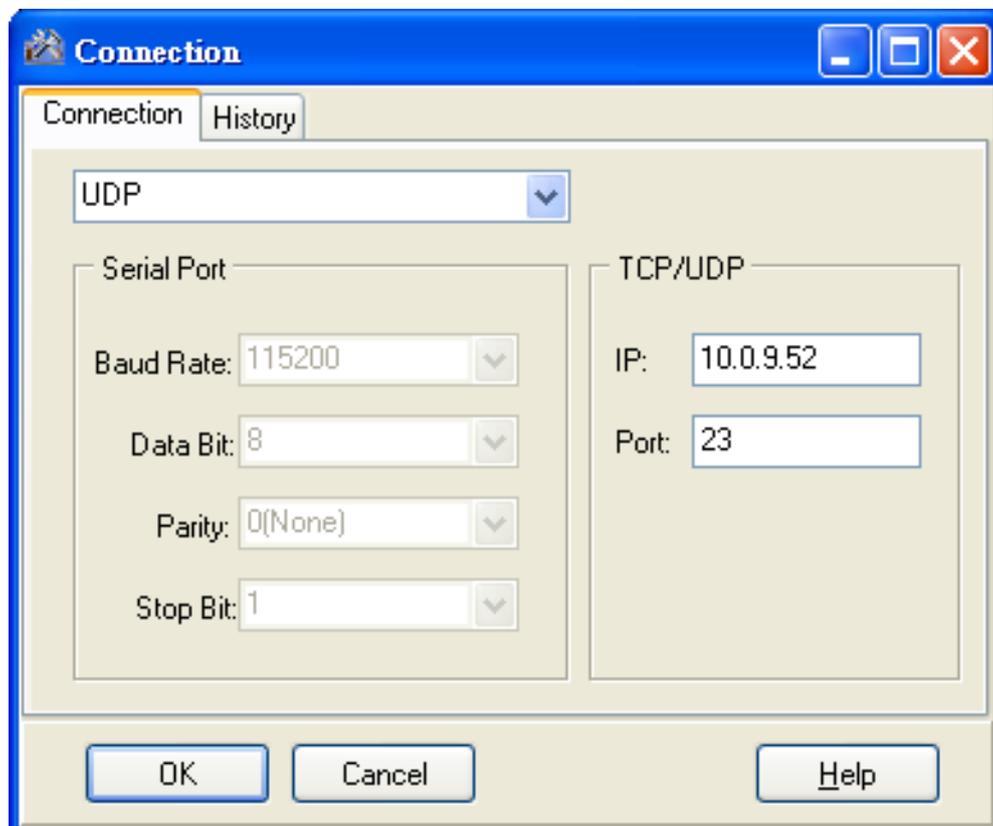
Step 7: On the “Confirm” dialog box, click “Yes”



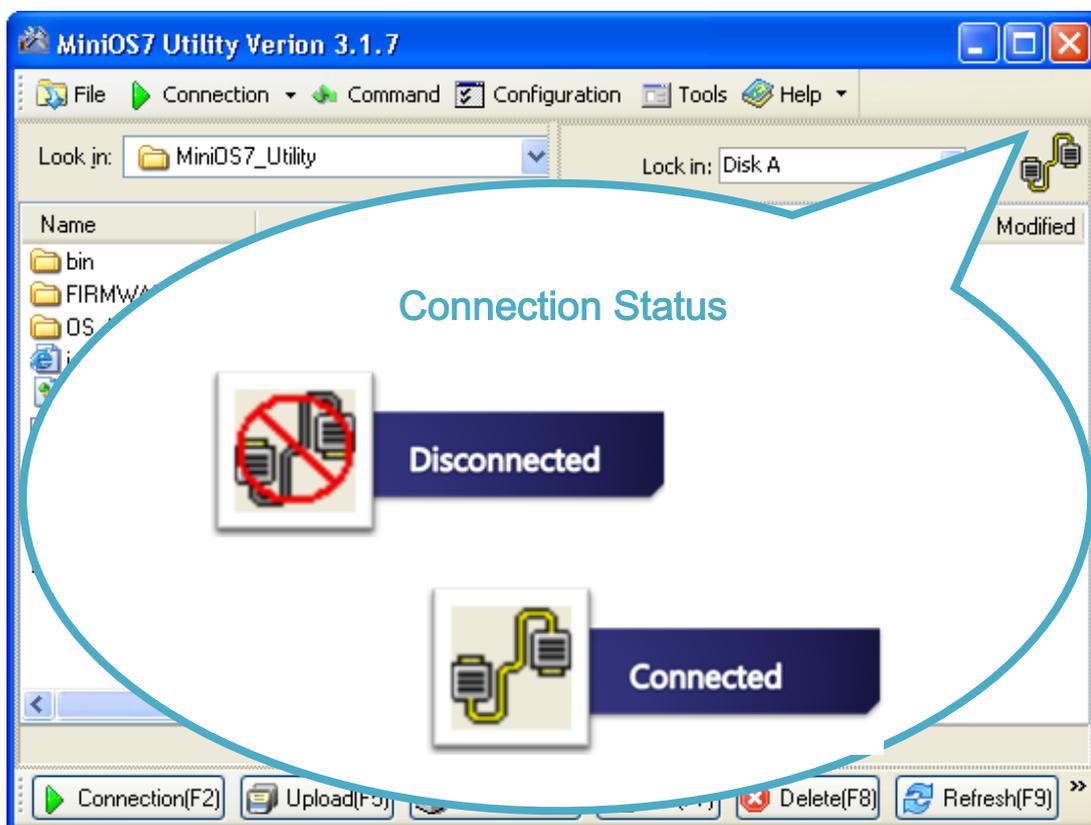
Step 8: Click the “New connection” function from the “Connection” menu



Step 9: On the “Connection” tab of the “Connection” dialog box, select “UDP” from the drop down list, type the IP address which you are assigned, and then click “OK”



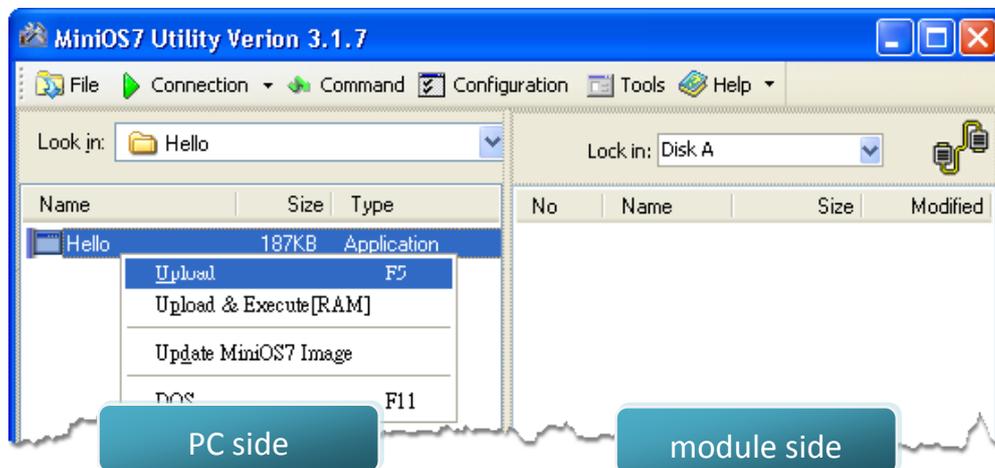
Step 10: The connection has already established



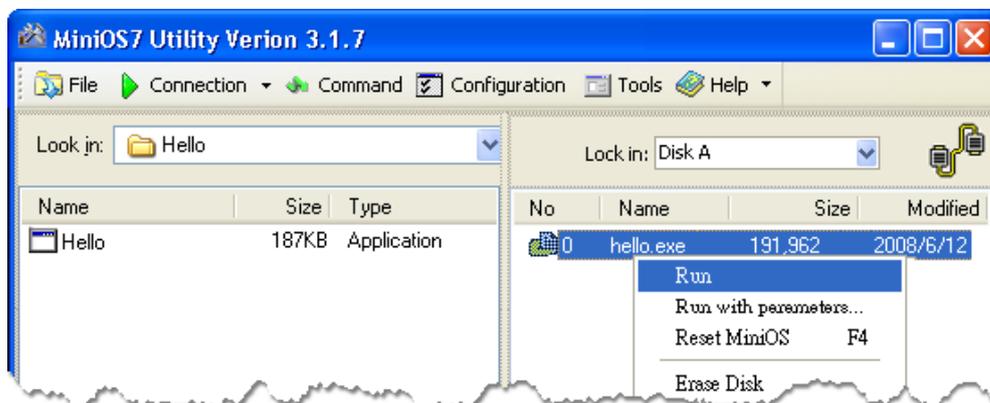
2.4.2. Uploading and executing ViewPAC programs

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

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



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



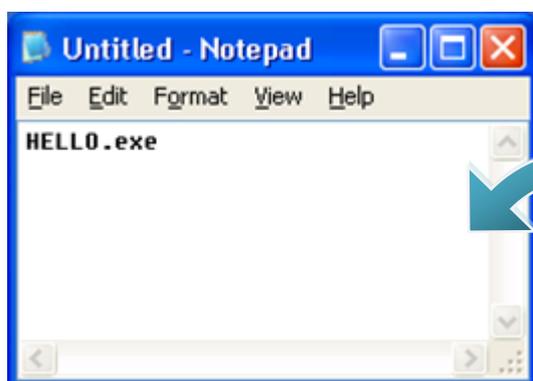
2.4.3. Making programs start automatically

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

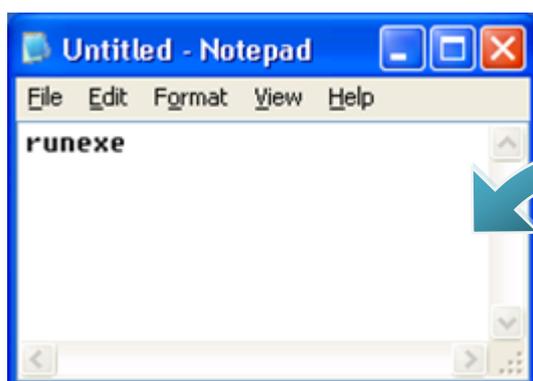
For example, to make the program “hello” run on start-up.

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



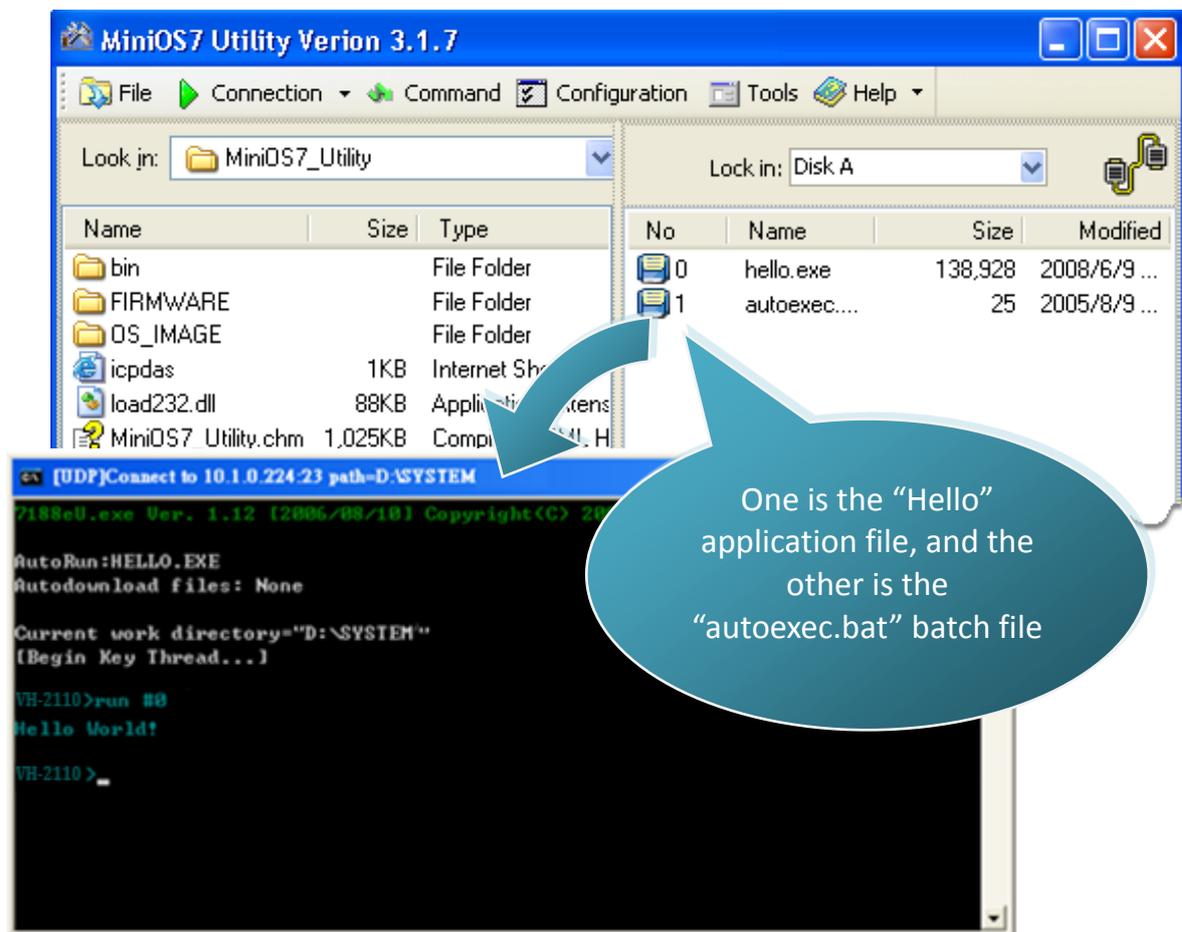
The file name:
Run the specified file.



Runexe:
Run the last exe file.

Step 2: Upload programs to ViewPAC using MiniOS7 Utility

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



Tips & Warnings



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



2.5. Updating the ViewPAC OS image

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

Step 1: Get the latest version of the ViewPAC OS image



The latest version of the ViewPAC OS image can be obtained from:

VP-2111:

CD:\NAPDOS\vp-2000\os_image\vp-2111\

http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/os_image/vp-2111/

VH-2110:

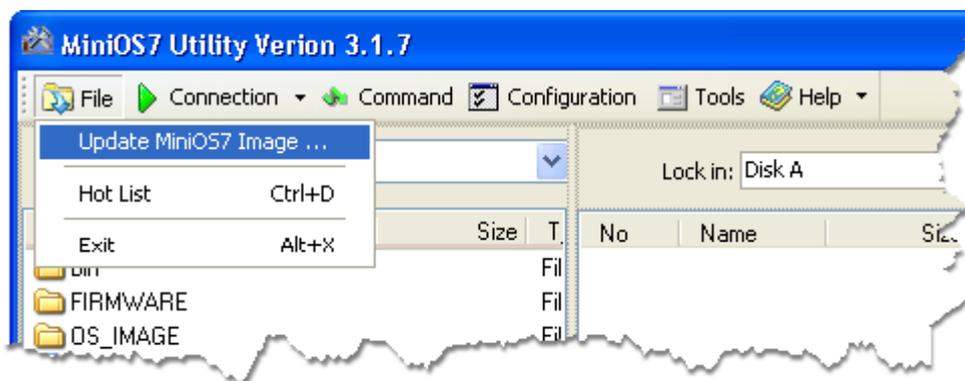
CD:\NAPDOS\vp-2000\os_image\vh-2110\

http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/os_image/vh-2110/

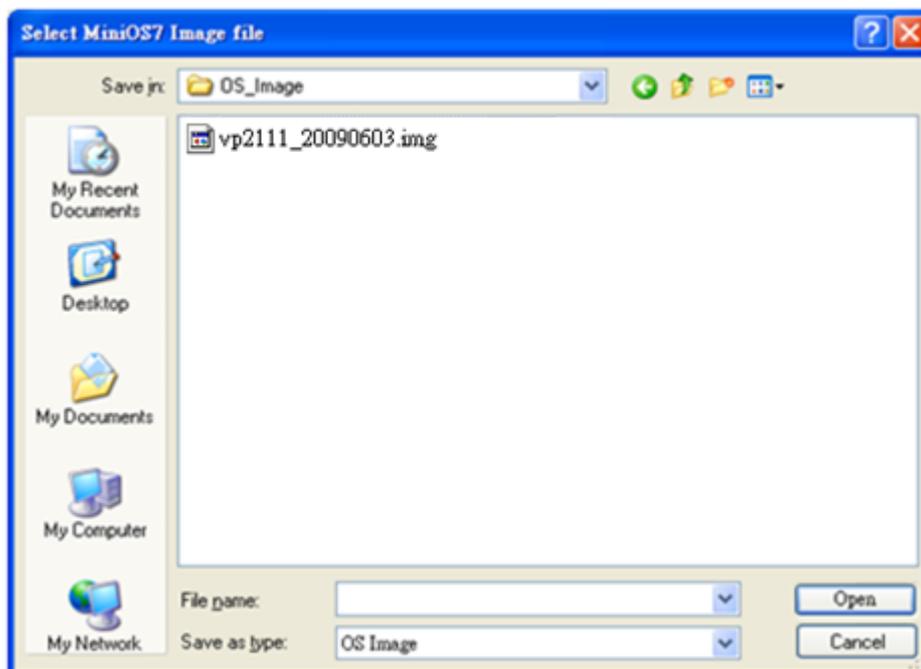
Step 2: Establish a connection

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

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



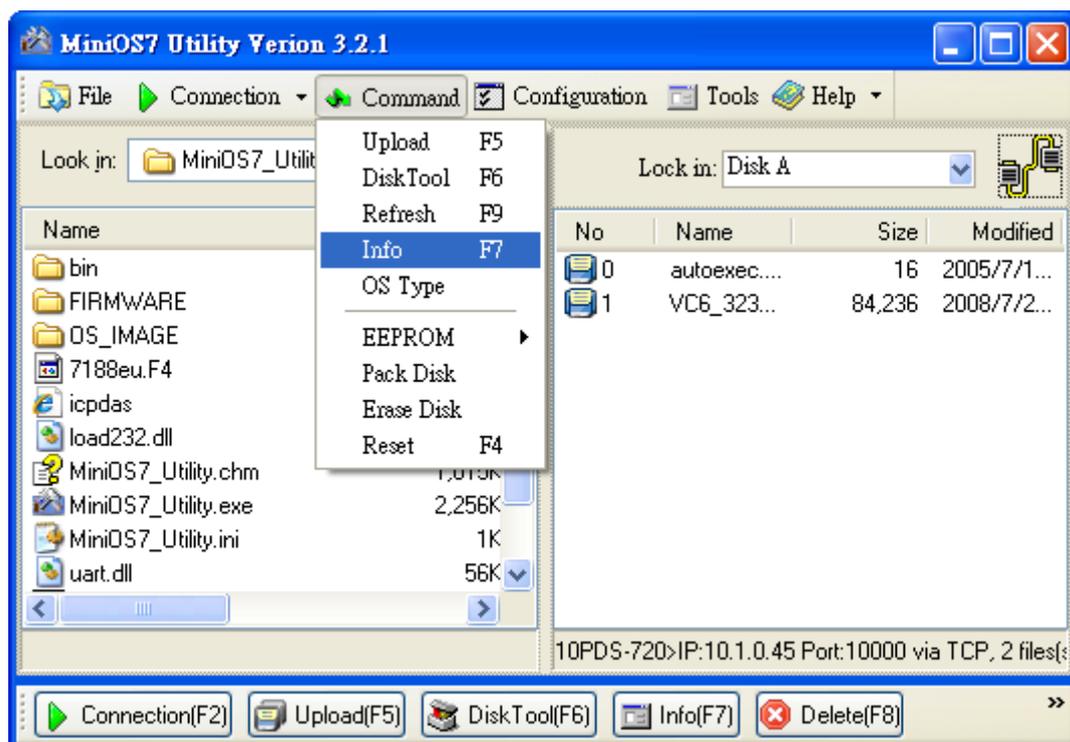
Step 4: Select the latest version of the MiniOS7 OS image



Step 5: Click the “OK”



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



3. “Hello World” – Your First Program

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

This chapter is step-by-step guide on how to write your first ViewPAC program - “Hello World”.

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

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

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

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

Tips & Warnings



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

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

Here we have used the Turbo C++ 1.01 to write your first program as an example.

3.1.1. Installing the Compiler

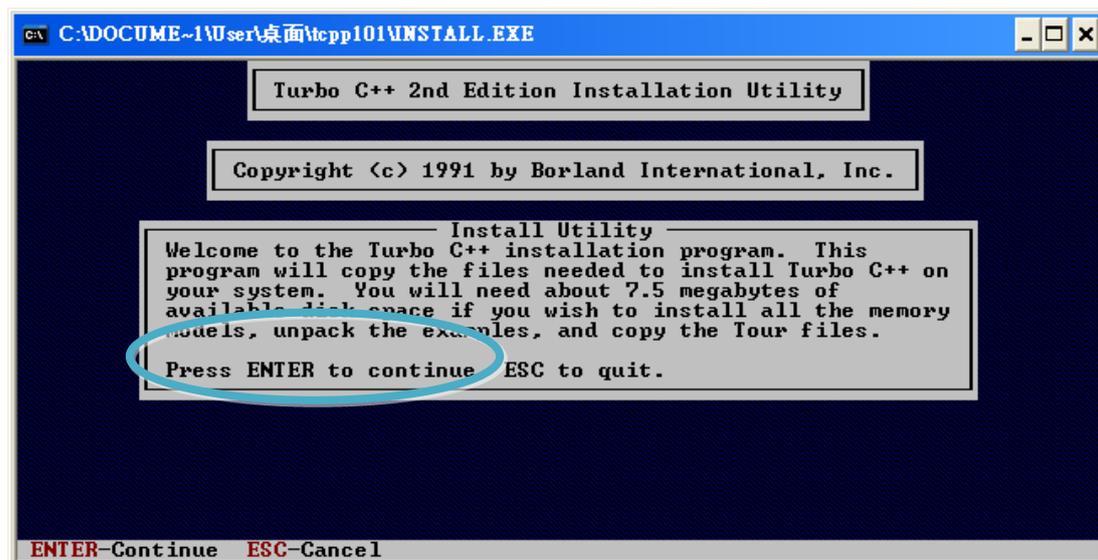
If there is no compiler currently installed on your system, installation of the compiler should be the first step.

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

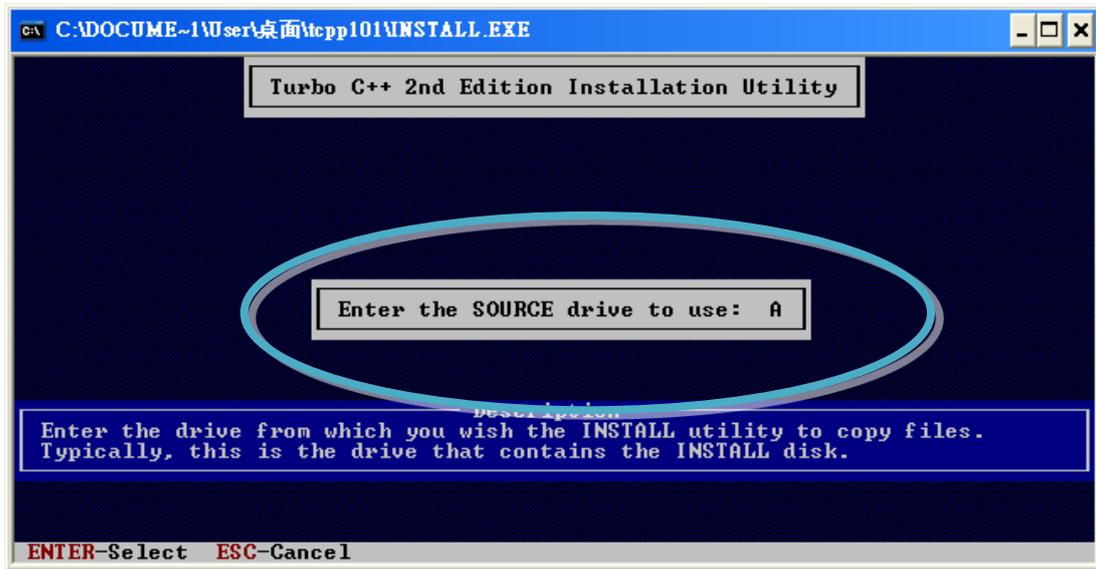
Step 1: Double click the Turbo C++ executable file to start setup wizard



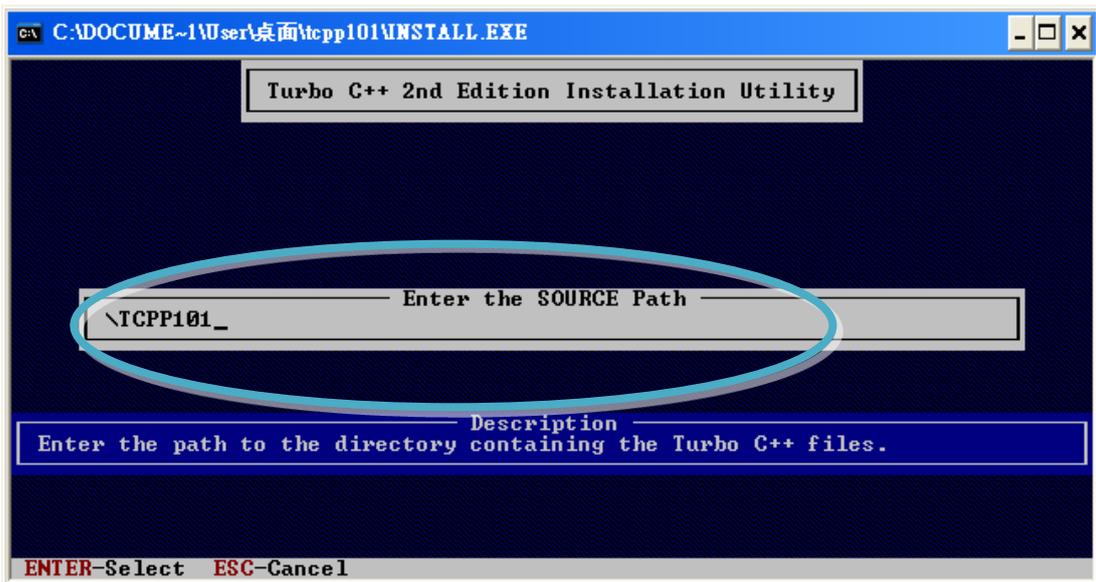
Step 2: Press "Enter" to continue



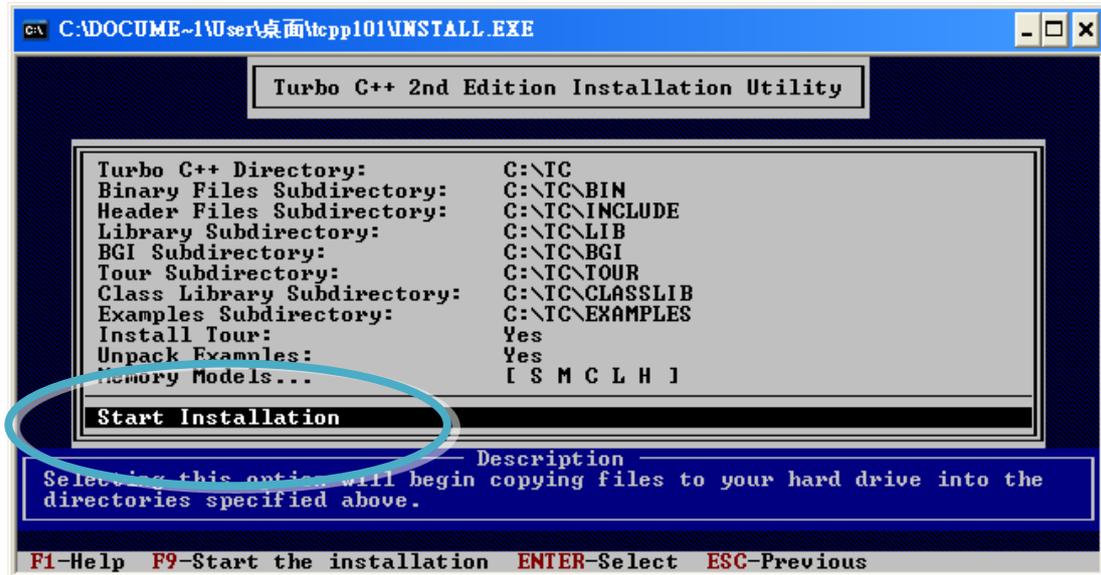
Step 3: Enter the letter of the hard drive you wish to install the software



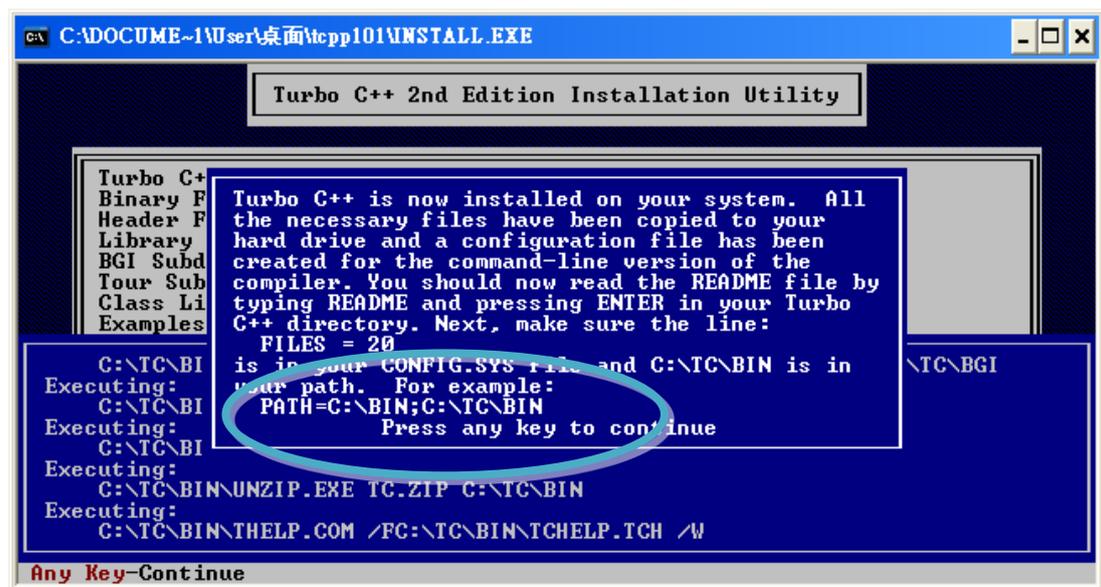
Step 4: Enter the path to the directory you wish to install files to



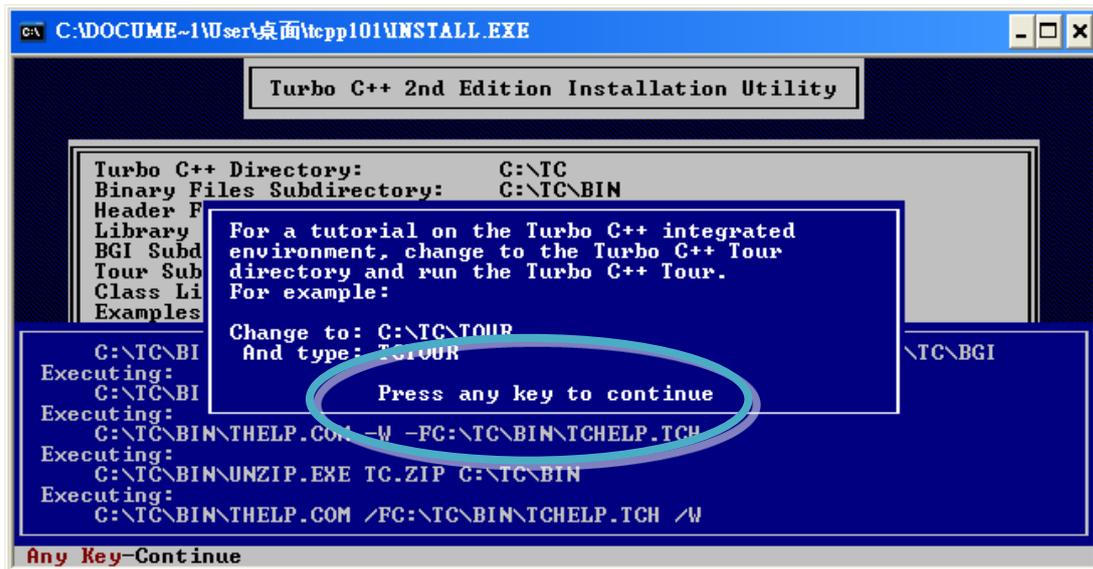
Step 5: Select "Start Installation" to begin the install process



Step 6: Press any key to continue



Step 7: Press any key to continue



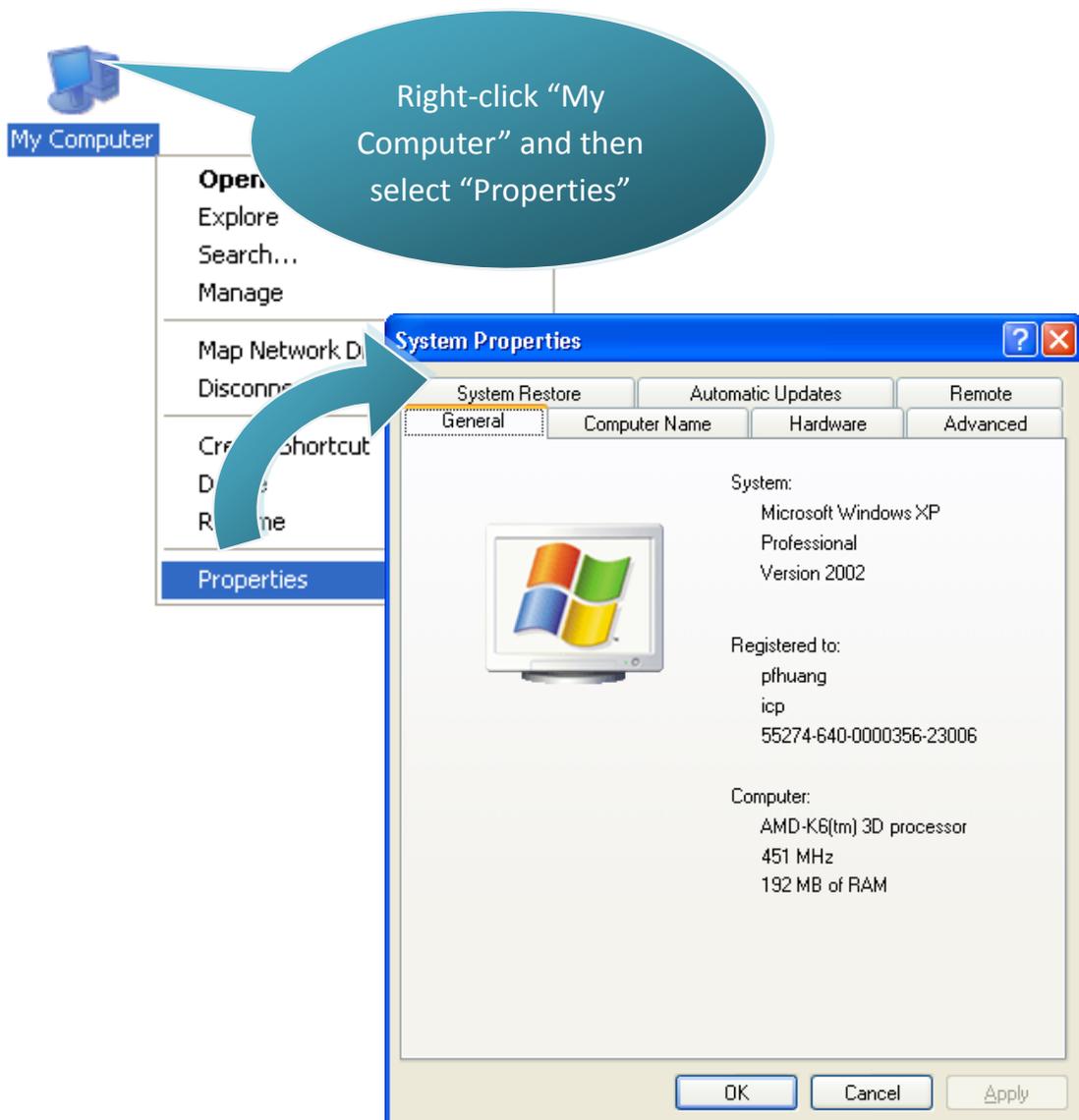
The screenshot shows a DOS-style window titled "C:\DOCUME~1\User\桌面\tcpp101\INSTALL.EXE". The main window is titled "Turbo C++ 2nd Edition Installation Utility". It contains a list of installation options on the left: Turbo C++ Directory, Binary Files Subdirectory, Header Files, Library, BGI Subdirectory, Tour Subdirectory, Class Library, and Examples. A central text box provides instructions for a tutorial: "For a tutorial on the Turbo C++ integrated environment, change to the Turbo C++ Tour directory and run the Turbo C++ Tour. For example: Change to: C:\TC\TOUR And type: TC\TOUR". Below this, a blue box with white text says "Press any key to continue", which is circled in red. The bottom of the window shows a list of commands being executed, such as "C:\TC\BIN\THELP.COM /W -FC:\TC\BIN\TCHELP.TCH" and "C:\TC\BIN\UNZIP.EXE TC.ZIP C:\TC\BIN". At the very bottom, it says "Any Key-Continue".

Step 8: Installation is complete

3.1.2. Setting up the Environment Variables

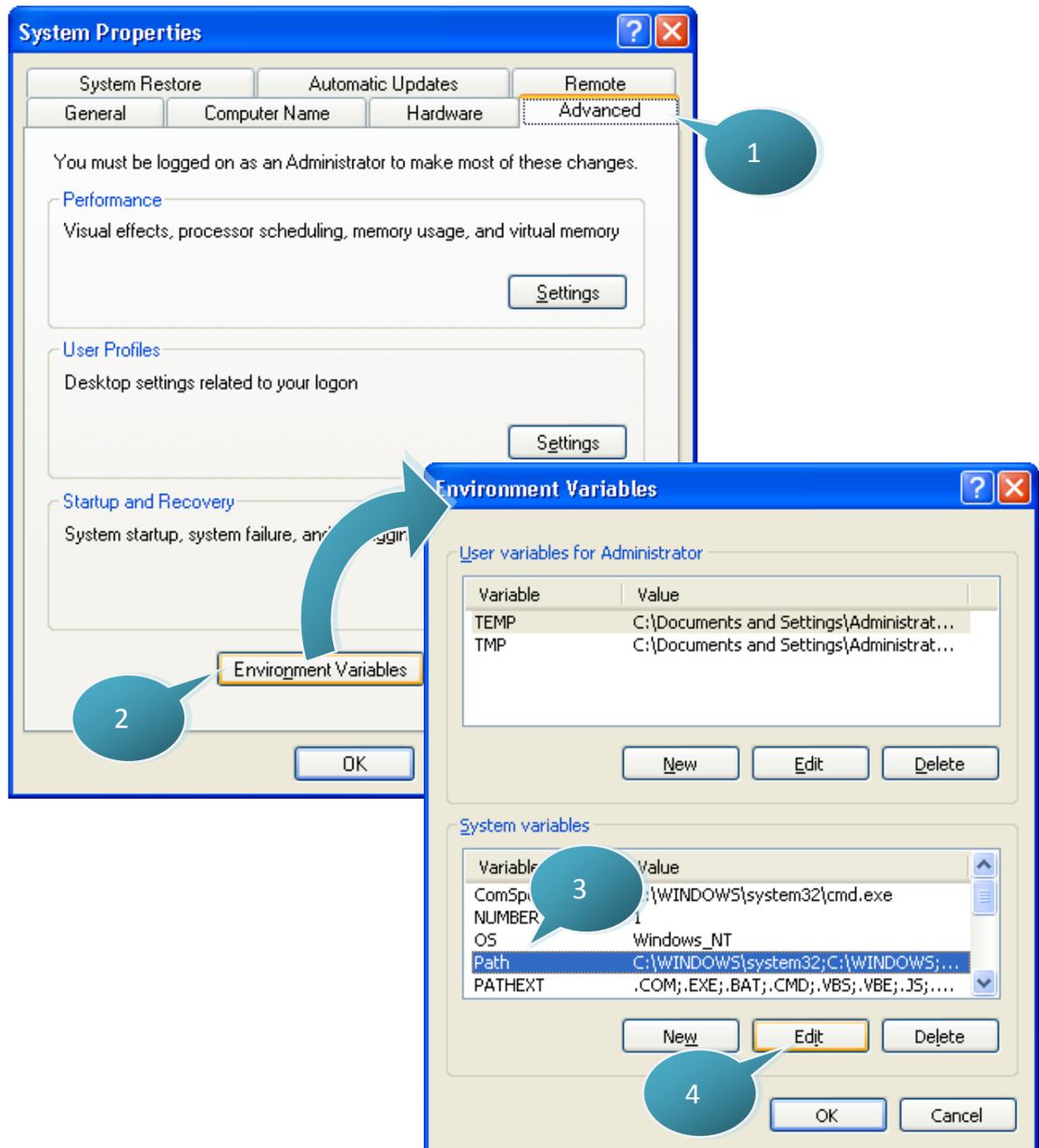
After installing the compiler, several compilers will be available from the Windows Command line. You can set the path environment variable so that you can execute this compiler on the command line by entering simple names, rather than by using their full path names.

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



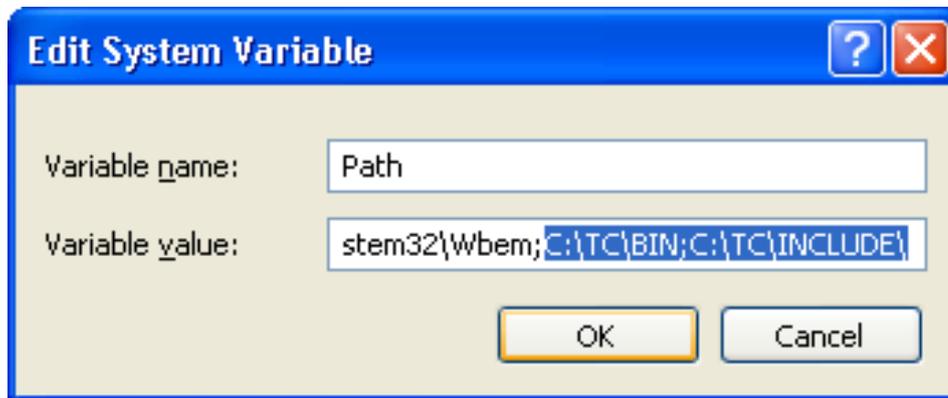
Step 2: On the “System Properties” dialog box, click the “Environment Variables” button located under the “Advanced” sheet

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



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

A semi-colon is used as the separator between variable values.
For example, ";c:\TC\BIN;c:\TC\INCLUDE\"



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

3.2. ViewPAC APIs

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

For more detailed information regarding ViewPAC APIs, please refer to

CD:\NAPDOS\vp-2000\Readme.txt

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/Readme.txt>

Before creating the application, ensure that you have installed the required APIs, demo programs and tools. If they are not installed, please refer to “section 2.2. Software Installation”.

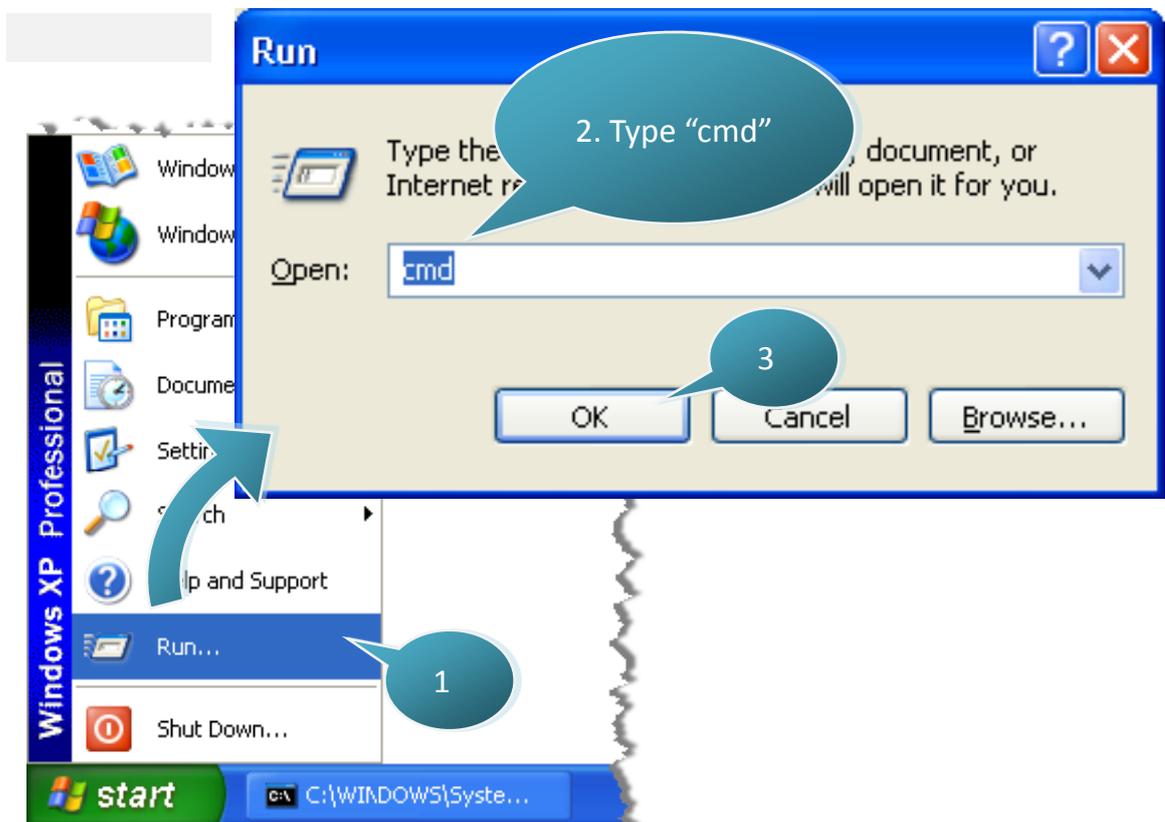
3.3. First Program in ViewPAC

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

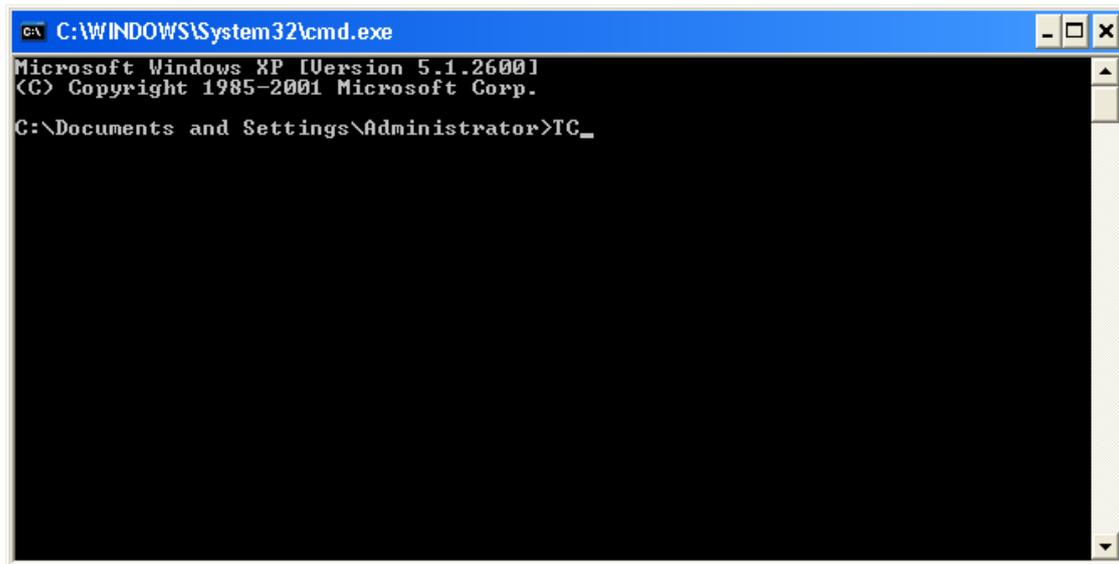
Below are step-by-step instructions for writing your first program.

Step 1: Open a MS-DOS command prompt

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

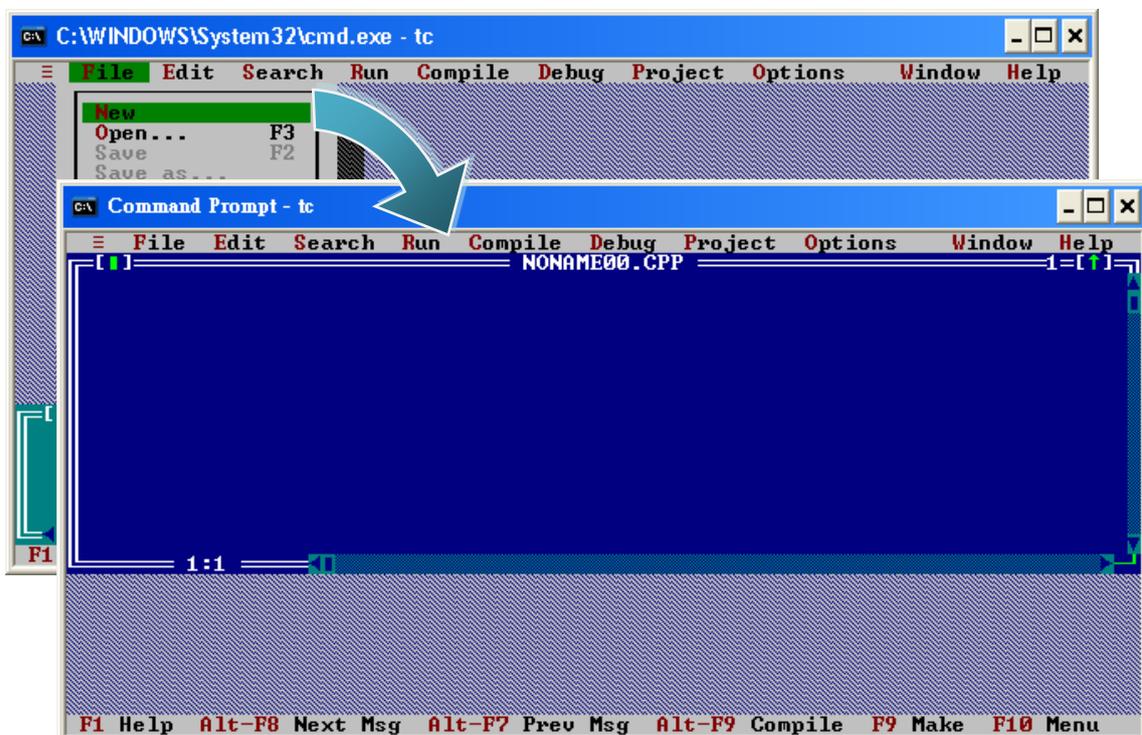


Step 2: At the command prompt, type "TC" and then press "Enter"



```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>TC_
```

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



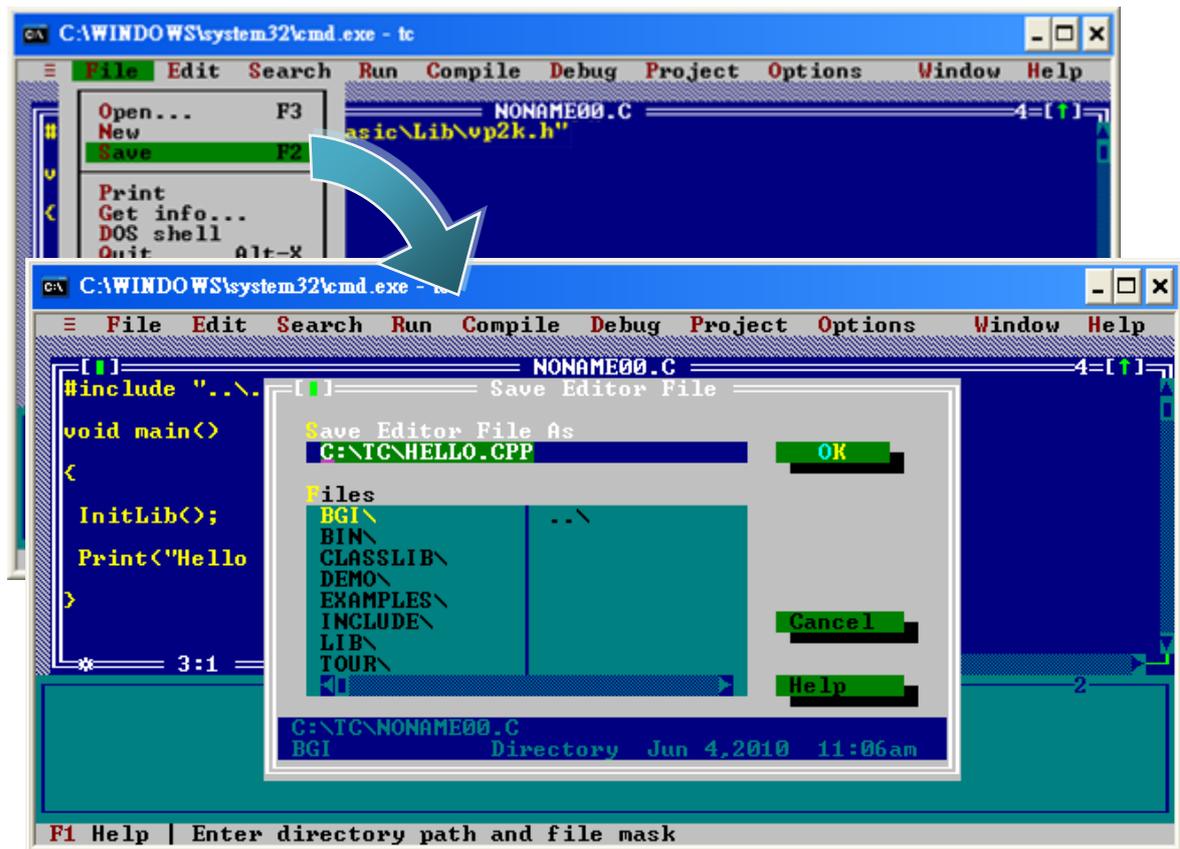
Step 4: Type the following code. Note that the code is case-sensitive

```
#include "..\\..\\Demo\\basic\\Lib\\vp2k.h"
/* Include the header file that allows vp2k.lib functions to be used */

void main(void)
{
    InitLib();    /* Initiate the ViewPAC library */
    Print("Hello world!\\r\\n");    /* Print the message on the screen */
}
```

Step 5: Save the source file

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



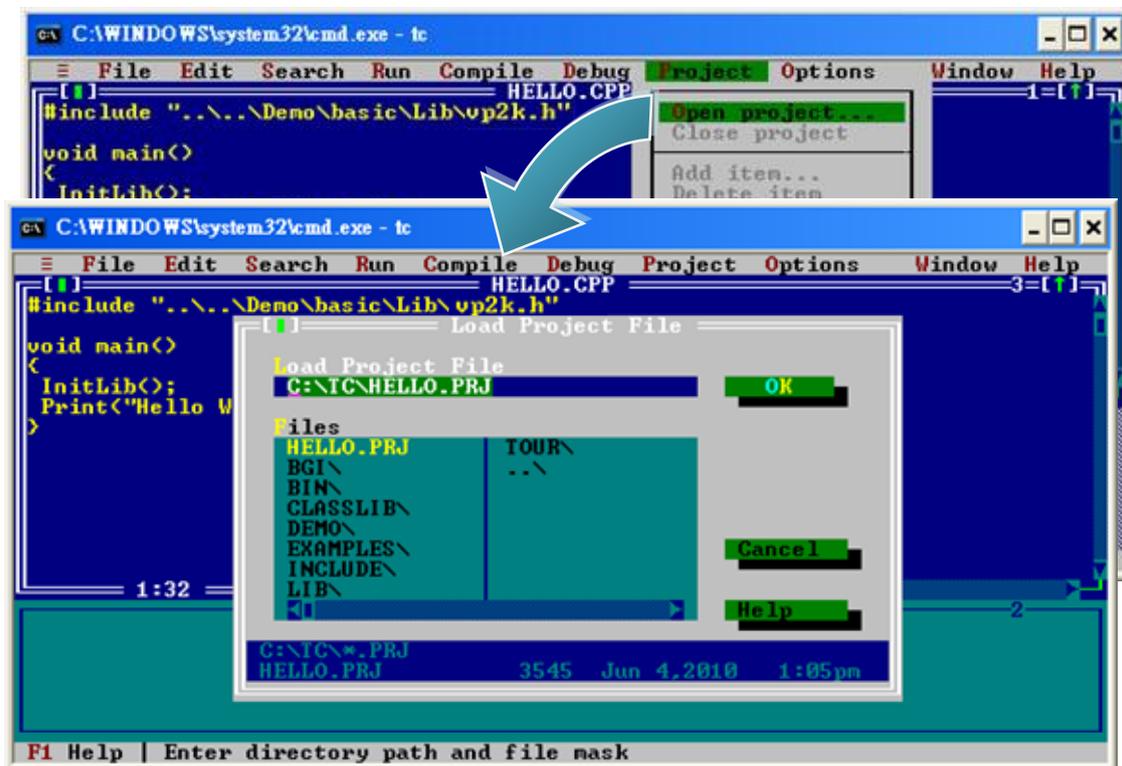
Tips & Warnings



You can write the code as shown below with your familiar text editor or other tools; please note that you must save the source code under a filename that terminates with the extension “C”.

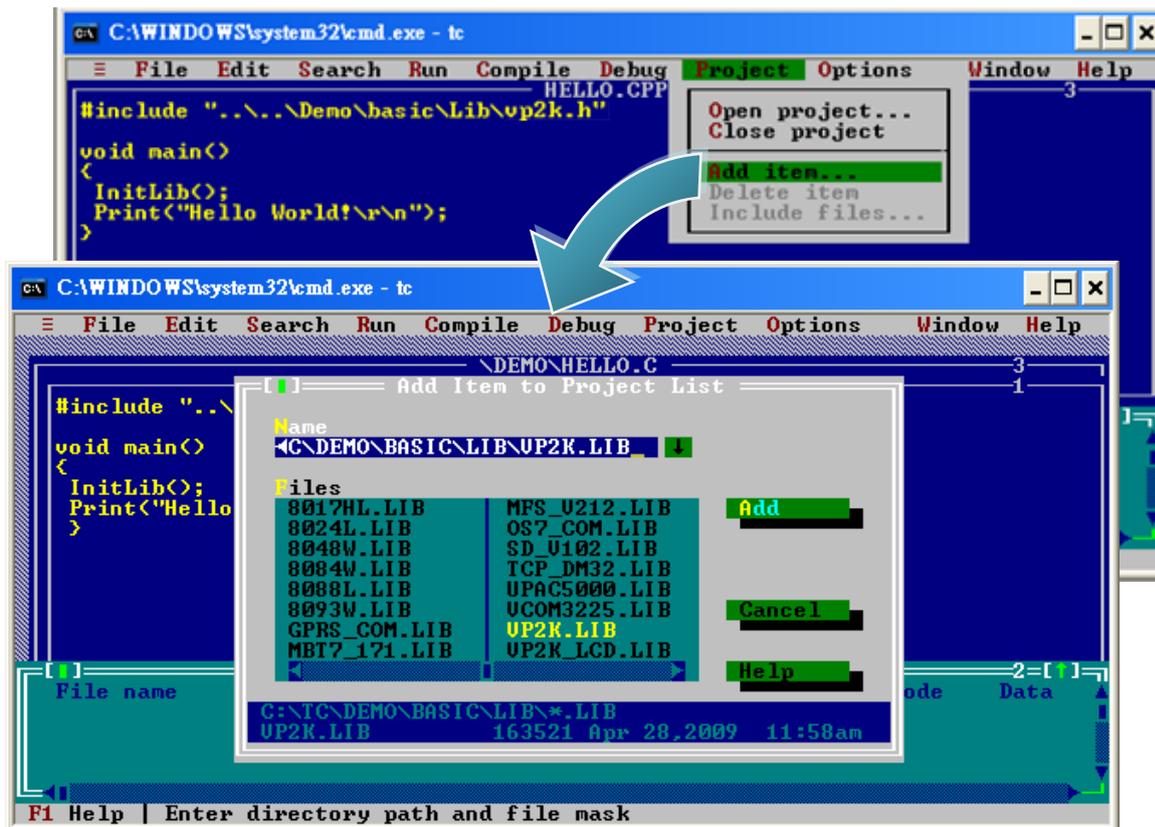
Step 6: Create a project (*.prj)

- i. Select “Open project...” from the “Project” menu
- ii. Type the project name “Hello”
- iii. Select “OK”
- iv. Select “Add”
- v. Select “Done” to exit



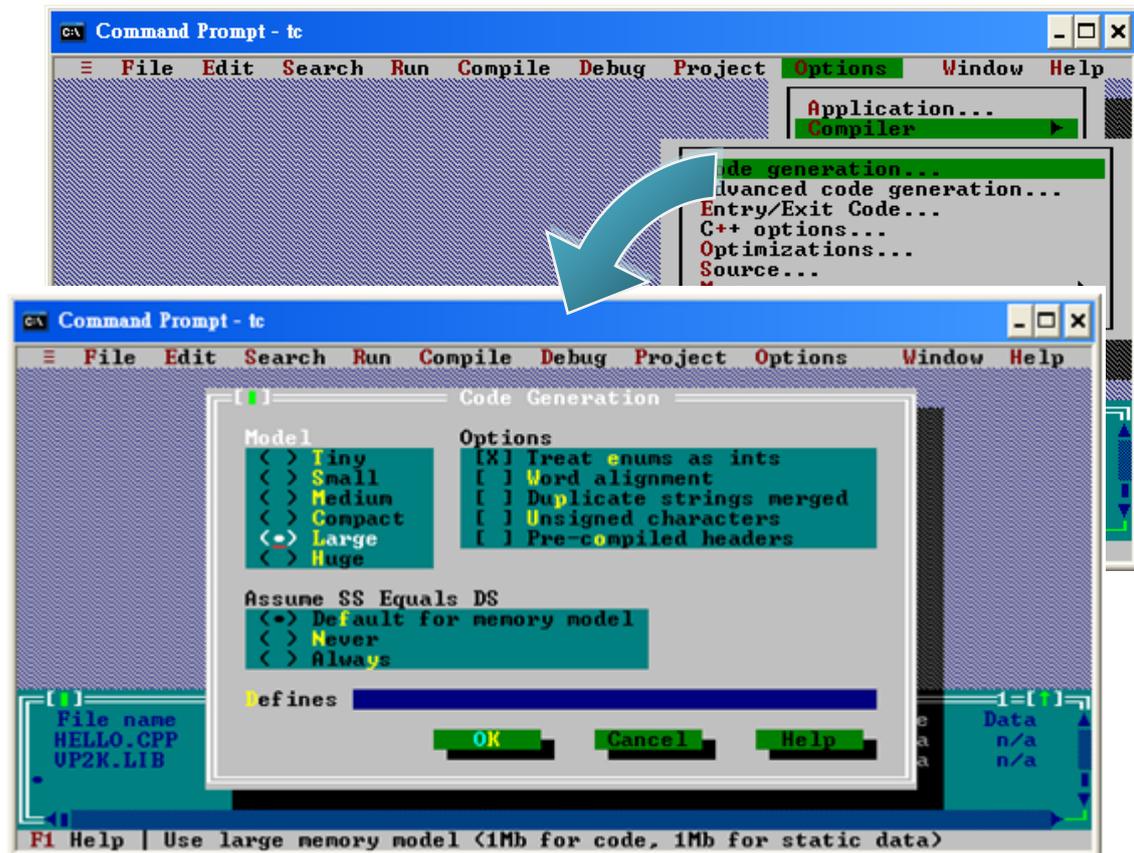
Step 8: Add the necessary function libraries to the project (*.lib)

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



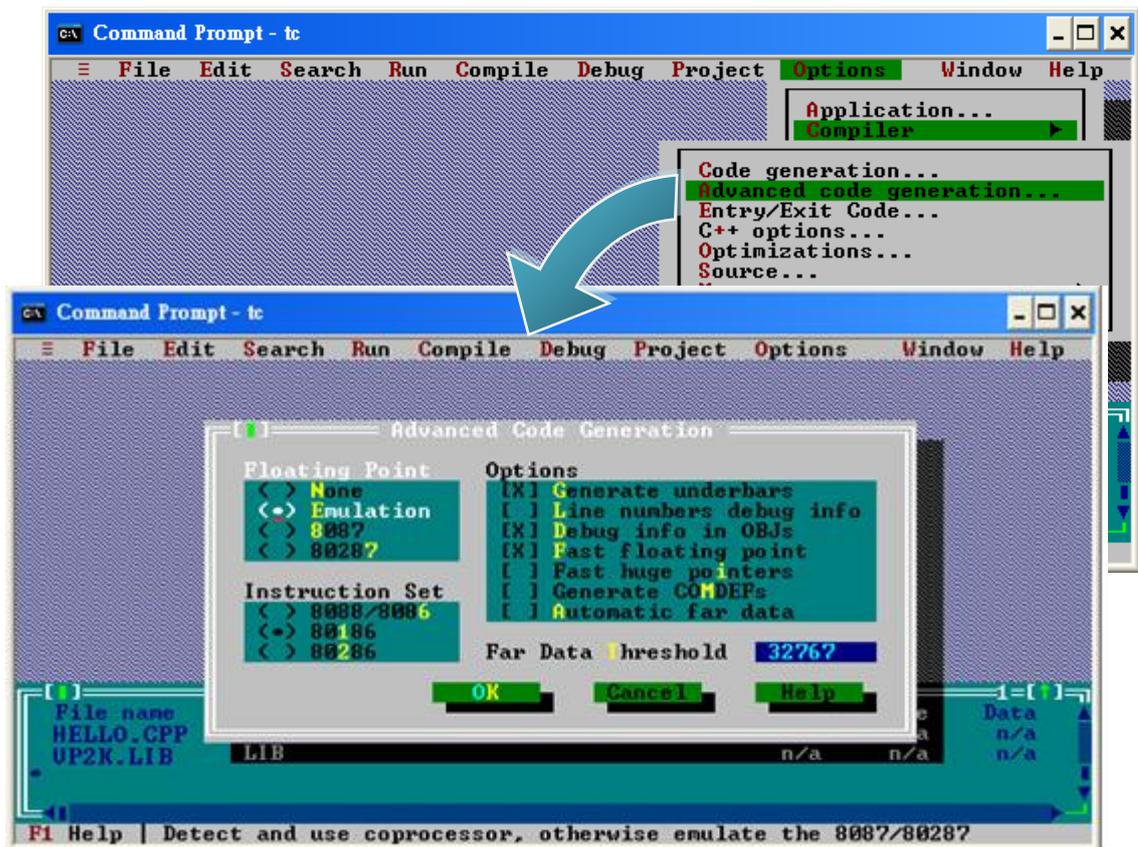
Step 9: Set the memory model to large

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



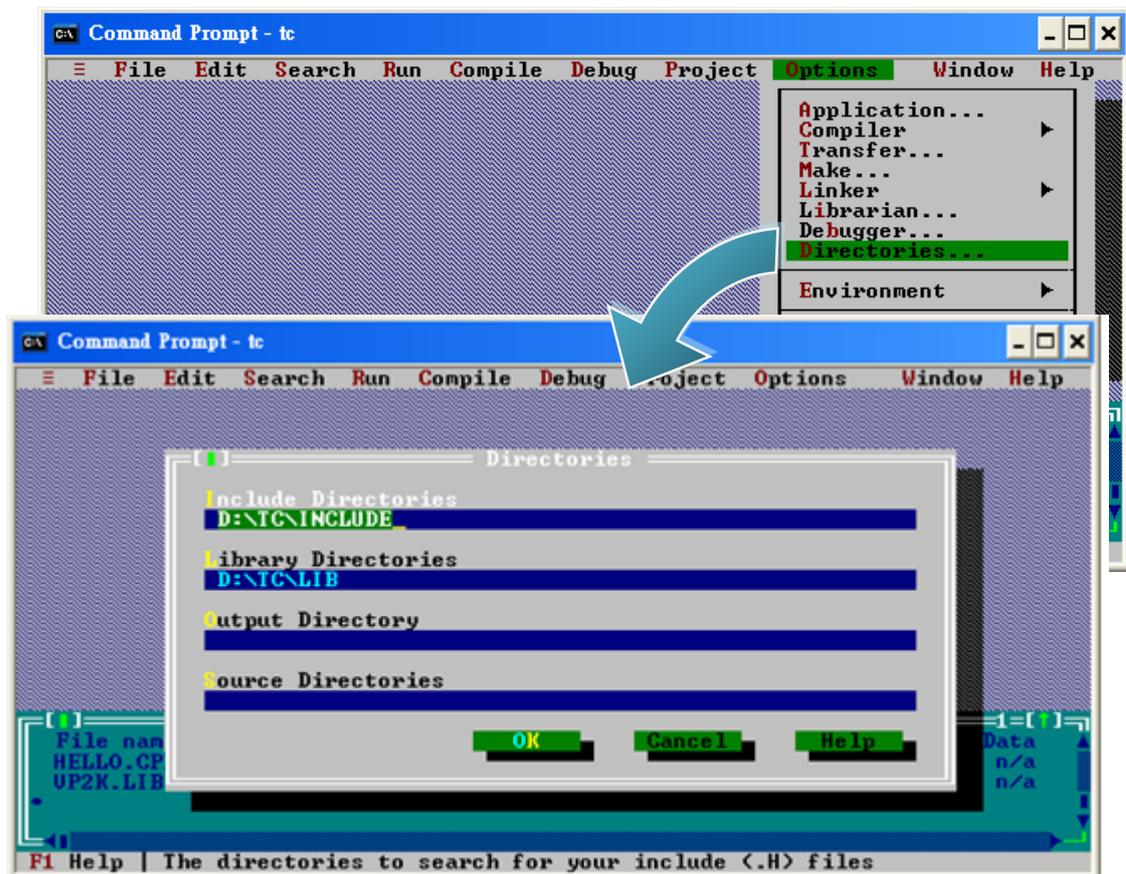
Step 10: Set the memory model to large

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

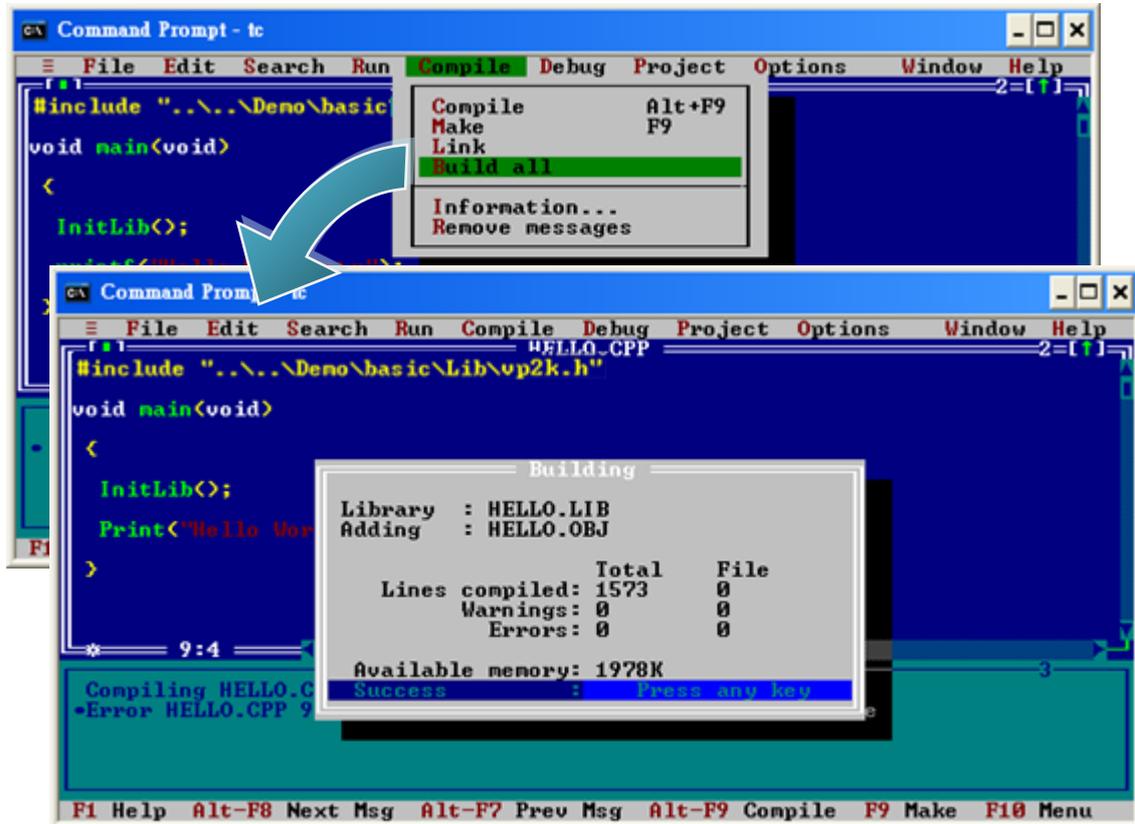


Step 11: Set the memory model to large

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



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

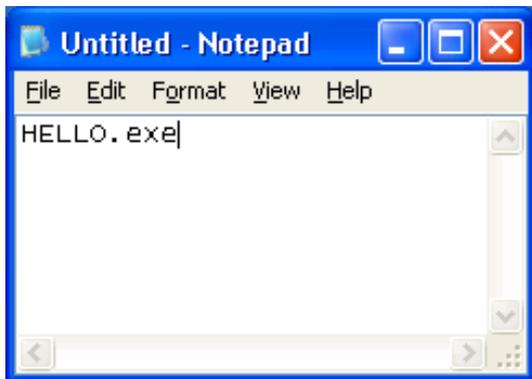


Step 13: Configure the operating mode

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



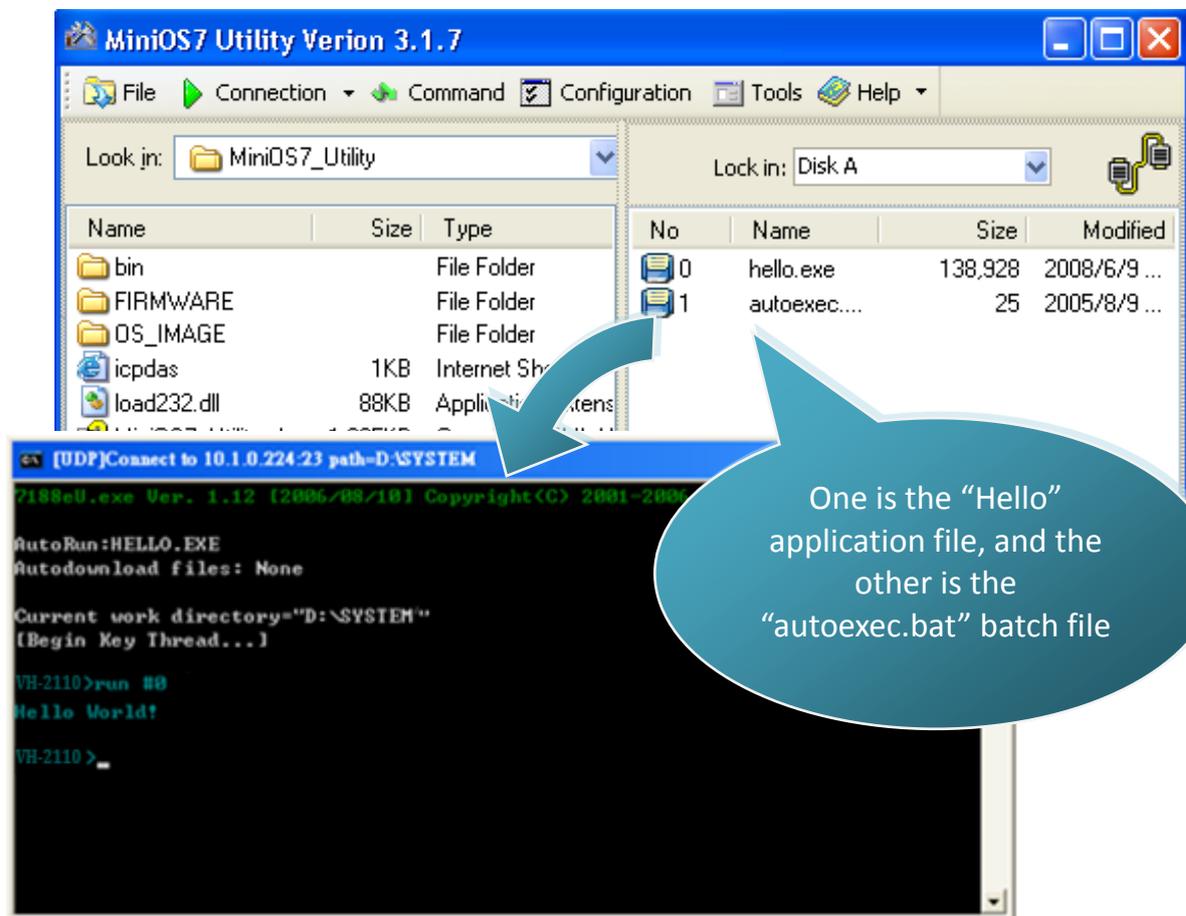
Step 14: Create an autoexec.bat file



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

Step 15: Upload programs to ViewPAC using MiniOS7 Utility

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

A screenshot of the MiniOS7 Utility Verion 3.1.7 interface. The "Look in:" field shows "MiniOS7_UTILITY" and the "Lock in:" field shows "Disk A". A file list table is visible with columns for Name, Size, Type, No, Name, Size, and Modified. The table contains two entries: "hello.exe" (138,928 bytes, modified 2008/6/9) and "autoexec..." (25 bytes, modified 2005/8/9). A blue callout bubble points to these files with the text: "One is the 'Hello' application file, and the other is the 'autoexec.bat' batch file". Below the file list, a terminal window shows the command prompt output: "AutoRun:HELLO.EXE", "Autodownload files: None", "Current work directory='D:\SYSTEM'", and "VH-2110>run #0", "Hello World!".

Name	Size	Type	No	Name	Size	Modified
bin		File Folder	0	hello.exe	138,928	2008/6/9 ...
FIRMWARE		File Folder	1	autoexec...	25	2005/8/9 ...
OS_IMAGE		File Folder				
icpdas	1KB	Internet Sh...				
load232.dll	88KB	Applicati...				

4. APIs and Demo References

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

The following table lists the APIs grouped by functional category.

API Description	Header File		Library	
	VP-2111	VH-2110	VP-2111	VH-2110
CPU	vp2k.h	VH2K.h	vp2k.lib	VH2K.lib
LCD	vp2k_lcd.H		vp2k_lcd.lib	
Ethernet	Tcpi32.h		tcp_dm32.lib	
64MB Flash Disk	MFS.h	None	MFS_V211.lib	None
Framework	MFW.H		FW_09314.lib	

For more detailed information regarding ViewPAC APIs, please refer to
CD:\NAPDOS\vp-2000\demo\Readme.txt

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/readme.txt>

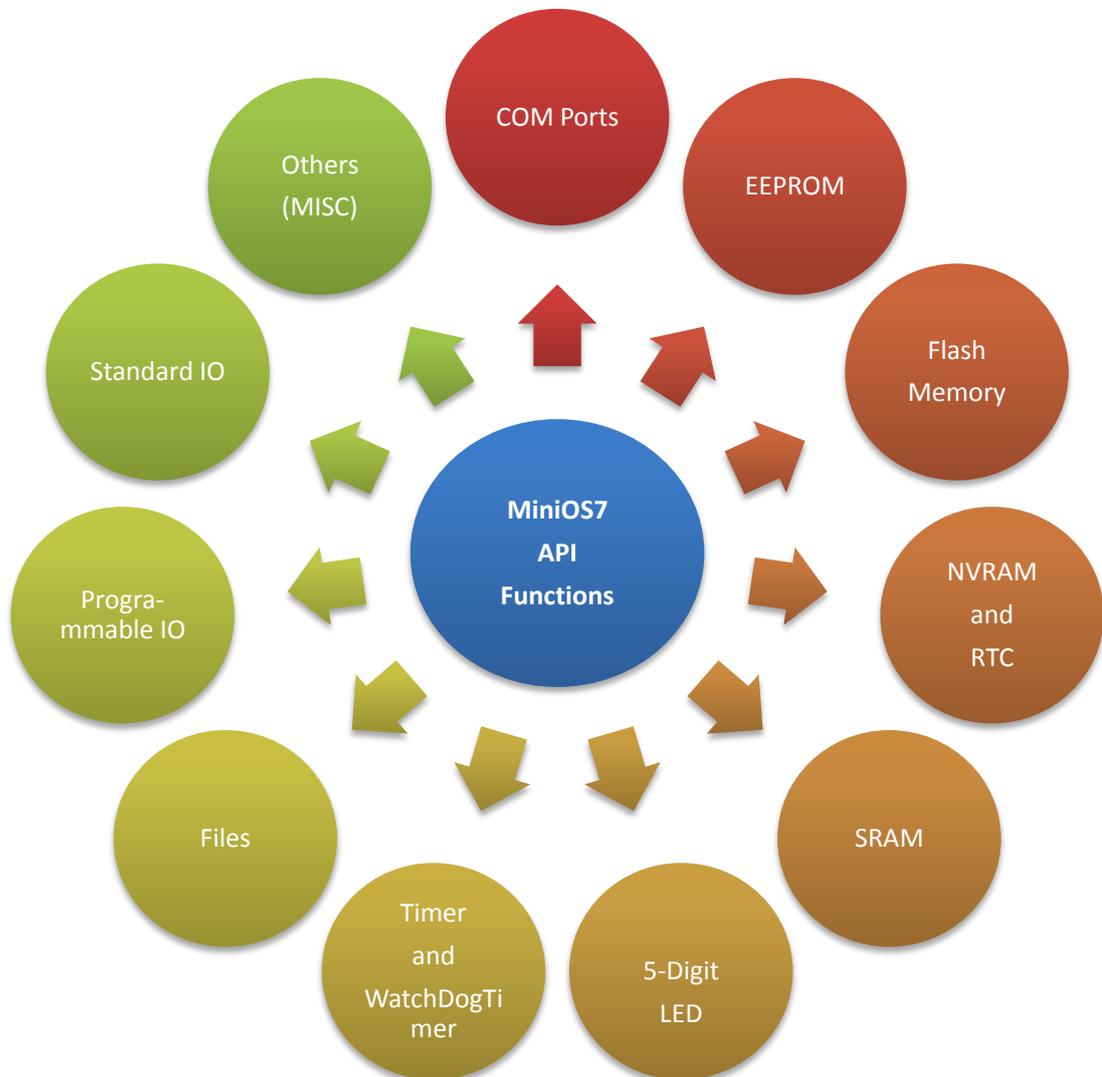
The following introduces the core API, MiniOS7 API, which is integrated into the ViewPAC API set.

Functions Library – VH-2110: vh2k.lib, VP-2111: vp2k.lib

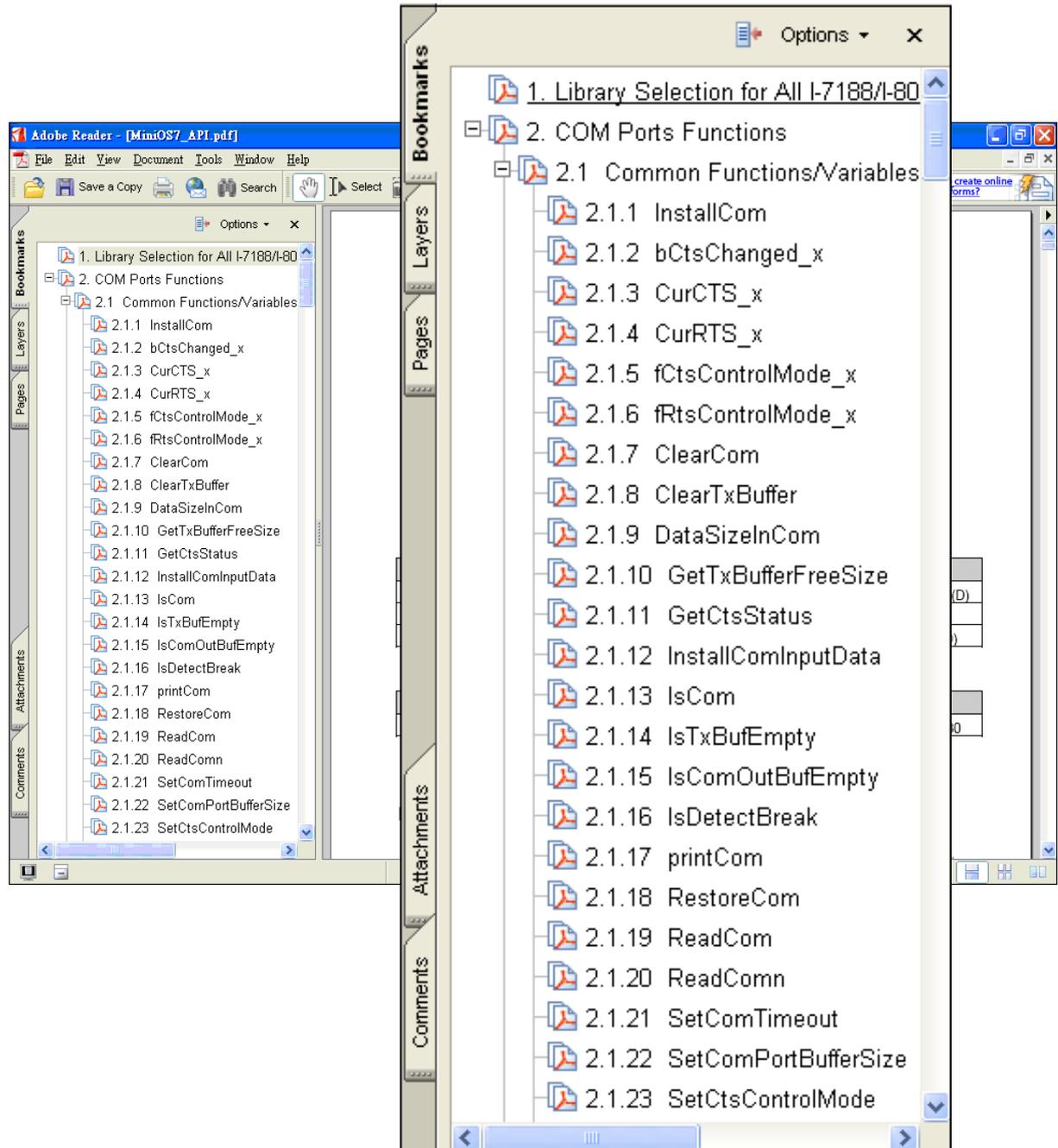
This file contains the MiniOS7 API (Application Programming Interface) and has hundreds of pre-defined functions.

Header File –VH-2110: vh2k.h, VP-2111: vp2k.h

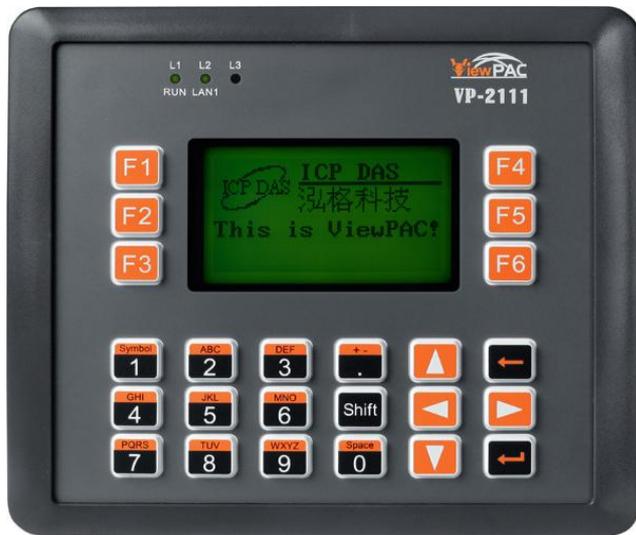
This file contains the forward declarations of subroutines, variables, and other identifiers used for the MiniOS7 API.



For full usage information regarding the description, prototype and the arguments of the functions, please refer to the “MiniOS7 API Functions User Manual” located at:
CD:\Napdos\MiniOS7\Document\
<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/minios7/document/>



► Demo programs for display (LCD) and sound control



Folder	Demo	Explanation
lcd_key	Showchar	Shows characters
	showbmp	Shows BMP pictures
	showTC	Shows Traditional Chinese font
	lcd_backlight	Adjusts LCD backlight
	cursor	Shows cursor
	keystatus	Shows status of key down/up
	presskey	Shows key value
	testkey	Shows key value
	ViewPAC	Over all function test
	showSC	Shows Simplified Chinese font
sound	change_key_sound	Changes sound of key pressing
	sound1	Changes sound of key pressing

➤ Demo programs for basic application

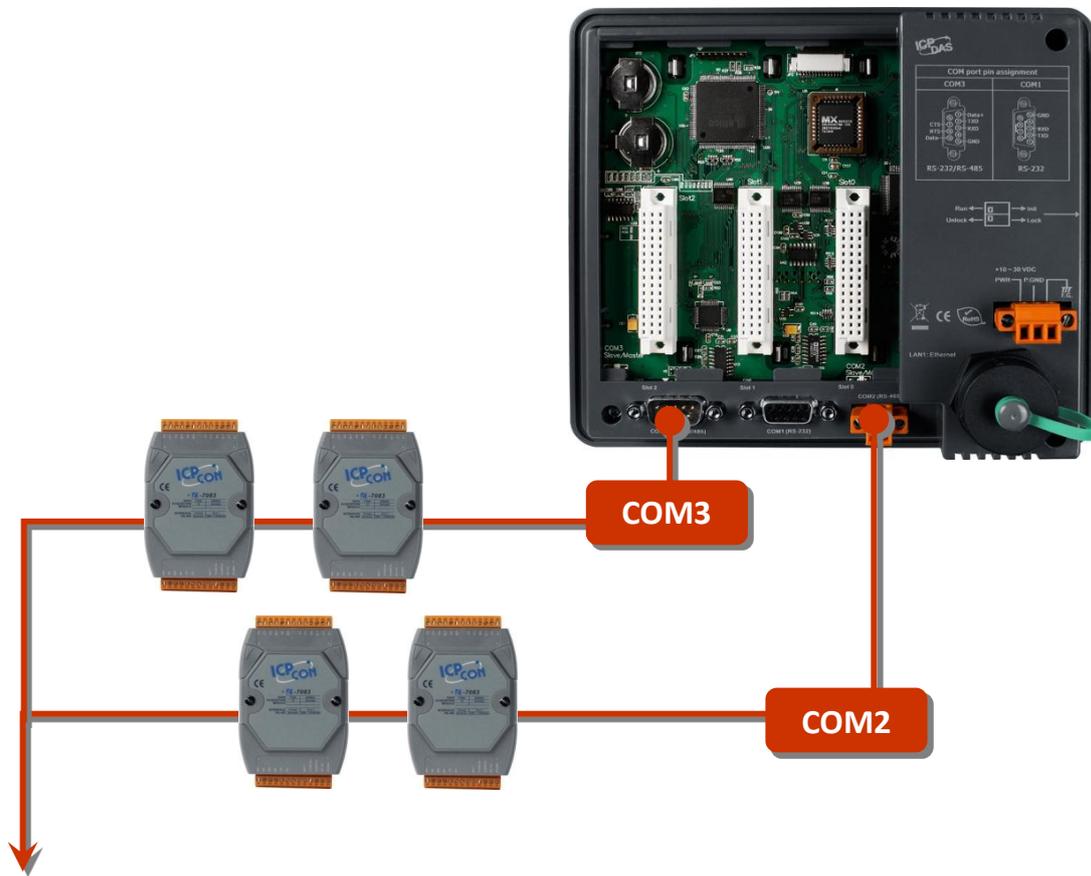
Folder	Demo	Explanation
File	Config_1_Basic	Reads information from text files(basic)
	Config_2_Advanced	Reads configure file (text file)(advanced)
Hello	Hello_C	Reads library version and flash memory size
	Hello_C++	Reads library version and flash memory size
Misc	Reset	Software's reset
	Runprog	To select item and run it
	SerialNumber	To get 64-bit hardware unique serial number
	Watchdog	Enabled WDT or bypass enable watchdog
Misc\Memory	512k_flash	Read/write integer, float and string data from/to flash
	512k_SRAM	Read/write data from/to SRAM
	EEPROM	Read/write integer and float data from/to flash
DateTime	DateTime	User can read and write the date & time of RTC
Timer	Timer	Please refer to the following location: ftp://ftp.icpdas.com.tw/pub/cd/8000cd/napdos/8000/common/minios7/demo/
Com_Ports	C_Style_IO	(1) Show how to write a function for input data (2) To get a string (3) To use C function: scanf or just use Scanf()
	Receive	Receive COM port Slv_COM.c is non-blocked mode Receive.c is blocked mode
	Slv_COM	Slave COM port demo for (request/reply) or (command/response) application
	ToCom_In_Out	How to Read/Write the byte data via COM port

► Demo programs for 64MB Flash Memory (for VP-2111/VH-2111P/VH-2211/VH-2311 module only)



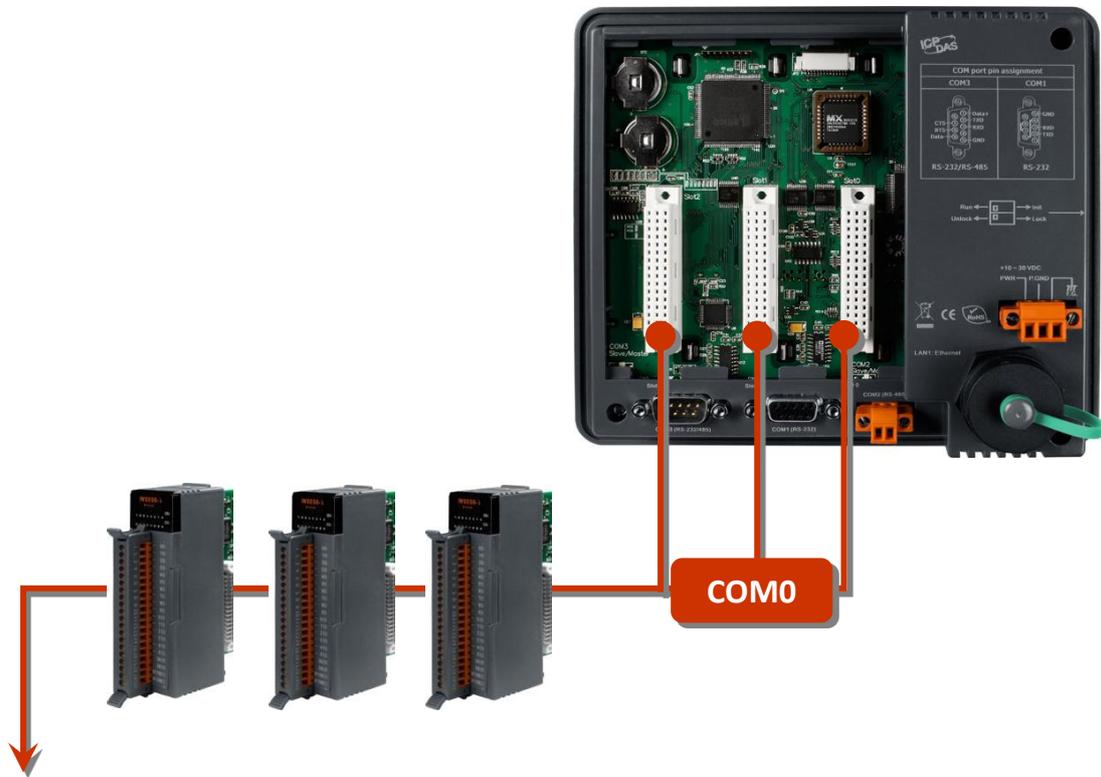
Folder	Demo	Explanation
64MB_Flash	Gets	How to get a string from a file in the 64MB flash memory.
	mFS_QA	Quality assurance program for the MiniOS7 File System. Including function test, read/write performance test.
	Puts	How to write a string to a file in the 64MB flash memory.
	Utility	Utility for the MiniOS7 File System. Operations Include Dir, Read, Write, etc.

► Demo programs for expanded I/O



Folder	Demo	Explanation
7K87K_for_COM	7K87K_demo_for_com	User can use "Com port" to connect and control I-7K or I-87K modules.
	7K87K_DI_for_Com	
	7K87K_DO_for_Com	Com port: 8410/8810/8411/8811 can use Com2,Com3: 8430/8830/8431/8831(CPU 40 and 80M) can use Com3, Com4
	7K87K_DIO_for_Com	
	7K87K_AI_for_Com	
	AO_22_26_for_Com	
	AO_024_for_Com	

► Demo programs for I/O slots (For VP-2111 module only)



Folder	Demo	Explanation
IO_in_Slot	8K_DI	This demo program is used in 8K's DI module. Such as I-8040W, I-8051W.
	8K_DO	This demo program is used in I-8K's DO module. Such as I-8041W, I-8056W.
	8K_DIO	This demo program is used in I-8K's DO module. Such as I-8042W, I-8054W.
	8050w	I-8050W is a 16-channel Universal Digital I/O Module. I/O Type: I/O select by programming. User can refer to the demo to write I-8050W's code.
	8017hw	I-8017HW is a 14-bit 100K sampling rate 8-channel analog input module. User can refer to the demo to write I-8017HW's code.
	8024w	I-8024W is a 4-channel Isolated Analog Output Module. User can refer to the demo to write I-8024W's code.
	87K_demo	This demo program is for I-87K General Function in

		Com0.
	87K_DI	This demo program is for I-87K Digital Input Module in Com0. Such as I-87040W, I-87051W
	87K_DO	This demo program is for I-87K Digital Output Module in Com0. Such as I-87041W, I-87057W
	87K_DIO	The demo program is for I-87K DIO Module in Com0. Such as I-87042W, I-87054W, I-87055W
	87K_AI	This demo program is for I-87K Analog Input Modules. Such as I-87013W, I-87015W, I-87017W, I-87018W
	8088w	I-8088W is a 8 PWM output channels and 8 Digital Input Module. User can refer to the demo to write I-8088W's code.
	87024w	This demo program is for I-87024W Analog Output Module
	Find_IO	This demo program is the basic function. Let you know how to get the card information in the MCU of I-8000 series
IO_in_Slot/8084w/BC_Demo	ABPhase	Demo for I-8084W (Pulse/Dir mode).
	Freq	Demo for I-8084W (Frequency mode).
	PulseDir	Demo for I-8084W (Pulse/Dir mode).
	Up	Demo for I-8084W (Up counter mode).
	Updown	Demo for I-8084W (Up/Down mode).

For more detailed information regarding the I/O expansion module APIs, please refer to

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\lib

<http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/lib/>

4.1. API for COM Port

The ViewPAC provides built-in COM ports as below shown.

► VP-2111



► VH-2110



► VH-2111P/VH-2211/VH-2311



4.1.1. Types of COM port functions

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

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

Tips & Warnings



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

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

Types of Functions	COM Port	Buffer	Functions				
MiniOS7 COM port	1, 2, etc.	1 KB	1 KB	IsCom()	ToCom()	ReadCom() ()	printCom ()
(C style) Standard COM port	1	512 Bytes	256 Bytes	Kbhit()	Puts() Putch()	Getch()	Print()

4.1.2. API for MiniOS7 COM port

API for using COM ports

1. InstallCom()

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

2. RestoreCom()

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

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

3. IsCom()

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

API for reading data from COM ports

4. ReadCom()

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

API for sending data to COM ports

5. ToCom()

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

For example, reading and receiving data through the COM1.

```
#include <stdio.h>
#include "vp2k.h"

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

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

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

API for showing data from COM ports

6. printCom()

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

For example, showing data from the COM1 port.

```
#include <stdio.h>
#include "vp2k.h"

void main(void)
{
    int i;

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

4.1.3. API for standard COM port

The standard COM port is used to upload program from PC to the ViewPAC.

Tips & Warnings



(C style) Standard COM port functions only can be used with the COM1 port, the following configurations of the COM1 port are fixed:

Baudrate = 115200 bps, Data format = 8 bits

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

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

1. Kbhit()

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

API for reading data from standard I/O port

2. Getch()

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

API for sending data to standard I/O port

3. Puts() – For sending a string

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

4. Putch() – For sending one character

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

API for showing data from standard I/O port

5. Print()

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

For example, reading and receiving data through COM1.

```
#include<stdio.h>
#include "vp2k.h"

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

    InitLib(); /* Initiate the ViewPAC 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 "vp2k.h"

void main(void)
{
    int i;

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

4.1.4. COM Port functions Comparison

For example, learning to show the ASCII code.

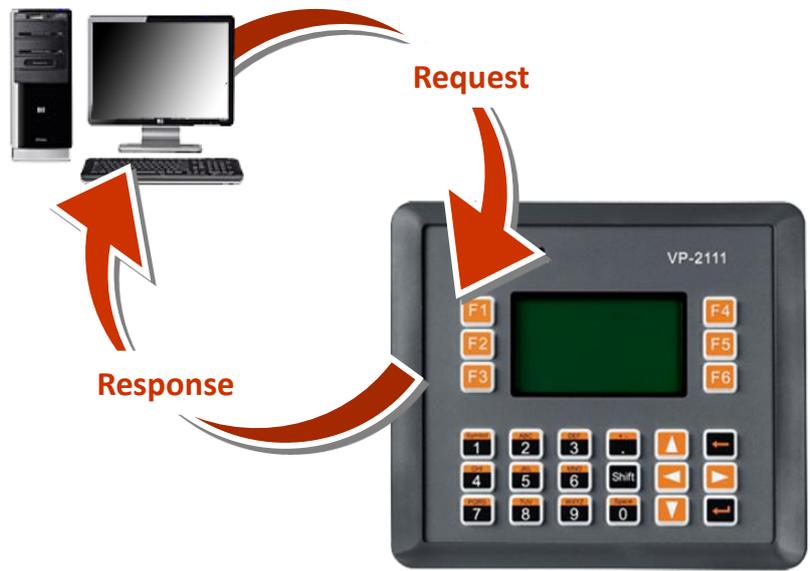
MiniOS7 COM port functions	Standard COM port functions
<pre>#include<stdio.h> #include "vp2k.h" void main(void) { unsigned char item; InitLib(); InstallCom(1, 115200, 8, 0, 1); printCom(1,"Hits any key.\n"); printCom(1,"Hit the ESC to exit!\n"); for(;;) { if(IsCom(1)) { item=ReadCom(1); if(item=='q') { return; } } else { printCom(1,"-----\n\r"); } } }</pre>	<pre>#include<stdio.h> #include "vp2k.h" void main(void) { unsigned char item; InitLib(); Print("Hits any key.\n"); Print("Hits the ESC to exit !\n"); for(;;) { if(kbhit()) { item=Getch(); if(item=='q') { return; } } else { Print("-----\n\r"); } } }</pre>

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

4.1.5. Request/Response protocol define on COM port

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

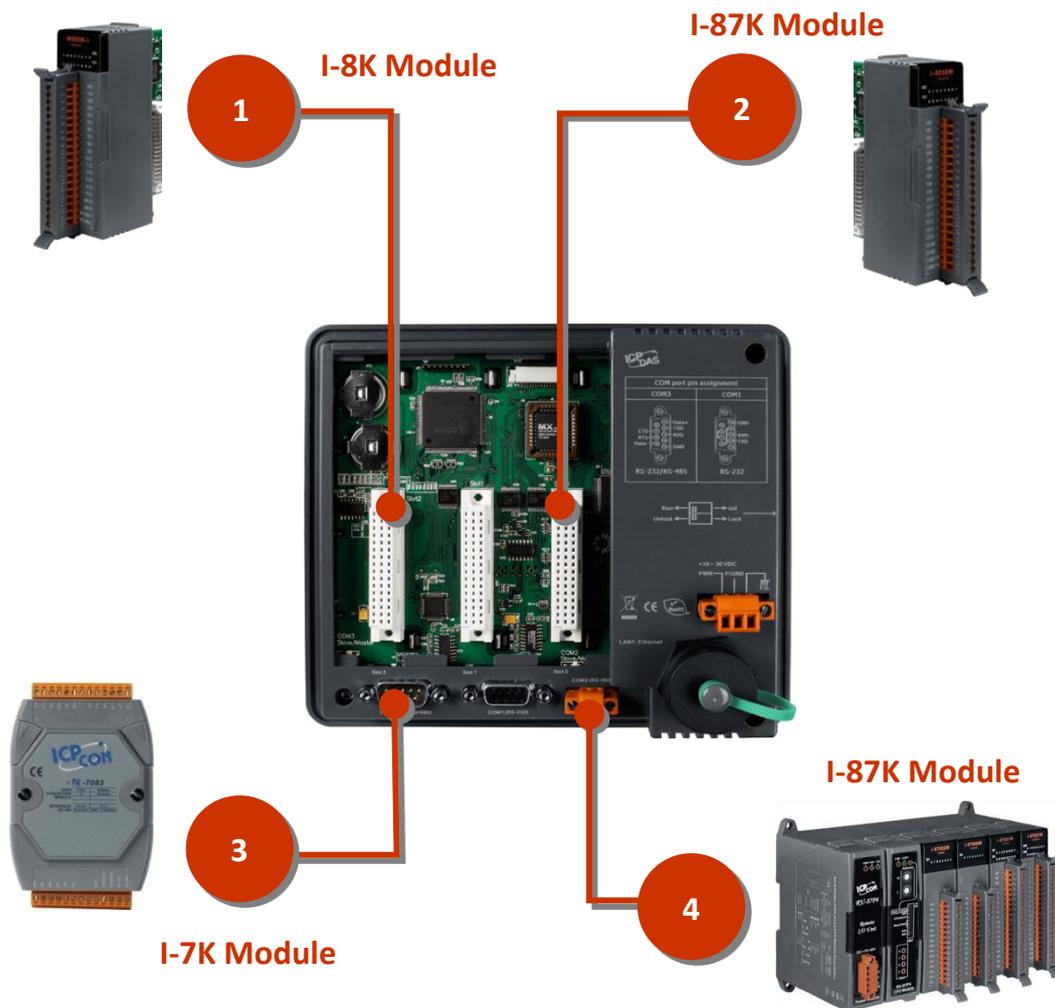
For a request/response application, please refer to “slave_com” demo



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

4.2. API for I/O Modules

- The ViewPAC is equipped with 3 I/O slots to access the I-8k and I-87k series I/O modules (High profile) as shown the point 1 and point 2 in the figure below. (for VP-2111 module only)
- The ViewPAC is equipped with multi-serial ports to access the I-7K series I/O modules for a wide range of RS-485 network application, as shown the point 3 in the figure below.
- The ViewPAC can connect to RU-87P2/4/8 to access the I-87k I/O series modules through RS-485, as shown the point 4 in the figure below.



The demo programs used for I-7K, I-8k and I-87k can be divided into the following:

For I-8k and I-87k I/O modules in slots, please refer to:

VP-2111:

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\IO_in_Slot\

http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/io_in_slot/

For I-7K and I-87k I/O modules is connected to the COM ports, please refer to:

VP-2111:

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\7K87K_for_COM\

http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/7k87k_for_com/

VH-2110:

CD:\NAPDOS\vp-2000\demo\vh-2110\Basic\7K87K_for_COM\

http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vh-2110/basic/7k87k_for_com/

4.2.1. Steps to use I-8K series I/O modules in slots (for VP-2111 module only)

API for reading DI modules



DI_8(), DI_16(), DI_32()

The DI_8(), DI_16(), DI32() must be called to read input value of DI modules.

For example, reading the input value of slot 3 DI modules:

```
#include <stdio.h>
#include "vp2k.h"

void main(void)
{
    Int DI_data, iSlot=3;
    InitLib(); /*Initiate the vp2k library*/
    For(;;)
    {
        DI_data=DI_8(iSlot); /*Read the input value of slot 3 DI module*/
        printf("DI Status==%x\n\r", DI_data);
    }
}
```

4.2.2. Steps to use I-87K series I/O modules in slots (for VP-2111 module only)

Follow the following steps to use I-87K series I/O modules in slots:

1. Use Installcom() to install the COM port driver.
2. Use ChangeToSlot() to assign the slot which the I-87k I/O module plugged in
3. Use SendCmdTo7000(0,...) to send commands
4. Use ReceiveResponseFrom7000_ms() to get the response.
5. Use RestoreCom() to restore the COM port driver



Tips & Warnings



The following settings of the COM0 are fixed:

Baud rate = 115200 bps, Data bit = 8 bits,

Parity check = None, Stop bit = 1

The following settings of I-87k series I/O modules that plugged in slots are fixed:

Address = 0, Check sum = Disable

Besides, the ChangeToSlot() function must be called.

For example, sending a command, '\$00M', to the I-87k I/O module that plugged on the slot 7 for getting the module name:

```
#include <stdio.h>
#include "vp2k.h"

void main(void)
{
    unsigned char InBuf0[60];
    InitLib(); /*Initiate the vp2k library*/
    InstallCom(0, 115200, 8, 0, 1); /*Install the COM0 driver*/
    InstallCom(1, 115200, 8, 0, 1); /*Install the COM1 driver*/

    ChangeToSlot(7);
    SendCmdTo7000(0, "$00M", 0); /*Send a command to COM0*/

    /*Timeout=50ms, check sum disabled*/

    ReceiveResponseFrom7000_ms(0, InBuf0, 50, 0);
    printCom(1, "Module Name=%s", InBuf0);
    Delay(10); /*Wait for all data are transmitted to COM port*/
    RestoreCom(0); /*Uninstall the COM0 driver*/
    RestoreCom(1); /*Uninstall the COM1 driver*/
}
```

4.2.3. Steps to use I-7K and I-87K series I/O modules that are connected with COM ports

Follow the following steps to use I-7K and I-87K series I/O modules that are connected with COM port:

1. Use Installcom() to install the COM port driver.
2. Use SendCmdTo7000(0,...) to send commands
3. Use ReceiveResponseFrom7000_ms() to get the response.
4. Use RestoreCom() to restore the COM port driver



For example, sending a command, '\$00M', to the I-7K and I-87k I/O module that are connected with COM2 for getting the module name:

```
#include <stdio.h>
#include "vp2k.h"

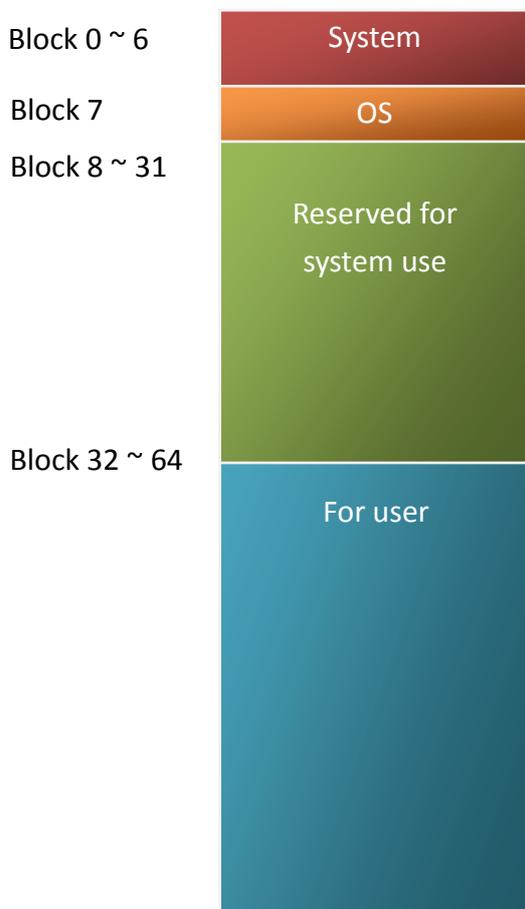
void main(void)
{
    unsigned char InBuf0[60];
    InitLib(); /*Initiate the vp2k library*/
    InstallCom(1, 115200, 8, 0, 1); /*Install the COM1 driver*/
    InstallCom(2, 115200, 8, 0, 1); /*Install the COM2 driver*/

    SendCmdTo7000(2, "$00M", 0); /*Send a command to COM2*/

    /*Timeout=50ms, check sum disabled*/
    ReceiveResponseFrom7000_ms(2, InBuf0, 50, 0);
    printCom(1, "Module Name=%s", InBuf0);
    Delay(10); /*Wait for all data are transmitted to COM port*/
    RestoreCom(1); /*Uninstall the COM1 driver*/
    RestoreCom(2); /*Uninstall the COM2 driver*/
}
```

4.3. API for EEPROM

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



API for writing data to the EEPROM

1. EE_WriteEnable()

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

2. EE_WriteProtect()

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

3. EE_MultiWrite()

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

API for reading data from the EEPROM

4. EE_MultiRead()

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

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

```
#include <stdio.h>
#include "vp2k.h"

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

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

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

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

VP-2111:

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\Misc\

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/misc/>

VH-2110:

CD:\NAPDOS\vp-2000\demo\vh-2110\Basic\Misc\

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vh-2110/basic/misc/>

4.4. API for Flash Memory

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



API for writing data to the Flash Memory

1. FlashWrite()

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

API for reading data from the Flash Memory

2. FlashRead()

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

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

```
#include <stdio.h>
#include "vp2k.h"

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

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

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

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

4.5. API for NVRAM

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

API for writing data to the NVRAM

1. WriteNVRAM()

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

API for reading data from the NVRAM

2. ReadNVRAM()

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

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

```
#include <stdio.h>
#include "vp2k.h"

void main(void)
{
    int data=0x55, data2;
    InitLib();    /* Initiate the ViewPAC library */
    WriteNVRAM(0,data);
    data2=ReadNVRAM(0);    /* Now data2=data=0x55 */
}
```

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

```
#include <stdio.h>
#include "vp2k.h"

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

    InitLib();    /* Initiate the ViewPAC library */
    WriteNVRAM(0, *dataptr);    /* Write the low byte */
    WriteNVRAM(1, *dataptr+1);    /* Write the high byte */
    dataptr=(char *) &data2;
    *dataptr=ReadNVRAM(0);    /* Read the low byte */
    (*dataptr+1)=ReadNVRAM(1);    /* Read the high byte */
}
```

4.6. API for Timer

- The ViewPAC can support a single main time tick, 8 stop watch timers and 8 counts down timers.
- The ViewPAC uses a single 16-bit timer to perform these timer functions, with a timer accuracy of 1 ms..

API for starting the Timer

1. TimerOpen()

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

API for reading the Timer

2. TimerResetValue()

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

3. TimerReadValue()

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

API for stopping the Timer

4. TimerClose()

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

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

```
#include <stdio.h>
#include "vp2k.h"

void main(void)
{
    Unsigned long time iTime;

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

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

VP-2111:

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\Timer\

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/timer/>

VH-2110:

CD:\NAPDOS\vp-2000\demo\vh-2110\Basic\Timer\

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vh-2110/basic/timer/>

4.7. API for WatchDog Timer (WDT)

- The ViewPAC 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 "vp2k.h"

void main(void)
{
    Unsigned long time iTime;

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

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

VP-2111:

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\Timer\

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/timer/>

VH-2110:

CD:\NAPDOS\vp-2000\demo\vh-2110\Basic\Timer\

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vh-2110/basic/timer/>

4.8. API for MFS (For

VP-2111/VH-2111P/VH-2211/VH-2311 modules only)



Required library and header files:

MFS_V211.lib and MFS.h

The VP-2111/VH-2111P/VH-2211/VH-2311 equip an extra 64MB flash memory, the MFS is designed to read/write file from/to the 64MB flash memory.

For full usage information regarding the hardware supported, applications, and the specification, please refer to section “Appendix C. What is MiniOS7 File System (MFS)”

- Summarize of the MFS functions:

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

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

API for starting 64MB flash memory

1. mfs_Init()

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

2. mfs_Stop()

If the program calls the mfs_Init() to initialize the 64MB flash memory, the mfs_Stop() must be called to allocate buffers to free upon closing.

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

3. mfs_OpenFile()

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

4. mfs_CloseFile()

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

API for writing data to the 64MB flash memory

5. mfs_Puts()

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

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

```
#include <stdio.h>
#include "vp2k.h"
#include "MFS.h"

#define _DISK_A 0
#define _DISK_B 1

int main(void)
{
    int iFileHandle, iRet;

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

    iFileHandle=mfs_OpenFile(_DISK_A,"Test.txt","w");
    if(iFileHandle>0)
    {
        Print("Write string to Test.txt...");
        mfs_Puts(iFileHandle,"test mfs on 64MB flash");
        mfs_CloseFile(iFileHandle);
        Print("done");
    }
    else
        Print("Open file error\n\r");
    mfs_Stop();
    return;
}
```

API for reading data from the 64MB flash memory

6. mfs_Gets()

After using the mfs_OpenFile() to open the file, the mfs_Gets() must be called to read data from the 64MB flash memory.

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

```
#include <stdio.h>
#include "vp2k.h"
#include "MFS.h"

#define _DISK_A 0
#define _DISK_B 1

int main(void)
{
    int iFileHandle, iRet;

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

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

        mfs_CloseFile(iFileHandle);
    }
}
```

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

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

CD:\NAPDOS\vp-2000\demo\vp-2111\Basic\64MB_Flash\

http://ftp.icpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/64mb_flash/

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.

Tips & Warnings

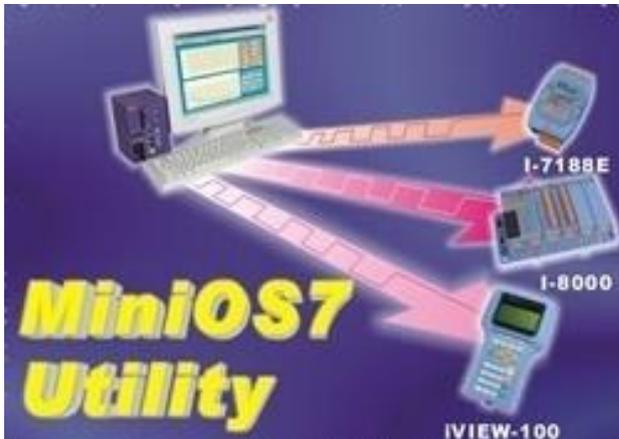


DOS (whether PC-DOS, MS-DOS or ROMDOS) is a set of commands or code that tells the computer how to process information. DOS runs programs, manages files, controls information processing, directs input and output, and performs many other related functions.

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

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

Appendix B. What is MiniOS7 Utility



MiniOS7 Utility is a tool for configuring, uploading files to all products embedded with ICPDAS MiniOS7 with easiness and quickness.

Note : Since version 3.1.1, the Utility can allow users remotely access the controllers (7188E,8000E,...ect) through the Ethernet.

Functions

Supported connection ways

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

Maintenance

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

Configuration

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

Check product information

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

Including Frequently Used Tools

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

PC System Requirements

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

Supported Products

1. 7188XA/XB/XC
2. 7188EX series
3. All i-8000 series
4. iView100
5. uPAC-7186EX
6. ET-6000 series
7. ET-7000 series

Download location :

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

Appendix C. What is MiniOS7 File System (MFS)

(For VP-2111/VH-2111P/VH-2211/VH-2311 modules only)

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



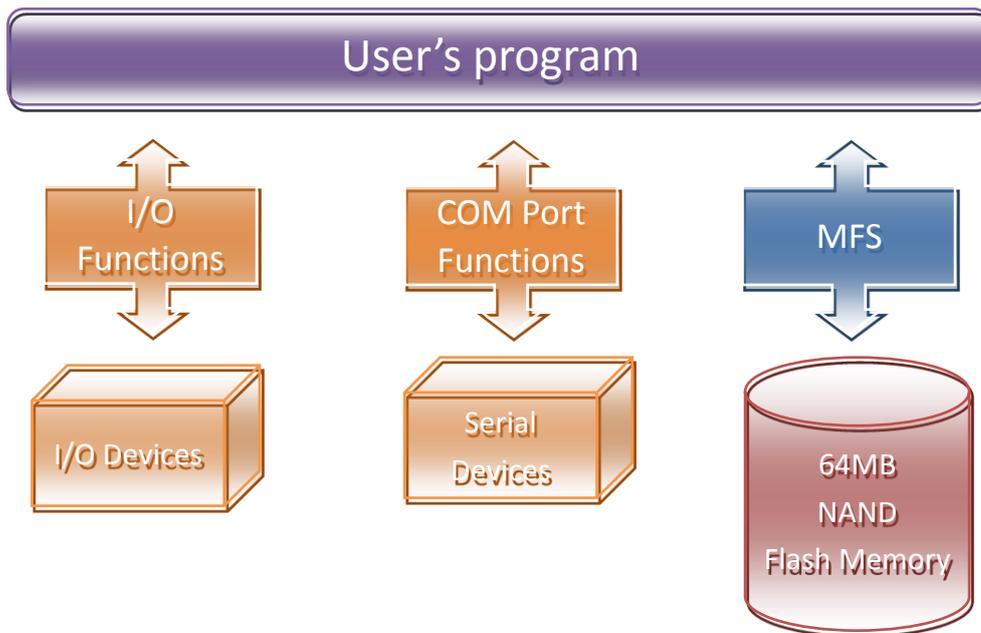
Using the MFS (MiniOS7 File System) library, you can dynamically read/write files from/to the 64MB flash memory. Many kinds of applications related to data logger can be implemented. For example: log analog signal values with timestamp, log RS-232/485 communication data for analysis.

► Supported Product

- uPAC-7186ED-FD
- iP-8441-FD
- IP-8841-FD
- VP-2111/VH-2111P/VH-2211/VH-2311

► Applications

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



► MFS Specifications

Item	Description
Disk number	2 (disk A and B)
Disk size	1/2 size of the flash memory size
File number	456 files max. for each disk
File size	64MB max. for each file
File name	12 bytes max (case sensitive)
File operation modes	Read: Write: Creates a new file to write data, or overwrite a file (if the file is already exist). Append: appends data to a file.
File handle	10 max. for each disk. For read mode: the 10 file handles can all be used for reading operation on each disk. Total 20 files can be opened for reading mode. For write and append mode: only 1 file handle can be used for writing operation on all disks.
Writing verification	Yes. Default is enabled. Calling <code>mfs_EnableWriteVerification</code> and <code>mfs_DisableWriteVerification</code> can change the setting.
Automate file system recovery	Yes. If an unexpected reset or power loss occurs, closed files, and files opened for reading are never at risk. Only data written since the last writing operation (<code>mfs_WriteFile</code> ,) might be lost. When the file system reboots, it restores the file system to its state at the time of the last writing operation.
Writing speed	<code>mfs_WriteFile</code> : 147.5 KB/Sec (verification enabled) (default) 244.0 KB/Sec (verification disabled) <code>mfs_Puts</code> : 142.1 KB/Sec (verification enabled) (default) 229.5 KB/Sec (verification disabled)
Reading speed	<code>mfs_ReadFile</code> : 734.7 KB/Sec <code>mfs_Gets</code> : 414.2 KB/Sec

Max. length of writing data	32767 bytes.
Max. length of reading data	32767 bytes.

► Resources upload

- MFS SDKs

<http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/basic/lib/>

- MFS Demos:

[http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/64mb flash/](http://ftp.lcpdas.com/pub/cd/8000cd/napdos/vp-2000/demo/vp-2111/64mb_flash/)

Appendix D. I-8K and I-87K serial Modules (For VP-2111 module only)

There are 3 slot to expand local I/O. And the I/O modules can be parallel bus type (high profile I-8k series) and serial bus type (high profile I-87k series).

The difference between them is

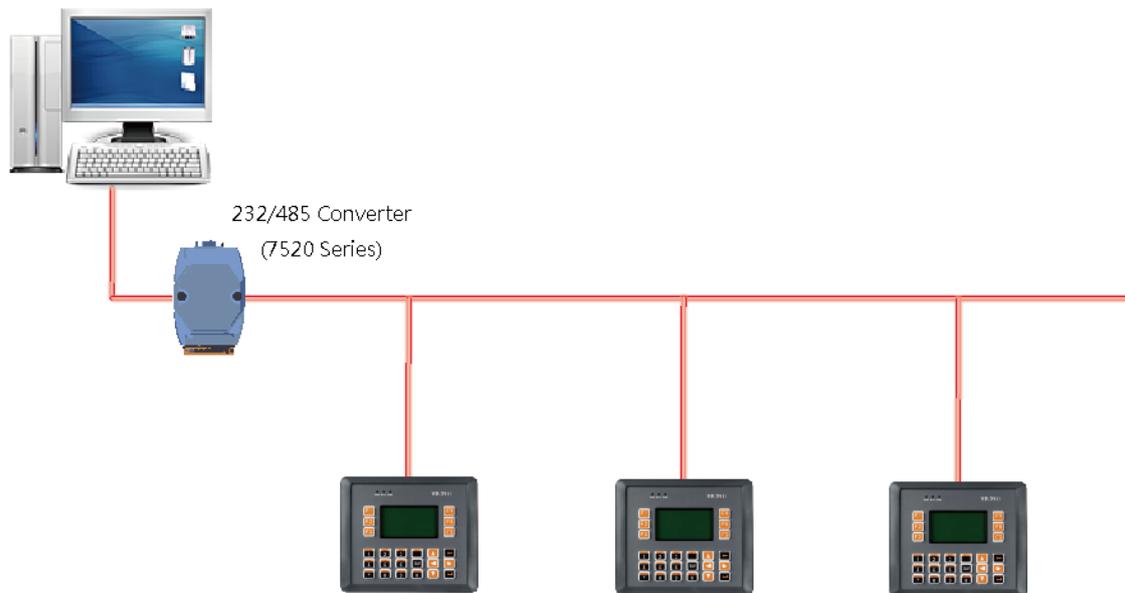
Item	I-8k Series	I-87k Series
Microprocessor	No	Yes (8051)
Communication interface	Parallel bus	Serial bus
Communication speed	Fast	Slow
DI latched function	No	Yes
Counter input (for digital input module)	No	Yes (100 Hz)
Power on value	No	Yes
Safe value	No	Yes
Programmable slew-rate for AO module	No	Yes

Appendix E. Application of RS-485 Network

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

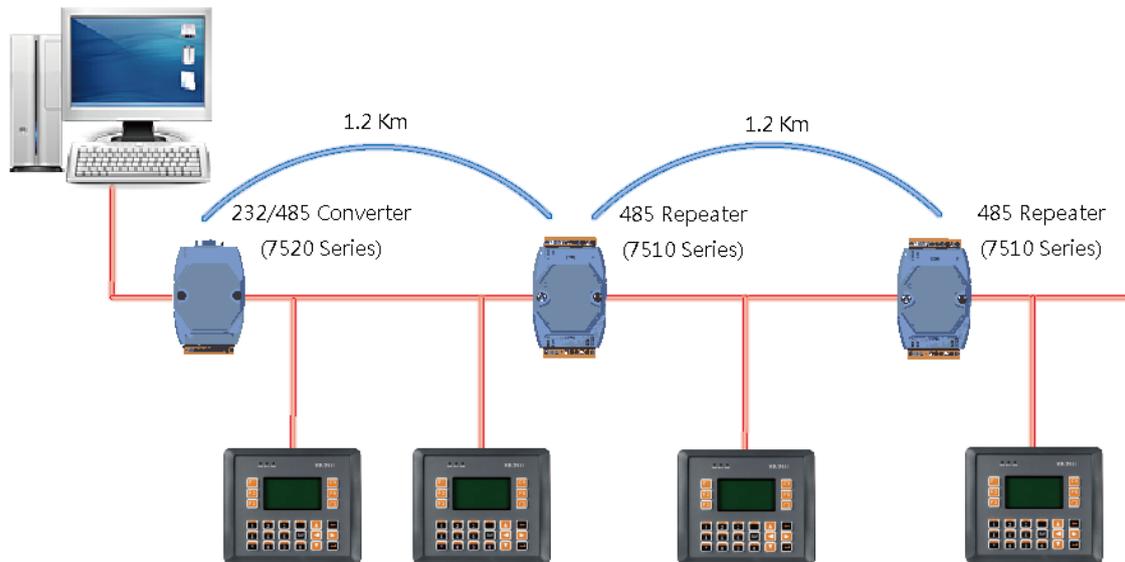
E.1. Basic RS-485 Network

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



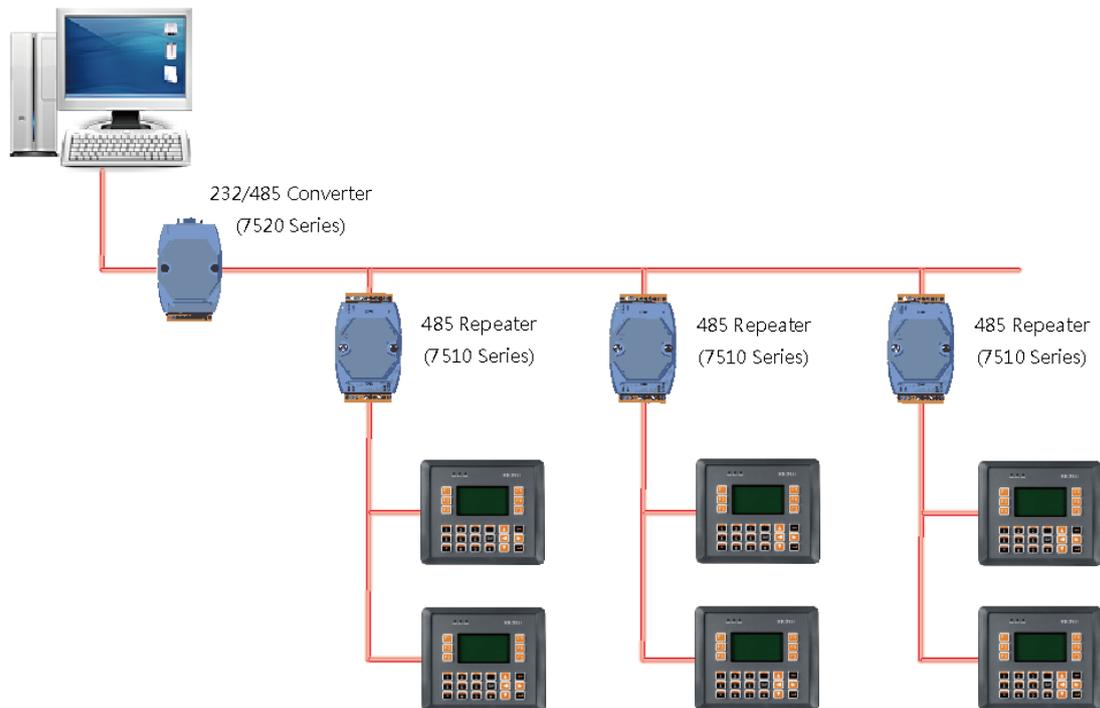
E.2. Daisy Chain RS-485 Network

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

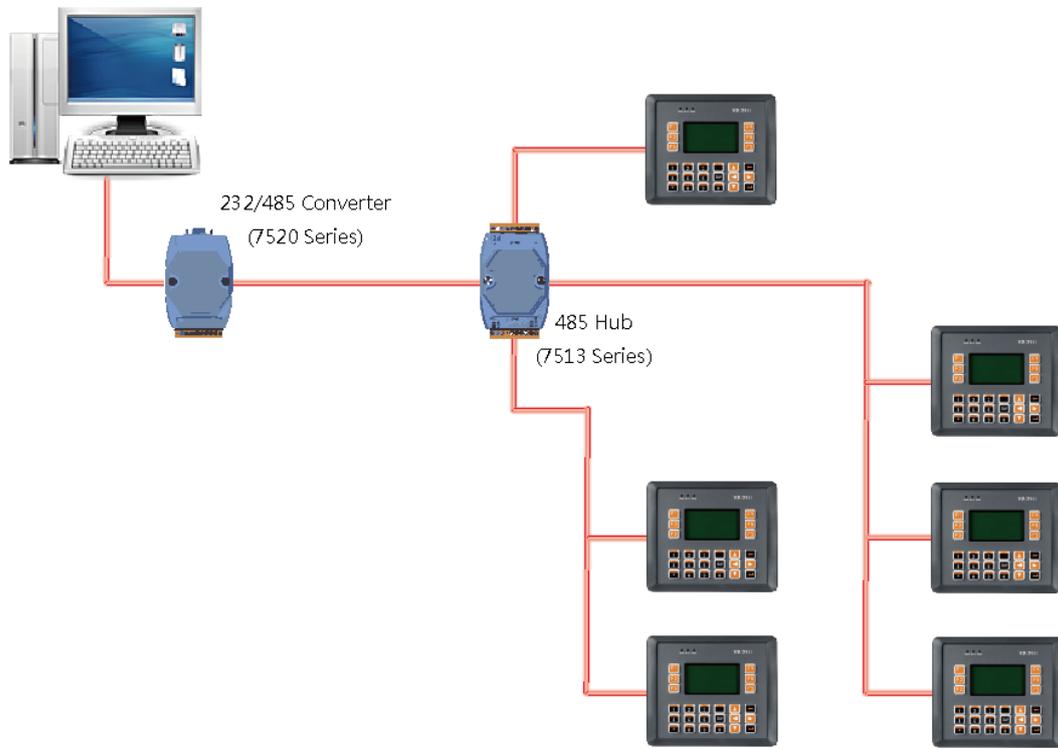


E.3. Star Type RS-485 Network

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

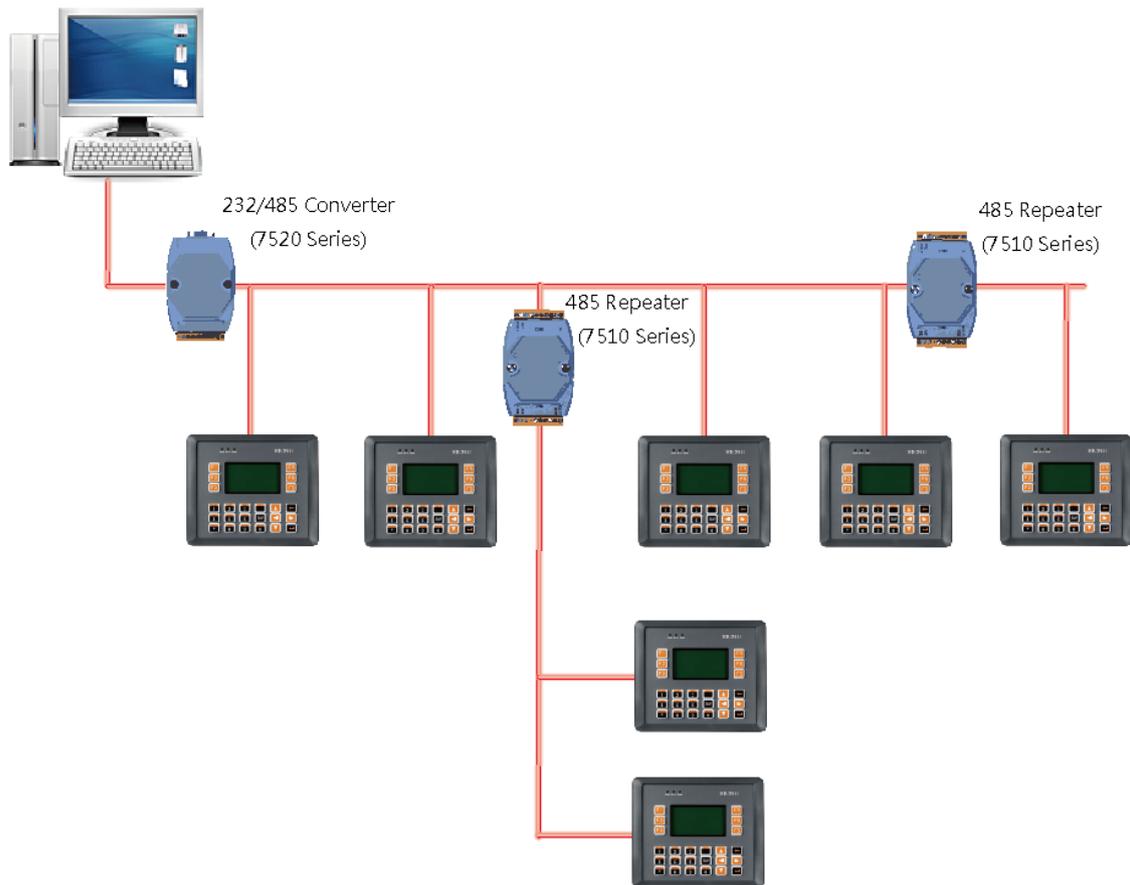


There is a better choice to use 7513 as a RS-485 hub on start type network.



E.4. Random RS-485 Network

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



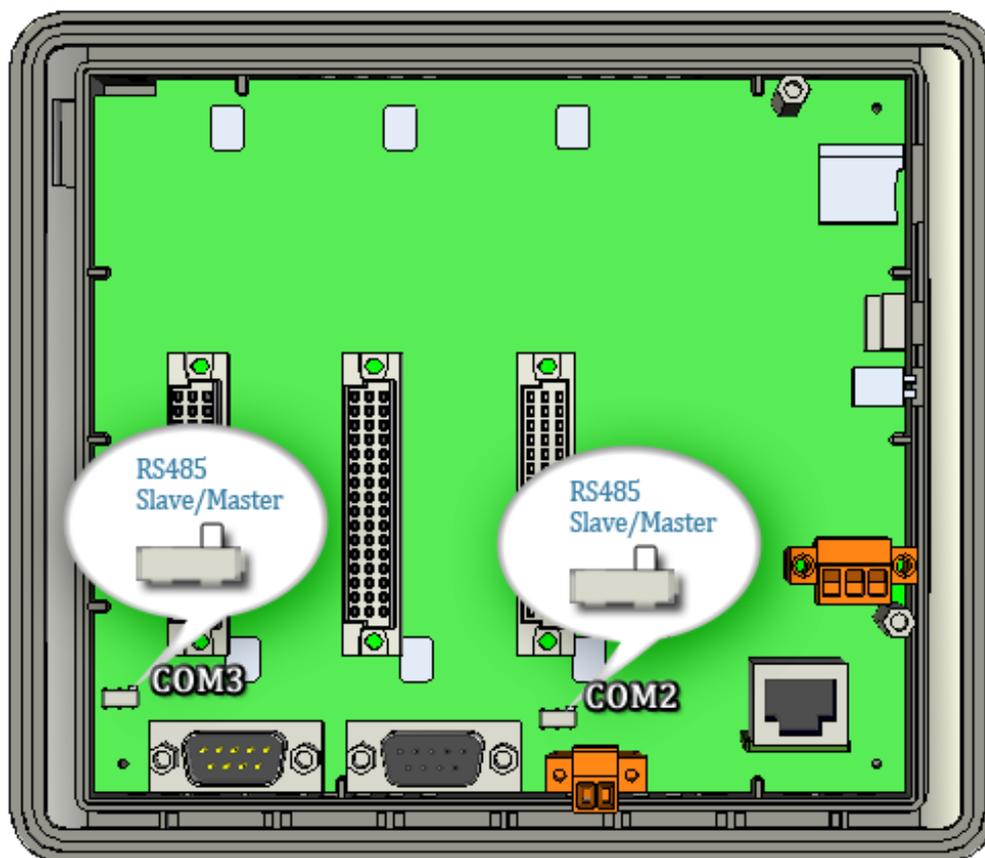
E.5. Master/Slaves Settings

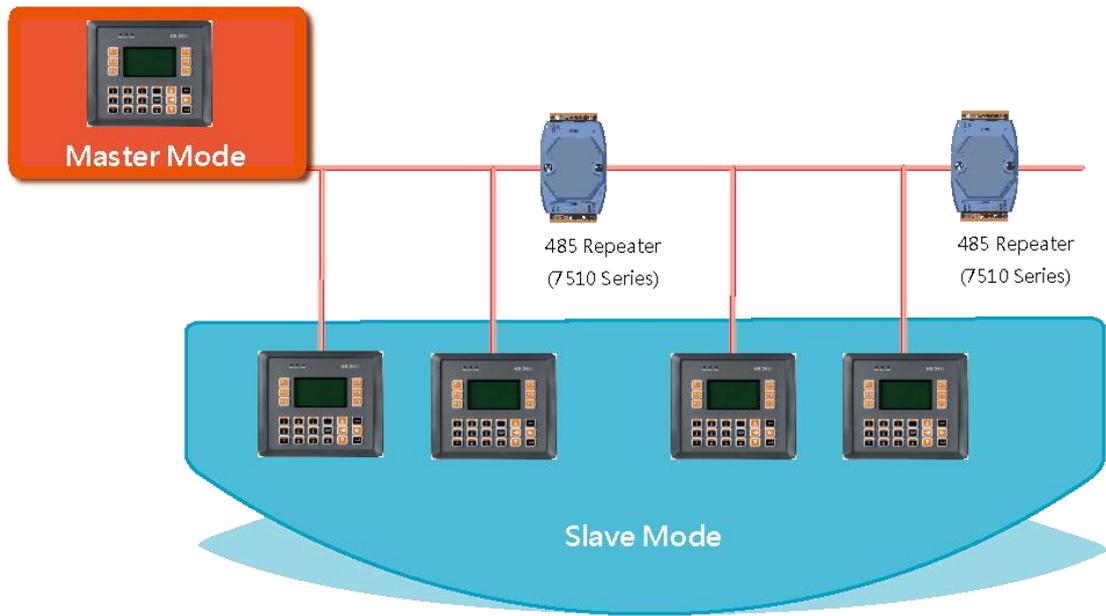
There must exist one master to have a pull-high/pull-low resistor in the same network. In a master/slave applications, “Master” is the default configuration of ViewPAC.

E.5.1. ViewPAC as a Master (default):

When one of ViewPAC is set to the master mode, then all the other devices on the same network must be set to the slave mode.

Set a ViewPAC to the master mode by adjusting the jumpers on the power board of ViewPAC (the pull-high/pull-low resistors are adjusted to be enabled.) Refer to the following figure:

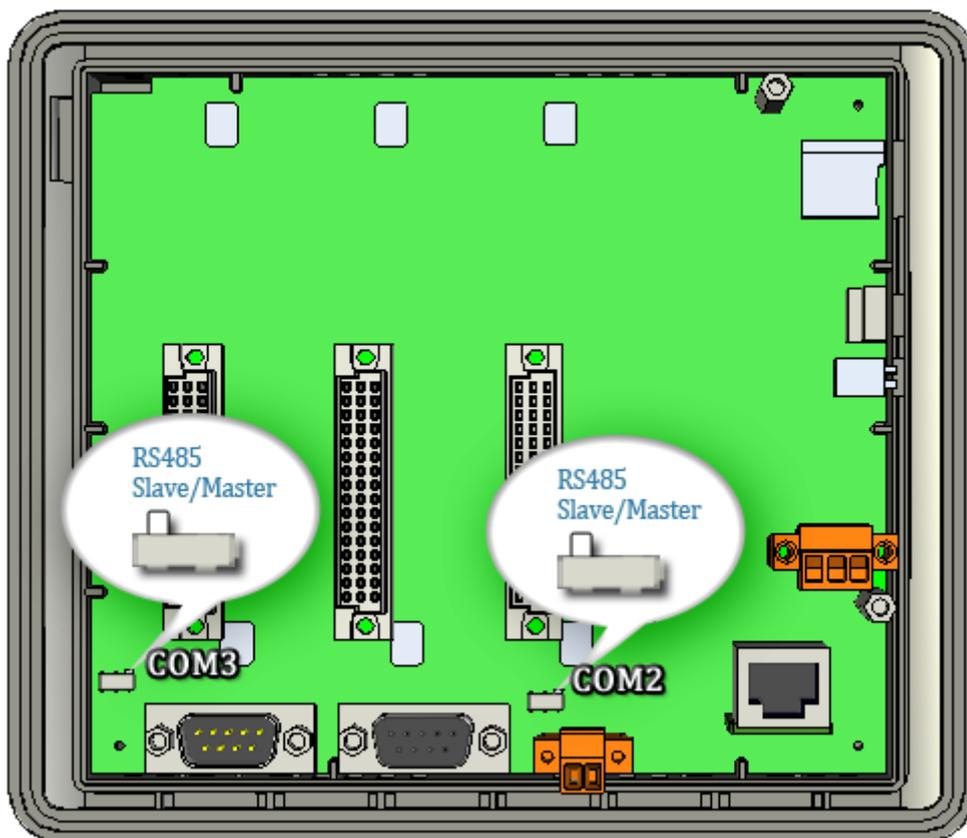


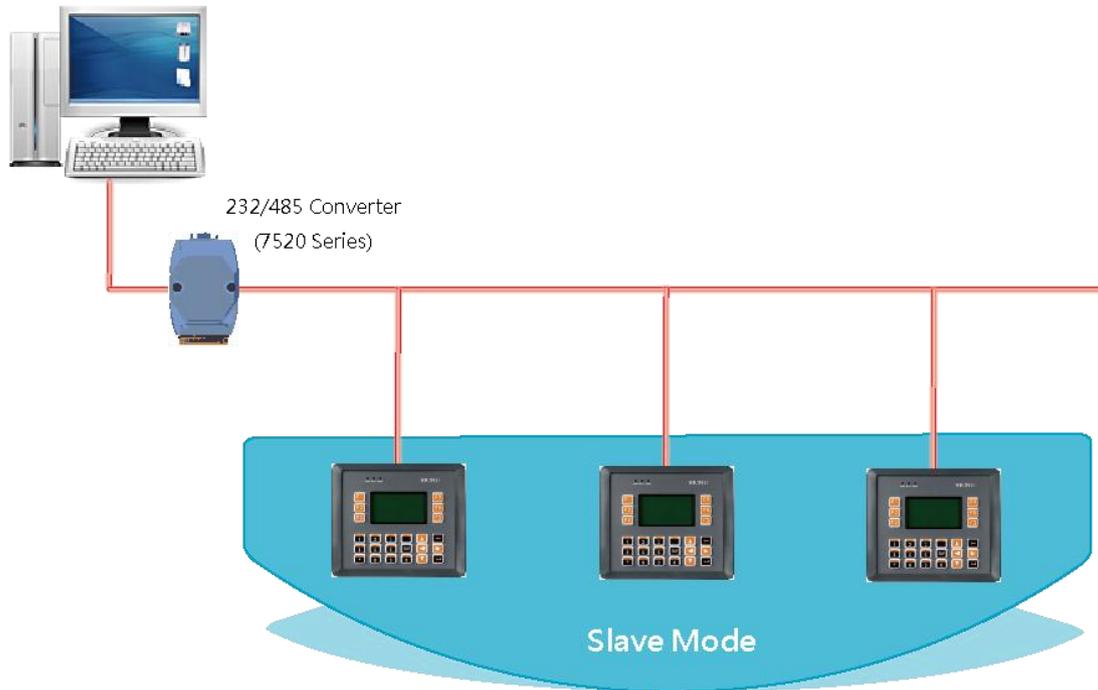


E.5.2. ViewPAC as a Slave:

For most of application, only one 7520 series module is used as RS-232/485 converter, and its pull-high/pull-low resistors are set to be enabled. Then the ViewPAC and all the other devices on this network must be in their slave mode (the pull-high/pull-low resistors must be disabled).

Please refer to the following figure to set the jumpers to the slave mode. The jumpers are located at the power board of ViewPAC





If there are repeaters on the RS-485 network, you can see that there are pull-high/pull-low resistors on both sides of the repeaters (i-7510)

