I-8120 (I-8KCAN)

User Manual

Warranty

All products manufactured by IPC DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to 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, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

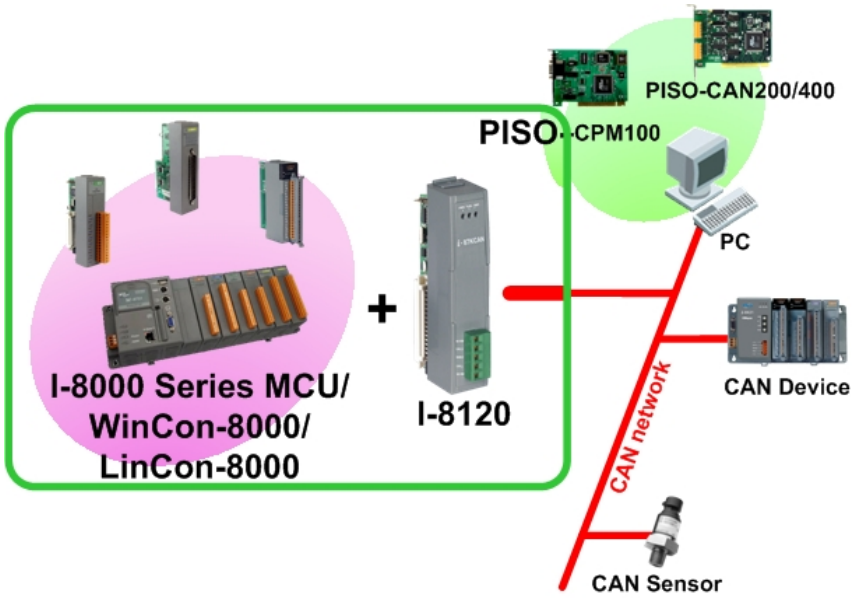
The names used for identification only maybe registered trademarks of their respective companies.

Tables of Content

1 Introduction

1.1 Overview

The CAN (Controller Area Network) is a serial communication bus especially suited to interconnect smart devices to build smart systems or sub-system. As standalone CAN controller, I-8120 with WinCon, LinCon and I-8000 series MCU (main control unit) represents an economic solution. It has one CAN communication ports with 5-pin screw terminal connector, and is useful for a wide range of CAN applications. Besides, I-8120 uses the new Phillips SJA1000T and transceiver 82C250, which provide both CAN 2.0A and 2.0B specific, re-transmission function, bus arbitration and error detection. Combining the benefits of WinCon, LinCon and I-8000 series MCU without increasing the CPU loading heavily, it is a powerful multi CAN port programmable device server by driving the CAN port of I-8120 with dual port RAM. Users can also combine the advantage of WinCon, LinCon and I-8000 series MCU with I-8120, and apply the WinCon, LinCon and I-8000 series MCU on various industrial applications. Therefore, users can design the various applications between different communication protocols.



1.2 Specifications

- CPU:80186, 80MHz
- Philip SJA1000 CAN controller with 16MHz clock
- Philip 82C250 CAN transceiver
- Power LED, Transmission / Reception LED, and Error LED
- 120Ω terminal resister selected by jumper
- CAN bus interface: ISO 11898-2, 5-pin screw terminal with on-board optical isolator protection.
- 2500 Vrms isolation on CAN side
- Power Consumption: 2W
- 8K bytes dual port RAM
- Operating Temperature:-25°C to +75°C
- Storage Temperature:-30°C to +85°C
- Humidity:5%~95%

1.3 Features

Common Features:

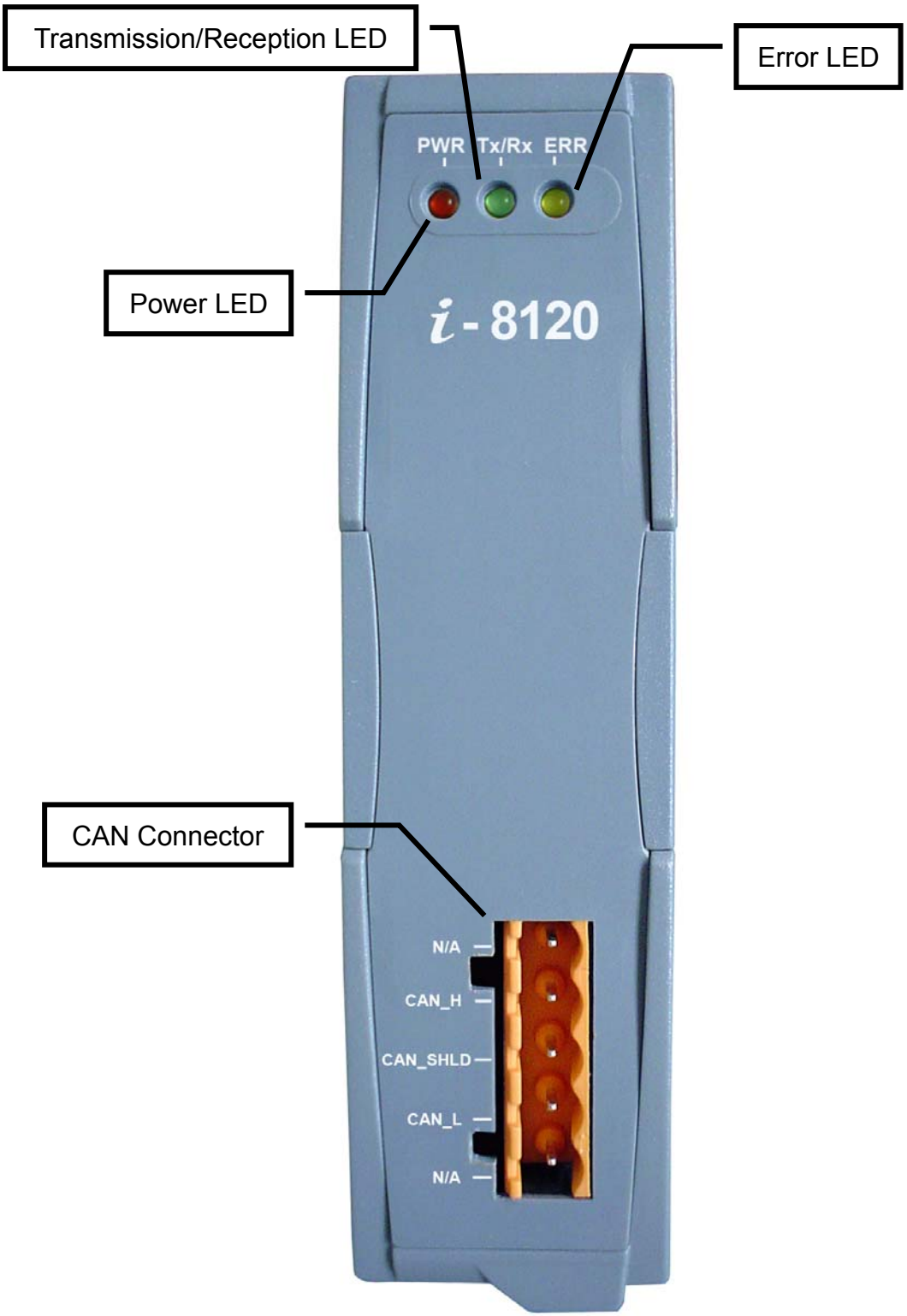
- One CAN port expansion for WinCon-8000, LinCon-8000 and I-8000 series main control unit. **(Note: In I-8000 series main control unit, I-8120 only can be plugged in the first four slots.)**
- Hardware timestamp.
- 2048 records CAN message reception buffer size.
- 256 records CAN message transmission buffer size
- Dual port RAM communication mechanism.
- Provide C/C++ function libraries to send and receive CAN messages in WinCE, Linux and MiniOS7 platform.
- Demos and utility are provided.
- 3 indication LEDs (Power, Transmission/Reception and Error LEDs)

CAN Port Features:

- Support user-defined baud
- 2500 Vrms isolation
- Baud : 5Kbps, 10Kbps, 20Kbps, 25Kbps, 50Kbps, 100Kbps, 125Kbps, 200Kbps, 250Kbps, 500Kbps, 800Kbps, 1Mbps.

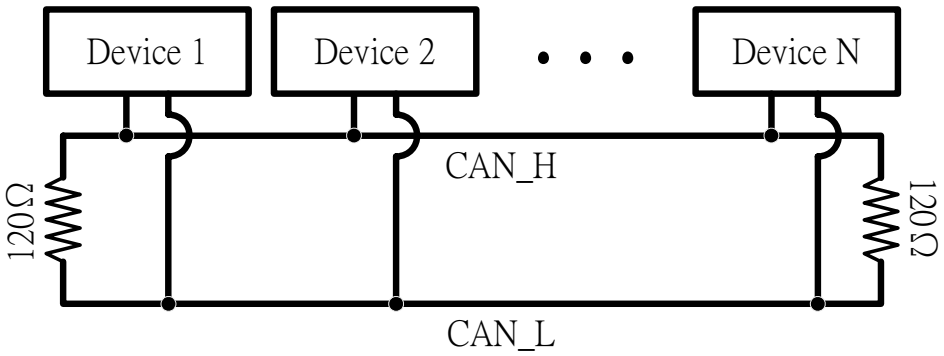
2 Hardware Specification

2.1 Hardware Structure



2.2 Wire Connection

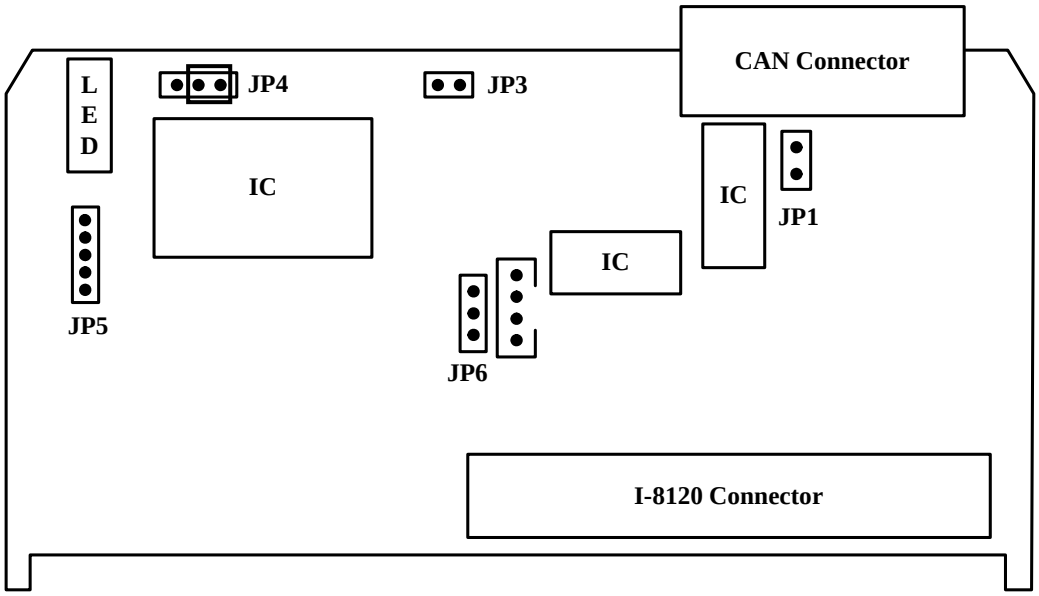
In order to minimize the reflection effects on the CAN bus line, the CAN bus line has to be terminated at both ends by two terminal resistances as in the following figure. According to the ISO 11898-2 spec, each terminal resistance is 120Ω (or between 108Ω~132Ω). The length related resistance should have 70 mΩ/m. Users should check the resistances of the CAN bus, before they install a new CAN network.



Moreover, to minimize the voltage drop over long distances, the terminal resistance should be higher than the value defined in the ISO 11898-2. The following table can be used as a good reference.

Bus Length (meter)	Bus Cable Parameters		Terminal Resistance (Ω)
	Length Related Resistance (mΩ/m)	Cross Section (Type)	
0~40	70	0.25(23AWG)~ 0.34mm^2(22AWG)	124 (0.1%)
40~300	< 60	0.34(22AWG)~ 0.6mm^2(20AWG)	127 (0.1%)
300~600	< 40	0.5~0.6mm^2 (20AWG)	150~300
600~1K	< 20	0.75~0.8mm^2 (18AWG)	150~300

In the I-8120, the 120Ω terminal resistance is supplied. The JP1 of the I-8120 is for the terminal resistance. Its location is shown in the following figure.

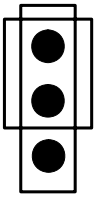


I-8120 hardware location

The following connection statuses are presented for the condition if the terminal resister is enabled or disabled.

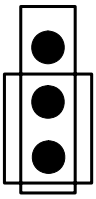
JP1 Jumper selection

When users use I-8120 in Lincon-8000 series MCU and Wincon-8000 series MCU, the JP6 of I-8120 must be set as follows:



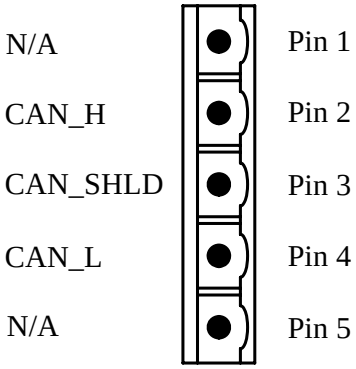
JP6

But if users use I-8210 in I-8000 series MCU, the JP6 of I-8120 must be set as follows:



JP6

The pin assignment of I-8120 CAN bus connector is shown below.



Pin No.	Signal	Description
1	N/A	No use
2	CAN_H	CAN_H bus line (dominant high)
3	CAN_SHLD	Optional CAN Shield
4	CAN_L	CAN_L bus line (dominant low)
5	N/A	No use

2.3 Power LED

I-8120 slot module needs 2W power consumption. If the electric power is supplied normally, the Power LED will be turn on always. If any other situation, please check the power supply or contact to your distributor.

2.4 Tx/Rx LED

Each I-8120 slot module provides Tx/Rx LED to check the CAN messages transmission and reception situation. If the I-8120 is transmitting or receiving a CAN message, the Tx/Rx LED will blink. If I-8120's loading is heavy, the Tx/Rx LED will always turn on.

2.5 ERR LED

The ERR LED indicates the error status of the CAN physical layer and indicates the errors due to missing CAN messages. When the I-8120 ERR LED is turned on, users can use the function GetStatus( ) to obtain the error status and know what is happen. If the ERR LED is turned on due to the software buffer overflow, the buffer error flag can be clear by using the function ClearStatus(). For more detail about these functions, please refer to the section 3.

3 I-8120 Function Library

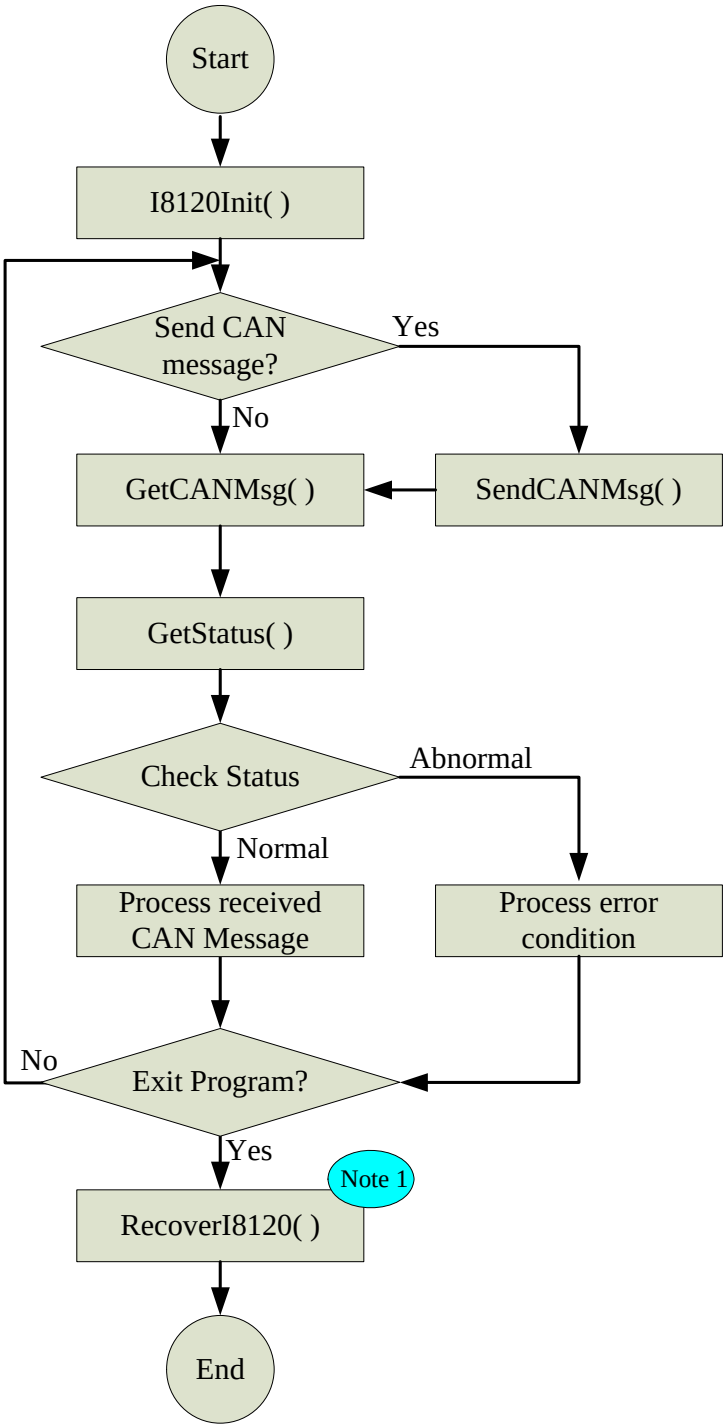
3.1 Function List

In order to use the I-8120 more easily, the I-8120 function library is provided. There are several function libraries for different compiler, such as EVC (for WinCon-8000), GCC (for LinCon-8000) and BC/TC (for I-8000 series MCU). Users can use these functions to control the I-8120. The following table shows the all functions provided by the I8120 library.

Function Name	Description
SetCANBaud	Set the I-8120 CAN baud rate
GetCANBaud	Obtain the CAN baud rate stored in the I-8120
SetCANMask	Set the I-8120 CAN message filter
GetCANMask	Obtain the CAN message filter stored in the I-8120
ClearStatus	Clear the I-8120 software buffer overflow status
GetStatus	Get the I-8120 software buffer or CAN status
ResetI8120	Reset the I-8120 module
I8120Init	Initiate the I-8120 module. Users need to set the proper CAN and I-8120 configuration to the I-8120
GetCANMsg	Obtain the CAN message received by the I-8120
SendCANMsg	Send a CAN message to the CAN network
RecoverI8120	Recover the parameters of system. This function is only supported by LinCon and only exists in the GCC library.

3.2 Programmable Flow Chart

If users want to develop the program with I-8120 module, the following procedure may be a good reference.



Note1: This function is only supported by LinCon and I-8000MCU. When the program is terminal in LinCon or I-8000 MCU, this function must be called to recover the system parameters.

3.3 SetCANBaud

- **Description:**
  Call this function to set the I-8120 CAN baud rate.

- **Syntax:**
  int SetCANBaud(unsigned char Slot, unsigned long Baud,
  unsigned char BT0, unsigned char BT1)

- **Parameters:**
  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.
  Baud: I-8120 slot module has several predefine CAN baud rates. Use this parameter to decide what kind of baud rate users want to use. Here, twelve kinds of baud rates are supported. They are 5K, 10K, 20K, 25K, 50K, 100K, 125K, 200K, 250K, 500K, 800K and 1000K bps. For example, set the Baud value to 250000L to set I-8120 CAN baud to 250K bps. The letter L means that the value 250000 is the long integer format. If users can't find out the proper CAN baud rate for their application, the user-defined baud rate functionality may be needed. The parameters BT0 and BT1 are specially used for setting the user-defined baud rate. When users call the function SetCANBaud() without using the Baud value listed above, the Baud value is useless, and the parameters BT0 and BT1 will be applied for user-defined CAN baud rate.

  BT0, BT1: These parameters are useful for user-defined CAN baud rate. The values of BT0 and BT1 need to be calculated according the SJA1000 CAN controller datasheet. For more information about how to calculate the CAN baud, please refer to the following web site. Here, the 16M clock is used for SJA1000.
  http://www.semiconductors.philips.com

- **Return:**
  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR
  CAN8K_TIMEOUT

3.4 GetCANBaud

- **Description:**
  Call this function to get the CAN baud used by the I-8120.

- **Syntax:**
  int GetCANBaud(unsigned char Slot, unsigned long *Baud,
  unsigned char *BT0, unsigned char *BT1)

- **Parameters:**
  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.
  * Baud: The pointer for obtain the CAN baud rate. If users use the predefine baud rate. The function GetCANBaud() will return the CAN baud rate used by I-8120. If the return value of Baud is 0, it means that the I-8120 use user-defined CAN baud. In this case, the CAN baud rate will be indicated by using the parameters *BT0 and *BT1.
  * BT0, * BT1: The pointer for obtain the user-defined CAN baud used by I-8120.

- **Return:**
  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR

3.5 SetCANMask

- **Description:**

  Use this function to set the I-8120 message filter. This message filter is hardware filter in CAN controller, SJA1000.

- **Syntax:**

  int SetCANMask(unsigned char Slot, unsigned long AccCode,
  unsigned long AccMask)

- **Parameters:**

  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.

  AccCode, AccMask: The AccCode is used for deciding what kind of CAN message ID the CAN controller will accept. The AccMask is used for deciding which bit of CAN message 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 CAN message ID need to be checked, and that CAN message ID bit value needs to match the bit of AccCode in the same position. The following table shows each situation of AccCode and AccMask.

For 11-bit ID Message:

AccCode and AccMask	Bit Position	Filter Target
low byte of the low word	bit7~bit0	bit10 ~ bit3 of ID
high byte of the low word	bit7~bit5	bit2 ~ bit0 of ID
high byte of the low word	bit4	RTR
high byte of the low word	bit3~bit0	no use
low byte of the high word	bit7~bit0	bit7 ~ bit0 of 1st byte data
high byte of the high word	bit7~bit0	bit7 ~ bit0 of 2nd byte data

For 29-bit ID Message:

AccCode and AccMask	Bit Position	Filter Target
low byte of the low word	bit7~bit0	bit28~ bit21 of ID
high byte of the low word	bit7~bit0	bit20 ~ bit13 of ID
low byte of the high word	bit7~bit0	bit12 ~ bit5 of ID
high byte of the high word	bit7~bit3	bit4 ~ bit0 of ID
high byte of the high word	bit2	RTR
high byte of the high word	bit1~bit0	no use

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.)

- **Return:**

  CAN8K_OK

  CAN8K_SLOT_NUM_ERROR

  CAN8K_TIMEOUT

3.6 GetCANMask

- **Description:**
  Call this function to get the CAN message filter situation.

- **Syntax:**
  int GetCANMask(unsigned char Slot, unsigned long *AccCode,
  unsigned long *AccMask)

- **Parameters:**
  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.
  *AccCode, *AccMask: These pointer for obtain the AccCode and AccMask used by the I-8120. For more information about these two parameters, please refer to the section 3.5.

- **Return:**
  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR

3.7 ClearStatus

- **Description:**

  This function is used for cleaning the overflow error flags of the CAN transmission software buffer and CAN reception software buffer. When the CAN transmission software buffer is full, the CAN transmission overflow error flag will be set to 2. If the CAN reception software buffer is full, the CAN reception overflow error flag will be set to 1. The return value may be 3 if both the CAN transmission and reception software buffers are full. In this case, users need to use this function to clear the error flag to acknowledge the error information. Afterward, the Err LED will turn off if there is no error detected by I-8120.

- **Syntax:**

  int ClearStatus(unsigned char Slot)

- **Parameters:**

  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.

- **Return:**

  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR

3.8 GetStatus

- **Description:**
  Read I-8120 CAN controller status and software buffer error flag.

- **Syntax:**
  int GetStatus(unsigned char Slot, unsigned char *CANReg,
  unsigned char *OverflowFlag)

- **Parameters:**
  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.
  * CANReg: The pointer for obtain the I-8120 current CAN controller status. For the information about the CANReg value meaning, 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 and host command buffer overflow flag information. For the information about the OverflowFlag value meaning, please refer to the following table.

Bit NO.	Description
Others	Reserved
1	1 for host command buffer overflow. 0 for normal.
0 (LSB)	1 for CAN receive message buffer overflow. 0 for normal.

- **Return:**
  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR

3.9 ResetI8120

- **Description:**

  This function is used to reset the I-8120 module. Calling this function will clear not only the CAN controller register error flag but also CAN and host command software buffer error flags. When the CAN controller is bus-off, it may be called. After applying this function, the CAN controller of I-8120 will be restart, and wait for receiving and transmitting CAN messages.

- **Syntax:**

  int ResetI8120(unsigned char Slot)

- **Parameters:**

  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.

- **Return:**

  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR

3.10 I8120Init

- **Description:**

  When users want to use I-8120 module, this function must be called first. Afterwards, The CAN baud, CAN message filter and I-8120 configurations will be applied to the I-8120.

- **Syntax:**

  int I8120Init(unsigned char Slot, unsigned long CANBaud,
  unsigned char BT0, unsigned char BT1,
  unsigned long CAN_AccCode,
  unsigned long CAN_Mask)

- **Parameters:**

  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.

  CANBaud: Set this parameter to configure I-8120 CAN baud rate. For example, use the value 250000L to set CAN baud rate to 250K bps. The letter “L” means long integer format.

  BT0, BT1: These parameters are useful for user-defined CAN baud rate. The values of BT0 and BT1 need to be calculated according the SJA1000 CAN controller datasheet. For more information about how to calculate the CAN baud, please refer to the following web site. Here, the 16M clock is used for SJA1000.

  http://www.semiconductors.philips.com

  CAN_AccCode, CAN_Mask: Set this parameter for CAN message filter. The using method is the same with the AccCode and AccMask parameters of function SetCANMask. Therefore, please refer to the section 3.5 to know about the CAN message filter configuration.

- **Return:**

  CAN8K_OK
  CAN8K_SLOT_NUM_ERROR
  CAN8K_TIMEOUT

3.11 GetCANMsg

- **Description:**

  If the CAN message is received by I-8120 module, use this function to read back the CAN message. If the FIFO is full, the error code is given. Users need to use ClearStatus buffer to clear this status.

- **Syntax:**

  int GetCANMsg(unsigned char Slot, unsigned char *Mode,
  unsigned long *ID, unsigned char *RTR,
  unsigned char *DataLen, unsigned char *Data)

- **Parameters:**

  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.

  * Mode: The parameter *Mode is used to point the CAN message specification. If the CAN message is CAN 2.0A specification, the Mode value is given to 0. If it is CAN 2.0B specification, the value is 1.

  *ID: The parameter *ID is used to point to the attribution ID of a CAN message. If the I-8120 receive a CAN message with specification 2.0A, the ID value range is from 0x0 to 0x3FF. If CAN message is 2.0B, the ID value range is from 0x0 to 0x1FFFFFFF.

  *RTR: This parameter is used to get the RTR status of a CAN message. The RTR value is 0 if the CAN message is normal CAN frame. If it is 1, the CAN message is remote-transmit-request CAN frame.

  *DataLen: The *DataLen is point to the value indicates how many data stored in the array Data[ ] (see the parameter *Data below). The maximum value is 8.

  *Data: This parameter needs a pointer to point an array which is used to stored the CAN message data. Because the maximum data numbers of one CAN message is 8 bytes, it is recommend that the array Data[ ] size is 8 bytes.

Return:

CAN8K_OK
CAN8K_NOT_INIT
CAN8K_SLOT_NUM_ERROR
CAN8K_FIFO_FULL
CAN8K_FIFO_EMPTY

3.12 SendCANMsg

- **Description:**

  If the CAN message is received by I-8120 module, use this function to read back the CAN message.

- **Syntax:**

  int SendCANMsg(unsigned char Slot, unsigned char Mode,
  unsigned long ID, unsigned char RTR,
  unsigned char DataLen, unsigned char* Data)

- **Parameters:**

  Slot: The number of the slot (0~7 for WinCon and LinCon, 0~3 for I-8000) where the I-8120 is plugged.

  Mode: The parameter Mode is used set the transmitting CAN message specification. If the CAN message is CAN 2.0A specification, the Mode value is 0. If it is CAN 2.0B specification, the value is 1.

  ID: The parameter ID is used to set the attribution ID of a CAN message. If the transmitting CAN message is for specification 2.0A, the ID value range is from 0x0 to 0x3FF. If the CAN message is for specification 2.0B, the ID value range is from 0x0 to 0x1FFFFFFF.

  RTR: Use this parameter to set the RTR status of the transmitting CAN message. The RTR value is 0 if the CAN message is normal CAN frame. If it is 1, the CAN message is remote-transmit-request CAN frame.

  DataLen: The DataLen indicates how many data will be transmitted in the array Data[ ] (see the parameter *Data below). The maximum value of DataLen is 8.

  *Data: This parameter needs a pointer to point an array. This array include the date which will be transmitted to the CAN network. Because the maximum data numbers of a CAN frame is 8 bytes, more than 8 bytes data in the array Data[ ] will be ignored.

- **Return:**

  CAN8K_OK
  CAN8K_NOT_INIT
  CAN8K_SLOT_NUM_ERROR
  CAN8K_FIFO_FULL
  CAN8K_TIMEOUT;

3.13 RecoverI8120

- **Description:**
  Before users want to terminal the users' program, this function must be called to recover the system resource. This function is only supported by LinCon-8000 and I-8000 series main control unit.

- **Syntax:**
  void RecoverI8120(void)

- **Parameters:**
  None

- **Return:**
  None

3.14 Function Return Error Code

The following table displays the function return error codes which may return from the functions provided by I8120 library.

Error code	value	Description
CAN8K_OK	0	OK
CAN8K_TIMEOUT	21	No message response before time is over.
CAN8K_FIFO_EMPTY	22	The software buffer which stores the CAN messages is empty.
CAN8K_FIFO_FULL	23	The software buffer which stores the CAN message is full.
CAN8K_SLOT_NUM_ERROR	27	The parameters Slot of functions is invalid for users WinCon/ LinCon/I-8000.
CAN8K_NOT_INIT	28	Users don’t call the I8120Init function before using any other function.

4 Demo Programs

The following architecture is shown in the I_8120 folder.

```
|--\manual              → Users manual
|--\WinCE               → demos and library for WinCon series MCU
   |--\WinConLib        → WinCon library
   |--\EVC              → I-8120 demos and library for EVC++ 4.0
      |--\I8120Lib      → I-8120 library for WinCon series MCU
      |--\Demos         → I-8120 demos for EVC++ 4.0
|--\Linux               → demos and library for LinCon series MCU
   |--\GCC              → I-8120 demos and library for GCC
      |--\I8120Lib      → I-8120 library for LinCon series MCU
      |--\Demos         → I-8120 demos for GCC
|--\I_8000              → demos and library for I-8000 series MCU
   |--\I8120Lib         → I-8120 library for I-8000 series MCU
   |--\MCU_Lib          → I-8000 series MCU library
   |--\Download_Tool    → I-8000 series MCU program download tool
   |--\MCU_OS           → I-8000 series MCU OS image
      |--\40MHzCPU      → I-8000 series MCU with 40MHz CPU OS image
      |--\80MHzCPU      → I-8000 series MCU with 40MHz CPU OS image
  |--\Demos             → I-8120 demo programs
      |--\BCPP31        → demos for Borland C++ 3.1
      |--\TCPP101       → demos for Turbo C++ 1.01
      |--\MSC6          → demos for MSC 6.0
```

Here, we provide the demo programs about how to use the I-8120 library in I-8000 series MCU, WinCon series MCU and LinCon series MCU. For I-8000 series MCU, the demo program for BC++3.1 (Borland C++ version 3.1), TC++1.01 (Turbo C++ version 1.01), and MSC6 (Microsoft C++ 6.0) compliers is given. In WinCon series MCU, the demos for EVC++ 4.0 (Embedded Visual C++) are provided. In LinCon series MCU, we provide the GCC demo program. The step-by-step demo procedure for EVC++ and GCC will be given in the following section.

All of these demos may give a good model to show how to build an

execution file with I8120 library. Take a note that if users don't have any program development tools, the TC++ 1.01 and EVC++ 4.0 can be free download form the following web site.

Download TC++ 1.01:
http://community.borland.com/museum

Download EVC++ 4.0:
http://www.microsoft.com/downloads/details.aspx?familyid=1DACDB3D-50D1-41B2-A107-FA75AE960856&displaylang=en

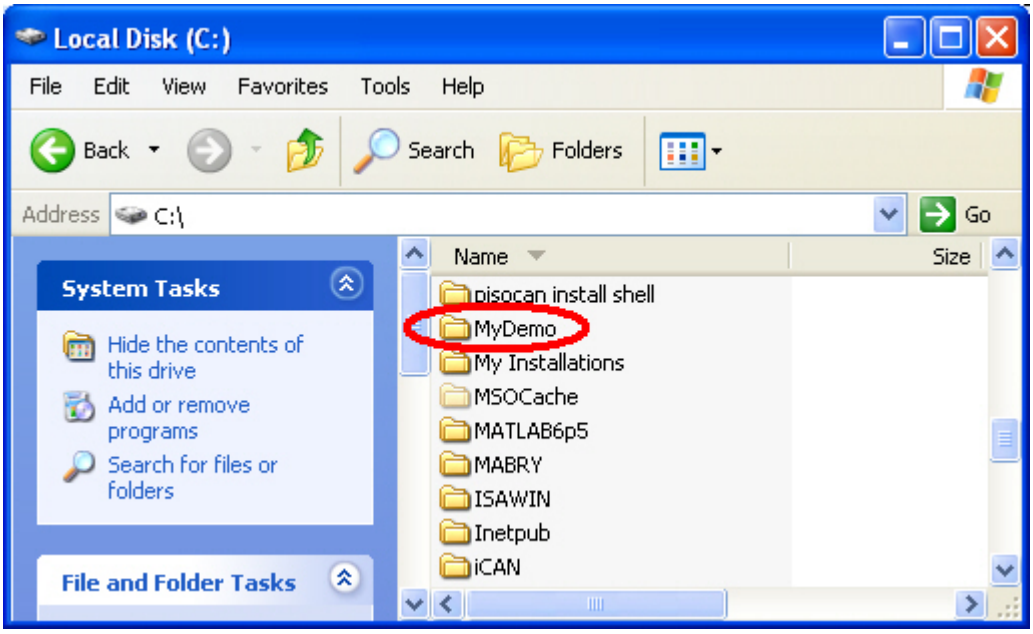
The LinCon program development tool, GCC, can also free download on the ICPDAS web site:

http://ftp.icpdas.com.tw/pub/cd/linconcd/napdos/linux/sdk/

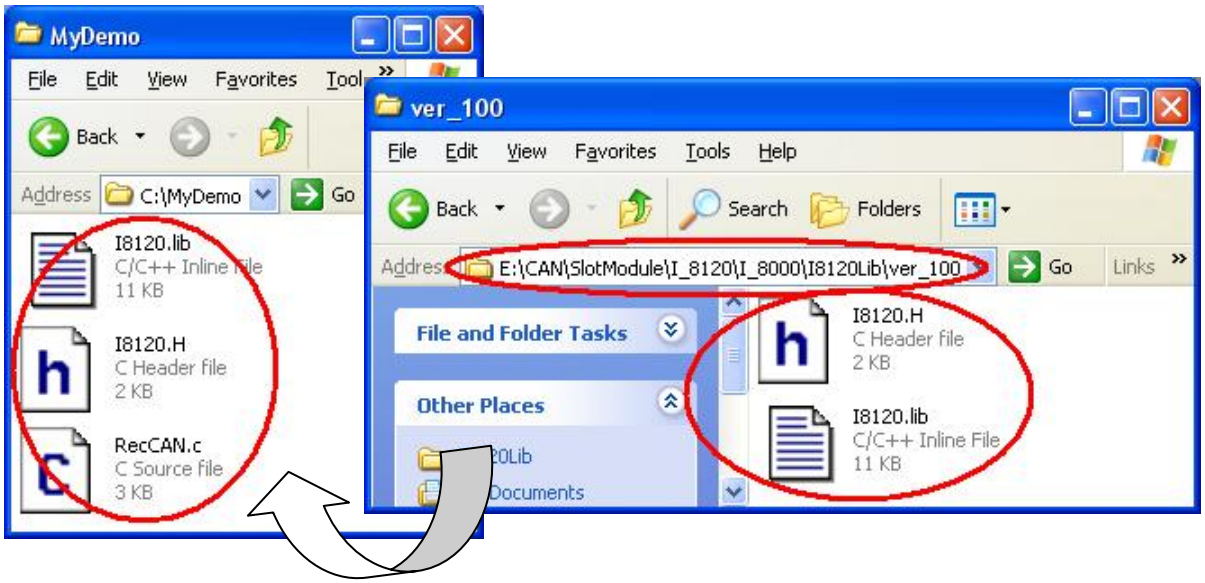
4.1 TC++1.01 Demo For I-8000 Series MCU

Here, it is considered that how to build an execution file with C8KH100L.lib and how to run this program on the I-8000 series MCU. In this demo, the I-8811 MCU with 40 MHz and TC++1.01 compiler will be used. The procedure for all the other I-8000 MCUs will be the same with this demo.

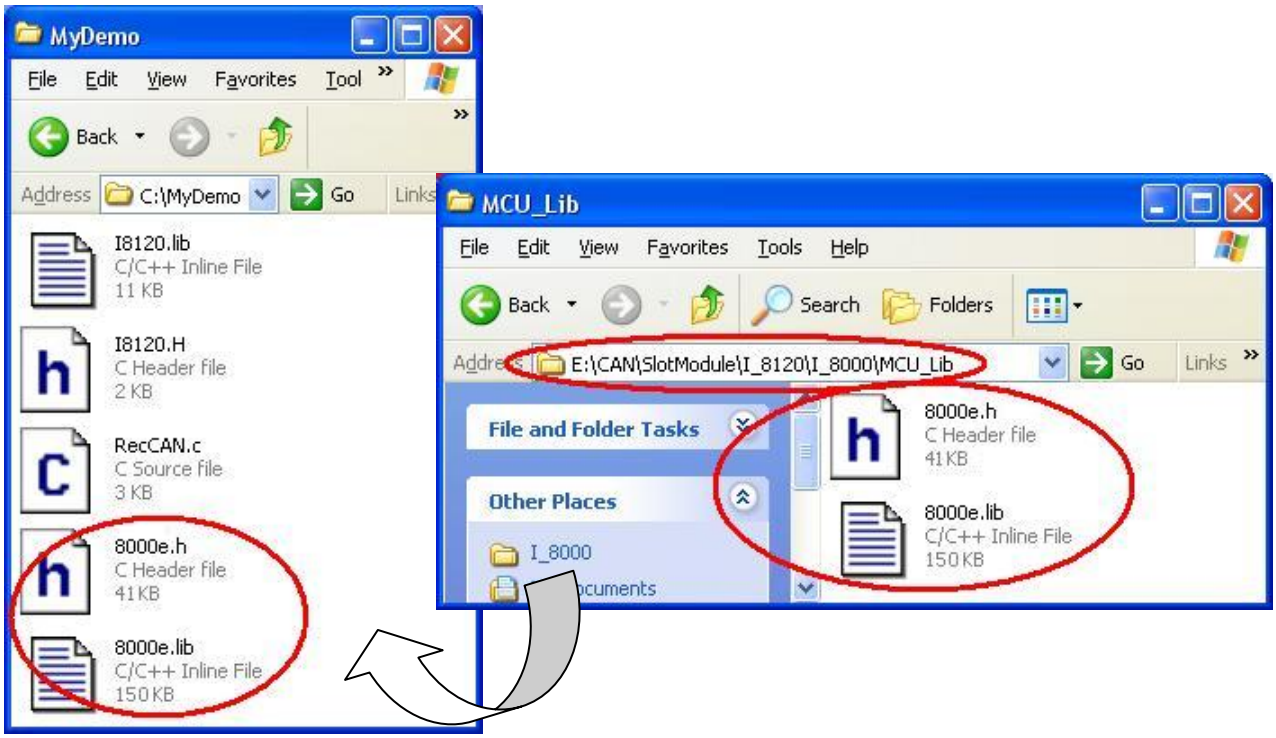
Step1: Create a folder named “MyDemo” in the C disk.



Step2: Copy users’s .c file, and I-8120 library files (I8120.lib and I8120.h) into MyDemo folder. Users can find them with version 1.00 in the path CAN\SlotModule\I_8120\I_8000\I8120Lib\ver_100 in CAN product CD.



Step3: Copy I-8000 series MCU library files (8000.h and 8000e.lib) into MyDemo folder, too. Users can find them in the path CAN\SlotModule\I_8120\I_8000\MCU_Lib in CAN product CD.

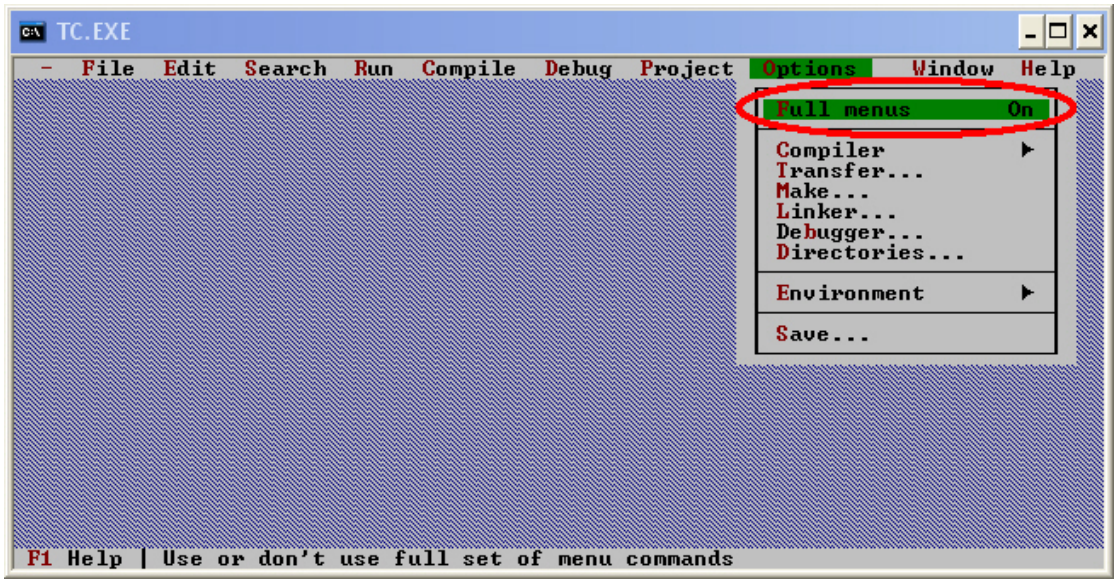


Step4: Open your .c file with Notepad. Confirm the 8000.h and I8120.h path in the “#include” syntax. They are shown below.

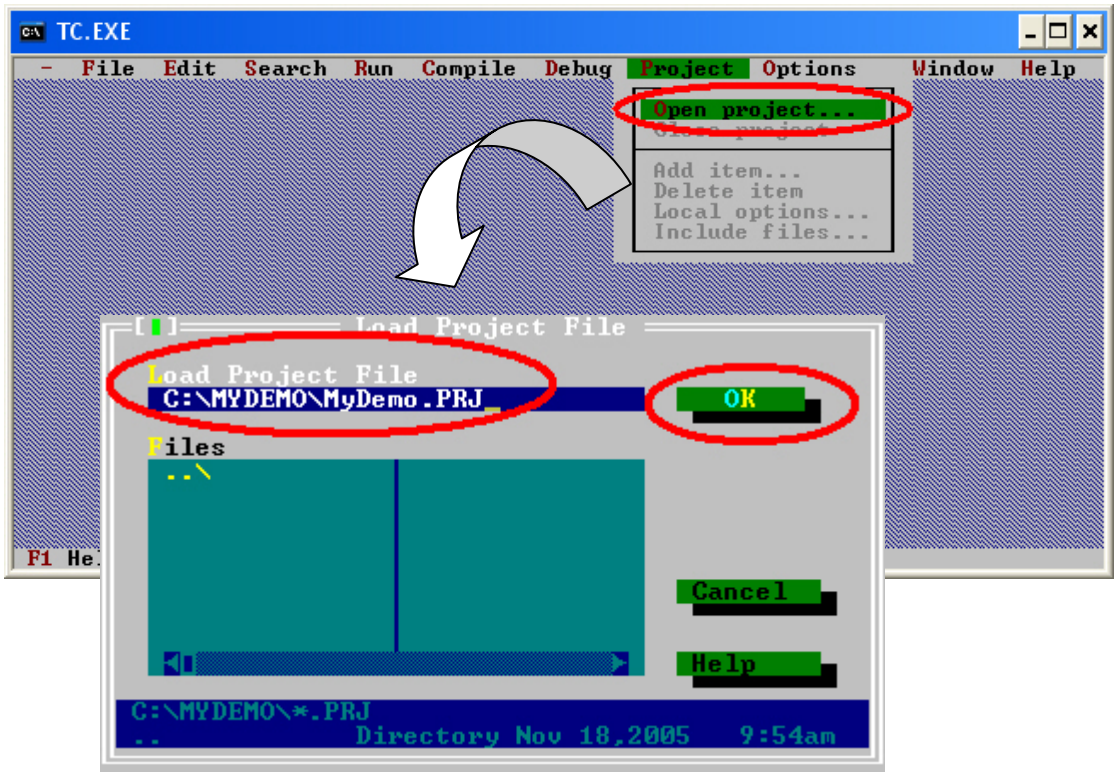
#include “8000.h”

#include “I8120.h”

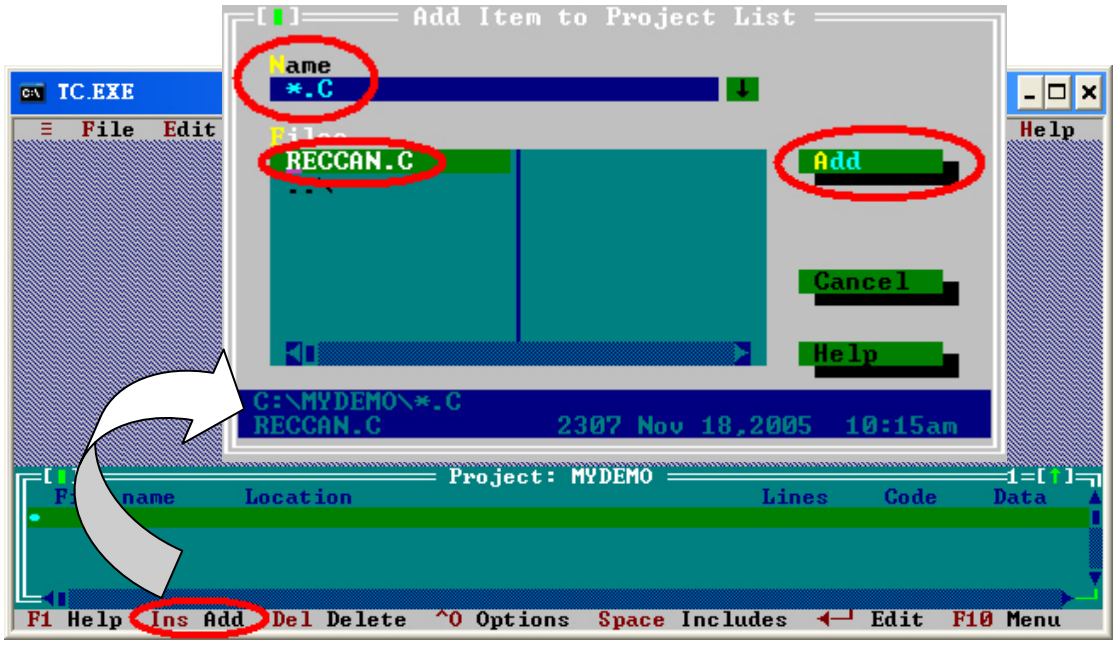
Step5: Run the TC++1.01 development environment. Click the “Options\Full menus” to expand the all functions list in the menus.



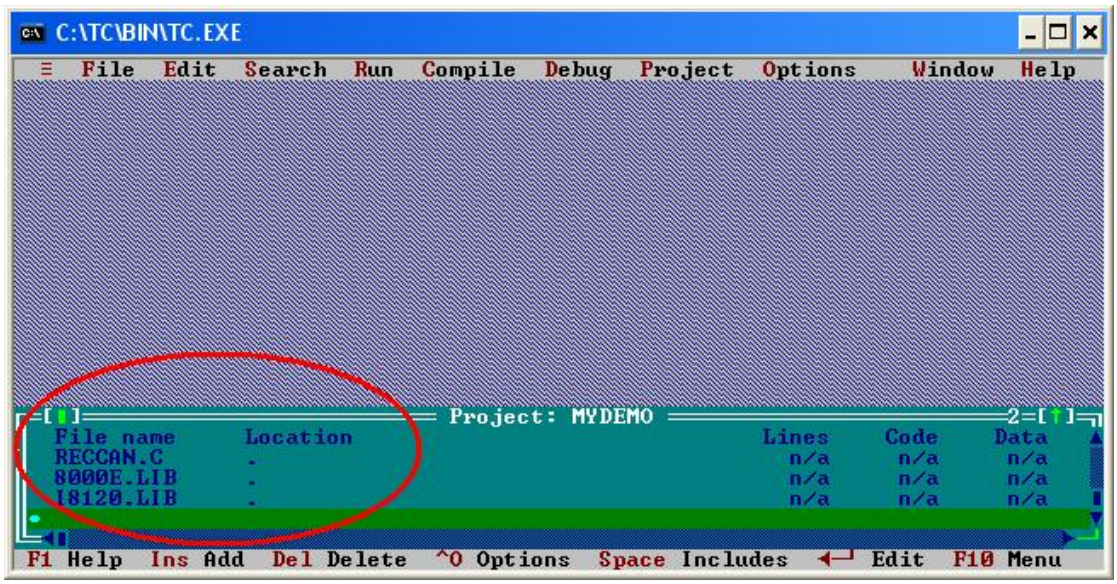
Step6: Click the “Project\Open project…” to create a new project. Input the project name “MyDemo.PRJ”, and click OK button to continue.



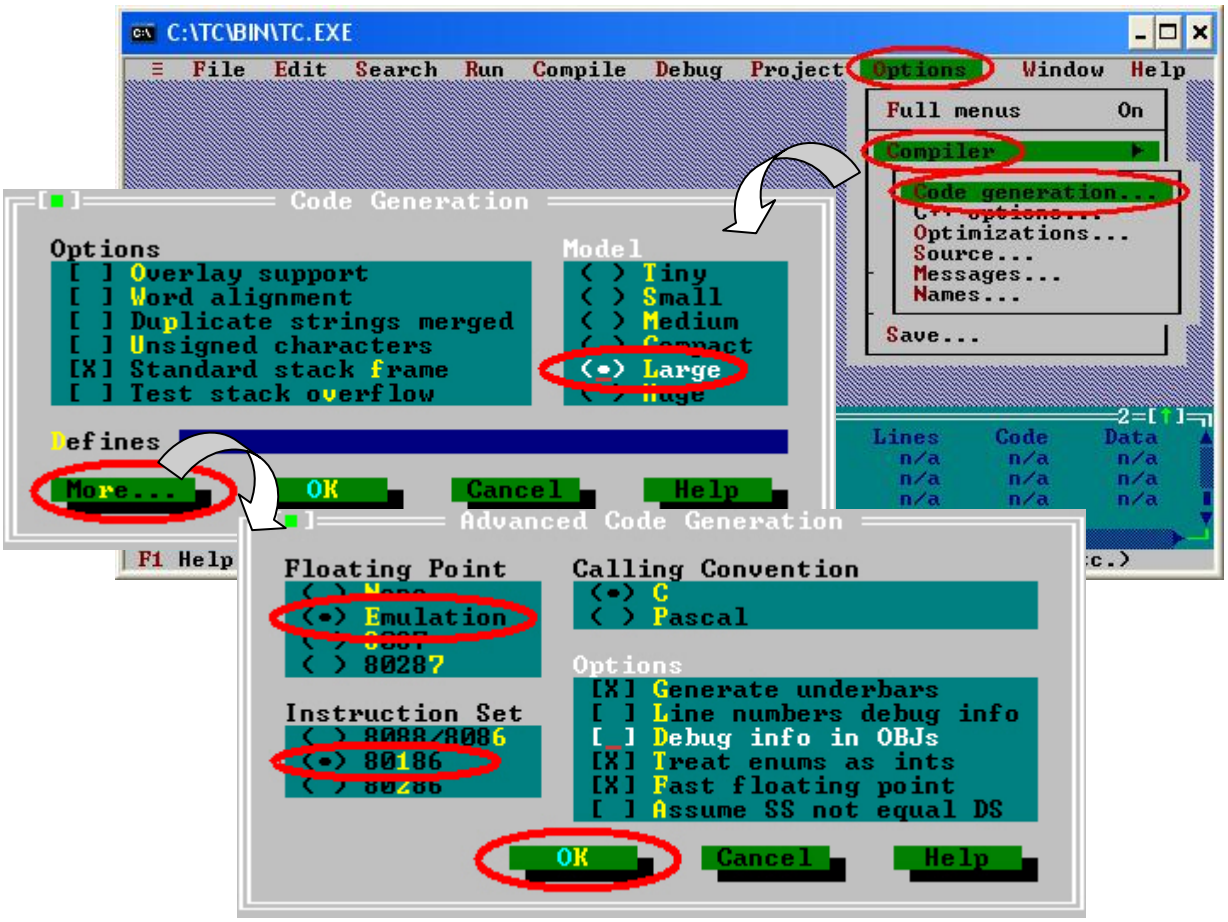
Step7: Click Add function on the bottom of TC++1.01 window. Search all .c file by setting c:\MyDemo*.c in the Name field of popup window. Then, use the Add button to add the users' .c file in to MyDemo project. Then, change the search command from "c:\MyDemo*.c" to "c:\MyDemo*.lib" in the Name field. Add the library files I8120.lib and 8000e.lib into MyDemo project by the same way.



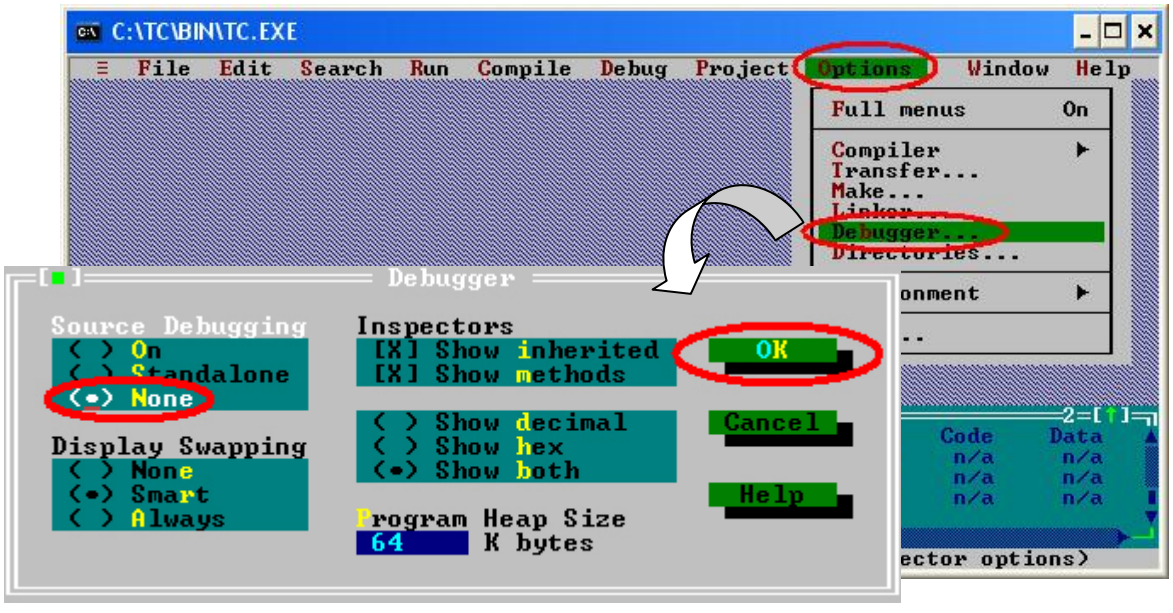
Step8: After finishing the Step7, the TC++1.01 window will look like as follows.



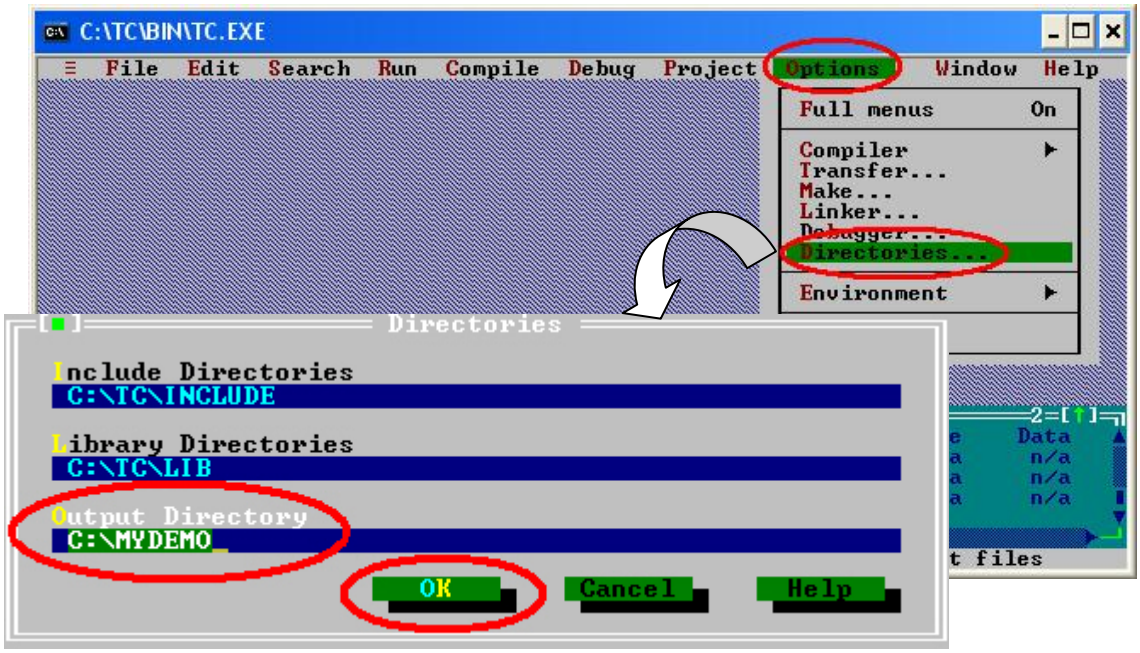
Step9: Click the “Options/Compiler/Code generation…” to set the compiler model to the large mode. Afterwards, click “More…” to set the “Floating point” and “Instruction Set” parameters, the Emulation and 80186 item will be used respectively. Then, click OK to save the configuration.



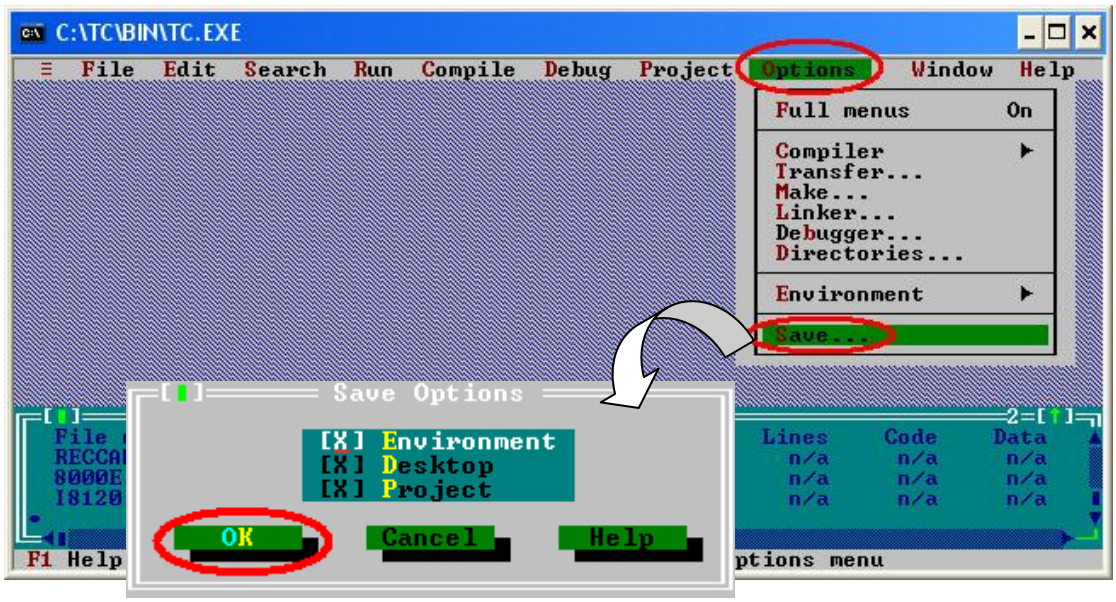
Step10: Click the “Option/Debugger...” to set the “Source Debugging” parameter. Here, select “None” for this parameter setting.



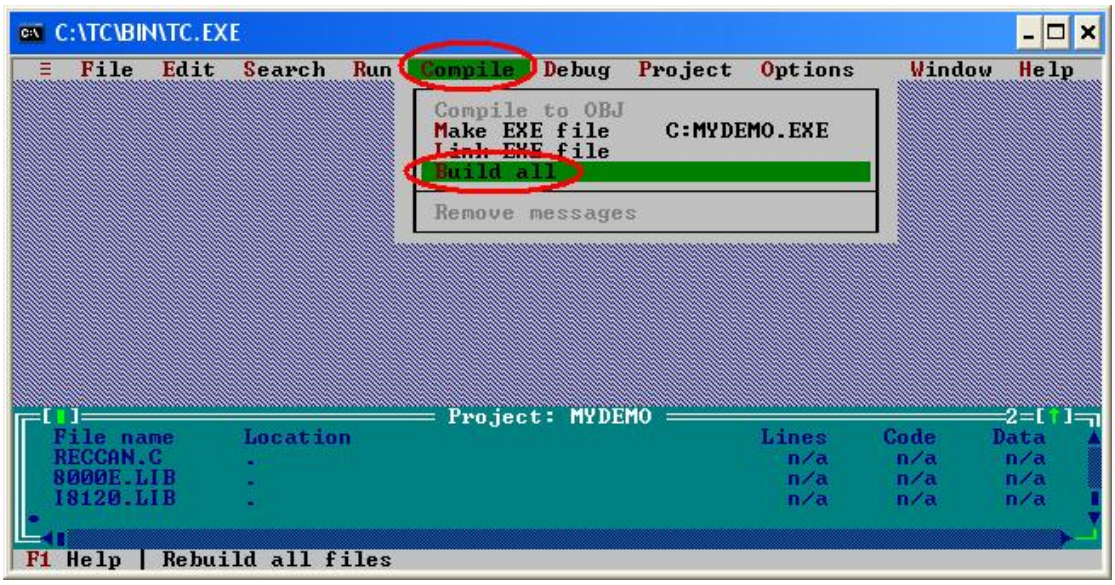
Step11: Click the “Option/Directories...” to set the “Output Directory” parameter.
Here, set the “C:\MyDemo” for the “Output Directory” parameter.



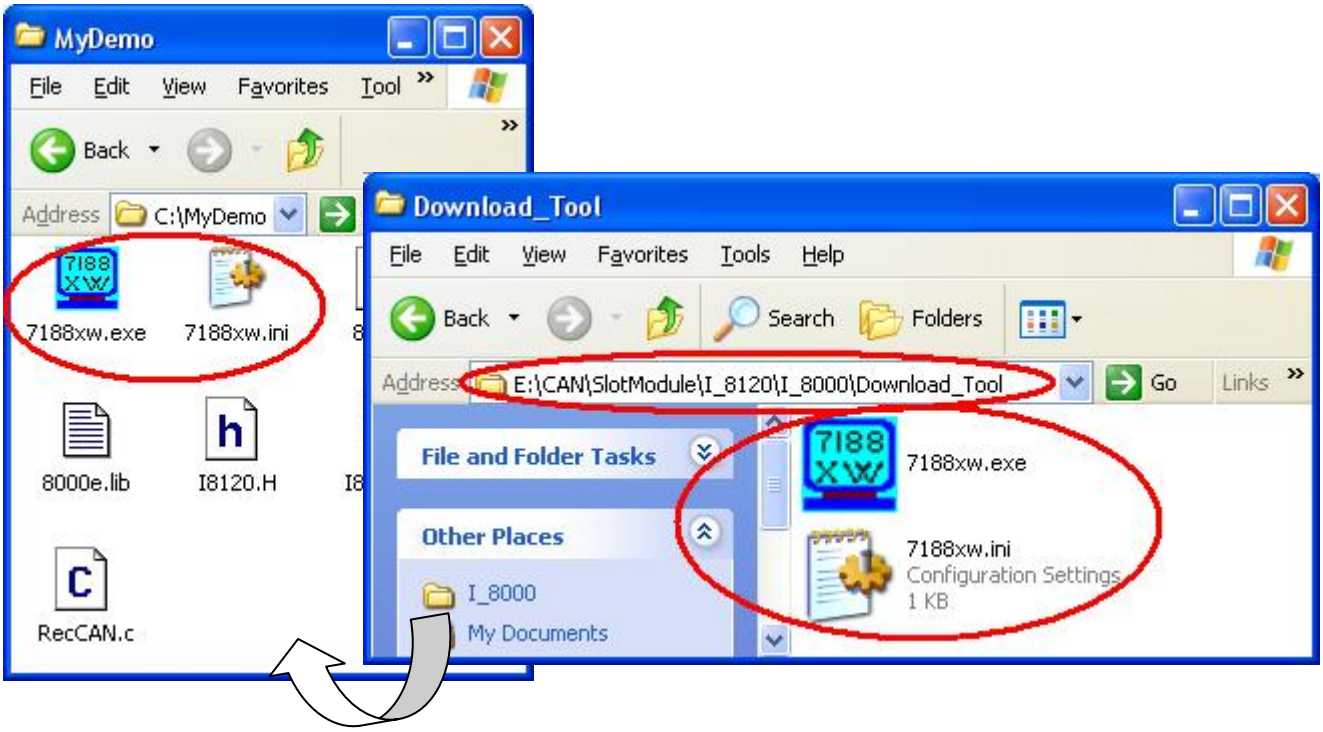
Step12: After finishing the parameters setting, click the “Options/save” to save this project.



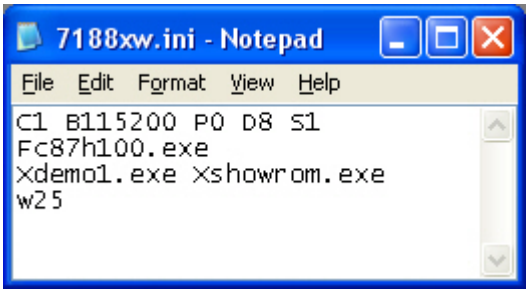
Step13: After finishing the parameters setting, click the “Compile/build all” to produce the execution file. Users can find the execution file in the MyDemo folder. Its name is MyDemo.exe.



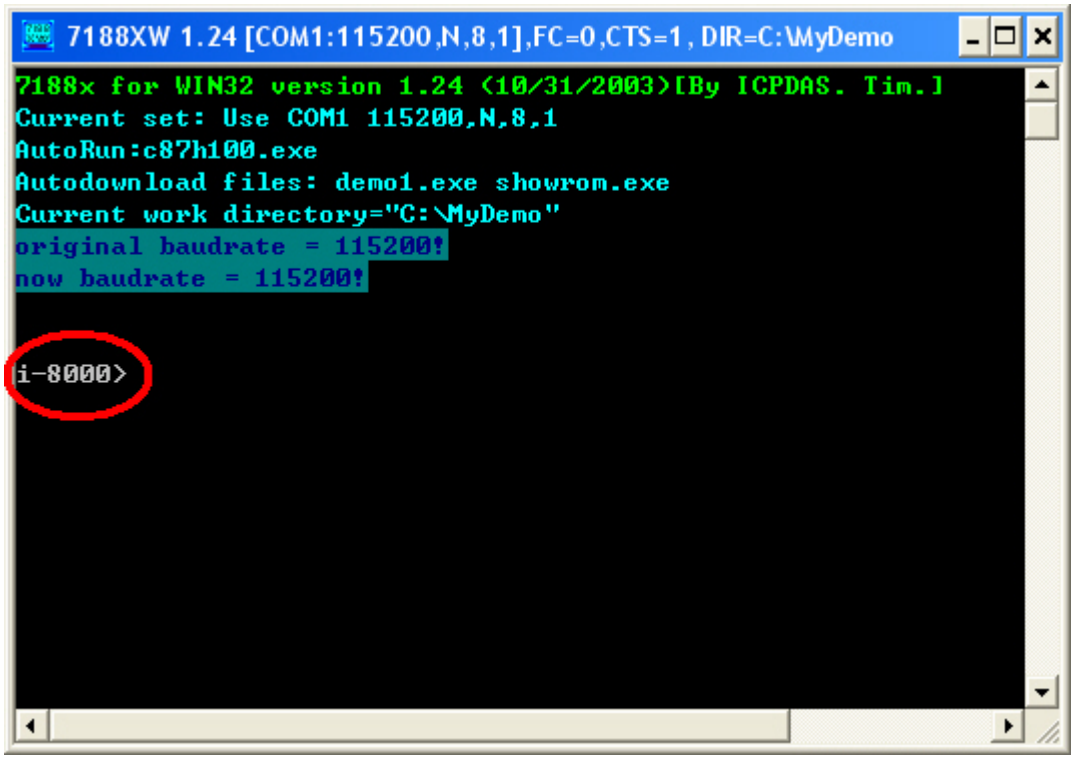
Step14: Copy the file 7188xw.exe and 7188xw.ini files into the MyDemo folder. These two files can be found in CAN product CD. Their path is “CAN\SlotModule\I_8120\I_8000\Download_Tool”.



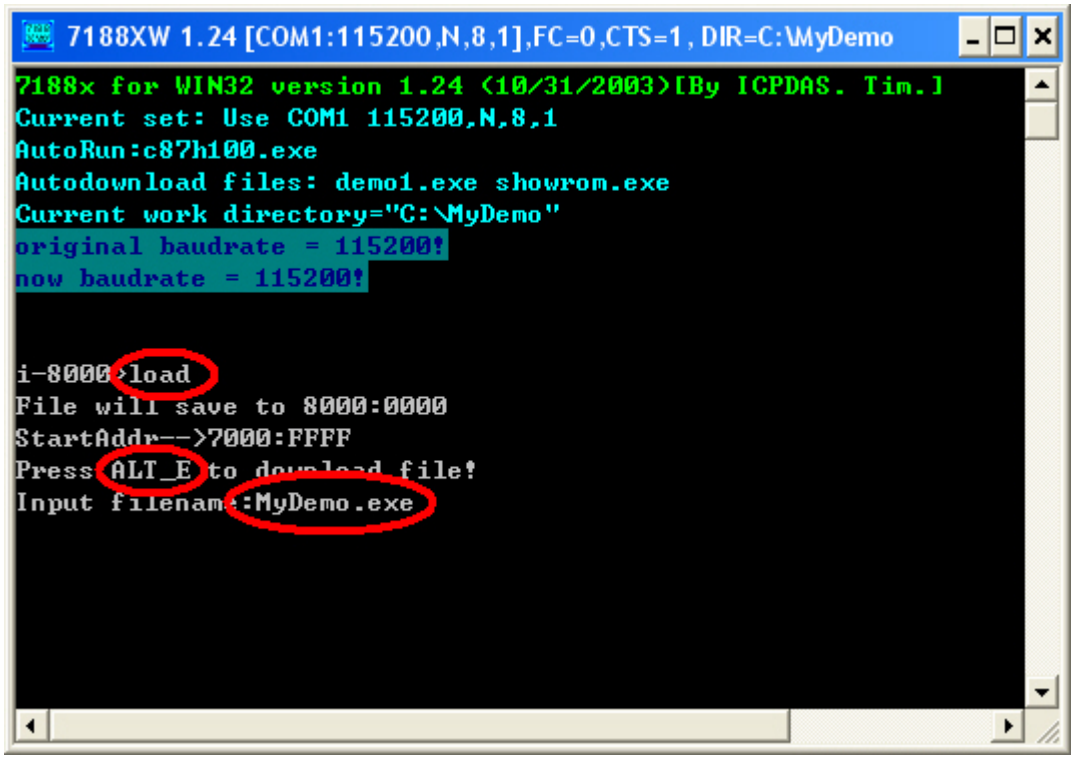
Step15: Use Notepad to modify the first line of 7188xw.ini in MyDemo folder. This part of parameters is used to set the PC RS-232 com port parameters. Words “C1” means the PC COM port number. “B115200” indicates the baud of PC COM port. “P0” is parity setting. “D8” is data bit setting. “S1” is stop bit setting. For example, if users use PC COM1 to connect with the COM1 of I-8000 series MCU for program download, the first line of 7188xw.ini is “C1 B115200 P0 D8 S1”. If users use PC COM2 to connect with the COM1 of I-8000 series MCU, the first line is set to “C2 B115200 P0 D8 S1”.



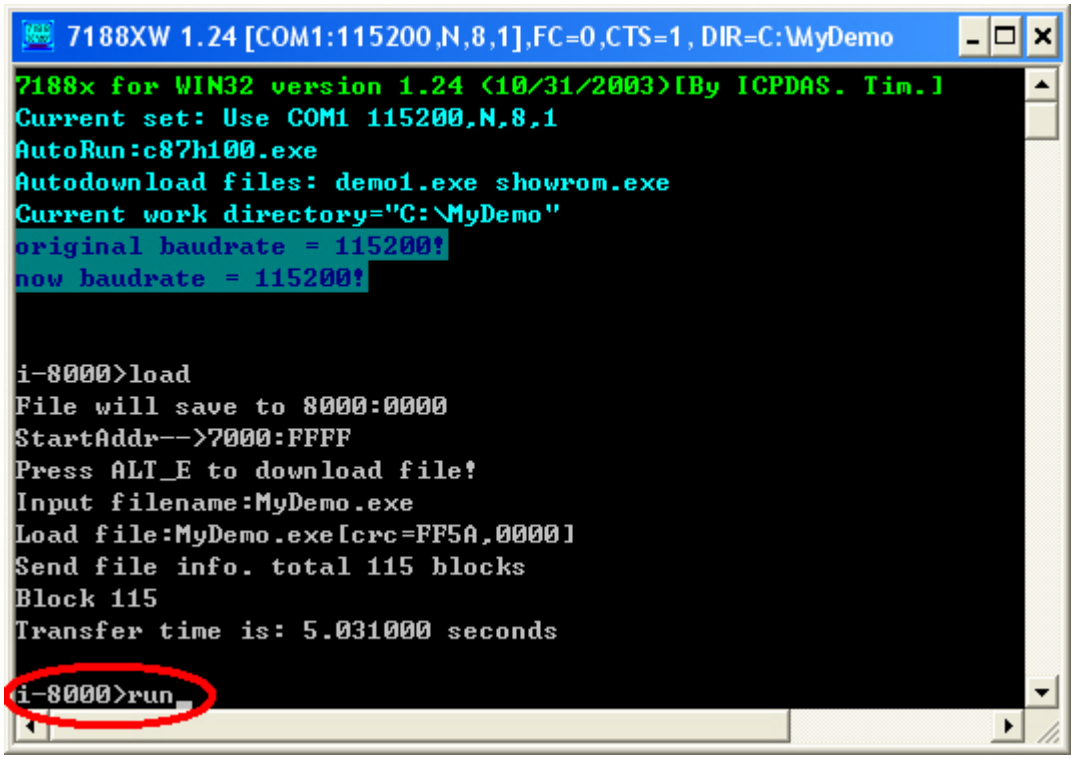
Step16: If the COM1 of I-8811 has connected to the PC COM1, the hint sign,”I-8000>”, will be shown in the 7188xw.exe window after pressing the Enter key in the 7188xw.exe program. (Note: Different type of I-8000 series MCU or different OS version of I-8000 series MCU will have different hint sign.)



Step17: Key the command, “load” in the 7188xw.exe program. Follow the hint command to press “Alt+E” and input the file name “MyDemo.exe“ to download the execution file. Then, press Enter to continue.



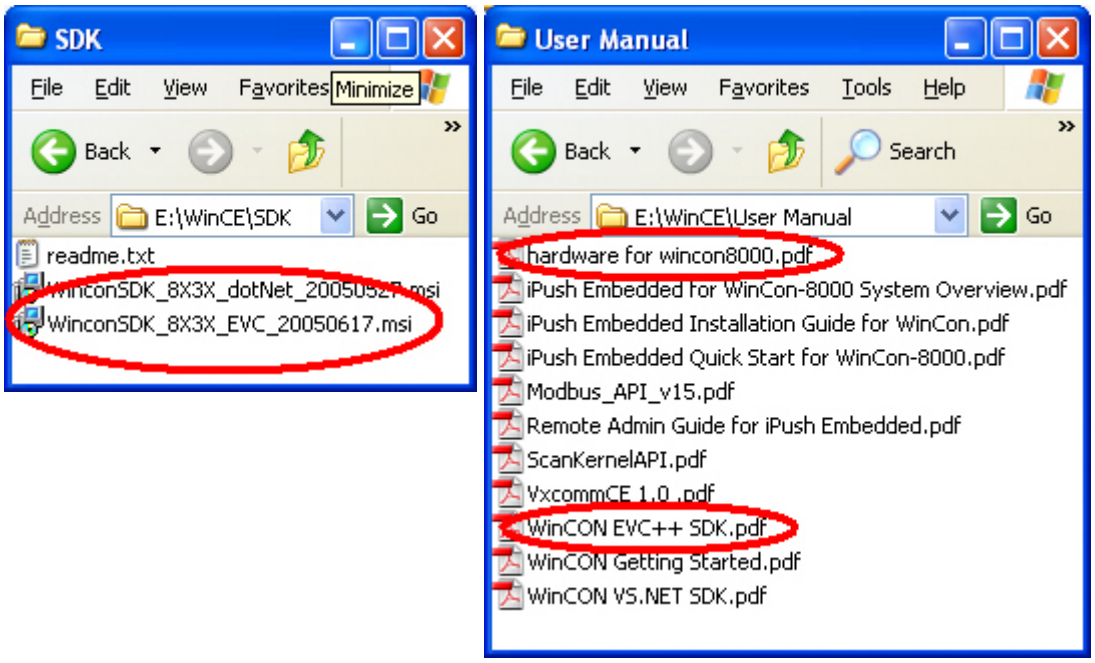
Step18: After finishing the download procedure, key in the command “run” to implement the execution file ”MyDemo.exe”.



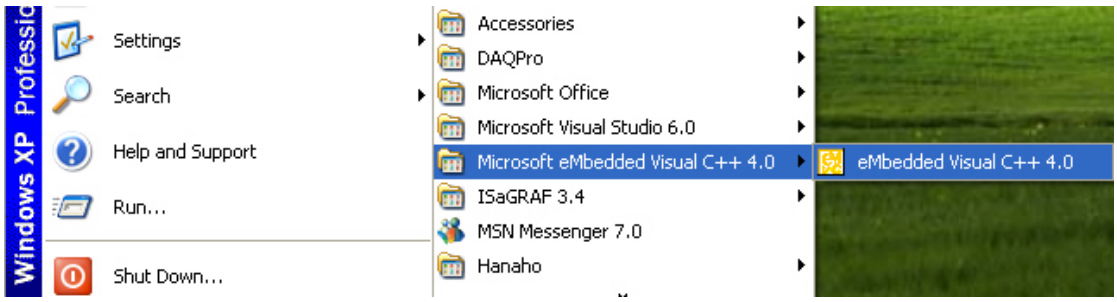
4.2 EVC++ Demo For WinCon Series MCU

Step1: Download EVC++ 4.0 from Microsoft website. Then, install EVC++4.0 in your PC. (Note: About the hardware and OS limitation of EVC++, please refer to the Microsoft website)

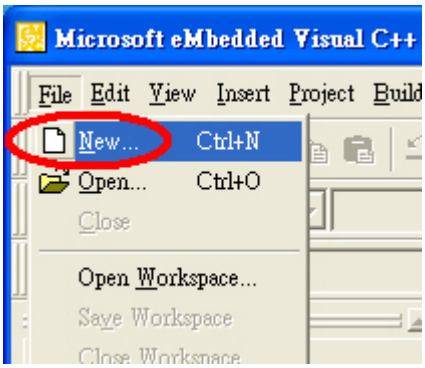
Step2: Double click WinConSDK_8X3X_EVC_20050617.msi file to install the WinConSDK in your PC (users may install the newest WinCon SDK. In this case, the file name may be a little difference from the name shown here.). Users can find this file in the path “WinCE\SDK\” in the WinCon product CD. For more information about WinCon hardware and how to use WinConSDK, please refer to the user manuals in the path “WinCE\User Manual” in WinCon product CD.



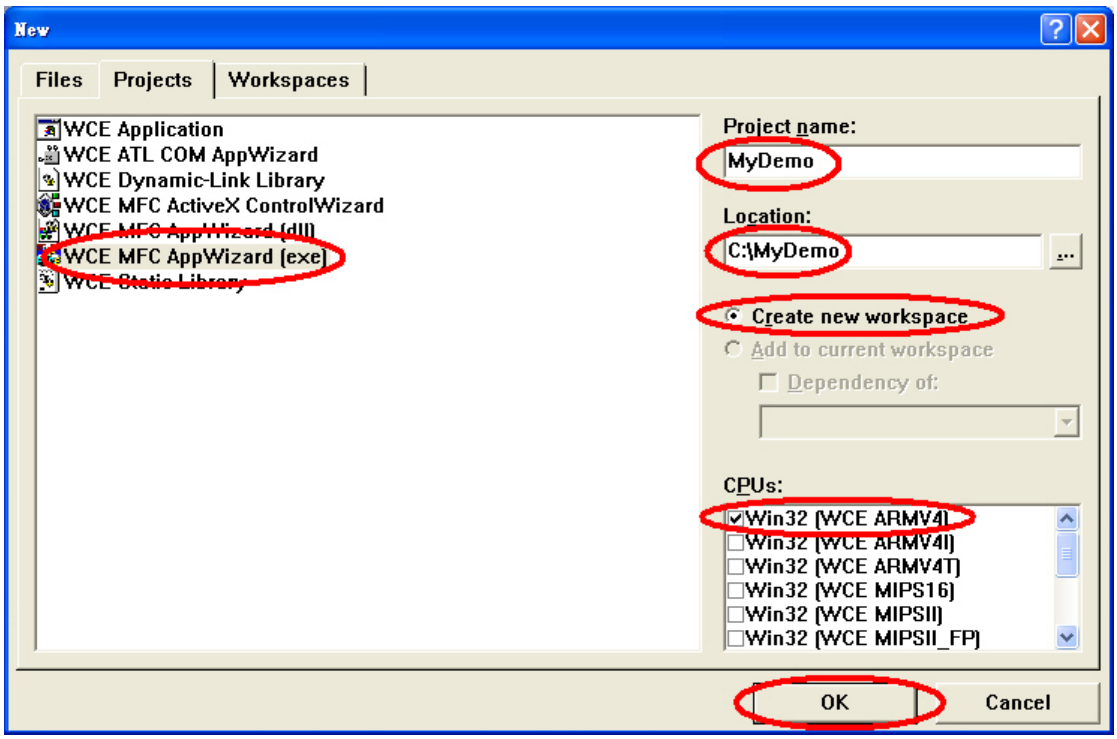
Step3: Execute the EVC++.



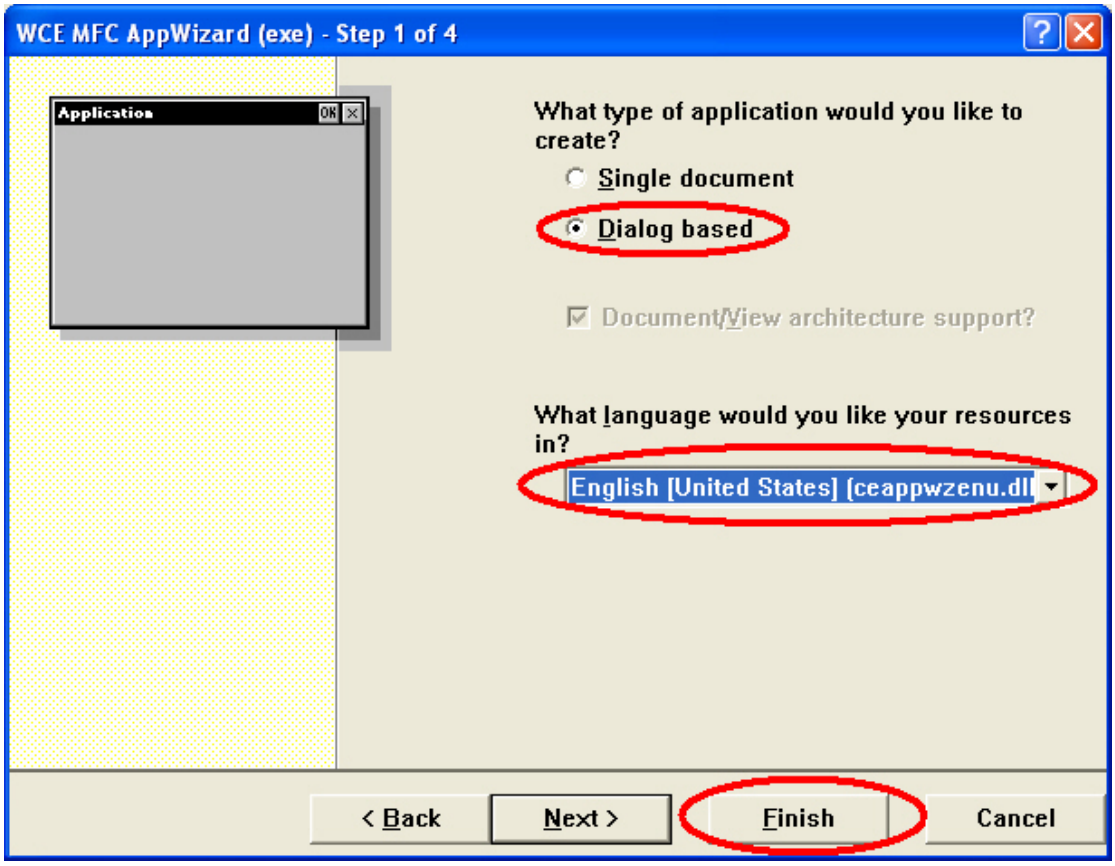
Step4: Click “File\New…” to create a new project.



Step5: Select “WCE MFC AppWizard (exe)” for this project template. The project name is “MyDemo”. The location of this project is “C:\MyDemo”. The CPU type in the CPUs field is set to “Win32 (WCE ARMV4)” because the WinCon series MCU use the ARMV4 series CPU. Then, click OK to the next.



Step6: Select “Dialog based” item for this demo. Choose the language which you want to see in your resources file. Here, “English (United States)” item is used. Click “Finish” button to finish the project creation.

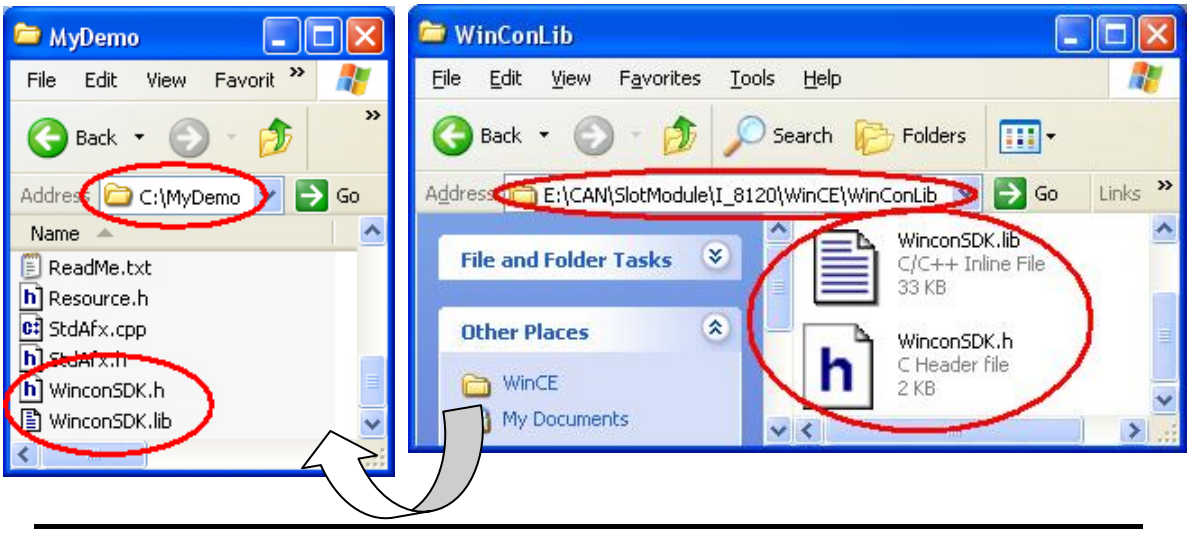


Step7: Copy the WinConSDK library files “WinconSDK.lib” to the folder “MyDemo” in C disk. Users can find the WinConSDK library files in the following path in PC.

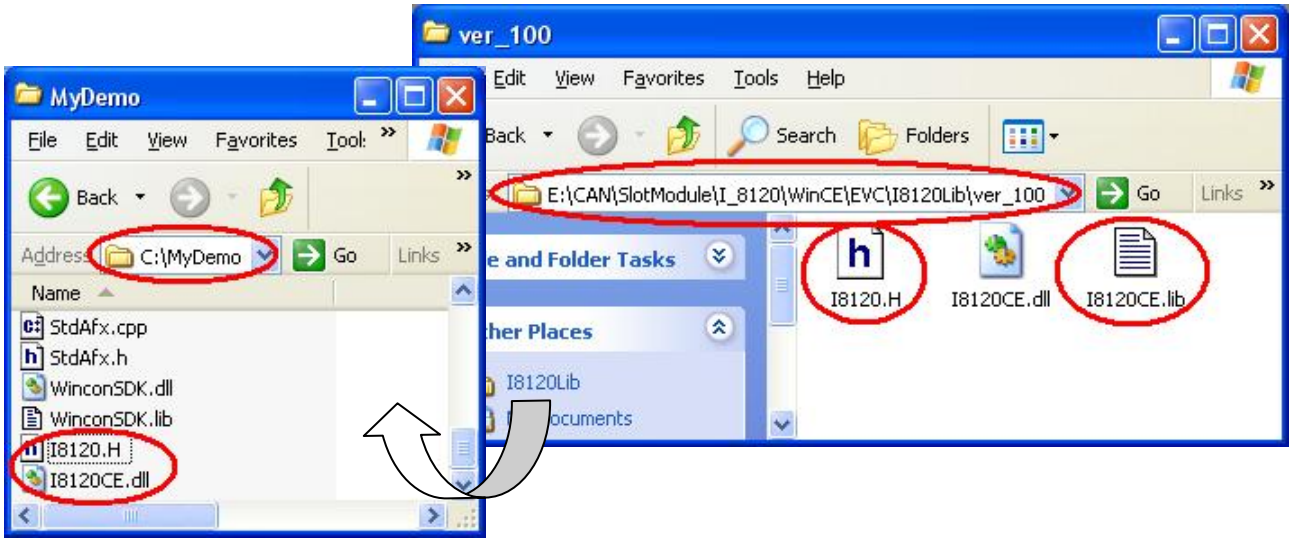
“C:\Program Files\Windows CE Tools\wce410\SA_IA\Lib\ARMV4”

Users can also find these files in the following path in CAN product CD.

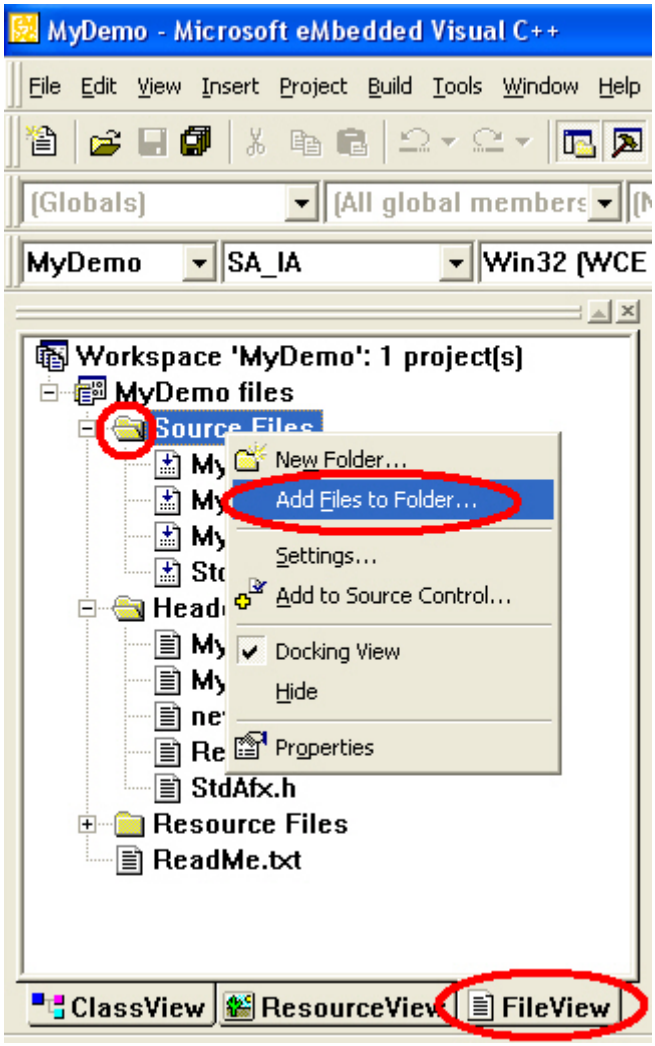
“CAN\SlotModule\I_8120\WinCE\WinConLib”



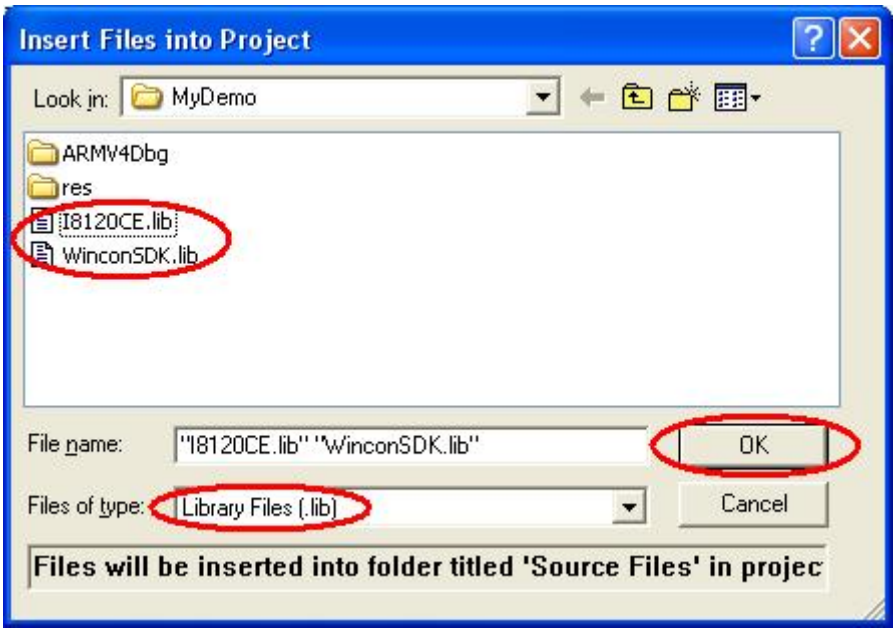
Step8: Copy the I-8120 library files “I8120.h” and “I8120CE.lib” into the MyDemo folder in disk C.



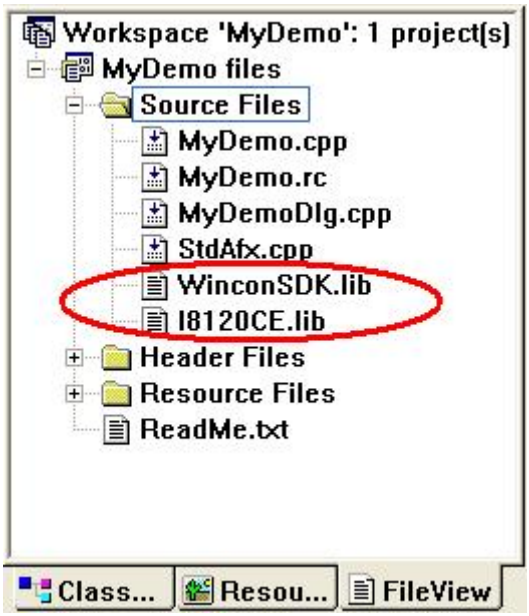
Step9: Select “File View” tag, and expand the tree view. Right click the “Source Files” folder icon and click Add Files to Folder…” item.



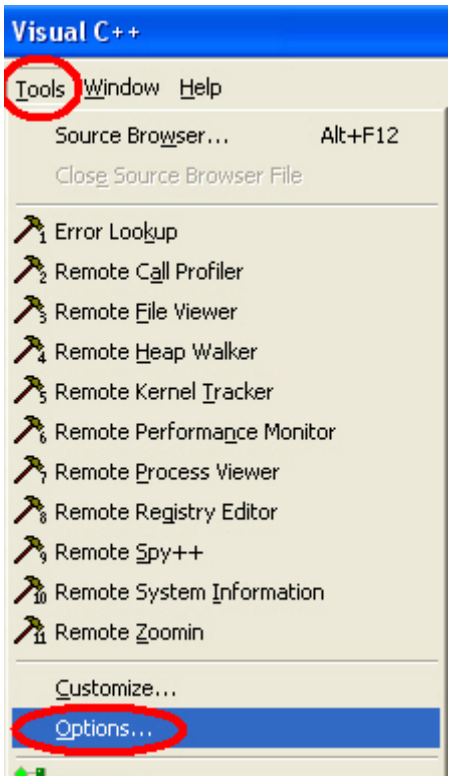
Step10: In the popup dialog, select “Library Files (.lib)” in the “Files of type” filed. Hold the Ctrl key and left click the library files (I8120CE.lib and WinconSDK.lib) to select these files. Then, click OK button to add these library files into MyDemo project.



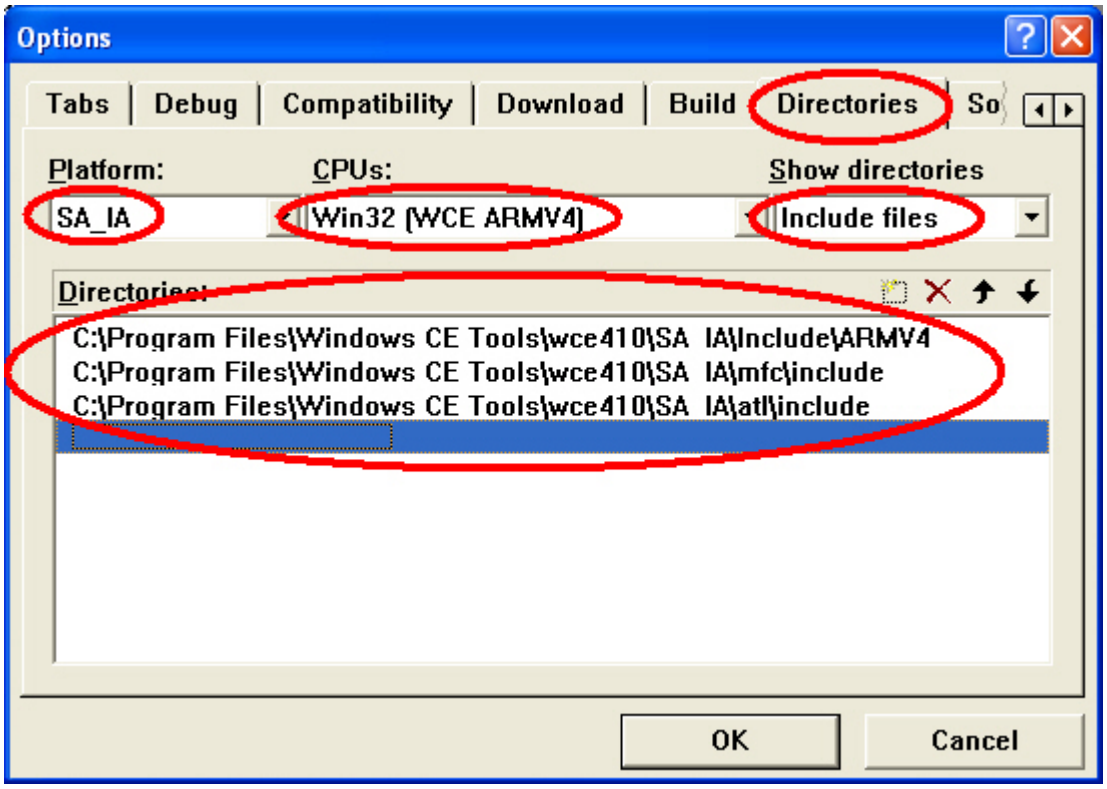
Step11: After finishing the Step 9, the tree view of “File View” is shown below.



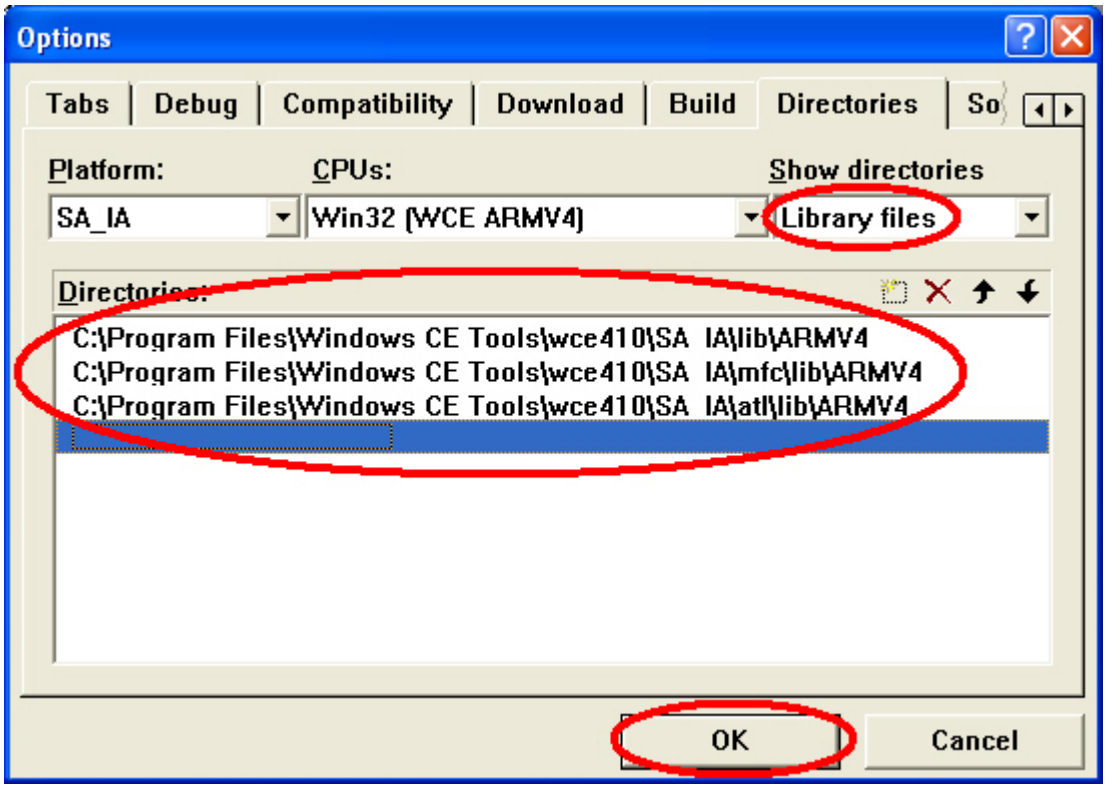
Step12: Click the “Tools\Options…” to set the “include” and “library” directory.



Step13: Click “Directories” tab, and select “Include files” in the “Show directories” field. Set the directories as follows.



Step14: Select the “Library files” in the “Show directories” field, and set the library directory as follows. After finishing the Step13 and Step14, click OK button to save the parameter settings.

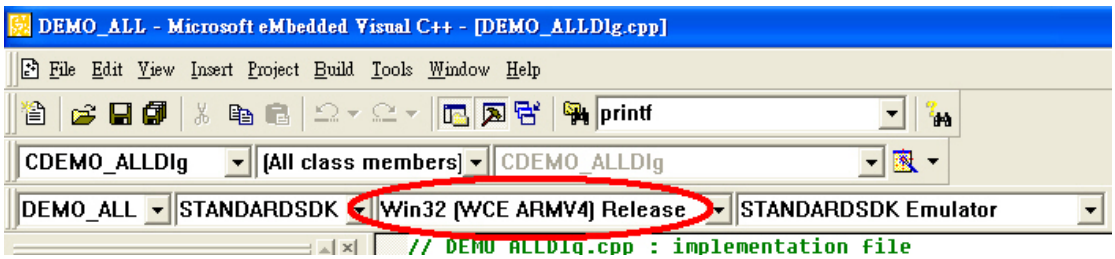


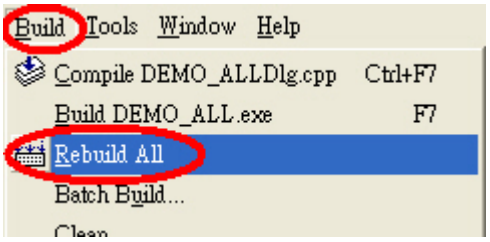
Step15: Program users’ application and design the dialog screen in EVC++ environment. When program the .cpp program, user need to include the “WinconSDK.h” and “I8120.h”. The “#include” syntaxes are shown below.

#include “WinconSDK.h”
#include “I8120.h”

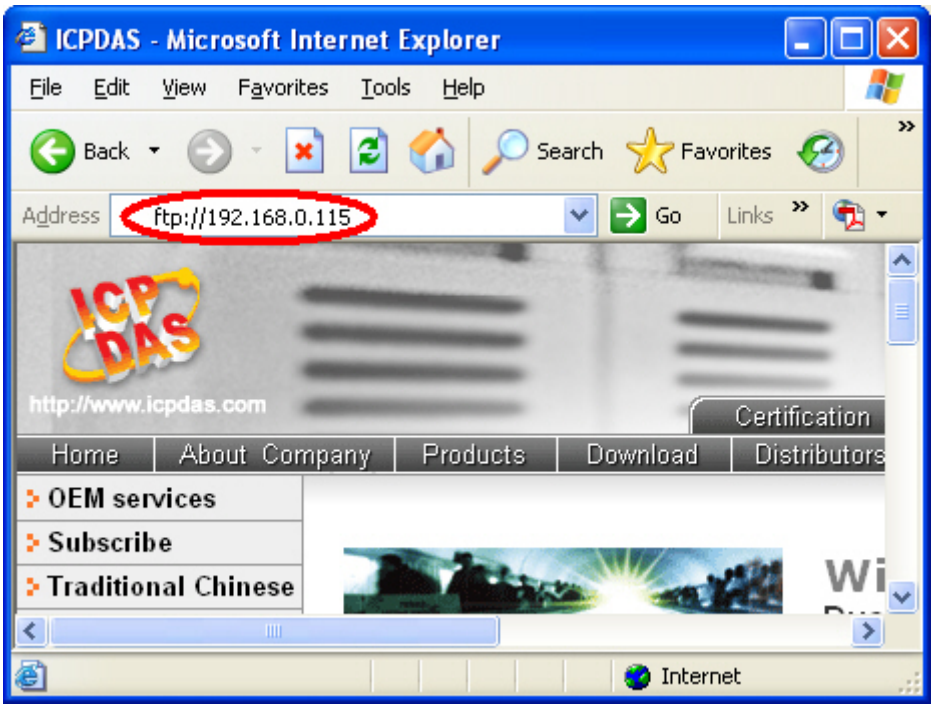
```
// MyDemoDlg.cpp : implementation file
//

#include "stdafx.h"
#include "MyDemo.h"
#include "MyDemoDlg.h"
#include "WinconSDK.h"
#include "I8120.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

Step16: When finishing the program, select the project configuration to “Win32 (WCE ARMV4) Release” and click “Build\Rebuild All” to build an execution file.



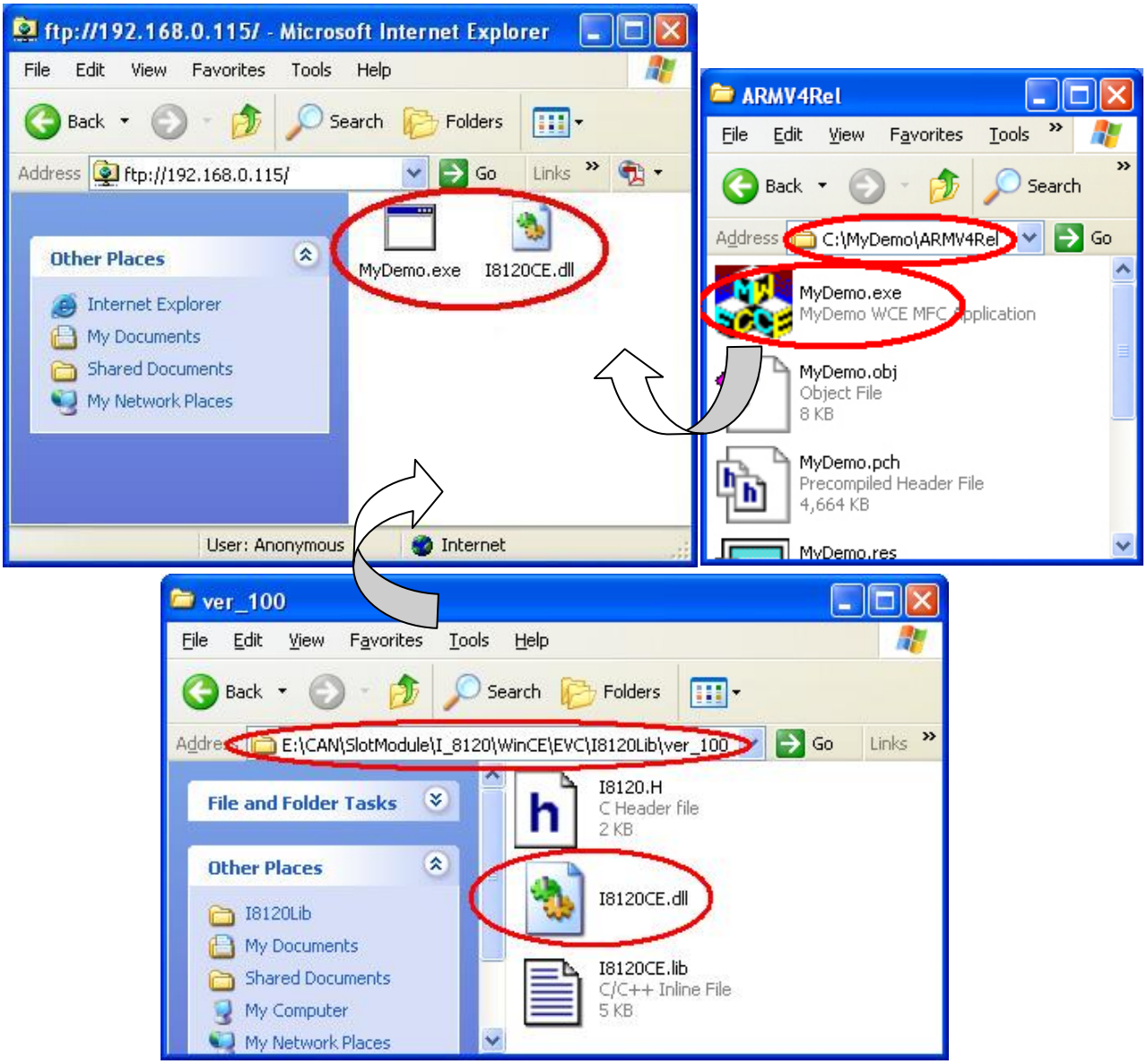


Step17: Use ftp method to copy the execution file and necessary .dll file into WinCon series MCU. First, Open the IE (internet explorer) and key the “ftp://192.168.0.115” and press Enter key in the Address filed. The 192.168.0.115 is your WinCon’s IP address.



Step18: After connecting to the WinCon series MCU, drag and drop the execution MyDemo.exe and I8120.dll file into the IE window. Users can find the MyDemo.exe in the “C\MyDemo\ARMV4Rel” folder. The I8120.dll is in CAN product CD. Its path is as follows.

CAN\SlotModule\I_8120\WinCE\EVC\I8120Lib\ver_100



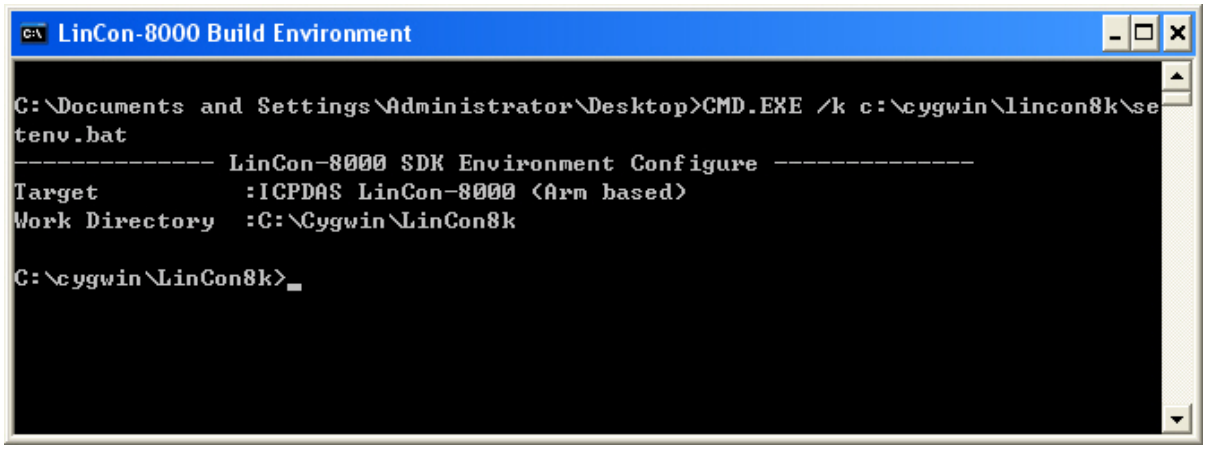
Step19: Then, users can find these two files in the WinCon default ftp folder “\Temp”, and execute the execution file MyDemo.exe by double click the MyDemo.exe icon in the WinCE platform of WinCon series MCU. If users want to change the default ftp folder, the WinCon Utility is needed. For more detail information, please refer to the “WinCON Getting Started.pdf” in the following path in WinCon product CD.

\WinCE\User Manual\WinCon Getting Started.pdf

4.3 GCC Demo For LinCon Series MCU

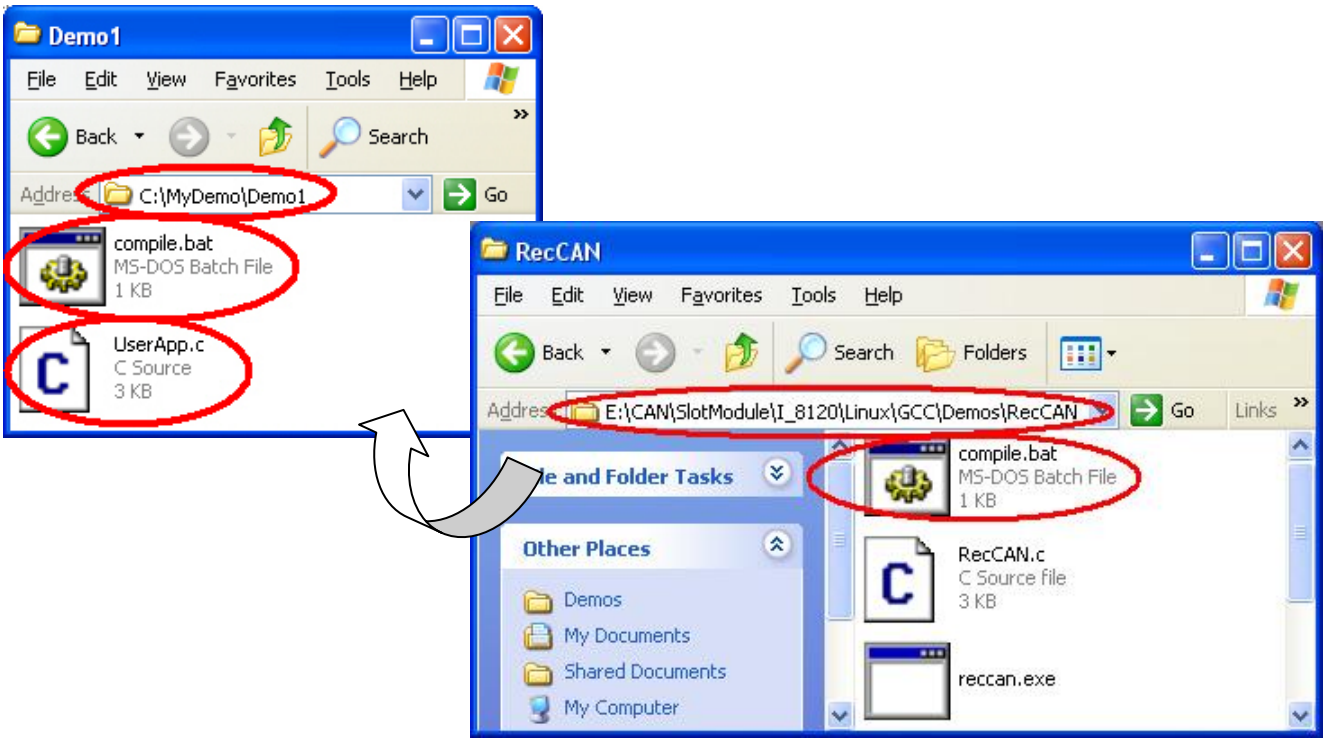
Step1 : Download the GCC from web site and install it in the disk C of your PC. Users can check the chapter 2 of the LinCon manual, LinCon_Manual.pdf, to understand the installation procedure.

Step2: After finishing the installation, double click the icon, LinCon-8000 Build Environment, on the desktop to enter the LinCon-8000 SDK environment.

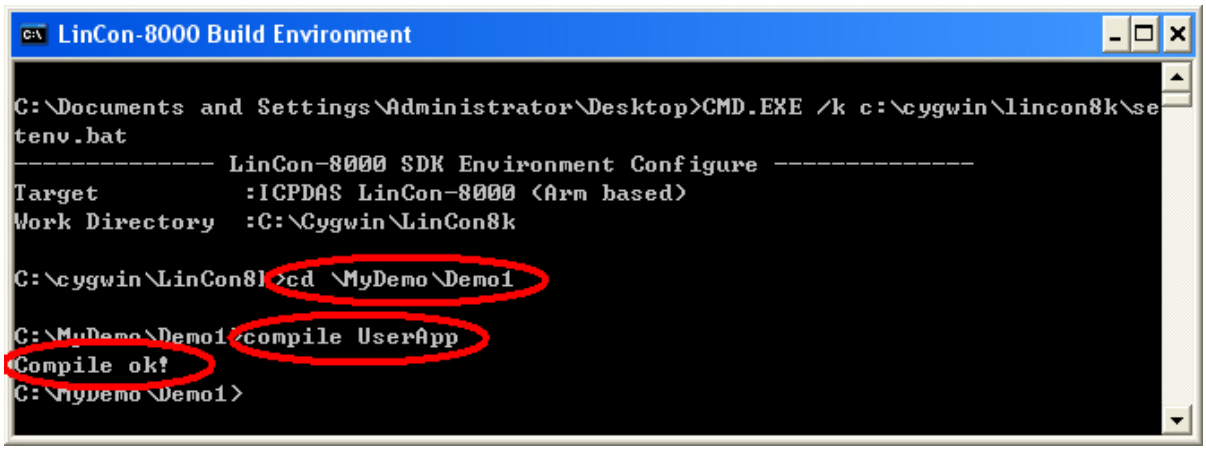


Step3: Create a folder named as MyDemo in your disk C. Then, copy the I8120Lib folder to the MyDemo folder and create a new folder named as Demo1 in the MyDemo folder. Users can find the 8120lib folder in following path of CAN CD: CAN\SlotModule\I_8120\Linux\GCC

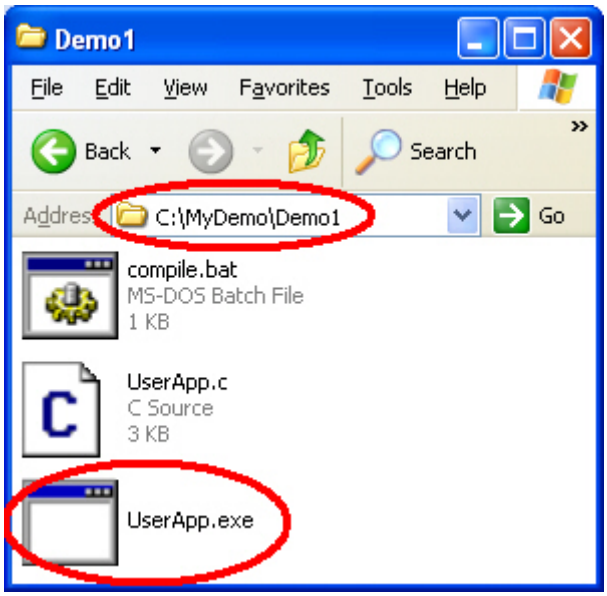
Step4: Put users program into Demo1 folder. Here, the UserApp.c is used. Copy the compile.bat file into the Demo1 folder. Users can find the compile.bat file in the each folder of LinCon I-8120 demos.



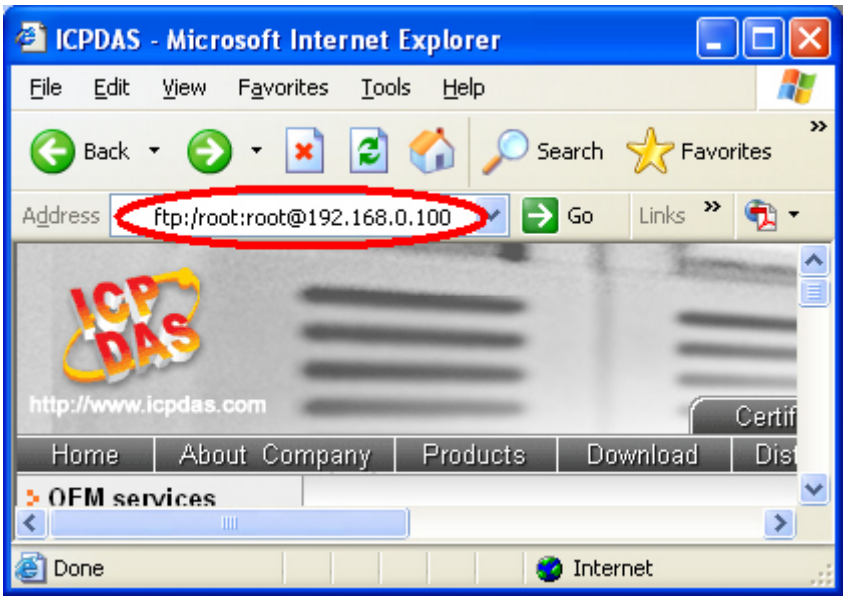
Step5: In order to build an execution file, key the “cd \MyDemo\Demo1” command in the LinCon-8000 Environment window to enter the folder path “C:\MyDemo\Demo1”. Then, key the “compile UserApp” under the hint “C:\MyDemo\Demo1>”. The letters “UserApp” are user’s program name without auxiliary file name. If there is no error and warning during the compile procedure, the “Compile OK!” will be shown in the LinCon-8000 Environment window.



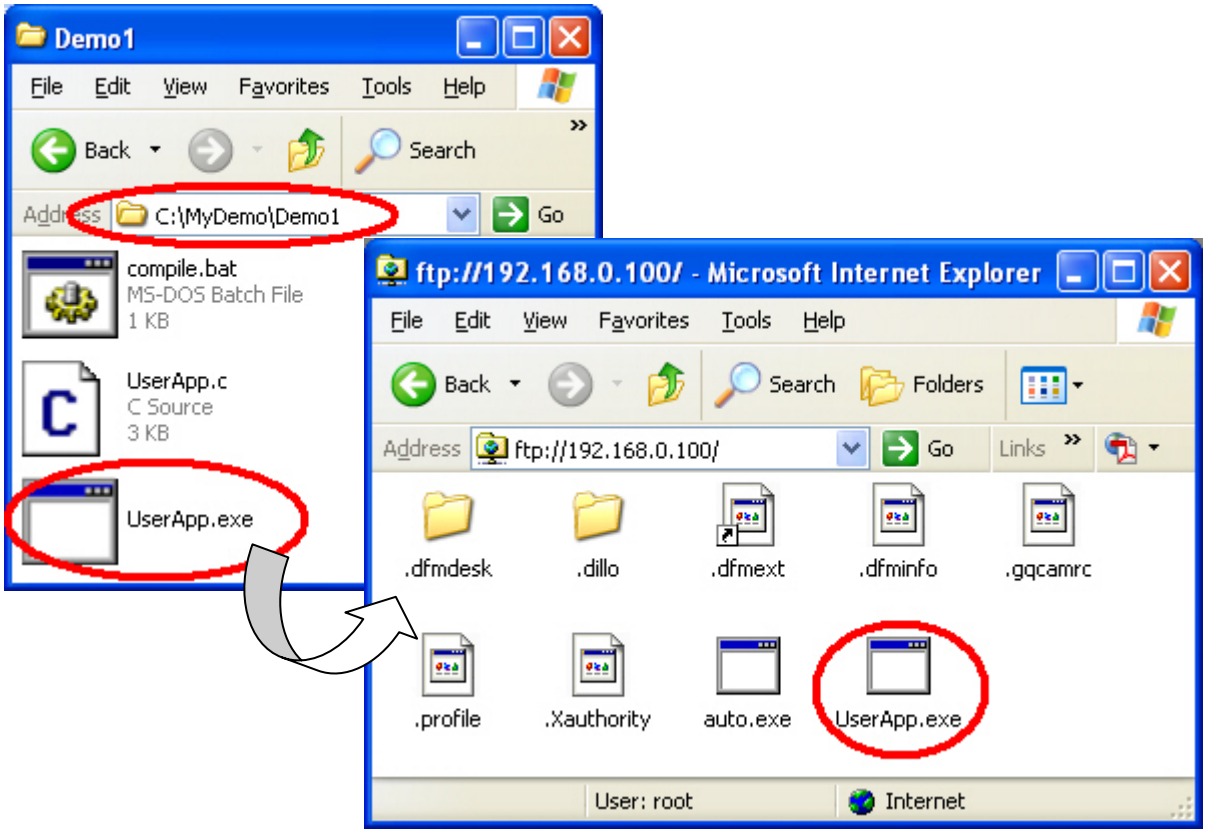
Step6: After finishing the compiling, the execution file, UserApp.exe, is produced in the Demo1 folder.



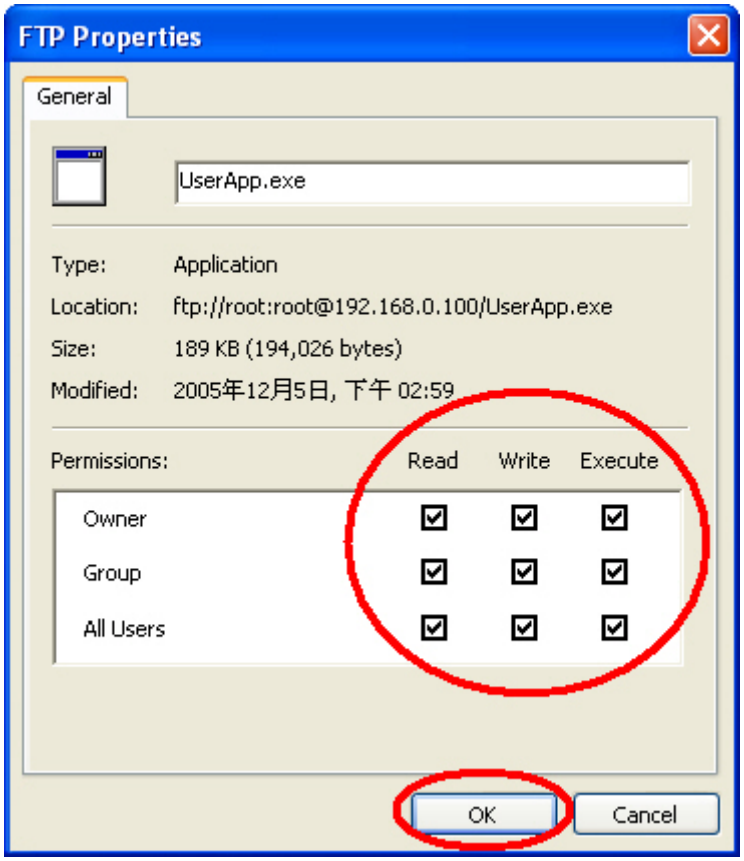
Step7: Open the IE window and input the URL “ftp:/root:root@192.168.0.100” to download the file into LinCon by using FTP. The first root is the user name of LinCon FTP server, and the second root is the password of LinCon FTP server. The letters “192.168.0.100” are the LinCon IP address.



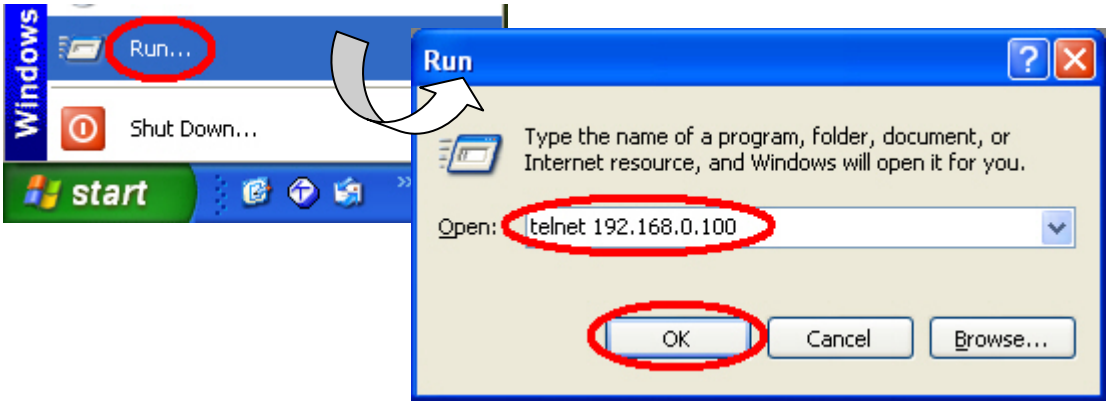
Step8: Copy the file into the IE window of LinCon FTP site.



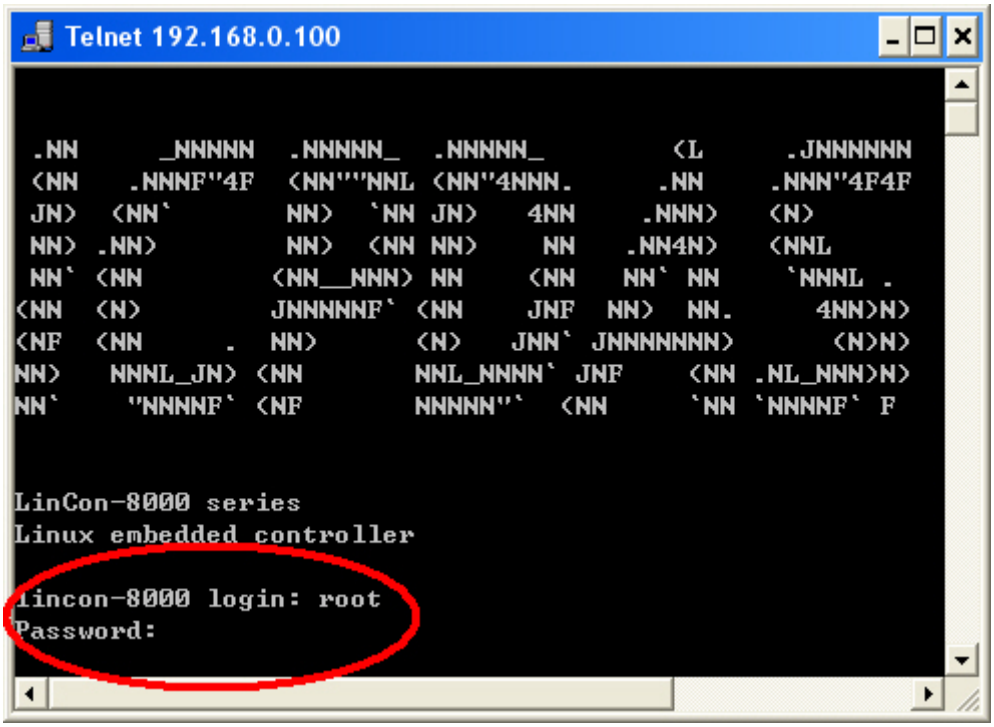
Step9: Right click the UserApp.exe icon and set the file properties as follows.



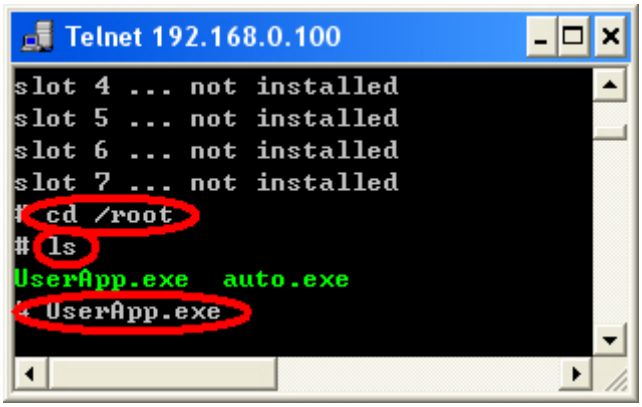
Step10: Key the “telnet 192.168.0.100” in the path “Start\Run…”. The “192.168.0.100” is LinCon IP address.



Step11: Enter the user name “root” and password “root” in the Telent window.



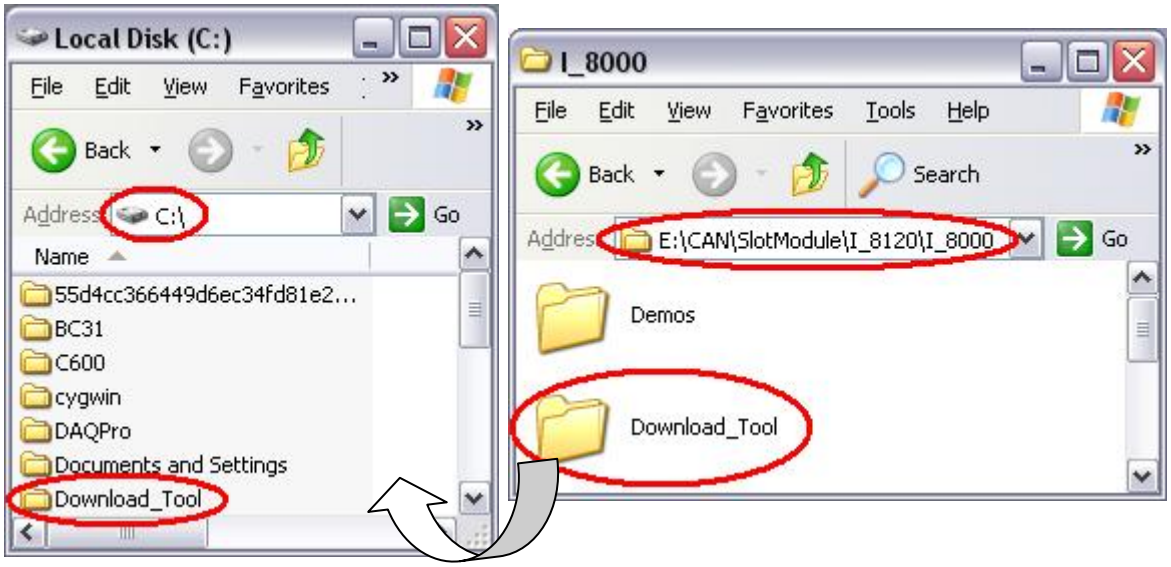
Step11: Key “cd /root” to enter the root folder, and use “ls” to check all the files in the root folder. Then, key “UserApp.exe” to execute the .exe file.



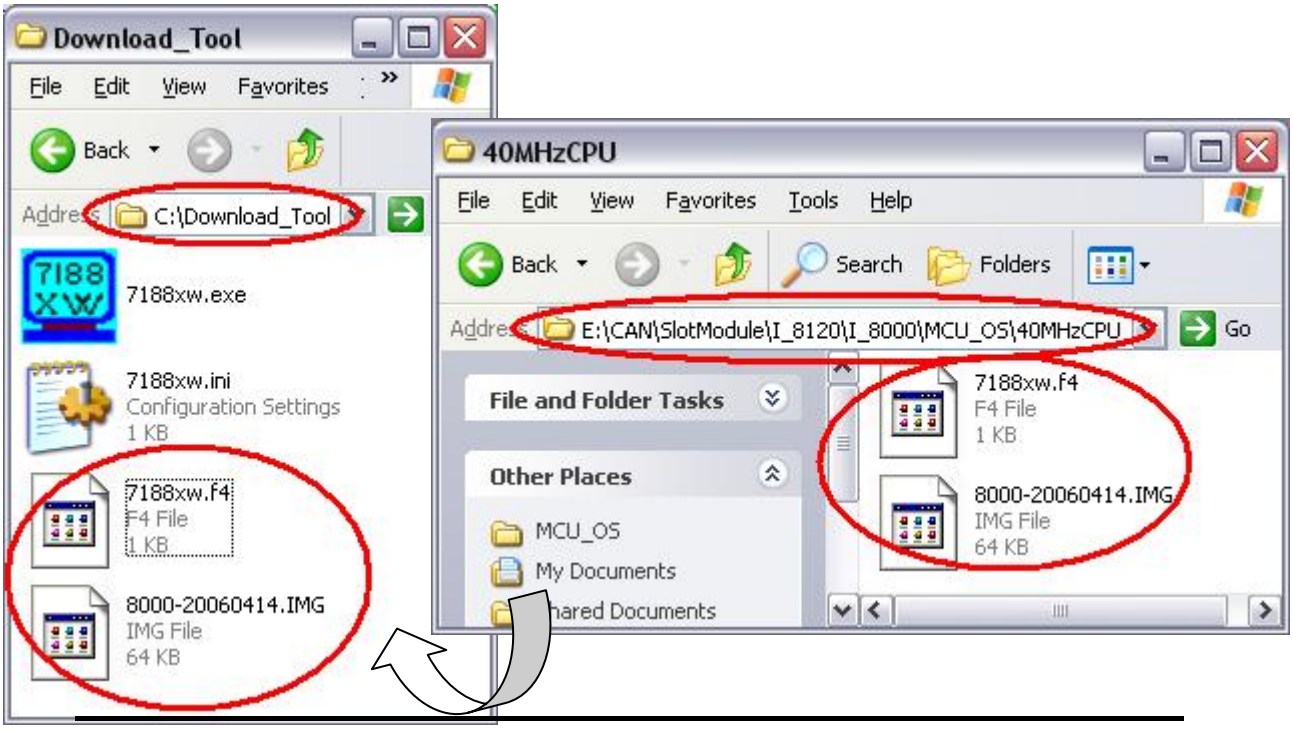
4.4 OS Update for I-8000 series MCU

If necessary, users can update the OS image of your I-8000 series MCU. There are step-by-step update methods shown below.

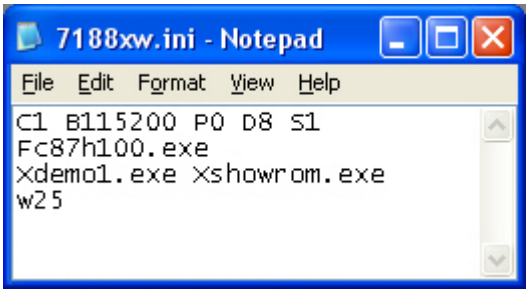
Step1: Copy the “Download_Tool” folder from CAN CD to disk C. Users can find this folder in the path “**CAN\SlotModule\I_8120\I_8000**“.



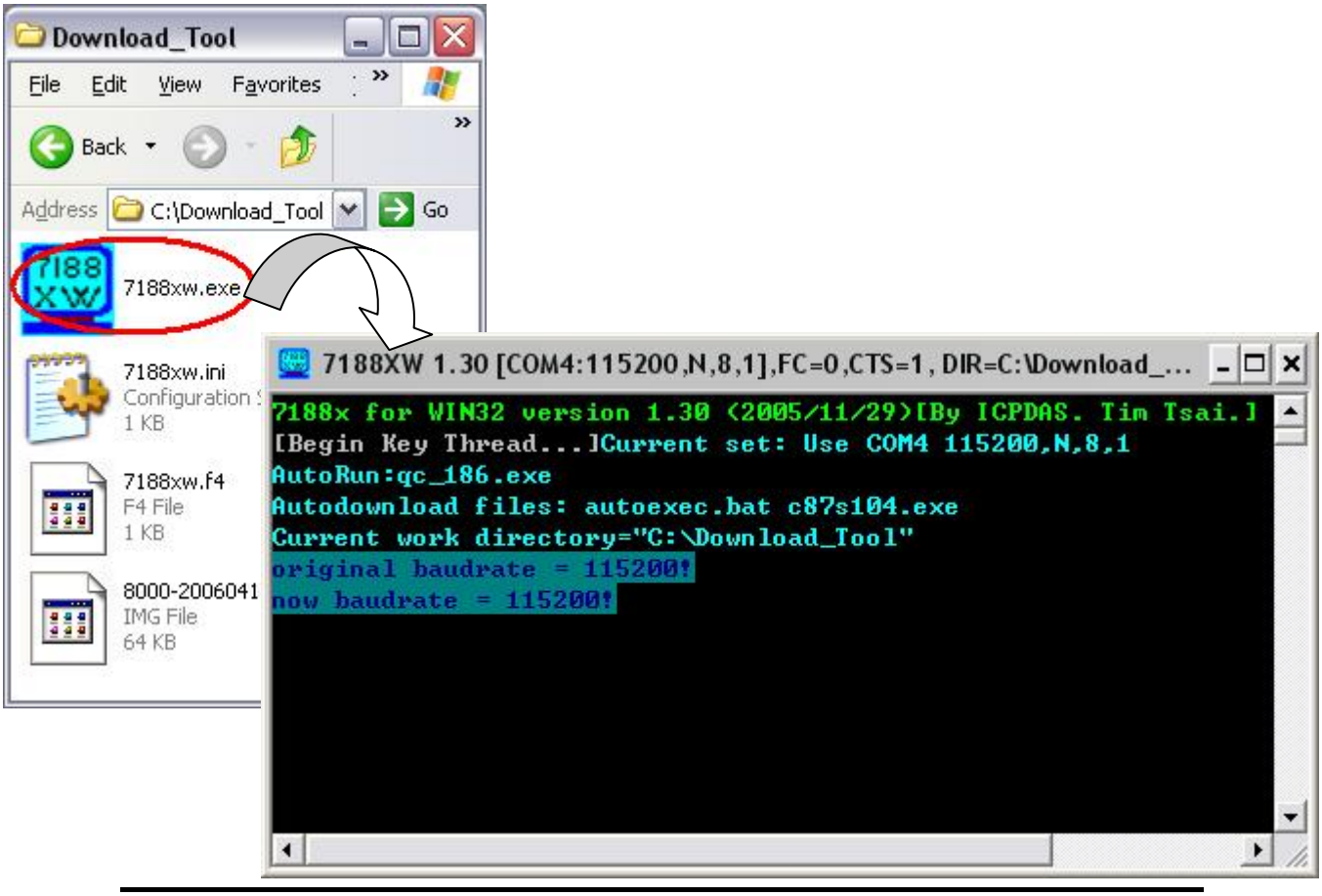
Step2: Copy all the OS image files, 7188xw.f4 and 8000-20060414.img, into “C:\Download_Tool\”. If users use I-8000 series MCU with 40MHz CPU, users need to use the OS image files of “40MHzCPU” folder. If the CPU of I-8000 series MCU is 80MHz, use the files of “80MHzCPU” folder. Here, use I-8000 series MCU with 40MHz for this demo.



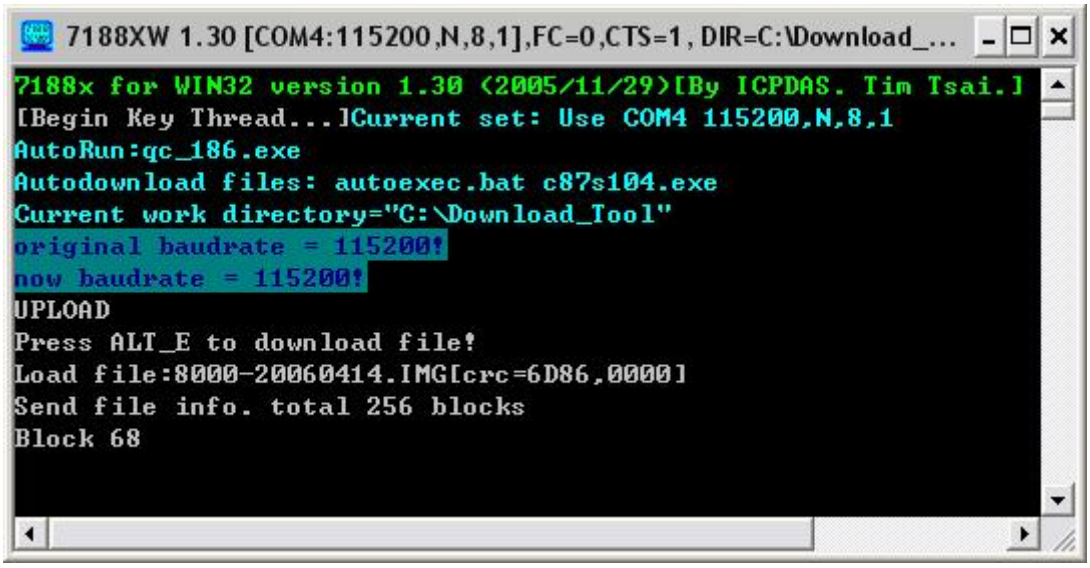
Step3: Use Notepad to modify the first line of 7188xw.ini in “Download_Tool folder. This part of parameters is used to set the PC RS-232 com port parameters. Words “C1” means the PC COM port number. “B115200” indicates the baud of PC COM port. “P0” is parity setting. “D8” is data bit setting. “S1” is stop bit setting. For example, if users use PC COM1 to connect with the COM1 of I-8000 series MCU for program download, the first line of 7188xw.ini is “C1 B115200 P0 D8 S1”. If users use PC COM2 to connect with the COM1 of I-8000 series MCU, the first line is set to “C2 B115200 P0 D8 S1”.



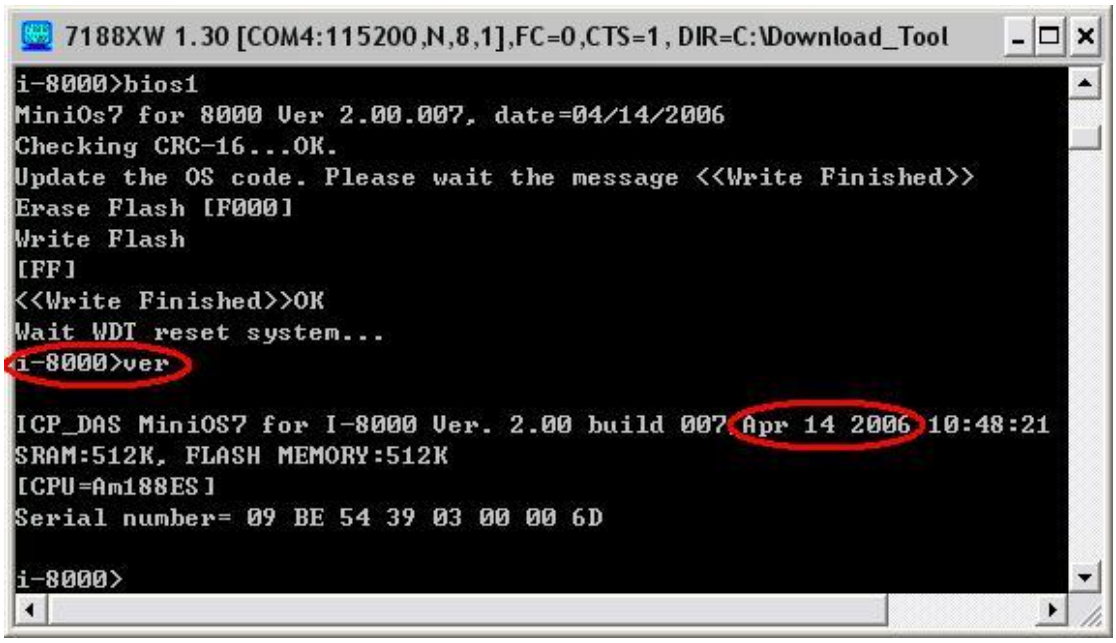
Step4: Power off the I-8000 series MCU, wire the pin Init* and Init*COM of the I-8000 series MCU, and power on the I-8000 series MCU. Then, double click the 7188xw icon to execute the 7188xw.exe.



Step5: Press key “F4” of PC’s keyboard to update the OS image. Then, the 7188xw.exe will start to download the newer OS image and update it automatically.



Step6: After finishing the OS update, use command “ver” to check the version of OS image of I-8000 series MCU.



5 Troubleshooting

When users call the I8120 library functions, the error code may return if some occurs. This section will give some basic diagnostic methods for reference.

Error code	Troubleshooting
CAN8K_TIMEOUT	1. Check if any huge current or huge voltage bypasses the I-8120 module. 2. Reset the I-8120 module and try it again.
CAN8K_FIFO_EMPTY	1. No CAN message is received. 2. CAN channel is not connected. 3. CAN baud rate is different to the CAN network. 4. CAN network terminal resister is not equipped properly. 5. CAN network span distance is too long.
CAN8K_FIFO_FULL	1. The reception FIFO is overflow. 2. The transmission FIFO is overflow. 3. Host program is too much to process. 4. Bus loading is too heavy to process. 5. Users need to call the ClearStatus function to clear this condition.
CAN8K_SLOT_NUM_ERROR	Check if the function parameter Slot is out of range of users' WinCon-8000/LinCon-8000 /I-8000.
CAN8K_NOT_INIT	Check if the I8120Init function is called in the beginning of the users' program.

If the I-8120 Err LED is turned on, uses can use function GetStatus( ) to check what is happen on the I-8120. The troubleshooting methods and return value meanings of the function GetStatus( ) is shown as follows.

Error Status	Troubleshooting
Bus off	Reset the I-8120 module by using the function ResetI8120( ).
At least one error	These errors are occurred by different problems in the CAN network. This kind of errors may disappear after receiving or transmit several CAN message successfully. If users want to clean this error status immediately, call the function ResetI8120( ) for the purpose.
Transmit incomplete	This problem may due to the different CAN baud between I-8120 and CAN network. Use functions GetCANBaud( ) and SetCANBaud( ) to fix this problem.
Transmit buffer is locked	This problem may due to the different CAN baud between I-8120 and CAN network. Use functions GetCANBaud( ) and SetCANBaud( ) to fix this problem.
Reception buffer overrun	When the CAN bus loading is heavy, this problem mab be happen. In this case, the received CAN messages will be lost. Reducing the bus loading will improve this situation.
CAN message buffer overflow	This problem is caused by that the host unit can't receive the CAN message in time. Use functions ClearStatus( ) to clean this error flag.
host command buffer overflow	When the I-8120 is too busy to process the host commands, this problem is happen. Use functions ClearStatus( ) to clean this error flag.

If users can't fix the problem after following the steps of the troubleshooting table, please contact to your local distributor to fix the problem.