NAP7000P

User Manual

MEASUREMENT COMPUTING™

Revision 1A

July 1998

Table of Contents

1 Introduction

The **NAP7000P** is a Windows DLL designed for Windows 3.1/95/NT user. It can be called by VC++, BC++, VB, Delphi, BC++ Builder, LabVIEW and TestPoint. Some of the key features of NAP7000P are shown below:

1. multi-tasking DLLs design
2. multi-threaded DLLs design
3. provide general-purpose RS-232 application functions
4. provide general-purpose CB-7000 send/receive functions
5. provide high performance CB-7000 application functions
6. multi-speed demos
7. multi-data-format demos
8. includes over 40 demo programs
9. provides complete diagnostic & utility program, ICPCON.EXE

1.1 Readme.doc

We will co ntinuously upgrade this software, NAP70 0P. Therefore some in formation may not be given in this manual. All the ex tra information will be given in the **readme.doc**. There are **many readme.doc** given in the companion floppy disk as following:

The contents of readme.doc can be as following:

- release notes
- user manual updates
- demo program documentation
- compiler & link documentation
- application notes

We recommended you read the readme.doc file prior to using this software.

2 Windows 95/NT Applications

The **UART.DLL & I7000.DLL** are **Win32 DLLs** drivers designed for Windows 95 and Windows NT 3.51/4.0 applications. The user can us e many programming languages such as **VC++, BC++, BC++ Builder, VB, and Delphi**. There are also many demo program provided for VC++, VB, Delphi and BC++ Builder.

User programs which call UART.DLL & I7000.DLL can be executed under Windows 95 and Windows NT 3.51/4.0 without changing any code.

2.1 UART.DLL

The PC RS-232 COM port related DLLs are provided in UART.DLL. The header file of UART.DLL is provided in I7000.H. The import library of UART.DLL is in UART.LIB. The related files are given in the NAP7000P as following:

\NAP7000P\W32\UART\uart.dll	:DLLs for Windows 95/NT applications
\NAP7000P\W32\I7000\I7000.h	:header file for VC++
\NAP7000P\W32\UART\uart.lib	:import library for VC++
\NAP7000P\W32\BC_LIB\uart.lib	:import library for BC++ and BC++ Builder

2.2 I7000.DLL

The I7000.DLL is d esigned for CB-700 0 series app lications. The I7000.DLL will call UART.DLL to send command and receive result to the CB-7000 series modules. The related files are given in the NAP7000P as following:

\NAP7000P\W32\I7000\i7000.dll	:DLLs for Windows 95/NT applications
\NAP7000P\W32\I7000\i7000.h	:header file for VC++
\NAP7000P\W32\I7000\i7000.lib	:import library for VC++
\NAP7000P\W32\BC_LIB\i7000.lib	:import library for BC++ and BC++ Builder

2.3 Required Files

The VC++ user has to include the following five files:

1. \NAP7000P\W32\UART\uart.dll	:RS232 related DLLs
2. \NAP7000P\W32\UART\uart.lib	:import library of uart.dll
3. \NAP7000P\W32\I7000\i7000.h	:declaration file for uart.dll & i7000.dll
4. \NAP7000P\W32\I7000\i7000.dll	:CB-7000 series related DLLs
5. \NAP7000P\W32\I7000\i7000.lib	:import library of i7000.dll

The BC++ & BC++ Builder user has to include the following four files :

1. \NAP7000P\W32\UART\uart.dll	:RS232 related DLLs
2. \NAP7000P\W32\BC_LIB\uart.lib	:import library of uart.dll
3. \NAP7000P\W32\I7000\i7000.dll	:CB-7000 series related DLLs
4. \NAP7000P\W32\BC_LIB\i7000.lib	:import library of i7000.dll

The VB, Delphi & LabVIEW user has to include only two files:

1. \NAP7000P\W32\UART\uart.dll	:RS232 related DLLs
2. \NAP7000P\W32\I7000\i7000.dll	:CB-7000 series related DLLs

2.4 Demo Programs

There are many demo program designed for VC++, VB, Delphi and BC++ Builder. These demo program in the following directories:

1. \NAP7000P\W32\VCDEMO\*.*	:VC++ demo program
2. \NAP7000P\W32\VBDEMO\*.*	:VB demo program
3. \NAP7000P\W32\Delphi\*.*	:Delphi demo program
4. \NAP7000P\W32\BCB\*.*	:BC++ Builder demo program

The following provides a brief description of the various demo programs.

Demo1	:send/receive command to CB-7000 with checksum disable
Demo2	:send/receive command to CB-7000 with checksum enable
Demo3	:send/receive command to Counter, ANC PC-202
Demo4	:send/receive command to OMRON PLC, CQM1 or C200
Demo5	:multi-speed demo
Demo6	:muti-data-format demo
Demo7	:multi-speed and multi-data-format demo
Demo20	:I7000_AnalogIn demo1
Demo21	:I7000_AnalogIn demo2
Demo22	:I7000_AnalogIn8 demo
Demo23	:I7000_AnalogOut demo
Demo24	:I7000_DigitalIn demo
Demo25	:I7000_DigitalOut demo
Demo26	:multi-speed demo
Demo27	:I7000_I7017In8 demo
Demo28	:I7000_AnalogOutReadBack demo
Demo29	:I7000_DigitalOutReadBack demo
Demo30	:I7000_NetworkAnalogIn demo
Demo31	:I7000_NetworkDigitalIn demo

2.5 VC++ Call DLLs

All the demo program given in \NAP7000P\W32\Vcdemo\*.* are designed with VC++ language. **They have been run under Windows 95/NT and Visual C++ 4.0**. The key points for proper operation are:

1. Enter the DOS command prompt under Windows.
2. Make sure the PATH include the Visual C++ compiler
3. Execute the \MSDEV\BIN\VCVARS32.BAT one time to setup the environment. The VCVARS32.BAT is provided by Visual C++.
4. The source program must include "I7000.H"
5. Copy the UART.LIB, I7000.LIB, UART.DLL and I7000.DLL to the same directory with source program
6. Edit the source program (refer to \nap7000p\w32\vcdemo\demo?\demo?.C)
7. Edit the NMAKE file (refer to \nap7000p\w32\vcdemo\demo?\demo?.MAK)
8. Edit the BATCH file (refer to \nap7000p\w32\vcdemo\demo?\c.bat)
9. Execute the batch file
10. Execute the execution file

2.6 MFC Call DLLs

The usage of NAP7000P DLLs for MFC user is very similar to that for C user. **These have also been tested in Windows 95/NT and Visual C++ 4.0**. The key points for proper operation are:

1. Use MFC wizard to create source code
2. The source program must include “I7000.H”
3. Copy the UART.LIB, I7000.LIB, UART.DLL and I7000.DLL to the same directory with source program
4. Select **Build/Settings/Link** and key-in “UART.LIB I7000.LIB” in the **object/library modules** field

3 Windows 3.1 Applications

7000W31.DLL is a **16-bit** DLL driver designed for Windows 3.1 applications. The user can use **VC++ or VB** to call the 7000W31.DLL.. There are also many demo programs provided for VC++ and VB.

The DLLs for Windows 3.1 and DLLs for Windows 95/NT are virtually identical in usage.

3.1 7000W31.DLL

The header file of 7000W31.DLL is provided in 7000W31.H The import library of 7000W31.DLL is shown in 7000W31.LIB. The key files in NAP7000P are as follows:

\NAP7000P\W31\DLL\7000w31.dll	:DLLs for Windows 3.1 applications
\NAP7000P\W31\DLL\7000w31.h	:header file for VC++
\NAP7000P\W31\DLL\7000w31.lib	:import library for VC++
\NAP7000P\W31\7000W31\*.*	:driver source

3.2 Required Files

The VC++ user must include the following three files:

1. \NAP7000P\W31\DLL\7000w31.dll	:16-bit DLLs for Windows 3.1
2. \NAP7000P\W31\DLL\7000w31.lib	:import library
3. \NAP7000P\W31\DLL\7000w31.h	:declaration file for 7000w31.dll

The VB & LabVIEW user must include only one file:

1. \NAP7000P\W31\DLL\7000w31.dll	:RS232 related DLLs

3.3 Demo Programs

There are many demo programs designed for VC and VB. These demo program are given in the NAP7000P as following:

1. \NAP7000P\W31\VCDEMO\*.* :VC++ Quick-Win demo program
2. \NAP7000P\W31\VBDEMO\*.* :16-bit VB demo program

These demo programs are given as following:

Demo1	:send/receive command to CB-7000 with checksum disable
Demo2	:send/receive command to CB-7000 with checksum enable
Demo3	:send/receive command to Counter, ANC PC-202
Demo4	:send/receive command to OMRON PLC, CQM1 or C200
Demo5	:multi-speed demo
Demo6	:muti-data-format demo
Demo7	:multi-speed and multi-data-format demo

Demo20	:I7000_AnalogIn demo1
Demo21	:I7000_AnalogIn demo2
Demo22	:I7000_AnalogIn8 demo
Demo23	:I7000_AnalogOut demo
Demo24	:I7000_DigitalIn demo
Demo25	:I7000_DigitalOut demo
Demo26	:multi-speed demo
Demo27	:I7000_I7017In8 demo
Demo28	:I7000_AnalogOutReadBack demo
Demo29	:I7000_DigitalOutReadBack demo
Demo30	:I7000_NetworkAnalogIn demo
Demo31	:I7000_NetworkDigitalIn demo

4 UART.DLL for Windows 95/NT

There are six DLL functions provided in the UART.DLLs. This section briefly describes these functions and the various parameters used in each.

4.1 Open_Com

- **Description:**

This DLL will initialize the COM port. This DLL must be **called once before** the other DLLs are called to send/receive command.

- **Syntax:**

WORD Open_Com(char cPort, DWORD dwBaudRate, char cData, char cParity, char cStop)

- **Input Parameter:**

cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

dwBaudRate: 1200/2400/4800/9600/19200/38400/57600/115200

cData: 5/6/7/8 data bit

cParity: 0=NonParity, 1=OddParity, 2=EvenParity

cStop: 0=1-stop, 1=1.5-stp, 2=2-stop

NOTE: cData=8, cParity=0, cStop=0 for CB-7000 modules

- **Return Value:**

NoError = OK

others = Error code, refer to I7000.H

4.2 Close_Com

- **Description:**

This DLL will free all the resources used by Open_Com. This DLL must be **called before** the program exit. The Open_Com will return error message if the program exit without calling Close_Com function.

- **Syntax:**

WORD Close_Com(char cPort)

- **Input Parameter:**

cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

- **Return Value:**

NoError = OK

others = Error code, refer to I7000.H

4.3 Send_Cmd

- **Description:**

  This DLL will create a thread to send a command to CB-7 000 and receive the response -result from CB-7000. If the wCheckSum=1, this DLL will automatically **add the two checksum bytes** to the input string. This DLL will **add the [0x0D]** to the end of the input string, szCmd. The Send_Cmd is a multi-task, multi-thread DLL.

- **Syntax:**

  WORD Send_Cmd(char cPort, char szCmd[], WORD wTimeOut, WORD wCheckSum)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szCmd: the starting address of the original command string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wCheckSum: 0=DISABLE, 1=ENABLE

- **Return Value:**

  NoError : OK

others = Error code, refer to I7000.H

4.4 Read_Com_Status

- **Description:**

  The **Send_Cmd(…)** will create a thread to send a command to CB-7000 and receive the response-result from CB-7000. The Read_Com_Status will return the status of this send/receive thread. If the thread is working, the status value will be smaller than 0x100. If the thread is finished, the status value will be larger than 0x100. **The return value will be equal to 0x105 if the send/receive operation is OK.**

  If the wCheckSum in Send_Cmd is 1, the Recom_Com_Status will check the two checksum bytes of result string. **If the checksum is incorrect, the D13 of return value will be set to 1.**

- **Syntax:**

  WORD Read_Com_Status(char cPort, char szResult[], WORD *wT)

- **Input Parameter:**

  cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szResult: the starting address of the result string (terminated with 0)

  wT: time of send/receive interval, unit = 1 ms

- **Return Value:**

  D0-D7=thread status code, thread start=1 and stop=5

  D8=0 : send/receive not finished, if 1: send/receive done

  D9=1:send/receive timeout

  D10=reserved

  D11=1:Com handle error

  D12=1:send/receive overflow

  D13=1 :checksum error

4.5 Send_Str

- **Description:**

This DLL will create a th read to send a comman d and receive the response-result from a g eneral purpose RS-232 device. The Send_Str is a multi-task, multi-thread DLL. This DLL is very similar to Send_Cmd except that this DLL will **not add any char** to the input string.

- **Syntax:**

WORD Send_Str(char cPort, char szCmd[], WORD wTimeOut, WORD wSendLen, WORD wReceiveLen)

- **Input Parameter:**

cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

szCmd: the starting address of the original command string (terminated with 0)

wTimeOut: constant for time-out control, unit = 1ms

wSendLen: string length of send-string

wReceiveLen: string length of receive-string

- **Return Value:**

NoError : OK

others = Error code, refer to I7000.H

4.6 Send_Receive_Cmd

Description:

This DLL will send a command t o CB-7000 and receive the response-result from CB-7000. If the wCheckSum=1, this DLL will automatically **add the two checksum bytes** to the input string a nd check the checksum status of the receive string. This DLL will **add the [0x0D]** to the end of the i nput string, szCmd. The Send_Receive_Cmd is not a multi-task DLL.

- **Syntax:**

WORD Send_Receive_Cmd(char cPort, char szCmd[], char szResult[], WORD wTimeOut, WORD wCheckSum, WORD *wT)

- **Input Parameter:**

cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

szCmd: the starting address of the input string (terminated with 0)

szResult: the starting address of the result string

wTimeOut: constant for time-out control, unit = 1ms

wCheckSum: 0=DISABLE, 1=ENABLE

wT: time of send/receive interval, unit = 1 ms

- **Return Value:**

NoError : OK

others = Error code, refer to I7000.H

5 I7000.DLL for Windows 95/NT

The I7000.DLL provides a wide variety of high level calls. These are described in the following sections. For specific information regarding call usage, please refer to one of the many demo programs.

5.1 Short_Sub_2

- **Description:**

  Compute C=A-B in **short** format, **short=16 bits sign integer.** This function is provided for testing purpose. To test this DLLs can be called by your programming language, call this subroutine for testing. If this subroutine return the correct value, the other DLLs will work OK also.
- **Syntax:**

  short Short_Sub_2(short nA, short nB);
- **Input Parameter:**

  nA : short integer

  nB : short integer
- **Return Value:** return=nA-nB :short integer

5.2 Float_Sub_2

- **Description:**

  Compute C=A-B in **float** format, **float=32 bits floating pointer number.** This function is prov ided for testing purpose. To test this DLLs can be called by your programming language, call this subroutine for testing. If this subroutine return the correct value, the other DLLs will work OK also.
- **Syntax:**

  float Float_Sub_2(float fA, float fB);
- **Input Parameter:**

  fA : floating point value

  fB : floating point value
- **Return Value:** return=fA-fB :floating point value

5.3 Get_Dll_Version

- **Description:** Read the software version of the NAP7000P DLLs.
- **Syntax:** WORD Get_Dll_Version(void) ;
- **Input Parameter:** void
- **Return Value:** return=0x202 :Version 2.2

5.4 I7000_Test

- **Description:** Test function.
- **Syntax:**
  I7000_Test(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**
  w7000: WORD Input/Output argument table
  f7000: float Input/Output argument table
  szSendTo7000: command string send to CB-7000
  szReceiveFrom7000: result string read from CB-7000
- **Return Value:** NoError

5.5 I7000_AnalogIn

- **Description:**
  Read the analog input value from CB-7000.
- **Syntax:**
  I7000_AnalogIn(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**
  w7000: WORD Input/Output argument table
  f7000: float Input/Output argument table
  szSendTo7000: command string send to CB-7000
  szReceiveFrom7000: result string read from CB-7000
- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**

  w7000[0] : RS-232 port number, 1/2/3/4

  w7000[1] : module address, from 0x00 to 0xFF

  w7000[2] : module ID, 0x7011/7012/7013/7014/7017/7018

  w7000[3] : 0=checksum disable, 1=checksum enable

  w7000[4] : TimeOut constant, normal=100

  w7000[5] : channel number for CB-7017/7018

  w7000[6] : 0 :no save to szSendTo7000&szReceiveFrom7000

  1 :szSendTo7000=command string send to CB-7000

  :szReceiveFrom 7000=result string receive from CB-7000

- **F7000: Float Input/Output Table**

  f7000[0] : analog input value return

```
w7000[0]=cPort;     // port
w7000[1]=wAddr;    // Address
w7000[2]=0x7012;   // ID
w7000[3]=0;        // CheckSum disable
w7000[4]=wTimeOut; // TimeOut constant, normal=100
w7000[6]=1;        // string debug
wRet=I7000_AnalogIn(w7000, f7000, szSend, szReceive);
```

5.6 I7000_AnalogIn8

- **Description:**
  Read the 8 channels of analog input values from CB-7017 or CB-7018.
- **Syntax:**
  I7000_AnalogIn8(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**
  w7000: WORD Input/Output argument table
  f7000: float Input/Output argument table
  szSendTo7000: command string send to CB-7000
  szReceiveFrom7000: result string read from CB-7000
- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**
  w7000[0] : RS-232 port number, 1/2/3/4
  w7000[1] : module address, from 0x00 to 0xFF
  w7000[2] : module ID, 0x7017/7018
  w7000[3] : 0=checksum disable, 1=checksum enable
  w7000[4] : TimeOut constant, normal=100
  w7000[6] : 0 :no save to szSendTo7000&szReceiveFrom7000
  1 :szSendTo7000=command string send to CB-7000
  :szReceiveFrom 7000=result string receive from CB-7000
- **F7000: Float Input/Output Table**
  f7000[0] : analog input value of channel_0
  f7000[1] : analog input value of channel_1
  ……………………………………………
  f7000[7] : analog input value of channel_7

5.7 I7000_7017In8

- **Description:**

  Read the 8 channels of analog input values from CB-7017. This DLL will send high speed command, **$AAA**, to CB-7017. **The I7000_7017In8 is faster than I7000_AnalogIn8.**

- **Syntax:**

  I7000_7017In8(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])

- **Input Parameter:**

  w7000: WORD Input/Output argument table

  f7000: float Input/Output argument table

  szSendTo7000: command string send to CB-7000

  szReceiveFrom7000: result string read from CB-7000

- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**

  w7000[0] : RS-232 port number, 1/2/3/4

  w7000[1] : module address, from 0x00 to 0xFF

  w7000[2] : module type, 08/09/0A/0B/0C/0D

  w7000[3] : 0=checksum disable, 1=checksum enable

  w7000[4] : TimeOut constant, normal=100

  w7000[6] : 0 :no save to szSendTo7000&szReceiveFrom7000

  1 :szSendTo7000=command string send to CB-7000

  :szReceiveFrom 7000=result string receive from CB-7000

- **F7000: Float Input/Output Table**

  f7000[0] : analog input value of channel_0

  f7000[1] : analog input value of channel_1

  ……………………………………………

  f7000[7] : analog input value of channel_7

5.8 I7000_AnalogOut

- **Description:**
  Send the analog output command to CB-7000.
- **Syntax:**
  I7000_AnalogOut(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**
  w7000: WORD Input/Output argument table
  f7000: float Input/Output argument table
  szSendTo7000: command string send to CB-7000
  szReceiveFrom7000: result string read from CB-7000
- **Return Value:**

NoError: OK
others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**
  w7000[0] : RS-232 port number, 1/2/3/4
  w7000[1] : module address, from 0x00 to 0xFF
  w7000[2] : module ID, 0x7021
  w7000[3] : 0=checksum disable, 1=checksum enable
  w7000[4] : TimeOut constant, normal=100
  w7000[6] : 0 :no save to szSendTo7000&szReceiveFrom7000
  1 :szSendTo7000=command string send to CB-7000
  :szReceiveFrom 7000=result string receive from CB-7000
- **F7000: Float Input/Output Table**
  f7000[0] : analog output value

```
w7000[0]=cPort;     // port
w7000[1]=wAddr;    // Address
w7000[2]=0x7021;   // ID
w7000[3]=0;        // CheckSum disable
w7000[4]=wTimeOut; // TimeOut constant
w7000[6]=1;        // string debug
f7000[0]=5.432;     // DA output value
wRet=I7000_AnalogOut(w7000, f7000, szSend, szReceive);
```

5.9 I7000_AnalogOutReadBack

- **Description:**

  Read back the current D/A output value of CB-7021. There are two types of analog output read back as following:

  1. **command read back by $AA6 command**
  2. **analog output of current path read back by $AA8 command**

- **Syntax:**

  I7000_AnalogOutReadBack(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])

- **Input Parameter:**

  w7000: WORD Input/Output argument table

  f7000: float Input/Output argument table

  szSendTo7000: command string send to CB-7000

  szReceiveFrom7000: result string read from CB-7000

- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**

  w7000[0]: RS-232 port number, 1/2/3/4

  w7000[1]: module address, from 0x00 to 0xFF

  w7000[2]: module ID, 0x7021

  w7000[3]: 0=checksum disable, 1=checksum enable

  w7000[4]: TimeOut constant, normal=100

  w7000[5]: 0: command read back ($AA6)

  1: analog output of current path read back ($AA8)

  w7000[6]: 0 :no save to szSendTo7000&szReceiveFrom7000

  1 :szSendTo7000=command string send to CB-7000

  :szReceiveFrom 7000=result string receive from CB-7000

- **F7000: Float Input/Output Table**

  f7000[0] : analog output read back value

5.10 I7000_DigitalIn

- **Description:**

  Read the digital input value from a CB-7000 series DIO module.
- **Syntax:**

  I7000_DigitalIn(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**

  w7000: WORD Input/Output argument table

  f7000: float Input/Output argument table

  szSendTo7000: command string send to CB-7000

  szReceiveFrom7000: result string read from CB-7000
- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**

  w7000[0] : RS-232 port number, 1/2/3/4

  w7000[1] : module address, from 0x00 to 0xFF

  w7000[2] : module ID, 0x7050/7052/7053/7060/7041/7044

  w7000[3] : 0=checksum disable, 1=checksum enable

  w7000[4] : TimeOut constant, normal=100

  w7000[5] : 16-bit digital input data

  w7000[6] : 0 :no save to szSendTo7000&szReceiveFrom7000

  1 :szSendTo7000=command string send to CB-7000

  :szReceiveFrom 7000=result string receive from CB-7000
- **F7000: Float Input/Output Table**

  void

```
w7000[0]=cPort;    // port
w7000[1]=wAddr;    // Address
w7000[2]=0x7053;   // ID
w7000[3]=0;        // CheckSum disable
w7000[4]=wTimeOut; // TimeOut constant
w7000[6]=1;        // string debug
wRet=I7000_DigitalIn(w7000, f7000, szSend, szReceive);
```

5.11 I7000_DigitalOut

- **Description:**
  Set the digital output value of CB-7000 series digital I/O module.
- **Syntax:**
  I7000_DigitalOut(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**
  w7000: WORD Input/Output argument table
  f7000: float Input/Output argument table
  szSendTo7000: command string send to CB-7000
  szReceiveFrom7000: result string read from CB-7000
- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**
  w7000[0] : RS-232 port number, 1/2/3/4
  w7000[1] : module address, from 0x00 to 0xFF
  w7000[2] : module ID, 0x7050/7060/7067/7042/7043/7044
  w7000[3] : 0=checksum disable, 1=checksum enable
  w7000[4] : TimeOut constant, normal=100
  w7000[5] : 16-bit digital output data
  w7000[6] : 0 :no save to szSendTo7000&szReceiveFrom7000
  1 :szSendTo7000=command string send to CB-7000
  :szReceiveFrom 7000=result string receive from CB-7000
- **F7000: Float Input/Output Table**
  void

```
w7000[0]=cPort;     // port
w7000[1]=wAddr;    // Address
w7000[2]=0x7053;   // ID
w7000[3]=0;        // CheckSum disable
w7000[4]=wTimeOut; // TimeOut constant
w7000[5]=wDoVal;  // digital output value
w7000[6]=1;        // string debug
wRet=I7000_DigitalOut(w7000, f7000, szSend, szReceive);
```

5.12 I7000_DigitalOutReadBack

- **Description:**
  Read back the digital output value of CB-7000 series module.
- **Syntax:**
  I7000_DigitalOutReadBack(WORD w7000[], float f7000[], char szSendTo7000[], char szReceiveFrom7000[])
- **Input Parameter:**
  w7000: WORD Input/Output argument table
  f7000: float Input/Output argument table
  szSendTo7000: command string send to CB-7000
  szReceiveFrom7000: result string read from CB-7000
- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

- **W7000: WORD Input/Output Table**

```
 w7000[0] : RS-232 port number, 1/2/3/4
 w7000[1] : module address, from 0x00 to 0xFF
 w7000[2] : module ID, 0x7050/7060/7067/7042/7043/7044
 w7000[3] : 0=checksum disable, 1=checksum enable
 w7000[4] : TimeOut constant, normal=100
 w7000[5] : 16-bit digital output data read back
 w7000[6] : 0      :no save to szSendTo7000&szReceiveFrom7000
           1      :szSendTo7000=command string send to CB-7000
             :szReceiveFrom             7000=result string receive from CB-7000
```

- **F7000: Float Input/Output Table**
  void

```
w7000[0]=cPort;     // port
w7000[1]=wAddr;    // Address
w7000[2]=0x7053;   // ID
w7000[3]=0;        // CheckSum disable
w7000[4]=wTimeOut; // TimeOut constant
w7000[6]=1;        // string debug
wRet=I7000_DigitalOutReadBack(w7000, f7000, szSend, szReceive);
// w7000[5] = digital output read back
```

5.13 I7000_NetworkAnalogIn

- **Description:**

 Read the multi-module analog input value from CB-7000 RS-485 network.. The user can call I7000_AnalogIn to read analog input value one by one or call this function once for easy programming.

- **Syntax:**

I7000_NetworkAnalogIn(WORD wPort, WORD wTotal, WORD wT, WORD wID[], WORD wConfig[], WORD wChksum[], float f7000[])

- **Input Parameter:**

wPort: RS-232 port number, 1/2/3/4

wTotal: number of modules to read

wT: time out constant, normal=100

wID: wID[?]=module address of module_?, from 0x00 to 0xFF

wConfig: if wID[?]=0x7017 then wConfig[?]=08/09/0A/0B/0C/0D

if wID[?] !=0x7017 then wConfig[?] is ignored

wChksum: if wChksum[?]=1 :the checksum of module_? is enable

f7000[0]: analog value of module_0, channel_0

f7000[1]: analog value of module_0, channel_1

……………………………………………..

f7000[7]: analog value of module_0, channel_7

f7000[8]: analog value of module_1, channel_0

……………………………………………..

f7000[15]: analog value of module_1, channel_7

f7000[n*8]: analog value of module_n, channel_0

f7000[n*8+1]: analog value of module_n, channel_1

……………………………………………..

f7000[n*8+7]: analog value of module_n, channel_7

- **Return Value:**

NoError: OK

others: Error code, refer to I7000.H

5.14 I7000_NetworkDigitalIn

- **Description:**

  Read the multi-module digital input value from CB-7000 RS-485 network.. The user can call I7000_DigitalIn to read analog input value one by one or call this function once for easy programming.

- **Syntax:**

  I7000_NetworkDigitalIn(WORD wPort, WORD wTotal, WORD wT, WORD wID[], WORD wConfig[], WORD wChksum[], WORD w7000[])

- **Input Parameter:**

  wPort: RS-232 port number, 1/2/3/4

  wTotal: number of modules to read

  wT: time out constant, normal=100

  wID: wID[?]=module address of module_?, from 0x00 to 0xFF

  wConfig: reserved.

  wChksum: if wChksum[?]=1 :the checksum of module_? is enable

  w7000[0]: 16-bit digital value of module_0

  w7000[1]: 16-bit digital value of module_1

  ……………………………………………

  w7000[n]: 16-bit digital value of module_n

- **Return Value:**

NoError: OK

others: Error code, refer to 7000W31.H.

5.15 VC I7000 Demo Programs

- Demo20 :**I7000_AnalogIn** demo1
- Demo21 :**I7000_AnalogIn** demo2
- Demo22 :**I7000_AnalogIn8** demo
- Demo23 :**I7000_AnalogOut** demo
- Demo24 :**I7000_DigitalIn** demo
- Demo25 :**I7000_DigitalOut** demo
- Demo26 :multi-speed demo
- Demo27 :**I7000_I7017In8** demo
- Demo28 :**I7000_AnalogOutReadBack** demo
- Demo29 :**I7000_DigitalOutReadBack** demo
- Demo30 :**I7000_NetworkAnalogIn** demo
- Demo31 :**I7000_NetworkDigitalIn** demo

5.16 I7000.DLL Driver Source

The driver source of I7000.DLL are given as following:

\NAP7000P\W32\I7000\I7000.H	: declaration file
\NAP7000P\W32\I7000\I7000.C	: program source file
\NAP7000P\W32\I7000\I7000.DEF	:definition file
\NAP7000P\W32\I7000\I7000.MAK	:VC++ 4.0 make file
\NAP7000P\W32\I7000\UART.LIB	:import library of UART.DLL

- Use VC++ 4.0 to make this DLLs
- The I7000.DLL will call UART.DLL, therefore the UART.LIB must be put in the same directory with I7000.MAK

6 7000W31.DLL for Windows 3.1

6.1 Open_Com

Description:

This DLL will initialize the COM port. This DLL must be **called once before** the other DLLs are called to send/receive command.

- **Syntax:**

  WORD Open_Com(char cPort, DWORD dwBaudRate, char cData, char cParity, char cStop)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  dwBaudRate: 1200/2400/4800/9600/19200/38400/57600/115200

  cData: 5/6/7/8 data bit

  cParity: 0=NonParity, 1=OddParity, 2=EvenParity

  cStop: 0=1-stop, 1=1.5-stp, 2=2-stop

  NOTE: cData=8, cParity=0, cStop=0 for CB-7000 modules

- **Return Value:**

NoError = OK

others = Error code, refer to 7000W31.H

6.2 Close_Com

Description:

This DLL will free all the resources used by Open_Com. This DLL must be **called before** the program exit. The Open_Com will return error message if the program exit without calling Close_Com function.

- **Syntax:**

  WORD Close_Com(char cPort)

- **Input Parameter:**

  cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

- **Return Value:**

NoError = OK

others = Error code, refer to 7000W31.H

- **Demo Program:**

6.3 Send_Cmd

Description:

This DLL will send a command string to CB-7000. If the wCheckSum=1, this DLL will automatically **add the two checksum bytes** to the input string. This DLL will **add the [0x0D]** to the end of the input command string, szCmd.

- **Syntax:**

  WORD Send_Cmd(char cPort, char szCmd[], WORD wTimeOut, WORD wCheckSum)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szCmd: the starting address of the original command string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wCheckSum: 0=DISABLE, 1=ENABLE

- **Return Value:**

  NoError : OK

others = Error code, refer to 7000W31.H

6.4 Receive_Cmd

Description:

This DLL will receive a result string from CB-7000. If the wCheckSum=1, this DLL will automatically **check the two checksum bytes** of the result string. This DLL will check the result string terminator, 0x0D.

- **Syntax:**

  WORD Receive_Cmd(char cPort, char szResult[], WORD wTimeOut, WORD wCheckSum)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szResult: the starting address of the result string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wCheckSum: 0=DISABLE, 1=ENABLE

- **Return Value:**

  NoError : OK

others = Error code, refer to 7000W31.H

6.5 Send_Receive_Cmd

Description:

This DLL will send a command stri ng to CB-7000 then receive the result string from I 7000. This DLL will call Send_Cmd first. Then call Receive_Cmd next.

- **Syntax:**

  WORD Send_Receive_Cmd(char cPort, char szCmd[], char szResult, WORD wTimeOut, WORD wCheckSum)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szCmd: the starting address of the original command string (terminated with 0)

  szResult: the starting address of the result string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wCheckSum: 0=DISABLE, 1=ENABLE

- **Return Value:**

  NoError : OK

others = Error code, refer to 7000W31.H

6.6 Send_Str

Description:

This DLL will send a command to a general purpose RS-232 device. This DLL is very similar to Send_Cmd except that this DLL will **not add any char** to the input string.

- **Syntax:**

  WORD Send_Str(char cPort, char szCmd[], WORD wTimeOut, WORD wLenT)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szCmd: the starting address of the original command string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wLenT: string length of send-string

- **Return Value:**

  NoError : OK

others = Error code, refer to 7000W31.H

6.7 Receive_Str

Description:

This DLL will receive a re sult string from a general purpose RS-232 device. This DLL is very similar to Receive_Cmd except that this DLL will **not check** result string terminator, 0x0d.

- **Syntax:**

  WORD Receive_Str(char cPort, char szReceive[], WORD wTimeOut, WORD wLenR)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szReceive: the starting address of the original command string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wLenR: string length of receive-string

- **Return Value:**

  NoError : OK

others = Error code, refer to 7000W31.H

6.8 Send_Receive_Str

Description:

This DLL will send a c ommand to a general purpose RS-232 device and receive the response string. This DLL will call Send_Cmd first. Then call Receive_Cmd next.

- **Syntax:**

  WORD Send_Str(char cPort, char szCmd[], char szResult, WORD wTimeOut, WORD wLenT, WORD wLenR)

- **Input Parameter:**

  cPort: 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

  szCmd: the starting address of the original command string (terminated with 0)

  szResult: the starting address of the result string (terminated with 0)

  wTimeOut: constant for time-out control, unit = 1ms

  wLenT: string length of send-string

  wLenR: string length of receive-string

- **Return Value:**

  NoError : OK

others = Error code, refer to 7000W31.H

6.9 VC Demo Programs

A wide variety of demo programs showing the usage of the 7000W31 DLL are provided. The following provides a very brief description of the demos provided.

- Demo20 :**I7000_AnalogIn** demo1
- Demo21 :**I7000_AnalogIn** demo2
- Demo22 :**I7000_AnalogIn8** demo
- Demo23 :**I7000_AnalogOut** demo
- Demo24 :**I7000_DigitalIn** demo
- Demo25 :**I7000_DigitalOut** demo
- Demo26 :multi-speed demo
- Demo27 :**I7000_I7017In8** demo
- Demo28 :**I7000_AnalogOutReadBack** demo
- Demo29 :**I7000_DigitalOutReadBack** demo
- Demo30 :**I7000_NetworkAnalogIn** demo
- Demo31 :**I7000_NetworkDigitalIn** demo

6.10 7000W31.DLL Driver Source

The driver source of 7000W31.DLL are given as following:

\NAP7000P\W31\7000W31\7000w31.h	:declaration file
\NAP7000P\W31\7000W31\7000w31.def	:definition file
\NAP7000P\W31\7000W31\7000w31.mak	:NMAKE file
\NAP7000P\W31\7000W31\7000w31.c	:program source file

- Use VC++ 2.0 to make this DLLs

For Your Notes

For Your Notes