C502 Dual-Port Sync Board

User's Manual

First Edition, July 2000

Moxa Technologies Co., Ltd.

Tel: +866-2-8919-1230

Fax: +886-2-8919-1231

www.moxa.com

support@moxa.com.tw

C502 Dual-Port Sync Board Userver's Manual

MOXA Internet Services

Customer's satisfaction is always our number one concern. To ensure that customers get the full benefit of our services, Moxa Internet Services have been built for technical support, product inquiry, new driver update, user's manual update, etc.

The followings are the services we provide.

E-mail for technical support
    address: service@moxa.com.tw

FTP site for free driver update
    address: ftp.moxa.com
  or
    ftp.moxa.com.tw
    user ID: ftp
    password: your_email_address

World Wide Web (WWW) Site for product info
    address: www.moxa.com
  or
    www.moxa.com.tw

Document Organization

Chapter 1, “C502 Overview”, describes features and specifications for MOXA C502.

Chapter 2, “C502 Hardware Installation”, describes how to install C502 board in your PC.

Chapter 3, “C502 Software Installation”, describes how to install/remove C502 Windows NT driver.

Chapter 4, “API Programming Library”, lists all library functions for MOXA C502 with C/++, VB or Delphi language in Windows NT.

Table of Contents

1 Overview

Features

MOXA C502 is a high-speed intelligent dual-port control board with synchronous communication modules for PC/AT under Windows NT environment. It is equipped with a RISC CPU, 1Mbytes dual ported RAM and 128Kbytes SRAM for firmware download. With C/C++, VB and Delphi self-developing package, high-speed synchronous communication programming is just a breeze. Excellent hardware components and software techniques make C502 perfect for high throughput front-end processing applications.

Designed for high-speed synchronous communication, MOXA C502 is suitable for IBM PC/AT and compatible systems under Windows NT environment.

Specifications

- On board RISC CPU
- 1M bytes dual port RAM buffer
- 128K bytes SRAM
- Baud rate up to 4Mbps(ISA) and 8Mbps(PCI) for V.35, while 128Kbps for RS-232 respectively
- Cable selection V.35/RS-232 interface compatible
- Free Windows NT 4.0 developing tool
- High performance SCA HD64570-10 serial communication adapter with DMA controller for ISA, while HD64570-16 is suitable for PCI

- ❑ IRQ:2,3,4,5,7,9,10,11,12,15 jumper selectable for ISA, while no jumper selection for PCI
- ❑ System: PC ISA/EISA/PCI bus
- ❑ Support HDLC, SDLC, Mono-Sync, Bi-Sync

Packaging List

Upon unpacking MOXA C502, you will find the following items:

- ❑ MOXA C502/ISA or C502/PCI Sync Board
- ❑ RS-232 or V.35 Connection cable
- ❑ MOXA C502 user's manual.
- ❑ MOXA C502 driver diskette for Windows NT

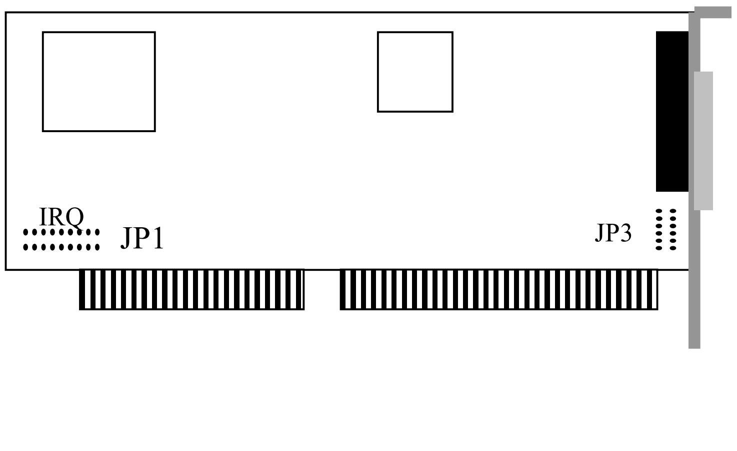
2
Hardware Installation

1. Power-off PC and remove PC cover.

2. Configure C502/ ISA board, while C502/PCI board is directly designated by PC Bios assignment.

❑ IRQ number: Find an available IRQ number in your system and setup jumper JP1. There are 9 IRQ numbers you can choose from. **If you want to add more than one C502/ISA board, their IRQ numbers must be set the same.**

❑ Base address: Choose a base address (occupying 16KB) which is not used by expansion memory or other add-on cards. There are 6 memory banks you can choose from at jumper JP3. **If you want to add more than one C502/ISA board, each board must have a unique address.**



Warning: Make sure your system is powered-off before you start installing the I/O board. If not, you may risk damaging both of your system and the board.

3. After the setting has been done, choose an available 16-bit expansion slot for ISA board and 32-bit expansion slot for PCI board separately. Remove the retaining screw and put it aside.

4. Remove the slot cover.

5. Orient C502 edge connector facing downward. Place it in the I/O slot. Press the board firmly into the plastic edge connector socket on the computer motherboard.

6. Use retaining screw to secure C502 to the rear panel. You can install up to four C502 boards in your system at one time.

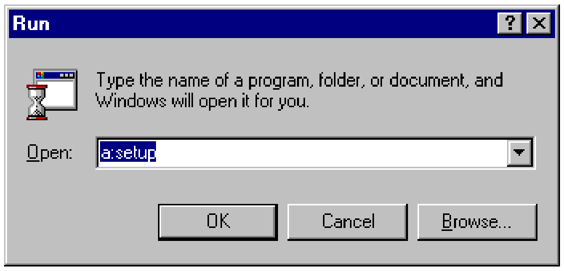
7. Put back PC cover.

3
Software Installation

C502 software includes Windows NT driver, Configuration, Win32 API, and uninstallation program.

Install C502 Windows NT Driver

1. Insert C502 Driver for Windows NT disk into drive A. From "**Start**"menu, click on "**Run**" to continue.



Type the name of a program, folder, or document, and Windows will open it for you.

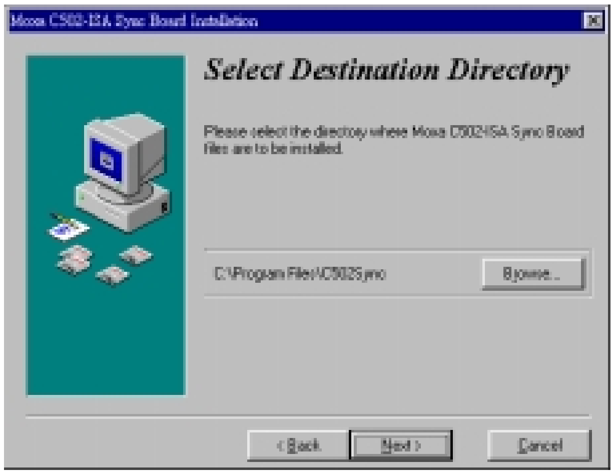
a:setup

Figure 3-1

2. Type in"**a:\setup.exe**", then click "OK" to continue.

3. Setup program prompts you a welcome message and asks if you want to install C502 program now. Click "**Next**" to continue

4. Enter the name of directory to install the C502 files. You click "**Next**" to use default directory name.


Moxa C502-ISA Sync Board Installation

Select Destination Directory

Please select the directory where Moxa C502-ISA Sync Board files are to be installed.

C:\Program Files\C502Sync

Figure 3-2

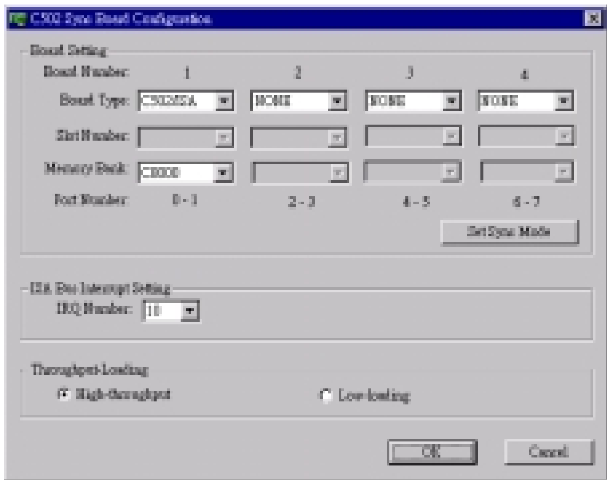
5. Configuration program will start automatically after installation completes..

C502 Configuration

For ISA boards:

1. From “**Start**” menu, select “**Program**”→“**Moxa Sync Board**”→”**Configuration**” in sequence.

2. There are three kinds of “**Board** Type” field: None, C-502/ISA, and C-502/PCI. Select “**C502/ISA**” from the “**Board Type**” pull-down list.


C502 Sync Board Configuration

Board Setting

Board Number:	1	2	3	4
Board Type:	C502ISA	NONE	NONE	NONE
Slot Number:				
Memory Bank:	C8000			
Port Number:	0 - 1	2 - 3	4 - 5	6 - 7

ISA Bus Interrupt Setting
IRQ Number: 10

Throughput-Loading
[✓ High-throughput] Low-loading

Figure 3-3

3. Select a specific “**Memory Bank**” number from the “**Memory Bank**” pull-down list. Each C502/ISA board must have a unique memory bank address. Enter the value you set on Jumper 3 while configuring C502/ISA board.

4. Select a specific “**IRQ** Number” from the “**ISA Bus Interrupt Setting**” field. The IRQ number is shared by each C502/ISA board.

5. You must select at least one ISA board from the “**Board type**” pull-down list.

6. Reboot the system.

For PCI boards:

7. One PCI board should be plugged into the main board of PC before the system is powered-on.

8. There are three kinds of "Board **Type**" field. Select "**C-502 PCI**" from the "**Board Type**" pull-down list. Then, cancel one board, and select "**None**" from the "**Board Type**" pull-down list. The main board will find out a piece of PCI board at least under the main board, so that you can configure C-502/PCI board respectively.

9. Select a specific "**Slot Number**" respectively from the "**Slot Number**" pull-down list.

10. The software should be reconfigured whenever a new PCI slot of hardware is being changed accordingly.

11. Reboot the system.

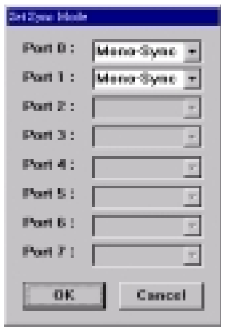
Note the following:

1. Both of the C502/ISA and C502/PCI boards can be plugged-in under the same system. Up to four(4) C502/ISA and PCI boards are allowed in a system.

2. If you want to add more than one C502/ISA board, their IRQ numbers must be set the same. However, each C502/PCI board must has its own IRQ number separately.

3. Slot number is available whenever selecting "**C-502/PCI**" from the "**Board Type**" pull-down list. Oppositely, memory bank and IRQ Bus are available whenever selecting "**C-502/ISA**" from the "**Board type**" pull-down list.

4. Click on **Set Sync Mode** for figure 3-3 to select the synchronous mode. You must set at least one board for the **Set Sync Mode** button to be active.


Set Sync Mode

Port 0 : Mono-Sync
Port 1 : Mono-Sync
Port 2 :
Port 3 :
Port 4 :
Port 5 :
Port 6 :
Port 7 :

Figure 3-4

Select a synchronous mode from the pull down list. If you change the mode then you need to reboot the system. If you want to activate a change to the synchronous mode, you must run this configuration program and then reboot the system.

5. PCI boards feature:

a) If you cannot find out C502/PCI board in the system, then you cannot install the PCI board completely. Though you may use one PCI board in the previous time, you still cannot install PCI board without any PCI boards in the system.

b) Once the PCI board is plugged-in the main board, then you may install PCI board.

c) Once unplugged the board, you will not see the previous configuration which aims at a specific PCI board.

d) Once you replug the PCI board, then you may see the configuration both of board type and slot number that have been done the first time.

6. "Throughput-Loading" for Figure 3-3 is optional. You may choose either "High-throughput" or "Low-loading".

a) High-throughput: whenever this driver send data, throughput is the first priority. The system's loading is heavy, however, the throughput is much better to compare with that of low-loading.

Morever, its throughput can reach up to 4 Mbps of the full-rate status.

b) Low-loading: the system's loading is light, however the throughput is lower to compare with that of high-throughput oppositely. Furthermore, its throughput cannot reach up to 2 Mbps.

Remove C502 Windows NT Driver

From '**Start**' menu, select '**Program**'→'**Moxa Sync Board**'→'**Uninstall**' in sequence. Then, it will automatically remove all C502 programs.

4

API Programming Library

API Programming Library Notes

MOXA C502 supports C/C++, VB and Delphi language. If you use VB, include ‘syncapi.bas’ file in your project. If you use Delphi, include ‘syncapi.pas’. All of the languages need ‘syncapi.dll’ file, which is copied to your PC when you install C502 driver.

The ‘syncapi.lib’ library file is used for Microsoft C/C++. If you’re using Borland C/C++ Compiler, please use the utility ‘implib.exe’ of Borland C++ to execute "**implib -c syncapib.lib syncapi.dll**" and obtain Borland-compliant library file ‘syncapib.lib’ from the dynamic link library "syncapi.dll".

MOXA C502 supports block/non-block mode for reading/writing function with your application.

Following is the return code list you may encounter when calling these library functions:

Return Code	**Output**	**Description**
SYIO_OK	0	function OK
SYIO_BADPORT	-1	no such port or not opened
SYIO_OPENED	-2	port opened
SYIO_BADPARM	-3	parameter error
SYIO_WIN32FAIL	-4	call win32 function fail, call GetLastError() function to get the return code
SYIO_ABORT	-5	abort writing
SYIO_TIMEOUT	-6	read or write timeout
SYIO_BUFFERTOOSHORT	-8	buffer too short

Library Function Description

syio_Open

Description:

Open one port and set port to default value. Port default value is Tx clock out, baud rate 38400, CRC CCITT_1, and data encoding NRZ.

Syntax:

C/C++

```
Int WINAPI syio_Open (int port);
Input     : int port (port number 0~7)
Output    : refer to return code list
```

VB

```
Declare Function syio_Open Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

```
function syio_Open (port: Longint): Longint; stdcall;
implementation
function syio_Open; external ‘syncapi.dll’;
```

syio_Close

Description:

Close one opened port. If there is no need to use one port, you can call this function. It will wait the data to send over. If there is no data to send in 3 seconds, it will flush output and input data on the buffer of the driver.

Syntax:

C/C++

```
Int WINAPI syio_Close (int port);
Input     : int port (port number 0~7)
Output    : refer to return code list
```

VB

Declare Function syio_Close Lib "syncapi.dll" (ByVal port As Long) As Long

Delphi

```
function syio_Close (port: Longint): Longint; stdcall;
implementation
function syio_Close; external 'syncapi.dll';
```

syio_Write

Description:

Send data. If you set write-timeout to zero, it will write the data to dual-port DRAM on board and return as soon as possible. If you set write-timeout to a specific value, it will block syio_Write function call until data writing is completed or times out.

Syntax:

C/C++

```
int WINAPI syio_Write (int port, char *buf, int len);
Input    : int port     : port number 0~7
           char*buf     : to-send data buffer pointer
           Int len      : to-send data buffer pointer
Output   : >=0          : sent data length
           <0           : refer to return code list
```

VB

Declare Function syio_Write Lib "syncapi.dll" (ByVal port As Long, ByRef buf As Byte, ByVal len As Long) As Long

Delphi

```
function syio_Write (port: Longint; buf: PChar; len: Longint): Longint; stdcall;
implementation
function syio_Write; external 'syncapi.dll';
```

syio_Read

Description:

Receive data from remote device. If you set the read-timeout to zero, it will return as soon as possible when there is no incoming data. If you set read-timeout to non-zero value, it will block syio_Read function call until data reading is over or times out.

Syntax:

C/C++

```
int WINAPI syio_Read (int port, char *buf, int len);
Input    : int port     : port number 0~7
           char*buf     : to-receive data buffer pointer
           Int len      : to-receive data buffer pointer
Output   : >=0          : receiving data length
           <0           : refer to return code list
```

VB

```
Declare Function syio_Read Lib "syncapi.dll" (ByVal port As Long, ByRef buf As
Byte, ByVal len As Long) As Long
```

Delphi

```
function syio_Read (port: Longint; buf: PChar; len: Longint): Longint; stdcall;
implementation
function syio_Read; external 'syncapi.dll';
```

syio_Flush

Description:

Flush received or to-be-send data on the driver.

Syntax:

C/C++

```
int WINAPI syio_Flush (int port, int mode);
Input    : int port     : port number 0~7
           int mode     : FLUSH_INPUT, FLUSH_OUTPUT or FLUSH_ALL
Output   : refer to return code list
```

VB

```
Declare Function syio_Flush Lib "syncapi.dll" (ByVal port As Long, ByVal mode As
Long) As Long
```

Delphi

```
function syio_Flush (port, mode: Longint): Longint; stdcall;
implementation
function syio_Flush; external 'syncapi.dll';
```

syio_View

Description:

Preview data. It functions like syio_Read, but data stays on the driver afterward. It has not timeout value.

Syntax:

C/C++

```
int WINAPI syio_View (int port, char *buf, int len);
Input     : int port      : port number 0~7
            char*buf      : to-view data buffer pointer
            Int len       : to-view data buffer pointer
Output    : >=0           : viewing data length
            <0            : refer to return code list
```

VB

Declare Function syio_View Lib “syncapi.dll” (ByVal port As Long, ByRef buf As Byte, ByVal len As Long) As Long

Delphi

function syio_View (port: Longint; buf: PChar; len: Longint): Longint; stdcall;
implementation
function syio_View; external ‘syncapi.dll’;

syio_SetBaud

Description:

Set baud rate. Baud rate setting is invalid if Tx Clock is set as ‘in’. You can set Tx clock ‘out’ to activate baud rate setting.

Syntax:

C/C++

```
int WINAPI syio_Set Baud (int port, int speed);
Input     : int port      : port number 0~7
            int speed     : to-set baud rate
Output    : refer to return code list
```

VB

```
Declare Function syio_SetBaud Lib "syncapi.dll" (ByVal port As Long, ByVal speed
As Long) As Long
```

Delphi

```
function syio_SetBaud (port, speed: Longint): Longint; stdcall;
implementation
function syio_SetBaud; external 'syncapi.dll';
```

syio_GetBaud

Description:

Get baud rate setting value.

Syntax:

C/C++

```
int WINAPI syio_GetBaud (int port);
Input      : int port       : port number 0~7
             int speed      : set baud rate
Output     : >=0            : set baud rate
             <0             : refer to return code list
```

VB

```
Declare Function syio_GetBaud Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi

```
function syio_GetBaud (port: Longint): Longint; stdcall;
implementation
function syio_GetBaud; external 'syncapi.dll';
```

syio_SetReadTimeouts

Description:

Set the syio_Read timeout value. Please refer to syio_Read function.

Syntax:

C/C++

```
int WINAPI syio_View (int port, DWORD *timesouts);
Input     : int port              : port number 0~7
            DWORD timeouts        : to-set timeouts value. Time unit is millisecond
Output    : reter to return code list
```

VB

Declare Function syio_SetReadTimeouts Lib “syncapi.dll” (ByVal port As Long, ByVal timeouts As Long) As Long

Delphi

function syio_SetReadTimeouts(port, timeouts: Longint): Longint; stdcall;
implementation
function syio_SetReadTimeouts; external ‘syncapi.dll’;

syio_GetReadTimeouts

Description:

Get read-timeout setting value. Please refer to the syio_Read and syio_SetReadTimeouts function.

Syntax:

C/C++

```
int WINAPI syio_GetReadTimeouts (int port, DWORD *timesouts);
Input     : int port              : port number 0~7
            DWORD*timesouts       : to get timesouts pointer
Output    : refer to return code list
```

VB

Declare Function syio_GetReadTimeouts Lib “syncapi.dll” (ByVal port As Long, ByRef timeouts As Long) As Long

Delphi

function syio_GetRedTimeouts (port: Longint; var timeouts: Longint): Longint; stdcall;
implementation
function syio_GetReadTimeouts; external ‘syncapi.dll’;

syio_SetWriteTimeouts

Description:

Set write-timeout setting value. Please refer to the syio_Write function.

Syntax:

C/C++

```
int WINAPI syio_SetWriteTimeouts (int port, DWORD *timesouts);
Input     : int port                : port number 0~7
            DWORD*timesouts         : to set write timesouts pointer
Output    : refer to return code list
```

VB

```
Declare Function syio_SetWriteTimeouts Lib "syncapi.dll" (ByVal port As Long,
ByVal timeouts As Long) As Long
```

Delphi

```
function syio_SetWriteTimeouts (port, timeouts: Longint): Longint; stdcall;
implementation
function syio_SetWriteTimeouts; external 'syncapi.dll';
```

syio_GetWriteTimeouts

Description:

Get write-timeout setting value. Please refer to syio_Write and syio_SetWriteTimeouts function for more details.

Syntax:

C/C++

```
int WINAPI syio_GetWriteTimeouts (int port, DWORD*timeouts);
Input     : int port                : port number 0~7
            DWORD*timesouts         : to get timesouts pointer
Output    : refer to return code list
```

VB

```
Declare Function syio_GetWriteTimeouts Lib "syncapi.dll" (ByVal port As Long,
ByRef timeouts As Long) As Long
```

Delphi

function syio_GetWriteTimeouts (port: Longint; var timeouts: Longint): Longint; stdcall; implementation
function syio_GetWriteTimeouts; external ‘syncapi.dll’;

syio_AbortRead

Description:

Abort the blocked syio_Read function call.

Syntax:

C/C++

```
int WINAPI syio_AbortRead (int port);
Input     : int port      : port number 0~7
Output    : refer to return code list
```

VB

```
Declare Function syio_AbortRead Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

```
function syio_AbortRead (port: Longint): Longint; stdcall;
implementation
function syio_AbortRead; external ‘syncapi.dll’;
```

syio_AbortWrite

Description:

Abort the blocked syio_Write function call.

Syntax:

C/C++

```
int WINAPI syio_AbortWrite (int port);
Input     : int port               : port number 0~7
Output    : refer to return code list
```

VB

```
Declare Function syio_AbortWrite Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

```
function syio_AbortWrite (port: Longint): Longint; stdcall;
implementation
function syio_AbortWrite; external 'syncapi.dll';
```

syio_DTR

Description:

Set DTR pin on or off.

Syntax:

C/C++

```
Int WINAPI syio_DTR (int port, int mode);
Input    : int port          : port number 0~7
           int mode          : 0 for off, 1 for on
Output   : refer to return code list
```

VB

```
Declare Function syio_DTR Lib "syncapi.dll" (ByVal port As Long, ByVal mode As Long) As Long
```

Delphi

```
function syio_DTR (port, mode: Longint): Longint; stdcall;
implementation
function syio_DTR; external 'syncapi.dll';
```

syio_RTS

Description:

Set RTS pin on or off.

Syntax:

C/C++

```
int WINAPI syio_RTS (int port, int mode);
Input    : int port          : port number 0~7
           int mode          : 0 for off, 1 for on
Output   : refer to return code list
```

VB

Declare Function syio_RTS Lib "syncapi.dll" (ByVal port As Long, ByVal mode As Long) As Long

Delphi

```
function syio_RTS (port, mode: Longint): Longint; stdcall;
implementation
function syio_RTS; external 'syncapi.dll';
```

syio_SkipFrame

Description:

Skip first received frame on the buffer of the driver. The skipped frame will be aborted and not be read by the application.

Syntax:

C/C++

```
int WINAPI syio_SkipFrame (int port);
Input     : int port              : port number 0~7
Output    : refer to return code list
```

VB

Declare Function syio_SkipFrame Lib "syncapi.dll" (ByVal port As Long)

Delphi

```
function syio_SkipFrame (port: Longint): Longint; stdcall;
implementation
function syio_SkipFrame; external 'syncapi.dll';
```

syio_InFrame

Description:

Get the number of received frames on the buffer of the driver.

Syntax:

<u>C/C++</u>

```
int WINAPI syio_InFrame (int port);
Input     : int port      : port number 0~7
Output    : >=0           : received frames
             <0           : refer to return code list
```

<u>VB</u>

```
Declare Function syio_InFrame Lib “syncapi.dll” (ByVal port As Long) As Long
```

<u>Delphi</u>

```
function syio_InFrame (port: Longint): Longint; stdcall;
implementation
function syio_InFrame; external ‘syncapi.dll’;
```

syio_OutFrame

Description:

Get the number of to-be-send frames on the buffer of the driver.

Syntax:

<u>C/C++</u>

```
int WINAPI syio_OutFrame (int port);
Input     : int port      : port number 0~7
Output    : >=0           : to-send frames
             <0           : refer to return code list
```

<u>VB</u>

```
Declare Function syio_OutFrame Lib “syncapi.dll” (ByVal port As Long) As Long
```

<u>Delphi</u>

```
function syio_OutFrame (port: Longint): Longint; stdcall;
implementation
function syio_OutFrame; external ‘syncapi.dll’;
```

syio_InFreeFrame

Description:

Get the number of free input frames on the buffer of the driver.

Syntax:

C/C++

```
int WINAPI syio_InFreeFrameint port);
Input     : int port      : port number 0~7
Output    : >=0           : input free frames
            <0            : refer to return code list
```

VB

```
Declare Function syio_InFreeFrame Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

```
function syio_InFreeFrame (port: Longint): Longint; stdcall;
implementation
function syio_InFreeFrame; external ‘syncapi.dll’;
```

syio_OutFreeFrame

Description:

Get the number of free output frames on the buffer of the driver.

Syntax:

C/C++

```
int WINAPI syio_OutFreeFrame (int port);
Input     : int port      : port number 0~7
Output    : >=0           : output free frames
            <0            : refer to return code list
```

VB

```
Declare Function syio_OutFreeFrame Lib “syncapi.dll” (ByVal port As Long) As
Long
```

Delphi
```
function syio_OutFreeFrame (port: Longint): Longint; stdcall;
implementation
function syio_OutFreeFrame; external 'syncapi.dll';
```

syio_SetDataEncoding

Description:

NRZ and NRZI are supported for setting data encoding mode.

Syntax:

C/C++
```
int WINAPI syio_SetDataEncoding (int port, int mode);
Input    : int port     : port number 0~7
           int mode     : NRZ, NRZI, FM0 or FM1
Output   : refer to return code list
```
VB
```
Declare Function syio_SetDataEncoding Lib "syncapi.dll" (ByVal port As Long,
ByVal mode As Long) As Long
```

Delphi
```
function syio_SetDataEncoding(port, mode: Longint): Longint; stdcall;
implementation
function syio_SetDataEncoding; external 'syncapi.dll';
```

syio_GetDataEncoding

Description:

Get data encoding mode setting value.

Syntax:

C/C++
```
int WINAPI syio_GetDataEncoding (int port);
Input    : int port     : port number 0~7
Output   : >=0          : data encoding mode NRZ, NRZI, FM0 or FM1
           <0           : refer to return code list
```

VB

Declare Function syio_GetDataEncoding Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

function syio_GetDataEncoding(port: Longint): Longint; stdcall;
implementation
function syio_GetDataEncoding; external ‘syncapi.dll’;

syio_SetCRCMode

Description:

Set CRC mode. CCITT initialized 0, all 1’s, or none CRC are supported. HDLC protocol can only use CCITT CRC.

Syntax:

C/C++

```
int WINAPI syio_SetCRCMode (int port, int mode);
Input    : int port      : port number 0~7
Output   : >=0           : NONE,CCITT_00,CRC16_0 or CRC16_1
           <0            : refer to return code list
```

VB

Declare Function syio_SetCRCMode Lib “syncapi.dll” (ByVal port As Long, ByVal mode As Long) As Long

Delphi

function syio_SetCRCMode (port, mode: Longint): Longint; stdcall;
implementation
function syio_SetCRCMode; external ‘syncapi.dll’;

syio_GetCRCMode

Description:

Get CRC mode setting value.

Syntax:

C/C++

```
int WINAPI syio_GetCRCMode (int port);
Input     : int port      : port number 0~7
Output    : >=0           : CRC value setting
            <0            : refer to return code list
```

VB

```
Declare Function syio_GetCRCMode Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi

```
function syio_GetCRCMode (port: Longint): Longint; stdcall;
implementation
function syio_GetCRCMode; external 'syncapi.dll';
```

syio_LineStatus

Description:

Get line status. Then, the firmware will poll line status every 50ms.

Syntax:

C/C++

```
int WINAPI syio_LineStatus (int port);
Input     : int port      : port number 0~7
Output    : >=0           : the line status-bit 0 for DCD, bit 1 for CTS, bit
                            on(1) for status pin on, bit off(0) for tatus pin off
            <0            : refer to return code list
```

VB

```
Declare Function syio_LineStatus Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi

```
function syio_LineStatus (port: Longint): Longint; stdcall;
implementation
function syio_LineStatus; external 'syncapi.dll';
```

syio_InQueue

Description:

Get the received data bytes on the buffer of the driver.

Syntax:

<u>C/C++</u>

```
int WINAPI syio_InQueue (int port);
Input    : int port     : port number 0~7
Output   : >=0          : received bytes
           <0           : refer to return code list
```

<u>VB</u>

```
Declare Function syio_InQueue Lib “syncapi.dll” (ByVal port As Long) As Long
```

<u>Delphi</u>

```
function syio_InQueue (port: Longint): Longint; stdcall;
implementation
function syio_InQueue; external ‘syncapi.dll’;
```

syio_OutQueue

Description:

Get to-be-sent data bytes on the buffer of the driver.

Syntax:

<u>C/C++</u>

```
int WINAPI syio_OutFrame (int port);
Input    : int port     : port number 0~7
Output   : >=0          : to-be-sent bytes
           <0           : refer to return code list
```

<u>VB</u>

```
Declare Function syio_OutQueue Lib “syncapi.dll” (ByVal port As Long) As Long
```

<u>Delphi</u>

```
function syio_OutQueue (port: Longint): Longint; stdcall;
implementation
function syio_OutQueue; external ‘syncapi.dll’;
```

syio_InFree

Description:

Get free data bytes space on the buffer of the driver.

Syntax:

C/C++
```
int WINAPI syio_InFree (int port);
Input     : int port     : port number 0~7
Output    : >=0          : input free bytes
            <0           : refer to return code list
```

VB
```
Declare Function syio_InFree Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi
```
function syio_InFree (port: Longint): Longint; stdcall;
implementation
function syio_InFree; external 'syncapi.dll';
```

syio_OutFree

Description:

Get free output data bytes space on the buffer of the driver.

Syntax:

C/C++
```
int WINAPI syio_OutFrame (int port);
Input     : int port     : port number 0~7
Output    : >=0          : output free bytes
            <0           : refer to return code list
```

VB
```
Declare Function syio_OutFree Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi
```
function syio_OutFree (port: Longint): Longint; stdcall;
implementation
function syio_OutFree; external 'syncapi.dll';
```

syio_FrameIrq

Description:

Set the event 'number of received frame'. You can specify a function to be called when frame event happens. If the function is set as "NULL", frame event will be cleared.

Syntax:

C/C++

int WINAPI syio_FrameIrq (int port, VOID (CALLBACK * func)(int port),int framecnt);

Input	: int port	: port number 0~7
	VOID (CALLBACK*func)(int port)	: the function to be called when this event happens
	Int framecnt	: Number of received frames to call the function. It must be greater greater than zero
Output	: refer to return code list	

VB

Declare Function syio_FrameIrq Lib "syncapi.dll" (ByVal port As Long, ByVal func As Long, ByVal framecnt As Long) As Long

Delphi

```
Type
IrqProc1 = procedure (port: Longint); stdcall;
function syio_FrameIrq (port: Longint; func: IrqProc1; framecnt: Longint): Longint; stdcall;
implementation
function syio_FrameIrq; external 'syncapi.dll';
```

syio_ModemIrq

Description:

Set the event 'modem status change '. You can specify a function to be called when modem CTS, DCD, DSR on/off status changes. If the function is set NULL, modem event will be cleared.

Syntax:

C/C++

int WINAPI syio_ModemIrq (int port, VOID(CALLBACK * func)(int port, int status), int mode);

Input	: int port	: port number 0~7
	VOID (CALLBACK*func)(int port,int status)	: the function to be called when this event happens
	Int mode	: Types of modem status change At last one modem status has to be set.
Output	: refer to return code list	

VB

Declare Function syio_ModemIrq Lib "syncapi.dll" (ByVal port As Long, ByVal func As Long, ByVal mode As Long) As Long

Delphi

type
IrqProc2 = procedure (port, status: Longint); stdcall;
function syio_ModemIrq (port: Longint; func: IrqProc2; mode: Longint): Longint; stdcall;
implementation
function syio_ModemIrq; external 'syncapi.dll';

syio_TxEmptyIrq

Description:

Set the event 'Tx Empty'. You can specify a function to be called when Tx Empty event happens. If the function is set NULL, Tx Empty event will be cleared.

Syntax:

C/C++

int WINAPI syio_TxEmptyIrq (int port, VOID(CALLBACK * func)(int port);

Input	: int port	: port number 0~7
	VOID (CALLBACK*func)(int port)	: the function to be called when this event happens
Output	: refer to return code list	

VB

Declare Function syio_TxEmptyIrq Lib “syncapi.dll” (ByVal port As Long, ByVal func As Long) As Long

Delphi

```
type
IrqProc1 = procedure (port: Longint); stdcall;
function syio_TxEmptyIrq (port: Longint; func: IrqProc1): Longint; stdcall;
implementation
function syio_TxEmptyIrq; external ‘syncapi.dll’;
```

syio_SetTxClockDir

Description:

Set Tx clock direction ‘in’ or ‘out’. Tx clock ‘in’ uses different pin on connector from clock ‘out’.

Syntax:

C/C++

```
int WINAPI syio_SetTxClockDir (int port, int direction);
Input     : int port       : port number 0~7
          : int direction  : IN or OUT
Output    : refer to return code list
```

VB

Declare Function syio_SetTxClockDir Lib “syncapi.dll” (ByVal port As Long, ByVal direction As Long) As Long

Delphi

```
function syio_SetTxClockDir (port, direction: Longint): Longint; stdcall;
implementation
function syio_SetTxClockDir; external ‘syncapi.dll’;
```

syio_GetTxClockDir

Description:

Get Tx clock direction setting value.

Syntax:

C/C++

```
int WINAPI syio_GetTxClockDir (int port);
Input      : int port      : port number 0~7
Output     : >=0           : clock direction
             <0            : refer to return code list
```

VB

Declare Function syio_GetTxClockDir Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

```
function syio_GetTxClockDir (port: Longint): Longint; stdcall;
implementation
function syio_GetTxClockDir; external ‘syncapi.dll’;
```

syio_TxDisable

Description:

Disable Tx transmission.

Syntax:

C/C++

```
int WINAPI syio_TxDisable (int port);
Input      : int port      : port number 0~7
Output     : refer to return code list
```

VB

Declare Function syio_TxDisable Lib “syncapi.dll” (ByVal port As Long) As Long

Delphi

```
function syio_TxDisable (port: Longint): Longint; stdcall;
implementation
function syio_TxDisable; external ‘syncapi.dll’;
```

syio_TxEnable

Description:

Enable transmission halted by syio_TxDisable.

Syntax:

C/C++
```
int WINAPI syio_TxEnable (int port);
Input     : int port       : port number 0~7
Output    : refer to return codelist
```

VB
```
Declare Function syio_TxEnable Lib "syncapi.dll" (ByVal port As Long) As Long
```
Delphi
```
function syio_TxEnable (port: Longint): Longint; stdcall;
implementation
function syio_TxEnable; external 'syncapi.dll';
```

syio_TxStatus

Description:

Get Tx status, 'disable' or 'enable'.

Syntax:

C/C++
```
int WINAPI syio_TxStatus (int port);
Input     : int port       : port number 0~7
Output    : >=0            : Tx status, 0 for disable, 1 for enable
             <0            : refer to return code list
```

VB
```
Declare Function syio_TxStatus Lib "syncapi.dll" (ByVal port As Long) As Long
```

Delphi
```
function syio_TxStatus (port: Longint): Longint; stdcall;
implementation
function syio_TxStatus; external 'syncapi.dll';
```

syio_GetFirstFrameLen

Description:

Get first received frame length.

Syntax:

C/C++

```
int WINAPI syio_GetFirstFrameLen (int port);
Input:    : Int port    : port number 0~7
Output    : >=0         : the first frame length
            <0          : refer to return code list
```

VB

```
Declare Function syio_GetFirstFrameLen Lib “syncapi.dll” (ByVal port As Long) As
Long
```

Delphi

```
function syio_getFirstFrameLen (port: Longint): Longint; stdcall;
implementation
function syio_GetFirstFrameLen; external ‘syncapi.dll’;
```

syio_GetBoardID

Description:

Get board ID number. Default number is 1. Other ID numbers are available for OEM user.

Syntax:

C/C++

```
int WINAPI syio_GetBoardID (int port);
Input     : int port    : port number 0~7
Output    : >=0         : 1 only
            <0          : refer to return code list
```

VB

```
Declare Function syio_GetBoardID Lib “syncapi.dll” (ByVal port As Long) As Long
```

Delphi

```
function syio_GetBoardID(port: Longint): Longint; stdcall;
implementation
function syio_GetBoardID; external 'syncapi.dll';
```

syio_SetSyncChar

Description:

Set synchronous character pattern for transmission and reception in byte synchronous mode.

Syntax:

C/C++

```
int WINAPI syio_SetSyncChar (int port, USHORT syncchar);
Input    : int port              : port number 0~7
           USHORT syncchar       : the synchronous character, use low byte for mono-sync,
                                   two bytes for bi-sync
Output   : Refer to return code list
```

VB

```
Declare Function syio_SetSyncChar Lib "syncapi.dll"(ByVal port As Long, ByVal
syncchar As Integer) As Long
```

Delphi

```
function syio_SetSyncChar (port:Longint; syncchar:Word): Longint; stdcall;
implementation
function syio_SetSyncChar; external 'syncapi.dll';
```

syio_SetSyncLength

Description:

Set synchronous character pattern number for transmission and reception in byte synchronous mode.

Syntax:

C/C++
```
int WINAPI syio_SetSyncLength (int port, int length);
Input     : int port              : port number 0~7
          : int length            : pattern number, max 255
Output    : Refer to return code list
```

VB
```
Declare Function syio_SetSyncLength Lib "syncapi.dll"(ByVal port As Long,
ByVal length As Long) Ad Long
```

Delphi
```
function syio_SetSyncLength (port, length: Longint): Longint; stdcall;
implementation
function syio_SetSyncLength; external 'syncapi.dll';
```

Syio_SetIdleCode

Description:

Set the idle pattern output by the transmitter when it is in idle state.

Syntax:

C/C++
```
int WINAPI syio_SetIdleCode (int port, UCHAR idlecode);
Input     : Int port              : port number 0~7
            UCHAR idlecode        : idle pattern
Output    : Refer to return code list
```

VB
```
Declare Function syio_SetIdleCode Lib "syncapi.dll"(ByVal port As Long, ByVal
idlecode As Byte) As Long
```

Delphi
```
function syio_SetIdleCode (port:Longint; idlecode:Byte): Longint; stdcall;
implementation
function syio_SetIdleCode; external 'syncapi.dll';
```

syio_GetOpMode

Description:

Get the synchronous mode

Syntax:

C/C++
```
int WINAPI syio_GetOpMode (int port);
Input        : int port     : port number 0~7
Output:      : >=0          : the synchronous mode, 0 for HDLC, 1 for mono-sync, 2 for
                              bi-sync
               <0           : refer to return code list
```

VB
```
Declare Function syio_GetOpMode Lib “syncapi.dll”(ByVal port As Long) As Long
```

Delphi
```
function syio_GetOpMode (port:Longint): Longint; stdcall;
implementation
function syio_GetOpMode; external ‘syncapi.dll’;
```

syio_GetSyncChar

Description:

Get the synchronous character pattern in byte synchronous mode.

Syntax:

C/C++
```
int WINAPI syio_GetOpMode (int port);
Input        : Int port     : port number 0~7
Output       : >=0          : the synchronous character, use low byte for mono-sync,
                              two bytes for bi-sync
             : <0           : refer to return code list
```

VB
```
Declare Function syio_GetSyncChar Lib “syncapi.dll” (ByVay port As Long) AsLong
```

Delphi

function syio_GetSyncChar (port:Longint): Longint; stdcall;
implementation
function syio_GetSyncChar; external 'syncapi.dll';

syio_GetSyncLength

Description:

Get the synchronous character pattern number in byte synchronous mode.

Syntax:

C/C++

```
int WINAPI syio_GetSyncLength (int port);
Input       : int port    : port number 0~7
Output      : >=0         : the synchronous character pattern number
               <0         : refer to return code list
```

VB

Declare Function syio_GetSyncLength Lib "syncapi.dll"(ByVal port As Long) As Long

Delphi

function syio_GetSyncLength (port:Longint): Longint; stdcall;
implementation
funciton syio_getSyncLength; external 'syncapi.dll';

syio_GetIdleCode

Description:

Get the idle pattern when it is in idle state.

Syntax:

C/C++

```
int WINAPI syio_GetIdleCode (int port);
Input       : int port    : port number 0~7
Output      : >=0         : the idle pattern
            : <0          : refer to return code list
```

VB

```
Declare Function syio_GetIdleCode Lib “syncapi.dll”(ByVal port As Long) As Long
```

Delphi

```
function syio_GetIdleCode (port:Longint): Longint; stdcall;
implementation
function syio_GetIdleCode; external ‘syncapi.dll’;
```

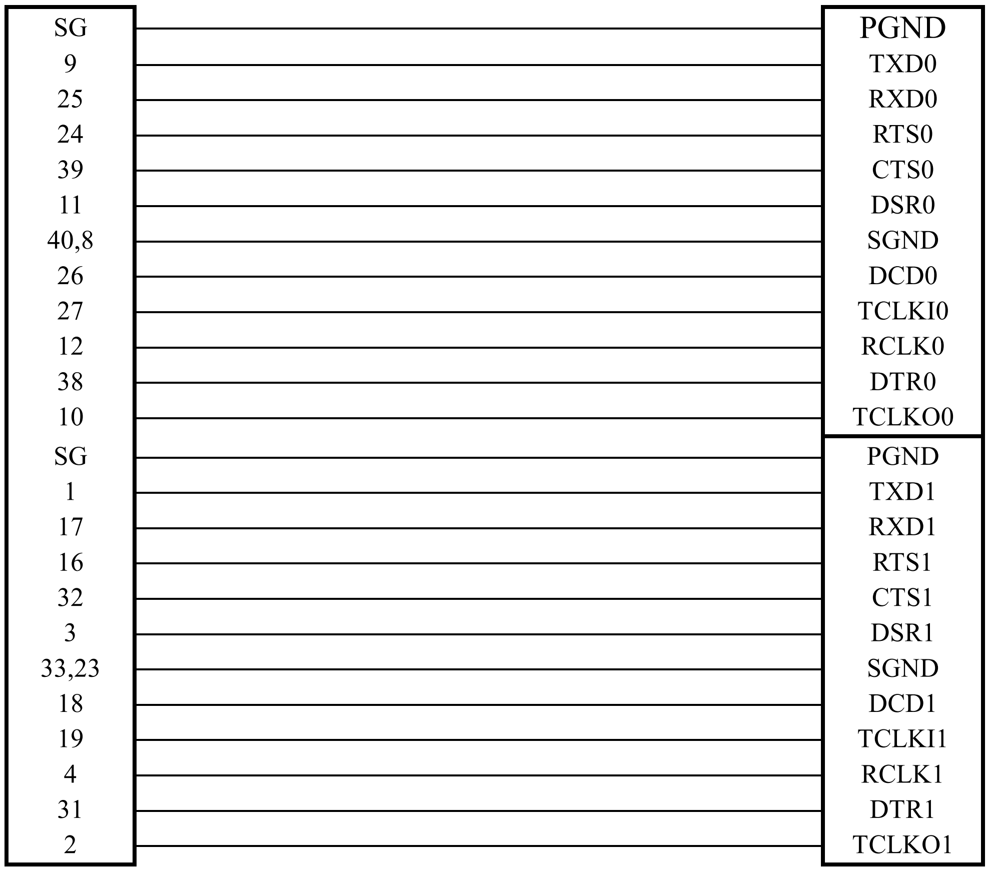
A

RS-232/V.24 Connection

RS-232/V.24 Pin Assignment for MOXA C502

Pin #	Signal	Name	Direction	CCITT #
2	TXD	Transmit Data	Output	103
3	RXD	Receive Data	Input	104
4	RTS	Request to Send	Output	105
5	CTS	Clear to Send	Input	106
6	DSR	Data Set Ready	Input	107
7	SGND	Signal Ground	Common	102
8	DCD	Data Carrier Detect	Input	109
15	TCLKI	Transmit Clock	Input	114
17	RCLK	Receive Clock	Input	115
20	DTR	Data Terminal Ready	Output	108
24	TCLKO	Transmit Clock	Output	113

RS-232 Dual-Port DB44 Cable Connections (Port 0 and Port 1)



SG	PGND
9	TXD0
25	RXD0
24	RTS0
39	CTS0
11	DSR0
40,8	SGND
26	DCD0
27	TCLKI0
12	RCLK0
38	DTR0
10	TCLKO0
SG	PGND
1	TXD1
17	RXD1
16	RTS1
32	CTS1
3	DSR1
33,23	SGND
18	DCD1
19	TCLKI1
4	RCLK1
31	DTR1
2	TCLKO1

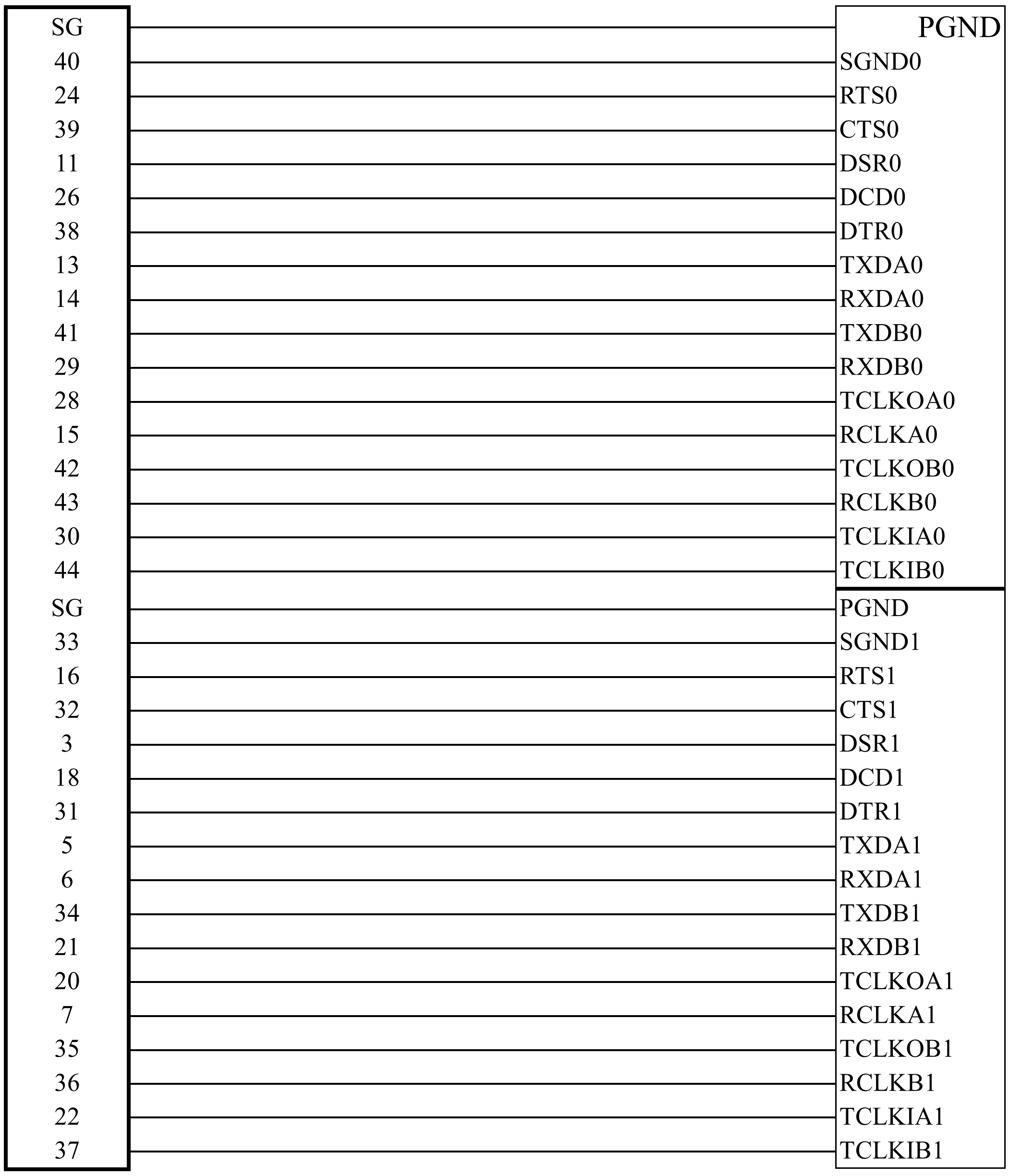
B

V3.5 Connection

V.35 Pin Assignment for MOXA C502

Pin #	Signal	Name	Direction	CCITT #
A	PGND	Protective Ground	Common	101
B	SGND	Signal Ground	Common	102
C	RTS	Request to Send	Output	105
D	CTS	Clear to Send	Input	106
E	DSR	Data Set Ready	Input	107
F	DCD	Data Carrier Detect	Input	109
H	DTR	Data Terminal Ready	Output	108
P	TXDA	Transmit Data	Output	103A
R	RXDA	Receive Data	Input	104A
S	TXDB	Transmit Data	Output	103B
T	RXDB	Receive Data	Input	104B
U	TCLKOA	Transmit Clock(DTE)	Output	113A
V	RCLKA	Receive Clock(DCE)	Input	115A
W	TCLKOB	Transmit Clock(DTE)	Output	113B
X	RCLKB	Receive Clock(DCE)	Input	115B
Y	TCLKIA	Transmit Clock(DCE)	Input	114A
AA	TCLKIB	Transmit Clock(DCE)	Input	114B

V.35 Dual-Port DB44 Cable Connections (Port 0 and Port 1)



SG	PGND
40	SGND0
24	RTS0
39	CTS0
11	DSR0
26	DCD0
38	DTR0
13	TXDA0
14	RXDA0
41	TXDB0
29	RXDB0
28	TCLKOA0
15	RCLKA0
42	TCLKOB0
43	RCLKB0
30	TCLKIA0
44	TCLKIB0
SG	PGND
33	SGND1
16	RTS1
32	CTS1
3	DSR1
18	DCD1
31	DTR1
5	TXDA1
6	RXDA1
34	TXDB1
21	RXDB1
20	TCLKOA1
7	RCLKA1
35	TCLKOB1
36	RCLKB1
22	TCLKIA1
37	TCLKIB1

C

Trouble Shotting

1. Download BIOS or firmware file fails
Possible problem types and solutions for ISA boards:

a) C502/ISA base address conflicts with the BIOS ROM Shadow. Disable the BIOS ROM Shadow C502/ISA uses. For example, if you set C502/ISA to base address C8000 (or C800:0000), then C800:0000 ROM Shadow must be disabled.

b) C502/ISA base address conflicts with that of other interface cards such as SCSI or LAN cards. Adjust the address to forestall the conflict.

c) C502/ISA is not properly plugged-in a 16-bit slot. Reinstall C502/ISA and make sure it fits well this time.

d) C502/ISA does not function well. Kindly return for repair.

Possible problems types and solutions for PCI boards:

a) The C502/PCI board is unplugged into the main board.

b) Slot replacement of hardware and software configuration should be matched each other. Whenever you want to replace another slot for the hardware, the configuration for software should be set again.

2. C502 driver initializes OK but can not transfer any data.
Check if wrong cable wiring. Refer to Appendix for precise pin assignment of communication port and its cable wiring. To be sure the transmit clock direction is OK.

Problem Report Form

C502 Dual-Port Sync Board

Customer name:	
Company:	
Tel:	Fax:
Email:	Date:

1. **Moxa Product**: ☐ C502-ISA/RS232 ☐ C502-ISA/V.35
2. **Serial Number**: ____________(Please see the rear panel of the board)
3. **Driver Version**: __________________
4. **Problem Description**: Please describe the symptom as clear as possible including the return message you see. We may have to follow your description to reproduce the symptom.

RETURN PROCEDURE

For product repair, exchange or refund, the customer must:

- Provide evidence of original purchase.
- Obtain a Product Return Agreement (PRA) from the sales representative or dealer.
- Fill out the Problem Report Form (PRF) as detailed as possible for shorter product repair time.
- Carefully pack the product in anti-static package, and send it, pre-paid, to the dealer. The PRA should show on the outside of the package, and include a description of the problem along with the return address and telephone number of a technical contact.