Installation Guide

NXC-COM2 Dual COM Port Card, 2 RS-232/422/485

NetLinx Control Cards and NetModules

NetLinx Control Cards can be installed in either the NXF CardFrame, NI-4000, or NetModules. For detailed information on the cards, refer to the *NetLinx CardFrame, Control Cards, and NetModules* Instruction Manual available on-line via www.amx.com.

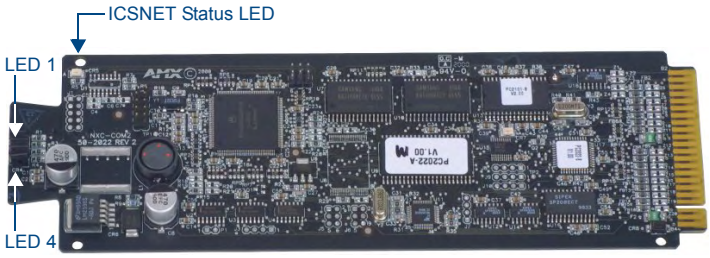

FIG. 1 NXC-COM2 Dual COM Port Control card

NXC-COM2 Specifications

The NXC-COM2 Dual COM Port Control Card provides two RS-232/422/485 control ports and LED feedback for remote sources connected to the NetLinx CardFrame, NI-4000, or NetModule:

NXC-COM2 (FG 2022) Specifications	
Power Requirements:	140 mA @ 12 VDC
Operation:	Data 1-2: Two RS-232/422/485 control ports, supports XON/XOFF, CTS/RTS
Status LEDs (2 per channel):	• Red LEDs show TX (transmit) data activity • Yellow LEDs show RX (receive) data activity - LED 1: CH1 TX (red) - LED 2: CH1 RX (yellow) - LED 3: CH2 TX (red) - LED 4: CH2 RX (yellow) • Green ICSP status LED (located on the board): - On = card is not in communication with the Master. - Blinks (1-second intervals) during normal operation.
Connections/wiring:	Two 10-pin 3.5 mm captive-screw terminals
DEVICE_ID:	$0107

NXC-COM2 Pinouts and Wiring Information:

NXC-COM2 Pinouts			NXC-COM2 Wiring Configuration		
Pin	**Signal**	**Function**	**RS-232**	**RS-422**	**RS-485**
1	GND	Signal ground	X	X	
2	RXD1	Receive data	X		
3	TXD1	Transmit data	X		
4	CTS1	Clear to send	X		
5	RTS1	Request to send	X		
6	TX1+	Transmit data		X	X (strap to pin 8)
7	TX1-	Transmit data		X	X (strap to pin 9)
8	RX1+	Receive data		X	X (strap to pin 6)
9	RX1-	Receive data		X	X (strap to pin 7)
10	+12 VDC	Power	Optional	Optional	
11	GND	Signal ground	X	X	
12	RXD2	Receive data	X		
13	TXD2	Transmit data	X		
14	CTS2	Clear to send	X		
15	RTS2	Request to send	X		
16	TX2+	Transmit data		X	X (strap to pin 18)
17	TX2-	Transmit data		X	X (strap to pin 19)
18	RX2+	Receive data		X	X (strap to pin 16)
19	RX2-	Receive data		X	X (strap to pin 17)
20	+12 VDC	Power	Optional	Optional	

NXC-COM2 Channel Assignment:

Channel 255 is the CTS (Clear To Send) push channel. It reflects the state of the CTS input if a 'CTSPSH' command was sent to the device.

NXC-COM2 Programming Information:

NXC-COM2 Send_Commands	

B9MOFF
Disable 9-bit in 232/422/455 mode.
This command works in conjunction with the B9MON command.
	Syntax:
	`SEND_COMMAND <DEV>,"'B9MOFF'"`
	Example:
	`SEND_COMMAND SOMEDEVICE_1,"'B9MOFF'"`
	Sets the port settings on SOMEDEVICE to match the port's configuration settings.

B9MON
Enable 9-bit in 232/422/455 mode.
This command works in conjunction with the B9MOFF command.
	Syntax:
	`SEND_COMMAND <DEV>,"'B9MON'"`
	Example:
	`SEND_COMMAND SOMEDEVICE_1,"'B9MON'"`
	Resets the SOMEDEVICE port's communication parameters to nine data bits and one stop bit.

CHARD
Set the delay time between all transmitted characters to the value specified (in 100 micro-second increments).
	Syntax:
	`SEND_COMMAND <DEV>,"'CHARD-<Time>'"`
	time: 0 - 255. Measured in 100 microsecond increments.
	Example:
	`SEND_COMMAND RS232_1,"'CHARD-10'"`
	Sets a 1-millisecond delay between all transmitted characters.

CHARDM
Set the delay time between all transmitted characters to the value specified (in 1 millisecond increments).
	Syntax:
	`SEND_COMMAND <DEV>,"'CHARDM-<Time>'"`
	time: 0 - 255. Measured in 1 millisecond increments.
	Example:
	`SEND_COMMAND RS232_1,"'CHARDM10'"`
	Sets a 10-millisecond delay between all transmitted characters.

CTSPSH
Enable Pushes, Releases, and status information to be reported via channel 255 using the CTS hardware handshake input.
	If Clear To Send (CTS) is set high, then channel 255 is On.
	Syntax:
	`SEND_COMMAND <DEV>,"'CTSPSH'"`
	Example:
	`SEND_COMMAND RS232_1,"'CTSPSH'"`
	Sets the RS232_1 port to detect changes on the CTS input.

CTSPSHOFF
Disable Pushes, Releases, and Status information to be reported via channel 255.
	Syntax:
	`SEND_COMMAND <DEV>,"'CTPSH OFF'"`
	Example:
	`SEND_COMMAND RS232_1,"'CTSPSH OFF'"`
	Disables the RS232_1 port to detect changes on the CTS input.

GET BAUD
Get the RS-232/422/485 port's current communication parame-ters.
Device sends the response out the Master program port.
	Syntax:
	`SEND_COMMAND <DEV>,"'GET BAUD'"`
	Example:
	`SEND_COMMAND RS232_1,"'GET BAUD'"`
	Device responds with:
	`<port #>,<baud>,<parity>,<data>,<stop>`
	`485 <ENABLED | DISABLED>`

HSOFF
Disable hardware handshaking (default).
	Syntax:
	`SEND_COMMAND <DEV>,"'HSOFF'"`
	Example:
	`SEND_COMMAND RS232_1,"'HSOFF'"`
	Disables hardware handshaking on the RS232_1 device.

HSON
Enable RTS (ready-to-send) and CTS (clear-to-send) hardware handshaking.
	Syntax:
	`SEND_COMMAND <DEV>,"'HSON'"`
	Example:
	`SEND_COMMAND RS232_1,"'HSON'`
	Enables hardware handshaking on the RS232_1 device.

RXCLR
Clear all characters in the receive buffer waiting to be sent to the Master.
	Syntax:
	`SEND_COMMAND <DEV>,"'RXCLR'"`
	Example:
	`SEND_COMMAND RS232_1,"'RXCLR'"`
	Clears all characters in the RS232_1 device's receive buffer.

RXOFF
Disable the transmission of incoming received characters to the Master (default).
	Syntax:
	`SEND_COMMAND <DEV>,"'RXOFF'"`
	Example:
	`SEND_COMMAND RS232_1,"'RXOFF'"`
	Stops the RS232_1 device from transmitting received characters to the Master.

NXC-COM2 Send_Commands (Cont.)

RXON
Start transmitting received characters to the Master (default).
	This command is sent automatically when a CREATE_BUFFER Send_Command is executed.
	Syntax:
	`SEND_COMMAND <DEV>,"'RXON'"`
	Example:
	`SEND_COMMAND RS232_1,'RXON'"`
	Stops the RS232_1 device from transmitting received characters to the Master.

SET BAUD
Set the RS-232/422/485 port's communication parameters.
	Syntax:
	`SEND_COMMAND <DEV>,"'SET BAUD <baud>,<parity>,<data>,<stop> [485 <Enable | Disable>]'"`
	• Baud: 230400, 115200, 76800, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 150
	• Parity: N (none), O (odd), E (even), M (mark), S (space)
	• Data Bits: 7 or 8 data bits
	• Stop Bits: 1 and 2 stop bits
	The only valid 9 bit combination is (baud),N,9,1.
	• 485 Enable: Enables RS-485 mode and disables RS-232/422
	• 485 Disable: Disables RS-485 mode and enables RS-232/422
	Examples:
	`SEND_COMMAND RS232_1,"'SET BAUD 115200,N,8,1 485 ENABLE'"`
	Sets the RS232_1 port's communication parameters to 115,200 baud, no parity, 8 data bits, 1 stop bit, and enables RS-485 mode.

TSET BAUD
Temporarily set the RS-232/422/485 port's communication parameters for a device.
	Syntax:
	`SEND_COMMAND <DEV>,"'TSET BAUD <baud>,<parity>,<data>,<stop> [485 <Enable | Disable>]'"`
	• Baud: 230400, 115200, 76800, 57600, 38400, 19200, 9600, 4800, 2400, 1200, 600, 300, 150
	• Parity: N (none), O (odd), E (even), M (mark), S (space)
	• Data Bits: 7 or 8 data bits
	• Stop Bits: 1 and 2 stop bits
	The only valid 9 bit combination is (baud),N,9,1.
	• 485 Enable: Enables RS-485 mode and disables RS-232/422
	• 485 Disable: Disables RS-485 mode and enables RS-232/422
	Examples:
	`SEND_COMMAND RS232_1,"'SET BAUD 9600,N,8,1 485 DISABLE'"`
	`SEND_COMMAND RS232_4,"'SET BAUD 115200,N,8,1 485 ENABLE'"`
	TSET BAUD works the same as SET BAUD, except that the changes are not permanent, and the previous values will be restored if the power is cycled on the device.

TXCLR
Stop and clear all characters waiting in the transmit out buffer and stops transmission.
	Syntax:
	`SEND_COMMAND <DEV>,"'TXCLR'"`
	Example:
	`SEND_COMMAND RS232_1,"'TXCLR'"`
	Clears and stops all characters waiting in the RS232_1 device's transmit buffer.

XOFF
Disable software handshaking (default).
	Syntax:
	`SEND_COMMAND <DEV>,"'XOFF'"`
	Example:
	`SEND_COMMAND RS232_1,"'XOFF'"`
	Disables software handshaking on the RS232_1 device.

XON
Enable software handshaking.
	Syntax:
	`SEND_COMMAND <DEV>,"'XON'"`
	Example:
	`SEND_COMMAND RS232_1,"'XON'"`
	Enables software handshaking on the RS232_1 device.

NXC-COM2 Send_String Escape Sequences

This card has some special SEND_STRING escape sequences:
If any of the 3 character combinations below are found anywhere within a SEND_STRING
program instruction, they will be treated as a command and not the literal characters:

27,17,<time>
Send a break character for a specified duration to a specific device.
	Syntax:
	`SEND_STRING <DEV>,"27,17,<time>"`
	time: 1 - 255. Measured in 100 microsecond increments.
	Example:
	`SEND_STRING RS232_1,"27,17,10"`
	Sends a break character of 1 millisecond to the RS232_1 device.

27,18,0
Clear the ninth data bit by setting it to 0 on all character transmissions. This command is used in conjunction with the B9MON Send_Command.
	Syntax:
	`SEND_STRING <DEV>,"27,18,0"`
	Example:
	`SEND_STRING RS232_1,"27,18,0"`
	Sets the RS232_1 device's ninth data bit to 0 on all character transmissions.

27,18,1
Set the ninth data bit to 1 for all subsequent characters to be transmitted. This command is used in conjunction with the B9MON Send_Command.
	Syntax:
	`SEND_STRING <DEV>,"27,18,1"`
	Example:
	`SEND_STRING RS232_1,"27,18,1"`
	Sets the RS232_1 device's ninth data bit to 1 on all character transmissions.

27,19,<time>
Insert a time delay before transmitting the next character.
	Syntax:
	`SEND_STRING <DEV>,"27,19,<time>"`
	time: 1 - 255. Measured in 1 millisecond increments.
	Example:
	`SEND_STRING RS232_1,"27,19,10"`
	Inserts a 10 millisecond delay before transmitting characters to the RS232_1 device.

27,20,0
Set the RTS hardware handshake's output to high (> 3V).
	Syntax:
	`SEND_STRING <DEV>,"27,20,0"`
	Example:
	`SEND_STRING RS232_1,"27,20,0"`
	Sets the RTS hardware handshake's output to high on the RS232_1 device.

27,20,1
Assert sets the RTS hardware handshake's output to low (< 3V).
	Syntax:
	`SEND_STRING <DEV>,"27,20,1"`
	Example:
	`SEND_STRING RS232_1,"27,20,1"`
	Sets the RTS hardware handshake's output to low on the RS232_1 device.

3000 RESEARCH DRIVE, RICHARDSON, TX 75082 • 800.222.0193 • fax 469.624.7153 • technical support 800.932.6993 • www.amx.com