instruction manual

AXC-232++

RS-232/422/485 Interface Card

Axcess Control Cards

Table of Contents

Introduction

Many audiovisual and computer-based devices can be remotely controlled using RS-232, RS-422, or RS-485 commands. The AXC-232++ RS-232/422/485 Interface Card can generate virtually unlimited numbers of product/function commands as they are programmed into the AXC-EM Master Card. As needs and technology change, existing cards can be reprogrammed or new cards added, one for each device.

One AXC-232++ card (FIG. 1) is required for each RS-232/422/485 controlled piece of equipment. The AXC-232++ card has a DIP switch to set communications parameters (data bits, stop bits, parity, and baud rates from 300 to 115,200 bps).

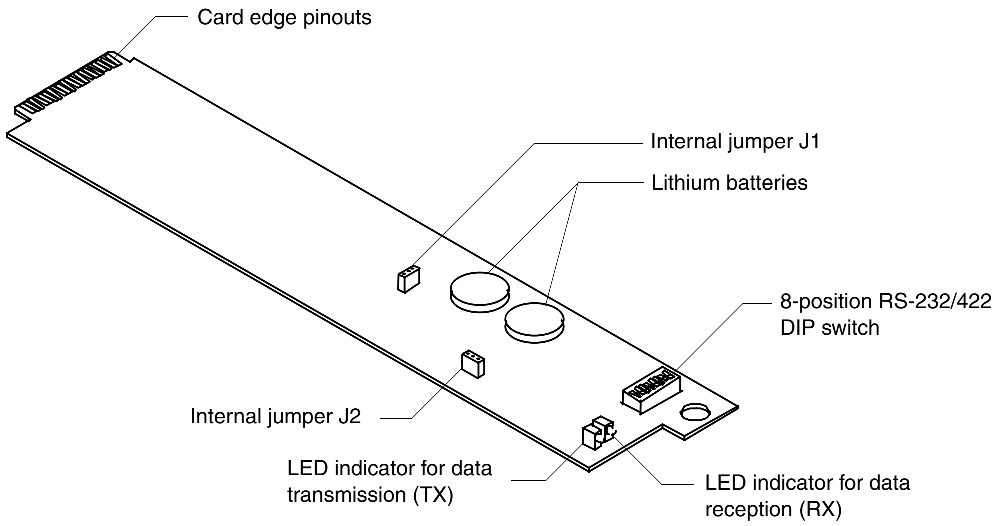

Card edge pinouts
Internal jumper J1
Lithium batteries
8-position RS-232/422 DIP switch
Internal jumper J2
LED indicator for data transmission (TX)
LED indicator for data reception (RX)

FIG. 1 AXC-232++

Onboard processing and memory allows the controller to take on complex tasks by itself, reducing the processing burden for the AXlink master program. For example, the master program can use simplified commands for generic video switcher or codec control, while a modular driver program in the AXC-232++ handles control for specific makes and models.

Specifications

The table below lists the AXC-232++ specifications.

Specifications	
Dimensions	2.7" (69.3 mm) height 12.8" (325.6 mm) depth
Power requirement	12 VDC
Power consumption	160 mA
Weight	4.8 oz (136.1 g)
Mounting	Inserts into an Axcess CardFrame

Specifications (Cont.)	
LEDs:	
RX/TX:	Red LED indicators for data transmission
RX:	Red LED blinks to indicate that the AXC-232++ is receiving RS-232, RS-422, or RS-485 data.
TX:	Red LED blinks to indicate that the AXC-232++ is sending RS-232, RS-422, or RS-485 data.
DIP Switch	8-position RS-232/422DIP switch. See *Setting the RS-232/422 DIP Switch* section on page 4 for configuration details.
Asynchronous data standards	• Baud rates - *300*, *600*, *1200*. *2400*, *4800*, *9600*, *19200* and *38400*. *57600* is supported via the BAUDMED Send Command (see *Send_Commands* section on page 9 for details). *115200* is supported via the BAUDHIGH Send Command (see the *Send_Commands* section on page 9 for details). • Data bits - 7, 8, and 9 • Stop bits - 1 and 2 • Parity - *None*, *Odd*, *Even*, *Mark* and *Space*
Connectors:	
• RS-232	Hardware handshaking (RTS/CTS) data control jack (3-pin male)
• RS-422	RS-232/422 data control jack (4-pin male)
• +12VDC	Auxiliary power jack (1-pin male)
• RS-232	RS-232 data control jack (3-pin male)
• AXlink	AXlink bus jack (4-pin male)
Buffer	• 1000 byte input buffer • 1000 byte AXlink buffer
Internal jumpers	Sets differential input termination to enable RS-485 output.
Processor	32-bit processor
Memory	384K of non-volatile memory (memory runs Axcess programs independent of the master program. This relieves the AXlink bus and the master of the processing time for controlling those devices; requires Axcess version 3.0 or higher).

Installation and Wiring

Setting the Internal Jumpers

The AXC-232++ can be used as an independent RS-232/422/485-controlled interface by setting the internal jumpers (JP1 and JP2). FIG. 1 indicates the location of the jumpers on the card.

CAUTION

Static electricity can damage electronic components. Before opening the device, discharge any accumulated static electricity from your body by touching a grounded metal object.

Setting jumper JP1 to terminate RS-422 input with 100 ohms

Terminating a device involves installing a 100 ohm line terminator. This is typically used to achieve better communication and signal integrity. You will want to terminate when the communication is at a high data rate or over a long distance. Termination can be harmful because it increases the current in the line, and causes more radiation that could interfere with signals.

Jumper JP1 places 100 Ohms termination across RS422 receive data pins 5 & 6:

1. Disconnect the RS-232/422/485 connectors.
2. Slide the card out of the cardframe.
3. Locate the JP1 jumper (see FIG. 1 on page 1).
4. Install the jumper in the 'ON' position (default setting = OFF).
5. Slide the card back into the cardframe.
6. Reconnect the RS-232/422/485 connectors.

Setting jumper JP2 to set the RS-422 port for RS-485 use

1. Disconnect the RS-232/422/485 connectors.
2. Slide the card out of the cardframe.
3. Locate the JP2 jumper (see FIG. 1 on page 1).
4. Set jumper JP2 to the ON position (default setting is OFF).
5. Slide the card back into the cardframe.
6. Reconnect the RS-232/422/485 connectors.

Setting the RS-232/422 DIP Switch

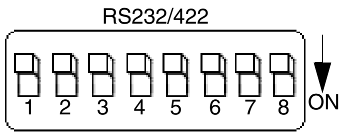
Set the stop bits, data bits, parity, and baud rate on the RS-232/422 DIP switch (see FIG. 1). The AXC-232++ supports the following asynchronous data standards:

- **Stop bits** 1 and 2
- **Data bits** 7, 8, and 9
- **Parity** None, Odd, Even, Mark, and Space
- **Baud rates** 300, 600, 1,200, 2,400, 4,800, 9,600, 19,200 and 38,400.
    - **57,600** is achieved by setting the DIP switch to 300 baud, and using the ’BAUDMED’ Send_Command (see the *Send_Commands* section on page 9 for details).
    - **115,200** is achieved by setting the DIP switch to 300 baud, and using the ’BAUDHIGH’ Send_Command (see the *Send_Commands* section on page 9 for details).

The table below shows the DIP switch numbers, functions, and their corresponding values.

RS-232/422 DIP Switch Settings

Switch	1	2	3	4	5	6	7	8
Function	Stop Bits	Data Bits		Parity			Baud Rates	
Setting	Off	Off	Off	Off	Off	Off	Off	Off
Value	2 bits	7 bits		Unused			300	
	On	On	On	Off	Off	On	Off	Off
	1 bit	8 bits		Unused			600	
			Off	On	Off	Off	On	Off
				Unused			1,200	
			On	On	Off	On	On	Off
				Unused			2,400	
			Off	Off	On	Off	Off	On
				Mark			4,800	
			On	Off	On	On	Off	On
				Even			9,600	
			Off	On	On	Off	On	On
				Odd			19,200	
			On	On	On	On	On	On
				None			38,400	



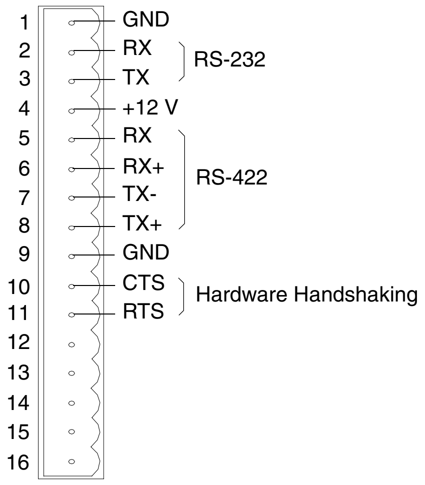
Wiring Devices to the AXC-232++

Card Edge Pinouts

The table below lists the card edge pinouts and the function of each.

Card Edge Pinouts

Pin	Signal	Function
1	GND	Ground (RS-232)
2	RX	Receive data (RS-232)
3	TX	Transmit data (RS-232)
4	+12V	Power
5	RX -	Receive data (RS-422)
6	RX +	Receive data (RS-422)
7	TX -	Transmit data (RS-422)
8	TX +	Transmit data (RS-422)
9	GND	Ground
10	CTS	Clear-to-send (hardware handshaking)
11	RTS	Ready-to-send (hardware handshaking)
12 - 16	Factory Use	



Preparing captive wires

To secure wires in a captive-wire connector:

1. Strip 1/4 inch off the wire insulation for all four wires.
2. Tin 2/3 of the exposed wire.
3. Insert each wire into the appropriate captive-wire connector, according to figures 9 through 12. Insert the wire up to the insulation.
4. Turn the captive screws clockwise to secure the fit in the connector.

CAUTION

If the device is using a separate power supply, do not connect the power wiring from the AXC-232++ to that device.

Using RS-232

When communicating via RS-232, connect the wiring as shown in FIG. 2.

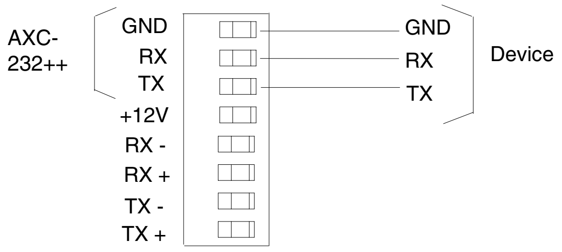


FIG. 2 RS-232 wiring

Using RS-422

When communicating via RS-422, connect the wiring as shown in FIG. 3.

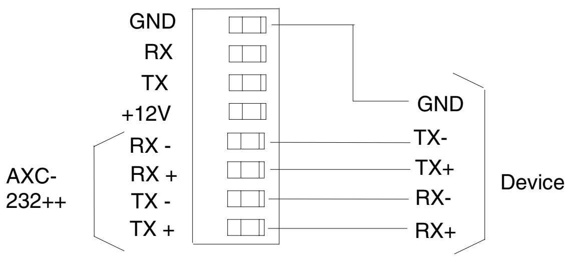


FIG. 3 RS-422 wiring

Using RS-485

When communicating via RS-485, connect the wiring as shown in FIG. 4.

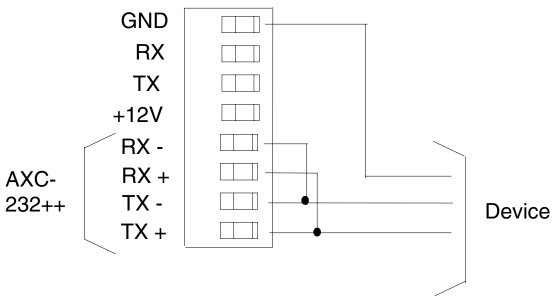


FIG. 4 RS-485 wiring

NOTE

To enable RS-485, set jumper J2 to the ON position (refer to Setting the Internal Jumpers on page 7).

Using Hardware Handshaking

When the controlled device requires hardware handshaking, connect the wiring as shown in FIG. 5.

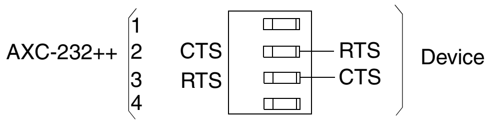


FIG. 5 Hardware handshaking wiring

Installing the AXC-232++ in a CardFrame:

1. Remove all static electricity by grounding yourself.
2. Hold the card by the tab end and slide the card into the frame, into one of the slots. Make sure the card is correctly positioned in the guide slot.
3. Gently push the card in until it is firmly seated in the card connector slot. You do not have to disconnect the power to the CardFrame when you install the card. When power is applied to the card, the LEDs blink to indicate the connection.

Replacing the Lithium Batteries

The AXC-232++'s lithium batteries have a life of approximately 5 years to protect its memory. When DC power is on, the batteries are not used. When you install the AXC-232++, record the date the batteries should be replaced.

CAUTION

There is a danger of explosion if you replace the batteries incorrectly. Replace batteries with the same or equivalent type recommended by the manufacturer. Dispose of the used batteries according to the manufacturer's instructions. Never recharge, disassemble, or heat batteries above 212°F (100°C). Never solder directly to the batteries or expose the contents of the batteries to water.

Before you remove the lithium batteries, contact your dealer and verify that they have a current copy of your program; this avoids any inadvertent loss of data and prevent an unnecessary service outage.

1. Discharge the static electricity from your body.
2. Unplug the two-pin power connector and any other connectors.
3. Slide the card out of the cardframe.
4. Carefully slide each battery out of its socket (FIG. 6), and insert the new battery.

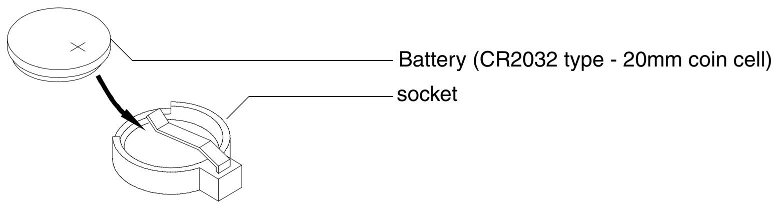


FIG. 6 Lithium battery and socket

5. Slide the card back into the cardframe.
6. Replace the front panel and refasten the two screws.
7. Reconnect any connectors that you removed.

Programming

Send_Commands

The AXC-232++ supports the same command set as the AXC-232. Figure 20 lists the AXC-232++ Send_Commands.

AXC-232++ Send_Commands	

B9MOFF
Sets data bit mode to normal with DIP switch settings (default).
	Syntax:
	`'B9MOFF'`
	Example:
	`SEND_COMMAND AXC232,'B9MOFF'`
	Sets the data bit mode to normal.

B9MON
Enables a special mode to override DIP switch settings.
	Syntax:
	`'B9MON'`
	The mode, nine data bits with one stop bit, overrides the data, stop, and parity settings. When the AXC-232++ issues this command, it locks in the baud rate determined by the current DIP switch setting.
	Example:
	`SEND_COMMAND AXC232,'B9MON'`
	Overrides the DIP switch settings.

BAUDHIGH
Enables 115,200 baud rate when the DIP switch is set to 300 baud.
	Syntax:
	`'BAUDHIGH'`
	Example:
	`SEND_COMMAND AXC232,'BAUDHIGH'`
	Enables 115,200 baud rate.

BAUDLOW
Enables 300 baud rate when the DIP switch is set to 300 baud setting (default).
	Syntax:
	`'BAUDLOW'`
	Example:
	`SEND_COMMAND AXC232,'BAUDLOW'`
	Enables 300 baud rate.

BAUDMED
Enables 57,600 baud rate when the DIP switch is set to 300 baud.
	Syntax:
	`'BAUDMED'`
	Example:
	`SEND_COMMAND AXC232,'BAUDMED'`
	Enables 57,600 baud rate.

CB1ON
Enables placement of characters in the buffer specified by CREATE_BUFFER 1 (default).
	Syntax:
	`'CB1ON'`
	Example:
	`SEND_COMMAND AXC232,'CB1ON'`
	Enables placement of characters in the buffer.

CB1OFF
Disables placement of characters in the buffer specified by CREATE_BUFFER 1.
	Syntax:
	`'CB1OFF'`
	Example:
	`SEND_COMMAND AXC232,'CB1OFF'`
	Disables placement of characters in the buffer.

AXC-232++ Send_Commands (Cont.)	

CHARD
Sets delay between all transmitted characters to increment specified.
	Syntax:
	`'CHARD-<time>'`
	Variable:
	time = 100 microsecond increments 0 - 255.
	Example:
	`SEND_COMMAND AXC232,'CHARD-10'`
	Sets 1mS delay between all transmitted characters.

CTSPSH
Enables PUSHes and RELEASEs.
	Syntax:
	`'CTSPSH'`
	Example:
	`SEND_COMMAND AXC232,'CTSPSH'`
	Enables PUSHes, RELEASEs, and status on channel 255 ([0,255] within the AXC-232++) for CTS hardware handshake input. If CTS is high, the channel is on.

CTSPSHF
Disables the 'CTSPSH' command.
	Syntax:
	`'CTSPSHF'`
	Example:
	`SEND_COMMAND AXC232,'CTSPSHF'`
	Disables the 'CTSPSH' command.

EOFF
Disables the 'EON ' command (default).
	Syntax:
	`'EOFF'`
	Example:
	`SEND_COMMAND AXC232,'EOFF'`
	Disables the 'EON' command.

EON
Forces the AXC-232++ to ignore the transmitted characters on its receiver.
	Syntax:
	`'EON'`
	Example:
	`SEND_COMMAND AXC232,'EON'`
	Forces the AXC-232++ to ignore the transmitted characters on its receiver. When using RS-485, the transmitter and receiver are tied together.

HSOFF
Disables hardware handshaking (default).
	Syntax:
	`'HSOFF'`
	Example:
	`SEND_COMMAND AXC232,'HSOFF'`
	Enables placement of characters in the buffer.

HSON
Disables placement of characters in the buffer specified by CREATE_BUFFER 1.
	Syntax:
	`'HSON'`
	Example:
	`SEND_COMMAND AXC232,'HSON'`
	Enables hardware handshaking.

RXCLR
Clears any characters in the Receive buffer waiting to be sent to the AXC-EM.
	Syntax:
	`'RXCLR'`
	Example:
	`SEND_COMMAND AXC232,'RXCLR'`
	Clears any characters in the Receive buffer waiting to be sent to the AXC-EM. If 'RXCLR' is sent while RTS is low, RTS returns high.

AXC-232++ Send_Commands (Cont.)	

RXOFF
AXC-232++ does not pass on received characters to the AXC-EM (default).
	Syntax:
	`'RXOFF'`
	Example:
	`SEND_COMMAND AXC232,'RXOFF'`
	AXC-232++ does not pass on received characters to the AXC-EM.

RXON
Enables AXC-232++ to send incoming characters received to the AXC-EM.
	Syntax:
	`'RXON'`
	Example:
	`SEND_COMMAND AXC232,'RXON'`
	Enables AXC-232++ to send incoming characters received to the AXC-EM.
	The AXC-EM automatically sends this command to the AXC-232++ when it executes a 'CREATE_BUFFER' program instruction.

TXCLR
Clears and stops any characters waiting in the Transmit buffer.
	Syntax:
	`'TXCLR'`
	Example:
	`SEND_COMMAND AXC232,'TXCLR'`
	Clears and stops any characters waiting in the Transmit buffer.

XOFF
Disables software handshaking (default).
	Syntax:
	`'XOFF'`
	Example:
	`SEND_COMMAND AXC232,'XOFF'`
	Disables software handshaking.

XON
Enables software handshaking.
	Syntax:
	`'XON'`
	Example:
	`SEND_COMMAND AXC232,'XON'`
	Enables software handshaking.

ZAP!
Clears the Axcess program in the AXC-232++.
	Syntax:
	`'ZAP!'`
	Example:
	`SEND_COMMAND AXC232,'ZAP!'`
	Clears the Axcess program in the AXC-232++.

Axcess program characteristics

The AXC-232++ is capable of running Axcess programs. It handles string processing, relieving AXlink and the AXC-EM of the processing times.

Send_String Escape Sequences

The AXC-232++ does not regard certain three-character combinations within a Send_String program as literal characters, but as commands. The following table lists those combinations.

Send_String Escape Sequences	

27,17,<time>
Sends a break character of the specified length of time.
	Syntax:
	`"27,17,<time>"`
	Variable:
	time = 100 microsecond increments 1 - 255.
	Example:
	`SEND_STRING AXC232,"27,17,10"`
	Sends a break character of 1mS.

27,18,1
Sets the ninth data bit to 1 for all of the following characters to be transmitted.
	Syntax:
	`"27,18,1"`
	Example:
	`SEND_STRING AXC232,"27,18,1"`
	Sets the ninth data bit to 1 for all of the following characters to be transmitted. Used in conjunction with the 'B9MON' command.

27,18,0
Clears the ninth data bit to 0 for all of the next characters to be transmitted.
	Syntax:
	`"27,18,0"`
	Example:
	`SEND_STRING AXC232,"27,18,0"`
	Clears the ninth data bit to 0 for all of the next characters to be transmitted. Used in conjunction with the 'B9MON' command.

27,19,<time>
Inserts a delay before the next character to be transmitted.
	Syntax:
	`"27,19,<time>"`
	Variable:
	time = 100 microsecond increments 1 - 255.
	Example:
	`SEND_COMMAND AXC232,"27,19,10"`
	Inserts a 1mS delay before the next character to be transmitted.

27,20,0
Asserts RTS hardware hand-shake output high.
	Syntax:
	`"27,20,0"`
	Example:
	`SEND_COMMAND AXC232,"27,20,0"`
	Asserts RTS hardware handshake output high.

27,20,1
Asserts RTS hardware hand-shake output low.
	Syntax:
	`"27,20,1"`
	Example:
	`SEND_COMMAND AXC232,"27,20,1"`
	Asserts RTS hardware handshake output low.

AXC-232++ statements

The Axcess program of the AXC-232++ communicates with the AXC-EM as Device 0. Its device communicates with the RS232/422 Input/Output (I/O) of the AXC-232++ as Device 1. The following table lists AXC-232++ statements.

AXC-232++ Program Statements

Statement	Function
`CREATE_BUFFER 0,`*`buffer`*	Places strings that come from the AXC-EM into *buffer*. If no `CREATE_BUFFER 0` exists, the incoming strings from the AXC-EM are sent out the RS232 port.
`CREATE_BUFFER 1,`*`buffer`*	Places strings that come from the RS-232 port into *buffer*. If no `CREATE_BUFFER 1`, exists, the incoming strings are sent to the AXC-EM.
`CREATE_BUFFER 2,`*`buffer`*	Places commands (i.e. `SEND_COMMAND`s) that come from the AXlink Master into the *buffer*. Each command will be preceded by 2 characters: The first character will always be an '*', the second character is the length of the command, the remaining *n* characters (as given in the second character) are the command itself).
`CREATE_LEVEL 0,`*`level,variable`*	Places levels sent by the AXC-EM for *level* in *variable*.
`OFF[0,`*`channel`*`]`	• Sends a RELEASE to the AXC-EM. • Sends message that *channel* is off.
`OFF[1,`*`channel`*`]`	Turns off a *channel*. This command has no external effect. It can be used as status.
`ON[0,`*`channel`*`]`	• Sends a PUSH to the AXC-EM. • Sends a message that *channel* is on.
`ON[1,`*`channel`*`]`	Turns on *channel*. This command has no external effect. It can be used as status.
`SEND_COMMAND 0,`*`command`*	Sends *command* to the AXC-EM. Provides support for commands like `'RDS'`.
`SEND_COMMAND 1,`*`command`*	Sends *command* to the AXC-232++ as if the AXC-EM had sent it.
`SEND_COMMAND AXC_232, 'BAUDHIGH'`	Sets the baud rate to 115.2k (DIP switch set to 300).
`SEND_COMMAND AXC_232, 'BAUDLOW'`	Sets the baud rate to 300 (DIP switch set to 300).
`SEND_COMMAND AXC_232, 'CB1ON'`	Enables placement of characters in the *buffer* specified by `CREATE_BUFFER 1`.
`SEND_COMMAND AXC_232, 'CB1OFF'`	Disables placement of characters in the *buffer* specified by `CREATE_BUFFER 1`.
`SEND_LEVEL 0,`*`level,variable`*	Sends *variable* as the value for level..
`SEND_STRING 0,`*`string`*	Sends *string* to the AXC-EM.
`SEND_STRING 1,`*`string`*	Sends *string* out the RS-232 port.

Xmodem Timing Commands

The following table lists the AXC-232++ Xmodem timing commands. Xmodem timeouts exist to accommodate potential Ethernet delays and for consistency among and within products. Any Xmodem timing command will change timing and retries for Axcess code download as well as SOFTROM transfer.

Xmodem Timing Commands	
Command	**Description**
`'TIMEOUT XX'`	Xmodem timeouts via the Program Port. (Default is 10 sec.)
`SEND_COMMAND SERIAL, 'XMTO XX'`	Over AXlink, where `XX` is from one to 50 seconds in 1-second increments.
`'RETRY XX'`	Xmodem retries via the Program Port. (Default is 5.)
`SEND_COMMAND SERIAL, 'XMRT XX'`	Over AXlink where `XX` is from one to 10 in increments of one.

AXlink Master Statements

The following table lists the AXlink Master statements for the AXC-232++.

AXlink Master Statements	
Statement	**Function**
`MIN_TO`	Activates a channel or variable for a minimum amount of time-even if the corresponding device-channel is released. The time duration is determined by SET_PULSE_TIME.
`ON[AXC_232, `*`channel`*`]`	Executes PUSH[0,*channel*] within AXC-232++.
`OFF[AXC_232, `*`channel`*`]`	Executes RELEASE[0,*channel*] within AXC-232++.
`PULSE`	Turns on a channel or variable for a certain amount of time. Once the time elapses, the channel or variable is turned off. The Pulse time remains the same value until it is changed within the program. Example: `SET_PULSE_TIME(12)` Sets the current duration of future Pulses to 1.2 seconds.
`SEND_COMMAND AXC_232,`*`command`*	AXC-232++ interprets *command*.
`SEND_LEVEL AXC_232,`*`level, variable`*	Places the *variable* into the CREATE_LEVEL variable, if it were defined in the AXC-232++ DEFINE_START section.
`SEND_STRING AXC_232,`*`string`*	Places the *string* into the CREATE_BUFFER buffer, if you defined the buffer in the AXC-232++ DEFINE_START section. If it were not, this command transmits the *string* out the RS-232 port.
`TO`	Activates a channel or variable for as long as the corresponding device-channel of its Push statement is activated. When the device-channel referenced by the Push statement changes from off to on, the TO starts activating the device-channel or variable in the brackets following it. When the device-channel of its Push is released, the TO statement stops activating its device-channel or variable. For this reason, TO must be found only underneath a Push statement.

Reserved Channels

The following table lists the channels reserved on Device 0.

Reserved Channels	
Channel	**Function**
Channel 254	This is only valid within the device and is not sent to the AXlink master. Reflects AXlink status: • When AXlink is active, this channel is on. • When AXlink is inactive, this channel is off. • When this channel changes state, the AXC-232++ generates a PUSH or RELEASE.
Channel 255	Reflects the state of the CTS input. If the AXC-232++ receives a 'CTSPSH' command, it generates PUSHes for both the AXC-EM and the AXC-232++.

Axcess Master Mode

When an Axcess device is placed in "Master Mode", the Central Controller's PROGRAM port is moved to the Axcess device's RS-232 port.

Press the escape key, then type either **MC** or **MD**:

- <esc>**MC** - connects the device in Master Mode
- <esc>**MD** - disconnects the device

where <esc> means "press the Esc key".

NOTE

Master Mode can be very useful in situations where physical access to the Central Controller's PROGRAM port is not practical (for example, in installations where the Central Controller is located a long distance from the bus device(s)).

AMX reserves the right to alter specifications without notice at any time.

Last Revision: 9/12/05