The *Acela* Network Bridge Programmer's Guide

For Acela Firmware Revision 1.3

CTI Electronics

This document describes the use of CTI's "*Acela*" *Network Bridge* circuit card, and defines the message formats used to communicate between the *Acela Network Bridge* and a host PC.

This document is intended for use by those developing PC software for use with CTI's model railroad control hardware.

Some familiarity with the CTI Model Railroad Control System is assumed. For more details, consult the *CTI Model Railroad Computer Control System User's Guide*, available for download from our website at www.cti-electronics.com.

Table of Contents

1. Introducing the "*Acela*" Network Bridge

In a conventional CTI installation, the host PC communicates directly over the CTI network, using the network's low-level protocol to disseminate control data to and retrieve sensor data from one or more CTI modules.

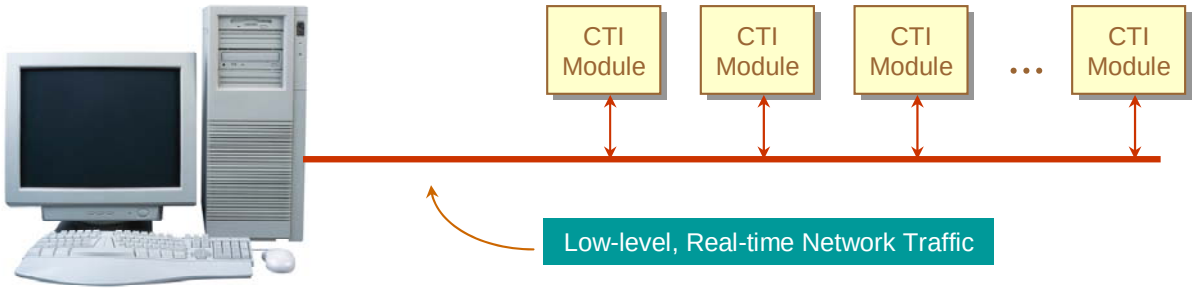


Figure 1-1 A Conventional CTI Network

In this approach, the PC is directly responsible for the control and monitoring of the modules. As such, the user's software must address the need for real-time responsiveness of the network. This is becoming more and more difficult for the programmer, as each new generation of the Windows operating system further isolates application software from direct access to the interface hardware in the PC.

To ease the task of developing software for use with the CTI network, CTI introduced the "*Acela" Network Bridge.* The *Acela* serves as an "intermediary" between software in the PC and the low-level communications continually taking place over the CTI network.

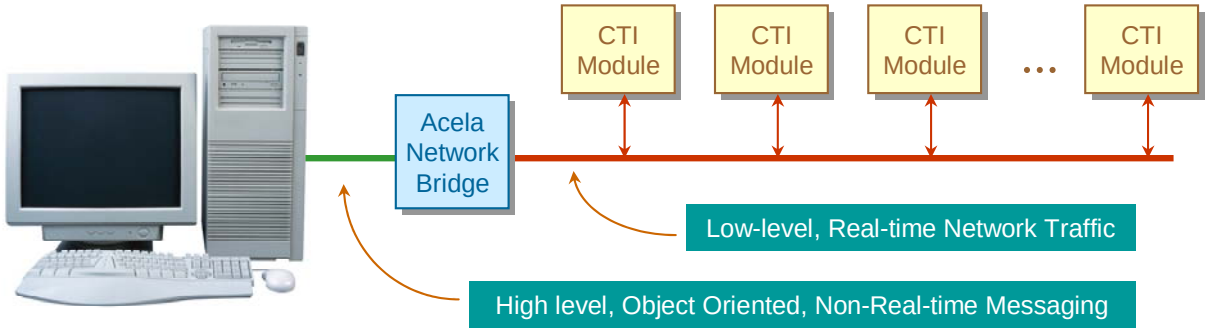


Figure 1-2 A "Bridged" CTI Network

The *Acela* handles all of the tasks associated with managing the CTI modules, freeing applications software from the real-time burden of network communications. In this way, software can take a high level, "object-oriented" view of the train layout, without worrying about the specifics of the underlying control network. These low-level details become completely transparent to software, being performed efficiently in hardware by the network bridge.

The *"Acela"* :

- continually monitors the CTI network's sensors and notifies software whenever the layout requires attention due to a change in the state(s) of one or more sensors, removing the burden of continual polling of the network by software.

- latches all sensor state changes until they can be read by the PC, eliminating the need for software to respond in real-time to momentary sensor actuations.

- applies sophisticated digital filtering algorithms to eliminate sensor false alarms resulting from intermittent track contact, car-to-car gaps, mechanical switch bounce, etc. removing the burden of performing this filtering in software.

- maintains a high precision time-base, so that real-time actions (such as the pulsing of controls) may be handled completely in hardware, eliminating the need for precise real-time event timing in software.

- continually refreshes control data to all modules throughout the CTI network, yielding a robust control system, immune to the inevitable noise and power glitches found in a model railroad environment.

- performs validity checking on all responses from the network, detecting and correcting transmission errors resulting from power glitches and noise.

- provides easy-to-use signaling macros, allowing virtually any signaling protocol to be implemented with a single high-level command from the PC.

During operation, the *Acela's* status LED provides a simple indication of the state of the network bridge. When initially powered on, the LED should blink yellow. This means that the Acela has powered up, has successfully checked itself out, and is o*ffline*, waiting to begin communications with the CTI network.

Once the PC commands the network bridge to go *online*, the LED should begin rapidly flashing green. Each time it does, the network bridge has refreshed all controls in the network and has retrieved new data from all sensors in the network. If the PC commands the network bridge to go o*ffline*, the LED will return to blinking Yellow.

If the *Acela* detects a communications failure with the CTI network, the LED will blink red.

The *Acela* is available in two versions. The first (CTI part # TB12-COM) connects to the PC via an RS232 (COM) port. The second (CTI part # TB12-USB) connects to the PC via the Universal Serial Bus (USB). The application software interface to the two versions is identical.

AcelaApp is an applications programming example written in Visual Basic intended to accompany this programming guide. While rudimentary, it fully illustrates the use of the commands described in this document to perform the control and monitoring functions needed to implement a more sophisticated Centralized Traffic Control facility.

1.1 Installing the "*Acela*" COM Port Network Bridge

To install the COM port version of the Acela, connect the Red and Green ports of the Acela to the Green (head) and Red (tail) endpoints of the CTI network, respectively. Connect the Yellow port of the Acela to the COM port interface adapter. Finally, plug the interface adapter into a COM port on the PC. An example of an Acela-based CTI network is shown in Figure 1-3.

The COM port should be set for operation at 9600 Baud, with 8 data bits, 1 stop bit, and no parity.

The COM port version of the Acela requires a power supply in the range of 9 to 12 Volts DC, the same as all other CTI network modules. This power supply plugs into the black power jack on the Acela board. The center conductor is (+); the outer conductor is ground (-). Worst-case current draw is about 50 milliamps.

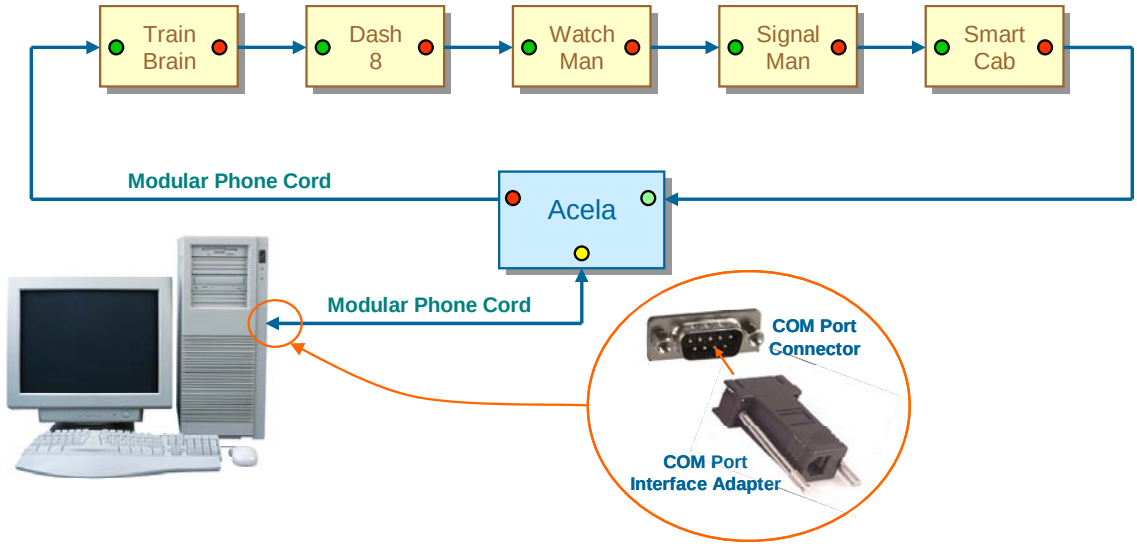


Figure 1-3 An Example of a COM Port-Connected Acela-Based CTI Network

1.2 Installing the "*Acela*" USB Network Bridge

Unlike the simpler COM-port version of the Acela, the USB Acela requires a plug-and-play device driver to operate under Windows operating systems. The necessary driver information file (*acela.inf*) is included in the same self-extracting zip file (AcelaZip.exe) that contained this App Note (available at www.cti-electronics.com\app_sw.htm). Extract the driver from the zip file to your local hard drive, for example in C:\temp. (If you wish, you can delete the file once installation is complete.)

To install the USB version of the Acela, connect the Red and Green ports of the Acela to the Green (head) and Red (tail) endpoints of the CTI network, respectively. Connect the Blue port of the Acela to a USB port on the PC using a standard Type A-to-Type B USB cable. An example of an Acela-based CTI network is shown in Figure 1-4.

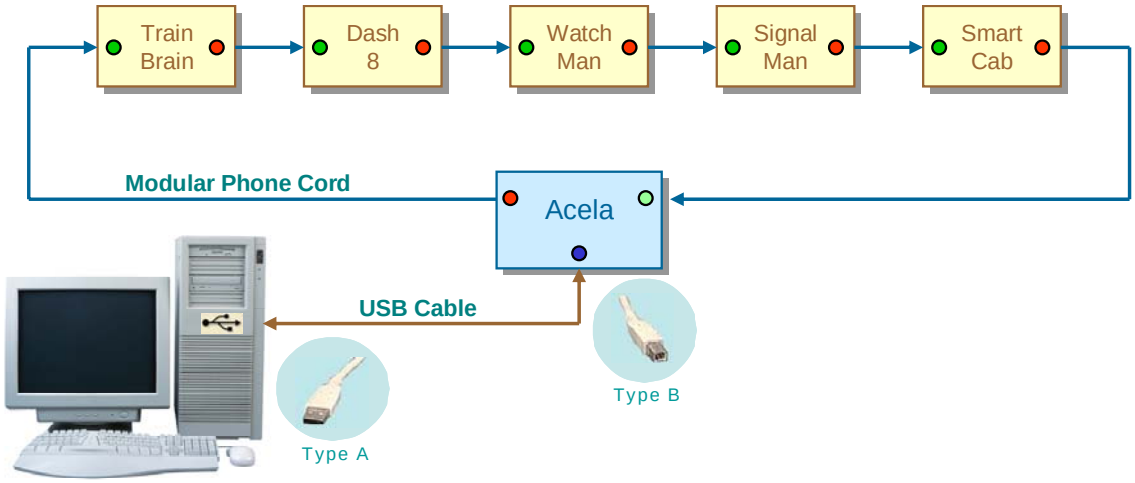
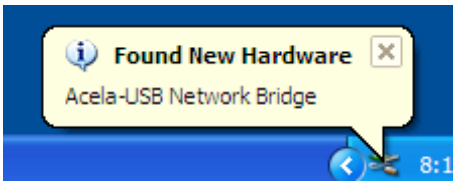


Figure 1-4 An Example of a USB-Connected Acela-Based CTI Network

With the Acela hardware installed, we'll next turn our attention to installing the USB device driver used by the Acela. The next several paragraphs will walk you through the process of driver installation. The screen shots captured here are representative of a Windows-XP environment. The process under Windows Vista is similar, but the appearance of the windows will differ slightly.

The first time the Acela is plugged into a USB port, Windows should respond with its familiar "da-DING" sound, indicating that a new plug-and-play device has been detected. A pop-up message indicates that the new device is an Acela Network Bridge.



Windows will automatically run its "plug-and-play" driver installation Wizard.

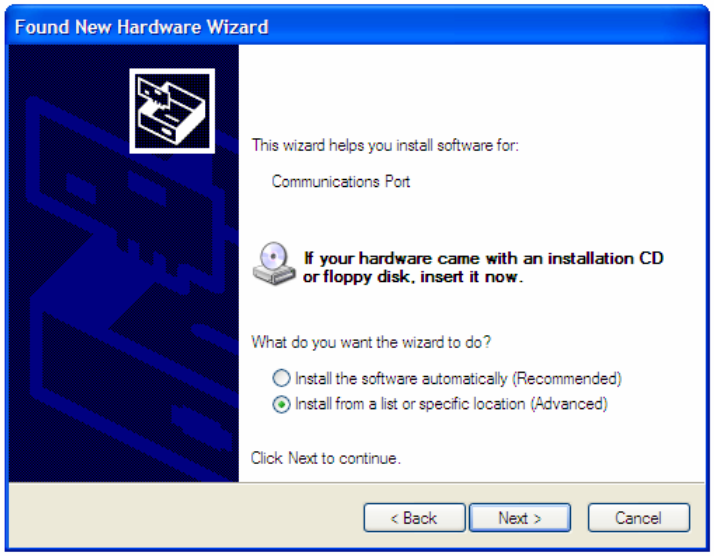
Windows will ask for permission to search the Internet for a software driver for the newly installed hardware.

But since we already have the driver stored locally, we'll instead opt to locate it ourselves by selecting the "No, not this time" option.

Then click "Next".

Since we downloaded the driver file from the CTI website, there is no physical Installation CD, so we'll choose the "Install from a list or specific location" option.

Then click "Next".



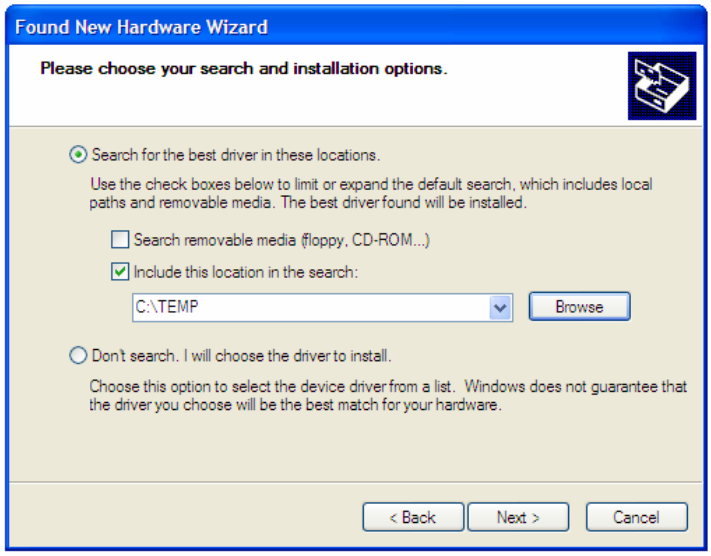
Next, we'll tell Windows where the driver file is located.

Choose the "Search for the best driver in these locations" option.

Check the "Include this location in the search" option, and browse to the folder where you stored the driver information file (*acela.inf*), when you extracted it from the self-extracting zip file.

(In this example, it has been stored in the "Temp" folder on the C: drive.)

Then click "Next".

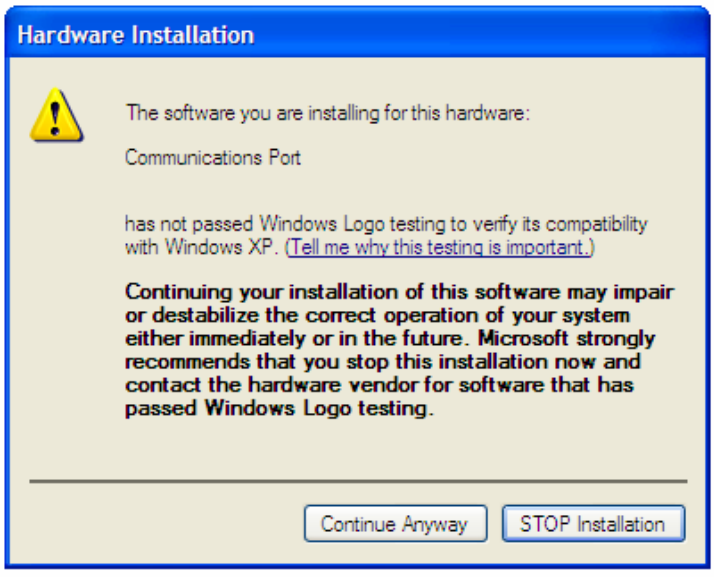


Once Windows has located the driver information file, it will warn us of impending Armageddon since the driver does not have a Windows Logo.

But in fact, we aren't installing any new drivers at all. The Acela uses Microsoft's own USB driver that's already built into Windows, as instructed by the *acela.inf* file.

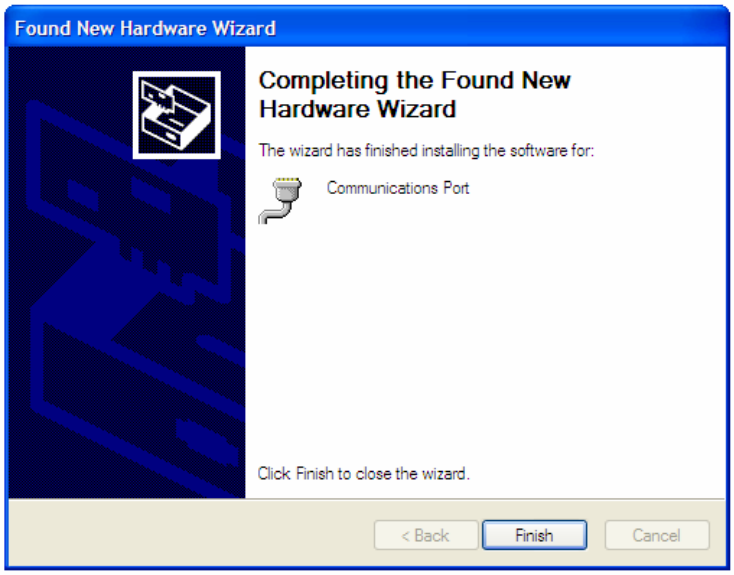
Unfortunately, Windows isn't smart enough to figure that out.

Simply click the "Continue Anyway" option.



If all goes well, Windows should inform us that the driver has been installed and the Acela is now ready for use.

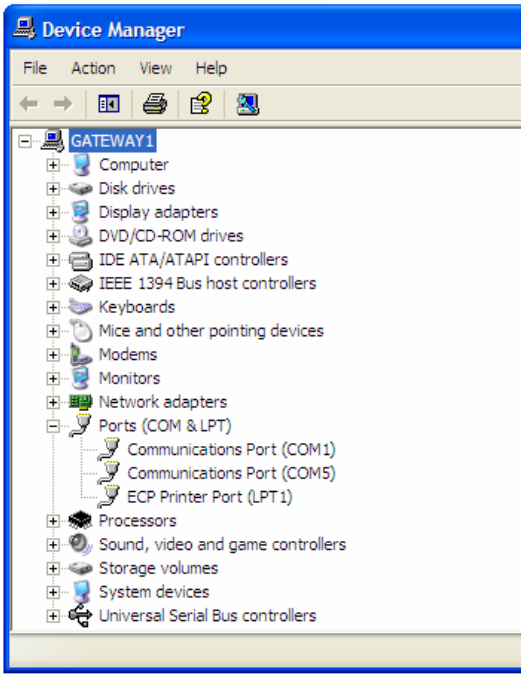
Click “Finish”.



The Acela registers with Windows as a “virtual” COM port. This means that your application software can use the familiar (and easy-to-use) COM port programming model. You won’t need to worry about all the intricacies of USB interfacing.

You’ll just need to find out the port number of the new virtual COM port that Windows created for use with the Acela. You’ll find that answer in the Windows Device Manager. To get there …

1) On the Desktop click Start-Control Panel.
2) Double click the “System” icon.
3) Select the Hardware tab, and click the “Device Manager” button.
4) Click on the “+” sign next to “Ports (COM & LPT) to expand this item.



Here you’ll find a list of all the printer and COM ports on your PC. One of them should be the virtual COM port that Windows has just created for use with the Acela. Note its COM port number. This will be the port number that your application will use to access the Acela. (In the example, Windows has assigned the Acela to COM port #5.)

If you’re not sure which COM port is the Acela, simply unplug the Acela from the USB port. Its COM port should disappear from list of ports in the Device Manager. Plug it back in and its COM port should reappear.

2. Communications Protocol

2.1 Command Messages

The host PC sends *command* messages to the network bridge to instruct it to carry out some action. The formats of these *command* messages are described in Section 3 of this document.

Command messages always begin with an *opcode* byte. In some cases, the opcode byte contains all of the information needed by the Acela to carry out the requested action. In other cases, the *opcode* is followed by one or more additional bytes containing information specific to a given command.

Often, commands are targeted at a particular control or sensor in the CTI network. In these cases, a 2-byte *address* is sent as part of the message indicating the selected control or sensor.

In the CTI network, the *address* of a control or sensor is determined by its position in the network. Controls and sensors each occupy independent address spaces. The first control is at control address '0'; the second is at control address '1', etc. Similarly, the first sensor is at sensor address '0'; the second is at sensor address '1', etc. Up to 65536 controls and 65536 sensors may be addressed in this way.

Each CTI module consumes a number of control and/or sensor addresses as shown in Table 2-1.

Table 2-1 Control and Sensor Address Utilization of Various CTI Modules

Module Style	Control Addresses Occupied	Sensor Addresses Occupied
Train-Brain	4	4
Dash-8	8	0
Watchman	0	8
Signalman	16	0
Smart Cab	1	0
Switchman	16	0
YardMaster	16	0
Sentry	0	16

For example, in the network shown in Figure 1-3, the addresses occupied by each control or sensor are listed in Table 2-2.

Table 2-2 Addresses of Controls and Sensors in the example CTI Network of Figure 2-1

Entity		Network Address
Train Brain	Control #1	0
	Control #2	1
	Control #3	2
	Control #4	3
	Sensor #1	0
	Sensor #2	1
	Sensor #3	2
	Sensor #4	3
Dash-8	Control #1	4
	Control #2	5
	Control #3	6
	Control #4	7
	Control #5	8
	Control #6	9
	Control #7	10
	Control #8	11
Watchman	Sensor #1	4
	Sensor #2	5
	Sensor #3	6
	Sensor #4	7
	Sensor #5	8
	Sensor #6	9
	Sensor #7	10
	Sensor #8	11
Signalman	Control #1	12
	Control #2	13
	Control #3	14
	Control #4	15
	Control #5	16
	Control #6	17
	Control #7	18
	Control #8	19
	Control #9	20
	Control #10	21
	Control #11	22
	Control #12	23
	Control #13	24
	Control #14	25
	Control #15	26
	Control #16	27
Smart Cab	Throttle Control	28

2.2 Response Messages

The network bridge acknowledges all *command* messages sent to it from the PC by sending a *response* message back to the PC.

The PC may send a *command* message to the network bridge at any time. However, once a command has been sent, no further data should be transmitted until a *response* message has been received back from the network bridge, acknowledging that the current command has been received and processed.

In many cases, the network bridge responds with the single-byte "*Command Acknowledge"* message. The value returned in the *Command Acknowledge* message byte is defined as follows:

Table 2-3 The "*Command Acknowledge*" Message Byte

Command Acknowledge Message Byte Value	Meaning
0x00	The command has been processed successfully.
0x01	The command has been processed, but cannot be carried out at this time because the CTI network is offline. The command will take effect once the network is placed online.
0x02	The command addresses a control or sensor beyond the range of the current network hardware.
0x03	An unknown command message was received and discarded.

Some commands request that the network bridge return data to the PC.

In these cases, the network bridge response again begins with the "*Command Acknowledge"* byte.

If the *Command Acknowledge* byte indicates that the command was successfully processed (i.e. the acknowledge byte value equals '0'), then the requested data follows. (See Section 3 for the format of these response messages).

If the *Command Acknowledge* byte indicates that processing of the command was unsuccessful, then the requested data is not returned.

2.3 Service Request Messages

The network bridge sends a *Service Request* message to the PC whenever it detects a condition that requires the attention of software running on the PC.

This may occur when the network bridge observes a change in the state of one or more sensors, or when the bridge experiences a loss of communications with the network.

The value contained in the single-byte *Service Request* message is defined as follows:

Table 2-4 The *Service Request* Message Byte

Service Request Message Byte Value	Meaning
0x81	A change in the state of one or more sensors has been observed.
0x82	Communications with the CTI network have been lost.

2.4 Error Handling and Recovery

If the network bridge receives a corrupted response, or receives no response at all, from the CTI network, the bridge will make several attempts to reestablish communications. If the fault was momentary (e.g. due to a transient power glitch), and communications are immediately reestablished, the network bridge continues with normal operation.

If repeated attempts to reestablish communications fail, the network bridge will cease communicating with the network, will turn itself offline, and will send a *service request* message to the PC notifying it of the failure.

Once the problem is identified and corrected, software must turn the network bridge online to resume communications with the CTI network.

3. Command and Response Message Formats

This section documents the *command* and *response* messages used to communicate between the PC and the *Acela Network Bridge.*

3.1 Commands for Use with Discrete Controls

The commands in this section are intended for use in actuating discrete controls. A "discrete" control may be a Train Brain or Dash-8 control relay or a single Switchman, Yardmaster, or Signalman transistor control output.

3.1.1 Activating/Deactivating a Control

The "*Activate*" command may be used to activate a single control.

The "*Activate*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x01)	Control Address (Bits 15:8)	Control Address (Bits 7:0)

The CTI network activates the selected control. The control remains activated until it is again addressed by a command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

The "*Deactivate*" command may be used to deactivate a single control.

The "*Deactivate*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x02)	Control Address (Bits 15:8)	Control Address (Bits 7:0)

The CTI network deactivates the selected control. The control remains deactivated until it is again addressed by a command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.1.2 Pulsing a Control Momentarily On/Off

The "*Pulse On*" command may be used to momentarily activate a single control.

The "*Pulse On*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x03)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Pulse Width ('N' 10'ths of seconds)

The CTI network momentarily activates the selected control for $^{N}/_{10}$ seconds, where 'N' is the value contained in the *Pulse Width* byte (byte 4) of the "*Pulse On*" command. The control is then deactivated, and remains deactivated until it is again addressed by a command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

The "*Pulse Off*" command may be used to momentarily deactivate a single control.

The "*Pulse Off*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x04)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Pulse Width ('N' 10'ths of seconds)

The CTI network momentarily deactivates the selected control for $^{N}/_{10}$ seconds, where 'N' is the value contained in the *Pulse Width* byte (byte 4) of the "*Pulse Off*" command. The control is then activated, and remains activated until it is again addressed by a command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.1.3 Blinking a Control

The "*Blink*" command may be used to continually blink a single control.

The "*Blink*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x05)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Pulse Width ('N' 10'ths of seconds)

The CTI network activates the selected control for $^{N}/_{10}$ seconds, where 'N' is the value contained in the *Pulse Width* byte (byte 4) of the "*Blink*" command. The control is then deactivated for $^{N}/_{10}$ seconds. This On/Off cycle repeats indefinitely, until the control is again addressed by a command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

The "*Reverse Blink*" command may be used to continually blink a single control.

The "*Reverse Blink*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x06)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Pulse Width ('N' 10'ths of seconds)

The CTI network deactivates the selected control for $^{N}/_{10}$ seconds, where 'N' is the value contained in the *Pulse Width* byte (byte 4) of the "*Reverse Blink*" command. The control is then activated for $^{N}/_{10}$ seconds. This Off/On cycle repeats indefinitely, until the control is again addressed by a command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

Note: "Reverse Blinking" a control differs from Blinking a control in that the action sequence is *Off ... On ...Off ...On* rather than *On ... Off ... On ... Off*. Thus, for example, a crossing flasher could be implemented using two controls; Blinking the first, and Reverse Blinking the second. In this way, the two lamps would alternately flash in sequence.

3.1.4 Simultaneously Configuring 4 Controls

The "*Control 4*" command may be used to simultaneously configure four contiguous controls.

The *Control 4* command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x07)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Control State x x x x 3 2 1 0

The CTI network configures the selected control based on the state of bit 0 of the *Control State* byte (byte 4) of the *Control 4* command. A '1' in bit 0 causes the selected control to be activated; a '0' results in its deactivation. The CTI network simultaneously configures the next three controls, in similar fashion, based on the states of bits 1, 2, 3 of the *Control State* byte.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.1.5 Simultaneously Configuring 8 Controls

The "*Control 8*" command may be used to simultaneously configure eight contiguous controls.

The *Controls 8* command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x08)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Control State 7 6 5 4 3 2 1 0

The CTI network configures the selected control based on the state of bit 0 of the *Control State* byte (byte 4) of the *Control 8* command. A '1' in bit 0 causes the selected control to be activated; a '0' results in its deactivation. The CTI network simultaneously configures the next seven controls, in similar fashion, based on the states of bits 1, 2, … 7 of the *Control State* byte.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.1.6 Simultaneously Configuring 16 Controls

The "*Control 16*" command may be used to simultaneously configure sixteen contiguous controls.

The *Control 16* command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Op Code (0x09)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Control State F E D C B A 9 8	Control State 7 6 5 4 3 2 1 0

The CTI network configures the selected control based on the state of bit 0 of the *Control State* bytes (bytes 4, 5) of the *Control 16* command. A '1' in bit 0 causes the selected control to be activated; a '0' results in its deactivation. The CTI network simultaneously configures the next fifteen controls, in similar fashion, based on the states of bits 1, 2, … F of the *Control State* bytes.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.2 Commands for Use with Throttles

The commands in this section are intended for use in controlling Smart Cab throttles.

3.2.1 Controlling Train Speed, Direction, and Momentum

The "*Throttle*" command may be used to control a Smart Cab throttle.

The "*Throttle*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Op Code (0x0A)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Speed	Attributes

The CTI network configures the selected throttle based on the values contained in the *Speed* and *Attributes* bytes of the "*Throttle*" command.

Valid decimal values for the S*peed* byte (byte 4) are 0 (stop) through 100 (full throttle). Values greater than 100 are illegal.

The bits of the *Attribute* byte (byte 5) are defined as follows:

Table 3. The *Throttle* Command's *Attributes* Byte

Attribute Bits	Function	Acceptable Values
Bits (2:0)	Momentum Control	000 (Minimum Inertia) through 111 (Maximum Inertia)
Bit 3	Brake Control	0 = Brake Off 1 = Brake On
Bit 4	Direction Control	0 = Forward 1 = Reverse
Bit 5	Idle Voltage Control	0 = Do not apply an idling voltage 1 = Maintain a small idling voltage for use with current sensors
Bit 6	Unused	For future compatibility, software should set this bit to 0
Bit 7	Unused	For future compatibility, software should set this bit to 0

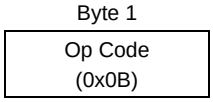
The network bridge replies to the command with a *Command Acknowledge* message to the PC.

Note: Software need not take any special precautions when changing the direction of a throttle. The CTI network will automatically bring a train to a smooth stop (using the currently selected momentum control setting) before changing the direction of the train. The train will then smoothly throttle back up to its original speed in the new direction.

3.2.2 Emergency Stop

The "*Emergency Stop*" command may be used to immediately halt all Smart Cab throttles.

The *Emergency Stop* command is defined as follows:



The CTI network immediately applies the brakes on all Smart Cab throttles with low momentum, thereby quickly stopping all trains.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.3 Commands for Use with Signals

The commands in this section are intended for use in controlling signals.

A signal may be viewed as a simple collection of discrete lamps, each independently controlled. As such, any of the commands described in Section 3.1 for use with discrete controls may be used to control individual signal lamps. However, the commands introduced in this section make it much easier to produce sophisticated signaling effects on multi-lamp signals using just a single command from the PC.

3.3.1 Controlling 2-lamp Signals

The "*Signal 2*" command may be used to configure any signal controlled by two adjacent Signalman outputs. This includes 2-lamp incandescent and LED-based signals, as well as single lamp bicolor LED-based searchlight signals. The command provides support for the generation of a yellow signal aspect through mixing of the red and green signal hues. Any signal aspect may be programmed using the command.

The *Signal 2*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x0C)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Signal Aspect

The CTI network configures the selected control pair (the address sent as part of the *Signal 2* command is the address of the lower-numbered controller) based on the value contained in the *Signal Aspect* byte (byte 4) of the "*Signal 2*" command.

The *Signal Aspect* byte is comprised of three 2-bit fields as shown below:

Bit #	7	6	5	4	3	2	1	0
Function	Unused		Synthetic Yellow Control		Lamp 2 Control		Lamp 1 Control	
Allowed Values	xx		00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink	

Independent 2-bit fields control each of the signal's 2 lamps. Using the values shown above, each of the signal's lamps may be programmed (in any combination) to be Off or On, to Blink, or to Reverse Blink.

In addition, a third 2-bit field controls the synthetic yellow aspect for bicolor LEDs. This combination allows any signal aspect to be programmed by a single command from the PC.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.3.2 Controlling 3-lamp Signals

The "*Signal 3*" command may be used to configure any signal controlled by three adjacent Signalman outputs. Any signal aspect may be programmed using the command.

The " *Signal 3*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x0D)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Signal Aspect

The CTI network configures the selected three controls (the address sent as part of the *Signal 3* command is the address of the lowest-numbered controller) based on the value contained in the *Signal Aspect* byte (byte 4) of the "*Signal 3*" command.

The *Signal Aspect* byte is comprised of three 2-bit fields as shown below:

Bit #	7	6	5	4	3	2	1	0
Function	Unused		Lamp 3 Control		Lamp 2 Control		Lamp 1 Control	
Allowed Values	xx		00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink	

Independent 2-bit fields control each of the signal's 3 lamps. Using the values shown above, each of the signal's lamps may be programmed (in any combination) to be Off or On, to Blink, or to Reverse Blink.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.3.3 Controlling 4-lamp Signals

The "*Signal 4*" command may be used to configure any signal controlled by four adjacent Signalman outputs. Any signal aspect may be programmed using the command.

The "*Signal 4*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x0E)	Control Address (Bits 15:8)	Control Address (Bits 7:0)	Signal Aspect

The CTI network configures the selected four controls (the address sent as part of the *Signal4* command is the address of the lowest-numbered controller) based on the value contained in the *Signal Aspect* byte (byte 4) of the "*Signal 4*" command.

The *Signal Aspect* byte is comprised of four 2-bit fields as shown below:

Bit #	7	6	5	4	3	2	1	0
Function	Lamp 4 Control		Lamp 3 Control		Lamp 2 Control		Lamp 1 Control	
Allowed Values	00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink		00 = Off 01 = On 10 = Blink 11 = Reverse Blink	

Independent 2-bit fields control each of the signal's 4 lamps. Using the values shown above, each of the signal's lamps may be programmed (in any combination) to be Off or On, to Blink, or to Reverse Blink.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.3.4 Setting Signal Parameters

The "*Signal x*" commands described above employ two user-definable parameters. The first controls the blink rate of any signal lamps programmed to blink (or reverse blink). The second controls the relative percentages of Red and Green mixed to produce yellow in bicolor LEDs.

The user may program these parameters by using the "*Signal Settings*" command.

The "*Signal Settings*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x0F)	Blink Rate	Yellow Hue

The CTI network blinks signals at a rate defined by the *Blink Rate* byte of the *Signal Settings* command. This value specifies the amount of *On* and *Off* time for the signal lamp in 10'ths of seconds. For example, a value of 10 yields a one second On, one second Off blink rate.

The CTI network mixes Red and Green to produce a synthetic yellow signal aspect in relative percentages determined by the value specified in the *Yellow Hue* control byte of the *Signal Settings* command. Higher values result in the use of more Green and less Red. Lower values result in the use of more Red and less Green. A value of 128 implies a 50% Red and Green mix. (In general, a mixture of 66% Green and 33% Red produces a good approximation of yellow.)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

Note: On power-up, the network bridge initializes the *Blink Rate* control to 10, for a 1 second On/Off rate for blinking signals. The network initializes the *Yellow Hue* control to 170, yielding a mix of 66% Green and 33% Red for signals programmed for synthetic yellow.

3.3.5 Adjusting Signal Brightness

The luminous intensity of signals may be adjusted at any time (e.g. in response to changing ambient light conditions) by using the "*Signal Brightness*" command.

The "*Signal Brightness*" command is defined as follows:

Byte 1	Byte 2
Op Code (0x1B)	Brightness

The CTI network sets the brightness of signals based upon the value specified in the *Brightness* control byte of the *Signal Brightness* command. Lower values result in dimmer signals; higher values yield brighter signals.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

Note: On power-up, the network bridge initializes signal *brightness* to 255 (full brightness).

3.4 Commands for Use with Sensors

The commands in this section are intended for use in working with sensors.

3.4.1 Configuring Sensors

The "*Configure Sensor*" command may be used to program the operation of sensors in the CTI network.

The "*Configure Sensor*" command is defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4
Op Code (0x10)	Sensor Address (Bits 15:8)	Sensor Address (Bits 7:0)	Sensor Attributes

The behavior of the selected sensor is defined by the value contained in the *Sensor Attributes* byte (byte 4) of the message. The bits of the attribute byte are defined as follows:

7	6	5	4	3	2	1	0
Filter Threshold					Filter Select		Polarity
00000 (Lightly Filtered) through 11111 (Heavily Filtered)					00 = Noise Filter Only 01 = Switch Bounce 10 = Car Gap 11 = Dirty Track		0 = Normal 1 = Invert

The ***Polarity*** bit of the *Sensor Attributes* byte controls the logical sense of the sensor indication reported by the CTI network to software.

Note: Magnetic and current detecting sensors normally respond as *False* when no train is present, and *True* when one is. Infrared and photocell sensors work in the opposite way. They detect light, and thus respond as *True,* when no train is present, and *False* when the light beam is interrupted by the presence of the train.

If desired, the user may invert the logical sense of a sensor by setting its *Polarity* bit to '1'. This will cause raw sensor reports from that sensor to be logically inverted before being forwarded to software. For example, setting the *Polarity* bit of an infrared sensor to '1' will cause it to report as *True* when a train is present, and *False* when one is not.

The ***Filter Select*** bits allow the user to select from a variety of digital filtering algorithms built into the CTI hardware to be applied to the raw sensor reports before being forwarded to software. All raw sensor reports are passed through a digital high-frequency noise rejection filter. In addition, the user may choose to apply a switch de-bounce filter (intended for use with magnetic reed switch sensors or manual pushbuttons), a car-cap filter (intended for use with infrared and photocell sensors), or a dirty track filter (intended for use with current detection sensors).

If one of the digital filtering algorithms is enabled for a given sensor, the user may also define the level of filtering applied to that sensor using the ***Filter Threshold*** bits in the *Sensor Attributes* byte. The magnitude of this 6-bit value determines the degree of filtering applied, with larger numbers corresponding to heavier filtering. In general, a mid-range value works well.

Note: On power-up, the CTI network initializes all sensors to apply only the low-pass noise rejection filter, and to report the actual logical sense of all raw sensor reports (without inversion). If this is the desired behavior, then no further action is necessary on the part of the user's software to configure the sensors.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.4.2 Reading the State of a Sensor

The "*Read*" command may be used to read the state of a sensor in the CTI network.

Note: The *network bridge* notifies software whenever it detects a change in the state of one or more sensors by sending a *Service Request* message to the PC (see Section 2.3 for details). This is normally the time when software would respond by reading sensor values. However, sensors may be read at any time.

The "*Read*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x11)	Sensor Address (Bits 15:8)	Sensor Address (Bits 7:0)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

If the Acknowledge byte indicates that the command was successfully processed, the network bridge then returns a second data byte to the PC. The value in bit 0 of this byte represents the latest sampled state of the selected sensor. (The value reported is equal to the state of the sensor after it has been subjected to any user-selected filtering algorithm and user-selected polarity inversion. (See "Configuring Sensors" in Section 3.4.1 for details on using these features.) been subjected to any user-selected filtering algorithm and user-selected polarity.

Byte 1	Byte 2
Command Acknowledge 0x00	Sensor State x x x x x x x 0

3.4.3 Reading the States of 4 Contiguous Sensors

The "*Read 4*" command may be used to read the states of 4 contiguous sensors.

The "*Read 4*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x12)	Sensor Address (Bits 15:8)	Sensor Address (Bits 7:0)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

If the Acknowledge byte indicates that the command was successfully processed, the network bridge then returns a second data byte to the PC. The value in bit 0 of this byte represents the latest sampled state of the selected sensor. The values in bits 1, 2, and 3 represent the latest sampled state of the next 3 sensors. (The values reported are equal to the states of the sensors after they have been subjected to any user-selected filtering algorithm and user-selected polarity

Byte 1	Byte 2
Command Acknowledge 0x00	Sensor State x x x x 3 2 1 0

3.4.4 Reading the States of 8 Contiguous Sensors

The "*Read 8*" command may be used to read the states of 8 contiguous sensors.

The "*Read 8*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x13)	Sensor Address (Bits 15:8)	Sensor Address (Bits 7:0)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

If the Acknowledge byte indicates that the command was successfully processed, the network bridge then returns a second data byte to the PC. The value in bit 0 of this byte represents the latest sampled state of the selected sensor. The values in bits 1 through 7 represent the latest sampled state of the next 7 sensors. (The values reported are equal to the states of the sensors after they have been subjected to any user-selected filtering algorithm and user-selected polarity inversion. (See "Configuring Sensors" in Section 3.4.1 for details on using these features.)

Byte 1	Byte 2
Command Acknowledge 0x00	Sensor State 7 6 5 4 3 2 1 0

3.4.5 Reading the States of 16 Contiguous Sensors

The "*Read 16*" command may be used to read the states of 16 contiguous sensors.

The "*Read 16*" command is defined as follows:

Byte 1	Byte 2	Byte 3
Op Code (0x1A)	Sensor Address (Bits 15:8)	Sensor Address (Bits 7:0)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

If the Acknowledge byte indicates that the command was successfully processed, the network bridge then returns two additional bytes to the PC. The value in bit 0 represents the latest sampled state of the selected sensor. The values in bits 1 through F represent the latest sampled state of the next 15 sensors. (The values reported are equal to the states of the sensors after they have been subjected to any user-selected filtering algorithm and user-selected polarity inversion. (See "Configuring Sensors" in Section 3.4.1 for details on using these features.)

Byte 1	Byte 2	Byte 3
Command Acknowledge 0x00	Sensor State 7 6 5 4 3 2 1 0	Sensor State F E D C B A 9 8

3.4.6 Reading the States of All Sensors

The "*Read All* " command may be used to read the states of all sensors in the CTI network.

Note: The *network bridge* notifies software whenever it detects a change in the state of one or more sensors by sending a *Service Request* message to the PC (see Section 2.3 for details). This is normally the time when software would respond by reading sensor values. However, sensors may be read at any time.

The *Read All* command is defined as follows:

Byte 1
Op Code (0x14)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

If the Acknowledge byte indicates that the command was successfully processed, the network bridge then returns N+1 data bytes to the PC. The first of these bytes indicates the number of subsequent data bytes to follow (N). The remaining N bytes carry the latest sampled states of the sensors in the CTI network. Bit 0 of the first of these bytes represents the state of the first sensor, bit 1 of the first byte represents the state of the second sensor, etc. (The values reported are equal to the state of each of the sensors after they have been subjected to any user-selected filtering algorithm and user-selected polarity inversion. See "Configuring Sensors" in Section 3.4.1 for details on using these features.)

For example, the table below shows the message that would be returned by a CTI network containing 20 sensors. Byte 1 is the Command Acknowledge byte (0x00). The value in byte #2 (binary 00000011) indicates that 3 more bytes follow. The next 3 bytes contain the states of the network's 20 sensors. In this case, since the number of sensors in the network is not equal to an integral multiple of 8, the most significant four bits in the final byte carry no meaningful values.

	Bit Number							
Byte Number	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	1
3	Sensor 8	Sensor 7	Sensor 6	Sensor 5	Sensor 4	Sensor 3	Sensor 2	Sensor 1
4	Sensor 16	Sensor 15	Sensor 14	Sensor 13	Sensor 12	Sensor 11	Sensor 10	Sensor 9
5	x	x	x	x	Sensor 20	Sensor 19	Sensor 18	Sensor 17

3.4.7 Querying for Sensor Activity

The network bridge automatically notifies software whenever it detects sensor activity using the *service request* mechanism described above (see Section 3.3). However, some users may prefer not to use this feature; instead polling the bridge periodically to check for sensor activity.

For such cases, the network bridge provides commands to disable the service request mechanism, and to query the bridge for an indication of sensor activity.

The ability of the network bridge to send service requests to the PC upon the occurrence of sensor activity may be controlled using the "*SRQ Control*" command.

The "*SRQ Control*" command is defined as follows:

Byte 1	Byte 2
Op Code (0x1C)	SRQ Control

A non-zero value in the *SRQ Control* byte disables the sensor activity service request mechanism in the network bridge. A zero value enables the service request mechanism.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

Note: On power-up, the sensor activity service request mechanism is enabled.

The "*Query*" command may be used to check for the occurrence of sensor activity.

The "*Query*" command is defined as follows:

Byte 1
Op Code (0x1D)

The network bridge replies to the command with a *Command Acknowledge* message to the PC. The network bridge then returns a second *Sensor Activity Flag* byte to the PC. A non-zero value in this byte indicates that activity has been detected on one or more sensors since the last time a *Query* command was issued. (In that case, software should use the commands previously described to read the values of the sensors.) A value of zero in this byte indicates that no new sensor activity has occurred since the last time a *Query* command was issued.

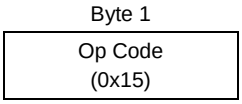
Byte 1	Byte 2
Command Acknowledge 0x00	Sensor Activity Flag

3.5 Commands for Controlling the CTI Network

The commands in this section are intended for use in configuring various aspects of the operation of the CTI network

3.5.1 Resetting the Network

The "*Reset Network*" command is defined as follows:

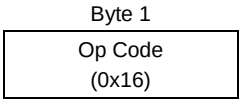


The CTI network deactivates all discrete controls and sets all throttles to a speed setting of 0, in the forward direction, with low momentum. The CTI network is then placed o*ffline.* (PC communications with the network bridge remain active.)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.5.2 Commanding the Network Online

The "*Network Online*" command is defined as follows:

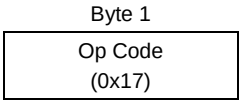


Upon receipt, the network bridge begins communicating with the CTI network.

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.5.3 Commanding the Network Offline

The "*Network Offline*" command is defined as follows:



Upon receipt, the network bridge stops communicating with the CTI network. (PC communications with the network bridge remain active.)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

3.5.4 Polling the Network Configuration

The "*Poll*" command is defined as follows:

Byte 1
Op Code (0x18)

The network bridge replies to the command with a *Command Acknowledge* message to the PC.

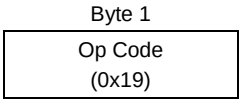
The CTI network then returns 'N+1' data bytes to the PC. The first byte returned indicates the number 'N' of CTI network modules that responded to the *Poll* command.

The remaining 'N' bytes each contain a *Module Identifier Code* indicating, in order, the type of each module found in the network. Module type codes are indicated in the table below:

Module Type	Module Identifier Code
Train Brain	1
Dash-8	2
Watchman	3
Signalman	4
Smart Cab	5
Switchman	6
YardMaster	7
Sentry	8
Reserved	9-254
Unrecognized module	255

3.5.5 Reading the Acela Network Bridge Firmware Revision Code

The "*Read Revision*" command is defined as follows:



The network bridge replies to the command with a *Command Acknowledge* message to the PC.

The network bridge then returns two bytes indicating the revision level of the Acela module's firmware as follows:

Byte	Meaning
1	Major Revision Level
2	Minor Revision Level