

Oak Tree Systems
Railroad Control Interface
HH1 Handheld Controller

User's Manual

Version 1.0
November 15, 1999

Contents Copyright © 1999 Oak Tree Systems LLC

Table of Contents

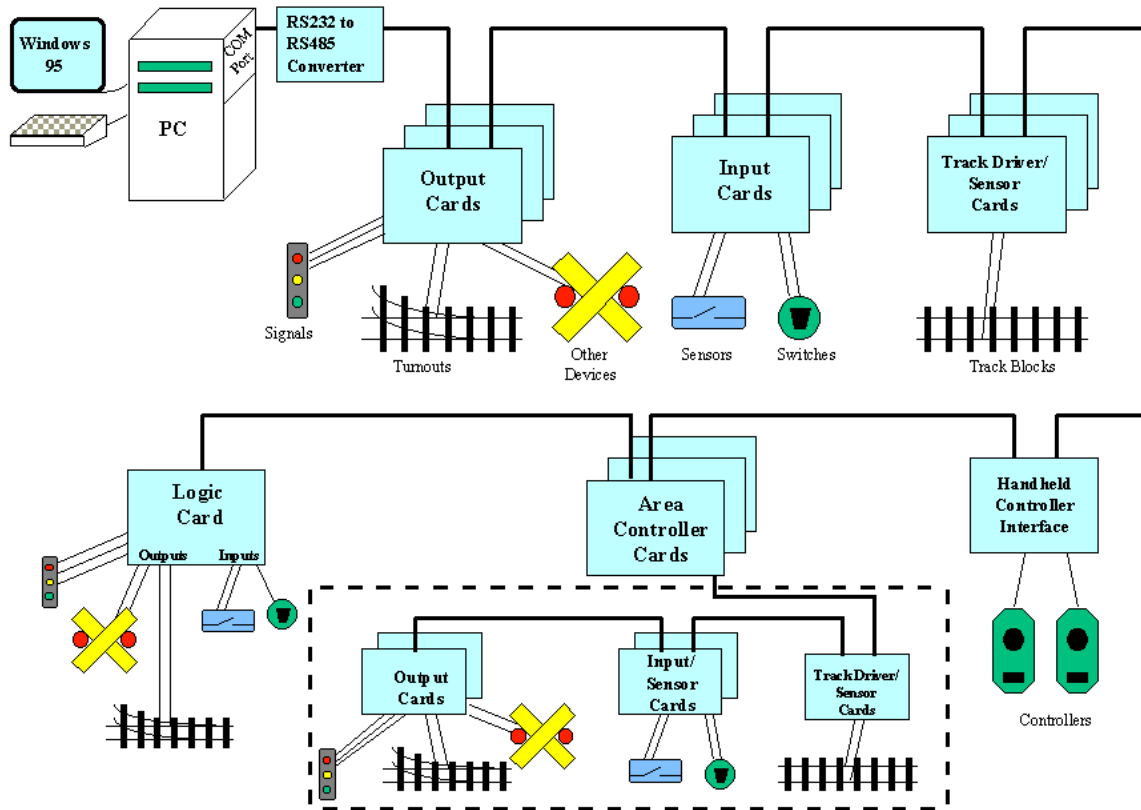
Overview of the Railroad Control Interface.....	3
Operation of the Railroad Control Interface	4
Communication Protocol.....	5
Output Messages	5
Polling Messages	7
Examples of BASIC Message Sending and Receiving software.....	8
Simple Output Subroutine.....	8
HH1 Commands	9
Overview.....	10
Handheld Controller Messages:.....	11
Command 0 - Type Inquiry:.....	12
Command 36 - Read Handheld:.....	13
Command 37 - Set Handheld:.....	14
Programming Considerations	15
Overview:.....	15
Enabling the Handheld:.....	16
Train Selection:.....	16
Braking and Emergency Stop:.....	17

Overview of the Railroad Control Interface

The Railroad Control Interface (RCI) is a family of products for controlling model railroad trains and accessories from a computer system (Personal Computer [PC] or micro-controller). The RCI family consists of several types of output cards, input cards, track driver (electronic throttle/DCC booster), and handheld controllers. Additional components to be added in the future include the Area Controller and Logic Card, which will provide control functions without the full-time use of a PC. This manual will primarily describe operation of the RCI using a Windows-based PC for control; however, examples of using small micro-controllers are also included.

The RCI is based on a multipoint, RS485 connection between a master station, and multiple slave stations. Each physical network (connected to a single COM port on the PC) can consist of one and only one master station and up to 255 slave stations. In particular, the RCI interface cards and/or Handheld Controller always act as slave devices. In later implementations, other Oak Tree Systems' devices such as the Area Controller may act as a master for their own sub-network, or may act as both a slave to the PC network and as a master for their own sub-network. Figure 1 shows an overview of the RCI system.

Diagram 1 - System Overview



Operation of the Railroad Control Interface

In order to control devices attached to the RCI, the master station sends messages to the RCI devices, and receives messages in reply. Depending on the device being controlled and the function being performed, the content of the messages will vary, but the basic formats will always be identical.

Outgoing messages from the master station to the RCI devices will always consist of 5 bytes of data. Each byte of the message can be considered to be an 8 bit unsigned binary field. Depending on the type of message, the individual bits within the byte may also have specific meanings. This message will have the following general format:

- Byte 1 - Address. Only the slave device having an address matching this value can respond to the message. The address can be any value from 1 to 255. In most currently available devices, the address is assigned permanently when the device is manufactured. Address 0 is reserved for "broadcast" messages, which may be processed by every device on the network.
- Byte 2 - Command. The command byte instructs the device to carry out a specific action. Commands are specific to each device type, and addressing an invalid command to a device will result in the command being rejected by the device and ignored.
- Bytes 3 and 4 - Data Bytes. The content of these bytes is specific to the type of command being sent. A later section explains the details of these bytes for each specific command.
- Byte 5 - Check Byte. This byte must contain the Exclusive-Or of the first 4 bytes of the message. The card will check this byte against the first four bytes received and ignore any message that does not pass this check.

Reply messages from the RCI devices to the master can be two types: a full message and a simple acknowledgement. A full message will be returned in those cases where the RCI device needs to provide some information to the master; an acknowledgement is sent in response to output messages. Full response messages will be returned in the following format:

- Byte 1 - Address. This byte will always contain the address of the device that responded to the command. Under normal conditions, this address must match the address contained in the message sent by the master.
- Bytes 2, 3, and 4 - Data Bytes. The content of these bytes will vary depending on the device and the command sent. Certain bytes or fields within the bytes may be undefined and should never be used by the master system software. These are specified in the section describing each individual command.
- Byte 5 - Check Byte. This byte will contain the exclusive-or of the first four bytes in the message. The master software must check this byte to ensure that a valid message has been received.

Acknowledgement messages from RCI devices can take two forms: positive and negative. A positive acknowledgement indicates a message was received and processed normally. A negative acknowledgement indicates that the message was received by the RCI device, but contained a command that was invalid for that device.

A positive acknowledgement consists of a one-byte return message, which contains the address of the device that is responding. A negative acknowledgement consists of one byte that contains zero. Handling of error conditions is discussed in the next section.

Communication Protocol

Communication Protocol simply refers to the sequence in which messages are sent, and the response messages that may result. The communication protocol used by the RCI is fairly simple because there is only one form of output message, and only a few possible responses. To be a complete definition, the communication protocol must also describe any errors that may occur and how the master program needs to deal with them.

The protocol definition for all RCI devices can be subdivided into two categories: output messages and polling messages. Output messages direct the RCI device to take some action, but do not require input data to be returned to the master system. Polling messages request input from the RCI device to be returned to the master. Both types of messages use the same format when the master sends them – the difference is how the RCI device responds to the message.

When a response is expected to an outgoing message, the master program must utilize a timeout function to recover from the situation where the expected response does not occur. The timeout should generally be set to 50 milliseconds, which allows sufficient time for the RCI device to receive the message and send its response. Failure to provide a timeout may cause the master software to lock up, waiting for a response that never arrives. The software examples in later sections show how to provide an appropriate timeout function.

NOTE – Before sending each message, the master program (when running on a PC) should clear any bytes of data that are waiting in the input buffer. Under certain error conditions, the RCI device may have sent bytes of data that the PC has not yet read. Clearing the input buffer ensures that these extra bytes are not utilized when the program reads the response to the message being sent. If the master is a micro-controller, this is generally not necessary because data is not buffered by the system.

After any type of error condition occurs, the master program should wait at least 50 milliseconds before sending the next message. This allows the RCI device time to clear any error condition it may have received and get set up to receive the next message. This is only necessary when an error has occurred; otherwise, the next message may be sent immediately.

Output Messages

All output messages direct the RCI device to take some action – set an output point on, change the speed setting of a throttle, flip a turnout, and so on. To perform an output message sequence, the master software needs to create a 5-byte message as described above and send it out the serial port. Software examples in later sections will show how to do this. The following diagram illustrates the possible sequence of events for an output message:

OUTPUT: Master Sends 5 Byte Output Message To Serial Port

INPUT: 1 Byte Response Read as Input From Serial Port is = Address of Device

- The message was received and processed normally
- Master Program Action – Proceed Normally

INPUT: 1 Byte Response Read as Input From Serial Port is = 0

- Either – The command sent was not valid for the RCI device to which it was directed (most likely), or

- The message was corrupted in such a way that the address or command were altered (much less likely)
- Master Program Action – Resend the message up to two times; if the error persists, either the command is wrong (indicative of a master software error), the address is wrong (the master software is sending the message to the wrong device), or there is has been a failure of the RCI device or the network.

RESPONSE: 1 Byte Response Read as Input From Serial Port is = A Non-zero Value, Which is Not Equal to the Address of the Device to Which the Message Was Sent

- A Network Error Corrupted the Original Message or the Response
- There are multiple devices on the network with the same address
- Master Program Action – Resend the message up to two times; if the error persists, it indicates a network problem. Be certain the outgoing message contains the address you intended.

RESPONSE: NO Response within .05 seconds (timeout occurs)

- The outgoing message was corrupted and was therefore ignored by the receiving device, or
- The incoming response byte was lost or corrupted, or
- The message was addressed to a device which does not exist, or
- There are multiple devices on the network with the same address
- Master Program Action – Resend the message up to two times; if the error persists, it indicates a network error or a device that is not functioning. Check Power and network connections to the device. Check the software to ensure the message is being sent to the device intended.

RESPONSE: An error occurs while reading the response (framing error or other communication-related error).

- The incoming response message was corrupted, or
- There are multiple devices on the network with the same address
- Master Program Action – Resend the message up to two times; if the error persists, it indicates a network error or hardware problem.

Polling Messages

Polling messages direct the RCI device to return some information to the master program. Unlike output messages, polling messages will result in a full 5-byte response being returned by the RCI device. The content of the message will vary by device, but will adhere to the format shown above.

When polling an RCI device, the expected response is a 5-byte message. However, under some unusual error conditions, a 1-byte response or no response at all may occur. These conditions are outlined in the sequences below.

OUTPUT: Master Sends 5 Byte Polling Message To Serial Port

RESPONSE: 5-Byte Response Read as Input From Serial Port is = Address of Device, Followed by 3 Data Bytes and Check Byte

- The message was received and processed normally, and the data requested was returned
- Master Program Action – Proceed Normally

RESPONSE: 5-Byte Response Read as Input From Serial Port, but Check Byte Does Not Match

- The message received was corrupted during transmission, or
- There are multiple devices on the network with the same address
- Do not process the message received, since it contains invalid data. Resend the message up to two times. If the error persists, it indicates a network problem or a failure of the RCI device.

RESPONSE: First Byte Read as Input From Serial Port is = 0. Although the master program will be attempting to read a 5-byte input, additional bytes will probably not be received because the RCI device only actually returned one byte.

- The command sent was not valid for the RCI device to which it was directed (most likely), or
- The message was corrupted in such a way that the address or command were altered (much less likely)
- Master Program Action – Resend the message up to two times; if error persists, either the command is wrong (indicative of a master software error), the address is wrong (the master software is sending the message to the wrong device), or there is has been a failure of the RCI device or the network.

RESPONSE: First Byte Read as Input From Serial Port is = a Non-Zero Address but is Different from the Address of the Polled Device.

- A Network Error Corrupted the Original Message or the Response, or
- There are multiple devices on the network with the same address

- Master Program Action – Resend the message up to two times; if the error persists, it indicates a network problem. Be certain the outgoing message contains the address you intended.

RESPONSE: NO Response within .05 seconds (timeout occurs)

- The outgoing message was corrupted and was therefore ignored by the receiving device, or
- The incoming response message was lost or corrupted, or
- The message was addressed to a device which does not exist, or
- There are multiple devices on the network with the same address
- Master Program Action – Resend the message up to two times; if the error persists, it indicates a network error or a device that is not functioning. Check Power and network connections to the device. Check the software to ensure the message is being sent to the device intended.

RESPONSE: An error occurs while reading the response (framing error or other communication-related error).

- The incoming response message was corrupted, or
- There are multiple devices on the network with the same address
- Master Program Action – Resend the message up to two times; if the error persists, it indicates a network error.

Examples of BASIC Message Sending and Receiving software

The following is an example of implementing a simple send and receive routine using Visual BASIC. Other BASIC languages will probably provide similar functionality: The routines are shown in the form of low-level subroutines. These subroutines are designed to have the output message passed to them in a parameter list, and also to return results to the calling program in the parameter list. The subroutines are presented with increasing degrees of complexity, starting at a very basic level, and then adding error checking and handling functions in later versions.

Simple Output Subroutine

This routine sends a 5 byte message to an RCI card, then reads a one-byte response to get the acknowledgement back.

**Sample software will be included in the next version
of the document**

The following routine sends a 5-byte polling message to an RCI card, then reads the 5 byte message that is returned.

**Sample software will be included in the next version
of the document**

HH1 Commands

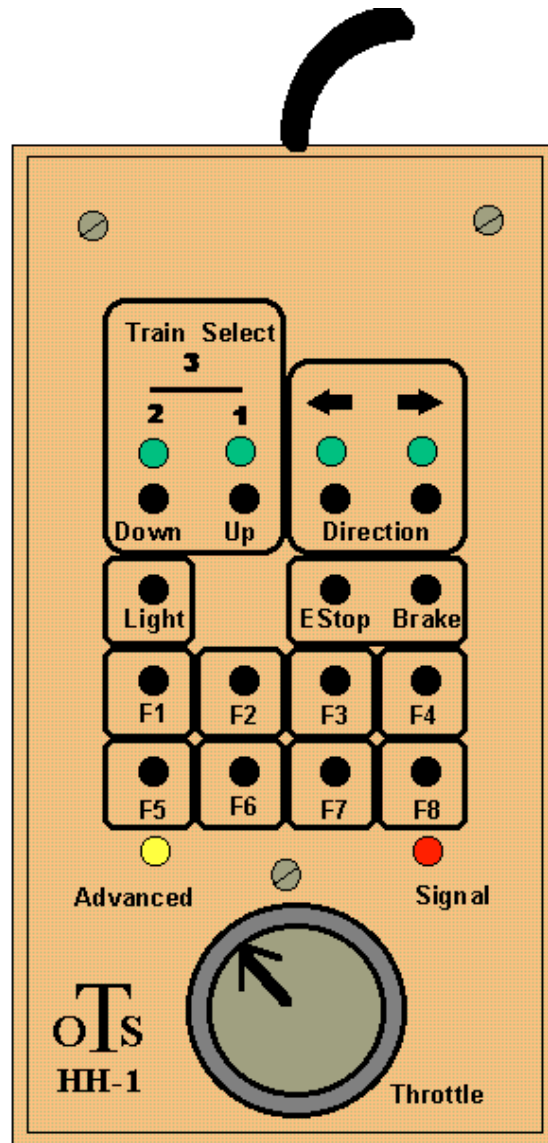
The following pages describe the various commands that are implemented on the HH1 Handheld Controller. Because Handheld Controllers may be plugged and unplugged from the network at various points in time during operation, special consideration must be given to the interaction between the PC and the Handheld, and the errors that can result from disconnection of the handheld during operation.

In the following pages, the details of each message format are presented, along with information on the actions that the device will take in response to receiving the message.

In the descriptions that follow, the term “Normal 1 Byte ACK” refers to the device’s 1 byte response that contains the address.

Overview

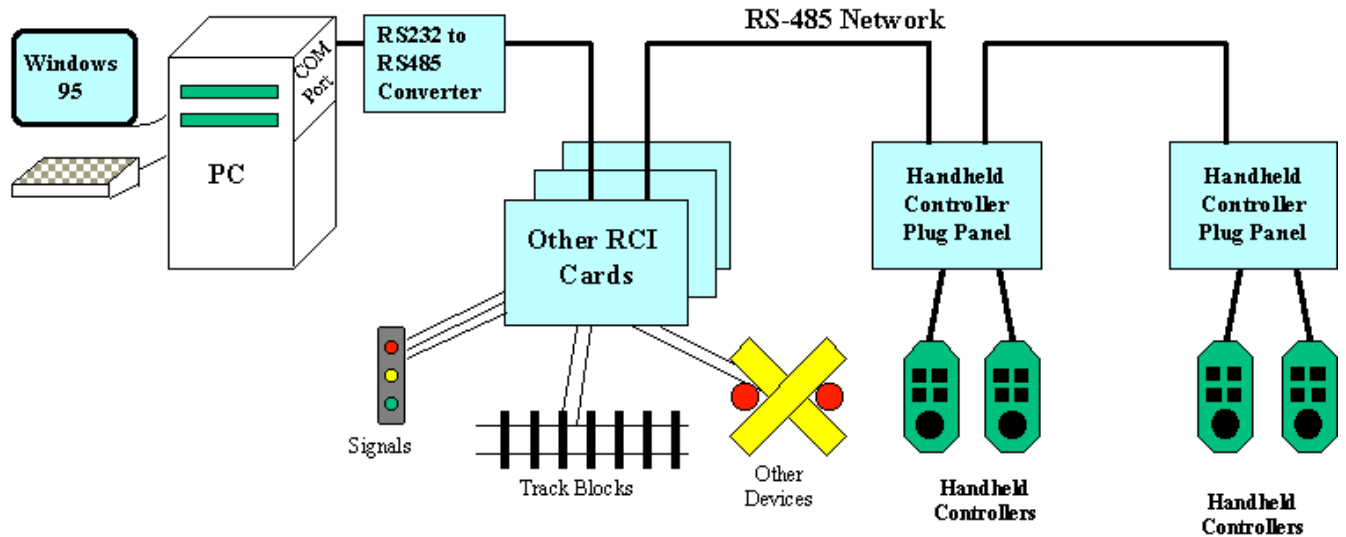
The RCI system allows use of a handheld controller for operating trains. A typical handheld controller is shown below. Other variations on this general design may also be available in the future.



Handheld Controller

The plug-in panel is connected to the same RS-485 interface and power supply as the rest of the RCI cards. The handheld controllers are then attached to the plug-in panel at a convenient location to operate the train. Multiple plug-in panels can be placed where needed around the layout. Communication with the handheld controllers is performed directly by the PC or other master station

Diagram A - Direct Handheld Connection



Note - When calculating the total length of the RS-485 network, each handheld controller should be counted as 20 feet toward the 1000 foot total wiring limitation.

Where cost needs to be minimized, the handheld controllers may be connected to the same network as the other RCI cards. Because it is desirable to read the controller input frequently, this may adversely affect performance of other RCI cards attached to the same network. Providing a dedicated serial port and network for the handheld controllers provides highest performance, allowing the support of a larger number of controllers.

Handheld Controller Messages:

The handheld communicates with the PC in much the same manner as the other RCI cards. Two commands have been defined specifically for the handheld. These messages are defined on the following pages. The handheld also responds to a "Send Type" command (command code 0). All other command types are invalid for the handheld controller.

Command 0 - Type Inquiry:

Function - This command causes the device to respond with its type and version information. This command is used to identify the cards attached to the network.

Message Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Card Address	Command	Not Used	Not Used	Check Byte
AAAAAAAA	0	XXXXXXXX	XXXXXXXX	CCCCCCCC

X – Not Used

Action Taken - The device sends a response message in the format shown below:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Card Address	Not Used	Card Type	Card Version	Check Byte
AAAAAAAA	XXXXXXXX	WXTTTTT	VVVVRRRR	CCCCCCCC

XX - Not used.

W - Watchdog timeout - set on when a card resets itself due to an internal error. Indicates device or firmware malfunction.

TTTTT - The card type code of the attached card (5 bits)

VVVV - Firmware Version (4 bits)

RRRR - Firmware Release Level (4 bits)

Command 36 - Read Handheld:

Function – This command reads three bytes of information from the handheld controller. Note that all key inputs are ‘latched’ in the handheld until they are read by this command, i.e. any button that is pressed will be remembered by the handheld until the value is read by the PC. However, the throttle setting and direction values are the current settings at the time the read is performed. Once the message has been transmitted, the latches are reset unless the key is still pressed. In the case of the Train Select Up and Down inputs, the keys are disabled for 0.5 seconds after either of these keys are pressed. This prevents “run-away” stepping of the train selection. At most, the PC will see one of these inputs every 0.5 seconds, even if the operator continues to press the key.

Output Message Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Address	Command	Not Used	Not Used	Check Byte
1111AAAA	36	XXXXXXXXXX	XXXXXXXXXX	CCCCCCCC

AAAA - Address: Handheld controllers always have an address between 240 and 255; therefore the four high order bits must be 1111.

X - Ignored

Returned Message Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Card Address	Flag Bits	Function Keys	Speed Setting	Check Byte
1111AAAA	ABCDELTT	FFFFFFFF	0SSSSSSS	CCCCCCCC

Byte 2:

A - All Stop indicator is on (Emergency Stop and Brake Pressed Simultaneously)

B - Brake button was pressed

C - Connected Bit - This bit is always on in the HH1

D - Direction Indicator; 1=forward, 0=reverse

E - Emergency Stop button was pressed

L - Light Function Button was pressed (FL in DCC system terminology)

TT - Train Select Bits; Bit 0 means move to next lower selection (Down Pressed), Bit 1 means move to next higher selection (Up Pressed)

Byte 3:

FFFFFFFF - Functions 8 through 1 were pressed (low order bit is function 1, high order bit is function 8)

Byte 4:

0SSSSSSS - Current Throttle Setting, speed from 0 to 127. High order bit is always 0.

Action Taken: The three data bytes are retrieved and returned to the PC in the format shown.

Command 37 - Set Handheld:

Function – This command sets the state of the Train Select Indicator, Signal Indicators, and Direction on the handheld.

Output Message Format:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Card Address	Command	Train Select	Signal	Check Byte
1111AAAA	37	EXXDXXTT	XVVVXSSS	CCCCCCCC

AAAA - Address: Handheld controllers always have an address between 240 and 255; therefore the high order bits must always be 1111.

Byte 3:

E - Enable Handheld - Setting this bit on removes the handheld from the standby state and allows normal keypad input and output displays to function.

X - Ignored

D - Sets the State of the Direction Indicator. 1=Forward, 0=Reverse. Normally the PC should not change the direction set by the operator, so this field should contain the same direction that was in the last message received from the handheld. However, in the case where the operator has just selected a new train, this field should be set to the last known direction for the train just selected.

TT - The state of the train select indicators. Bit 0 is the right indicator; bit 1 is the left indicator. Together they can indicate selection 0, 1, 2, or 3.

Byte 4:

X - Ignored

VVV - The state of the Advanced Output Signal. This multicolor LED is normally used to indicate track conditions in the upcoming block, however the signals may be used for any desired purpose by the PC software. The values of VVV are defined as follows:

000 (0) - Off

001 (1) - Green

010 (2) - Yellow

011 (3) - Red

100 (4) - Blinking Green

101 (5) - Blinking Yellow

110 (6) - Blinking Red

111 (7) - Alternating between Red and Green

SSS - The state of the Output Signal. This LED is used to indicate track conditions in the current block. The same codes are used as VVV above.

Returned Message Format: Standard 1 byte Ack

Action Taken: The Train Select indicators, Direction indicator, Signal output, and Advanced Signal output are set.

Note - the two signal outputs may be used to indicate special conditions to the operator. For example, when using Railroad and Company software, the indicators are also used to help the operator set his throttle to match the speed of the train after the operator has just plugged in his handheld or switched to a new train.

Programming Considerations

Overview:

The PC needs to deal with the fact that the handheld may be plugged in or unplugged at any point in time. This places several responsibilities on the PC software:

- Determining when a handheld is plugged into the network - to do this, the PC must occasionally attempt to read input from the handheld. If the attempt is unsuccessful, it can be assumed that the unit is unplugged. If the handheld responds, the PC can process the data it receives and begin to read data from the handheld at a higher frequency. If the handheld does not respond, the PC should wait a short time (2 to 3 seconds) before trying it again. The only detrimental effect of this strategy is that there will be a short delay between the moment the handheld is plugged in and when the PC actually starts responding to it.

Each handheld has a unique address. Because there are 16 possible addresses, the PC software should be configurable so that it only looks for handhelds that have been defined, rather than attempting to contact all 16 possible addresses. Each handheld can be set to one of the 16 addresses, from 240 to 255. This address must not conflict with any other handheld controllers on the same RS485 network.

- The PC must determine when a handheld has been unplugged from the network. The PC should be programmed to “time out” whenever it attempts to read a response from a handheld (or any RCI card). If a byte is not received within 30 milliseconds, the handheld can be treated as disconnected. Note that this timeout should be used each time a byte of the input message is read, since the handheld may be unplugged in the middle of transmitting its 5-byte message.
- The PC must deal with other transmission errors. Unplugging the handheld while it is transmitting may cause other types of errors:
 - Framing error - The serial interface will report a framing error when it does not receive a stop bit for any byte.
 - Check byte error - The check byte is the 5th byte of each message. It should be equal to the bit-wise exclusive-or of the first four bytes of the message. If the check byte does not match, the message must be discarded and not processed.

Because errors of the types listed above are expected as a normal part of operation of the Handheld Controllers, they should not normally be logged or brought to the user's attention. The software may need to provide some other means to test a Handheld Controller to ensure that it is operational.

The PC must also decide what is to be done if a handheld is disconnected. One suggested strategy is to have the train gradually slow down as if the throttle were set to zero and the train's momentum is carrying it. If the handheld were not plugged in within a reasonable time, the train would gradually come to a halt. Of course, the train must obey signals and avoid collisions during this period as well. It may be desirable to allow the user to configure the rate of deceleration during this interval.

Since the handheld loses power when disconnected, the state of the train selector and signal outputs are lost. When the PC recognizes that a handheld has been reconnected, it should immediately send a “Set Handheld” message (37) to the affected unit. Upon being plugged in, the handheld will initially display alternating red signals and the train select indicators will be turned off. See below for more information on train selection.

Enabling the Handheld:

When first powered up, the handheld is placed in a disabled state. In this state the signal outputs alternately flash red and inputs are ignored. When the handheld receives a "Set Handheld" message with the Enable bit on, all lights are set to the indicated states and the input functions may proceed normally. Sending a message with the Enable bit off disables the handheld once again and causes the signals to begin flashing. This is intended to inform the operator that the PC has not assigned him a train to control. All messages sent to an active handheld must have the Enable bit set on to ensure normal operation.

Train Selection:

The handheld unit supports the operation of multiple trains by the same operator. The state of the "Train Select" indicators on the handheld shows the user which train is currently being operated. The selector indicators may be set by the PC using message 37. Unlike the Direction buttons, the user's pressing the Train Select Up or Down buttons does not directly change the state of the indicators. It is up to the PC to determine what indicators should be lit and set them accordingly.

The PC software should allow the user to establish a relationship between a list of dispatched trains and the operator's handheld. The operator may then move up and down through the list by pressing the UP or DOWN buttons to select the train he wishes to operate. The indicators on the handheld can be used to remind the operator which train he is currently controlling. If the operator is assigned more than three trains, the indicators should be set off to remind the operator that he is running one of the higher numbered trains, and that he should refer to the computer for the train assignment. The UP and DOWN keys could still be used to step him through the list of trains, regardless of how long the list is.

If the train selector on the handheld was set to 1, 2, or 3 at the time the unit was unplugged, the PC should send that setting to the handheld as soon as it recognizes that the handheld is once again plugged in. The handheld will always show both train indicators off when it is first plugged in.

As in the case of a disconnected handheld, the PC must decide what action to take when the operator selects a different train. The following two points are suggested:

- First, the PC should set the proper direction into the handheld. The PC may also be programmed to use the signal outputs to guide the operator in setting his throttle to the proper setting. One way of doing this would be to set one of the signals to blinking red to indicate which way to turn the throttle. When the throttle reaches the right range, the PC can set the signals for normal operation.
- Second, if the previously selected train was in motion, it should be treated in the same manner as a train whose handheld has been disconnected, i.e. it should gradually slow to a stop until some valid input is received for that train.

Braking and Emergency Stop:

The Brake and Emergency Stop Buttons work as follows: Whenever the Brake Button is pressed, the Brake bit in the Handheld's input message is immediately set on. As long as the Brake button is held down, this bit will continue to be sent to the PC. The PC must determine the appropriate rate of braking since there is no variable brake input on the handheld. It may be desirable to allow the user to configure this value for each train.

When the E-STOP button is pressed, the Emergency Stop bit is set on and sent to the PC on the next read. A typical response would be to have the PC stop ALL trains assigned to the affected handheld controller. The PC software should determine how long the Emergency stop condition remains in effect and what the procedure is for resetting operations back to normal.

If BOTH the E-STOP button and the Brake button are pressed simultaneously, the handheld will set on the All Stop bit in the input message. The PC should interpret this to mean that all activity on the layout should halt at once. Because this is very disruptive to operation of the railroad, it may be desirable for the PC to determine that this condition persists for some period of time (say ½ second) to ensure that it was not caused by an accidental pressing of the buttons. Again, it is up to the PC program to determine what the procedure is for restarting operations after an All Stop condition occurs.