API information

keene electronics

Contents

For the KIRA modules

Overview

There are two main ports used in communicating with the KIRA modules; the IR port, which can be changed by the user, and the discovery port, which is fixed.

The IR port defaults to 65432 and normally you would only change this if you were using port forwarding with a number of modules. The discovery port is 30303.

Some modules can store a limited number of IR codes internally. These when sent are always sent via the target.

It is not necessary for a code to be stored in the module. Code can be sent from a PC or HHD and acted upon immediately. All modules listen constantly on the IR port for an IR data string of the correct format, which they will then process. Just send in the correct format, by this means large IR code libraries can be built and used by a PC or network enabled device.

See further down this document for an explanation of the code string format and "to send an IR code string"

The modules can receive an IR signal, convert to ASCII and send to two destinations in real time. The destination IP address 1 (target) and destination IP address 2 (PC).

KIRA Utility

The KIRA utility provides a quick way to see what the module is doing and to find them on the network

With the Kira utility most of the commands can be tested by typing the command into the code window and clicking the test button. The test button will send whatever is in the window to the current module on the UDP data port. It can also be used to acquire IR codes, to update firmware.

The utility can be downloaded from http://www.keene.co.uk/pages/downloads/dnl_files/IRAnywhere.zip

Using the Kira utility with WireShark network monitor gives a developer a good starting point as it will allow you to see what the utility is sending and what is coming back from the module.

WireShark is nothing to do with us and it is free from here

http://www.wireshark.org/download.html

Commands

General commands

K (string) convert this string and and emit as IR - now!

cmdI set destination address 2 (PC) address my IP address (send to me)

cmdP set **P**ort (caution! will lose connection)

cmdM set **M**ode. (Receiver, Target or StandAlone)

cmdT **T**ransmit a stored IR code

cmdR **R**eceive a code for storage.

cmdU Update stored memory

cmdB re**B**oot module

cmdD change **D**hcp setting

cmdC **C**onfigure IP addresses and hostname

Discovery command list

These all take place on the discovery port of UDP 30303

Commands received by modules

disD Standard **D**iscovery command

disC **C**onfigure command

disI Request for **I**P address of a target

disN Request a list of stored IR code **N**ames.

disS **S**et target address (receiver only)

Responses sent

General discovery response.

disS Response to target address request, unicast.

disR Response to request for IR code list.

To send an IR code string to a module (emit ir)

To send an IR code string to a module you send a UDP packet to the IP address of the module with both the source and destination ports set to the IR port. If the string is in the correct format the module will process the string and output as IR. If successful it will return “ACK” to the senders IP address on the same port. Please note that routers do not guarantee delivery of UDP packets so on a busy network you may sometimes lose a packet (Although this is rare in practice).

To receive an IR signal on a PC or HHD (learn ir)

The module can send a string representation of an IR signal to two destinations a “Target” typically another module for real time IR distribution, and a PC or HHD. A receiver module can send to both destinations while a stand-alone module or target can send to one (Target or StandAlone cannot send to another target). Modules are normally pre-set to Target, Receiver or StandAlone, but can be changed from the option web page or via an application

To avoid unnecessary net traffic modules don't attempt to send to the PC unless the option 'find PC' is checked on the options web page.

A developer can send a command to the module to force a change to the destination PC IP address, this also enables the find PC option.

For an application to learn IR you would need to have a module configured to send the ASCII text representation of the IR signal to your IP address, where your app can process and store it.

When learning IR, have the remote control a normal operating distance from the receiver module I.e. about 3 - 4 meters. They will give distorted timings if you have the remote too close.

Much of the necessary set-up can be done from the web page for the module, but it can also be done programmatically.

You need to have a module configured as either a receiver or stand-alone. If need be this can be done with the cmdM command.

Next you need to direct the second output nominally referred to as 'PC', to you. This is done with the cmdI command.

You should now be able to receive and learn IR codes. They will be received on the UDP IR port in the same format as you would send to a module. Once received you can send to a module to test.

Command Details

Commands are sent as ASCII text to the IP address of a module on the IR port, all commands will respond with an ACK if they are received OK.

cmdI **I**P address - set the second destination (PC) address as my IP address (send to me)

Just send ASCII text 'cmdI' to the IP address of a module to set its destination PC IP address to the IP address of the sender.(your app)
Send with both source and destination ports set to the IR port - default is UDP port 65432

cmdP **P**ort set

Send ASCII text 'cmdP12345' to the IP address of the module to set that module's IR port to 12345. Note all subsequent communication will need to be on the new port!
Send with both source and destination ports set to the IR port - default is UDP port 65432. Use the web page or discovery to find port if you lose it. The port number is in decimal and max is 65535

cmdM Mode set.

Send ASCII text 'cmdM0' to the IP address of a module to set it as a receiver. 'cmdM1' sets as a target and 'cmdM2' sets as stand-alone. You can also send cmdM3 to the module. This has the same effect as sending cmdM2 but will first copy the MAC address string, excluding the colons, to the hostname. This gives a very easy way of creating a unique host name for each module. The mode is then set as "2" stand-alone.
Send with both source and destination ports set to the IR port - default is UDP port 65432

cmdT Transmit a stored IR code

Send ASCII text 'cmdT001' to the IP address of a module to send the code stored in location 1 ('cmdT002' to send code 2 etc.). If no code is stored in that location nothing will be sent. 3 digit decimal, and all 3 digits must be sent.
Send with both source and destination ports set to the IR port - default is UDP port 65432.

cmdR Receive a code for storage.

Send ASCII text 'cmdR01+code' to the IP address of a module to cause the attached code to be stored in that module and in location 1

This is sent as 2 digit hex and both digits must be sent.

Send with both source and destination ports set to the IR port - default is UDP port 65432

Example using the code from stored code section further down this document
Send
cmdR1#Sny Cmdr Stp:K 280D 096A 0269 025C 025C 025B 025E 025B 025D 04B4 025E 04B4 025D 025B 025E 025B 025D 04B5 025E 04B4 025E 04B3 025E 025B 025D 025B 2000

Note the maximum length of code for storage in a standard Kira is 49 bits. Longer code can be handled but not stored. Some special versions of the Kira128 can handle and store codes over 128 bits in length. Typically for Air-conditioning units. (Kira47)

It is also possible in most modules to use an http request to mimic the action of a button press.
The web page for the request will be either code.htm or remote2.htm (earlier modules)

so if the module is on 192.168.1.45

http://192.168.1.45/code.htm?button002

will mimic button 2 being pressed and the contents will be actioned. Note the 3 digit decimal location number.

cmdD Dhcp on or off

Send ASCII text cmdD0 to turn off dhcp or cmdD1 to turn it on. Send to IP address of module on UDP IR port

cmdU Update stored settings

Send ASCII text cmdU to the IP address of the module on the current UDP IR port.

To update all stored setting now.

The actual update will take place 2 ½ seconds from the command being sent, as it has to refresh the web page first.

Note:

Writing to the flash ROM in the microcontroller can only be done in large chunks, consequently all IP addresses and other setting are in one chunk and are all written at the same time. Any memory update will save all of these setting

cmdB re**B**oot

Send ASCII text cmdB to the IP address of the module on the UDP IR port to force a re-boot of the module. The unit will start re-boot 2 ½ seconds from the key press, as the module first has to serve the web page back to the user.

Note:
If using dhcp there is no guarantee that the module will be re-assigned the same IP address, although it usually will – allow about 30 seconds for the dhcp handshaking to take place before trying to access the module.

cmdCX

This is a general configure command for setting the IP addresses and hostname on the module

All have the same basic format cmdC +<identifying letter> + (String) <argument> and are sent to the IP address of the module on the UDP IR port.

Example subnet mask. (note no spaces or termination char)

Send "cmdCS255.255.0.0" to set the subnet mask to 255.255.0.0

cmd**CI** = Modules own **I**P address (caution will lose contact)

cmd**CS** = subnet mask

cmd**CG** = gateway

cmd**C1** = DNS 1

cmd**C2** = DNS 2

cmd**CT** = target

cmd**CP** = PC

cmd**CH** = hostname

cmd**CA** = MAC Address

Notes:

When updated these IP addresses are held in ram. You would need to send cmdU to save the addresses to ROM and make permanent

If DHCP is on, it is not possible to update the IP address for the module, the subnet mask, the gateway, and DNS server 1 and DNS server 2. This is because as soon as you change them they are re-set by the DHCP client in the module.

The IP address string is converted to IP address bytes by the micro in the module, invalid strings will simply be ignored although an ACK will still be sent if the command was actually received OK.

The hostname can be any ASCII chars equal to or above a space char 32 and below char 126. Any others are overwritten by a space. Note also that there are certain chars that are not legal in a hostname. for example the colon or backslash, forward slash etc.

The MAC address is the six bytes each as 2 digit ASCII separated by colons. E.g. "cmdCA00:12:C2:90:AB:02" If the Mac address is changed the module will save settings and re-boot.

General notes on changing configuration

These modules will respond to http://hostname where hostname is the hostname of the module. But that gets stored in the PC's network cache and if the IP address or MAC addresses are changed, the cache will still contain the incorrect values and will need to be flushed. Typing 'nbtstat -R' into a command window should do the job. This can be a problem when using the web page to access the units after a re-boot if the web page is called using the hostname. This is also true if the DHCP server changes the module's IP address.

Firmware upload

The firmware can be upgraded via the network. It is done via TFTP on UDP port 69 the usual one for TFTP.

It can be done from the command line in most operating systems along the lines of

TFTP <module IP address> put <file path>

Our KIRA utility java app also does a TFTP upload. And if you require the source for that just drop us an email. The java app will also try and detect the module's Mac address and current mode (receiver, target or stand alone) and modify the hex file so that the module is re-programmed to these values. If you use a command line TFTP program to upload the firmware you will need to ensure the mode and MAC addresses are set to the correct values as they are all programmed the same in the firmware. Note: most modules now use a dedicated MAC address chip to provide a unique MAC address. In these cases the firmware MAC address is simply used as a backup and will not normally be seen.

Find Modules

Discovery commands are sent as ASCII text on the discovery port 30303

disD Discovery.

The modules have a discovery function built in.

To find which modules are on the network you would need to send a UDP packet to port 30303 with disD as the first 4 chars and with a broadcast IP address (255.255.255.255). This packet gets sent to any device on your local network.

Each device will respond with a sixteen-line answer, a CrLf follows each line

The received lines are:

Hostname
IP Address
MAC address
mode of unit
IR port
last IP address
Subnet mask
Gateway address
DNS 1 address
DNS 2 address
Target address
PC address
DHCP state
DDNS state
Firmware version
Text description of answer type

Example:

RECEIVER
192.168.100.3
00:50:C2:96:D0:00
0 = receiver
65432
192.168.1.32
255.255.255.0
192.168.100.77
194.74.65.68
194.74.65.69
192.168.100.1
192.168.100.2
DHCP = on
DDNS = off
1.60
Discovery reply

The modules also produce a discovery packet on re-boot or when the dhcp server assigns a new IP address

A developer can use this to get the IP addresses, MAC address, mode or port of any connected modules

Remember the user can change the host name! This defaults to Receiver, Target or standAlone but in a multi module system would typically be given meaningful names such as kitchen bedroom1 lounge etc. So that the user can access the module's web page via http://kitchen etc.

disC Configuration

Can be either sent as broadcast to change all modules or unicast to change just the one

Send disC as ASCII text to the module(s) on port 30303.

This moves the IP addresses on block to the subnet of the sender. This should only be used on a local network! It would typically be used when DHCP is not available or is disabled.

It changes the first 3 bytes of the IP address of gateway, 1st DNS, target address and the modules own address to match the first 3 bytes of the address from which the command was sent. All four bytes of the PC address are changed to match those of the sender.

Example

Old addresses

Gateway	192.168.1.1
Module own IP	192.168.1.31
Destination PC	192.168.1.2
Target IP	192.168.1.32

Assume the modules are connected to a network 192.192.8.xxx based with a PC at 192.168.8.55, send "disC" from the PC on port 30303 to set first 3 bytes of IP addresses and four bytes of PC IP address. Note all four bytes of the PC address are updated, so that the module can reply and send to you.

New addresses

Gateway	192.192.8.1
Module own IP	192.192.8.31
Destination PC	192.192.8.55
Target IP	192.168.8.32

You will also get the normal discovery response followed by a re-boot discovery response as the module re-boots after saving the new addresses.

disI IP address - search for IP address of a target

Broadcast on the usual port for discovery tasks, only a target or stand-alone module will respond to this. The module will respond with a disS.

Send disI as ASCII text to the broadcast IP address on port 30303.

disS Send IP address Update Target address

Send disS192.168.1.31 as ASCII text to the IP address of the requesting module on port 30303.

Is sent by the module in response to the disI command, (although it could be sent by a PC) to the unicast address of the device that issued the request above, it has the format disI plus module's own IP address as an ASCII string e.g. disI192.168.1.31. Strip off the 1st 4 chars to get the IP address as a string. This command is acted upon by a receiver module and has the effect of setting the receiver's target address to the IP address of the sender. To discover each other the modules make use of this command.

disN Request a list of stored IR codes **N**ames

Send disN as ASCII text to the module(s) on port 30303.
Sent as unicast to 1 module or broadcast to request a list of stored IR codes

disR Response to the above names request.

The First line is disR to identify as a response to request
Second line is IP address of sender in case request was broadcast
Third line is name of code stored in location 1
Fourth line is name of code stored in location 2
Fifth line is name of code stored in location 3
Sixth line is name of code stored in location 4

IR Code Format

This code format is a simple ASCII text representation of the IR code. The first character in the string is a capital "K" followed by a space. The next four characters are the frequency expressed as a 2-digit hex number and number of bursts of carrier including the lead in burst followed by a space.

The remaining blocks of four ASCII characters are the times in uS in hex of the burst then no burst. All are separated by a space.
The final value is always 2000 and is the lead out no-burst time.
An example here is 12 bit Sony code
K 280D 097A 0253 0272 0241 04C9 0240 0273 0241 0272 0240 04CA 0241 0273 0241 0271 0242 04C9 0241 0272 0240 0273 0240 0273 0240 0273 2000
K followed by a space = identifier

28 = decimal 40 = carrier frequency for Sony
0D = 13 = 12 bit data plus 1 for the lead in burst pair
The rest are times in uS
097A = decimal 2426 = 2426uS lead in burst of carrier
0253 = decimal 595 = 595 lead in no carrier or space-time.
Etc. till end of string, ignoring the K 280D there should be 13 (0x0D) x 2 blocks of four digits

Upload Code

To format a code for upload to the module for storage, add '#' name-string and a ':' (colon) to the code string.

The name can be no more than 14 chars long.

#Sny CmCdr Stp:K 280D 096A 0269 025C 025C 025B 025E 025B 025D 04B4 025E 04B4 025D 025B 025E 025B 025D 04B5 025E 04B4 025E 04B3 025E 025B 025D 025B 2000

When accepted by the module the leading # is stripped off and the text between the # and the colon is set as the name. Codes are uploaded using the web page or via an application with the cmdR command.

NOTE: The IR codes are rarely exactly the same. This is caused by a variety of reasons but is mainly due to the IR receiver chips themselves. IR receiving routines have a wide tolerance and if the expected value is for example 600uS they will typically accept anything within a window of 540uS to 660 uS or even wider.

So if yo use a Kira as a source for code matching you will need to match the code using a windowed technique rather than absolute values. In the Sony example above both the 0272 and 0240 should be 0258

But they easily fall within the required window of 540 - 660.

&H0258 = 600
&H0240 = 576
&H0272 = 626

Routing and Repeats

Routing and repeats are combined into one string suffix

The routing part only relates to the IR Commander Matrix or KiraCC. The format is the same for either and the IR string can be used across the KIRA range. It is simply ignored by modules that do not require it. All IR modules now accept the repeats

It is now possible to tell the module to repeat the code a specified number of times with a specified gap (pause) between them.

Looking at a typical IR code string - this one is for a Sony

K 280D 096A 0269 025C 025C 025B 025E 025B 025D 04B4 025E 04B4 025D 025B
025E 025B 025D 04B5 025E 04B4 025E 04B3 025E 025B 025D 025B 2000

To add repeats and routing to that string requires an additional string to be added at the end

K 280D 096A 0269 025C 025C 025B 025E 025B 025D 04B4 025E 04B4 025D 025B
025E 025B 025D 04B5 025E 04B4 025E 04B3 025E 025B 025D 025B 2000 00 4000 2

The "<space>00<space>4000<space>2" tell the module to send an additional 2 repeats of this string ie 3 in all and use a gap of 16384 uS between them. 4000 hex or about 16mS.

Note for Sony the gap between the codes needs to be adjusted according to how long the codes takes to send.

So for standard Sony code (12bit) the gap can be about 4800 seems to work OK but for the 20 bit BluRay code a gap of 3000 works. It is best to send Sony code 3 times for the original codes and 6 for BluRay etc. Most other systems are happy with 1 send.

e.g Suffix for Sony Blu-ray 00 3000 5

For the routing element the first 00 (hex) is only applicable to the Matrix (4 ports) or the KiraCC (2 ports) and is a bit screen with the lower four bits (KiraCC 2 bits) setting which output ports the IR will be routed through. A value of

“00” is ignored and only the repeats acted upon. So “11110011” will route the IR through ports 1 and 2 only.

for example
“<space>00<space>4000<space>2” the 00 contains no routing so the string is sent to all available output ports
“<space>01<space>4000<space>2” the 01 instructs the module to send the string to port 1 only
“<space>03<space>4000<space>2” the 03 instructs the module to send the string to port 1 and 2 only
“<space>0B<space>4000<space>2” the 0B instructs the module to send the string to port 1 and 2 and 4 only

Specific Commands for the KiraCC

UDP commands on UDP data port (65432 default)

Note: With the Kira utility most of these commands can be tested by typing the command into the code window and clicking the test button.. The test button will send whatever is in the window to the current module on the UDP data port.

The utility can be downloaded from http://www.keene.co.uk/pages/downloads/dnl_files/IRAnywhere.zip

KCC Kira CC relays

A simple command structure allows control of the relays. The relays can be set to open, to close or pulse for a number of cycles. Each cycle is 1/4 second.

KCC<space>R<relay number><action>

<action> can be either OP for open, CL for close or PLS for pulse. If pulse a duration of the pulse must be given...

KCC R1CL = close relay 1
KCC R2OP = open relay 2
KCC R1PLS0A = pulse relay 1 for 10 periods (change from open to closed for a duration of 2.5 seconds then open again)

The number of pulses is in 2 digit hex, so 0A = 10 in above example.
When a relay is pulsed it simply changes its current state for the period. So if it is closed it will open for the period then close. And if it is already open it will close for the period then open.

To query the current state of the relays
KCC R1? it will query the state of relay1 and return KCC R1CL (closed) or KCC R1OP (open)

Same for relay 2

Read back of memory contents

cmdSxxx can be used to read back the contents of a memory location.

E.g. send to the module cmdS123 where the ‘123’ is the memory location you want to read back. The module will return the contents as ASCII text on the UDP data port, followed by a #ACK

The location number is in decimal and must be 3 digits in length so to read location number 3
send cmdS003

RS232 data format

“R”<space> <number data bytes> <space><data><optional CrLf>

The number of data bytes is a 2 digit ASCII hex number – if just a few bytes the leading ‘0’ must be included i.e. ‘02’

The RS232 data is whatever you require out of the RS232 port. So to get ASCII out send ASCII. Most devices actually require ASCII data and this way the data is just sent as it is required. If you require binary data then the program you use to generate the data will have to have a way for you to enter the data, but it will be accepted by the KIRACC.

The data can be either sent to the module on the UDP data port for immediate execution or stored into a memory location.

e.g. all lights on for an RS232 to X10 converter

data required by the converter is ASCII and is:- $>28001A01 AONFD#

The number of data bytes is 17 or &H11 - So string sent is

R 11 $>28001A01 AONFD#

Send on UDP data port for immediate execution or use with the cmdRxx command and store into memory locations.

e.g. cmdR20#codeName :R 11 $>28001A01 AONFD#

would store the string in location &H20 or 32 with the name 'codeName'

Max data length is 255 bytes.

An 'ACK' is returned by the module.

Macro Commands

Macros are a list of references to a memory location each preceded by a user definable pause.

Maximum number of steps to a macro is 30. (i.e. 30 delays with 30 locations)

Macros are stored in memory locations just like IR or RS232 codes

The first macro location is 151 and the last is 170 – 20 in all.

Attempts to load IR code or RS232 code to these locations will fail and attempts to load macros to other locations will also fail.

Macro format:

MD<no of steps><delay1><location1><delay2><location2>
All delays and locations and number of steps are 2 digit HEX and all are sent as ASCII
so for a 3 code macro targeting locations 3, 4, 5 with 2 second (8 * 1/4 second) delays,
the FIRST delay MUST be 00 (zero) for the macro to respond immediately.

A macro can be sent directly to the module on UDP IR data port default is 65432
for example - MD03000308040805

Same code stored to a macro location with name:-
cmdR98#Macro3:MD03000308040805

Macros can reference either IR codes or if the module supports RS232 then RS232 codes.
IR codes can contain repeats and routing, both will execute correctly with the pause starting after the final repeat

The pause is in steps of ¼ second and is a byte indicating the number of ¼ seconds to wait before issuing the command. Maximum pause is 255 * ¼ or 63.75 seconds minimum pause is 0.

The mandatory preface for the macro string is "MD" followed by the number of commands in the macro as 2 ASCII digits representing the hex value. i.e 0A = 10 or FF = 255

Macro Commands

(via RS232 CXT35 hardware and using a KiraRS232 or Matrix)

The RS232 compatible modules will now accept the X10 command strings sent as ascii to the module on the IR port. The string can be up to about 160 chars in length and may contain multiple strings. It is also possible to store these strings in the memory locations.

Example send using UDP on port 65432

$>28001B01 BONFF#$>28001B01 BDIM3C#$>28001B01 BDIM3C#$>28001B01 BDIM3C#

This is the string to turn on B1 and dim by 3 steps.

NOTE: On the KiraCC any ascii command can be used with the RS232 and stored into memory locations see KiraCC RS232 section above.

Any comments about this document or the KIRA product range please email sales@keene.co.uk.

Please check www.keene.co.uk regularly for updates.