iTach Flex TCP API Specification

Version 1.6

TABLE OF CONTENTS

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

1. ITACH FLEX - FLEXIBLE I/O DESIGN CONCEPT

The iTach Flex's small-footprint, multi-functional design offers a variety of capabilities. It uses standard micro-USB or PoE power, and offers Ethernet or WiFi connection capabilities, while its flexible I/O functionality is achieved through the Flex Link I/O Port. By connecting one of the available Flex Link cables, the base iTach Flex can easily be configured for a variety of functions, including infrared (IR) control, serial communications (RS232 or RS485), relays (contact closure), and sensors.

The iTach Flex Link Port is a 4-conductor female 3.5 mm I/O connector. Various Flex Link cables can connected to this port to provide many types of I/O connectivity. Configuration and settings for the Flex and Flex Link cable are accomplished through the unit's web interface, or API requests. (See Section 5 for additional details on TCP API requests.)

2. NETWORK CONFIGURATION

The iTach Flex is available with Ethernet (RJ45 100/10 Mbit/s), Power over Ethernet (PoE), and WiFi (802.11g) network connections.

The iTach Flex can be configured using embedded configuration webpages or by direct API requests. Factory default configuration settings provide specific predefined Ethernet and WiFi network configurations. Default settings can be restored at any time by depressing the iTach Flex reset button for more than 12 seconds. Factory default settings enable initial or recovery access to the device so it can be configured with the end-user's desired operating network settings.

The initial steps of network setup are different between Ethernet and WiFi devices. The WiFi device is unique in that it first requires connecting to the user's target WiFi network. This can be accomplished by two methods, either using the iTach Flex's adhoc WiFi network or using the WiFi Protected Setup (WPS) mechanism.

With the factory default settings, the iTach Flex WiFi operates in Adhoc mode where it broadcasts its own adhoc WiFi network. WiFi network provisioning is accomplished by connecting with an Adhoc-capable client (e.g., iOS, OS X, Linux, Windows). Once connected, the client device can connect to the iTach Flex WiFi at the fixed IP address of 192.168.1.70. Accessing this IP address with a browser displays the iTach Flex configuration webpage, which provides a means for provisioning the iTach Flex by selecting the desired target WiFi network, WiFi security mode, and IP address configuration (DHCP or static). When these settings are correctly selected and saved, the iTach Flex WiFi will reboot and connect to the configured WiFi network. By default it will obtain an IP address from the router via DHCP (if no DHCP server is present the device must be manually configured with a static IP address).

Alternatively, the iTach Flex WiFi model also supports WiFi Protected Setup (WPS) which is initiated by depressing the reset-button (located on the side of the iTach Flex) and releasing after 7 seconds (the power LED blinks faster to indicate WPS mode). Or, if access to the iTach Flex WiFi web configuration pages is already established, WPS mode can be activated in the network settings. In either case, once the WPS handshake is completed between the iTach Flex and the target WiFi router or access-point, the iTach Flex WiFi will connect to the target WiFi network.

The iTach Flex Ethernet defaults to DHCP IP address mode, and simply requires physical connection of an Ethernet network cable to the network. It can then be accessed for customized network configuration as necessary (For example, static IP).

To determine the IP address of an iTach Flex already connected to the network, the recommended method is to use the iHelp utility available for download at http://www.globalcache.com/downloads. iHelp has a number of useful functions, one of which is

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

to discover and display all Global Caché devices connected to the network. This is accomplished through a mechanism where the iTach Flex sends periodic (at 10 second intervals) beacon messages on the network that contain identifying information such as IP address, firmware revision, and MAC ID. The iHelp utility listens for these beacon messages, and then displays the information to the user in a device list. iHelp also provides additional functions such as firmware updates. Please refer to the iHelp documentation for further details.

3. NETWORK DISCOVERY

The iTach Flex provides two methods of network discovery – a periodic UDP beacon, and the multicast DNS (mDNS) mechanism.

The iTach Flex periodic UDP beacon message allows discovery of iTach Flex units on the network. The beacon is sent as a UDP packet to the multicast IP address 239.255.250.250, multicast group destination MAC 01:00:5E:7F:FA:FA, and destination port 9131. The beacon is sent shortly after successful network connection, and then at regular intervals of 10 seconds. Any network device which subscribes to the multicast group will receive the periodic beacon message.

The beacon message has the following format (field values are for example only):

```
AMXB
<-UUID=GlobalCache_000C1E024239>
<-SDKClass=Utility>
<-Make=GlobalCache>
<-Model=iTachFlexWiFi>
<-Revision=710-2000-15>
<-Pkg_Level=>
<-Config-URL=http://192.168.1.70>
<-PCB_PN=025-0033-10>(1)
<-Status=Ready>
```

The `Model` field can be: `iTachFlexEthernet|iTachFlexEthernetPOE|iTachFlexWiFi`

The `UUID` field value reflects the iTach Flex unique MAC address.

The iTach Flex also provides local name resolution using the multicast DNS mechanism, which also uses the UDP protocol. (See RFC 6762 for additional details, https://tools.ietf.org/html/rfc6762.) This mechanism makes the iTach Flex discoverable using a network name of the format <networkName>, where <networkName> is the value assigned in the Name field of the iTach Flex network settings.

(1) Starting with firmware version -14, PCB_PN version -09 or newer is required.

4. API CONNECTION

All iTach Flex TCP API requests and responses are communicated through TCP socket connection on port 4998. Up to 8 simultaneous TCP API client connections are supported. All TCP API requests are sent as a single line ASCII text string terminated with a carriage return. The requests follow a consistent format, which requires a command, and typically one or more associated

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

parameters specifying module and connector/port. Often additional comma delimited parameters for command-specific options are required. See the Section 5 API Requests for detailed explanation.

Each of the available Flex Link cables typically has its own associated TCP API commands. See specific details in the following sections for each Flex Link cable type and I/O functionality.

Note that in the unique case of the Flex Link Serial cable (RS232 or RS485) serial data is sent and received on TCP port 4999. In this case, data is communicated directly between the network TCP connection and Flex Link Serial cable connector in a TCP-to-serial bridge mode, and is not interpreted or altered by the iTach Flex in any way. See Section 5.3 for additional details.

5. API REQUESTS

The structure of all TCP API requests for iTach Flex and Flex Link cables are described in following section and subsections.

In general, all TCP API requests follow the following format:

`<command>,<module>:<port>,<parameter1>,<parameter2>,…,<parameterN>↵`

In the above line, the following details are important to note:

- Each parameter is represented by a name enclosed in brackets.
- Each parameter is separated by commas.
- Various commands require different number of parameters.
- The request must be a single line ending with a carriage-return, represented here by "↵".
- `<command>` is always a fixed text command string.
- `<module>:<port>` is sometimes a fixed value.
- `<parameterX>` always represents a user specified text value and must be replaced with a value as specified for the specific command. When multiple choices are available for the parameter they will be shown delimited by a vertical-bar separator "|" character.

Note: Commands and parameters are case sensitive.

Example: The get_NET command is used to query network settings:

`get_NET,<module>:<port>↵`

where:
`<module>:<port> = 0:1` (the iTach Flex host network module address is 0:1)

So, the literal TCP API request sent to the iTach Flex is:

`get_NET,0:1↵`

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

Note: get_NET command is fully documented in Section 5.1 below.

Errors

An ERR response is returned when a TCP API request is invalid.

Example invalid request:

```
setstate,1:1,↵          (the <state> parameter is missing - see Section 5.4)
```

Response:

```
ERR [error code]↵
```

Note: Error Code definitions are provided in the Error Codes table in Section 6.

5.1 GENERAL COMMANDS

getdevices

Query the current iTach Flex modules. The response is multi-line, and lists each module with its address and type, including the Host (0,0), and configured Flex Link cable (1,1). A complete response ends with an `endlistdevices` line.

Request:

```
getdevices↵          (query for modules and capabilities)
```

Response:

```
device,0,0 <Host type>↵
device,1,1 <Flex Link type>↵
endlistdevices↵
```

where:

```
<Host type>         = WIFI|ETHERNET
<Flex Link type>    = IR|IR_BLASTER|IRTRIPORT|IRTRIPORT_BLASTER
                      |SERIAL|RELAYSENSOR|SENSOR|SENSOR_NOTIFY
```

Following are several example responses:

```
device,0,0 WIFI↵          (WiFi device configured for Global IR Emitter cable)
device,1,1 IR↵
```


160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

```
endlistdevices↵


device,0,0 ETHERNET↵        (Ethernet device configured for Flex Link Serial cable)
device,1,1 SERIAL↵
endlistdevices↵
```

getversion

Query the iTach Flex firmware version.

Request:

```
getversion↵
```

Response:

```
<textversionstring>↵
```

where:

```
<textversionstring>          = firmware version part number
```

Example response from iTach Flex WiFi:

```
710-2000-15
```

get_NET

Query the current network settings. The response is a single comma delimited line.

Request:

```
get_NET,0:1↵
```

Response:

```
NET,0:1,<configlock>,<ipsetting>,<ipaddress>,<subnet>,<gateway>↵
```

where:

```
<configlock>        = LOCKED|UNLOCKED
<ipsetting>         = DHCP|STATIC
<ipaddress>         = IP address
<subnet>            = subnet mask
<gateway>           = network gateway
```


160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

5.2 IR (INFRARED)

The iTach Flex, when used with any of the available Flex Link IR cables, can output standard IR protocol signaling and control a wide range of IR controlled devices. TCP API commands for IR functionality are detailed in the following subsections.

5.2.1 IR COMMANDS

set_IR

This command allows configuration of the Flex Link Port to match the selected Flex Link IR cable. The available modes include Single IR, Single IR Blaster, Tri-port IR, and Tri-port with Blaster (blaster connected on port 3).

Request:

```
set_IR,<module>:<port>,<mode>↵

where:
<module>     = 1
<port>       = 1|…|3
<mode>       = IR|IR_BLASTER|IRTRIPORT|IRTRIPORT_BLASTER|SENSOR (1)|SENSOR_NOTIFY (1)|SERIAL
```

Response:

```
IR,<module>:<port>,<mode>↵
```

Example request and response:

```
set_IR,1:1,IR↵
```
Configures the Flex Link Port for a Global IR Emitter cable.

```
IR,1:1,IR↵
```

```
set_IR,1:1,IRTRIPORT↵
```
Configures the Flex Link Port for a Flex Link 3 Emitter cable or Flex Link Tri-port cable.

```
IR,1:1,IRTRIPORT↵
```

(1) Specifying a <mode> option of SENSOR or SENSOR_NOTIFY selects an iTach sensor compatibility mode. For detailed explanation, see Section 5.4.
(2) Setting serial cable type through this command is maintained for old implementations only. For new implementations we suggest using the set_SERIAL command in section 5.3.2.

get_IR

Retrieve the current mode setting for the Flex Link Port.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

Request:

```
get_IR,<module:port>↵
```

where:
<module> = 1
<port> = 1|…|3

Response:

```
IR,<module:port>,<mode>↵
```

where:
<mode> = IR|IR_BLASTER|IRTRIPORT|IRTRIPORT_BLASTER|SENSOR|SENSOR_NOTIFY

5.2.2 IR STRUCTURE

An IR transmission is created by sending an IR timing pattern to the iTach Flex. This pattern is a series of <on> and <off> states modulated with a carrier frequency (ƒ) which is present only during the <on> state. A carrier frequency is typically between 35 to 45 KHz, with some equipment manufacturers using as high as 500 KHz. The length of time for an <on> or <off> state is calculated in units of the carrier frequency period. For example, an <off> value of 24 modulated at 40 KHz produces an <off> state of 600µS, as calculated below.

A period is 1 /ƒ or 1/40000 or .000025 seconds or 25µS,
and a value of 24 periods is 600µS

Figure 5.2.1 illustrates an IR timing pattern that would be created for the value shown below. IR timing patterns typically have a long final <off> value (or rest state) to ensure the next IR command is interpreted as a separate IR transmission.

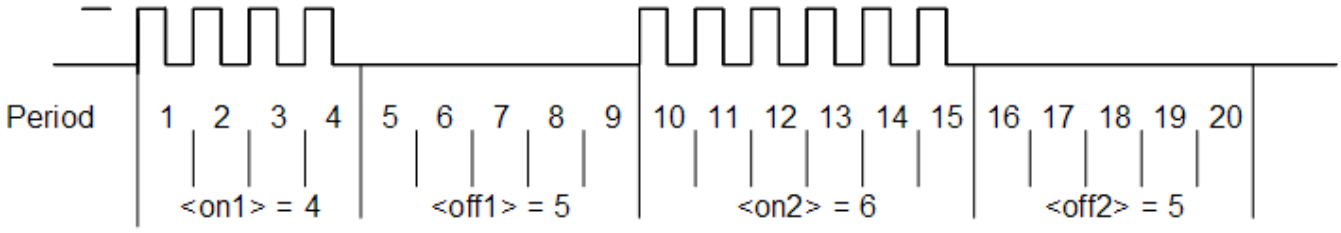

Figure 5.2.1

5.2.3 SENDING IR

Control of IR devices is accomplished through use of the `sendir` command. IR commands in some cases may take several hundred milliseconds to complete, so the iTach Flex provides a `completeir` acknowledgment to indicate when it is ready to accept the next IR command.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

sendir

Request:

```
sendir,<module:port>,<ID>,<freq>,<repeat>,<offset>,<on1>,<off1>,<on2>,<off2>,…,<onN>,<offN>↵
```

where:

<module:port>= [1] address of the desired IR channel (see below)
<ID> = 0|1|2|…|65535 [2] (for the completeir response, see below)
<freq> = 15000|…|500000 (in hertz)
<repeat> = 1|2|…|20 [3] (the IR command is sent <repeat> times)
<offset> = 1|3|5|…|383 [4] (used if <repeat> is greater than 1, see below)
<on1> = 4|5|…|50000 [5] (number of pulses)
<off1> = 4|5|…|50000 [5] (absence of pulse periods of the carrier frequency)
N = the count of <on>,<off> pairs, which must be less than 260 (total of 520 numbers)

[1] The Flex Link 3 Emitter, Flex Link 2E1B, and Flex Link Tri-port cables utilize the single 3.5 mm Flex Link Port to provide three (3) discrete channels of IR output. In these cases, sendir requests to any of the 3 IR channels must use the port component of the <module:port> parameter to address each of the 3 IR channels. Thus, for iTach Flex, each of the 3 IR channels is addressed as follows (not all sendir parameters are shown):

```
sendir,1:1,…
sendir,1:2,…
sendir,1:3,…
```

[2] <ID> is a number provided by the sendir request, which is later included in the completeir response to indicate successful completion of the requested IR transmission.

[3] <repeat> is the number of times an IR transmission is sent, if not aborted via a stopir, or subsequent IR command (see section 5.4). Values above 20 are accepted, but the maximum number of transmitted repeats is capped at 20. In all cases, the preamble is only sent once (see <offset> below).

[4] An <offset> applies when the <repeat> is greater than 1. For IR commands that have preambles, an <offset> is employed to avoid repeating the preamble during repeated IR timing patterns. The <offset> value indicates the location within the timing pattern to start repeating the IR command as indicated below. The <offset> will always be an odd value since a timing pattern begins with an <on> state and must end with an <off> state.

<offset> odd value	Repeat start locations	
1	<on1>	no preamble
3	<on2>	
….	….	
n-1	<on((n/2)-1)>	

[5] Since IR transmissions end in an <off> condition, there must be an equal number of <on> and <off> states. Also, every <on> and <off> state must meet an 80µS minimum time requirement for the IR codes to be sent properly.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

Example: With a carrier frequency of 60 KHz, the minimum value for <on> and <off> states is calculated below.

$$<off>_{min} = <on>_{min} \geq 80\mu S * f = 80\mu S * 60\ KHz = 4.8$$

For proper replication of an IR code at 60 KHz, all <on> and <off> values in the timing pattern must be 5 or higher.

All of the conditions above must be met for a valid sendir request. When a variable is missing or outside the accepted range, an error will be returned. For example, the example sendir commands below will result in an error response.

Request:
sendir,10:3,3456,23400,1,1,24,48,24,960↵ (invalid module number for iTach Flex)
Response:
ERR 003↵

Request:
sendir,1:1,23333,40000,2,3,24,48,24,48,960↵ (not an equal number of <on> and <off> parameters)
Response:
ERR IR006↵

Request:
sendir,1:1,0,40000,2,2,24,48,24,960↵ (<offset> is an even number)
Response:
ERR IR004↵

GLOBAL CACHÉ COMPRESSED IR FORMAT

The compressed IR format interprets the first 15 unique <on>,<off> pairs and assigns a single uppercase letter (A,B,C, etc.) to each of those unique pairs. The first time a unique pair occurs within an IR command it is assigned a single ASCII character. Any subsequent occurrences of that same pair are then represented with that same ASCII character instead of the <on>,<off> representation. This allows the sendir request string to be significantly shorter, which can be advantageous in the case of a long IR code.

Example:

sendir,1:1,2445,40000,1,1,**4,5**,4,5,**8,9**,4,5,8,9,8,9↵

The unique pairs are underlined in the example request above. By assigning A to 4,5 and B to 8,9, the request can be rewritten as follows:

sendir,1:1,2445,40000,1,1,4,5A8,9ABB↵

Both commands are syntactically correct and are accepted by the iTach Flex. Both will transmit an identical IR code.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

completeir

After the iTach Flex successfully processes and transmits an IR command, it responds by sending the `completeir` response to the requester. The `completeir` response can be associated with the originating `sendir` command by way of the <ID> parameter. When utilized, the <ID> can provide a unique identifier to determine which IR transmissions have completed.

Sent from iTach Flex in response to successful `sendir`:

```
completeir,<module>:<port>,<ID>↵
```

where:

<module>	= 1	
<port>	= 1|…|3	
<ID>	= 0|1|2|…|65535	(value is generated by the sender of the `sendir` request)

Examples:

Request:

```
sendir,1:1,2445,40000,1,1,4,5,6,5↵
```

Response:

```
completeir,1:1,2445↵
```

The following examples demonstrates two commands to send the same simple IR timing pattern of 24,12,24,960, repeated four times, and with a preamble of 34,48:

```
sendir,1:1,4444,34500,1,1,34,48,24,12,24,960,24,12,24,960,24,12,24,960,24,12,24,960↵
sendir,1:1,34,34500,4,3,34,48,24,12,24,960↵
```

Responses to the above requests:

```
completeir,1:1,4444↵
completeir,1:1,34↵
```

The same IR command is repeated four (4) times by either `sendir` request, but since the <ID> is different, a different `completeir` response is returned. The <ID> also allows associating each request to its response. Note that the second IR request is the recommended format, since it is shorter and also allows the command to be halted in between repeats, if necessary.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

stopir

This request will stop an active IR transmission. Any remaining <repeat> cycles will be aborted. A stopir command sent to a connector not configured as an IR output will return an error message. An IR transmission halted with the stopir command will return a stopir response. Furthermore, if an IR command is halted before its completion by another connection, the originating IR connection and the connection sending stopir will both receive a stopir response. If stopped, the originating connection will not receive a completeir response.

Request:

```
stopir,<module>:<port>↵
```

Response:

```
stopir,<module>:<port>↵
```

where:
<module> = 1
<port> = 1|…|3
A stopir command always returns a stopir response, regardless of whether the port is idle, or an IR active transmission is halted. A stopir response means only that the stopir command was successfully sent to the iTach Flex and any transmission has been halted.

busyIR

A busyIR response occurs when an IR command is received for a port which is actively transmitting an IR code. This might occur, for example, if multiple IP connections are present (such as from multiple network users). In this case, the sendir command that receives the busyIR response will have its IR code transmitted.

A busyIR response does not occur if the two simultaneous IR requests are sent to different IR ports.

Response to an attempt to interrupt IR transmission by another IP connection or socket:

```
busyIR,1:1,<ID>↵
```

where:
<ID> = |0|1|2|…|65535| (ID is specified in sendir command)

Note: The busyIR response is sent to the originator of the failed IR command. A stopir command will only return a stopir response. A stopir command will never return a busyIR response.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

5.2.4 SMOOTH CONTINUOUS IR COMMANDS

A general discussion is necessary to better understand how smooth continuous IR commands are executed by the iTach Flex. This feature is utilized for smooth volume control or repeating an operation without the appearance of choppy actions. The approach of sending an IR command with very large repeat counts and stopping it upon request will work, but can lead to undesirable incidents. Consider an IR command with a large repeat-count used to increase volume in a smooth fashion. The IR command continues repeating while volume continuously increases. But if the controlling client/application unexpectedly disconnects, the volume could continually increase (possibly damaging equipment) until the client reconnects and issues a `stopir` command.

The iTach Flex solution is to limit the repeat count. To create a smooth IR operation, the iTach Flex resets the IR repeat count each time the identical IR command (from the same IP connection) is resent. This method will not interrupt and restart the IR command, but reset the IR repeat count back to the original value.

Example: If the IR repeat count is set to 5, and the IR command has transmitted 3 times, receipt of the same command causes the repeat count to be reset back to 5. This process can continue indefinitely while a volume button is held down to create a smooth operation. However, at no time can the command repeat more than 5 times after the button is released or an IP connection is inadvertently lost, preventing a potentially serious issue.

By selecting an appropriate <repeat> value, the need for a `stopir` command is eliminated. In this example the volume continues to increase smoothly by retransmitting repeated IR commands due to the volume button being pressed. As long as the next repeated IR command is received before the previous command finishes, smooth operation is realized. By choosing a low repeat value, the volume increase will stop when the volume button is released. Also, proper IR operations happen even with unintended network delays due to traffic or WiFi connectivity. In this unlikely event, only small hesitations will be experienced during IR operation.

In the event that the identical command is not received before the original command is finished, the command will be registered as a brand new command, and is sent as such. The command in question will operate functionally the same, but delays between commands may be evident when used in this way. Increasing the <repeat> value will likely eliminate these discrepancies.

5.2.5 IR LEARNING

Each iTach Flex unit provides an onboard IR sensor, located inside the small recessed hole at the top center of the iTach Flex enclosure. IR Learner Mode is initiated as follows.

get_IRL

Request:

```
get_IRL↵
```

Response:

```
IR Learner Enabled↵
```


160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

This request initiates the iTach Flex IR learn mode. When a valid IR code is received by the external IR sensor, the iTach Flex responds by returning the full Global Caché format `sendir` command (uncompressed, terminated by a carriage return) to the connection which made the `get_IRL` request.

The iTach Flex remains in the IR learn mode until a `stop_IRL` or `sendir` request is received. Thus, multiple IR codes can be learned without additional `get_IRL` requests.

stop_IRL

Aborts a pending `get_IRL` request.

Request:

```
stop_IRL↵
```

Response:

```
IR Learner Disabled↵
```

5.3 SERIAL

The Flex Link RS232 Serial cable and Flex Link RS485 Serial cable are iTach Flex peripheral I/O cables which enable bi-directional serial communications over RS232 or RS485 interface standards, respectively. Serial communication parameters are configured using the iTach Flex embedded webpages or by direct API commands. Unless otherwise noted, serial commands are only available if the unit is configured for Flex Link Serial cable mode.

Note: When the iTach Flex is set to Flex Link Serial cable mode through the configuration webpages or API command, it automatically detects RS232 or RS485 cable type (no configuration of serial cable type is required). All TCP API commands are the same for either serial cable type, unless specifically noted otherwise.

To send and receive serial data, a TCP socket connection must be established on port 4999. Data is communicated directly between the TCP connection and Flex Link Serial cable connector in a TCP-to-serial bridge mode. The data is passed directly and is not interpreted or altered in any way by the iTach Flex.

If iTach Flex serial communication parameters are not configured correctly to match the connected device, buffer overflows (indicating data loss) or parity errors may occur.

5.3.1 MULTIPLE SERIAL CONNECTIONS

When used with a Flex Link Serial cable, the iTach Flex supports up to eight (8) simultaneous, bidirectional TCP-to-serial connections on port 4999.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

SENDING DATA TO A SERIAL DEVICE

In order to support multiple TCP clients simultaneously sending data to a single serial-connected device, the iTach Flex handles received TCP data on a packet basis for efficient transmission to a serial-connected device. When a data packet is received by the iTach Flex it is transmitted completely to the serial-connected device before any subsequent packets (received from the same or different socket) are sent. This is important when considering the case where TCP client A sends a single command in two (2) separate packets while a different TCP client B simultaneously sends a packet. In this case, the packet from TCP client B may be serviced in between the 2 separate packets received from TCP client A, possibly interrupting the data or command from TCP client A. It is very important for TCP clients sending data to the Flex to send a complete serial data sequence or command(s) within a single packet. This ensures complete data sequences or commands are received properly by the serially connected device.

RECEIVING DATA FROM A SERIAL DEVICE

Serial data received by the iTach Flex from a serial-connected device is also handled as packets. But unlike TCP-to-serial data flow, wherein packets are based on TCP protocol, serially received data is packetized according to timing and size. The initial character of received serial data initiates a receive packet. The completion of the packet occurs when either the iTach Flex receive buffer reaches a certain capacity or no characters are received for a calculated time delay (based on current baud rate). The completed serial data packet is then immediately transmitted to all active/open TCP socket connections. This allows all connected TCP clients to synchronously receive all data sent from the serial device.

5.3.2 SERIAL COMMANDS

set_SERIAL

Set the iTach Flex to Flex Link Serial cable mode, and configure the serial communication settings.

Requests:

```
set_SERIAL,<module>:<port>↵

set_SERIAL,<module>:<port>,<baudrate>,<flowcontrol/duplex>,<parity>,[stopbits]↵
```

Response:

```
SERIAL,1:1,<baudrate>,<flowcontrol/duplex>,<parity>,<stopbits>↵
```

where:

```
<module>:<port>     = 1:1
<baudrate>          = 300|…|115200
<flowcontrol/duplex> = FLOW_HARDWARE|FLOW_NONE(1)  (Flex Link Serial cable – RS232)
                       DUPLEX_HALF|DUPLEX_FULL(1)  (Flex Link Serial cable – RS485)

<parity>            = PARITY_NO|PARITY_ODD|PARITY_EVEN
[stopbits]          = STOPBITS_1|STOPBITS_2  (optional parameter)
```

Example request and response:

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

```
set_SERIAL,1:1,38400,FLOW_NONE,PARITY_NO↵

SERIAL,1:1,115200,FLOW_NONE,PARITY_NO,STOPBITS_1↵
```

get_SERIAL

Query current serial communications settings.

Request:

```
get_SERIAL,<module>:<port>↵
```

Response:

```
SERIAL,1:1,<baudrate>,<flowcontrol/duplex>,<parity>,<stopbits>↵
```

where:

```
<module>:<port>     = 1:1
<baudrate>          = 300|…|115200
<flowcontrol/duplex> = FLOW_HARDWARE|FLOW_NONE (Flex Link Serial cable – RS232)
                       DUPLEX_HALF|DUPLEX_FULL(1) (Flex Link Serial cable – RS485)

<parity>            = PARITY_NO|PARITY_ODD|PARITY_EVEN
<stopbits>          = STOPBITS_1|STOPBITS_2
```

Example request and response:

```
get_SERIAL,1:1

SERIAL,1:1,115200,FLOW_NONE,PARITY_NO,STOPBITS_1↵
```

(1) The <flowcontrol/duplex> parameter is shared between the two Flex Link Serial cable types (RS232 and RS485). It is used for two different settings, each specific to a particular Flex Link Serial cable type. Either setting can be changed at any time, but only the setting applicable to the connected Flex Link Serial cable will be displayed in the response to a set_SERIAL or get_SERIAL command. If no Flex Link Serial cable is connected, the <flowcontrol> setting will be displayed.

5.4 RELAYS AND SENSORS

The Flex Link Relay & Sensor cable is an iTach Flex peripheral I/O cable which provides dry contact closure relay outputs capable of switching a variety of devices, as well as bi-stable Sensor inputs for sensing of (on/off) voltage and current conditions. See the Flex Link Relay & Sensor Cable data sheet for detailed hardware specification.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

This section describes the TCP API commands used for configuration and control of the relay outputs, and the commands for configuring and reading the Sensor inputs in either a polled or automatic-notification mode. Unless specified otherwise, relay and sensor commands are only available if the unit is configured for Flex Link Relay & Sensor cable mode.

iTach Sensor Compatibility Mode

To allow backwards-compatibility with iTach IR sensors, the iTach Flex with Flex Link Relay & Sensor cable can be configured to operate in an iTach sensor compatibility mode. This mode is selected using the `set_IR` command (see Section 5.2.1). When in this mode, the iTach Flex with Flex Link Relay & Sensor cable operate exactly like an iTach IR device having all three IR ports configured in sensor or sensor notify mode. Any sensor API commands or operations use `<module>` = 1, same as the iTach (including sensor notify beacons - for more information see Section 5.4.3).

Note: Because iTach sensor compatibility mode is intended to operate exactly like an iTach IR in sensor mode, the relay functionality of the Flex Link Relay & Sensor cable is not available in this mode. To utilize the full Flex Link Relay & Sensor cable functionality (with all available relays and sensors) refer to the related commands in Section 5.4.

5.4.1 Relay and Sensor Commands

set_RELAYSENSOR

Set the iTach Flex to Flex Link Relay & Sensor cable mode.

Requests:

`set_RELAYSENSOR,<module>:<port>↵`	Set the Flex to Flex Link Relay & Sensor cable mode

where:

`<module>` = 1
`<port>` = 1|…|4

Response:

`RELAYSENSOR,<module:port>↵`

get_RELAYSENSOR

Query the iTach Flex Flex Link Relay & Sensor cable type.

Requests:

`get_RELAYSENSOR,<module>:<port>↵`	Set the Flex to Flex Link Relay & Sensor cable mode

where:

`<module>` = 1
`<port>` = 1|…|4

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

Response:

```
RELAYSENSOR,<module>:<port>,<type>↵
```

where:
<type> = 4R4S (4 relays and 4 Sensors)

5.4.2 RELAY COMMANDS

get_RELAY

Query the configured type of a specified logical relay port.

Request:

```
get_RELAY,<module>:<port>↵
```

where:
<module> = 1
<port> = 1|…|4

Response:

```
RELAY,<module>:<port>,<type>↵
```

where:
<type> = SPST|SPDT|DPDT
(**Note:** A detailed explanation of relay types and configuration can be found in the Flex Link Relay & Sensor cable data sheet.)

Example request and response:

```
set_RELAY,1:4,SPST
```

```
RELAY,1:4,SPST↵
```

set_RELAY

Set the type of a specified logical relay port.
Note: When setting relay type, all affected relays must disabled. See footnote [1] below for details.

Request:

```
set_RELAY,<module>:<port>,<type>↵
```
Configure a logical relay port to a specified type

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

where:

<module> = 1

<port> = 1|…|4

<type>[1] Depends on the specified <port>. Valid settings are as follows: [1]

<port> = 1 <type> = Disabled|SPST|SPDT|DPDT

<port> = 2 <type> = Disabled|SPDT

<port> = 3 <type> = Disabled|SPST|SPDT

<port> = 4 <type> = Disabled|SPDT

(**Note:** Additional information regarding relay types and configurations can be found in the Tech Guide: Flex Link Relay & Sensor Cable.)

Response:

RELAY,<module>:<port>,<type>↵

[1] Each of the available relay types (SPST, SPDT, and DPDT) maps to one or more physical ports, as shown in Figure 5.4.1 below. (The example configurations shown do not include all possible combinations).

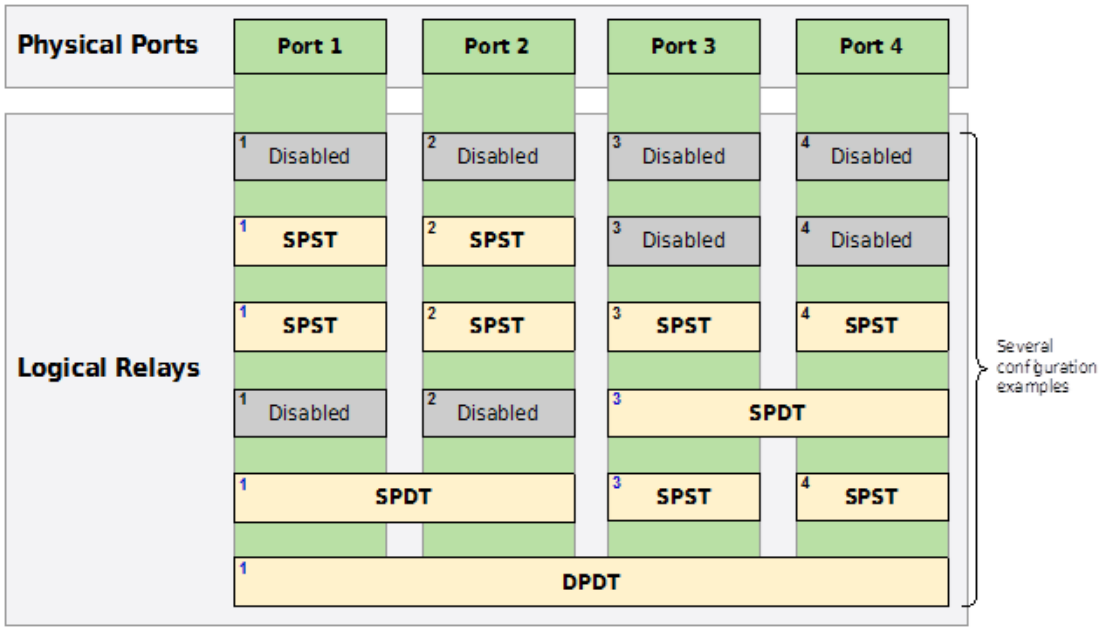

Figure 5.4.1

When setting a particular relay type:

1. All mapped relay ports must be set to the Disabled setting.
2. Hardware jumpers must be configured for the selected relay type (see Tech Guide: Flex Link Relay & Sensor Cable).
3. External wiring to controlled devices should be connected or adjusted accordingly.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

The requirement to set all mapped relay ports to Disabled ensures those ports are in a known off state. This safely allows any required changes to hardware jumpers and external device connections, thereby ensuring known conditions when the new relay type setting is applied.

Following is an example request/response sequence to change logical relay port 3 type from SPST to DPDT:

`get_RELAY,1:3↵` `RELAY,1:3,SPST↵`	Query the current configuration for logical relay port 3.
`set_RELAY,1:3,Disabled↵` `RELAY,1:3,Disabled↵`	Set logical relay port 3 to Disabled and make necessary changes to hardware jumpers and external device connections.
`set_RELAY,1:3,DPDT↵` `RELAY,1:3,DPDT↵`	Finally, set logical relay port 3 to DPDT type.

getstate

Query the current state of a logical relay port.

Request:

`getstate,<module>:<port>↵`

where:
`<module>` = 1 (1)
`<port>` = 1|…|4

Response:

`state,<module>:<port>,<state>↵`

where:
`<state>` Depends on the configured `<type>` for the specified `<port>`, as follows:

`<type>` = SPST
`0` = off (open)
`1` = on (closed)

`<type>` = SPDT|DPDT
`0` = off (center position)
`1` = on1 (1st "throw")
`2` = on2 (2nd "throw")

(**Note:** A detailed explanation of relay types and configuration can be found in the Tech Guide: Flex Link Relay & Sensor Cable.)

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

(1) A module value of 3 is also accepted for GC-100 backwards compatibility.

setstate

Set the state of a logical relay port.

Request:

```
setstate,<module>:<port>,<state>↵
```

where:

<module> = 1 (1)
<port> = 1|…|4
<state> Depends on the configured <type> for the specified <port>, as follows:
<type> = SPST
0 = off (open)
1 = on (closed)

<type> = SPDT|DPDT
0 = off (center position)
1 = on1 (1st "throw")
2 = on2 (2nd "throw")

Response:

```
state,<module>:<port>,<state>↵
```

Note: Relay states are not preserved if the Flex Link Relay & Sensor cable is unplugged or reinitialized or if the iTach Flex is reset and/or power cycled. Under any of those conditions, all relays will return to their default inactive (open) state.

(1) A module value of 3 is also accepted for GC-100 backwards compatibility.

5.4.3 SENSOR COMMANDS

The iTach Flex with Flex Link Relay & Sensor cable can query sensor input states in a Polled mode, as follows.

getstate

Query the state of a sensor input port.

Request:

```
getstate,<module>:<port>↵
```

where:

<module> = 2 (1) (2)

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

```
<port>        = 1|…|4
```

Response:

```
state,<module>:<port>,<state>↵
```

where:

```
<state>       = 0     contacts closed  (contact-closure mode) (3)
                      voltage below threshold  (voltage-sense mode) (3)
              = 1     contacts open  (contact-closure mode) (3)
                      Voltage above threshold  (voltage sense mode) (3)
```

(1) In Flex Link Relay & Sensor cable mode, sensors are addressed at `<module>` = **2** (since `<module>` = `1` is used for logical relay ports). Additionally, a module value of 4 or 5 may be used for GC-100 backwards compatibility.
(2) In iTach Sensor compatibility mode (see section 5.4), sensors are addressed at `<module>` = 1.
(3) For a detailed explanation of sensor modes, refer to Tech Guide: Flex Link Relay & Sensor Cable.

5.4.4 SENSOR NOTIFY

The iTach Flex with Flex Link Relay & Sensor cable supports a sensor notify feature, which provides automatic/asynchronous notification of changes in sensor input state. When enabled, a UDP multicast message can be broadcast when either of the following conditions occurs:

- A time interval elapses.
- A sensor input changes state.

Both the time-interval and UDP port-number are configurable for each of the sensor ports, as follows.

set_SENSORNOTIFY

Set the sensor notify configuration options for a specified sensor port.

Request:

```
set_SENSORNOTIFY,<module>:<port>,<notify_port>,<notify_interval>↵
```

where:

```
<module>          = 2 (1) (2)
<port>            = 1|…|4
<notify_port>     = 0|…|65535    ( '0' = all beacons disabled )
<notify_interval> = 0|…|65535    ( '0' = periodic beacon disabled )
```

Response:

```
SENSORNOTIFY,<module>:<port>,<notify_port>,<notify_interval>↵
```


160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

(1) In Flex Link Relay & Sensor cable mode, sensors are addressed at <module> = 2 (since <module> = 1 is used for logical relay ports). Additionally, a module value of 4 or 5 may also be used for GC-100 backwards compatibility.
(2) In iTach sensor compatibility mode (see section 5.4), sensors are addressed at <module> = 1.

The default values are <notify_port> = 0 (which disables both the state-change and periodic notification beacons) and <notify_interval> = 0 (which disables only the periodic notification beacon). These disabled-by-default settings avoid a potential flood of undesired network UDP traffic in default configuration and allow custom user configuration according to the specific network environment.

When selecting UDP port values, it is advisable to avoid conflicts with any ports already in use by the network environment. Please consider all connected network hardware, and refer to various available standards registries (such as the IANA Service Name and Transport Protocol Port Number Registry, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt) for a list of assigned vs. available port numbers. For example, according to the IANA registry, UDP ports 9132 - 9159 are unassigned and could be a good choice if not already used by other network devices.

6. ERROR CODES

The table below lists of error messages returned by the iTach Flex in response to various invalid TCP API requests. See section 5 for an example invalid request and resulting response format.

General Errors	Explanation
ERR 001	Invalid request. Command not found.
ERR 002	Bad request syntax used with a known command.
ERR 003	Invalid or missing module and/or connector address.
ERR 004	No carriage return found.
ERR 005	Command not supported by current Flex Link Port setting.
ERR 006	Settings are locked.
IR Errors	Explanation

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

ERR IR001	Invalid ID value.
ERR IR002	Invalid frequency.
ERR IR003	Invalid repeat.
ERR IR004	Invalid offset.
ERR IR005	Invalid IR pulse value.
ERR IR006	Odd amount of IR pulse values (must be even).
ERR IR007	Maximum pulse pair limit exceeded.
Serial Errors	Explanation
ERR SL001	Invalid baud rate.
ERR SL002	Invalid flow control or duplex setting.
ERR SL003	Invalid parity setting.
ERR SL004	Invalid stop bits setting.
Relay & Sensor Errors	Explanation
ERR RS001	Invalid logical relay type.
ERR RS002	Invalid logical relay state.
ERR RS003	Unsupported operation.
ERR RS004	Logical relay disabled or not available.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808

ERR RS008	Invalid sensor notify port value.
ERR RS009	Invalid sensor notify timer value.

160 East California Street, PO Box 1659
Jacksonville, Oregon 97530
Phone: 541-899-4800
Fax: 541-899-4808