CommandFusion Wiki

Documentation Resources

CFLink Protocol Introduction

This document is an overall view of the CFLink Protocol, used by CommandFusion hardware.
CFLink is an RS485 5-wire bus, used to interconnect various CommandFusion devices. For more details on the physical bus characteristics, please see the CFLink Bus documentation.

Basic Structure

Hex Data

Throughout the CFLink Protocol documentation, hex bytes will be displayed in square brackets, with uppercase hex characters inside.
For example, a carriage return is represented as `[0D]`.

Message Format

The basic structure of all messages on the CFLink bus is as follows:

```
// Basic message format
> [F2]<ID>[F3]<COMMAND>[F4]<DATA>[F5][F5]
```

All messages start with `[F2]` followed by a single ID byte representing the target (or source in the case of replies) device on the bus.
Then an `[F3]` byte, followed by 7 bytes representing the Command.
Then an `[F4]` byte, followed by optional Data associated with the command. The Data can be empty if the command does not require any associated data.
Finally, all messages are terminated with two `[F5]` bytes.

> **NOTE: THROUGHOUT THIS DOCUMENTATION, ALL CFLINK EXAMPLES WILL START WITH > (greater than) OR < (less than) - THIS IS NOT PART OF THE COMMAND, BUT JUST USED TO SIGNIFY THE START OF A NEW SENT OR RECEIVED PACKET WITHIN THE DOCUMENTATION.**

CFLink ID

Every device on the CFLink bus must have a unique ID. The valid ID range is `[02]` to `[EF]` (238 total per CFLink bus).
When sending a command, you must specify the ID of the device to send the command to.

Broadcast ID

In some circumstances, you may want to send a broadcast to all devices on the network. This can be done by using the Broadcast ID: `[FF]`.

> **NOTE: BROADCAST MESSAGES CANNOT BE GUARANTEED TO BE DELIVERED TO ALL DEVICES ON LARGE NETWORKS. BROADCASTING IS ONLY RECOMMENDED**

FOR NON-CRITICAL MESSAGES.

Commands and Replies

The **<COMMAND>** part of each message is always 7 characters, always upper case, and formatted as follows: `<TYPE><DEVICE><COMMAND_NAME>`

- **<TYPE>** = 1 char, types are documented below.
- **<DEVICE>** = 3 chars representing the model name of the device or type of port we are targeting (or the device/port type that the reply came from).
    - The **<DEVICE>** name `CFX` can be used to target any device. It is very useful for when you are sending a command that any device should respond to, `WHO` for example.
      It is also useful when you don't need to know the exact device type receiving the command, such as for on-board COM ports which share a common protocol across all devices.
      However, all *replies/notifications* will be sent with the correct 3 character device identifier for the device sending the data (`CFX` is never used in replies except from bootloader notifications).
- **<COMMAND_NAME>** = 3 chars representing the name of the actual command being performed.

Queries

Query messages begin with **<TYPE>** character `Q`. These messages are used to retrieve the configuration details or state of a specific device property.

A query will always get a reply, unless the target device could not be found (CFLink ID does not exist on the bus).
If the query was successful, the reply would contain the same **<COMMAND_NAME>** along with the **<DATA>** associated with the query.

If there was an error in the query, the reply would be an Error Reply with details of why the error occurred.

Configuration

Configuration messages begin with **<TYPE>** character `C`. These messages are used to manipulate the configuration settings of a device.

A configuration message will always get a reply, unless the target device could not be found (CFLink ID does not exist on the bus).

There are two situations that reflect what data will be replied:

1. The configuration property is set immediately
2. The configuration property requires a reboot of the device to take affect.

If the configuration change requires a reboot (ie. changing the DHCP mode for a LAN Bridge), the reply to the configuration message will contain the current configuration data.
The new configuration data will only be available after rebooting the device.
If the configuration change is instant (ie. changing the CFLink ID of a device), the reply will contain the new configuration data.

If the was an error in the configuration message, the reply would be an Error Reply with details of why the error occurred.

Transmission

Transmission messages begin with **<TYPE>** character `T`. These messages are used to tell the device to perform an action, such as set a relay state or send data out a serial port.

A transmission message will always get a reply, unless the target device could not be found (CFLink ID does not exist on the bus).
Most replies will simply echo the **<COMMAND_NAME>** along with any **<DATA>** from the transmission message.
But some replies will not include the **<DATA>** in order to reduce flooding of the network. These replies are from commands such as sending RS232 data or sending IR hex data, both of which can be quite long.

If the was an error in the transmission message, the reply would be an Error Reply with details of why the error occurred.

Replies

Reply messages begin with **<TYPE>** character `R`. Replies are always initiated by the device, in response to any of the other command types.
Generally the reply will contain the **<DEVICE>** name for the device sending the reply, along with echoing the **<COMMAND_NAME>**.
The only time a reply will not be sent is if the target CFLink ID does not exist on the CFLink bus.
More details on the reply formatting is given in the other message type documentation above.

Error Replies

If there is an error in any message, the device will reply with details via an error message in the following format:

```
// Error reply format example
< [F2]<ID>[F3]R<DEVICE>ERR[F4]<ERRCODE>:<SUMMARY>:<SENDERID>:<SENT_CMD>:<SENT_DATA>[F5][F5]
```

- **<ERRCODE>** = 3 digits unique to each error type (000 to 999).
- **<SUMMARY>** = A variable length summary in English of what the error was.
- **<SENDERID>** = 2 bytes representing the ID that the incorrect command was sent from (ID shown in plain text).
- **<SENT_CMD>** = 7 chars echoing the command that was sent which caused the error.
- **<SENT_DATA>** = If there was any data in the command that caused the error, it will be echoed in the error reply.

Data

Throughout the protocol, the <DATA> section of the message format is used to supply additional details to the commands and replies.

Port Definitions

When the protocol requires a port number to be defined, it is always written in P## format. The number must be two chars, 01-99 or ZZ.
`P01` = Port 1, `P10` = Port 10, etc.
`PZZ` = All Ports. This allows you to manipulate all ports of a single type at once. eg. Open all relays in a CF Mini.

Module Definitions

When the protocol requires a module number to be defined, it is always written in M# format. The number is always 1 char, 1-9 or Z.
`M1` = Module 1, `M4` = Module 4, etc.
`MZ` = All Modules. This allows you to manipulate all modules in a modular device at once. eg. Open all relays in a DIN-MOD4.

No Change Option

If you want to leave a parameter unchanged, and only change some of the options in a single protocol message, simply define the parameters that you want to change as normal, and for the ones you want to leave unchanged, use the uppercase `X` character in their place.
This `X` (no change) is only valid for parameters that use a fixed length, and must be repeated for each required length byte.
So if a parameter requires 2 bytes length, and you want to leave it unchanged, enter `XX` as the parameter value.
Throughout the documentation, any messages that accept `X` (no change) parameters will be listed in the commands 'Data' section.

Data Separators

There are 3 data separators used in the CFLink protocol:

1. Port separator | (single pipe)
2. Data separator : (colon)
3. Module separator , (comma)

Port Separator

When a command or reply targets multiple ports of a device, the port data is separated by a single pipe | character.
The port separator is also used to separate a Module Number from the port data.

```
// Port Separator example - Close relay port 1, open relay port 2 on a CF Mini on CFLink ID
> [F2][04][F3]TRLYSET[F4]P01:1|P02:0[F5][F5]
// Port Separator example, with module number - Close relay port 1, open relay port 2 on Modu
> [F2][04][F3]TRLYSET[F4]M2|P01:1|P02:0[F5][F5]
```

Data Separator

The data separator : (colon), is used to separate pieces of data for a single target port or device.

```
// Data Separator example - Device discovery reply for a LAN Bridge on ID [02]
< [F2][02][F3]RLANWHO[F4]LANBridge:192.168.0.100:00.04.A3.19.D5.70:1.0.0.0:1.0.0.0[F5][F5]
// Note: model, IP address, MAC address, boot loader version and firmware version data are a
```

Module Separator

When a command targets a modular device, it is possible to target multiple modules in one message. This is done by using the module separator , (comma)

```
// Module Separator example - Module 1, Close relay ports 1 and 2, Module 2, Open relay ports
// Within a MOD4 on CFLink ID [04]
> [F2][04][F3]TRLYSET[F4]M1|P01:1|P02:1,M2|P01:0|P02:0[F5][F5]
// Note the comma after the module 1 data, before the module 2 data.
```

---

hardware/cflink/cflink-introduction.txt · Last modified: 2014/02/20 11:57 by jarrod