



NetLinx Module Interface Specification

for

Clipsal C-Bus Lighting System

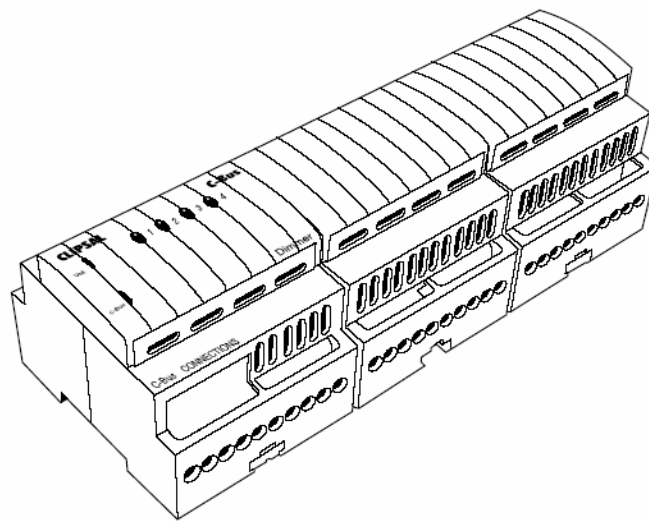


TABLE OF CONTENTS

- Control Life Pty Ltd License Agreement.....3
- Introduction5
- Overview5
- Implementation6
- Channels7
- Command Control8
- Command Feedback.....13
- Programming Notes15
- Adding Functions to Modules16
 - Commands to the device16

Revision History

Date	Initials	Comments
21-12-2007	KN	Initial release v1.0.0

Control Life Pty Ltd License Agreement

By using the software included with this End-User License Agreement ("EULA"), you ("Licensee") agree to be bound by, and Control Life Pty Ltd shall be entitled to enforce, this Control Life Pty Ltd License Agreement ("Agreement"). IF YOU DO NOT AGREE, DO NOT USE THE SOFTWARE, YOU MAY RETURN IT TO Control Life Pty Ltd FOR A FULL REFUND.

1. **LICENSE GRANT.** Control Life Pty Ltd grants to Licensee the non-exclusive right to use the Control Life Pty Ltd Software in the manner described in this License. This license does not grant Licensee the right to create derivative works of the Control Life Pty Ltd Software. The Control Life Pty Ltd Software consists of programming and development software, product documentation, sample applications, tools and utilities, and miscellaneous technical information. The Control Life Pty Ltd Software is subject to restriction on distribution described in this License Agreement. **YOU MAY NOT SUBLICENSE, RENT OR LEASE THE Control Life Pty Ltd SOFTWARE.** You may only use the Control Life Pty Ltd Software for its intended purpose. You may not reverse engineer, decompile, or disassemble the Control Life Pty Ltd Software.
2. **INTELLECTUAL PROPERTY.** The Control Life Pty Ltd Software is owned, by Control Life Pty Ltd and is protected by Australian copyright laws, patent laws, international treaty provisions, and/or state of NSW trade secret laws. Licensee may make copies of the Control Life Pty Ltd Software solely for backup or archival purposes. Licensee may not copy any written materials accompanying the Control Life Pty Ltd Software. The Software is licensed, not sold.
3. **TERMINATION.** CONTROL LIFE PTY LTD RESERVES THE RIGHT, IN ITS SOLE DISCRETION, TO TERMINATE THIS LICENSE FOR ANY REASON AND UPON WRITTEN NOTICE TO LICENSEE. In the event that Control Life Pty Ltd terminates this License, then Licensee shall return all copies of the Control Life Pty Ltd Software to Control Life Pty Ltd and certify in writing that all copies have been destroyed.
4. **PRE_RELEASE CODE.** Portions of the Control Life Pty Ltd Software may, from time to time, as identified in the Control Life Pty Ltd Software, include PRE-RELEASE CODE and such code may not be at the level of performance, compatibility, and functionality of final code. **THE PRE-RELEASE CODE may not operate correctly and may be substantially modified prior to final release or certain features may not be generally released.** Control Life Pty Ltd is not obligated to make or support any PRE_RELEASE CODE. **ALL PRE-RELEASE CODE IS PROVIDED "AS IS" WITH NO WARRANTIES.**
5. **LIMITED WARRANTY.** Control Life Pty Ltd warrants that the Control Life Pty Ltd Software will perform substantially in accordance with any accompanying written materials for a period of 90 days from the date of receipt. **CONTROL LIFE PTY LTD DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH REGARD TO THE Control Life Pty Ltd SOFTWARE. THE LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS.**

6. LICENSEE REMEDIES. Control Life Pty Ltd's entire liability and your exclusive remedy shall be repair or replacement of the Control Life Pty Ltd Software that does not meet Control Life Pty Ltd's Limited Warranty and which is returned to Control Life Pty Ltd. The Limited Warranty is void if failure of the Control Life Pty Ltd Software has resulted from accident, abuse, or misapplication. Any replacement Control Life Pty Ltd Software will be warranted for the remainder of the original warranty period or 30 days, whichever is longer. Outside of Australia, these remedies may not be available.

7. NO LIABILITY FOR CONSEQUENTIAL DAMAGES. IN NO EVENT SHALL CONTROL LIFE PTY LTD BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OR THE INABILITY TO USE THIS Control Life Pty Ltd PRODUCT, EVEN IF CONTROL LIFE PTY LTD HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/COUNTRIES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

This Agreement is governed by the laws of the State of NSW and Australia, and all disputes will be resolved in the courts of Australia.

Introduction

This is a reference manual to describe the interface provided between an AMX NetLinx system and Clipsal C-Bus Lighting System. C-Bus 5500PC interface supports RS232 control. The interface firmware must be version 4.05.00 or later. The required serial communication settings are a baud rate of 9600, 8 data bits, 1 stop bit, no parity, and handshaking off. The wiring diagram for this cable is as follows:

AMX		5500PC
TX(3)	-----	TX(3)
RX(2)	-----	RX(2)
Gnd(5)	-----	Gnd(5)

The control adapters are meant to be used in conjunction with a compatible C-Bus Lighting control unit. See the Clipsal web site for more details.

This module was written using NetLinx Studio version v2.6 build 2.6.0.191, based on Standard NetLinx API (SNAPI) R.1.9.0

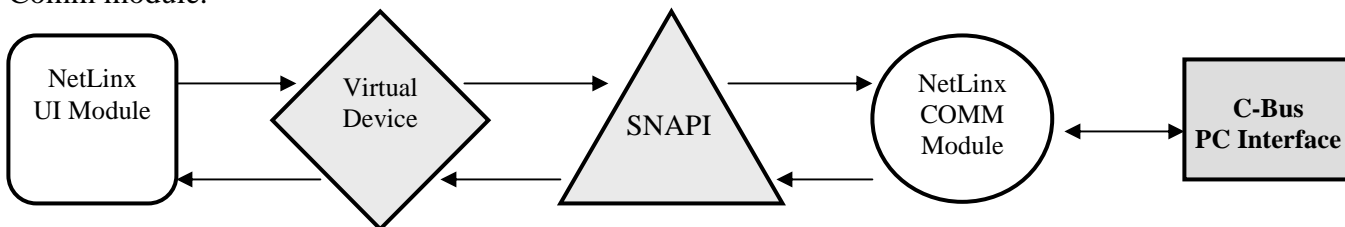
This module only support C-Bus lighting control in the local network (Network Address (Hex): FE). For more details, please refer to the Clipsal web site.

Overview

The COMM module translates between the standard interface described below and the device serial protocol. It parses the buffer for responses from the device, sends strings to control the device, and receives commands from the UI module.

A sample User Interface (UI) module is also provided. This module uses the standard interface described below and parses the command responses for feedback.

The following diagram gives a graphical view of the interface between the interface code and the NetLinx Comm module.



Some functionality in the device interface may not be implemented in the API interface. In cases where device functions are desired but not API-supported, the PASSTHRU command may be used to send any and all device-protocol commands to the device. See the PASSTHRU command and [the Adding Functions to Modules](#) section for more information.

A sample UI module and a touch panel file are provided in the module package. These are not intended to cover every possible application, but can be expanded as needed by a dealer to meet the requirements of a particular installation.

Implementation

To interface to the CL_C-Bus_Lighting_Comm_dr1_0_0 module, the programmer must perform the following steps:

1. Define the device ID for the unit that will be controlled.
2. Define the virtual device ID that the CL_C-Bus_Lighting_Comm_dr1_0_0 module will use to communicate with the main program and User Interface. Virtual devices use device numbers 31000 - 32000.
3. The CL_C-Bus_Lighting_Comm_dr1_0_0 module must be included in the program with a DEFINE_MODULE command. This command starts execution of the module and passes in the following key information: the device ID, and the virtual device ID for communicating to the main program.

An example of how to do this is shown below.

```

DEFINE_DEVICE
dvTP           = 10001:1:0 // The touch panel used for output
dvLight       = 5001:1:0 // C-Bus Serial Control
vdvLight      = 31001:1:0 // The virtual device use for communication between the
                        // Comm module interface and User_Interface (UI) module interface

DEFINE_START // Place define_module calls to the very end of the define_start section.
// Comm module
DEFINE_MODULE 'CL_C-Bus_Lighting_Comm_dr1_0_0 'comm.(vdvLight, dvLight)

```

Channels

The UI module controls the device via channel events (NetLinx commands *pulse, on, and off*) sent to the COMM module. The channels supported by the COMM module are listed below. These channels are associated with the virtual device(s) and are independent of the channels associated with the touch panel device.

Note: An '*' indicates an extension to the standard API.

Channel	Description
*238	Tx: Transmitting Data
*239	Rx: Receiving Data
251	ON: Device communicating (feedback only) OFF: Device not communicating (feedback only)
252	ON: Data initialized (feedback only) OFF: Data not initialized (feedback only)

Table 1 - Virtual Device Channel Events

Command Control

The UI module controls the device via command events (NetLinx command *send_command*) sent to the COMM module. The commands supported by the COMM module are listed below.

Note: An '*' indicates an extension to the standard API.

Command	Description
DEBUG-<state>	<p>Set the debug feature. The module will display debug statements to device 0.</p> <p><state> 1 - ERROR (default) 2 - WARNING 3 - INFO 4 - DEBUG</p> <p>Example: DEBUG-3</p>
?DEBUG	<p>Get the debug state.</p> <p>Example: ?DEBUG</p>
?FWVERSION	<p>Query for the device firmware version</p> <p>Example: ?FWVERSION</p> <p>Note: Please use the "PROPERTY-DEVICE-REVISION" Command that described later to update the value of firmware version if necessary.</p>
?LIGHTSYSTEMSTATE-<address>	<p>Query the state of a light, at the given light <address>. Responds with "LIGHTSYSTEMSTATE-<address>,<state>" where <state> is ON or OFF. This command is relevant for light loads and presets/scenes.</p> <p><address>: light address. See programming notes.</p> <p>Example: ?LIGHTSYSTEMSTATE-FE:38:01</p>

<p>LIGHTSYSTEMSTATE- <address>,<state></p>	<p>Set the state of a light, at the given light <address>. This command is relevant for light loads and presets/scenes.</p> <p><address>: light address. See programming notes. <state>: ON, OFF, TOGGLE</p> <p>Example:</p> <p>LIGHTSYSTEMSTATE- FE:38:01,ON</p> <p>Note: You may set a preset to a <state> of ON only (recall preset). A <state> of OFF or TOGGLE has no meaning for presets.</p>
<p>?LIGHTSYSTEMLEVEL-<address></p>	<p>Query the level of a light, at the given light <address>. Responds with "LIGHTSYSTEMLEVEL-<address>,<level>", where <level> is 0-255. This command is relevant for light loads only.</p> <p><address>: light address. See programming notes.</p> <p>Example:</p> <p>?LIGHTSYSTEMLEVEL-FE:38:01</p>
<p>LIGHTSYSTEMLEVEL- <address>,<level>[,<time>]</p>	<p>Set the level of a light, at the given light <address>, to a given <level>, ramped over the specified <time>. This command is relevant for light loads/dimmers only.</p> <p><address>: light address. See programming notes. <level>: 0..255 <time>:optional time parameter.</p> <p>There are set fade-rates of 0, 4, 8, 12, 20, 30, 40, 60 (1min), 90 (1.5min), 120 (2min), 180 (3min), 300 (5min), 420 (7Min), 600 (10min), 900 (15min) and 1020 (17min). If values other than these are entered, the next highest fade rate is automatically selected.</p> <p>Example:</p> <p>LIGHTSYSTEMLEVEL- FE:38:01,255 (turn light matching the address to level 255 immediately)</p> <p>LIGHTSYSTEMLEVEL- FE:38:01,10 (turn light matching the address to level 128 in 12 seconds. i.e. the next highest fade rate after 10 seconds)</p> <p>LIGHTSYSTEMLEVEL- FE:38:01,62 (turn off light matching the address in 1.5 minutes. i.e. the next highest fade rate after 60 seconds)</p>

<p>LIGHTSYSTEMRAMP- <address>, <state></p>	<p>Ramp Up/Down a light level, at the given light <address>, until LIGHTSYSTEMRAMP-<address>,STOP is sent. This command is relevant for light loads/dimmers only.</p> <p><address>: light address. See programming notes <state>: UP, DOWN, STOP</p> <p>Example:</p> <p>LIGHTSYSTEMRAMP- FE:38:01,UP</p>
<p>PASSBACK-<state></p>	<p>Set the passback feature. If enabled, the module will forward all replies received by the module from the device to the application/UI side for processing. The data is received a string events to the virtual device defined. This will enable you to handle replies for commands that are not directly supported by the module. For more information, see the "Adding Functions to Modules" section. Also see PASSTHRU-<cmd>.</p> <p><state> 0 off (default at reboot) 1 on</p> <p>Example:</p> <p>PASSBACK-1</p>
<p>PASSTHRU-<cmd></p>	<p>Allows user the capability of sending commands directly to whatever unit is attached without processing by the module. User must be aware of the protocol implemented by the unit to use this command. This gives the user access to features that may not be directly supported by the module. For more information, see the "Adding Functions to Modules" section. The duet communication module adds the start-of-text and end-of-text characters for you so you need not include them with the passthru command.</p> <p><cmd>: command to send to unit</p> <p>Example:</p> <p>PASSTHRU-G</p>

<p>?PROPERTY-<key></p>	<p>Get the module properties. If a property is not set, the module will return an empty string.</p> <p><key>:</p> <p>MODULE-NAME (Name of this module)</p> <p>MODULE-VERSION (Version number of this module. The version is in the format: major.minor.micro)</p> <p>DEVICE-MAKE (The manufacturer name)</p> <p>DEVICE-MODEL (The specific model number of the device being configured.)</p> <p>DEVICE-CATEGORY (The control method used by the device)</p> <p>DEVICE-REVISION (The firmware version installed within the device being used.)</p> <p>DEVICE-CHANNELS (The number of available device channels)</p> <p>DEVICE-LEVELS (The number of available device levels)</p> <p>UI-TEMPLATE (Specifies the type of UI required for a given module.)</p> <p>DEVICE-GUID (This is an abbreviation for Device Global Unique Identification.)</p> <p>MODULE-CONTACTADDRESS(The contact address of Control Life Pty Ltd)</p> <p>MODULE-DESCRIPTION (This is a short description of this module.)</p> <p>MODULE-DOCURL (This is a URL used to document this module.)</p> <p>MODULE-UPDATELOCATION (If the module is ever updated at some later date, this is the location that should be used (if present) to retrieve the updated tko files.)</p> <p>MODULE-COPYRIGHT (The copyright information for this module.)</p> <p>MODULE-VENDOR (description of the vendor: Control Life Pty Ltd)</p> <p>DEVICE-TYPE (Type of Device: LightSystem)</p> <p>Example:</p> <p>?PROPERTY-</p>
------------------------------	---

PROPERTY-<key> , <value>	<p>Sets the module property. You must REINITialize the module for the new settings to take affect.</p> <p><key>: DEVICE-REVISION (Update the firmware version of the device) <value>: xxx.xxx.xxx</p> <p>Example: PROPERTY- DEVICE-REVISION,4.05.00</p>
REINIT	<p>Reinitializes the module and communications with the device. Any commands pending in the module will be cleared.</p> <p>REINIT</p>
?VERSION	<p>Get the module version.</p> <p>Example: ?VERSION</p>

Table 2 – Send Command Definitions

Command Feedback

The COMM module provides feedback to the User Interface module via command events. The commands supported are listed below.

PLEASE NOTE: Feedback is only provided when there is a state change. If no state change resulted from the command sent in, then no feedback will be returned.

Command	Description
DEBUG-<state>	Reply for the debug feature. <state>: 1 = error 2 = warning 3 = info 4 = debug Example: DEBUG-1
FWVERSION-<version>	Reply for the firmware version of the device Example: FWVERSION-4.05.00
KEYPADSYSTEMBUTTONSTATE-<address>,<state>	Reply to a keypad button push or release. <address>: keypad address. See programming notes <state>: PUSH, RELEASE Example: KEYPADSYSTEMBUTTONSTATE-11:B1,PUSH
LIGHTSYSTEMSTATE-<address>,<state>	State of a light changed at the given <address>. <address>: light address. See programming notes <state>: ON, OFF Example: LIGHTSYSTEMSTATE-FE:38:01,ON LIGHTSYSTEMSTATE- FE:38:02,OFF
LIGHTSYSTEMLEVEL-<address>,<level>	Level of a light changed at the given <address>. <address>: light address. See programming notes <level> : 0..255 Example: LIGHTSYSTEMLEVEL-FE:38:01,200

LIGHTSYSTEMRAMP- <address>, <state>	<p>Light is ramping at the given <address>.</p> <p><address>: light address. See programming notes. <state>: UP, DOWN, STOP</p> <p>Example:</p> <p>LIGHTSYSTEMRAMP- FE:38:01,STOP</p>
PASSBACK-<data>	<p>Reply received from the passback feature. The data received is the raw data as received from the device.</p> <p><data>: string</p> <p>Example:</p> <p>PASSBACK-\05FF00730738E0\$0D</p>
PROPERTY-<key>, <value>	<p>Returns the module properties. If a property is not set, the module will return an empty string. All property keys are case-sensitive.</p> <p>Example:</p> <p>PROPERTY-MODULE-VENDOR,Control Life Pty Ltd</p>
VERSION-<version>	<p>Reply for the module version</p> <p><version>: string</p> <p>Example:</p> <p>VERSION-1.0.0</p>

Table 3 - Command Feedback Definitions

Programming Notes

- This module uses a specific addressing scheme to address a certain lighting component. The general format of any light address is:

<Network Number>:<Application Number>:<Group Number>

Where <Network Number>: FE (Hex) for local network. This module supports local network only.

<Application Number>: 01-FE (Hex), where:

01-0F	free use for developers
30-5E	Lighting (Default is 38)
88	Heating
CA	Trigger Control
CB	Enable Control
70	Ventilation (dampers and fans)
71	Irrigation Control
72	Pool, spa, fountain and pond control

<Group Number>: 01-255(Hex)

This module can also accept address formats:

<Application Number>:<Group Number> (assume network number is FE (Hex))

or

<Group Number> (assume network number is FE (Hex) and application number is 38 (Hex))

This module uses the concept of a preset to represent a Lutron scene. Each dimmer is considered an individually controllable zone or light load on the particular control unit. This API makes no distinction between a dimmer and preset other than the <address> parameter format.

- The module will not automatically reinitialize itself after a reboot. You must determine when to do this step. Using this method will allow greater control over when the re-initialization sequence should be performed in a loaded system. In order to communicate with the device, you must issue a REINIT.
- Once connected, the module will automatically determine if it lost connection with the device and will attempt to reinitialize automatically when that happens. The responsiveness of this feature is dependent on the DefaultPollRate property value.
- C-bus will send a level feedback of FF when a light group is ramping up and a level of 01 when ramping down by a C-Bus wall panel. It will affect the level feedback of the module. However, the C-bus will send a correct level feedback when the ramping stopped.

Adding Functions to Modules

Commands to the device

This module supplies a mechanism to allow additional device features to be added to software using the module. This is the 'PASSTHRU-' command, which allows protocol strings to be passed through the module. The device-specific protocol must be known in order to use this feature. **This passthru functionality is available only over RS232.**

As an example, suppose that a module for a projector has not implemented the 'white balance adjustment' feature. The command that the projector protocol requires is 03H, 10H, 05H, 14H, followed by a checksum. The documentation for the 'PASSTHRU-' command specifies that the module will automatically generate the checksum. In this case, the following string should be sent from the UI code to implement 'white balance adjustment'.

```
send_command vdvDevice, "PASSTHRU-', $03, $10, $05, $14"
```

The reason to use 'PASSTHRU-' instead of sending a protocol string directly to the device port is that the device may require command queuing, calculation of checksums, or other internal processing, which would not be done if the string was sent directly. Because of this, it is best to filter all communication TO the device through the module. (The module documentation will indicate any processing that will be automatically done to the 'PASSTHRU-' command like checksum calculation.)