Serial Interface User Guide
(for firmware version 4)

C-Bus Serial Interface & C-Bus Development Kit

Document Number: CBUS-SIUG

Issue: 1.17

Date: 9 December 2008

CHANGE HISTORY

Date	Change Reference	Comments
7 Dec 2001	-	Original
5 Feb 2002	Issue 1.1	Minor typographical corrections only.
6 Mar 2002	Issue 1.2	Added limitation about use of serial interface at speeds below 9600 bits/s.
10 May 2002	Issue 1.3	Changed C-Bus Inside references to C-Bus Enabled.
27 Aug 02	Issue 1.4	Better explanation of LOCAL_SAL mode. Updates to CIS restricted sections.
29 Apr 03	Issue 1.5	Removed reference to non-existent documents
8 May 03	Issue 1.6	Updated change history for PCI f/ware ver 4.2.0
24 Jul 03	Issue 1.7	Added section & description for compound devices, and use of the Interface Options 1 Power Up Settings. Added custom type convention for chipset customers under the C-Bus Enabled program.
25 Aug 03	Issue 1.8	Added single quote reply from Serial Interface and description.
8 Sep 03	Issue 1.9	Added version 4.3.0 to the change history.
10 Nov 03	Issue 1.10	Make corrections to section on “Extended Format Status Replies”
1 Oct 04	Issue 1.11	Update QA logo
17 Mar 05	Issue 1.12	Corrected basic mode short form monitored SAL diagram. Added ver 4.4.0 to change history.
10 Mar 06	Issue 1.13	Minor updates
28 Sep 06	Issue 1.14	Clarify Monitored SAL can have empty payload
10 May 07	Issue 1.16	Correct message representation in section 10.2
9 Dec 08	Issue 1.17	Typos before publication

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope

This document describes the protocol used by the C-Bus Serial Interface. This device is used as a gateway onto a C-Bus network and permits control and monitoring of a C-Bus network using any serial device, such as an IBM-compatible PC, as shown in Figure 1. The C-Bus Serial Interface also provides third party access to C-Bus, and is used by Clipsal as the access method into C-Bus for its high value-added devices.

The C-Bus serial interface is used in the PC Interface (5100PC and 5500PC) and in the C-Bus SIM (5000SM and 5000SM/2).

This document refers to the functionality of Serial Interface version 4.0.00 and higher.

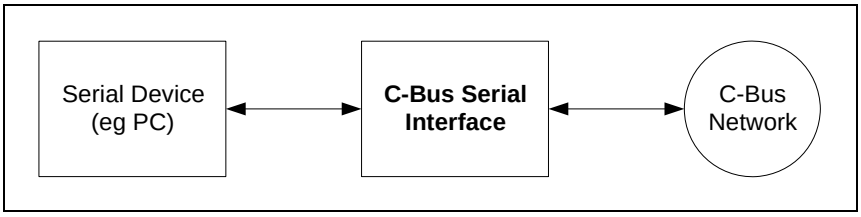


Figure 1 Scope of Serial Interface

1.2 Referenced Documents

Document Reference	Title	Summary / Purpose
CBUS-APP	C-Bus Application Messages & Behaviour	Requirements for the assigned C-Bus Application Addresses, the messages permitted for each Application, and Application behaviour. Multiple chapters, each separately issued and controlled.
CBUS-IFR	C-Bus Interface Requirements	Description of the different levels of C-Bus interface compatibility, and the requirements for devices at each level.
ISO 7498	Basic Reference Model for Open Systems Interconnection	International Standard. Describes the 7 layer reference model for computer networks.

1.3 Definition of Terms

Application A well-defined set of behaviours for one or more devices connected to a C-Bus network.

Application Address A unique number used to access the behaviour of a group of devices connected to a C-Bus network.

Compound Device	A device containing a C-Bus Serial Interface and a control processor, each separately powered.
Network Variable	A network wide control variable maintained and/or controlled by C-Bus units. Within the Lighting Application, a Network Variable is called a Group Address Variable (GAV), and its value is called a Group Address Level.
Group Address Variable	A Network Variable, which can be separately addressed in an Application. Usually used on connection with the C-Bus Lighting Application.
Parameter	A stored property of the Serial Interface
Serial Device	A device attached to the Serial Interface
Serial Interface	The C-Bus Serial Interface device, which converts the C-Bus protocol to and from an asynchronous serial RS-232 protocol. Allows complex devices like Personal Computers or similar to connect to and control devices on a C-Bus network.
Unit	A device attached to a C-Bus network

1.4 Acronyms and Abbreviations

APDU	Application Protocol Data Unit
ASCII	American Standard Code for Information Interchange
CAL	Common Application Language
CRC	Cyclic Redundancy Check
GAV	Group Address Variable
ISO	International Standards Organisation
LED	Light Emitting Diode
NPDU	Network Protocol Data unit
PC	Personal Computer
PCI	Protocol Control Information
PCN	Parameter Change Notify
PDU	Protocol Data Unit
PUN	Power Up notofy
RAM	Random Access Memory
SAL	Specific Application Language
SIM	Single In-line Module

1.5 Presentation Format for Numbers

Throughout this document, numbers are presented in decimal (base 10), hexadecimal (base 16) and binary (base 2). The following conventions are used:

a. Numbers without any prefix are decimal (for example, 127);

b. Numbers with a $ prefix are hexadecimal (for example, $7F);

c. Numbers with a % prefix are binary (for example %01111111).

In some diagrams, the context clearly shows that a number is binary, and the % prefix is omitted.

In communication with the C-Bus Serial Interface, all information transfer occurs using ASCII hexadecimal (two bytes transferred per byte of information). Where this is clear by context, the $ prefix is omitted.

1.6 Typographic Presentation

The following conventions are used in this guide:

a. Normal descriptive text (like this) is presented in a standard Helvetica font with no emphasis;

b. Section headings are always numbered, appear in Helvetica, and are always boldfaced. Major section (level 1) headings are all upper case. All other headings are italicised.

c. Serial Interface option names or reference to the options appear in upper case italics, for example *MONITOR*, *CONNECT*.

d. Important points or recommendations in the text appear with bold face, italicised, inside a box, with a stop sign, for example:

Important: blah blah.

e. Cross reference between syntax diagrams uses the name in the diagram, underlined. For example Status Report

2 C-BUS ENABLED

"C-Bus Enabled" is an approach used by Clipsal to specify, test and certify that devices attached to a C-Bus network:

a. Comply with the network protocols;

b. Avoid excessive bus traffic; and

c. Have acceptable interoperability with other C-Bus devices.

The above objectives are achieved by specifying the interface requirements for C-Bus devices, and by specifying, for each application, the message traffic that may be sent over C-Bus. Finally, guidance is provided for the use of the C-Bus Serial Interface.

The documents and their relationships are shown in Figure 2.

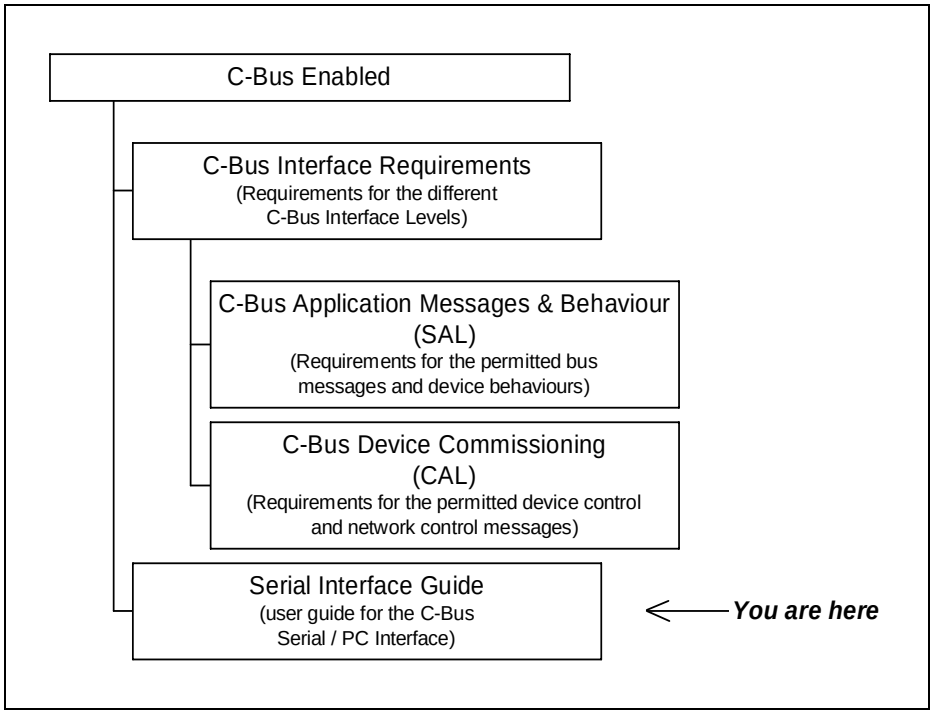


Figure 2 C-Bus Enabled Documents and Relationships

3 C-BUS INTRODUCTION AND CONCEPTS

3.1 ISO 7-Layer Protocol Model

The International Standards Organisation (ISO) has defined a model for network protocols, known as the ISO 7-layer model, and described in ISO 7498.

This model defines the functions and operation of different layers of a network protocol. C-Bus has been designed in accordance with the ISO model, with the omission of layers not relevant. The model defines some of the terms used in this document.

Figure 3 shows the network protocol model used for C-Bus. In this figure, Protocol Control Information (PCI) is used by different protocol layers to add special information used to aid in transporting the information. A Protocol Data Unit (PDU) refers to the information content being handled by a layer. For example, the Link Layer transports a Network PDU (NPDU). This contains embedded information used by the Network Layer, but the Link layer neither knows or cares about this.

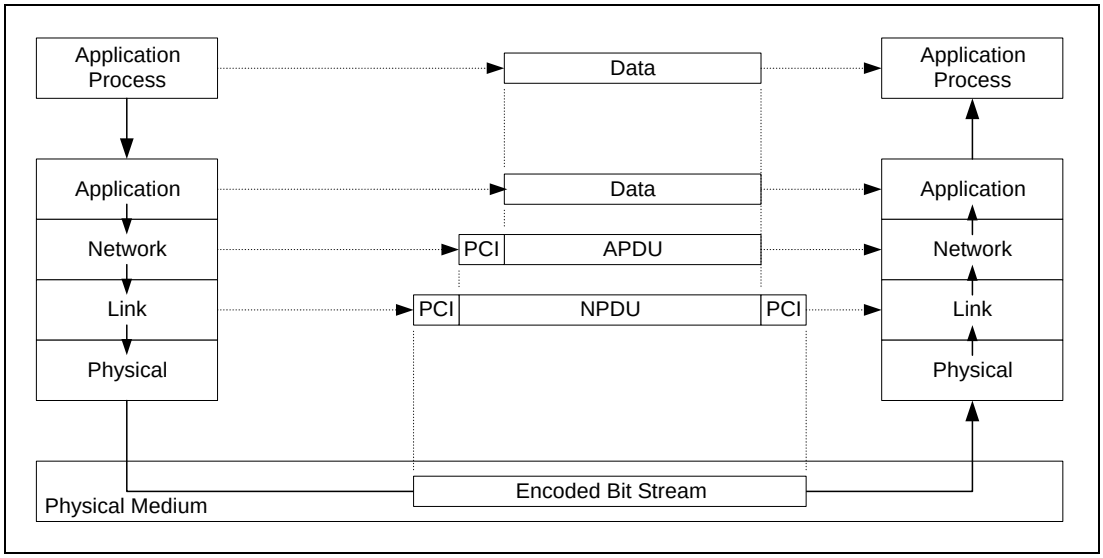


Figure 3 C-Bus Protocol Layers and Interactions

3.2 Types of Information Transported by C-Bus

A C-Bus network can transport the following message types:

Point to Point: Used to transport a message between two single units on the network. Normally used for loading configuration data into a unit, or for network management and control.

Point to Point messages are addressed to a destination unit.

Point to point messages can also be transmitted through one or more C-Bus bridges. In this case, a Network PCI is used to specify the path through bridges to the destination network and unit.

The data transferred (the APDU) is known as the C-Bus Common Application Language (CAL)**Error! Reference source not found.**.

Point to Multi-Point: Used to transport a message from one transmitting unit to many receiving units, simultaneously. Point to Multi-Point messages allow creation of Network Variables, and ensure that Network Variables are updated simultaneously in all units.

Point to Multi-Point messages are addressed to a destination Application. Units can participate in one or more applications, depending on their purpose.

The data transferred (the APDU) is known as the C-Bus Specific Application Language (SAL), and is defined in the document CBUS-APP.

Point to Point to Multi-Point: Used to transport a message from one transmitting unit to many receiving units, simultaneously, but where the receiving units are in a remote network.

Similarly to Point to Point messages, Network Variables can be created and will be updated simultaneously in all receiving units.

Point to Point to Multi-Point messages are addressed to a C-Bus bridge, and contain a Network PCI that specifies the path to the destination network and Application.

As above, the data transferred (the APDU) is known as the C-Bus Specific Application Language (SAL).

3.3 Addresses

All C-Bus addresses are 1 byte, and encode $00 to $FF.

Unit Address: Every unit attached to a C-Bus network has a unique unit address, used when transmitting information to that unit exclusively. The information transmitted is referred to as CAL DATA, and is used for unit configuration and network management.

All units are shipped from the factory with the Unit Address set to $FF. The C-Bus installation software is able to scan a network, detect multiple units at the same address, and can then re-assign unit addresses.

Application Address: Application addresses are used to define related information transfer over a network. Different applications can transfer different types of information, depending on the application. (For example, lighting control commands are unrelated to clock update commands. These are placed into separate applications.) A unit can be a party to more than one application.

Important: Application Addresses follow a fixed convention, described in document CBUS-APP.

Application Address $FF is reserved for network management functions and must not be used.

The most common C-Bus Application is "Lighting, Switching and Load Control", which is summarised in section 6.

Network Address: Networks are addressed with a network address.

By convention, the network address corresponds to the unit address of the *other* side of the bridge, as shown in Figure 4. A bridge has two unit addresses, one in each network. In Figure 4, Network A1 can send data to network A3. It would send to unit addresses A2 and then A3.

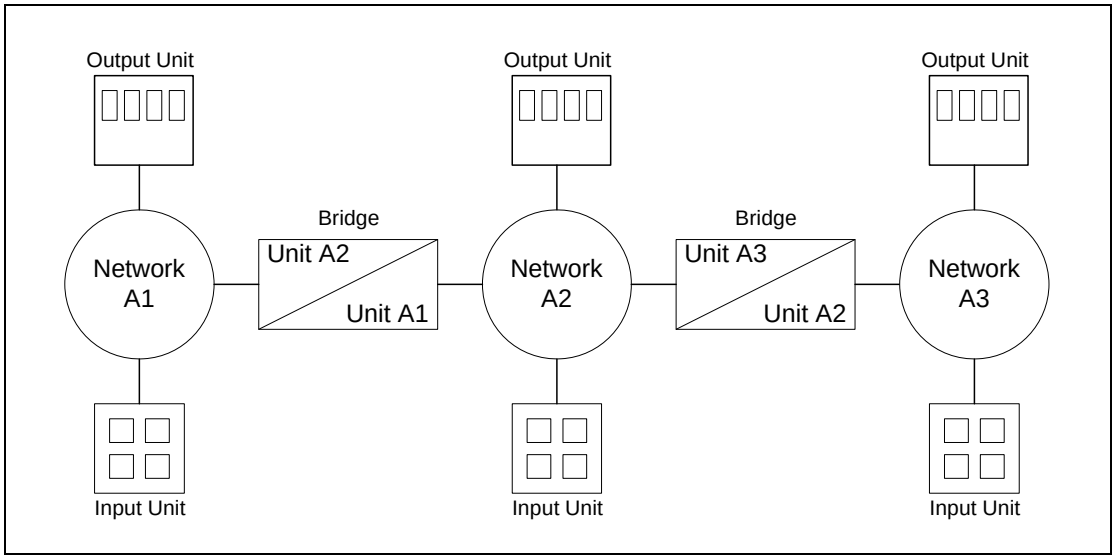


Figure 4 C-Bus Network Addresses

3.4 Header

Each transmission onto C-Bus begins with a Header.

The Serial Interface takes a Header as part of its protocol and initially passes this through to C-Bus, unchanged. During transmission by the Serial Interface, the retry and dynamic priority fields may be automatically changed. This is transparent to a user of the Serial Interface.

The C-Bus Header and the Header used for transmissions to the Serial Interface has the structure shown in Figure 5.

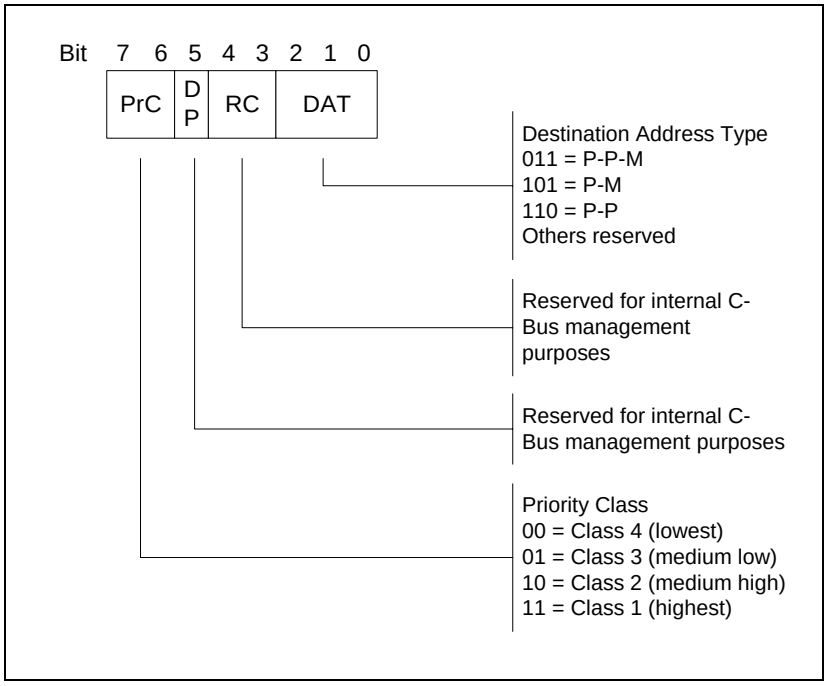


Figure 5 C-Bus Header Structure

When determining a Header, the following points must be noted:

a. The 'DP' and 'RC; fields shown as reserved for C-Bus management purposes must always be set to 0.

b. The priority class is normally either Class 4 (lowest) or Class 3 (medium low). Some C-Bus management functions make use of the higher priority classes. Use by application devices of Priority Class 2 is strongly discouraged.

Important: Use of Priority Class 1 is forbidden without prior consultation and approval by Clipsal.

Consequently, the most common headers are:

$03 - Point – point – multipoint, lowest priority class

$05 - Point – multipoint, lowest priority class

$06 - Point – point, lowest priority class

3.5 Network Variables and Group Address Variables

A C-Bus network allows messages to be transmitted to an Application. All units that participate in the Application receive the same message at the same time (known as a multicast). The C-Bus protocol guarantees that either *all* units or *no* units receive the message.

As a consequence, a C-Bus network allows the creation of a Network Variable. This is simply a value in an Application, where all units having an interest in the value are assured it will be updated simultaneously.

Network Variables are extremely powerful. They allow the creation of a distributed system where no single unit on the network acts as a master controller. Instead, the intelligence is distributed amongst all units on the network.

The most common use of Network Variables is the C-Bus Lighting Application Group Address Variable (GAV). A GAV in the Lighting Application creates the equivalent to conventional wiring – it acts as the link between user operated input units (light switches) and the output units which switch the load.

For example, three input units and an output unit could be grouped to create three way control switching. The four devices would share the same GAV. Because the GAV is not just binary (on and off), but has a range from 0 to 255, it is easy to create a multi-way dimmer. The lamp brightness is determined by the value of the GAV (0 = fully off, 255 = fully on, etc). Many input units can then control the dimming level of a single lamp – something which is impossible with conventional wiring.

Important: The format of information stored in a Network Variable is dependent on the Application and the Network Variable number.

These definitions are created and controlled by Clipsal. The document set CBUS-APP contains the definitions of the C-Bus Applications, their message format and their behaviour.

To ensure inter-operation of different C-Bus devices, it is vitally important that all devices use only the defined Application Numbers and SAL message formats defined in CBUS-APP.

3.6 Checksums

Checksums are used throughout C-Bus to provide an error detection capability.

All C-Bus checksums are calculated as the 2 ’s complement of the modulo 256 sum of the preceding bytes.

In other words:

a. Sum all of the bytes (for the Serial Interface, the hex character pairs);

b. Find the remainder when the sum is divided by 256;

c. Take the 2’s complement (invert the bits and then add 1) of the remainder.

This checksum has the property that when all of the bytes, including the checksum, are summed (modulo 256), the result will always be zero if the check passes, and non-zero will indicate the presence of an error in the data stream.

The Serial Interface checksums are performed on the hexadecimal data (the ASCII character pairs).

For example, suppose a Lighting command is being sent to the Serial Interface:

`\0538007920`

Summing the bytes: $05 + $38 + $79 + $20 = $D6

Modulo 256 ($100), this is $D6.

The checksum (2’s complement and add 1) = $2A.

Checking: $05 + $38 + $79 + $20 + $2A = $100, which is $00 modulo 256 ($100).

So, if the Serial Interface has the *SRCHK* option on, this command would be sent as:

\05380079202A

3.7 Notation

In this User Guide, the syntax for commands sent to the Serial Interface and replies received from the Serial Interface are shown using syntax diagrams. These diagrams show the sequence of information needed. Figure 6 shows the notation.

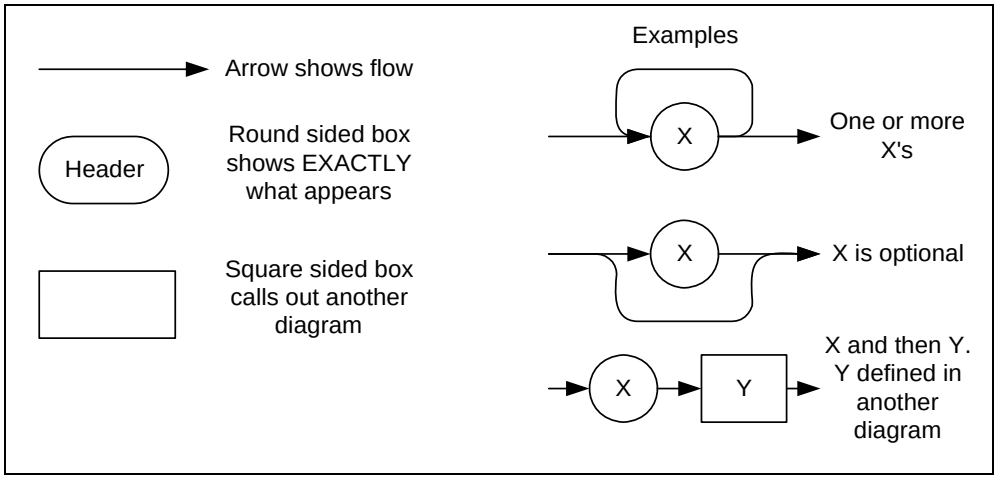


Figure 6 Syntax Diagram notation

4 SERIAL INTERFACE

The Serial Interface protocol is described by what is sent to the interface, and what the interface sends back. The interface is not symmetric: the information sent in each direction is not identical, but there are substantial similarities.

The Serial Interface has two major modes of operation: *BASIC* Mode and *SMART* Mode. There are also a number of options which can further modify its behaviour. The major difference between *BASIC* Mode and *SMART* Mode is the way in which the Serial Interface sends its replies.

Options used to modify Serial Interface behaviour are described in section 10.

Important: This description applies to Serial Interface version 4.0.00 and later.

Some previous versions of the Serial Interface did not support all of the options or features shown.

Where possible, the differences to older versions are described using footnotes.

4.1 Power Up

The Serial Interface retrieves its modes and option settings from parameter $41 on power up. In the factory, this parameter is set to $00 so that a Serial Interface will power up in *BASIC* Mode[1].

If a user wishes to have the Serial Interface behave with non-volatile Modes and options, any changes should be set in parameter $41 as well as parameter $30.

Refer to section 10 for more information about setting parameters

4.2 Transmission to the Serial Interface

4.2.1 Structure of Commands

Commands sent to the Serial Interface comprise a combination of special characters used to control the serial interface behaviour, and sequences of hexadecimal ASCII that are used by the serial interface to form a transmission onto C-Bus.

Each command is terminated with a carriage return (ASCII character 13, hex $0D), and denoted <cr> in the sections which follow.

4.2.2 Command Length Limit

For any command sent to the Serial Interface, there cannot be more than 45 characters between the leading \ and the <cr>.

[1] Earlier version of the Serial interface powered up into *BASIC* Mode, and the mode and options were always lost when C-Bus power was removed. This behaviour can be preserved by ensuring parameter $41 is not changed from the factory default of $00.

4.2.3 Special Character: Reset

Sending a single ~ character cancels all options of the Serial Interface, and changes to *BASIC* Mode. This character can be optionally followed by <cr>.

For example:

```
~
```

After this reset sequence, the Serial Interface default behaviour is to accept point-point commands, with a default unit address set to itself. This allows the serial device to reconfigure the Serial Interface without knowing its unit address. It is preferred that the "@" syntax (refer section 4.2.7) be used instead.

4.2.4 Special Character: Cancel

Sending a single ? character in any position in a command will cancel (clear) all characters entered since the last ? or <cr> character sent.

For example:

```
AB0123?9876
```

is equivalent to sending "9876".

4.2.5 Special Character: Smart+Connect Shortcut

Sending a single | character will enter *SMART* Mode and set the *CONNECT* option. (Other modes and options are not changed.)

If used, the | should be send before and after a <cr>.

For example:

```
<cr>|<cr>
```

Important: Use of the | shortcut is discouraged.

It is preferable to set the exact combination of modes and options to suit the desired behaviour.

Refer to section 10 for information about options, and the methods of setting Serial Interface parameters.

4.2.6 Special Character: Confirmation

Inclusion of a non-hexadecimal lower case alphabetic character anywhere in the command prior to the <cr> will cause:

a. Entry to *SMART* Mode if in *BASIC* Mode, but without otherwise changing any of the other options; and

b. A confirmation to be returned for each command, without otherwise affecting any interpretation of the command.

Clipsal recommended that if a confirmation character is used, the characters follow a sequence from “g” to “z”, and then roll back around to “g” again. This permits unambiguous association of a transmission and its confirmation.

The confirmation is described in section 4.3.3.3.

4.2.7 Special Character: Direct Command Access

The character @ can be used to send a Device Management (CAL) command directly to the Serial Interface, bypassing all addressing, and irrespective of any other options or modes[2].

This is equivalent to using the Reset character and then sending a short form command to the Serial Interface, but does not require the preceding Reset.

For example the Identify 2 command:

@2102

is equivalent to:

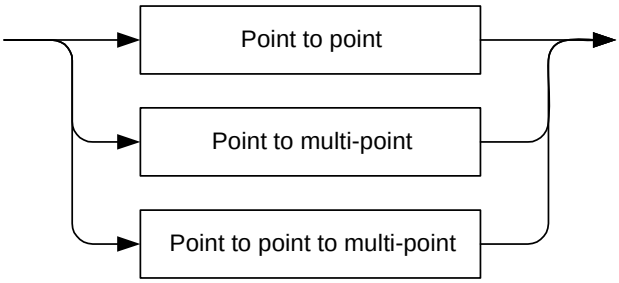
~2102

but has the advantage that all of the existing modes and options are unaffected (whereas the ~ would reset them).

[2] Earlier versions of the Serial Interface do not support the @ command prefix. For those versions, the same effect can be achieved using ~, sending the command into the Serial Interface, and restoring the previous modes and options afterwards.

4.2.8 Format of Commands Transmitted to the Serial Interface

Commands for transmission to the Serial Interface are either point-point, point-multipoint or point-point-multipoint, as shown below:



Each of these different command types is described in the following sections.

4.2.9 Effect of Options on Transmission from the Serial Interface

Serial Interface Options have the following effects for transmission:

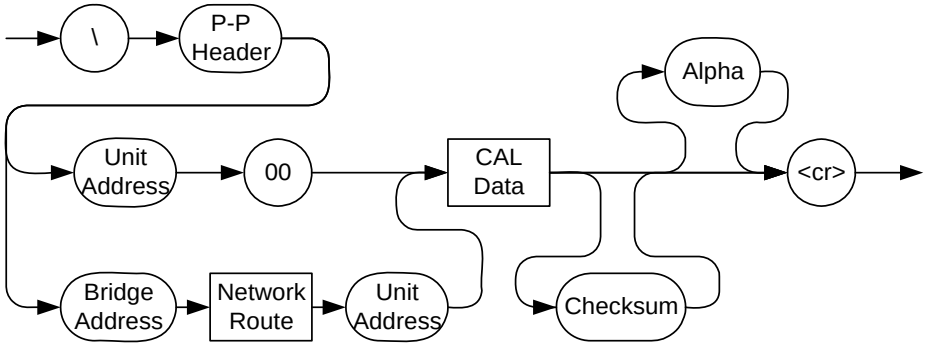
Mode / Option	Effect
LOCAL_SAL[3] *(option)*	When ON, the Serial Interface outputs C-Bus Point to Multi-Point (SAL) messages in a form which looks identical to a locally connected unit (one which does not use a Serial Interface, for example a key input unit). When OFF, the Serial Interface outputs SAL messages using a form which may not be compatible with the *CONNECT* option of some other Serial Interfaces.

Recommendation: All new C-Bus devices connecting using the Serial Interface should have the LOCAL_SAL option switched ON.

[3] Earlier versions of the Serial Interface do not support the *LOCAL_SAL* option. Setting this option in earlier versions has no effect.

4.2.9.1 Point to Point Commands

Point to Point commands have the format:



Notes:

a. P-P Header is a byte, as defined in section 3.4. In binary, its value must be %xx000110. Normal value: $06.

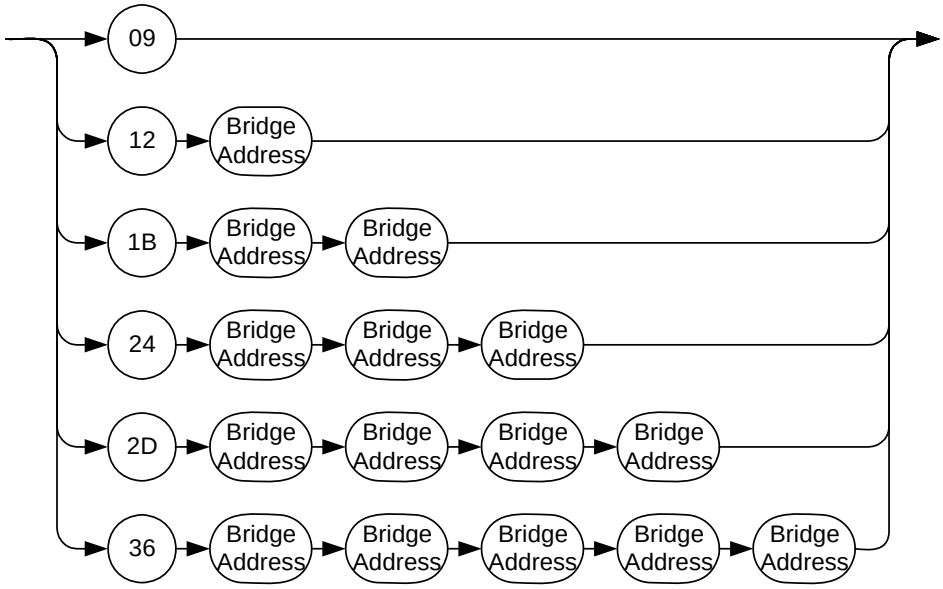
b. Unit Address and Bridge Address are bytes – refer section 3.3.

c. Checksum only applies if *Input Checksum (SRCHK)* option has been set, in which case omission of the checksum is invalid. Refer to sections 3.6 and 10.3.2.

d. Alpha is a lower case alphabetic character between ‘g’ and ‘z’, which, if present will cause a confirmation status of the transmission to the Serial Interface to be returned. Refer to section 4.2.6.

e. CAL Data defines the commands which can be sent to a unit. These commands comprise a sequence of bytes, and are described in section 7.

Network Route applies when a Point to Point message, or a Point to Point to Multi-Point message, is addressed through one or more bridges, and is defined as:



Examples of Point to Point commands:

\0603002102D4

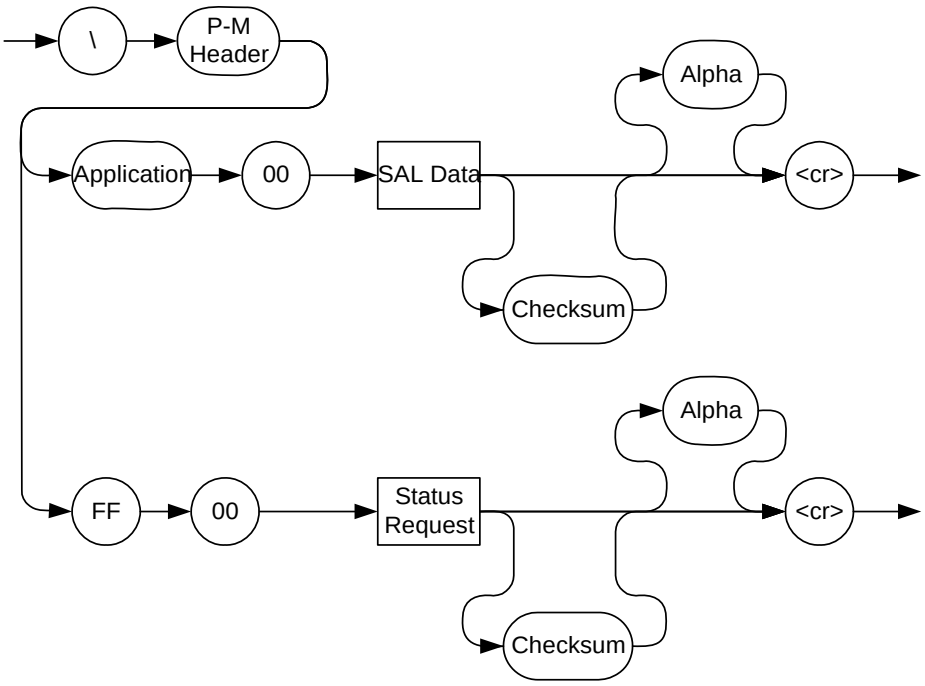
This Point to Point command is sent to unit address $03. The data sent to the unit is $2102, and there is a checksum of $D4. The data ($2102) is an Identify command.

\06420903210289

This Point to Point command is sent to bridge address $42 (the $09 follows the first bridge address), and then to unit address $03. The data sent to the unit is $2102, and there is a checksum of $89. The data ($2102) is an Identify command.

4.2.9.2 Point to Multi-Point Commands

Point to Multi-Point commands are the most commonly issued C-Bus commands. These have the format:



Notes:

a. P-M Header is a byte, as defined in section 3.4. In binary, its value must be %xx000101. Normal value: $05.

b. Application is a byte. Refer to document set CBUS-APP for the allocation of C-Bus Application numbers.

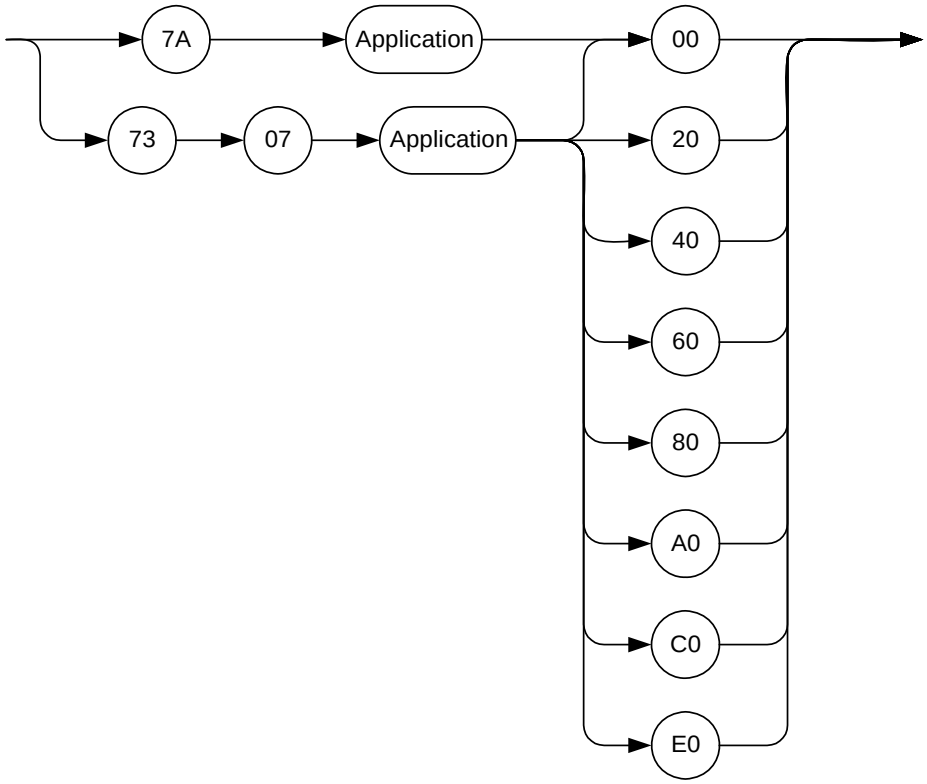
c. Checksum only applies if *Input Checksum (SRCHK)* option has been set, in which case omission of the checksum is invalid. Refer to sections 3.6 and 10.3.2.

d. Alpha is a lower case alphabetic character between ‘g’ and ‘z’, which, if present will cause a confirmation status of the transmission to the Serial Interface to be returned. Refer to section 4.2.6.

e. SAL Data defines the commands which can be sent to an Application. These commands comprise a sequence of bytes, and are fully defined in document set CBUS-APP. A summary for the C-Bus lighting Application is shown in section 6.

f. Status Reporting is only useful for those Applications that support it. Refer to document set CBUS-APP for the Application definitions and the support of status reports in Applications.

Status Request is used as part of a Point to Multi-Point command to request a C-Bus Binary Status Report or Level Report:



Notes:

a. Status Requests have two versions:
   - one version (beginning $7A) that reports the Binary State of C-Bus Group Address Variables (ie ON, OFF or Error)[4]; and
   - one version (beginning $7307) that reports the Level of C-Bus Group Address Variables.

b. The Level Status Request, using the sequence $7307, is only supported in Serial Interface version 4.0.00 and later.

c. The Level Status Request finishes with a byte that can be $00, $20, $40, etc. This byte specifies the starting Group Address Variable number to be reported. Only 32 Group Address Variables can be reported in response to a single Level Status Request.

d. The information returned from a Binary Status Request and Level Request is part of the C-Bus CAL and is described in section 7.

e. A Status Request will return information in either a standard format or an extended format, depending on the setting of the *Extended Status Format (EXSTAT)* option.

[4] The Binary Status Report can also be invoked using $FA instead of $7A. However, this usage is discouraged and support may be removed from future releases.

f. The Switched Status Report (beginning $7A) can also be used to indicate the presence or absence of Units, when issued against the Master Application (Application number $FF).

Recommendation: New devices which:

a. need to monitor status information on a C-Bus network, and

b. use the Serial Interface as their method of attachment to C-Bus

should turn the Extended Status Format (EXSTAT) option ON.

In addition to providing addressing information, the Extended Status Format allows separation of unsolicited Status Replies, and Status Replies initiated by this Serial Interface. Refer to section 7.4.

Important: A Status Request is a fast, efficient, simultaneous poll of all devices on the C-Bus network. In spite of this, Status Requests increase network loading and should be used very carefully. Excessive use of Status Requests can dramatically degrade network performance.

Clipsal recommend that devices which are permanently connected to a C-Bus network should issue a Status Request to get an initial state, and then monitor the network traffic to keep a local, up-to-date state record. Periodically (every few minutes), this local state record can be refreshed by issuing another Status Request.

Refer to CBUS-IFR for more information.

Examples of Point to Multi-Point commands:

\0538000108BA

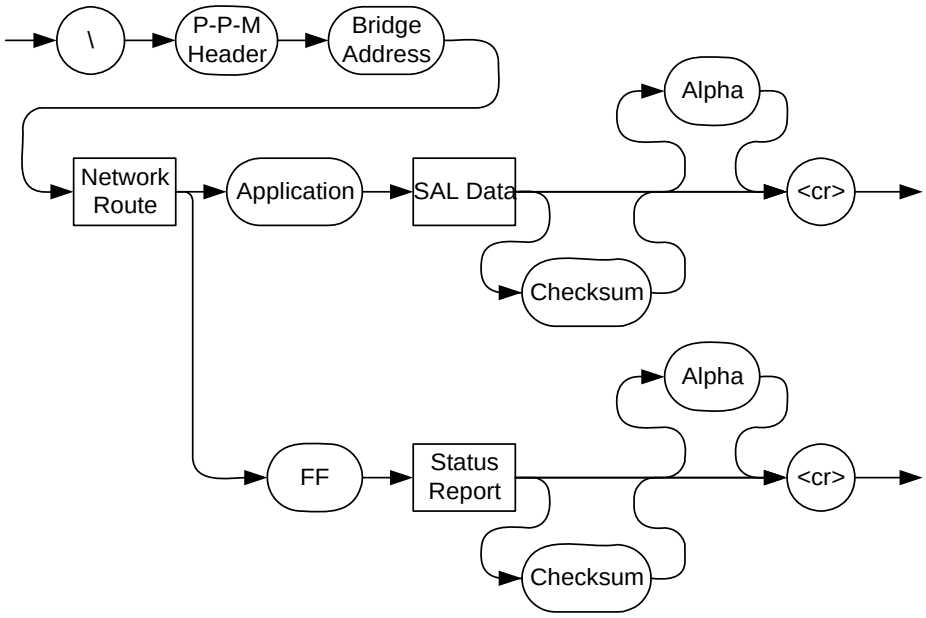
This Point to Multi-Point command is sent to Application Address $38, the standard C-Bus Lighting Application. The data sent to the Application is $0108, and there is a checksum of $BA. The data ($0108) is an Off command to Lighting Group $08.

\05FF007A38004A

This Point to Multi-Point command is a status request ($FF007A), on Application $38, the standard C-Bus Lighting Application. There is a checksum of $4A.

4.2.9.3 Point to Point to Multi-Point Commands

Point to Point to Multi-Point commands have the format:



Notes:

a. P-P-M Header is a byte, as defined in section 3.4. In binary, its value must be %xx000011. Normal value: $03.

b. Bridge Address is a byte – refer section 3.3.

c. Checksum only applies if *Input Checksum (SRCHK)* option has been set, in which case omission of the checksum is invalid. Refer to sections 3.6 and 10.3.2.

d. Alpha is a lower case alphabetic character between ‘g’ and ‘z’, which, if present will cause a confirmation status of the transmission to the Serial Interface to be returned. Refer to section 4.2.6.

e. Network Route, SAL Data, and Status Report have been defined in the preceding sections.

Example of Point to Point to Multi-Point commands:

`\03420938010871`

This Point to Point to Multi-Point command is sent to bridge address $42, and then to Application Address $38, the standard C-Bus Lighting Application. The data sent to the Application is $0108, and there is a checksum of $71. The data ($0108) is an Off command to Lighting Group $08.

4.3 Reception From the Serial Interface

Replies from the Serial Interface have two different major formats, depending on the selected mode, and are further modified by various options.

Replies fall into three categories:

a. Echo.

   In *BASIC* Mode the Serial Interface will echo characters it receives, as they are received. Echo is disabled in *SMART* Mode.

b. Response to a previous transmission to the Serial Interface.

   Some C-Bus transmissions through the Serial Interface cause a response to be returned, for example a Status Request. Generally, though, responses are the exception rather than the norm.

c. Monitored information showing C-Bus activity, but which was not initiated by the Serial Interface.

   When the *CONNECT* option, the *MONITOR* option, or both is selected, the Serial Interface will output information about traffic passing by on C-Bus. In each case respectively, it will asynchronously output SAL messages seen on the network, status messages, or both. The number of such messages arriving, and their rate is non-deterministic

Replies from the Serial Interface are presented in either a Short Form or a Long Form.

When in *BASIC* Mode, replies are always presented in the Short Form.

When in *SMART* Mode, Monitored SAL replies and some CAL Replies are presented in the Long Form. To get all CAL replies in the Long Form, the *IDMON* and *EXSTAT* options should be selected in *SMART* Mode.

4.3.1 Effect of Modes and Options on Reception from the Serial Interface

Serial Interface Modes and Options are further described in section 10. The Modes and options that can modify the presentation of replies from the Serial Interface[5] are:

Mode / Option	**Effect (when ON)**
CONNECT *(option)*	The Serial Interface can return C-Bus Point to Multi-Point (SAL) messages, at any time, because it has a connection to one or more C-Bus network applications.
SMART *(mode)*	Disables echo of characters being sent to the Serial Interface. Selects the Long Form for Point to Multi-Point (SAL) messages seen when the *CONNECT* option is set. Selects the Long Form for most Point to Point (CAL) Replies.

[5] Earlier versions of the Serial Interface do not support the *IDMON, EXSTAT, PCN or PUN* options. Setting these options in earlier versions will have no effect

Mode / Option	**Effect (when ON)**
IDMON *(option)*	Selects the Long Form for any reply for Point to Point (CAL) messages that were initiated by the Serial Interface and sent to itself. Useful only when *SMART* Mode is set.
EXSTAT *(option)*	Selects the Long Form, Extended Format for the reply for all monitored and initiated Status Requests, including Binary Status Replies. Most useful when the *SMART* Mode is set.
MONITOR *(option)*	The Serial Interface will monitor all traffic on C-Bus looking for Status Requests. When it sees any, it will return the Status Reply as Point to Point (CAL) format reply. The Status Request could originate from this Serial Interface or from any other unit on the C-Bus network. The replies when *MONITOR* is ON are further modified by the *Extended Status Format (EXSTAT)* option. *MONITOR* is normally used in conjunction with the *CONNECT* option. NOTE: In most C-Bus networks, Status Reports occur every 3 seconds. Use of the *MONITOR* option significantly increases the amount of unsolicited information to be received from the Serial Interface and processed.
MONALL *(option)*	Same as Connect (the Serial Interface can return C-Bus Point to Multi-Point (SAL) messages, at any time, because it has a connection to one or more C-Bus network applications.) In addition, the Serial Interface will return SAL commands destined for a remote network. This option needs to be used with extreme care.
PUN *(option)*	Causes a Power Up Notification (PUN) message to be emitted by the Serial Interface after it has completed its initialisation and before accepting any characters from its serial input port. The PUN option can be used by a device to initialise (or re-initialise) any special parameter settings in the Serial Interface in the event of power failure. Some devices have separate power for the C-Bus interface and the remainder of the design. This option helps the two portions of such designs to remain synchronised. When off, no PUN message is emitted. Set this option if it is possible for device connected to the Serial Interface to be separately powered up or down, and where it is important to restore options and modes after a power cycle.

Mode / Option	Effect (when ON)
PCN *(option)*	Causes a Parameter Change Notification (PCN) message to be emitted by the Serial Interface any time it detects that an attempt to update one or more of its internal parameters, from C-Bus (eg C-Bus installation software). When off, no PCN message is emitted. Set this option when the serial device relies on parameters of the Serial Interface, and where those parameters could be changed over C-Bus (by for example the C-Bus installation software). When a parameter is changed, the device can then be notified and re-load any internal setting records from the Serial Interface.

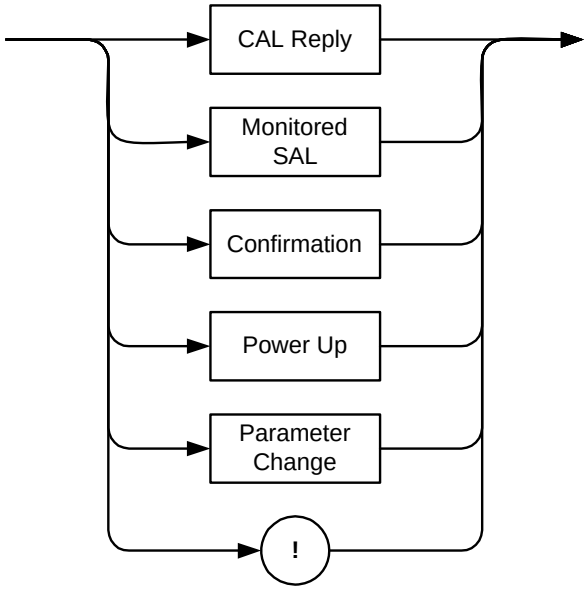
4.3.2 Structure of Replies

Replies from the Serial Interface comprise a series of (mostly) hexadecimal ASCII characters, used to convey information, and a small number of special characters used to provide status information about the operation of the Serial Interface.

Each reply (except Confirmation) is terminated with a carriage return, line feed pair (ASCII character 13, hex $0D; followed by ASCII character 10, hex $0A), and denoted <cr><lf> in the sections which follow.

4.3.3 Format of Replies Received from the Serial Interface

Replies from the Serial Interface return information depending on options selected and the previously transmitted command, as shown below:



Notes:

a. CAL Reply only appears in response to a preceding Point to Point (CAL) command, or a preceding Point to Multi-Point (SAL) Status Request command.

b. Monitored SAL can appear unsolicited at any time if the *CONNECT* option is enabled.

c. Confirmation appears whenever the preceding transmission to the Serial Interface requested a confirmation, as described in section 4.2.6.

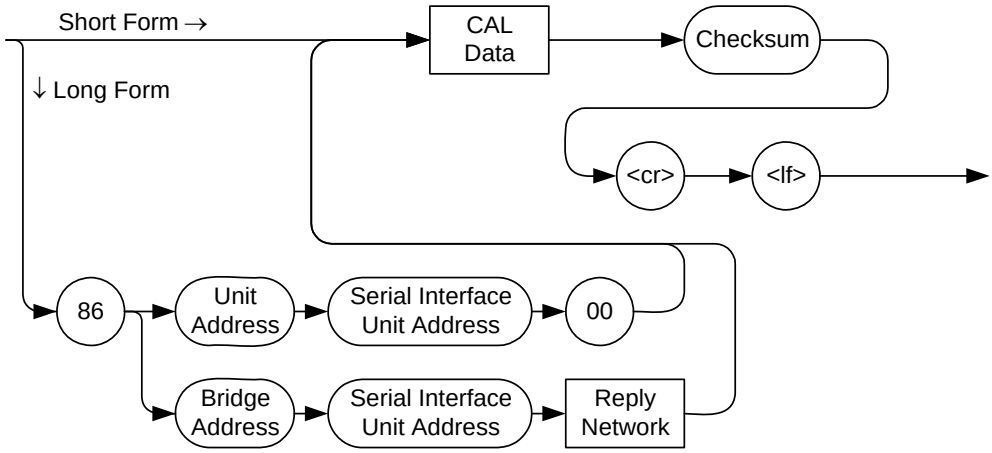
d. Power Up can appear unsolicited at any time after the Serial Interface has power applied, provided the *POWER UP NOTIFY (PUN)* option is set.

e. Parameter Change can appear unsolicited at any time if the Serial Interface detects an attempt to update one or more of its parameters over C-Bus and the *PARAMETER CHANGE NOTIFY (PCN)* option is set.

f. The “!” character reply is returned at any time if the PCI cannot accept the data supplied. For example, because of a checksum failure or because of the buffers were full and more information could not be accepted.

4.3.3.1 CAL Reply

The CAL Reply has the format:



Notes:

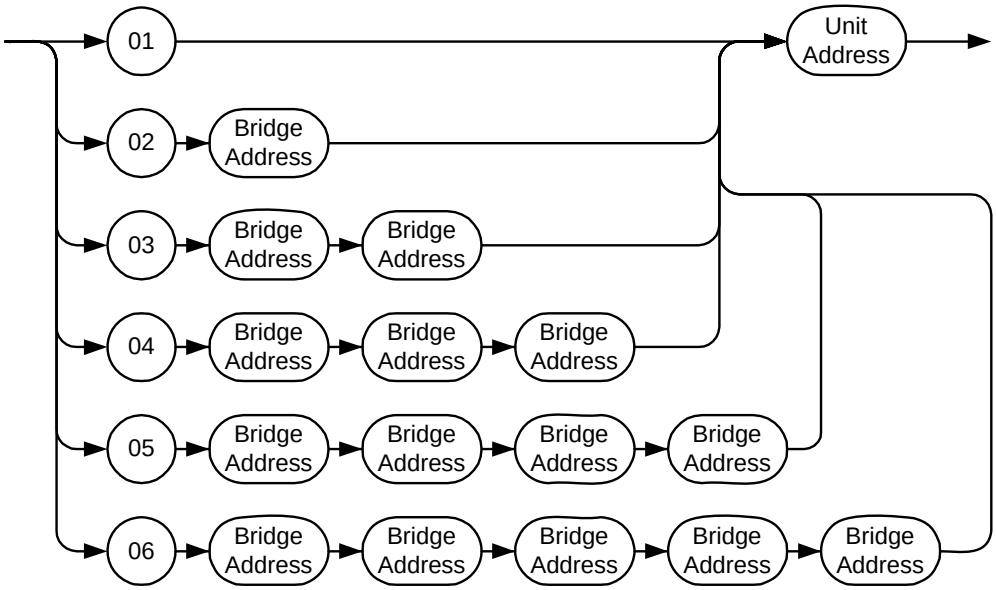
a. CAL Data defines the commands which can be received from a unit. These commands comprise a sequence of bytes, and are described in section 7.

b. The format of the information returned will be either a Short or Long Form, as shown, depending on the selected Modes and Options.

c. Unit Address, Bridge Address and Serial Interface Unit Address are all single bytes, and are described in section 3.3. Unit Address is the unit address of the unit which sent the CAL Reply.

d. The CAL Reply is terminated with a Checksum, calculated as described in section 3.6.

Reply Network has the format:



Notes:

a. Unit Address and Bridge Address are all single bytes, and are described in section 3.3.

Example of CAL Reply, assuming operation in *BASIC* Mode, and with no checksum sent to the Serial Interface (*SRCHK* option off). The request is a an IDENTIFY 2 sent to unit address $05.

To Serial Interface:

\0605002102

From Serial Interface:

8902312E322E363620200A

This CAL Reply has no addressing information, so only the CAL Data portion and checksum on the end apply. Section 7 shows that this is a REPLY command of 9 bytes, which codes, in ASCII, the firmware version of unit 5.

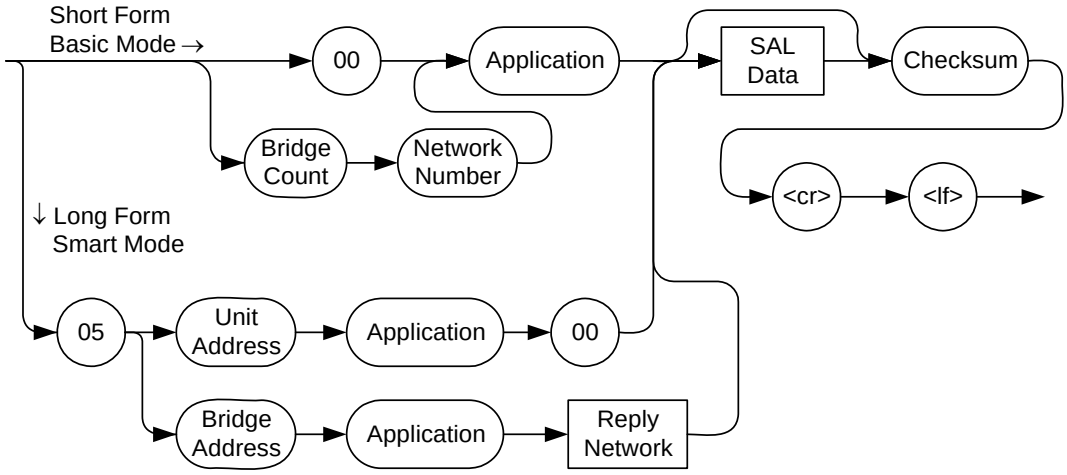
The same reply in *SMART* Mode might look like:

860593008902312E322E363620207F

This CAL Reply includes addressing information showing it came from unit address $05, that the Serial Interface unit address is $93, and the CAL Data portion is as above.

4.3.3.2 Monitored SAL

Monitored SAL has the format:



Notes:

a. The format of the information returned will be either a short form in *BASIC* Mode, or a long form in *SMART* Mode.

b. Bridge Count is a byte, in the range 1 to 7, indicating the number of number of bridges the message passed through to enter this network.

c. Network Number is the number of the originating network (address of the last bridge in the path to the originating network).

d. Unit Address, Bridge Address and Application are all single bytes, and are described in section 3.3.

e. SAL Data is determined by the Application, and is described for each Application in document set CBUS-APP. SAL Data is optional.

f. The Monitored SAL is terminated with a Checksum, calculated as described in section 3.6.

g. Reply Network is defined in section 4.3.3.1.

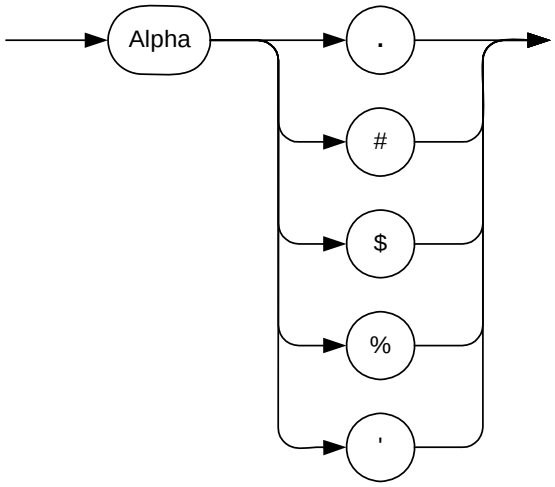
Example of Monitored SAL, assuming operation in *SMART* Mode, with *CONNECT* option and *SRCHK* option set:

`0503380079083F`

This is a message from unit address $03, on application $38. The message portion (SAL Data) is $7908, and the checksum is $3F. Section 6 shows that this is a lighting ON command on Group Address Variable $08.

4.3.3.3 Confirmation

Confirmation has the form:



Notes:

a. Alpha is the lower case alphabetic character transmitted to the Serial Interface, as described in section 4.2.6.

b. The other characters have the following meanings:

. means the command transmitted to the Serial Interface with the matching Alpha was successfully transmitted into the local C-Bus network.

means the command transmitted to the Serial Interface with the matching Alpha was not transmitted into the local C-Bus network, due to too many re-transmissions with no acknowledgment.

$ means the command transmitted to the Serial Interface with the matching Alpha was not transmitted into the local C-Bus network, due to corruption in transmission.

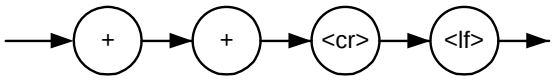
% means the command transmitted to the Serial Interface with the matching Alpha was not transmitted into the local C-Bus network, due to Serial Interface losing C-Bus synchronising clock at any time during an attempted C-Bus transmission.

‘ (single quote) means that the character sequence received by the Serial Interface was too long.

Important: Confirmation is not followed by a <cr><lf> pair.

4.3.3.4 Power Up

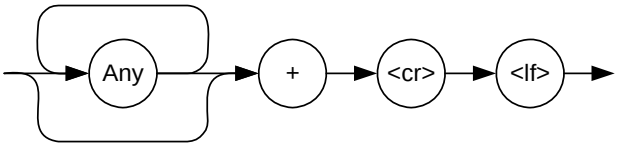
Power Up has the form:



Notes:

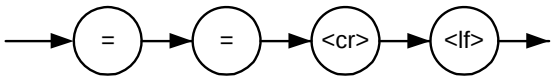
a. Power Up is only emitted by the Serial Interface if the *Power Up Notification (PUN)* option is set.

b. On power up, the first + character could be corrupted, missing, or preceded by garbage. It's presence should not be assumed. Therefore, the following diagram more accurately represents what should be searched for:



4.3.3.5 Parameter Change

Parameter Change has the form:



Notes:

a. Parameter change is only emitted by the Serial Interface if the *Parameter Change Notification (PCN)* option is set.

5 COMPOUND DEVICES

A “Compound Device” is one where:

a. a C-Bus Serial Interface as the means of attachment to C-Bus;

b. the Serial Interface is connected to a separate control processor; and

c. that control processor is powered from a separate power source (for example, a mains connection or a plug pack).

Important: Because compound devices can have the control processor power removed and applied separately from the C-Bus network power, special precautions need to be observed.

The following setup instructions are strongly recommended:

1. Always set the Power Up Notification (PUN) option during Serial Interface Initialisation

   Ensures that the Serial Interface reports when C-Bus power is cycled, so the control processor can re-initialise the Serial Interface, update models, etc.

2. Always check for the PUN coming from the Serial Interface, and if it is found, be sure to completely re-initialise all settings of the Serial Interface.

   The control processor can only find a power up of the Serial Interface if it looks for the PUN messages.

3. Always set the modes and options into parameter $30 (Interface Options 1) and parameter $41 (Interface Options 1 Power Up Settings).

   Setting Interface Options 1 Power Up Settings ensures that on power up, the Serial Interface has the same modes and settings that were active before C-Bus network power was removed. Ensure that the control processor uses the same protocol variant for communication with the Serial Interface.

4. When initialising the Serial Interface: always set *BASIC* mode using the “~” character before issuing other commands, always check that the “~” character at the very least is echoed, AND always set Interface Options 1 last.

   Setting *BASIC* mode ensures a known Serial Interface mode for the initialisation commands that follow.

   Checking for echo ensures that the Serial Interface is present. If there is no echo, it means the Serial Interface is not operating – probably because there is no C-Bus power.

   Setting Interface Options 1 mode last ensures that after all other settings are made in *BASIC* mode, the subsequent communication takes place using the mode set, so the control processor and the Serial Interface keep their operating modes synchronised.

6 APPLICATION (SAL) MESSAGES

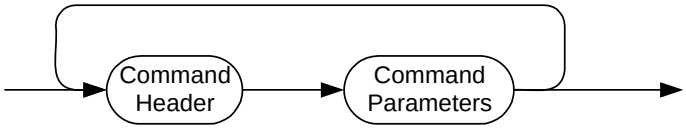
C-Bus Application Messages and behaviour are described in detail in the document set CBUS-APP.

This section provides a summary of common commands used for the C-Bus Lighting Application only. Read this in conjunction with CBUS-APP chapter 2.

6.1 Lighting Command Format

All C-Bus Lighting Commands are transported as Point to Multi-Point (SAL) commands, or as Point to Point to Multi-Point commands. In sections 4.2.9.2 and 4.2.9.3, the Lighting Commands are transported as the SAL Data portion of a C-Bus message transmission.

Lighting SAL commands can be concatenated:



6.2 Standard Lighting Application Address

The standard C-Bus Lighting Application Address is $38.

6.3 Lighting Application Commands

The following Lighting Application commands are supported by C-Bus units shipped by Clipsal:

Name	Command Header	Command Parameters	Description / Meaning
OFF	$01	Group	Sets the specified Group Address Variable to 0. 'Group' is the Group Address Variable number to be set.
ON	$79	Group	Sets the specified Group Address Variable to 255. 'Group' is the Group Address Variable number to be set.

Name	Command Header	Command Parameters	Description / Meaning
RAMP to LEVEL	\$02 or \$0A or \$12 or \$1A or \$22 or \$2A or \$32 or \$3A or \$42 or \$4A or \$52 or \$5A or \$62 or \$6A or \$72 or \$7A	Group, Level	Ramp the specified Group Address Variable from its present level to the level specified, at the specified rate. The Command header determines the ramp rate, as follows: \$02: instantaneous, \$0A: 4 second, max to min, or min to max \$12: 8 second, max to min, or min to max \$1A: 12 second, max to min, or min to max \$22: 20 second, max to min, or min to max \$2A: 30 second, max to min, or min to max \$32: 40 second, max to min, or min to max \$3A: 1 minute, max to min, or min to max \$42: 1.5 minute, max to min, or min to max \$4A: 2 minute, max to min, or min to max \$52: 3 minute, max to min, or min to max \$5A: 5 minute, max to min, or min to max \$62: 7 minute, max to min, or min to max \$6A: 10 minute, max to min, or min to max \$72: 15 minute, max to min, or min to max \$7A: 17 minute, max to min, or min to max 'Group' is the Group Address Variable number to be ramped. 'Level' is the target level at the completion of the ramp.
TERMINATE RAMP	\$09	Group	Terminates a ramp currently in progress on the specified Group Address Variable. 'Group' is the Group Address Variable number to terminate the ramp against.

6.4 Examples

Lighting Messages are sent as Point to Multi-Point commands, using the form described in section 4.2.9.2. The lighting command part is the SAL Data part in the diagram.

Examples of lighting commands:

\0538000108BA

This Point to Multi-Point command is sent to Application Address \$38, the standard C-Bus Lighting Application. The data sent to the Application is \$0108, and there is a checksum of \$BA. The data (\$0108) is an Off command to Lighting Group \$08.

\05380001087909090A25

This Point to Multi-Point command is sent to Application Address $38, the standard C-Bus Lighting Application. The data sent to the Application is $01087909090A, and there is a checksum of $25.

In this case, several lighting commands are being sent at the same time. These are to turn off lighting group $08, to turn on lighting group $09, and to terminate a ramp running on lighting group $0A.

6.5 Practical Limitations

Although multiple commands can be transmitted to a single application, as shown in the example above, there is an upper limit to the command that can be sent. This limits the SAL Data part of the stream to the Serial Interface to between 14 and 21 bytes (28 and 42 characters) depending on the amount of network routing present.

Clipsal recommends limiting the SAL Data portion to 14 bytes. This allows turning up to 7 groups on or off in a single command, or sending 4 ramps in a single command, or any other combination that uses up to or including 14 bytes.

7 DEVICE AND NETWORK MANAGEMENT (CAL) MESSAGES

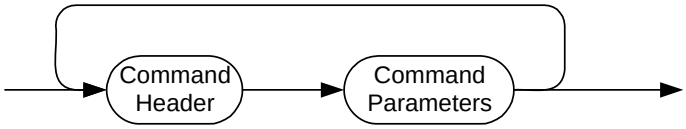
This section contains a subset of the C-Bus Device and Network Management (CAL) CAL commands.

There may be a small number of specialised cases where these commands are needed, but generally these commands should never need to be used. Many CAL commands are unit-specific and are subject to change without notice.

7.1 CAL Commands

All CAL messages follow exactly the same format, so the only distinction between a CAL Request and a CAL Reply is slightly artificial. Both CAL Requests and CAL Replies are subsets of CAL commands.

Some CAL commands can be concatenated[6]:



Important: Do not use concatenated CAL commands without clear proof that behaviour is as expected. In general, concatenate CAL commands should be avoided.

The following CAL commands are supported by C-Bus units:

Name	Command Header	Command Parameters	Description / Meaning
CAL Requests			
RESET	$08	- none -	Reset any errors latched in a unit. (Latched errors can be retrieved from a unit using a GETSTATUS command with parameter $00.)

[6] Some commands will cause truncation of CAL processing. In general, Identify and Reset commands can be concatenated in any combination on the same line, and can precede any Recall or GetStatus command. Keep in mind that the reply message length is limited, so as soon as the reply buffer is filled, processing will simply be truncated.

Name	Command Header	Command Parameters	Description / Meaning
RECALL	$1A	Param No, Count	Recall information from the non volatile (parameter) memory of a unit. 'Parameter No' is the Parameter number at which the recall should start. 'Count' is the number of bytes to return. Generally, between 1 and 16 bytes can be returned. The limit of 16 bytes decreases by 1 byte for each C-Bus bridge between the source and destination. The information is returned as CAL REPLY command.
IDENTIFY	$21	Attribute	Extract information from a unit. 'Attribute' determines the type of information extracted. Attributes and replies are described in section 7.2. The information is returned as CAL REPLY command.
GETSTATUS	$2A	Param No, Count	Recall parameters from the volatile (RAM) memory of a unit. This facility is primarily a diagnostic tool for the use of Clipsal. The only special use is GETSTATUS 0, 1; which returns the unit error status. 'Parameter No' is the location at which the extraction should start. 'Count' is the number of bytes to return. Generally, between 1 and 16 bytes can be returned. The limit of 16 bytes decreases by 1 byte for each C-Bus bridge between the source and destination. The information is returned as CAL REPLY command.

Name	Command Header	Command Parameters	Description / Meaning
CAL Replies			
REPLY	$80 + bytes following	Param No, Byte1, Byte2, …, ByteN	Returns information from a previous RECALL, IDENTIFY or GETSTATUS. For RECALL and GETSTATUS, the returned starts at 'Param No, and 'Byte1', 'Byte2', etc are the bytes that have been retrieved. For IDENTIFY, the Attribute requested is returned as 'Param No, and the byte stream is the information returned from the IDENTIFY request. At most 31 bytes can be returned, though in practice no more than 16 bytes are returned, and this can drop to 9 bytes if the message passes through the maximum number of C-Bus bridges. The count portion of the Command Header reflects the number of bytes attempted for return (1 to 31). Less bytes may be returned if the number requested exceeds the 9 to 16 byte limit described above.
ACKNOWLEDGE	$32	Param No, Code	An acknowledge.
STATUS (Standard)	$C0 + bytes following	Application, Block Start, Byte1, Byte2, …, ByteN	Information returned by previous Status Request. 'Application' is the application number on which status was requested. 'Block Start' is the starting block number for the bytes that follow 'Byte1' .. 'ByteN' are the status data bytes, described in section 7.3.

Name	Command Header	Command Parameters	Description / Meaning
STATUS (Extended)	$E0 + bytes following	Coding, Application, Block Start, Byte1, Byte2, …, ByteN	Information returned by previous Status Request, in Extended format. Extended format is used for a reporting a Level Status Request at all times, and for a Switched Status Request only when the *Extended Status Format (EXSTAT)* option is enabled. 'Coding' is a special code indicating the type of status information being returned and the nature of the device which initiated the Status Request, as described in section 7.4. 'Application' is the application number on which status was requested. 'Block Start' is the starting block number for the bytes that follow 'Byte1' .. 'ByteN' are the status data bytes, described in sections 7.3 and 7.4.

7.2 The IDENTIFY Command

The IDENTIFY command is used to extract information from C-Bus units. The command returns the information in a REPLY command, with the format:

`zzyyaabbccddeeffgghhiijjkkllmmnnooppqquurrsstt`

Where:

zz = header

yy = the identify attribute number

the last byte is a checksum

The maximum length of data in the reply from any identify command is 18 ($12) bytes.

The following is a subset of IDENTIFY replies that are supported by all units, that provide useful information about a unit.

Attribute	Meaning	Bytes Returned*	Meaning
$00	manufacturer	$08	aa-hh = ASCII text, manufacturer name
$01	type	$08	aa-hh = ASCII text, unit type
$02	firmware version	$08	aa-hh = ASCII text, firmware version, as 8 ASCII bytes Example: 1.00 = 2020312E30302020

Attribute	Meaning	Bytes Returned*	Meaning
$03	summary	$09	aa-ff = ASCII text, part name, 6 bytes gg = hex code, unit service type, 1 byte hh-ii = hex, version, 2 hex bytes for four characters (example: 1.2.43 = 1243)
$04	Extended-diagnostic summary	$0C	aa-bb = Applications supported by this unit cc = Area (if supported, else 00) dd-ee = 16 bit CRC of patch area, $70 to $F2 inclusive ff-ii = 4 hex bytes representing 32 bit serial number (deprecated) jj = raw value from network voltage A/D, convert to decimal then divide by 6.375 for volts kk = COMMON STATUS: bit 0 is 1 for output unit bit 1 is 1 for enable checksum alarm bit 2 is reserved bit 3 is reserved bit 4 is reserved bit 5 is 1 if network voltage is marginal (network LED flashing) bit 6 is 1 if network voltage is low (network LED off) bit 7 is 1 if unit is in LEARN mode ll = ERROR flags byte from unit: bit 0 = micro power on reset bit 1 = internal stack overflow bit 2 = comms tx error bit 3 = micro reset bit 4 = EE data error bit 5 = EE checksum error bit 6 = EE write error bit 7 = installation MMI error
$05	network terminal levels $0C	$0C (dynamic)	aa-ll = values, values of terminals after logic, terminal 1 first Applies to output units only
$06	terminal levels	$0C (dynamic)	aa-ll = values, values of terminals after logic and all local inputs, terminal 1 first Applies to output units only

Attribute	Meaning	Bytes Returned*	Meaning
$07	network voltage	$05	aa-bb = 2 ASCII bytes to represent whole volts cc = $2E, always the same, ASCII of decimal place dd = 1 ASCII byte to represent first decimal place of volts ee = $56, always the same, ASCII 'V'
$08	GAV values current	$10	aa-pp = values, values of group address variables as on network
$09	GAV values stored	$10	aa-pp = values, values of group address variables as stored in EEPROM Applies to output units only
$0A	GAV physical addresses	$10	aa-pp = values, group address variable physical addresses on network
$0B	logic assignment	$0C (dynamic)	aa-ll = values, bit assignment of logic allocation per terminal aa = terminal 1: bit 0 = assigned to GAV 13 bit 1 = assigned to GAV 14 bit 2 = assigned to GAV 15 bit 3 = assigned to GAV 16 bit 4 = reserved bit 5 = reserved bit 6 = set for re-strike delay on this terminal bit 7 = set for 'greater of, OR' logic Applies to output units only
$0C	delays	$0D (dynamic)	aa-ll = values, power-up delays for each terminal mm = value for re-strike delay - relay units only Applies to output units only
$0D	minimum levels	$0C (dynamic)	aa-ll = values, minimum terminal levels or relay turn-on thresholds Applies to output units only
$0E	maximum levels	$0C (dynamic)	aa-ll = values, maximum terminal levels (unused bytes returned as $FF) Applies to output units only
$0F	current sense levels	$08 (dynamic)	aa-hh = values, raw values from current sense A/D's, $FF represents full range Applies to output units only

Attribute	Meaning	Bytes Returned*	Meaning
$10	output unit summary	$04 (dynamic)	aa = flags byte from unit #1: bit 0 = 1 if unit is generating clock bit 1 = 1 for clock generation enable bit 2 = 1 if unit in a local toggle active state bit 3 = 1 if local toggle is enabled bit 4 = 1 if remote ON input is asserted bit 5 = 1 if remote OFF input is asserted bit 6 = 1 if restrike timing is active bit 7 = 1 if unit is asserting network burden bb-cc = GAV STORE enable bytes dd = hex, time from last recovery of mains (seconds)
$11	DSI status	$0A	aa = Channel 1 status: 00 = OK 02 = lamp fault 03 = current limit or short bb = Channel 2 status, as for channel 1 cc = Channel 3 status, as for channel 1 dd = Channel 4 status, as for channel 1 ee = Channel 5 status, as for channel 1 ff = Channel 6 status, as for channel 1 gg = Channel 7 status, as for channel 1 hh = Channel 8 status, as for channel 1 ii = Unit status: 00 = OK 01 = C-Bus to dimming uC comm failure (NACK) 02 = C-Bus to dimming uC comm failure (no response, or PIC side power failure) jj = Dimming uC revision number Applies to DSI output units only.
* Where the bytes returned is shown as (dynamic), the reply length is set by the number of associated variables in that unit. The description shows the maximum possible reply lentgh.			

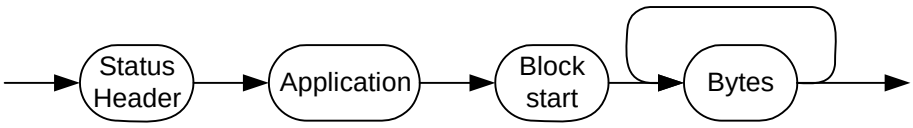
7.3 Standard Format Status Replies

Important: Standard Status Replies are returned only in response to a Switched Status Request when the Extended Status Format (EXSTAT) option is turned off. Refer to section 7.4.

The reply format is short form (no addressing information), irrespective of operation in BASIC Mode or SMART Mode.

A Standard Format Status Reply is too large to be returned in a single line by the Serial Interface. Standard Format Status Replies are returned in at least 3 separate lines.

Each line is an item of CAL Data, and has the form:



Each line of the reply:

a. Starts with the Status Header, which is “C0” + the number of character pairs to follow;

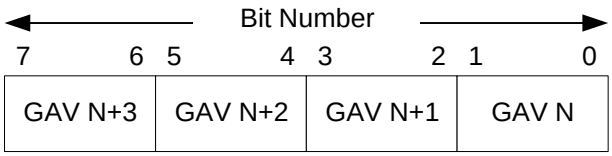
b. Includes the application being reported, following the header;

c. Includes the block start number for this line, which correspond to the starting Lighting Application Group Address Variable for this line, following the application; and

d. Includes the status bytes, following the starting block number.

As part of a CAL Reply, each line is also terminated by a checksum and <cr><lf> pair.

The status bytes use a coding of 2 bits per Group Address Variable, so each byte encodes the status of 4 consecutive Group Address Variable numbers, in order from least to the most significant bit pairs, as shown below:



The bit coding is:

00 = This Group Address does not exist in the network

01 = This Group Address is ON

10 = This Group Address is OFF

11 = This Group Address is in the ERROR state

For example, if a report started at group 00, then:

- a byte “01” would indicate Group 00 is ON, and groups 1, 2 3 do not exist;
- a byte “80” would mean group 3 is OFF, and groups 0, 1, 2 do no exist,
- a byte “0C” would mean:
  - group 1 is represented in more than one output unit and has conflicting states in these units, or one or more output units are requesting initialisation of that group address variable; and
  - groups 0,2 and 3 do not exist.

An example of a Standard Format Status Reply, over 3 lines, is:

D8380068AA01405505500010000000140000000000000000CF

D83858000000000000000000000000000000000000000000098

D638B000000000FF000000000000000000000000000000043

This reply codes the binary status of all 256 possible groups in application 38, the standard C-Bus Lighting Application.

Using this example reply:

The first line above “D8380068AA01…” codes up a reply with 48 characters to follow, on application 38. The first line starts reporting group address 00. The status bytes “68AA01…” are interpreted as:

68: group 0 does not exist, group 1 = OFF, group 2 = OFF, group 3 = ON

AA: group 4 = OFF, group 5 = OFF, group 6 = OFF, group 7 = OFF

01: group 8 = ON, and groups 9, 10, 11 do not exist

The second line of output for the status report has exactly the same format, but starts the report group “58” (hex).

The third line is a slightly shorter, but is otherwise the same, and starts reporting from group “B0” (hex).

Status replies from a remote C-Bus network are generally slightly shorter and run over more lines. Status replies from remote networks can also have other replies interleaved between the status reply lines.

7.4 Extended Format Status Replies

Important: Extended Format Status Replies are returned in response to either:

a. A Level Status Request; or

b. A Switched Status Request when the Extended Status Format (EXSTAT) option is turned on.

When the EXSTAT option is turned off, a Level Status Request will cause a short form reply with no addressing information, as described in section 4.3.3.1, and with the information presented in the Extended Status Format.

When the EXSTAT option is turned on, all Status Requests will cause a long form reply with addressing information included, as described in section 4.3.3.1, and with the information presented in the Extended Status Format.

	Switched Request	**Level Request**
***EXSTAT* Off**	Short Form No addressing Standard Format	Short Form No addressing Extended Format
***EXSTAT* On**	Long Form Addressing Extended Format	Long Form Addressing Extended Format

An Extended Format Status Reply is presented in a similar format to a Standard Status Reply. The changes are a different prefix and an additional byte of information indicating further information about the Status Request and the type of information in the Status Reply:

Each line of the Extended Format Status Reply:

a. Starts with the Status Header, which is “E0” + the number of character pairs to follow;

b. Includes a coding byte (see below) following the header;

c. Includes the application being reported, following the coding byte;

d. Includes the block start number for this line, which correspond to the starting Lighting Application Group Address Variable for this line, following the application; and

e. Includes the status bytes, following the starting block number.

The Coding byte[7], has the following meanings:

Value	Status Reply Type	Initiated Where?
$00	Binary	By this Serial Interface
$40	Binary	Elsewhere
$07	Level	By this Serial Interface
$47	Level	Elsewhere
Others	Reserved	N/A

Binary Status Reports code the status information as described for the Standard Status Reply described in section 7.3.

A Level Status Reply is only supported by the C-Bus Lighting Application. It uses a different encoding of the level information. It can report the dimming level of 32 C-Bus Lighting Application Group Address Variables at a time. To find the Level status of the entire network, 8 separate Level Status Requests are needed.

The Level information for a single Group Address Variable is reported in 2 bytes (4 characters). The bytes are split into 4 bit nibbles, each of which can only be either $5, $6, $9 or $A; these code the bit pairs 11, 10, 01 and 00 respectively.

Thus: (nibble pair -> binary -> hex):

$55 = 1111 = $F
$56 = 1110 = $E
$59 = 1101 = $D
$5A = 1100 = $C
$65 = 1011 = $B
$66 = 1010 = $A
$69 = 1001 = $9
$6A = 1000 = $8
$95 = 0111 = $7
$96 = 0110 = $6
$99 = 0101 = $5
$9A = 0100 = $4
$A5 = 0011 = $3
$A6 = 0010 = $2
$A9 = 0001 = $1
$AA = 0000 = $0

The 2 nibble pairs associated with each Group Address Variable are decoded in reverse order to reveal the current level eg. 9599 represent a level of $57.

In the Level Status Reply, there are therefore 4 nibbles per Group Address Variable, and therefore the level (0 to 255) for each can be extracted. The absence of a Group Address Variable is indicated by the nibbles having a value of $0000. If nibbles of value

[7] Use of this coding byte allows a device using the *MONITOR* option (and which therefore sees all Status Replies) to determine if a given reply was initiated by itself or by some other unit. This distinction is required only for determination of 'MMI Master' for some Applications.

0, 1, 2, 3, 4, 8 or $C are received, then the level information for that Group Address Variable has been corrupted by the presence of noise during transmission of the status report. If nibbles of value 7, $B, $D, $E or $F are received, then the level information may have been corrupted by the presence of noise, or the levels reported by two or more output devices associated with the same group address are not the same.

For example, a Level Status Reply (in short form, with *EXSTAT* turned off) might be:

F9073800AAAA00009599000000005555000000000005555555548

F907380B0000000000005555000000000000000000000000013

F707381600000000000000000000000000000000000000B4

This codes the Lighting Application level status of groups 0 to 31 inclusive in application 38, the standard C-Bus Lighting Application:

The first line above “F9073800AAAA00009599…” indicates a reply with 48 characters to follow, coding 07 (Level status initiated by this Serial Interface), on application 38. The first line starts reporting group address 00. The status bytes “AAAA00009599…” are interpreted as:

AAAA: group 0 exists and is at level 0 (ie OFF)

0000: group 1 does not exist

9599: group 2 exists and is at level $57

0000: group 3 does not exist

0000: group 4 does not exist

5555: group 5 exists and is at level $FF (ie ON)

and so on

The second line of output has exactly the same format, but starts reporting from Group Address Variable $0B (11).

The third line is slightly shorter, and starts reporting from parameter number $16 (22).

As for Standard Status Replies, those from a remote C-Bus network are generally slightly shorter, and run over more lines. Extended Status Replies from remote networks can also have other replies interleaved between the status reply lines.

8 C-BUS BRIDGES AND NETWORK ROUTING

A C-Bus bridge can be used to extend the range of a C-Bus network by adding additional segments.

A C-Bus Point to Point (CAL) command, or a Point to Point to Multipoint (SAL) command can be routed between up to 7 interlinked networks, by passing through up to 6 intervening bridges.

8.1 Network Routing

The method of communicating over bridges is known as Source Routing. This means that the unit transmitting a command has to know the full path to the destination network.

When transmitting a command through one or more bridges:

a. The command is transmitted to the Unit Address of the first bridge; and

b. Additional routing information needs to be supplied.

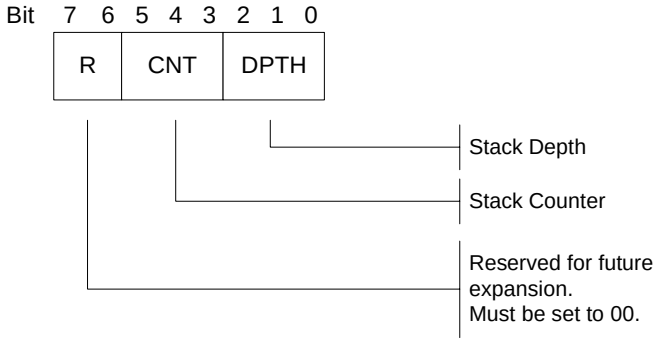
This is shown in the definition of the Point to Point command (section 4.2.9.1), and Point to Point to Multipoint command (section 4.2.9.3).

8.2 Network Protocol Control Information

The routing information consists of Network Protocol Control Information (Network PCI), which comprises a header byte followed by a list of bridge unit addresses, and finishes with the destination address.

The final destination address is either unit address for a Point to Point (CAL) command, or an Application Address for a Point to Point to Multipoint (SAL) command.

The header byte encodes a Stack Depth and a Stack Counter:



When the header byte is $00, this means there is no routing information.

The fields in the header have the following meanings:

Stack Depth: The number of bridges that must be used to transmit the information from the source to the destination.

Stack Counter: The number of bridges that must be used to complete the transmission from the current bridge to the destination.

When creating a new message for transmission to a remote network, the Stack Depth and Stack Counter are set the same.

This header byte shows the origin of the magic numbers for routing in sections 4.2.9.1 and 4.2.9.3.

When transmitting a message through 1 bridge, the header byte will be %00001001 = $09. When transmitting through 2 bridges, the header byte will be %00010010 = $12, and so on.

8.3 Reverse Route

As a message passes through bridge units, the routing information is rearranged, and the Stack Counter is decremented. By the time the message arrives at the destination, the routing information has been reconstructed by the bridges to form the reverse route.

This means a unit receiving a message from a remote unit can extract the addressing information to obtain the path back to the unit which originated the message.

This behaviour is shown in the example below, which assumes a Point to Point (CAL) message is being transmitted from 'src' to 'dest', through three bridges, 'br1', br2' and 'br3'. Each bridge has two unit addresses, 'br1a', 'br1b', 'br2a', 'br2b', 'br3a', 'br3b'.

The address rearrangement is:

	C-Bus Address		Network PCI Header		Network PCI routing bytes		
step	**Source address**	**Dest address**	**Stack Count**	**Stack Depth**	**A1**	**A2**	**A3**
start	src	br1a	3	3	br2a	br3a	dest
Exit from first bridge	br1b	br2a	2	3	br3a	dest	src
Exit from second bridge	br2b	br3a	1	3	dest	br1b	src
Exit from third bridge	br3b	dest	0	3	br2b	br1b	src

An example is shown in section 9.3.

8.4 Bridge Behaviour for Point to Point to Multipoint Commands

The last bridge in a routing chain transforms a Point to Point to Multipoint command into a Point to Multipoint command before relaying it.

It is therefore possible for a received Point to Multipoint command to have a valid network routing stack, which can be used to extract the reverse route.

However, transmitting a Point to Multipoint command with a routing stack is not valid, and will be ignore by bridges and all other units.

9 EXAMPLES AND COMMON PROCEDURES

These examples are used to show common operations, using the C-Bus Lighting Application.

9.1 Point to Multi-Point commands into a Local C-Bus Network

To issue a lighting application ($38) OFF command to a load on Group Address $14, in *BASIC* Mode, with the *SRCHK* option set:

\0538000114AE

To issue the same command in SMART Mode, with the SRCHK option set, and request a confirmation of the delivery status:

\0538000114AEr

This will reply with either:

r.	if the transmission onto C-Bus was OK
r#	if the transmission onto C-Bus failed with too many attempts and no acknowledge
r$	if the transmission onto C-Bus failed due to corruption during transmission
r%	if the transmission onto C-Bus failed due to a lost C-Bus clock

To issue a lighting application ($38) Status Request, in *SMART* Mode, with the *EXSTAT* option off, and *SRCHK* on:

\05FF007A38004A

This will reply with something like:

D83800A8AA020000000000000000000000000000000000009C

D838580000000000000000000000000000000000000000000098

D638B00000000000000000000000000000000000000000000042

To do the same, but with the *EXSTAT* option on, the reply will be something like:

86999900F8003800A8AA020000000000000000000000000000000000000000C4

86999900F800385800000000000000000000000000000000000000000000C0

86999900F60038B000000000000000000000000000000000000000008F

9.2 Point to Point commands into a Local C-Bus Network

Assume *SMART* Mode, with *SRCHK* set. To recall the current value of parameter $30, on unit $04:

\0604001A3001AB

This will reply with:

8604990082300328

indicating a long form CAL Reply, from unit $04, through Serial Interface unit $99. The value $82 indicates 2 bytes of data, the first ($30) is the parameter number, the second ($03) is the value of the parameter, and the last ($28) is the checksum.

9.3 Point to Multi-Point Commands Into a Remote C-Bus Network

Assume *SMART* Mode, with the *SRCHK* option set.

From unit address $21, Issue a lighting application ($38) OFF command to a load on Group Address $14.

The load is located in a remote network through three bridges. The bridges have (nearside, farside) addresses ($42, $43), ($53,$54) and ($64,$65) respectively.

\03421B53643801149C

This will be received in the destination network as:

0565380354432101148E

This is a Point to Multipoint message ($05), received from the far side of the last bridge ($65), and addressed to the C-Bus lighting application ($38). It has 3 network addressing bytes ($03), which are the far sides of the bridges, and the originating unit address respectively ($54, $43, $21). The message data is a lighting application OFF command on group $14 ($0114). The final byte is a checksum ($8E).

9.4 Set SMART Mode, SRCHK, and CONNECT Options

These options are set by sending the ~ to force *BASIC* Mode and turn off all options, then using the special codes to set the mode and options. There is no checksum (because the ~ has turned *SRCHK* off):

~@A3300019

9.5 Issue Multiple Lighting Commands In A Single Message

This example operates in the C-Bus Lighting application ($38), on the local network. It will turn groups $21, $22, $23 and $24 ON, and ramp groups $25 and $26 to level $01 using the 4 second ramp rate. It assumes *SMART* Mode, and that the *SRCHK* option is set:

\05380001210122012301240A25010A2601D4r

This message will return a status of the transmission attempt.

10 SERIAL INTERFACE SETUP AND OPTIONS

The Serial Interface began as a very simple and limited interface to C-Bus. Its capabilities have been gradually enhanced over a long period, generally in response to specific requests for improvement.

In many cases, the enhancements have not been compatible with previous practice. Backward compatibility has been ensured by adding options to the Serial Interface to change the format of information returned. Consequently, not all combinations of options are useful.

The Serial Interface options are set by changing parameters, as shown below in section 10.2.

10.1 Recommended Option Combinations

The common Mode and option combinations involve *SMART* Mode, and the *CONNECT, IDMON, EXSTAT* and *MONITOR* options.

Recommendations for combinations of options:

CONNECT:	***Should only be used in conjunction with SMART mode, to ensure that echo is switched off. Use of CONNECT in BASIC mode could cause echoed characters to be interleaved with received SAL messages.***
MONITOR:	***As for CONNECT, should only be used with SMART mode.***
IDMON:	***Should always by turned on when in SMART mode to ensure all Point to Point commands return the same format replies under all circumstances.***
EXSTAT:	***If MONITOR is OFF, the use EXSTAT is optional.***
	If MONITOR is ON, EXSTAT is recommended because it allows easy separation of replies to Status and Level Requests, and of monitored replies initiated by other units in the network.

10.2 Setting Parameters

Parameters are updated by sending a special sequence of bytes to the Serial Interface. This sequence will generate an acknowledgment, the format of which depends on the Mode and options.

10.2.1 Preferred Method[8]

Parameters can be set by sending a special sequence to the Serial Interface, irrespective of any special settings or Modes:

Either (when *SRCHK* is off):

[8] Earlier versions of the Serial Interface do not support this preferred method and the "@" syntax.

@A3pp00vv<cr>

or (when *SRCHK* is on):

@A3pp00vvcc<cr>

Where:

pp = parameter number

vv = the value to which the parameter should be set

cc = a single byte checksum

Important: When the SRCHK option in parameter $30 is off and a command is used to turn it on, use the command form without the checksum, because the checksum test is not active until after the option has been set.

The Serial Interface will respond with an acknowledge, which will be in either a short or a long form depending on the setting of the *IDMON* option:

When *IDMON* is off:

32pp00cc<cr><lf>

or (when *IDMON* is on):

86uuuu0032pp00cc<cr><lf>

Where:

pp = parameter number specified

cc = single byte checksum

uu = the C-Bus unit address number of the Serial Interface

10.2.2 Obsolete Method

This method applies to all versions of the Serial Interface. Its use is now obsolete, but will be supported for the foreseeable future. The preferred method for setting parameters is described in section 10.2.1.

Parameters in the Serial Interface can be set by sending a special sequence to the Serial Interface when in *BASIC* Mode (with all other options switched off – ie issue ~ first):

A3pp00vv<cr>

Where:

pp = parameter number

vv = the value to which the parameter should be set

The Serial Interface will respond with an acknowledge:

32pp00cc<cr><lf>

Where:

pp = parameter number specified

cc = single byte checksum

Important: This method for setting parameters only applies when the Serial Interface is in BASIC Mode.

If the Serial Interface is operated in SMART Mode or with any other options active, a change back to BASIC Mode is required before parameters can be changed using this method.

10.3 Parameter Set

The parameters required for operation of the Serial Interface are divided into two groups: management parameters, and operational parameters.

Management parameters are required by all units connected to the C-Bus network, and include such thing as Unit Address, and Application Address.

Operational Parameters are used to set the operating modes or characteristics of a unit.

The serial interface parameters relevant to user setup of the Serial Interface are:

Param No(s)	Group	Parameter Description	Form	Volatile*	Protection Level
21	Mgmt	Application Address 1	Byte (Note 1)	No	Unlock required from C-Bus port
22	Mgmt	Application Address 2	Byte (Note 1)	No	Unlock required from C-Bus port
30	Unit	Interface options 1	8 Bits (Note 2)	Yes	No write access via C-Bus port
3D	Unit	Baud rate selector	Byte (Note 3)	No	No write access via C-Bus port
3E	Unit	Interface options 2	Byte (Note 4)	No	None
41	Unit	Interface options 1 power up settings	8 Bites (Note 5)	No	Unlock required from C-Bus port
42	Unit	Interface options 3	Byte (Note 6)	No	Unlock required from C-Bus port
EB - F2	Mgmt	Custom Manufacturer (8 bytes)	ASCII Chars (Note 7)	No	Read only
F3 – F6	Mgmt	Serial number	Bytes (Note 8)	No	Read only

Param No(s)	Group	Parameter Description	Form	Volatile*	Protection Level
F7 - FE	Mgmt	Custom Type (8 bytes)	ASCII Chars (Note 9)	No	Read only

***Volatile:**

When power is lost, the value of volatile parameters is not preserved. On power up, the volatile parameters are loaded with defaults, or have a value loaded from another non-volatile parameter as described in the notes.

10.3.1 Application Address (Note 1)

Application Addresses 1 and 2 can be used to define either one or two Applications to be monitored, in the range $00 through to $FE. The *MONITOR* option needs to also be set for monitoring to be active.

Setting Application 1 to $FF acts as a wildcard - the PC Interface will respond to activity on ALL Applications. In that case, Application 2 should be set to $FF, which indicates unused.

The default value is $FF when shipped from factory.

10.3.2 Interface Options 1 (Note 2)

Interface Options 1 is used to select various Serial Interface communication options. On power up[9], it is loaded from Interface Options 1 Power Up Settings (refer section 10.3.5).

This parameter can only be modified from the serial port.

This parameter is made up of a series of bits with following meanings:

Bit	Name	Description
0	CONNECT	When set, the Serial Interface makes a logical connection between C-Bus and the RS-232 port, for the Applications configured in parameters $21 and $22. (Refer to Note 2). This connection allows the device attached to the RS-232 port to receive all C-Bus Point to Multi-Point (SAL) messages addressed to those Applications. When the *CONNECT* option only is set, Status Reports are not relayed to the RS-232 port, but this can be enabled by setting the *MONITOR* option.
1		Reserved.

[9] When power is applied, earlier versions of the Serial Interface loaded this parameter with $00 (ie *BASIC* Mode, no other options set).

Bit	Name	Description
2	XONXOFF	When set, switches on the use of XON / XOFF Handshaking for serial communications. If set, the Serial Interface will issue an XON character once each time its serial buffer is emptied. The XOFF character will be sent once when its buffer (20 bytes) reaches the 14th byte.
3	SRCHK	When set, forces the Serial Interface to expect a checksum on all serial communications it receives. The checksum applies only to HEX characters (pairs of ASCII characters representing HEX values 00..FF), and is to be appended to the end of each message, before the carriage return character. The checksum is calculated as described in section 3.6. If the Serial Interface detects a checksum error, the command will be ignored and an exclamation point (!) character will be returned.
4	SMART	When set, the Serial Interface will not echo serial data it receives, and will include all path information and source address in monitored SAL messages and some CAL replies. Full addressing information is not included for CAL replies from itself and for Status Reports. (Use *IDMON* and *EXSTAT* to switch on addressing information for these types of CAL reply.)
5	MONITOR	When set, the Serial Interface will relay all Status Reports for Applications matching its parameter numbers $21 and $22. The form of the Status Report will depend on the setting of the *EXSTAT* option. This enhances the monitoring of local networks. As for *CONNECT*, setting parameter $21 to $FF acts as a wildcard, passing Status Reports through for all applications on the local network.
6	IDMON	When set, all messages returned from the Serial Interface in response to a command sent to the Serial Interface are given a format consistent with the SMART Mode for all other messages.
7		Reserved – must be 0 in normal use.

10.3.3 Baud Rate Selector (Note 3)

Selects the data rate for the serial connection.

If the data rate of the Serial Interface is unknown, it can be set to 9600 baud by sending the two ASCII characters ' \0' ($5C $30) at 9600 baud within 5 seconds of application of power to the Serial Interface.

Legal values are:

$01 = 4800 baud;
$02 = 2400 baud;
$03 = 1200 baud;
$04 = 600 baud;
$05 = 300 baud;
Any other value = 9600 Baud.

This setting can only be modified from the serial port.

Default value is $FF (9600 bits/sec) when shipped from factory.

Important: The C-Bus Serial Interface contains very limited buffering of C-Bus traffic destined for the RS-232 port.

As a consequence, the use of serial data rates less than 9600 bits/s is not recommended. Use of these lower rates could lead to occasional loss of information for devices that need to monitor C-Bus traffic.

10.3.4 Interface Options 2 (Note 4)

Interface Options 2 is used to select C-Bus communication options. This parameter should not be changed using the serial port, because these options are normally set by the C-Bus installation software. By default, these options are all disabled.

Changing these options may, in some circumstances, render a network inoperative.

Interface Options 2 has the following meanings:

Bit	Name	Explanation
0	CLOCK GEN	When set, this Serial Interface can act as a C-Bus clock generator. There must be at least one C-Bus clock generator on a network for useful communication to occur.
1		Reserved
2		Reserved
3		Reserved
4		Reserved
5		Reserved
6	BURDEN	When set, enables the C-Bus network burden attached to the Serial Interface. Only relevant if a software selectable burden is implemented in the interface hardware. Changes to Bit 6 must be set twice with two separate commands within a 10 second period in order to effect any change.
7		Reserved

10.3.5 Interface Options 1 Power Up Settings (Note 5)

This parameter mirrors the settings of Interface Options 1 (described above). Its value is copied to Interface Options 1 on power up.

Default value is $00 (*BASIC* Mode, no other options) when shipped from factory.

10.3.6 Interface Options 3 (Note 6)

Interface Options 3 is used to select various Serial Interface options.

This setting can only be modified from the serial port.

Default value is $00 when shipped from the factory.

Interface Options 3 has the following meanings:

Bit	Name	Explanation
0	PCN	When set, a Parameter Change Notification (PCN) message will be emitted by the Serial Interface any time it detects an attempt to updated one or more of its internal parameters, from the C-Bus (eg C-Bus installation software).
1	LOCAL_SAL	When set, the Serial Interface will: a. issue Point to Point commands on the local network as local messages – ie with an empty Network PCI; and b. issue Point to Point to Multipoint commands as though they originated from the local network[10]. This option allows Serial Interface initiated Point to Multipoint commands to behave exactly like commands issued by simple (key input) units. If this option is not set, Serial Interface Initiated Point to Multipoint and Point to Point to Multipoint commands have a slightly different format on the C-Bus network, which can cause minor incompatibility with some devices that do not accept and correctly process all components of a received C-Bus message[11].
2	PUN	When set, a Power Up Notification (PUN) message is emitted by the Serial Interface after it has completed its initialisation and before accepting any characters from its serial input port.

[10] Only effective for Point to Point to Multipoint from firmware version 4.1.0 onwards.

[11] If *LOCAL_SAL* is not set the PCI operates as half-bridge, and inserts additional routing information into outbound Point to Point and Point to Point to Multipoint messages. This extra routing information provides a reverse route to the PCI. This allows the serial port side of the PCI to be treated as another C-Bus network, to which messages can be sent. Because these messages are routed through the PCI, they have no effect on the network to which the PCI is attached.

Bit	Name	Explanation
3	EXSTAT	When set: 1. Switched Status Replies are presented in a format compatible with a Level Status Reply; and 2. All Status Replies are presented in a Long Form with addressing information.
4		Reserved
5		Reserved
6		Reserved
7		Reserved

Recommendation: Clipsal recommend that all new devices using the Serial Interface set the LOCAL_SAL option.

10.3.7 Custom Manufacturer (Note 7)

Custom Manufacturer is a text string, where Clipsal can pre-load a manufacturer name or other identifying information. This is read-only.

10.3.8 Serial Number (Note 8)

Serial number (4 bytes) is assigned to each unit at manufacturing time. It is a read-only parameter.

10.3.9 Custom Types (Note 9)

Custom type is a text string describing the unit type which can be pre-loaded by Clipsal. This is read-only, except to the Devkit software.

Third parties using the C-Bus Serial Interface SIM can set only the last 4 characters, and the custom type will always be set to "INT_XXXX". The C-Bus Development kit software can be used to set the custom type.

Customers using Serial Interface chipsets supplied by CIS under the C-Bus Enabled program will have a custom type allocated by CIS of the form "EN_XXXXX"

11 SERIAL INTERFACE: KNOWN PROBLEMS AND WORKAROUNDS

The Serial Interface firmware issues prior to 3.11 are all pre-release versions and may contain unspecified bugs and functional deficiencies.

Version 3 firmware prior to version 3.16 has a problem with processing a single '|' character. The workaround is to always send two vertical bar characters: '||'

All version 3 issues of C-Bus Serial Interface firmware up to and including 3.16 suffer from a problem which causes the communication baud rate to revert to 9600 baud on regular basis when certain burden clock generation options are selected. Therefore these units can only be used in applications operating at 9600 baud.