Quality System
Quality Endorsed Company
ISO 9001 Lic934
Standards Australia

CLIPSAL INTEGRATED SYSTEMS

C-BUS MODULE

WINDOWS DLL USAGE GUIDE

Document Number: CBUS-CBDLL
Issue: 2.3
Date: 18 May 2005

Applicability: C-Bus Module version 3

Authorised By: J. A. Gerard
CIS Engineering Manager

Comments on this document should be addressed to:

Engineering Manager
Clipsal Integrated Systems Pty Ltd
PO Box 103 Hindmarsh
South Australia 5007

CHANGE HISTORY

Date	Issue	Comments
1 Oct 04	1.1	Update QA logo
29 Nov 04	2.0	Updated to suit C-Bus Module version 3.0
20 Jan 05	2.1	Updated for release 3.0-3.
14 Feb 05	2.2	Updated for release 3.1
18 May 05	2.3	Updated for release 3.3

C-Bus Module Windows DLL Usage Guide

CONTENTS

1 PURPOSE

This document defines the use of the C-Bus DLL for Win32 platforms.

The complete functions and interface to the C-Bus Module are defined in the C-Bus Module Interface Specification.

2 SCOPE

The contents of this document apply to version 3 of the C-Bus Module and C-Bus DLL.

3 REFERENCES

C-Bus Module Interface Specification CBUS-CBMIS

4 INTRODUCTION

The C-Bus Module provides a general-purpose library for software that needs a C-Bus interface.

The C-Bus DLL is made by adding a small number of additional services to support PC/Windows serial ports, and then compiling to a Windows Dynamic Link Library.

The device in which the C-Bus DLL is used is generally an embedded system, communicating with C-Bus via a PC Interface as shown in Figure 1. The C-Bus DLL provides a user-friendly interface between the User Application Firmware and the C-Bus PC Interface.

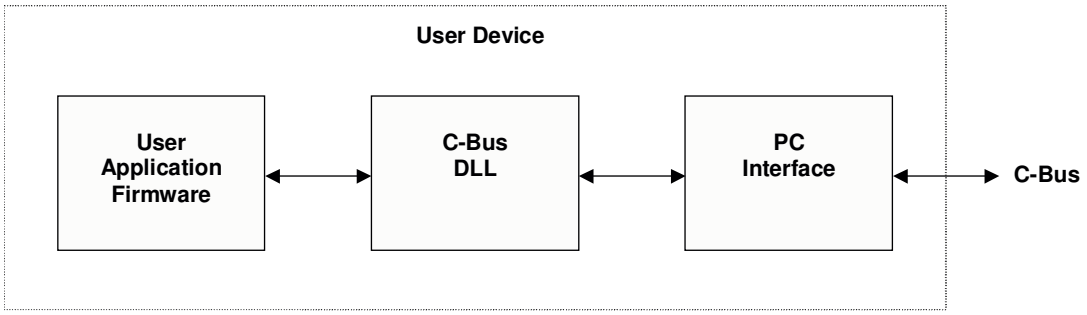

Figure 1 User Device Configuration

5 INSTALLATION AND REMOVAL

5.1 Installing

The C-Bus Module for Windows is supplied as a self-installing setup program. To install, just double-click the supplied program **`setup.exe`**.

5.2 Removing

To remove, open the Control Panel and select Add/Remove Programs. Select the item "C-Bus Module v3" from the list, and click the Remove button.

5.3 File locations

The setup program places files into the following locations:

What	Where
Documentation (PDF and text format)	**`C:\CLIPSAL\C-Bus Module\doc`**
Windows DLL	**`C:\WINNT\SYSTEM32`** **`Or`** **`C:\WINDOWS\SYSTEM32`**
C header files	**`C:\CLIPSAL\C-Bus Module\include`**
Delphi equivalent to C header files	**`C:\CLIPSAL\C-Bus Module\delphi`**
Library for static linking the DLL	**`C:\CLIPSAL\C-Bus Module\lib`**
Source for the windows com port driver	**`C:\CLIPSAL\C-Bus Module\source`**
Example programs	**`C:\CLIPSAL\C-Bus Module\examples`**

6 C-BUS MODULE PROPERTIES

The C-Bus DLL is built from the C-Bus Module with the following properties defined:

- Default Mode of event handling operation
- 256 Networks
- 10 Applications
- 256 Group Addresses per Application
- C-Bus Enabled level 4
- Transmit Queue length = 200
- Receive Queue length = 50
- Tick rate = 100ms
- Process queues on every tick
- Labels Enabled
- Application support for :
    - Lighting
    - Trigger Control
    - Enable Control
    - Air Conditioning
    - Discovery
    - Error Reporting
    - Measurement
    - Security
    - Telephony
    - Time

7 ADDITIONAL FUNCTIONS SUPPLIED WITH DLL

A series of additional functions are provided to ease use on a Windows PC.

Use of these functions is optional[1].

7.1 Enumerating the Communications Ports

```
int8u cbus_comms_if_enum_com_ports(char Ports[8][16])
```

This function that inspects the registry to determine what comm ports are available.

The function returns a count of the number of com ports, and fills in the supplied array with null-terminated strings corresponding to the ports.

The method used to determine the available comm ports may not work with some operating systems. In this case you will need to supply your own function to do this.

7.2 Opening a Com Port

```
bool cbus_comms_bf_open_com_port(char * gszPort)
```

Calling this function:

- Opens the selected comm port;
- Configures the port for 9600 baud, 8 bits per char, no parity and 1 stop bit;
- Registers a serial transmit handler;
- Initialises the C-Bus Module; and
- Starts a 100 ms timer that is used to implement the **`cbus_ef_update`** call

If any of these processes fails the function fails and returns **`false`**. If everything succeeds this returns **`true`**.

The parameter passed to the function is a null terminated string containing the name of the com port to be opened.

7.3 Closing the Comm Port

```
void cbus_comms_vf_close_com_port()
```

When called, this function closes the com port if open, and stops the 100 ms timer. This effectively shuts down the C-Bus Module. If does not unload the DLL.

[1] If these functions are not used, then equivalents will need to be written and used instead.

8 C-BUS DLL USAGE

The C-Bus DLL functions are defined in the C-Bus Module Interface Specification.

8.1 Using the DLL with Borland C-Builder

8.1.1 General notes

Calling the DLL functions is simply a matter of adding the CBM.LIB file to the project and then calling the functions as required. The include path and library path will need to be set appropriately.

8.1.2 Examples

The following examples are snippets of code from working applications and are here only as a guide.

8.1.2.1 Determining Com Ports

```
char Ports[8][16]; // character arrays to store com port strings
int Num_Coms;

// Determine what comm ports are available (DLL call)
   Num_Coms = cbus_comms_if_enum_com_ports(Ports);
```

8.1.2.2 Registering Event Handlers

```
//---------------------------------------------------------------
// Process incoming set level messages. Match the application and
// group before proceeding
//---------------------------------------------------------------
void My_event_handler(byte application,
                      byte group,
                      byte level,
                      byte rate)
{
  ...
}
```

elsewhere in the code:

```
// register the lighting event handler (DLL call)
cbus_lighting_vf_register_event_handler(My_event_handler);
```

8.1.2.3 Calling DLL Functions

```
// Send the message (DLL calls)
cbus_lighting_vf_set_level(0,
  CSpinButton1->Tag,
  CSpinButton2->Tag,
  cbus_if_percent_to_dec(100 - TrackBar1->Position),
  0, 0, 0);
```

8.1.2.4 Handling C-Bus Module Errors

This is just a matter of picking up the last C-Bus error and handling it in whatever way is appropriate to your application.

The following example displays the error code if the function failed. Note that not all C-Bus functions set error codes. Refer to the C-Bus Module Interface documentation to determine which modules do set error codes.

```
if (cbus_if_get_last_error() != CBM_SUCCESS)
 {
   sprintf(Msg, "CBM error = %02x", cbus_if_get_last_error());
   Application->MessageBox(Msg, "Set level error", IDOK);
 }
```

8.2 Using the DLL with Borland Delphi

8.2.1 General notes

When using the C-Bus module DLL, you will need to prepend the function/procedure calls with an underscore.

Pascal definition files are supplied, and need to be included to access the required functions.

8.2.2 Examples

The following examples are snippets of code from working applications and are here only as a guide.

8.2.2.1 Determining Com Ports

```
var
  Num_Coms: Integer;
  Ports: TArrayChar;  // Array of chars to store com port strings

// Determine what com ports are available (DLL call)
Num_Coms := _cbus_comms_if_enum_com_ports(Ports);
```

8.2.2.2 Opening Com Ports

```
// Open the selected com port
if _cbus_comms_bf_open_com_port(PChar(ComboBox1.Text)) then
  begin
    ...
  end
```

8.2.2.3 Calling DLL Functions/Procedures

```
// Send an ON message (DLL call)
_cbus_lighting_vf_set_level(0,
                            SpinButton1.Tag,
                            SpinButton2.Tag,
                            255, 0, 0, 0)
```

8.2.2.4 Including Definition Files

Use something similar to:

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Spin, ExtCtrls, jpeg,
  cbus_pas_defs, cbus_lighting_defs, ComCtrls, cbus_comms;
```

8.3 Using the DLL with Visual Basic

8.3.1 General notes

Prior to using a C-Bus DLL function/procedure, the function/procedure needs to be declared. At this point you can setup an alias to “hide” the requirement to prepend an underscore.

Because the DLL was written in ‘C’, VB applications will be required to do extra manipulation to access character arrays and multidimensional arrays.

8.3.2 Examples

The following examples are snippets of code from working applications written using VB.NET, and are here only as a guide.

8.3.2.1 Determining Com Ports

```
Public Declare Auto Function cbus_comms_if_enum_com_ports Lib
"CBM.DLL" Alias "_cbus_comms_if_enum_com_ports" (ByVal ports(,) As
Byte) As Int32

Dim Ports(7, 15) As Byte    ' Array of Char arrays for ports
Dim Num_Ports As Integer    ' Number of com ports found
Dim ascii As Encoding = Encoding.ASCII
Dim One_Port(15) As Byte

' Determine what com ports are out there (DLL call)
Num_Ports = cbus_comms_if_enum_com_ports(Ports)

' Add the comports to the combobox - a bit messy as VB doesn't
' handle char arrays well
For i = 0 To Num_Ports - 1
  For j = 0 To 15
    One_Port(j) = Ports(i, j)
  Next
  Me.ComboBox1.Items.Add(CType(ascii.GetChars(One_Port, 0, 15),
String))
Next
```

8.3.2.2 Opening Com Ports

```
Public Declare Auto Function cbus_comms_bf_open_com_port Lib
"CBM.DLL" Alias "_cbus_comms_bf_open_com_port" (ByVal port() As
Byte) As Integer

Dim [ascii] As Encoding = Encoding.ASCII
Dim Port As Byte() = [ascii].GetBytes(Me.ComboBox1.Text)
Dim x As Integer

' Open the selected comport (DLL call)
x = cbus_comms_bf_open_com_port(Port)
```

8.3.2.3 Registering Event Handlers

```
Public Declare Auto Sub cbus_lighting_vf_register_event_handler
Lib "CBM.DLL" Alias "_cbus_lighting_vf_register_event_handler"
(ByVal FPtr As MyEventHandler)

' register the lighting event handler (DLL call)
cbus_lighting_vf_register_event_handler(AddressOf
CBus_Event_Handler)
```

8.3.2.4 Calling DLL Functions/Procedures

```
Public Declare Auto Sub cbus_lighting_vf_set_level Lib "CBM.DLL"
Alias "_cbus_lighting_vf_set_level" (ByVal Network As Byte, ByVal
App As Byte, ByVal Grp As Byte, ByVal Level As Byte, ByVal Rate As
Byte, ByVal Retries As Byte)

' send set level message (DLL call)
cbus_lighting_vf_set_level(0, AppEdit.Value, GrpEdit.Value,
TrackBar1.Value, 0, 3)
```

9 COPYRIGHT