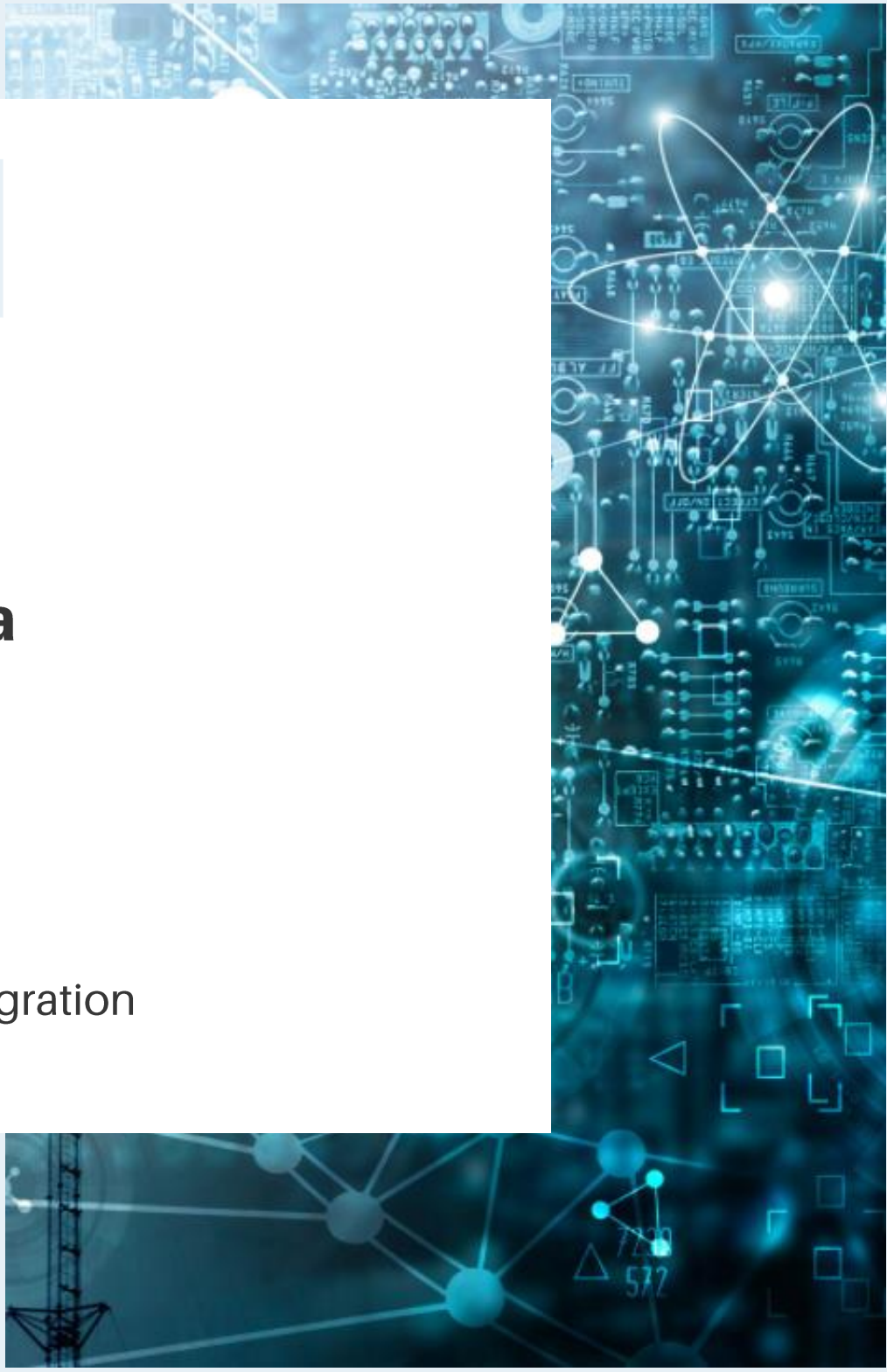
CHIPKIN



EOS Lightmedia
FS-8705-31

Chipkin - Enabling Integration

salesgroup1@chipkin.com

Tel: +1 866 383 1657

Driver Version:
Document Revision: 1

TABLE OF CONTENTS

1 Description

The original purpose of this Driver is to interface EOS with Douglas Lighting WNG-3131 Interface which is BACnet enabled. This driver accepts EOS Lightmedia Ascii Command Protocol messages. These are mapped onto data objects for other protocols like BACnet.

This driver is capable of being linked with other FieldServer drivers to form regular FieldServer firmware that can be installed on QuickServer and other FieldServer gateways. When messages from the EOS are received, they are parsed and the internal data caches / arrays of the FieldServer are updated with status information. Other drivers can access this data and serve using other protocols such as BACnet and Modbus -

The driver is a passive client driver in that it does not poll for data but rather sits passively for the EOS Server to send commands

Max Nodes Supported

FIELDSERVER MODE	NODES	COMMENTS
Passive Client	Many	One EOS Lightmedia Command Server per port.
Server	0	Not supported or documented. For testing and quality purposes only.

Former Driver Type

Serial RS232 / RS485 – Multidrop not supported.
Passive Client

Compatibility Matrix

FIELDSERVER MODEL	COMPATIBLE WITH THIS DRIVER
FS-2010/2011/4010 (Legacy)	Yes,
FS-35 Series	Yes,
FS-QS Series	Yes,

Devices Tested

DEVICE	TESTED (FACTORY, SITE)
	EOS Offices

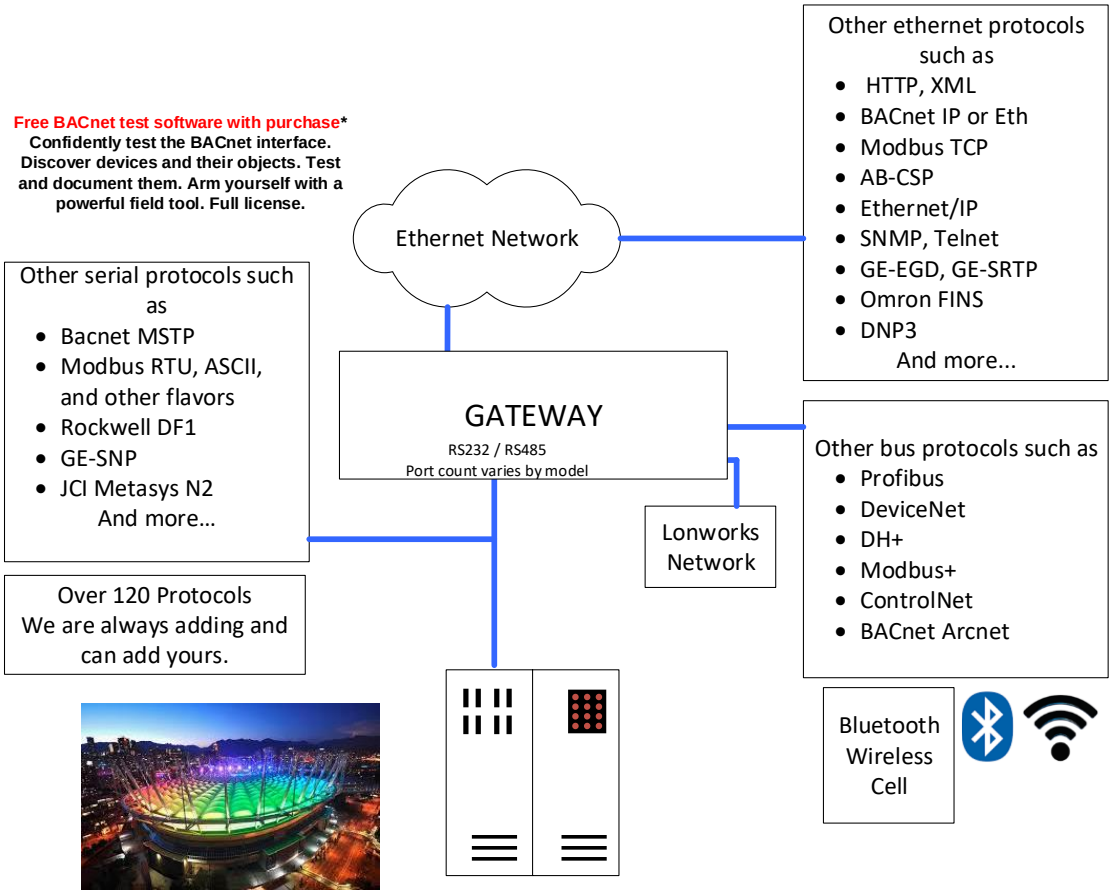
Limitations and Unsupported Devices or Protocol Options

The driver is uni directional – It can accept commands from an EOS Lightmedia Command Server which supports the EOS ASCII Command Protocol but cannot report success / errors or status. (These features can be added on request).

Connection Information

Connection type:	EIA232 / EIA485
Baud Rates:	Driver Supports : **9600**; 19200; 28800; 38400; 57600 Baud .
Data Bits:	Driver Supports : 7,**8**
Stop Bits:	Driver Supports : **1**,2
Parity:	Driver Supports : Odd, Even, **None**
Hardware interface:	N/A
Multidrop Capability	No

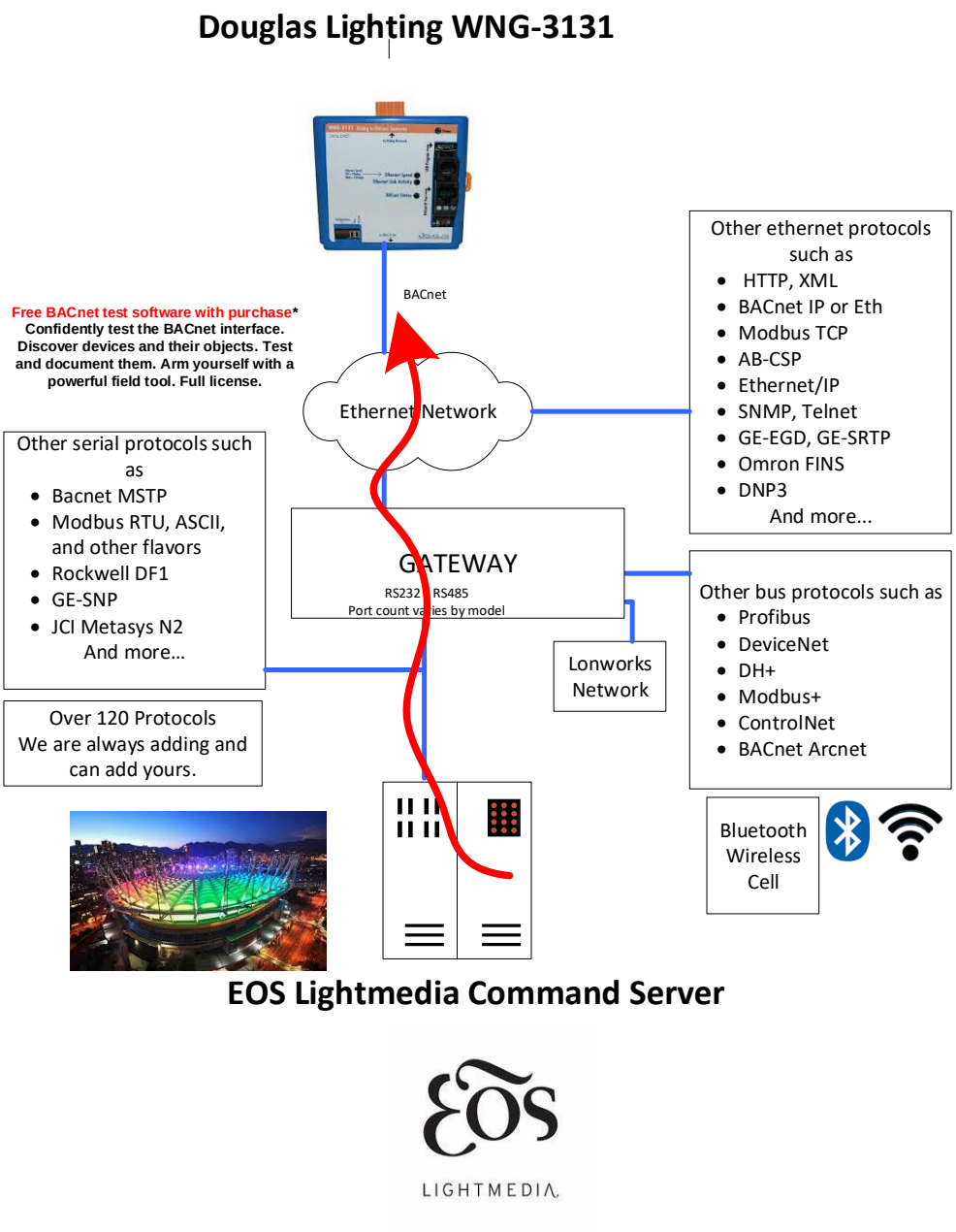
2 General Connection Configurations



EOS Lightmedia Command Server

3 Connection Configuration to WNG-3131

Multiple upstream protocols and connection supported. See list of FieldServer Drivers.



4 Configuration Examples

Define Data Arrays.

One Set for each device.

```
//=====================================================
//
//    Data Arrays,
//
Data_Arrays,,
Data_Array_Name ,Data_Format ,Data_Array_Length
eoslight-stats   ,UINT16      ,100
D1_Groups       ,Bit         ,256
D1_Presets      ,Bit         ,256
D1_Relays       ,Bit         ,256
D2_Groups       ,Bit         ,256
D2_Presets      ,Bit         ,256
D2_Relays       ,Bit         ,256
```

Define the conection and a node for the EOS command Server. R2 is the RS232 port at the bottom of a QickServer.

```
//=============================================================
//
//    Client Side Connections
//
Connections
Port ,Baud ,Data_Bits ,Stop_Bits ,Parity ,Protocol    ,
R2   ,9600 ,8         ,1         ,None   ,eoslight     ,


//=============================================================================,,,,
//
//    Client Side Nodes,,
//
Nodes
Node_Name     ,Node_ID ,Protocol
CommandServer ,1       ,eoslight


//=============================================================================,,
//
//    Client Side Map Descriptors
//
//
// None required
```

Serve the data in BACnet/IP. Remote BACnet devices can read this data.

This is where we define the BACnet objects .

```
//================================================================================,
//
//    Server Side Connections,
//

Connections,
Adapter ,Protocol
N1      ,Bacnet_IP


//================================================================================,
//,
//    Server Side Nodes,
//,,

Nodes,,
Node_Name         ,Node_ID ,Protocol
WNG-3131          ,389001  ,Bacnet_IP

//================================================================================,,
//
//    Server Side Map Descriptors
//

Map_Descriptors,,
Map_Descriptor_Name                ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name      ,Data_Type ,Object_ID ,Property      ,Units
,Data_Array_Low_Scale , Data_Array_High_Scale , Node_Low_Scale , Node_High_Scale ,
D1Group1                           ,d1_groups       ,001              ,Server   ,WNG-3131      ,MI        ,0001      ,Present_Value ,No_Units  ,0,1,1,2
D1Group2                           ,d1_groups       ,002              ,Server   ,WNG-3131      ,MI        ,0002      ,Present_Value ,No_Units  ,0,1,1,2
D1Group3                           ,d1_groups       ,003              ,Server   ,WNG-3131      ,MI        ,0003      ,Present_Value ,No_Units  ,0,1,1,2
D1Group4                           ,d1_groups       ,004              ,Server   ,WNG-3131      ,MI        ,0004      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset1                          ,d1_presets      ,001              ,Server   ,WNG-3131      ,MI        ,1001      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset2                          ,d1_presets      ,002              ,Server   ,WNG-3131      ,MI        ,1002      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset3                          ,d1_presets      ,003              ,Server   ,WNG-3131      ,MI        ,1003      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset4                          ,d1_presets      ,004              ,Server   ,WNG-3131      ,MI        ,1004      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-1                         ,d1_relays       ,005              ,Server   ,WNG-3131      ,bI        ,1001      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-2                         ,d1_relays       ,006              ,Server   ,WNG-3131      ,bI        ,0002      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-3                         ,d1_relays       ,007              ,Server   ,WNG-3131      ,bI        ,0003      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-4                         ,d1_relays       ,008              ,Server   ,WNG-3131      ,bI        ,0004      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay2-1                         ,d1_relays       ,009              ,Server   ,WNG-3131      ,bI        ,0005      ,Present_Value ,No_Units  ,0,1,1,2
```

You can see how the BACnet objects are connected to the Data Arrays that the EOS commands get stored in. This is how the EOS commands are connected to BACnet (or ay other protocol)

You can allocate any BACnet object types and instance numbers

Instead of Serve, in this sample configuration, we send (write) the data to a remote BACnet device #389001 at IP address = 192.168.1.168

Wrbx means write on update.

```
//=====================================================================,
//
//    Server Side Connections,
//

Connections,
Adapter ,Protocol
N1      ,Bacnet_IP


//=====================================================================
//,
//    Server Side Nodes,
//,,

Nodes,,
Node_Name         ,Node_ID ,Protocol    ,IP_Address
WNG-3131          ,389001  ,Bacnet_IP   ,192.168.1.168

//=====================================================================,,
//
//    Server Side Map Descriptors
//

Map_Descriptors,,
Map_Descriptor_Name                ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name      ,Data_Type ,Object_ID ,Property      ,Units
,Data_Array_Low_Scale , Data_Array_High_Scale , Node_Low_Scale , Node_High_Scale ,
D1Group1                           ,d1_groups       ,001              ,Wrbx   ,WNG-3131      ,MI        ,0001      ,Present_Value ,No_Units  ,0,1,1,2
D1Group2                           ,d1_groups       ,002              ,Wrbx   ,WNG-3131      ,MI        ,0002      ,Present_Value ,No_Units  ,0,1,1,2
D1Group3                           ,d1_groups       ,003              ,Wrbx   ,WNG-3131      ,MI        ,0003      ,Present_Value ,No_Units  ,0,1,1,2
D1Group4                           ,d1_groups       ,004              ,Wrbx   ,WNG-3131      ,MI        ,0004      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset1                          ,d1_presets      ,001              ,Wrbx   ,WNG-3131      ,MI        ,1001      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset2                          ,d1_presets      ,002              ,Wrbx   ,WNG-3131      ,MI        ,1002      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset3                          ,d1_presets      ,003              ,Wrbx   ,WNG-3131      ,MI        ,1003      ,Present_Value ,No_Units  ,0,1,1,2
D1Preset4                          ,d1_presets      ,004              ,Wrbx   ,WNG-3131      ,MI        ,1004      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-1                         ,d1_relays       ,005              ,Wrbx   ,WNG-3131      ,bI        ,1001      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-2                         ,d1_relays       ,006              ,Wrbx   ,WNG-3131      ,bI        ,0002      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-3                         ,d1_relays       ,007              ,Wrbx   ,WNG-3131      ,bI        ,0003      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay1-4                         ,d1_relays       ,008              ,Wrbx   ,WNG-3131      ,bI        ,0004      ,Present_Value ,No_Units  ,0,1,1,2
D1Relay2-1                         ,d1_relays       ,009              ,Wrbx   ,WNG-3131      ,bI        ,0005      ,Present_Value ,No_Units  ,0,1,1,2
```

When EOS sends commands the data is stored in these arrays. When a new command comes in, the data for that group/preset/relay is updated. When that happens it triggers a Write to the corresponding BACnet object.

You CANNOT allocate any/arbitary BACnet object types and instance numbers. You must write to objects that exist in the remote BACnet device.

5 How to Driver Works

The driver accepts ascii command strings. It extracts the device, group, preset and relay information as well as the required command to turn on/off.

If the command is a group command then :–

The driver finds the data array named “Dx_Groups”
Where x is the device number.
If the Data Array cannot be found an error is generated.
If the Data Array exists then a 1 (on) or zero (off) is written to offset y in the Data Array
Where y = group number.

Dx_Groups[y] = 1 or 0 where x=device# and y=group#

If the command is a preset command then :–

The driver finds the data array named “Dx_Presets”
Where x is the device number.
If the Data Array cannot be found an error is generated.
If the Data Array exists then a 1 (on) or zero (off) is written to offset y in the Data Array
Where y = preset number.

Dx_presets[y] = 1 or 0 where x=device# and y=preset#

If the command is a relay command then :–

The driver finds the data array named “Dx_Relays”
Where x is the device number.
If the Data Array cannot be found an error is generated.
If the Data Array exists then a 1 (on) or zero (off) is written to offset y in the Data Array
Relays are numbered 1-1 , 1-2 , 1-3 , 1-4 , 2-1 … 64-4
In general Relay m-n
Where y = m*4 + n
Example Relay 20-2 -> Offset y 20*4+2 = 82

Dx_presets[y] = 1 or 0 where x=device# and y=offset as discussed above.

6 Interfacing to other protocols.

Passive Server = Connect the Data Arrays discussed above to server data objects. The gateway waits passively for a remote device using another protocol such as BACnet to request the data. At that point it extracts data from the Data Arrays and responds.

Active Server = The gateway receives EOS commands and uses another protocol such as BACnet to write that data to a remote device (instead of waiting for the remote device to read the data as in the passive server example.) Connect the Data Arrays to active objects of the other protocol. When EOS sends a command that data triggers the active serve.

7 Support

Please contact Chipkin Automation Systems directly for driver support.

The following responses are supported.

8 Revision History

This table summarizes the update history for this document. Please contact Chipkin for an updated version of this document if required.

DATE	RESP	DOC. REV.	COMMENT
22 Mar 2018	PMC	0	Created initial document
21 Oct 2021	YC	1	Updated document format