Chipkin
Automation Systems

Driver Manual

(Supplement to the FieldServer Instruction Manual)

FS-8705-10

Barrington MicroStar Serial Driver

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after Aug 20, 2012

2014 © Chipkin Automation Systems, 3381 Cambie St- #211, Vancouver, BC, Canada, V5Z 4R3
Tel: (866) 383-1657, **Fax:** (416) 915-4024
Email: dfs@chipkin.com **Website:** www.chipkin.com

TABLE OF CONTENTS

1 Barrington MicroStar Driver Description

This driver is used to exchange data between a FieldServer and a 'MicroSTAR' devices. They are also commonly known as 'Barrington MicroSTAR' devices. For this driver to work, the LanStar must be removed as the gateway becomes the master on the trunk of Microstars.

Each device has 4x AO, 4x AI, 4x DI, 4x DO and 4x Counters. The devices can be daisy chained on a RS485 network. Setpoints, virtual Digital Outputs, any virtual points or virtual Microstars all reside in the memory of the LanStar and are not accessible using this driver.

The driver can monitor the status of these points and write to the outputs.

This is an active client driver. The FieldServer is the client. The MicroSTAR devices are the passive servers. The Client sends messages and processes responses from the MicroSTAR's.

The driver is a serial driver using a RS485 serial ports to connect between the FieldServer and the MicroSTAR's. If RS232 ports are used then a converter to RS485 must be used.

The driver is fully compatible with other FieldServer drivers and meets FieldServer's quality assurance standards. The driver was developed by Chipkin Automation Systems, an Approved FieldServer Integrator.

Both Client and Server functionality have been implemented. This means that you can use the driver as a client to monitor/command the devices or you can use it a server to make another type of device appear as if it were a MicroSTAR device. Ie. The server functionality allows you to make a foreign device emulate a MicroSTAR so it can be installed in a MicroSTAR trunk

FIELDSERVER MODE	NODES	COMMENTS
ACTIVE CLIENT	16	A MAXIMUM OF 16 DEVICES MAY BE CONNECTED PER TRUNK. THIS IS A LIMITATION OF THE PROTOCOL WHICH ONLY ALLOWS 16 POSSIBLE ADDRESSES.

Formal Driver Type – Serial - Active Client

2 Driver Scope of Supply

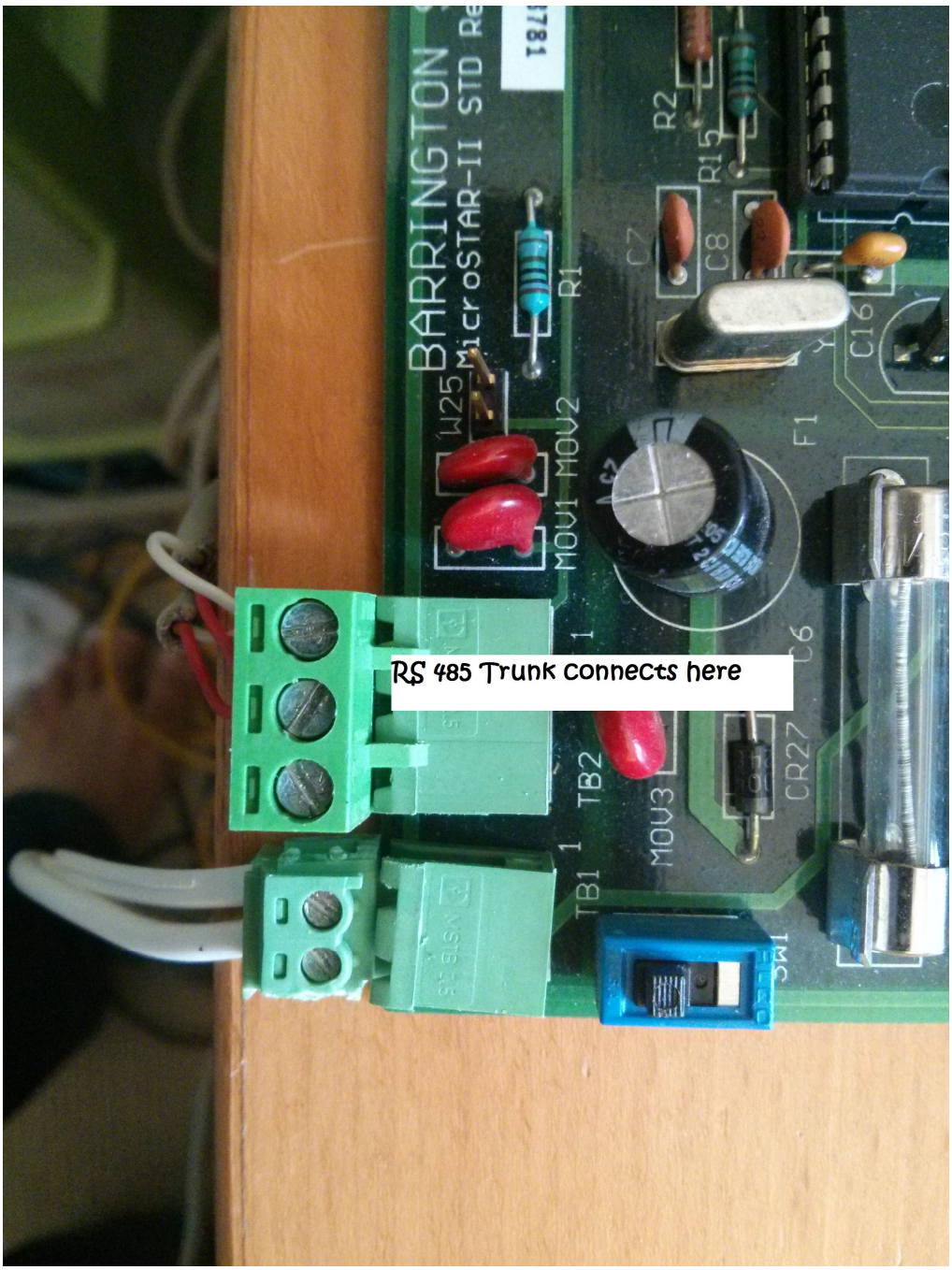
2.1 Supplied with this driver

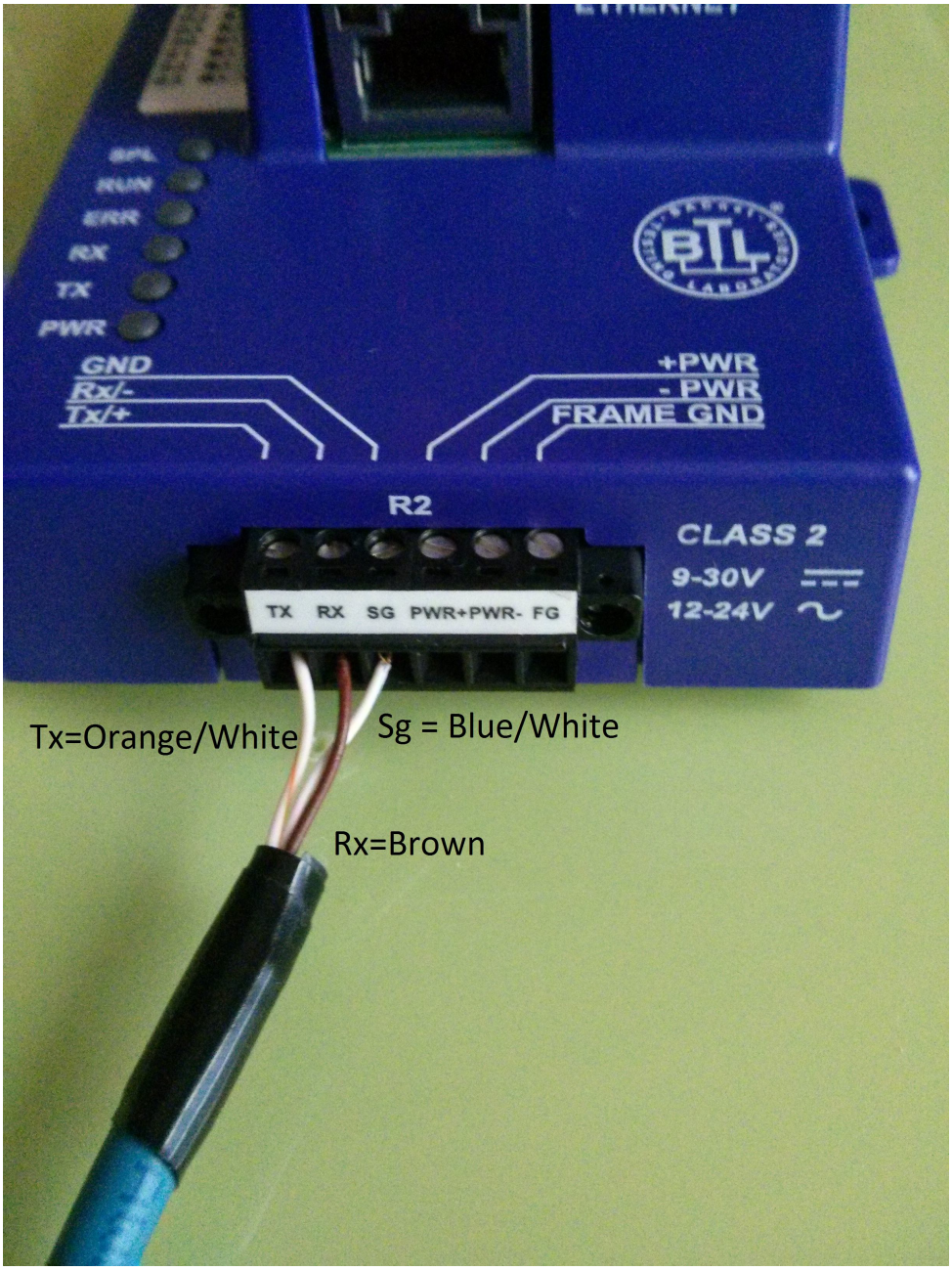
FieldServer Technologies PART #	Description
8915-10	No specific cables are shipped with this driver. A generic RJ45 Ethernet cable is shipped with the hardware.
FS-8705-26	Driver Manual.

3 Hardware Connections

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

3.1 Hardware Connections

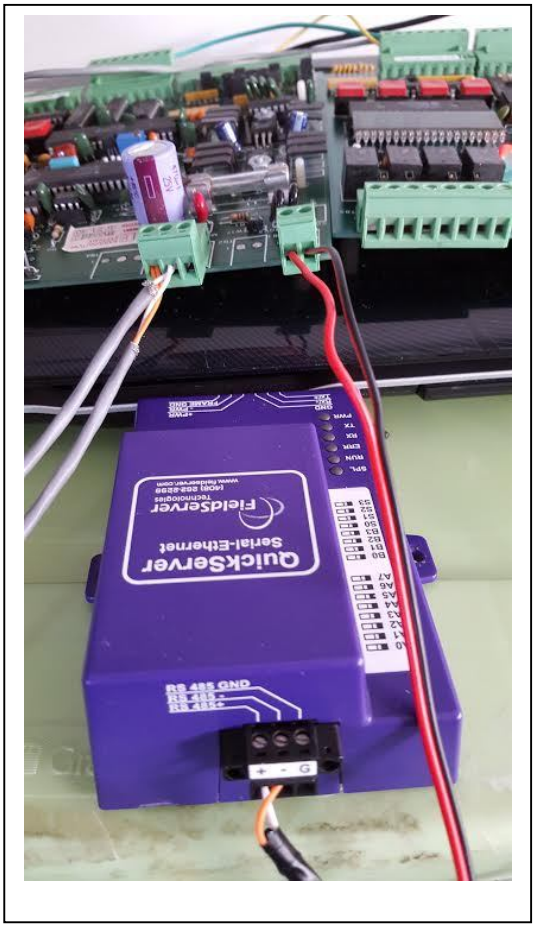


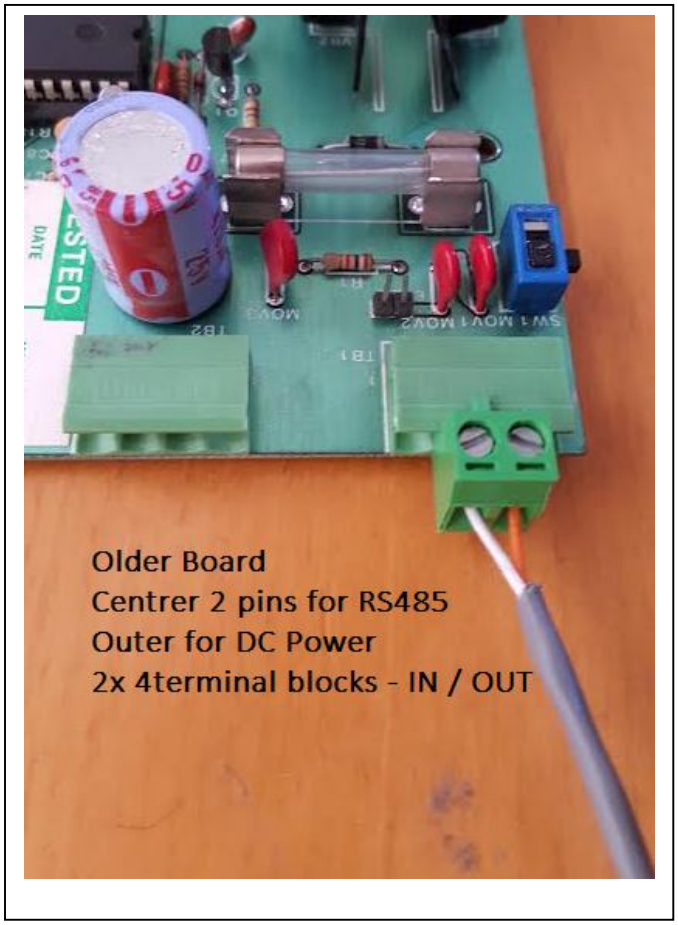


QuickServer Connections

The other end of this cable connect to RS485 port on the Microstar. That connection is shown in other images.

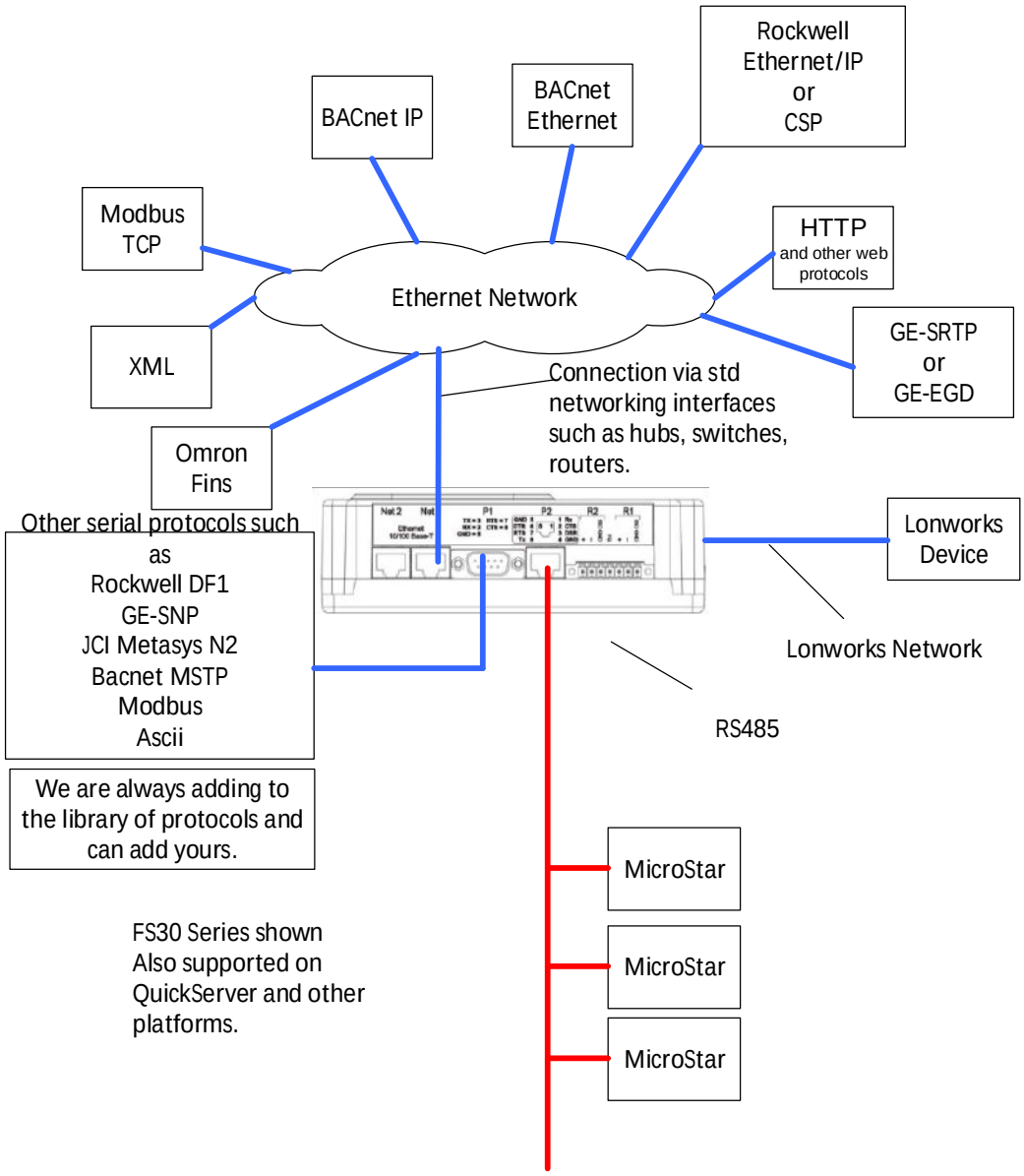
Tested 2015Oct


QuickServer
Serial-Ethernet
Fieldserver
RS 485 GND
+ - G


Older Board
Centrer 2 pins for RS485
Outer for DC Power
2x 4terminal blocks - IN / OUT

3.2 Block Diagram

FS30 shown as representing FS20, FS30 and FS40



FS-8705-10 Barrington Microstar Protocol (Lanstar Replacement)

4 Configuring the FieldServer as a Barrington MicroStar Client

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" sample files provided with the FS).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an Barrington MicroStar system.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Barrington MicroStar monitoring, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

4.1 Data Arrays

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	**Recommended**: Bit, UInt16, Also Supported: Float, Uint32, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10,000

4.1.1 Data Arrays - Example

// Data Arrays		
Data_Arrays		
Data_Array_Name,	Data_Format,	Data_Array_Length,
MicroStarStats,	UNT16,	200

4.2 Client Side Connections

Create one connection for each Barrington MicroStar serial port. Each connection can only be used to connect to a single Barrington MicroStar interface/port.

Section Title
Connections

Column Title	Function	Legal Values
Port	Specify which port the device is connected to the FieldServer	P1-P8
Protocol	Specify protocol used	Barrington MicroStar
Baud*	Specify baud rate	Driver Supports : 110; 300; 600; 1200; 2400; 4800; **9600**; 19200; 28800; 38400; 57600 Baud *Barrington MicroStar* supports: 9600
Data_Bits *	Specify parity	Driver Supports : 7,**8** *Barrington MicroStar* supports: 7
Stop_Bits*	Specify data bits	Driver Supports : **1**,2 *Barrington MicroStar* supports: 1
Parity *	Specify stop bits	Driver Supports : Odd, Even, **None** *Barrington MicroStar* supports: Even
Mstar_checksum_offset	In forming a poll to the Microstar, the driver starts computing the checksum at the 3rd byte. Ie buffer[2] and hence the value 2 is used.	Value should be set to 2 unless directed by CAS support
Mstar_data_offset	Driver expects the data to start	Value should be set to 0 unless

	in the 4th byte of a response. Adjust this expectation by adjusting the offset.	directed by CAS support
Mstar_invert_data	When set to 1 the byte order in the 16 bit word is reversed in computing the analog value.	Value should be set to 0 unless directed by CAS support

4.2.1 Client Side Connection Descriptions - Example

```
//      Client Side Connections


Connections

Port,       Baud,       Parity,     Protocol, Data_Bits , Stop_Bits

P1,         9600,       Even,       Barrington MicroStar , 7        , 1
```

4.3 Client Side Nodes

Create one Node per FACP in the network only.

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters **NB !** The name does not need to correspond to the Node name configured in the Barrington MicroStar system.
Node_ID	Station address of physical server node This parameter is not used directly by the driver. We recommend that a unique Node ID's be given to each node.	0-64 Corresponds to the Node numbers of panels. Master node is typically zero.
Protocol	Specify protocol used	Barrington MicroStar

4.3.1 Client Side Nodes - Example

```
//  Client Side Nodes


Nodes
Node_Name,      Node_ID,    Protocol,           Connection
MainPanel,      0,          Barrington MicroStar ,  P1
```

4.4 Client Side Map Descriptors

4.4.1 FieldServer Related Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from "Data Array" section above
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor..	Passive

4.4.2 Driver Related Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
Data_Type	This commonly used parameter is not used by this driver.	
Length	Length of Map Descriptor Reserves space in the Data Array.	Always set to 14 as 14 points are served from each Microstar device.
Address	The Point Number	Always set to zero.

4.5 Examples

4.5.1 Map Descriptor Example 1 – Read Points

In this example points are reach from 2 nodes

```
Map_Descriptors

Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , Function , Length , Address

Read_Node0          , DA_MSTAR00      , 0                 , MSTAR00   , rdbc     , 14     , 1

Read_Node1          , DA_MSTAR01      , 0                 , MSTAR01   , rdbc     , 14     , 1
```

Descrip tive only. Not used.

Data Array and offset where data will be stored.

Node Name of a Node defined in the Nodes section. Node Number must correspond to the Barrington Lanstar panel number. Name must match too.

Read continuously, over and over. Use Scan_Interval to control frequency.

Set Length to 14 (In some circumstances this is set to 15)

The 15th Data Item is the counter reset byte.

5 Sample Configuration File

```
//=====================================================================
//
//   Common Information
//

Bridge
Title                    ,system_Node_Id
2NodeSample         ,3


//=====================================================================
//
//   Data Arrays
//

Data_Arrays
Data_Array_Name ,Data_Format ,Data_Array_Length
DA_Node_01     ,Float      ,20
DA_Node_02     ,Float      ,20


//=====================================================================
//
//   Client Side Connections
//

Connections
Port ,Baud ,Data_Bits ,Stop_Bits ,Parity ,Protocol
R1   ,9600 ,8        ,1         ,None   ,STAR


//=====================================================================
//
//   Client Side Nodes
//

Nodes
Node_Name ,Node_ID ,Protocol ,Port
Node_01   ,1       ,STAR     ,R1
Node_02   ,2       ,STAR     ,R1


//=====================================================================
//
//   Client Side Map Descriptors
//

Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Length
ReadNode1          ,0s            ,DA_Node_01     ,0                 ,Rdbc     ,Node_01   ,14
ReadNode2          ,0s            ,DA_Node_02     ,0                 ,Rdbc     ,Node_01   ,14



//---------------------------------------------------------------------
```

```
//=====================================================================
//
//   Client Side Map Descriptors
//

Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name ,Length
ReadNode1        ,0s        ,DA_Node_01    ,0            ,Rdbc    ,Node_01  ,14
ReadNode2        ,0s        ,DA_Node_02    ,0            ,Rdbc    ,Node_01  ,14



//=====================================================================
//
//   Server Side Connections
//
Connections
Port , Baud  , Protocol    , Timeout , Connection_Type  , Max_Master
R2   , 38400 , BACnet_MSTP , 30      , MSTP_Master_Mode , 127



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

Nodes
Node_Name    ,Node_ID ,Protocol
Bac_Node_01  ,01      ,BACnet_MSTP
Bac_Node_02  ,02      ,BACnet_MSTP

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

Map_Descriptors
Map_Descriptor_Name ,Data_Array_Name ,Data_                                                                    ode_Low_Scale
,Node_High_Scale
AI_01          ,DA_Node_01    ,0          ,Server  ,Bac_Node_01 ,1     ,AI     ,0          ,100          ,0          ,100
AI_02          ,DA_Node_01    ,1          ,Server  ,Bac_Node_01 ,2     ,AI     ,0          ,100          ,0          ,100
AI_03          ,DA_Node_01    ,2          ,Server  ,Bac_Node_01 ,3     ,AI     ,0          ,100          ,0          ,100
AI_04          ,DA_Node_01    ,3          ,Server  ,Bac_Node_01 ,4     ,AI     ,0          ,100          ,0          ,100
Ao_01          ,DA_Node_01    ,4          ,Server  ,Bac_Node_01 ,5     ,Av     ,0          ,100          ,0          ,100
Ao_02          ,DA_Node_01    ,5          ,Server  ,Bac_Node_01 ,6     ,Av     ,0          ,100          ,0          ,100
Ao_03          ,DA_Node_01    ,6          ,Server  ,Bac_Node_01 ,7     ,Av     ,0          ,100          ,0          ,100
Ao_04          ,DA_Node_01    ,7          ,Server  ,Bac_Node_01 ,8     ,Av     ,0          ,100          ,0          ,100
DI_01          ,DA_Node_01    ,8          ,Server  ,Bac_Node_01 ,9     ,bi     ,0          ,100          ,0          ,100
DO_01          ,DA_Node_01    ,9          ,Server  ,Bac_Node_01 ,10    ,bv     ,0          ,100          ,0          ,100
CI_01          ,DA_Node_01    ,10         ,Server  ,Bac_Node_01 ,11    ,AI     ,0          ,100          ,0          ,100
CI_02          ,DA_Node_01    ,11         ,Server  ,Bac_Node_01 ,12    ,AI     ,0          ,100          ,0          ,100
CI_03          ,DA_Node_01    ,12         ,Server  ,Bac_Node_01 ,13    ,AI     ,0          ,100          ,0          ,100
CI_04          ,DA_Node_01    ,13         ,Server  ,Bac_Node_01 ,14    ,AI     ,0          ,100          ,0          ,100

Map_Descriptors
```

Of the 14 points stored they are stored at the following relative offsets in the Data Array

AI's: 0,1,2,3
AO's: 4,5,6,7
BI's: Offset 8 (use 4 bits)
Bo's: Offset 9 (use 4 bits)
Counters: Offset 10,11,12,13

Configuring the FieldServer as a Barrington MicroStar Server

This driver cannot be used to emulate an Barrington MicroStar Panel

Writing Data to a Microstar

Configure the gateway to read data from the Microstar. This is important because when a write is performed all Binary Output and Analog Output are written in the single command. Thus by reading before a write, you can ensure you will write back current values to the outputs.

To write to Analog Outputs:

The data for the Analog Outputs is stored at Data Array offsets 4,5,6,7

To write to Binary Outputs:

The data for the Binary Outputs is stored at Data Array offset 9. A byte value is extracted and written. Binary Output 0 is bit 0 of the the byte and BO(3) is bit 3.

To write to Reset Counters:

The data for the Reset is stored at Data Array offset 14 (the 15th element). A byte value is extracted and written. Counter 0 is bit 0 of the the byte and Co(3) is bit 3. A 1 in the bit positionis reset. A zero means no action.

DatA Array Offset	Point Type	Point Number	
0	Ai	0	
1	Ai	1	
2	Ai	2	
3	Ai	3	
4	Ao	0	Write to this offset to change Ao(0) value
5	Ao	1	
6	Ao	2	
7	Ao	3	
8	Bi	0..3	
9	Bo	0..3	Write a byte to this offset to change all Bo's.
10	Co	0	
11	Co	1	
12	Co	2	
13	Co	3	
14	Co Reset Byte		Set bit to 1 to reset.

Tel: (866) 383-1657, **Fax:** (416) 915-4024

Appendix 1 – Advanced Topics

Appendix 1.1. Scaling

Points CANNOT be scaled when they are read because each read returns 14 points of different types. They can be scaled by a task that executes on data update or a fixed frequency or they can be scaled when they are served.

Example: Scale before served. Ie scaled number is served.

```
Map_Descriptors

Map_Descriptor_Name ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name
,Object_Id,Data_Type  ,Data_Array_Low_Scale ,Data_Array_High_Scale ,Node_Low_Scale
,Node_High_Scale

AI_01               ,DA_Node_01      ,0                ,Server   ,Bac_Node_01  ,1       ,AI
,0                    ,100                  ,0             ,100

AI_02               ,DA_Node_01      ,1                ,Server   ,Bac_Node_01  ,2       ,AI
,0                    ,100                  ,0             ,100
```

Appendix 1.2. Supported Communications functions

Always check the Data Sheet for an accurate and up to date list.

Function	Notes
Poll Request with Download Data	Writes 4xAO values and 4xDO states. Can also be used to reset one/all of the 4 Counters.
Poll Request	Reads all data for all inputs. Analog inputs are 16 bits values.

Driver cannot read/write data to virtual microSTAR's.

Appendix 1.3. Trouble Shooting

Videos of LED operation are available at

Appendix 1.4. Reset Counters

Each read reads 14 data items and updates 14 consecutive elements in the Data Array.

The 15th element of the Data Array is used to reset the Counters. One bit for each counter. Bit 0 is for Counter 0. When a write occurs the driver extracts the 15th Element of the Data Array and uses that byte to reset the counters.

This 15th Element can be auto cleared (auto set back to zero) after a write. To do this define a Data Array with the name mstarstats. Set the 3rd Element (offset zero to 1). Without this, the driver does the default, sends the reset byte and leaves its value unchanged in the data array.

Appendix 1.5. Sample Configuration

```
//===============================================================================

//

// Notes :  None.

//

//

//===============================================================================




//===============================================================================

//

//    Common Information

//


Bridge

Title                                ,System_Station_Address

FSe4907 rev1 Uncfgrd 1-866-383-1657     ,11


//===============================================================================

//

//    Data Arrays

//

Data_Arrays

Data_Array_Name ,Data_Format ,Data_Array_Length

DA_01_DATA         ,UINT16      ,20

DA_02_DATA         ,UINT16      ,20

mstarstats         ,UINT16      ,1000


Preloads,

Data_Array_Name  ,Preload_Data_Index,Preload_Data_Value

// Emable special write mode - Set offset zero to 1 (enable) and offset 1 to byte value. lower nibble is used
to reset counters (1) or do nothing (0).

// Using the values below. All counters will be reset every time a write is performed. This is not the normal
requirement.
```

```
//mstarstats    ,0                ,1

// Special Write Value

//mstarstats    ,1                ,15

// Counter Reset Control - if the following item=1 then after a write, the Counter Reset byte (offset 14) is
set to zero in the DA after it has been used.

//mstarstats    ,1                ,15



//================================================================================

//

//    Client Side Connections

//



Connections

Port ,Baud ,Data_Bits ,Stop_Bits ,Parity ,Protocol ,Mstar_checksum_offset ,Mstar_data_offset ,Mstar_invert_data
,Poll_Delay,

R1   ,4800 ,8        ,1         ,none, Star     ,2                    ,0                ,0
,0.01s ,



//================================================================================

//

//    Client Side Nodes

//



Nodes

Node_Name  ,Node_ID ,Protocol ,Port,recovery_interval,retry_interval,timeout

Node_MS_01 ,1       ,Star     ,R1  ,1.1s             ,1.1s          ,2.0s

Node_MS_02 ,2       ,Star     ,R1  ,1.1s             ,1.1s          ,2.0s



//================================================================================

//

//    Client Side Map Descriptors

//

//

//      OFFSET 14 (15th element) IS USED.

//      When a write occurs offset 14 is extracted and sent as the counter reset byte.

//      The LSbit is for counter 0. Writing a 1 for the counter results in a reset.

//      Writing a zero results in no action. Set the Length to 15 if you want to trigger a reset whenever you
set the reset. Leave as length=14 if you only want resets of counters to occur when you write to other points
```

```
//
//


Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name  ,Address ,Length
ReadNode_01         ,1s            ,DA_01_DATA      ,0                 ,Rdbc     ,Node_MS_01 ,0       ,14
ReadNode_02         ,1s            ,DA_02_DATA      ,0                 ,Rdbc     ,Node_MS_02 ,0       ,14
```

Appendix 1.6. Auto Configuration

MicroStar Auto configuration

Qualifying

QS-1010 - 1 Mstar trunk up to 16 devices on R1, MSTP on R2

QS-1010 - 2 Mstar trunks up to 16 devices on R1, another 16 on R2, BACnetIP

QS-1010 - 1Mstar trunks up to 16 devices on R1, R2 unused, BACnetIP

Firmware and config

Firmware and driver version 5.01 and later is required.

Discover2.csv

- is loaded onto the gateway as config.csv (automatic name change when downloaded).
- Is use to discover devices on trunk R1 and R2
- When discovery is complete a message is printed in the combined message log reporting that the discovery is complete and the results of the discovery will be applied after a trigger.

Discovery

Phase 1 – Activate the discovery configuration

Poke the value 97 (BACNetIP) into the Data Array called AUTO_CONFIGURE at offset 99.

The driver changes the value to 96 and reboots itself about 10-20 seconds later.

In doing so it self installs a configuration suitable for discovery.

OR

Poke the value 95 (BACNetMSTO) into the Data Array called AUTO_CONFIGURE at offset 99.

The driver changes the value to 94 and reboots itself about 10-20 seconds later.

In doing so it self installs a configuration suitable for discovery only on trunk R1

Once rebooted look in the combined message log for a message that reports discovery is complete.

Now check the values assigned in AUTO_CONFIGURE or check the BACnetIP device 389001 and check the values of the objects.

This process can be instated at any time.

Phase II – Perform discovery and generate permanent configuration file.

For BACnetIP

Offset 90 == Number of Trunks = 2 or 1

Offset 91 == The BACnetIP Node_Id that is allocated to the gasteway – the gateway will appear as a BACnet device.

Offset 92 == Each Server Node is allocated a Node_Id instance number. The value found here is added to the Mstar number. Eg. If AUTO_CONFIGURE[92] = 1000 then Mstar00 on trunk 1 will be numbered as device 1000, MSTAR01 as device 1001 …

Offset 94 == MSTP required = 0

Offset 98 == BACnetIP Network Number

OR

Discover device=389001 named AutoConfigCtrl

And set values as in the above notes and then set the present_value of the Trigger object AV(99) to 99.

Trigger ,AV ,99 ,

MaxTrunks ,AV ,90 ,

NetworkNumber ,AV ,98 ,

GateWay_NodeNum (IP) ,AV ,91 ,

BACnetIP Base NodeNum ,AV ,92 ,

Finally – set the value of offset 99 to 99 to trigger the re-configuration based on the discovery. The gateway will reboot itself some 10-30 seconds later. Now it will have a configuration suitable for the discovered nodes.

You can begin this process again at any time.

Auto Discovery will not occur except when triggered.

For BACnetMSTP

Offset 90 == Number of Trunks = 1

Offset 96 == The MAC address that is allocated to the gateway – the gateway will appear as a BACnet device.

Offset 97 == Each Server Node is allocated a Node_Id instance number. The value found here is added to the Mstar number. Eg. If AUTO_CONFIGURE[92] = 1000 then Mstar00 on trunk 1 will be numbered as device 1000, MSTAR01 as device 1001 …

Offset 94 == MSTP required = 1

OR

Discover device=389001 named AutoConfigCtrl

And set values as in the above notes and then set the present_value of the Trigger object AV(99) to 99.

Trigger ,AV ,99 ,

MaxTrunks ,AV ,90 ,

MSTPRequired ,AV ,94 ,

GateWay_Mac(MSTP) ,AV ,96 ,

BACnetMSTP Base NodeNum ,AV ,97 ,

Finally – set the value of offset 99 to 99 to trigger the re-configuration based on the discovery. The gateway will reboot itself some 10-30 seconds later. Now it will have a configuration suitable for the discovered nodes.

You can begin this process again at any time.

Auto Discovery will not occur except when triggered.

Sample configuration suitable for discovery devices on 2 trunks

```
Bridge

Title                          ,System_Station_Address

Mstar - Discover 2 trunk       ,11

Data_Arrays

Data_Array_Name ,Data_Format ,Data_Array_Length

mstarstats      ,UINT16     ,1000

AUTO_CONFIGURE    ,UINT32     ,1000

Preloads,

Data_Array_Name  ,Preload_Data_Index,Preload_Data_Value

AUTO_CONFIGURE   ,90            ,2

AUTO_CONFIGURE   ,94            ,0

AUTO_CONFIGURE   ,91            ,22

AUTO_CONFIGURE   ,92            ,1000

AUTO_CONFIGURE   ,96            ,33

AUTO_CONFIGURE   ,97            ,20

AUTO_CONFIGURE   ,98            ,55

//   max_trunk           = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 90 );

//   gateway_ip_instance     = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 91 );

//   server_ip_base_nodeID   = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 92 );

//   mstp_required         = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 94 );

//   gateway_mstp_mac       = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 96 );

//   server_mstp_base_nodeID = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 97 );

//   network_number         = da_get_uint32 ( (DAH_TYP) mstar_global_auto_config_array , 98 );
```

```
Connections

Port ,Baud ,Data_Bits ,Stop_Bits ,Parity ,Protocol ,Mstar_checksum_offset ,Mstar_data_offset ,Mstar_invert_data ,Poll_Delay,

R1  ,4800 ,8     ,1      ,none, Star    ,2              ,0           ,0              ,0.01s ,

R2  ,4800 ,8     ,1      ,none, Star    ,2              ,0           ,0              ,0.01s ,

Nodes

Node_Name  ,Node_ID ,Protocol ,Port,recovery_interval,retry_interval,timeout

T1_M00   ,0    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M01   ,1    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M02   ,2    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M03   ,3    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M04   ,4    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M05   ,5    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M06   ,6    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M07   ,7    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M08   ,8    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M09   ,9    ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M10   ,10   ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M11   ,11   ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M12   ,12   ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M13   ,13   ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M14   ,14   ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T1_M15   ,15   ,Star   ,R1  ,0.1s         ,0.1s       ,0.6s

T2_M00   ,0    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s

T2_M01   ,1    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s

T2_M02   ,2    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s

T2_M03   ,3    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s

T2_M04   ,4    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s

T2_M05   ,5    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s

T2_M06   ,6    ,Star   ,R2  ,0.1s         ,0.1s       ,0.6s
```

```
T2_M07   ,7    ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M08   ,8    ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M09   ,9    ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M10   ,10   ,Star   ,R2 ,0.1s        ,0.1s      ,2.0s

T2_M11   ,11   ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M12   ,12   ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M13   ,13   ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M14   ,14   ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s

T2_M15   ,15   ,Star   ,R2 ,0.1s        ,0.1s      ,0.6s


Map_Descriptors

Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Function ,Node_Name  ,Address ,Length

discover T1_M00   ,1s       ,AUTO_CONFIGURE  ,200         ,Rdbc   ,T1_M00   ,0    ,14

discover T1_M01   ,1s       ,AUTO_CONFIGURE  ,220         ,Rdbc   ,T1_M01   ,0    ,14

discover T1_M02   ,1s       ,AUTO_CONFIGURE  ,240         ,Rdbc   ,T1_M02   ,0    ,14

discover T1_M03   ,1s       ,AUTO_CONFIGURE  ,260         ,Rdbc   ,T1_M03   ,0    ,14

discover T1_M04   ,1s       ,AUTO_CONFIGURE  ,280         ,Rdbc   ,T1_M04   ,0    ,14

discover T1_M05   ,1s       ,AUTO_CONFIGURE  ,300         ,Rdbc   ,T1_M05   ,0    ,14

discover T1_M06   ,1s       ,AUTO_CONFIGURE  ,320         ,Rdbc   ,T1_M06   ,0    ,14

discover T1_M07   ,1s       ,AUTO_CONFIGURE  ,340         ,Rdbc   ,T1_M07   ,0    ,14

discover T1_M08   ,1s       ,AUTO_CONFIGURE  ,360         ,Rdbc   ,T1_M08   ,0    ,14

discover T1_M09   ,1s       ,AUTO_CONFIGURE  ,380         ,Rdbc   ,T1_M09   ,0    ,14

discover T1_M10   ,1s       ,AUTO_CONFIGURE  ,400         ,Rdbc   ,T1_M10   ,0    ,14

discover T1_M11   ,1s       ,AUTO_CONFIGURE  ,420         ,Rdbc   ,T1_M11   ,0    ,14

discover T1_M12   ,1s       ,AUTO_CONFIGURE  ,440         ,Rdbc   ,T1_M12   ,0    ,14

discover T1_M13   ,1s       ,AUTO_CONFIGURE  ,460         ,Rdbc   ,T1_M13   ,0    ,14

discover T1_M14   ,1s       ,AUTO_CONFIGURE  ,480         ,Rdbc   ,T1_M14   ,0    ,14

discover T1_M15   ,1s       ,AUTO_CONFIGURE  ,500         ,Rdbc   ,T1_M15   ,0    ,14
```

```
discover T2_M00   ,1s       ,AUTO_CONFIGURE ,600        ,Rdbc   ,T2_M00   ,0    ,14

discover T2_M01   ,1s       ,AUTO_CONFIGURE ,620        ,Rdbc   ,T2_M01   ,0    ,14

discover T2_M02   ,1s       ,AUTO_CONFIGURE ,640        ,Rdbc   ,T2_M02   ,0    ,14

discover T2_M03   ,1s       ,AUTO_CONFIGURE ,660        ,Rdbc   ,T2_M03   ,0    ,14

discover T2_M04   ,1s       ,AUTO_CONFIGURE ,680        ,Rdbc   ,T2_M04   ,0    ,14

discover T2_M05   ,1s       ,AUTO_CONFIGURE ,700        ,Rdbc   ,T2_M05   ,0    ,14

discover T2_M06   ,1s       ,AUTO_CONFIGURE ,720        ,Rdbc   ,T2_M06   ,0    ,14

discover T2_M07   ,1s       ,AUTO_CONFIGURE ,740        ,Rdbc   ,T2_M07   ,0    ,14

discover T2_M08   ,1s       ,AUTO_CONFIGURE ,760        ,Rdbc   ,T2_M08   ,0    ,14

discover T2_M09   ,1s       ,AUTO_CONFIGURE ,780        ,Rdbc   ,T2_M09   ,0    ,14

discover T2_M10   ,1s       ,AUTO_CONFIGURE ,800        ,Rdbc   ,T2_M10   ,0    ,14

discover T2_M11   ,1s       ,AUTO_CONFIGURE ,820        ,Rdbc   ,T2_M11   ,0    ,14

discover T2_M12   ,1s       ,AUTO_CONFIGURE ,840        ,Rdbc   ,T2_M12   ,0    ,14

discover T2_M13   ,1s       ,AUTO_CONFIGURE ,860        ,Rdbc   ,T2_M13   ,0    ,14

discover T2_M14   ,1s       ,AUTO_CONFIGURE ,880        ,Rdbc   ,T2_M14   ,0    ,14

discover T2_M15   ,1s       ,AUTO_CONFIGURE ,900        ,Rdbc   ,T2_M15   ,0    ,14


Adapters

Adapter ,Protocol ,

N1    ,BACnet_IP ,

Nodes

Node_Name       ,Node_ID ,Protocol ,

AutoConfigCtrl   ,389001  ,BACnet_IP,

Map_Descriptors

Map_Descriptor_Name    ,Data_type ,Object_Id ,Data_Array_Name    ,Data_Array_Offset ,Function ,Node_Name     ,Length

Trigger            ,AV     ,99     ,Auto_Configure     ,99          ,Server   ,AutoConfigCtrl  ,1

MaxTrunks          ,AV     ,90     ,Auto_Configure     ,90          ,Server   ,AutoConfigCtrl  ,1

NetworkNumber      ,AV     ,98     ,Auto_Configure     ,98          ,Server   ,AutoConfigCtrl  ,1

GateWay_NodeNum (IP)   ,AV     ,91     ,Auto_Configure     ,91          ,Server   ,AutoConfigCtrl  ,1

BACnetIP Base NodeNum   ,AV     ,92     ,Auto_Configure     ,92          ,Server   ,AutoConfigCtrl  ,1

MSTPRequired       ,AV     ,94     ,Auto_Configure     ,94          ,Server   ,AutoConfigCtrl  ,1

GateWay_Mac(MSTP)     ,AV     ,96     ,Auto_Configure     ,96          ,Server   ,AutoConfigCtrl  ,1
```

Appendix 1.7. Revision History

Date	Resp	Format	Driver Ver.	Doc. Rev.	Comment
2008Feb11	PMC		1.0	1.0	Created
2014Sep21	PMC		2.0	2.0	Updated for new connection params
2015Oct20	PMC		3	3	Updated.
2016Mar02	PMC		5.01	5	Auto Discovery introduced.