FieldServer
FS-8700-68 CEI-ABI Protocol Driver for EN.54 Exchanges

Driver Manual

(Supplement to the FieldServer Instruction Manual)

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after November 2015

Kernel Version:	1.08
Document Revision:	1

Contact Information:

Thank you for purchasing the FieldServer.

Please call us for Technical support of the FieldServer product.

Contact Information:

Sierra Monitor Corporation
1991 Tarob Court
Milpitas, CA 95035

Contact number:
+1 408 262-6611
+1 800 727-4377

Email: info@sierramonitor.com

Website: www.sierramonitor.com

TABLE OF CONTENTS

LIST OF FIGURES

1 CEI-ABI DRIVER DESCRIPTION

The CEI-ABI Driver allows the FieldServer to transfer data to and from EN.54 exchanges over either RS-232 or RS-485 using CEI-ABI Driver protocol.

The protocol can work over point to point connections on RS-232 or over a multidrop RS-485 line allowing for multiple exchanges on the same communications line. Every exchange has to be manually set to a unique address. The addresses used on a line must be from 1 to 16 in single bit binary e.g. (1, 2, 4, 8… 16).

The Driver enters a configuration and alignment mode after startup. During this mode the following steps are followed:

- Zone alignment data is requested from the exchanges (80 zones)
- Loop alignment data is requested from the exchanges (16 loops)
- A request analogs command (27) is dispatched on each loop on every exchange (16 loops)
- Analog alignment data is requested on each loop from the exchanges (16 loops)

Once the configuration mode is finished, the Driver starts to poll the exchanges for event data. Event data are alarms, troubles and system events on the exchanges.

Every exchange essentially contains 16 loops with 99 sensors and 99 modules. The collection of sensors and modules over all 16 loops are called points. There are 198 points on a loop and 16*198 = 3,168 points on an exchange.

Every point has a **status value** retrieved from the exchange during loop alignment:

0 = NORMAL
1 = ALARM
7...14 = TROUBLE
57 = not installed
61, 67 = disabled
70 = ON
71 = OFF

Every point has an **analog value** from 0 to 255 also retrieved from the exchange during analog alignment.

Points can be mapped into zones. 32 points can be mapped into a single zone giving a total of 32*80 = 2560 points. A point's status is therefore displayed in the Map Descriptor Data Array assigned for the points as well as in the Data Array for the zone. This mapping is simply a convenient way to group specific points into functional zones representing areas in a building, e.g. the workshop is zone 0, etc.

During the zone alignment stage, the points to zone mappings are retrieved from the exchanges and stored. The Client Driver uses this setup information to store point data in the correct zone Map Descriptor Data Arrays. Each zone's label (e.g. 'workshop') is also retrieved and stored for user access during zone alignment. Each zone's state at alignment is also stored. Possible zone states in bit positions are:

0 = ZONE OK
2 = ZONE ENABLED
4 = ZONE DISABLED
8 = ZONE IN ALARM
16 = ZONE IN TROUBLE

Zone setup data can be stored in the Server Map Descriptors to map points to zones. A single zone's setup data consists of 64 bytes defined as follows:

Even bytes (0, 2, 4 …) = device number 1...99
Odd bytes (1, 3, 5 … (bits 8-12)) = loop number 0...15

Each word of zone setup data uniquely defines a point on a specific loop to belong to a zone. The zones start from 0 to 79.

During the normal polling stage the Driver checks for events from the exchange. Points and zone changes can cause events that will be reported to the Driver in response to a normal poll packet. The Driver will indicate a zone or point event in the Map Descriptor Data Array by placing the event status value in the array. The device that caused the event and a detail description code of the event will be placed in the correct point or zone device and point or zone event Data Arrays for user access. Refer to the CEI-ABI protocol specification for a detailed list of the device and event codes.

A special event code of value 138 will cause the Driver to re-enter the alignment mode. Exchanges report this code whenever a programming change has been made on the exchange (e.g. a zone label changed).

The Driver can also be set up in a listen only mode to act as a data tap. In this mode, another device polls the exchanges and the Driver stores the response data from the replying exchanges. The setup is done by specifying a node type of "Data_Tap" in a client csv file.

The Driver also supports a reset function whereby the client can send a reset to Servers. The data tap listens for the Server's "command accepted" message before clearing its own point faults bitmap Data Array. The Client clears its Data Array once the "command accepted" message is sent from the Server.

NOTE

The Driver is set up for 80 zones and 16 loops. The Data Arrays declared for each type of cei data contains applicable data arranged consecutively according to zone or loop number (from 0 to the last number).

A special Map Descriptor pointing to a bitmap Data Array of point faults (alarms and troubles) must also be declared in the Client and Data-Tap CSV files. The point faults bitmap Data Array contains a simple 0 or 1 to indicate a point fault or not. (1 = fault, 0 = normal). Note that only sensor module point faults are stored in this Data Array.

The following table shows the type of points found under each loop and their offsets in the Data Arrays. This table represents one complete exchange or node.

Detectors			**Modules**		
Loop	**Offset start**	**Offset end**	**Loop**	**Offset start**	**Offset end**
0	0	98	0	99	197
1	198	296	1	297	395
2	396	494	2	495	593
3	594	692	3	693	791
4	792	890	4	891	989
5	990	1088	5	1089	1187
6	1188	1286	6	1287	1385
7	1386	1484	7	1485	1583
8	1584	1682	8	1683	1781
9	1782	1880	9	1881	1979

Detectors			**Modules**		
Loop	**Offset start**	**Offset end**	**Loop**	**Offset start**	**Offset end**
10	1980	2078	10	2079	2177
11	2178	2276	11	2277	2375
12	2376	2474	12	2475	2573
13	2574	2672	13	2673	2771
14	2772	2870	14	2871	2969
15	2970	3068	15	3069	3167

2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by Sierra Monitor Corporation for this Driver

Sierra Monitor Corporation Part #	Description
-	RS-485 connection adapter

2.2 Provided by the Supplier of 3rd Party Equipment

2.2.1 Required 3rd Party Hardware

Part #	Description
	RS-232 or RS-485 serial cable

3 HARDWARE CONNECTIONS

The FieldServer is connected to the En.54 exchange as shown below. Configure the En.54 exchange according to manufacturer's instructions.

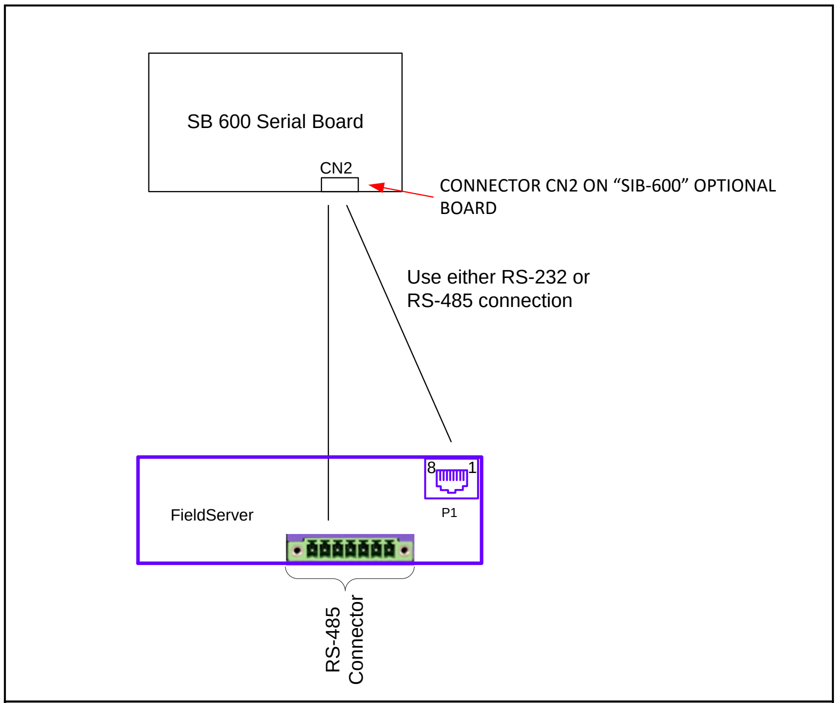
SB 600 Serial Board
CN2
CONNECTOR CN2 ON "SIB-600" OPTIONAL BOARD
Use either RS-232 or RS-485 connection
8
1
P1
FieldServer
RS-485 Connector

Figure 1 - Generic Connection Diagram

TERMIN. NR.		DENOMINATION		IT-485 WIRING	SERIAL PORT CONNECTOR ON FIELDSERVER
1		GROUND			
2		RTS			
3		CTS			
4	**RTS-232**	TX			**RJ45-01**
5		RX	SUPERVISION		**RJ45-08**
6		GND			**RJ45-04**
7		LIN + OUTWARD	PC	TERMINAL 4	
8		LIN – RETURN			
9	**RTS-485**	GND	WIRING	SHIELD	
10		LIN + RETURN			
11		LIN - OUTWARD		TERMINAL 3	

3.1 Connection Notes

- For wiring of maximum 25 meters use 232 serial line (terminals 4-5-6)
- For wiring exceeding 15 meters use 485 serial line with IT-485 interface (terminals 7-11)
- **NB:** If the FieldServer is to be connected to the Serial Printer Port of the AM6000 Panel, use the Driver "FS-8700-52 Notifier Italia AM6000". If the FieldServer is to be connected to the AM6000 panel using an SIB-600 device, use the Driver "FS-8700-68 CEI ABI

4 DATA ARRAY PARAMETERS

Data Arrays are “protocol neutral” data buffers for storage of data to be passed between protocols. It is necessary to declare the data format of each of the Data Arrays to facilitate correct storage of the relevant data.

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.	Bit, 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

Example

```
//   Data Arrays
Data_Arrays
Data_Array_Name     , Data_Format   , Data_Array_Length
Counts_1            , Byte          , 1
Pnt_sts_1           , Byte6         , 3168
Pnt_faults_bmap_1   , Bit           , 3168
Pnt_analogs_1       , Byte          , 3168
Pnt_Dev_codes_1     , Byte          , 3168
Pnt_Evt_codes_1     , Byte          , 3168
Zone_sts_1          , Byte          , 80
Zn_Dev_codes_1      , Byte          , 80
Zn_Evt_codes_1      , Byte          , 80
Zone_labels_1       , Byte          , 2560
Zone_pnt_sts_1      , Byte          , 2560
Zone_setup_1        , Byte          , 5120
Reset_1             , Bit           , 1
Analogs_1_0         , Byte          , 198
```

The counts Data Array is a dummy array for the poller Map Descriptor. It does not contain any data.

5 CONFIGURING THE FIELDSERVER AS A CEI-ABI DRIVER 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 FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an EN.54 exchange.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for CEI-ABI Driver communications, 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.

5.1 Client Side Connection Parameters

Section Title
Connections

Column Title	**Function**	**Legal Values**
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[1]
Baud*	Specify baud rate	2400, 4800, **9600**, 19200 (Vendor limitation)
Parity*	Specify parity	**Even** (Vendor limitation)
Data_Bits*	Specify data bits	**8** (Vendor limitation)
Stop_Bits*	Specify stop bits	**1** (Vendor limitation)
Protocol	Specify protocol used	CEI (Case insensitive)
Poll Delay*	Time between internal polls	0-32000s, **3.0s**
ReAlign_Mode*	This parameter allows the user to specify that only loops must be aligned at startup/reset. Refer to example in Appendix A.1.	**All**, Loops

Example

```
//   Client Side Connections

Port    , Protocol    , Baud     , Parity
P1      , CEI         , 19200    , Even
```

[1] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

5.2 Client Side Node Parameters

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Generic node id unique to port	0-255
Protocol	Specify protocol used	CEI (Case insensitive)
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[2]

Example

```
//    Client Side Nodes

Nodes
Node_Name    , Node_ID     , Protocol     , Connection
Node_0       , 0           , CEI          , P1
```

5.3 Client Side Map Descriptor Parameters

5.3.1 FieldServer Specific 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 Section 4.
Data_Array_Offset	Starting location in Data Array	0 to (Data_Array_Length-1) as specified in Section 4.
Function	Function of Client Map Descriptor	Rdbc, Passive

5.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the Node names specified in Section 5.2
Cei_Address	The remote exchange's address	1, 2, 4, 8, 16
Cei_Type	The specific type of cei data referred to by the Map Descriptor	poller, analogs_poller, points, p_faults_bmp, analogs, p_devices, p_events, zones, z_devices, z_events, labels, zone_points, setup, reset
Loop*	The Analog loop on the exchange to poll	0 – 15, **-**

[2] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

5.3.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval*	Rate at which data is polled	≥2.0s per exchange (Increase value if more than one exchange on a RS-485 line), **-**

5.3.4 Map Descriptor Example.

```
Map_Descriptor_Name  , Scan_Interval  Data_Array_Name      Data_Array_Offset , Function , Node_Name , Cei_Address , Cei_Type      , Length
Poller_1             , 7.0s           , Counts_1           , 0               , Rdbc     , Node_0    , 1           , Poller        , 3168
Exch_1_P_S           , 0s             , Pnt_sts_1          , 0               , Passive  , Node_0    , 1           , Points        , 3168
Exch_1_P_F_B         , 0s             , Pnt_faults_bmap_1  , 0               , Passive  , Node_0    , 1           , p_faults_bmp  , 3168
Exch_1_P_A           , 0s             , Pnt_analogs_1      , 0               , Passive  , Node_0    , 1           , Analogs       , 3168
Exch_1_P_D_C         , 0s             , Pnt_Dev_codes_1    , 0               , Passive  , Node_0    , 1           , p_devices     , 3168
Exch_1_P_E_C         , 0s             , Pnt_Evt_codes_1    , 0               , Passive  , Node_0    , 1           , p_events      , 3168
Exch_1_Z_S           , 0s             , Zone_sts_1         , 0               , Passive  , Node_0    , 1           , Zones         , 80
Exch_1_Z_D_C         , 0s             , Zn_Dev_codes_1     , 0               , Passive  , Node_0    , 1           , z_devices     , 80
Exch_1_Z_E_C         , 0s             , Zn_Evt_codes_1     , 0               , Passive  , Node_0    , 1           , z_events      , 80
Exch_1_Z_L           , 0s             , Zone_labels_1      , 0               , Passive  , Node_0    , 1           , Labels        , 2560
Exch_1_Z_P_S         , 0s             , Zone_pnt_sts_1     , 0               , Passive  , Node_0    , 1           , zone_points   , 2560
Exch_1_Z_S_U         , 0s             , Zone_setup_1       , 0               , Passive  , Node_0    , 1           , Setup         , 5120
Exch_1_Reset         , 2.0s           , Reset_1            , 0               , Rdbc     , Node_0    , 1           , Reset
```

This can be any name but each name must be unique. Name will appear in FieldServer Map Descriptor status information screens.

Scan interval must be adapted for multiple Map Descriptor scans. Only the poller has a real scan value.

A Data Array name defined in Section 4

The offset into the Data Array where the data will be stored.

Only read and passive allowed. Read is for pollers and passive for data.

Node name defined in Section 5.2. The Node_Name identifies the port on which the exchange is connected.

The remote exchange address.

The type of cei data the Map Descriptor refers to. Used by Driver internals to store data in multiple locations.

```
Map_Descriptor_Name  , Scan_Interval  , Data_Array_Name  , Data_Array_Offset  , Function  , Node_Name  , Cei_Address  , Cei_Type        , Loop  , Length
Exch_1_Analog_1      , 7.0s           , Analogs_1_0      , 0                  , Rdbc      , Node_0     , 1            , Analogs_poller  , 0     , 198
```

Add more Map Descriptors to poll other loops.

Analog loops 0 to 15 can be polled.

6 CONFIGURING THE FIELDSERVER AS A CEI-ABI DRIVER SERVER

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 FieldServer)

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for CEI-ABI Driver communications, the Driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the FieldServer virtual node(s) needs to be declared in the "Server Side Nodes" section, and the data to be provided to the Client needs to be mapped in the "Server 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.

6.1 Server Side Connection Parameters

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[3]
Baud*	Specify baud rate	2400, 4800, **9600**, 19200
Parity*	Specify parity	**Even**
Data_Bits*	Specify data bits	**8**
Stop_Bits*	Specify stop bits	**1**
Protocol	Specify protocol used	CEI (Case insensitive)
ReAlign_Mode*	This parameter allows the user to specify that only loops must be aligned at startup/reset. Refer to example in Appendix A.1.	**All**, Loops

Example

```
//   Server Side Connections

Connections
Port    , Protocol    , Baud     , Parity
P1      , CEI         , 19200    , Even
```

6.2 Server Side Node Parameters

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Generic node id unique to port	0-255
Protocol	Specify protocol used	CEI (Case insensitive)

[3] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

Example

```
//    Server Side Nodes

Nodes
Node_Name    , Node_ID    , Protocol
Node_0          , 0              , CEI
```

6.3 Server Side Map Descriptor Parameters

6.3.1 FieldServer Specific 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 Section 4
Data_Array_Location	Starting location in Data Array	0 to (Data_Array_Length -1) as specified in Section 4
Function	Function of Client Map Descriptor	Passive

6.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in Section 6.2
Cei_Address	The remote exchange's address	1, 2, 4, 8, 16
Cei_Type	The specific type of cei data referred to by the Map Descriptor	points, analogs, devices, events, zones, labels, setup

6.3.3 Map Descriptor Example.

```
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Cei_Address   , Cei_Type   , Length
Exch_1_P_S            , Pnt_sts_1         , 0                   , Passive    , Node_0      , 1             , Points     , 3168
Exch_1_P_A            , Pnt_analogs_1     , 0                   , Passive    , Node_0      , 1             , Analogs    , 3168
Exch_1_D_C            , Dev_codes_1       , 0                   , Passive    , Node_0      , 1             , Devices    , 3168
Exch_1_E_C            , Evt_codes_1       , 0                   , Passive    , Node_0      , 1             , Events     , 3168
Exch_1_Z_S            , Zone_sts_1        , 0                   , Passive    , Node_0      , 1             , Zones      , 80
Exch_1_Z_L            , Zone_labels_1     , 0                   , Passive    , Node_0      , 1             , Labels     , 2560
Exch_1_Z_S_U          , Zone_setup_1      , 0                   , Passive    , Node_0      , 1             , Setup      , 5120
```

This can be any name but each name must be unique. Name will appear in FieldServer Map Descriptor status information screens.

The Data Array as defined in Section 4. Data from the forth script file will be stored into the array at Data_Array_Offset . This data will be sent to a requesting client.

This value specifies the offset into the Data Array where the data from the forth script will be stored. Note that the script can offset the data in addition to this offset value.

Function may not be read or write since it implements a Server. Function may only be passive.

Node Name as defined in Section 5.2. . This defines the port on which the exchange is connected.

The exchange address.

The cei data type the Map Descriptor points to. Used by Driver internals to retrieve data.

7 CONFIGURING THE FIELDSERVER AS A CEI-ABI DATA TAP

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 FieldServer)

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an EN.54 exchange.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for CEI-ABI Driver communications, 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.

7.1 Data Tap Connection Parameters

Section Title
Connections

Column Title	Function	Legal Values
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[4]
Baud*	Specify baud rate	2400, 4800, **9600**, 19200
Parity*	Specify parity	**Even**
Data_Bits*	Specify data bits	**8**
Stop_Bits*	Specify stop bits	**1**
Protocol	Specify protocol used	CEI (Case insensitive)
ReAlign_Mode*	This parameter allows the user to specify that only loops must be aligned at startup/reset. Refer to example in Appendix A.1.	**All**, Loops

Example

```
//    Client Side Connections

Connections
Port    , Protocol     , Baud       , Parity
P1      , CEI          , 19200      , Even
```

[4] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

7.2 Data Tap Node Parameters

Section Title		
Nodes		
Column Title	**Function**	**Legal Values**
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Generic node id unique to port	0-255
Protocol	Specify protocol used	CEI (Case insensitive)
Port	Specify which port the device is connected to the FieldServer.	P1-P8, R1-R2[4]
Node_type	Specify this is a listen only node	Data_Tap

Example

```
//    Client Side Nodes

Nodes
Node_Name    , Node_ID     , Protocol    , Port    , Node_Type
Node_0       , 0           , CEI         , P1      , Data_Tap
```

7.3 Data Tap Map Descriptor Parameters

7.3.1 FieldServer Specific 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 Section 4
Data_Array_Location	Starting location in Data Array	0 to (Data_Array_Length -1 as specified in Section 4
Function	Function of Client Map Descriptor	Passive

7.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to tap data from	One of the node names specified in Section 7.2
Cei_address	The remote exchange's address	1, 2, 4, 8, 16
Cei_type	The specific type of cei data referred to by the Map Descriptor	points, p_faults_bmp, analogs, p_devices, p_events, zones, z_devices, z_events, labels, zone_points, setup

7.3.3 Map Descriptor Example.

```
Map_Descriptor_Name   , Data_Array_Name      , Data_Array_Offset  , Function  , Node_Name  , Cei_Address   , Cei_Type       , Length
Exch_1_P_S            , Pnt_sts_1            , 0                  ,Passive    , Node_0     , 1             , Points         , 3168
Exch_1_P_F_B          , Pnt_faults_bmap_1    , 0                  ,Passive    , Node_0     , 1             , p_faults_bmp   , 3168
Exch_1_P_A            , Pnt_analogs_1        , 0                  ,Passive    , Node_0     , 1             , Analogs        , 3168
Exch_1_P_D_C          , Pnt_Dev_codes_1      , 0                  ,Passive    , Node_0     , 1             , p_devices      , 3168
Exch_1_P_E_C          , Pnt_Evt_codes_1      , 0                  ,Passive    , Node_0     , 1             , p_events       , 3168
Exch_1_Z_S            , Zone_sts_1           , 0                  ,Passive    , Node_0     , 1             , Zones          , 80
Exch_1_Z_D_C          , Zn_Dev_codes_1       , 0                  ,Passive    , Node_0     , 1             , z_devices      , 80
Exch_1_Z_E_C          , Zn_Evt_codes_1       , 0                  ,Passive    , Node_0     , 1             , z_events       , 80
Exch_1_Z_L            , Zone_labels_1        , 0                  ,Passive    , Node_0     , 1             , Labels         , 2560
Exch_1_Z_P_S          , Zone_pnt_sts_1       , 0                  ,Passive    , Node_0     , 1             , zone_points    , 2560
Exch_1_Z_S_U          , Zone_setup_1         , 0                  ,Passive    , Node_0     , 1             , Setup          , 5120
```

This can be any name but each name must be unique. Name will appear in FieldServer Map Descriptor status information screens.

The Data Array name must be one found under Data_Arrays. Data from the scan will be stored into the array at Data_Array_Offset

The offset into the Data Array where the data will be stored.

Function may not be write. Only read and passive. Read is for pollers and passive for data.

Node name must be one found under Nodes, Node_Name. The Node_Name identifies the port on which the exchange is connected.

The remote exchange address.

The type of cei data the Map Descriptor refers to. Used by the Driver internals to store data in multiple locations.

Appendix A. USEFUL FEATURES

Appendix A.1. Align Loops only

This parameter allows the user to specify that only loops must be aligned at start-up/reset. The parameter needs to be set to “Loops” to activate this functionality.

```
Connections
Port    , Baud     , Data_Bits    , Stop_Bits    , Parity    , Protocol    , ReAlign_Mode
R1      , 19200    , 8            , 1            , Even      , cei         , Loops
```

Appendix B. VENDOR INFORMATION

Appendix B.1. Mapping the points to Modbus:

The following table shows how the cei points in a single Data Array containing all 16 loops may be mapped to Modbus points. The p_faults_bmap type should be used and the Data Array type should be set to bit.

Loop	Points type	Source_Data_Array	Source_Offset	Target_Data_Array	Target_Offset	Length
0	Detectors	Pnt_flts_bmap_1	0	MB_alarms	1	99
0	Modules	Pnt_flts_bmap_1	99	MB_alarms	101	99
1	Detectors	Pnt_flts_bmap_1	198	MB_alarms	201	99
1	Modules	Pnt_flts_bmap_1	297	MB_alarms	301	99
2	Detectors	Pnt_flts_bmap_1	396	MB_alarms	401	99
2	Modules	Pnt_flts_bmap_1	495	MB_alarms	501	99
3	Detectors	Pnt_flts_bmap_1	594	MB_alarms	601	99
3	Modules	Pnt_flts_bmap_1	693	MB_alarms	701	99
4	Detectors	Pnt_flts_bmap_1	792	MB_alarms	801	99
4	Modules	Pnt_flts_bmap_1	891	MB_alarms	901	99
5	Detectors	Pnt_flts_bmap_1	990	MB_alarms	1001	99
5	Modules	Pnt_flts_bmap_1	1089	MB_alarms	1101	99
6	Detectors	Pnt_flts_bmap_1	1188	MB_alarms	1201	99
6	Modules	Pnt_flts_bmap_1	1287	MB_alarms	1301	99
7	Detectors	Pnt_flts_bmap_1	1386	MB_alarms	1401	99
7	Modules	Pnt_flts_bmap_1	1485	MB_alarms	1501	99
8	Detectors	Pnt_flts_bmap_1	1584	MB_alarms	1601	99
8	Modules	Pnt_flts_bmap_1	1683	MB_alarms	1701	99
9	Detectors	Pnt_flts_bmap_1	1782	MB_alarms	1801	99
9	Modules	Pnt_flts_bmap_1	1881	MB_alarms	1901	99
10	Detectors	Pnt_flts_bmap_1	1980	MB_alarms	2001	99
10	Modules	Pnt_flts_bmap_1	2079	MB_alarms	2101	99
11	Detectors	Pnt_flts_bmap_1	2178	MB_alarms	2201	99
11	Modules	Pnt_flts_bmap_1	2277	MB_alarms	2301	99
12	Detectors	Pnt_flts_bmap_1	2376	MB_alarms	2401	99
12	Modules	Pnt_flts_bmap_1	2475	MB_alarms	2501	99
13	Detectors	Pnt_flts_bmap_1	2574	MB_alarms	2601	99
13	Modules	Pnt_flts_bmap_1	2673	MB_alarms	2701	99
14	Detectors	Pnt_flts_bmap_1	2772	MB_alarms	2801	99
14	Modules	Pnt_flts_bmap_1	2871	MB_alarms	2901	99
15	Detectors	Pnt_flts_bmap_1	2970	MB_alarms	3001	99
15	Modules	Pnt_flts_bmap_1	3069	MB_alarms	3101	99

Appendix C. TROUBLESHOOTING

Appendix C.1. Interpreting Ruidebug logs

Appendix C.1.1. Preparing the Log

When using the Data-Tap Driver, the comms log typically looks as follows:

```
==================================================================
PORT_LOGGING started on Thu Apr 18 02:31:51 2002
02 00 49 00 06 F0 51 03 02 00 29 00 06 EE 51 03 02 00 49 00 07 30 90 03 02 00 29 00 07 2E 90 03 02 00 49 00 05 F1 11 03 02
00 29 00 05 EF 11 03 02 00 09 00 06 24 50 03 02 00 29 00 06 EE 51 03 02 00 09 00 07 E4 91 03 02 00 29 00 07 2E 90 03 02 00
09 00 05 25 10 10 03 02 00 29 00 05 EF 11 03 02 00 49 00 06 F0 51 03 02 00 29 00 06 EE 51 03 02 00 49 00 07 30 90 03 02 00
29 00 07 2E 90 03 02 00 49 00 05 F1 11 03 02 00 29 00 05 EF 11 03 02 00 09 00 06 24 50 03 02 00 29 00 06 EE 51 03 02 00 09
00 07 E4 91 03 02 00 29 00 07 2E 90 03 02 00 09 00 05 25 10 10 03 02 00 29 00 05 EF 11 03 02 00 49 00 06 F0 51 03 02 00 29
00 06 EE 51 03 02 00 49 00 07 30 90 03 02 00 29 00 07 2E 90 03 02 00 49 00 05 F1 11 03 02 00 29 00 05 EF 11 03 02 00 09 00
06 24 50 03 02 00 29 00 06 EE 51 03 02 00 09 00 07 E4 91 03 02 00 29 00 07 2E 90 03 02 00 09 00 05 25 10 10 03 02 00 29 00
05 EF 11 03 02 00 49 00 06 F0 51 03 02 00 29 00 06 EE 51 03 02 00 49 00 07 30 90 03 02 00 29 00 07 2E 90 03 02 00 49 00 05
F1 11 03 02 00 29 00 05 EF 11 03 02 00 09 00 06 24 50 03
```

The Data-Tap comms log shows all the messages seen by the Data-Tap Driver. To better see the source and meaning of messages, the messages should be separated so they are each displayed on a separate line in the comms log file. To separate the messages, use the following procedure:

- Using a text editor e.g. Vslick, search for "03 02" which indicates the end of one message and the start of the next message.
- Move the editor's cursor to the start of the 02 and insert a new line. (Use a macro). The final result of the above log should look like this:

```
================================================================
PORT_LOGGING started on Thu Apr 18 02:31:51 2002
02 00 49 00 06 F0 51 03
02 00 29 00 06 EE 51 03
02 00 49 00 07 30 90 03
02 00 29 00 07 2E 90 03
02 00 49 00 05 F1 11 03
02 00 29 00 05 EF 11 03
02 00 09 00 06 24 50 03
02 00 29 00 06 EE 51 03
02 00 09 00 07 E4 91 03
02 00 29 00 07 2E 90 03
02 00 09 00 05 25 10 10 03
02 00 29 00 05 EF 11 03
02 00 49 00 06 F0 51 03
02 00 29 00 06 EE 51 03
02 00 49 00 07 30 90 03
02 00 29 00 07 2E 90 03
02 00 49 00 05 F1 11 03
02 00 29 00 05 EF 11 03
02 00 09 00 06 24 50 03
02 00 29 00 06 EE 51 03
02 00 09 00 07 E4 91 03
02 00 29 00 07 2E 90 03
02 00 09 00 05 25 10 10 03
02 00 29 00 05 EF 11 03
02 00 49 00 06 F0 51 03
...
```

Appendix C.1.2. Basic Message Protocol

A message such as the following consists of the following fields:

```
02 00 09 00 06 24 50 03
```

Field	Description	
02	ASCII start of text character (indicates the start of a message).	
00	Databyte count	
09	Flag byte. Indicates the source of a message among other things. The following flags are of importance:	
	09	Message is a poll from a supervision centre (may also be called a VDU).
	49	
	29	Message is a reply from an exchange (may also be called a fire control panel).
00 06	Address. Indicates the source or destination address of a message. Address = 6. A VDU does not have an address, so the address field always applies to an exchange.	
24 50	Checksum	
03	ASCII end of text character (indicates the end of a message).	

Appendix C.1.3. Finding events in the comms log

All events in the comms log can be found by searching for AA. The AA is a fixed byte in the data field which all event messages contain.

Appendix C.1.4. Reporting alarms and troubles

Point alarms and troubles are contained in messages sent from an exchange. Supervision centers receive these messages and indicate these alarms and troubles on a user interface. Point alarms are indicated by 01 01 in the comms log.

02 0F 29 00 10 03 08 01 08 AA 17 2D 66 A3 42 57 00 A0 10 03 **01 01** 9A 0D 03

Field	Description	
02	Start of text	
0F	Databyte count = 15	
29	Message is from exchange	
00 10 03	Message is from address 3. Note the 10 is an escape sequence character used to indicate that the following byte is not an “End of text” character. Escape sequence characters must be ignored.	
08 01 08 AA 17 2D 66 A3 42 57 00 A0 10 03 01 01	Data bytes to be interpreted according to the protocol spec. In this case the following byte fields indicate the following:	
	A0 10 03	The element number that is in alarm. The element number is a 12-bit number that consists of the lower nibble of the first byte and the 2nd byte eg. A(0) 10 (03) = 003.
	01 01	The first byte indicates the device and the 2nd byte the event. 01 device = (event from the free point, on loop). 01 event = (Point alarm).
9A 0D	Checksum	
03	End of text	

The event field distinguishes between alarms, troubles and other events. A typical trouble could be an event of 09 = (invalid reply failure, point on loop no longer responding to the exchange). Troubles cannot be searched for and must be interpreted by searching for AA and looking at the bytes in the fields at the 01 01 position

Appendix C.1.5. Remote control commands

Remote control commands are messages sent from supervision centers addressed to a specific exchange. Remote control commands can be found by searching for **08 01 10 03 AA** in the comms log. Remote control commands of interest are:

General Ack

A general ack is probably sent to an exchange to ack all alarms and troubles remotely. Note that ACKs will not reset the alarms in the FieldServer Data Arrays. Only resets will. A general ack can be found by searching for **08 01 10 03 AA 12**

```
02 07 49 00 07 08 01 10 03 AA 12 00 00 41 C2 03
```

Field	Description	
02	Start of text	
07	Databyte count = 7	
49	Message is from the supervision centre.	
00 07	Message is for exchange having an address of seven	
08 01 10 03 AA 12 00 00	Data bytes to be interpreted according to the protocol spec. In this case the following byte fields indicate the following:	
	12	Command code. 12 = GENERAL ACK
	00 00	Not used
41 C2	Checksum	
03	End of text	

General Reset

A general reset is sent to an exchange to have the exchange reset or clear all existing alarms and troubles. A general reset can be found by searching for **08 01 10 03 AA 13**

```
02 07 49 00 07 08 01 10 03 AA 13 00 00 81 93 03
```

Field	Description	
02	Start of text	
07	Databyte count = 7	
49	Message is from the supervision centre.	
00 07	Message is for exchange having an address of seven	
08 01 10 03 AA 13 00 00	Data bytes to be interpreted according to the protocol spec. In this case the following byte fields indicate the following:	
	13	Command code. 13 = GENERAL RESET
	00 00	Not used
81 93	Checksum	
03	End of text	

Appendix C.1.6. Reporting the result of remote control commands

After receiving a remote control command message from a supervision centre, an exchange responds with a status message to indicate the result of the remote control command. Remote control result messages can be found in a comms log by searching for **08 01 04 AA**

Note that result messages are sent immediately after receiving a remote control command and should always be found as the very next message after a remote control command message.

Remote control command results of interest are:

Command Accepted

Used to indicate the previous remote control command was accepted.

02 0A 29 00 07 **08 01 04 AA** 00 00 00 00 10 02 00 35 5F 03

Field	Description	
02	Start of text	
0A	Databyte count = 10	
29	Message is from the exchange.	
00 07	Message is from an exchange with an address of seven	
08 01 04 AA 00 00 00 00 10 02 00	Data bytes to be interpreted according to the protocol spec. In this case the following byte fields indicate the following:	
	10 02	Command result. 02 = COMMAND ACCEPTED (ignore the 10 which is an escape seq char).
	00	Not used
35 5F	Checksum	
03	End of text	

Command not performable

Used to indicate the previously received command cannot be performed.

02 0A 29 00 07 08 01 04 AA 00 00 00 00 10 03 00 35 60 03

The only change in format from the command accepted is that the command result is: 10 03 -> Command result. 03 = COMMAND NOT PERFORMABLE

Command with erroneous parameters

Used to indicate the previously received command had erroneous parameters.

02 0A 29 00 07 08 01 04 AA 00 00 00 00 04 00 35 61 03

The only change in format from the command accepted is that the command result is: 04 -> Command result. 04 = COMMAND WITH ERRONEOUS PARAMETERS

Appendix C.1.7. Reporting of operator actions on exchanges

Exchanges send messages that indicate when an operator performed some action at an exchange.

The following operator actions are of interest:

Global ack

Global acks can be found by searching for **09 32.** Note that ACKs will not reset the alarms in the FieldServer Data Arrays. Only resets will.

02 0F 29 00 07 08 01 0A AA 17 2D 68 2C 40 50 00 00 00 **09 32** F9 BB 03

Field	Description	
02	Start of text	
0F	Databyte count = 15	
29	Message is from the exchange.	
00 07	Message is from an exchange with an address of seven	
08 01 0A AA 17 2D 68 2C 40 50 00 00 00 09 32	Data bytes to be interpreted according to the protocol spec. In this case the following byte fields indicate the following:	
	09 32	Device 09 = operator, programming event. Event 32 = global ack.
F9 BB	Checksum	
03	End of text	

Global reset

Global resets can be found by searching for **09 35**

02 0F 29 00 07 08 01 0A AA 17 2D 68 32 40 50 00 00 00 **09 35** BB 7A 03

Field	Description	
02	Start of text	
0F	Databyte count = 15	
29	Message is from the exchange.	
00 07	Message is from an exchange with an address of seven	
08 01 0A AA 17 2D 68 32 40 50 00 00 00 09 35	Data bytes to be interpreted according to the protocol spec. In this case the following byte fields indicate the following:	
	09 35	Device 09 = operator, programming event. Event 35 = global reset,
BB 7A	Checksum	
03	End of text	

Appendix D. REFERENCE

Appendix D.1. Protocol Specification from which this Driver was developed

“CEI-ABI PROTOCOL FOR `EN.54 EXCHANGES, Models AM-6000, AM-2000, FSP-402”

It also contains a heading called: “Variation to the standard documentation of the protocol (CEI 79.5)”.

This Driver deviates from the following paragraph:

“Byte stuffing insertion methods: ….The added character (10h) is not counted in the data length field, but is used in the calculation of the checksum.”

This Driver does NOT use the added character (10h) in the calculation of the checksum.