FieldServer

FS-8700-144 Hochiki Packet Protocol Driver

Driver Manual

(Supplement to the FieldServer Instruction Manual)

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after July 2017.

Driver Version:	1.00
Document Revision:	1.B

Technical Support

Please call us for any technical support needs related to the FieldServer product.

Sierra Monitor Corporation
1991 Tarob Court
Milpitas, CA 95035

Website: www.sierramonitor.com

U.S. Support Information:

+1 408 262-6611
+1 800 727-4377

Email: support@sierramonitor.com

EMEA Support Information:

+44 2033 1813 41

Email: support.emea@sierramonitor.com

TABLE OF CONTENTS

1 HOCHIKI PACKET PROTOCOL DESCRIPTION

The Hochiki Serial driver allows the FieldServer to read data from Hochiki FireNET panels over RS-232. The FieldServer acts as a Client and reads the event log queue to record the status of the panel and various devices connected to the panel. The protocol used by the Fieldserver is independent of the user interface language of the panel.

This driver is not capable of emulating a Hochiki panel.

The Hochiki FireNET panel can be a standalone panel or part of network. Each Fire Alarm Panel on the Network is considered as a Node. Up to 64 Nodes can exist on one network.

The Hochiki panel logs the events as they occur. The FieldServer continuously reads the number of events in the log queue. When new events occur, they are read from the log via the PC (J5) port. The FieldServer parses and stores the data in Data Arrays. These Data Arrays can be monitored by third party tools.

Note that the FieldServer can be configured with a large number of points. The point limits purchased with the FieldServer prevent the entire database from being accessed in any one application. It is therefore strongly advisable to ensure that only the point addresses of interest are configured, and that the FieldServer is purchased with the correct point count.

Max Nodes Supported

FieldServer Mode	Nodes	Comments
Client	1	Only one Hochiki PC (J5) connection per port
Server	0	This driver cannot be configured as a Server

2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by Sierra Monitor

Part #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection
FS-8915-10	UTP cable (7 foot) for RS-232 use
FS-8917-18	RJ45 to DB9F connector adapter (FS-B3510)

2.2 Provided by the Supplier of 3rd Party Equipment

2.2.1 Required 3rd Party Hardware

Part #	Description
	Ethernet 10/100 BaseT switch[1]
X187/S187	Programming cable to connect via communication port J5 of the Hochiki panel

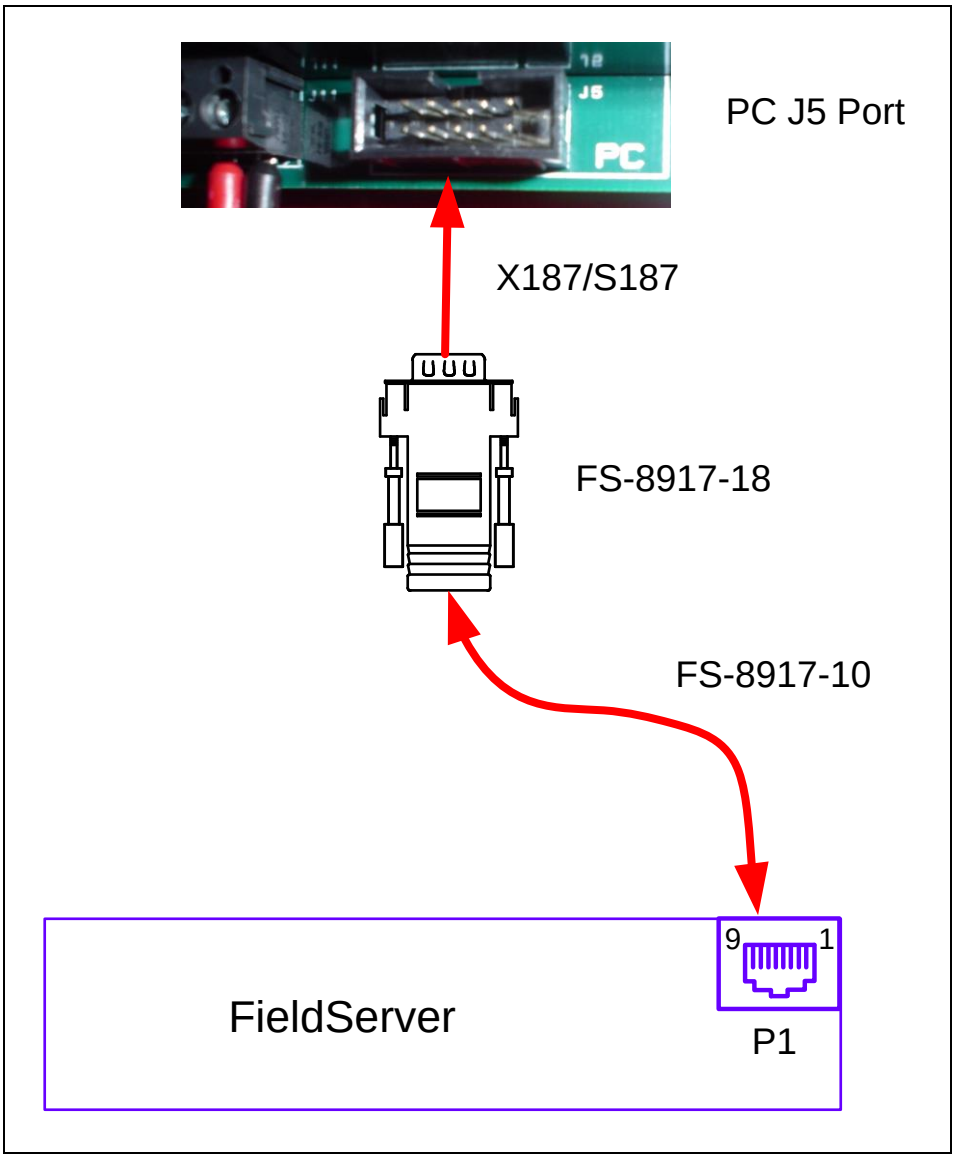
[1] Not all FieldServer models support 100BaseT. Consult the appropriate instruction manual for details of the Ethernet speed supported by specific hardware.

3 HARDWARE CONNECTIONS

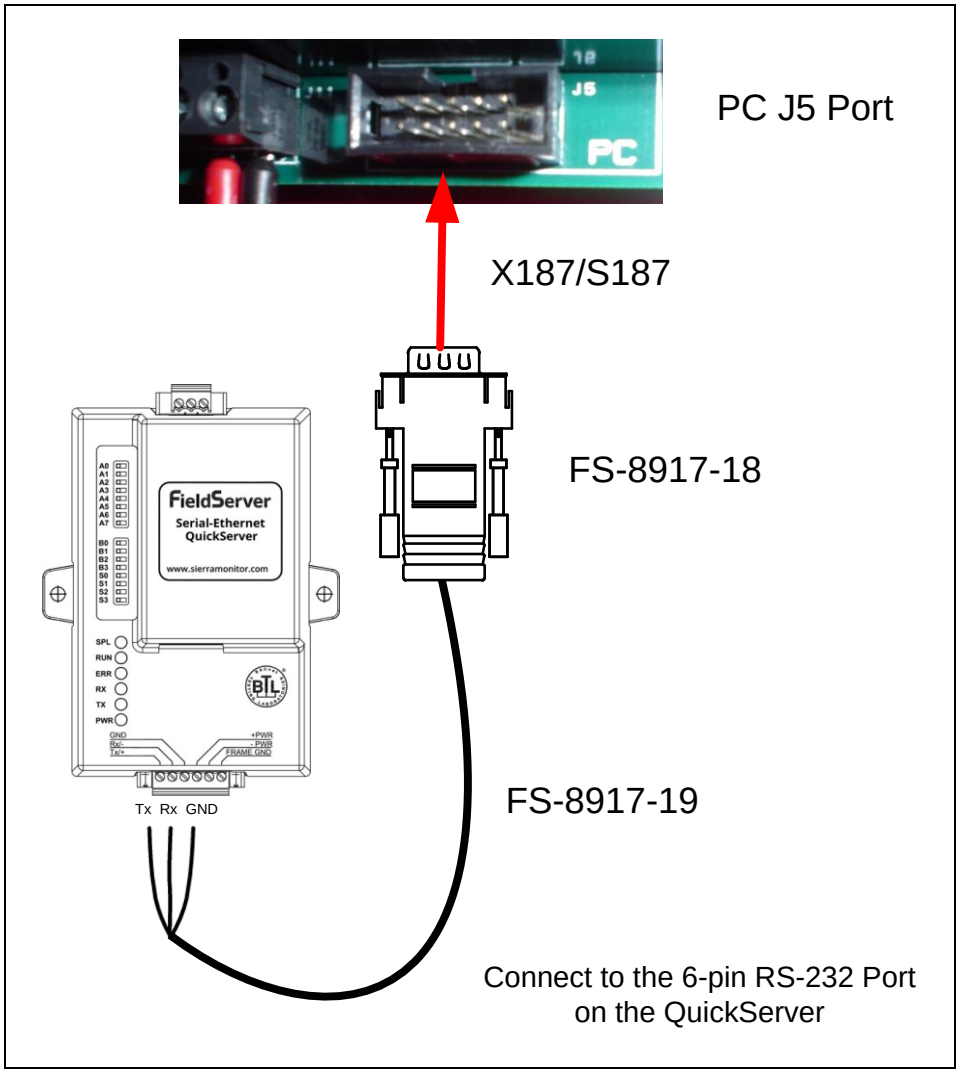
The FieldServer or QuickServer are connected to the Hochiki panel as shown in the connection drawings below.

Configure the Hochiki panel according to manufacturer's instructions.

3.1 FS-B3510



3.2 QuickServer



Connector Pinouts

RJ-45		
Wire Color	**Pin**	**Signal**
Brown	1	Rx
White/Orange	8	Tx
Blue/White	4	GND

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.	UInt16, 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_Array_Format    , Data_Array_Length
DA1_logEvents      , UInt16               , 2
DA1_Panel          , UInt16               , 11
DA1L1_E            , UInt16               , 32
DA1L1_T            , Byte                 , 32
DA1_NAC_E          , UInt16               , 4
DA1_NacT           , Byte                 , 4
DA1_IO_E           , UInt16               , 17
DA1_IO_T           , Byte                 , 17
DA1_T6_E           , UInt16               , 13
DA1_LedStatus      , Byte                 , 21
DA1_Version        , Byte                 , 10
```

5 CONFIGURING THE FIELDSERVER AS A HOCHIKI PACKET CLIENT

For detailed information on FieldServer configuration, 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 a Hochiki FireNet panel.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Hochiki panel 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: In the tables below, * indicates an optional parameter and bold legal values are 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-P2, R1-R2[2]
Protocol	Specify protocol used.	Hochiki_Packet, Hoc_Pkt, Hoc_Binary
Baud*	Specify baud rate.	**19200**
Parity*	Specify parity.	**None**
Data_Bits*	Specify data bits.	**8**
Stop_Bits*	Specify stop bits.	**1**

Example

```
//    Client Side Connections

Connections
Port    , Protocol          , Baud     , Parity
P1      , Hochiki_Packet    , 19200    , None
```

[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. R1 and R2 can be used only with RS485 to RS232 converter.

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	Provide the address of the Panel.	**1** – 64
Protocol	Specify Protocol used.	Hochiki_Packet, Hoc_Pkt, Hoc_Binary
Connection	Specify through which port the device is connected to the FieldServer.	P1-P2, R1-R2[3]

Example

```
//    Client Side Nodes

Nodes
Node_Name    , Node_ID,    , Protocol    , Connection
Hoc_1        , 1           , Hoc_Pkt     , 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_Client

[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.

5.3.2 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 **Section 5.2**
Data_Type	Data type for Passive_Client Map Descriptors.	Panel, SLC_Loop, Nac_Board, IO_Board, Others
Length	Length of Map Descriptor indicates the number of devices except in the case of panel related messages where it is the number of Data Array elements that will be used to store data.	1,2,3 …
Address*	Device address – offset into the data array where data will be stored for a particular device. Specify 0 to store loop events or troubles unrelated to any specific device on the loop.	**0**,1,2…
Max_Sub_Address*	Maximum number of sub addresses of device. See details in **Appendix A.1**.	-,1,2,3…
Loop*	If Data_Type is SLC_Loop, this parameter is used to specify the SLC loop number.	1,2,3, 4, **-**
Hoc_Address_Type*	Specify if Data_Type is 'Others'. See Appendix for details.	Any Integer value
Hoc_Task	Specify the task for RDBC Map Descriptors.	Poll_Event_Log, LED_STATUS, Panel_Version
DA_Byte_Name*	Specify the Data Array to be used to store the total number of troubles on the device.	One of the Data Array names from **Section 4**

5.3.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	≥0.001s

5.4 Map Descriptor Example – Read Event Log Queue

This Map Descriptor will read the number of events that exist in the log queue at the panel and then it will read new events one by one and store them on Passive_Client Map Descriptors as described in **Section 5.3** and **Section 5.6**.

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Hoc_Task        , Length   , Scan_Interval
Md_1_logEvents        , DA1_logEvents      , 0                   , Rdbc       , Hoc_01      , Poll_Event_Log  , 2        , 1s
```

The first offset in Data Array 'DA1_logEvents' will indicate the number of events in the panel's log queue and the next offset will indicate the last event number read by the Driver.

Continuously read data from the panel.

Name of the Node to which the FieldServer is connected. One of the Node Names declared in **Section 5.2**.

Specify the task name to read the number of events that exist in the log queue. The Driver will read new events if available.

Specify how often the driver should poll the panel to check for new events.

5.5 Map Descriptor Example – Store Panel Level Events

This Map Descriptor will be used to store Panel level events read by 'Md_1_logEvents' described in **Section 5.4**. Refer to **Appendix C.1** for information on the event types and storage structure in the Data Array.

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function        , Node_Name   , Data_Type   , Length
MD_1_Panel            , DA1_Panel          , 0                   , Passive_Client , Hoc_01      , Panel       , 11
```

One of the Data Arrays declared in **Section 4**.

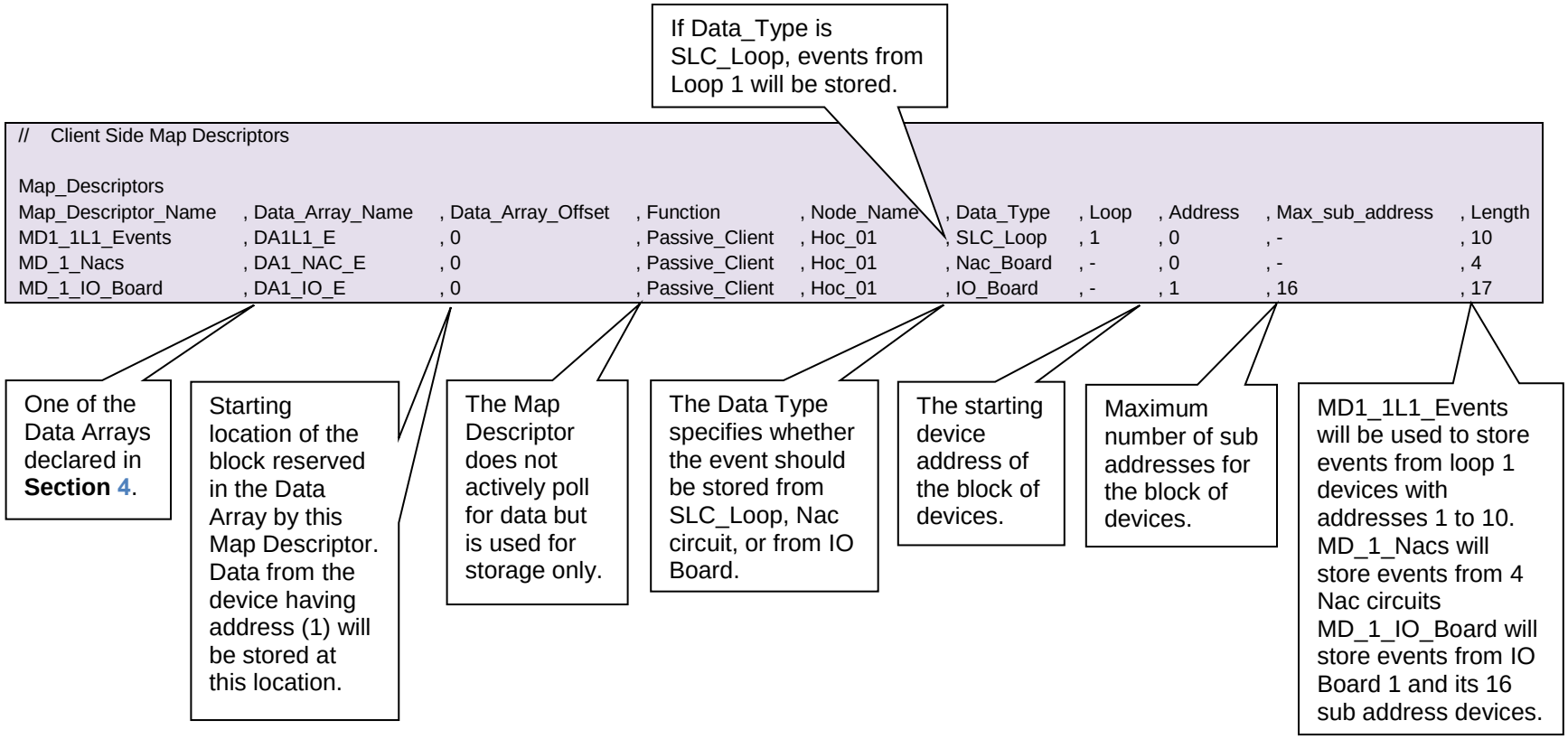
The Map Descriptor does not actively poll for data but is used for storage only.

The Data Type is 'Panel' to specify that it will be used to store Panel level events.

The number of data array elements that will be used to store data.

5.6 Map Descriptor Example – Store SLC Loop, Nac Board and IO Board Events

These Map Descriptors will be used to store SLC loop, Nac and IO Board events read by 'Md_1_logEvents' described in **Section 5.4**. Refer to **Appendix C.2** for details on the data value representing the type of events.



5.7 Map Descriptor Example – Read Panel LEDs Status

This Map Descriptor will read the panel LED statuses and store them in DA1_LedStatus. The value stored in the Data Array is 0 if the LED is OFF, 1 if the LED is ON and 2 if the LED is FLASHING. Refer to **Appendix C.4** for detail on the location of the relevant information for each LED in the Data Array.

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Hoc_Task        , Length   , Scan_Interval
HOC_1_LedStatus       , DA1_LedStatus      , 0                    , Rdbc       , Hoc_01       , LED_STATUS     , 21       , 4s
```

- One of the Data Arrays declared in **Section 4**.
- Continuously read data from panel.
- Node name of one of the nodes declared in **Section 5.2**.
- Specify the task name to read the panel LED statuses.
- The Hochiki panel always sends the statuses of all 20 LEDs.
- Specify how often the driver should read LED statuses.

5.8 Map Descriptor Example – Read Panel Version Information

This Map Descriptor will read panel version information and will store it in DA1_Version. This information could be useful for troubleshooting.

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name   , Data_Array_Name   , Data_Array_Offset   , Function   , Node_Name   , Hoc_Task        , Length   , Scan_Interval
HOC_1_Panel_Version   , DA1_Version        0                      , Rdbc       , Hoc_01       , Panel_Version  , 10       , 60s
```

- One of the Data Arrays declared in **Section 4**.
- Continuously read data from panel.
- Node name of one of the nodes declared in **Section 5.2**.
- Specify the task name to read panel version information.
- Specify how often the driver should poll for the panel version.

APPENDIX A. USEFUL FEATURES

Appendix A.1. Use of Data_Type “Others”

The Driver uses the Data_Type parameter in Passive_Client Map Descriptors to determine the location to store data for the known Address_Types (0. 2, 3, 4, 7, 8, 9). If a message is received from another Address_Type, the Driver will not be able to store the data and will print a message which identifies the Address_Type. The Hoc_Address_Type parameter can be used in conjunction with the “Others” Data_Type parameter to store this data.

In the following example a Passive_Client Map Descriptor is created to store data for events with Address_Type 6.

Appendix A.1.1. Map Descriptor Example – Use of Data Type “Others”

```
//    Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name  , Data_Array_Name  , Data_Array_Offset  , Function        , Node_Name  , Data_Type  , Hoc_Address_Type  , Address  , Length
MD_1_Type6s          , DA1_T6_E          , 0                  , Passive_Client  , Hoc_01     , Others     , 6                 , 0        , 13
```

- One of the Data Arrays declared in **Section 4**.
- Map Descriptor will not poll but will be used to store data returned by poll of another Map Descriptor.
- Node name of one of the nodes declared in **Section 5.2**.
- The Data Type to specify that this Map Descriptor will store data for unknown Data types.
- The starting device address of block of devices.
- Number of devices in this block.

Appendix A.2. Use of MX_SUB_ADDRESS Parameter

Most devices have only 1 address; e.g. there are 10 detectors on Loop1 and addresses 1- 10 are assigned to these detectors. Some devices, e.g. DIMM (Dual Input Monitor Module) have multiple inputs. In this case, the device will have 1 address (say address 15) but will also have 2 sub-addresses (15.1 and 15.2).

The MX_Sub_Address parameter is used to capture events from the device sub-addresses. In this case the length of the Map Descriptor should be 3 because there are actually 3 addresses for device 15 (15.0, 15.1 and 15.2). Events may be generated for each of the addresses; for example, if the DIMM is disconnected from the loop, the address reported will be 15.0. If there is an alarm on input 1, the address reported will be 15.1.

To avoid creating individual Map Descriptors for each device having sub-address, it is possible to group devices with contiguous addresses. If devices in the group do not have the same number of sub-addresses, use the biggest sub-address number for the MX_Sub_Address parameter. The driver will reserve the same number (mx_sub_adress+1) of memory locations in the Data Array for each device in the group; note however, that some may not be used.

Appendix A.3. Data Synchronization

The Hochiki panel logs event as they occur up to a maximum of 500 events after which the oldest event is overwritten. When the panel is rebooted, it clears the log and logs the first event as “System Initialize”.

When the FieldServer is connected to the panel, it will read the first event in the log. If this is a “System Initialize” event, the FieldServer will read all the events and synchronize its data with the panel.

If the first event is not “System Initialize” it means that the log queue has been overwritten. In this case the FieldServer will not read existing events, but will read only new events. If panel is in normal state (i.e. there is no active event (Fire, Troubles, etc) on the panel), then the FieldServer and panel are synchronized.

If there are active events when the FieldServer is connected, the FieldServer will not synchronize with the panel until the events have been cleared. This can be done by recycling power to the panel.

NOTE: Power to the Hochiki panel should not be removed if there are active fire alarm or other emergency events present! The emergency situation must be cleared by authorized personnel prior to connecting the FieldServer.

APPENDIX B. TROUBLESHOOTING

Appendix B.1. Error Messages

Error #	Msg Screen	Screen message	Meaning	Suggested Solution
HOC_PKT#01	DRIVER	Connection params are %d %d %d Even/Odd/None Timeout=%0.3fs	Non-Standard connection parameters are used.	Correct connection parameters in configuration file if they are not as expected.
DRV->HOC_PKT#02:	ERROR	Write-thru not Possible. MD <%s>. See Driver Manual for details.	Driver will not write data to Hochiki panel.	Make sure other protocol does not write to Data Array locations that are being used by specified Map Descriptor.
HOC_PKT#03	DRIVER	Unknown escape sequence 0x%X, assuming it as 0x01. See Driver Manual for details.	There are only two (0x00 and 0x01) known escape sequences at this time. If the driver receives an unknown escape sequence it will consider it as 0x01 and proceed to process the message.	In case there is no following error message, it could be ignored as long as data is still in synchronization with panel. If this message gets printed more than once, take a FS Toolbox capture and contact technical support.
HOC_PKT#04	DRIVER	Discarding short message. Got %d bytes, expected=%d bytes. See Driver Manual for details.	Received message is too short to process. Driver will ignore it.	Take a FS Toolbox capture and contact technical support.
HOC_PKT#05	DRIVER	1st event is not 'System Initialize', so assuming panel is in Normal State. See Driver Manual for details.	Panel's event log queue is overwritten; Driver will read only new events. Refer to **Appendix A.3**.	If panel is in normal state, then ignore this message. Otherwise to synchronize FieldServer and panel, either recycle power to the panel or turn OFF FieldServer until the panel is in normal state.
HOC_PKT#06	DRIVER	ERR. MD<%s> length is %d, reqd %d. See Driver Manual for details.	More data is available for this Map Descriptor.	Edit configuration file to increase the length of this Map Descriptor.
HOC_PKT#07	DRIVER	Transmission Failed on %s MD, timeout %dms	Data failed to be transmitted to the panel before the end of the timeout period.	Take a FS Toolbox capture and contact technical support.

Error #	Msg Screen	Screen message	Meaning	Suggested Solution
HOC_PKT#08	DRIVER	Discarding unsupported pkttype=0x%X. See Driver Manual for details.	Received packet type is unknown. Driver will ignore it.	Take a FS Toolbox capture and contact technical support.
HOC_PKT#09	DRIVER	Discarding too long packet of %d chars. See Driver Manual for details.	Received packet is too long to process. Driver will ignore it.	Take a FS Toolbox capture and contact technical support.
HOC_PKT#10	DRIVER	System Initialising...	Driver is clearing all its Data Arrays upon receipt of 'System Initializing'. See **Appendix A.3**.	This message is for information only, no action required.
HOC_PKT#10a	DRIVER	System Initialising completed in %lu ms.	Number of milliseconds taken to clear all of Hochiki Data Arrays.	This message is for information only, no action required.
HOC_PKT#11	DRIVER	Discarding packet with unknown Event Type<%d>. See Driver Manual for details.	Received packet contains unknown event type. Driver will ignore it.	Take a FS Toolbox capture and contact technical support.
HOC_PKT#11a	DRIVER	Discarding packet with unknown Event Code <%d> See Driver Manual for details.	Received packet contains unknown event code. Driver will ignore it.	Take a FS Toolbox capture and contact technical support.
HOC_PKT#11b	DRIVER	Discarding Packet with unknown address_type=%d , event_type=%d=<%s >=%d Node=%d. See Driver Manual for details.	Received packet contains unknown address type. Driver will ignore it.	Take a FS Toolbox capture and contact technical support.
HOC_PKT#12	DRIVER	No MD found for ND=%d event_type=%d=%s clr=%d Data_Type=%s ... Addr_type=%d Addr=%d sb_adr=%d LP=%d event_code=%d. See Driver Manual for details.	Driver got a message but there is no suitable Map Descriptor to store data.	Edit configuration file to add suitable Map Descriptor to store this data. If Data_Type specified in this is "* Illegal *", then Data_Type for new Map Descriptor should be "Others"
HOC_PKT#13	DRIVER	Decrementing Panel event=%d=%s value < 0, setting to 0 MD<%s> offset=%d. See Driver Manual for details	Driver got CLEAR message but the value at mapping memory location was already 0. It should happen only when FieldServer and Panel are not synchronized.	Follow procedure in **Appendix A.3** to synchronize the device.

Appendix B.2. Capturing Events with Address_Type 6

Address Type 6 is not one of the known address types and therefore has no allocated Data_Type. During testing it was found that the Hochiki panel logs events for Address Type 6 when a user presses/releases any button on the panel. The following errors were generated.

```
HOCHIKIP : Ignoring unknown Event Type
HOCHIKIP# : Ignoring unknown Event Code
```

A programmable input may be configured on the panel which produces an event with Address Type 6 that is important to capture, and the Data_Type “Others” can be configured to store this data. Refer to **Appendix A.1**. **Appendix C.3** lists the devices and their addresses which produce an event with Address_Type 6.

Appendix B.3. Zone Status

If the Zone_Status data array is used, the status for Supervisory and Trouble events is not updated until there are at least two of such events in a zone. For example, a single Supervisory event in zone 1 will not be indicated in the array, but when two Supervisory events are present in Zone 1, the array will indicate two Supervisory events. This is a bug in the FireNET firmware and may be addressed in the future. If such information is needed, it may be obtained by using one of the other Data Arrays.

Note that only Alarm (1), Supervisory (2), Trouble (4) and Disable (8) information is available when using the Zone_Status array.

APPENDIX C. REFERENCE

Appendix C.1. Panel Level Events Storage Structure

Panel level events include System Initializing, Low Battery Voltage, Battery Disconnected, etc. The number of events of each type is stored in the Data Array as an integer value at the offset reflected in the following table.

Offset Into Data Array	Event Type
0	Fire
1	Emergency
2	Auxiliary
3	Pre-Alarm
4	Supervisory
5	Fault (Trouble)
6	Security
7	Disable
8	Testing
9	Status
10	CEAction

Appendix C.2. Data Array Value for Events from SLC Loop Device, Nac Circuit or IO Board Device

The Driver will store a numeric value in the Data Array depending upon the event type. The value and the corresponding event are reflected in the following table.

DA Value	Event Type
0x0001	Fire
0x0002	Emergency
0x0004	Auxiliary
0x0008	Pre-Alarm
0x0010	Supervisory
0x0020	Fault (Trouble)
0x0040	Security
0x0080	Disable
0x0100	Testing
0x0200	Status
0x0400	CEAction

Where more than one event type exists the driver will store a single numeric value that represents the sum of the DA Values of the two event types.

For example if a device is in Fire and Trouble, the DA value will be 0x0021.

Appendix C.3. Devices Address Reporting Events with Address Type 6

The Driver will store a numeric value in the Data Array depending upon the type of event.

Address	Device (Button or input) Description
0	TBL Input
1	RES Input
2	INT Input
3	CNT Input
4	SIL Input
5	Prog Input 1
6	Prog Input 2
7	Prog Input 3
8	Fire Drill Button
9	Prog Function Button
10	Reset Button
11	Alarm Silence Button
12	Alarm Resound Button

Appendix C.4. LED Status Description and Storing structure

The following table details the location of the status information for each LED in the Data Array.

LED Number (relative location in DA)	LED Description (Value 0=Off, 1=ON, 2=Flashing)
0	Summary (1 If any Led is ON except Power Led)
1	Fire
2	Trouble
3	General Trouble
4	Nac Trouble
5	Supervisory
6	System Trouble
7	On Test
8	Panel Sounder Silenced
9	More Events
10	Fire Output Active
11	Delay Active
12	Pre-Alarm
13	Not Used
14	Not Used
15	Not Used
16	Not Used
17	Not Used
18	Not Used
19	Point Bypassed
20	AC Power On