



---

A Sierra Monitor Company

**Driver Manual**  
**(Supplement to the FieldServer Instruction Manual)**

**FS-8700-110 FCI 7200 Series Serial Driver**

**APPLICABILITY & EFFECTIVITY**

**Effective for all systems manufactured after May 1, 2001**

## **TABLE OF CONTENTS**

<b>1. FCI 7200 Series Serial Driver Description .....</b>	<b>3</b>
<b>2. Driver Scope of Supply .....</b>	<b>4</b>
2.1. Supplied by FieldServer Technologies for this driver .....	4
2.2. Provided by the Supplier of 3 <sup>rd</sup> Party Equipment .....	4
2.2.1. <i>Required 3<sup>rd</sup> Party Hardware</i> .....	4
2.2.2. <i>Required 3<sup>rd</sup> Party Software</i> .....	4
2.2.3. <i>Required 3<sup>rd</sup> Party Configuration</i> .....	4
2.2.4. <i>Optional Items</i> .....	4
<b>3. Hardware Connections .....</b>	<b>5</b>
<b>4. Configuring the FieldServer as a FCI 7200 Series Serial Driver Client .....</b>	<b>7</b>
4.1. Data Arrays/Descriptors .....	7
4.2. Client Side Connection Descriptions .....	8
4.3. Client Side Node Descriptors .....	9
4.4. Client Side Map Descriptors .....	9
4.4.1. <i>FieldServer Related Map Descriptor Parameters</i> .....	9
4.4.2. <i>Driver Related Map Descriptor Parameters</i> .....	10
4.4.3. <i>Timing Parameters</i> .....	11
4.5. Map Descriptor Example 1 – Panel Events .....	12
4.6. Map Descriptor Example 2 – Sensor / Module Events .....	13
4.7. Map Descriptor Example 3 – Bit Storage .....	14
<b>5. Configuring the FieldServer as a FCI 7200 Series Serial Driver Server .....</b>	<b>15</b>
<b>Appendix A. Advanced Topics .....</b>	<b>16</b>
Appendix A.1. Events and Event Categories .....	16
Appendix A.2. Extending the Event Table .....	17
Appendix A.3. How Data is stored .....	17
Appendix A.4. Panel Synchronization .....	21
Appendix A.5. What happens when the panel sends a Reset Messages .....	21
<b>Appendix B. Revision History .....</b>	<b>22</b>

## 1. FCI 7200 Series Serial Driver Description

The FCI 7200 Series System Control Units (SCU) are manufactured by Fire Control Instruments. A SCU with an enabled serial port can transmit data to a FieldServer which can, in turn, make the data available to other devices including those which communicate using different protocols (e.g. BACnet)

This is a passive Client driver which does not poll for data, nor does it send data or commands to the SCU. Messages received from the SCU are ignored or stored on the FieldServer depending on the status of the panel. The method of message processing and location on the FieldServer is determined in the FieldServer configuration file. Once stored in the FieldServer the data is available to be read or written using other protocols such as BACnet.

No automatic panel data synchronization technique exists. The data in the FieldServer and the panel status have to be synchronized manually. This typically requires a panel reset.

Since the driver cannot send data or commands to the SCU it cannot be used to acknowledge, silence or reset alarms and other events.

The driver can process the single line messages sent from SCU firmware versions earlier than 2.20 and 3 line messages produced in firmware versions 2.20 and later. Processing of 3 line messages requires the 20 character System ID label to be defined.

The driver provides both client and server emulation. The server side of the driver is intended to support FieldServer's Quality Assurance program and is not intended to provide complete emulation of a SCU and is thus not fully documented. Should you require the Server side functionality to be documented and enhanced, please contact FieldServer's sales group.

### Max Nodes Supported

FieldServer Mode	Nodes	Comments
Client	1	1 Node per serial port
Server	1	1 Node per serial port

**2. Driver Scope of Supply**

1.01 Supplied by FieldServer Technologies for this driver

<b>FieldServer Technologies PART #</b>	<b>Description</b>
FS-8915-10	UTP cable (7 foot) for Ethernet connection
23069	RJ45-RJ11/12 Cable assembly for FS connection to FCI panel.
-FS-8700-110	Driver Manual.

1.02 Provided by the Supplier of 3<sup>rd</sup> Party Equipment

**2.1.1. Required 3<sup>rd</sup> Party Hardware**

FCI Panel must be equipped with a RS-232 Serial Printer Port.

**2.1.2. Required 3<sup>rd</sup> Party Software**

None

**2.1.3. Required 3<sup>rd</sup> Party Configuration**

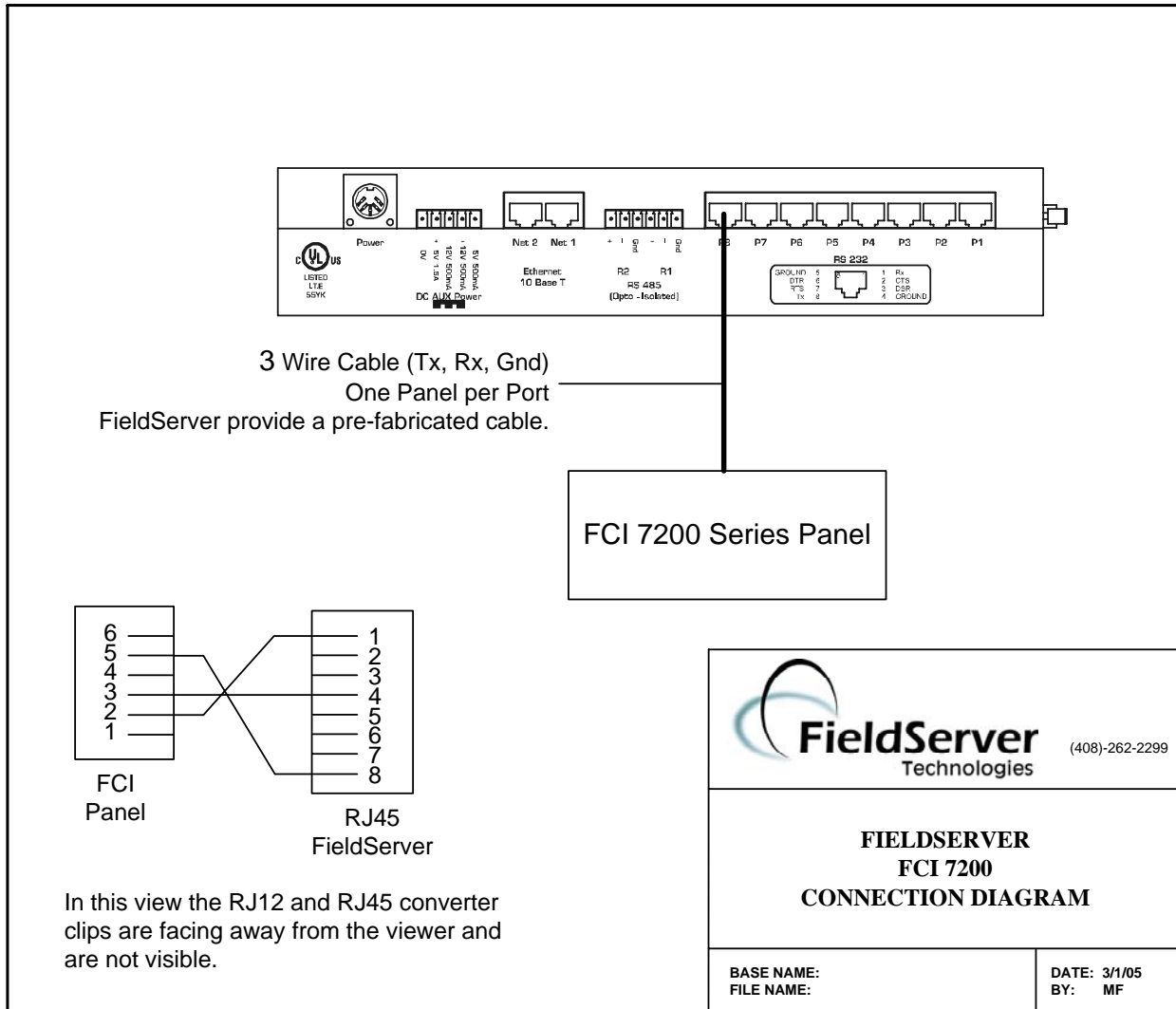
None known.

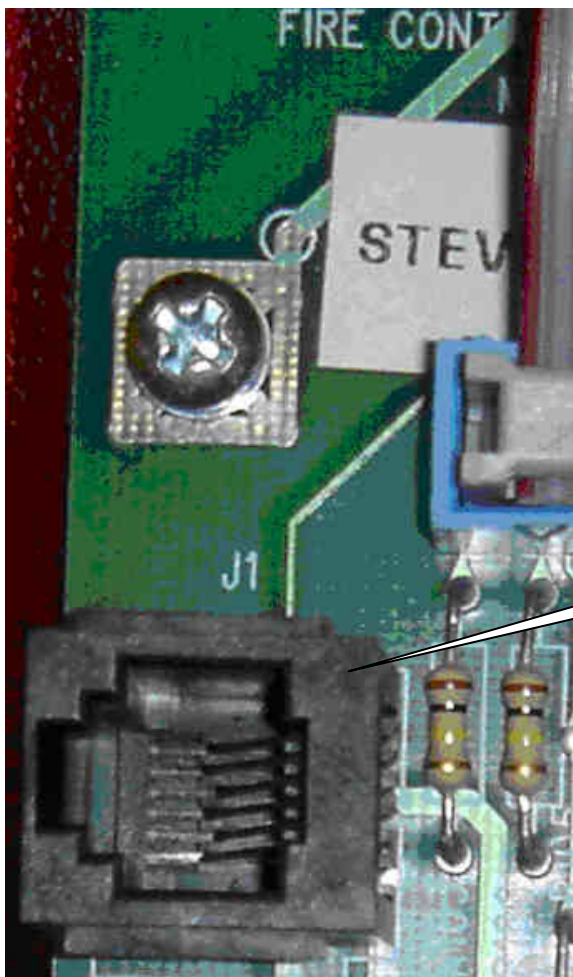
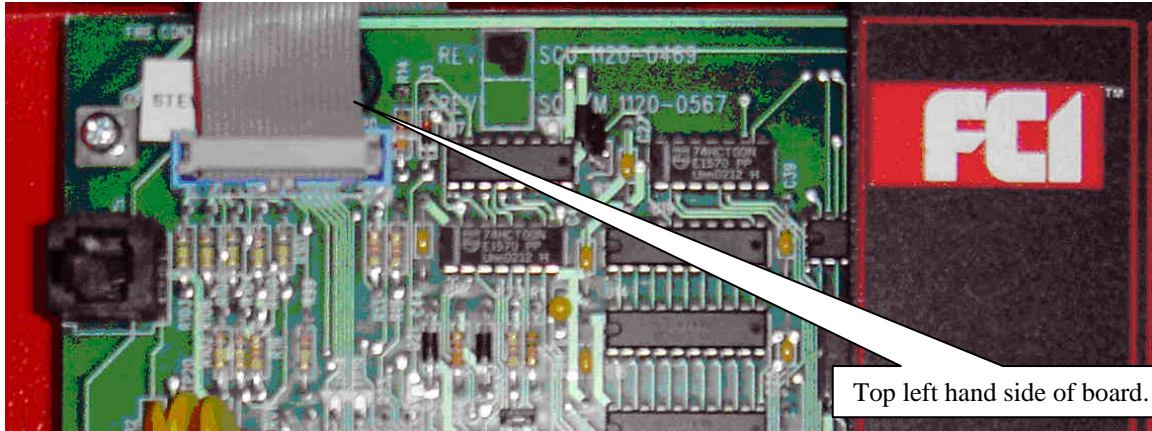
**2.1.4. Optional Items**

None.

### 3. Hardware Connections

The FieldServer is connected to the FCI panel as shown in the connection drawing.





#### 4. Configuring the FieldServer as a FCI 7200 Series Serial 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 a FCI 7200 Series SCU.

##### 1.03 Data Arrays/Descriptors

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for FCI 7200 Series Serial 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.

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.	Float, Bit, UInt16, 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

**Example**

```
// Data Arrays

Data_Arrays
Data_Array_Name,      Data_Array_Format,  Data_Array_Length
DA_AI_01,            UInt16,             200
DA_AO_01,            UInt16,             200
DA_DI_01,            Bit,                200
DA_DO_01,            Bit,                200
```

1.04 Client Side Connection Descriptions

Section Title		
Connections		
Column Title	Function	Legal Values
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2 <sup>1</sup>
Protocol	Specify protocol used	FCI_7200, FCI_7200_series, FCI7200
Baud*	Specify baud rate	<b>1200</b> – Vendor Limitation.
Parity*	Specify parity	<b>None</b> – Vendor Limitation
Data_Bits*	Specify data bits	<b>8</b> – Vendor Limitation
Stop_Bits*	Specify stop bits	<b>1</b> – Vendor Limitation
Handshaking*	Specify hardware handshaking  The 7200 series panels do not support hand shaking.	<b>None</b>
Poll _Delay*	This commonly used parameter has no meaning for this driver.	

**Example**

```
// Client Side Connections

Connections
Port,      Protocol,      Baud,  Parity
P8,        FCI_7200,      1200   None
```

<sup>1</sup> Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

1.05 Client Side Node Descriptors

Section Title		
Nodes		
Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	This commonly used parameter has no direct meaning for this driver.	The protocol is node-less. This means the messages do not contain information identifying a unique source and/or destination node address. For this reason this parameter need not be specified and no more than one SCU can be connected to a single FieldServer serial port.
Protocol	Specify protocol used	FCI_7200, FCI_7200_series, FCI7200
Connection	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2 <sup>2</sup>

**Example**

```
// Client Side Nodes

Nodes
Node_Name,      Protocol,      Connection
Panel-01,      FCI_7200,      P8
```

1.06 Client Side Map Descriptors

**4.1.1. 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 names from Section 1.03
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in Section 1.03
Function	Function of Client	Server, Passive_Client

<sup>2</sup> Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

	Map Descriptor.	
--	-----------------	--

**4.1.2. 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 1.05
Event Type	<p>This driver uses this parameter to determine the suitability of a Map Descriptor when looking for a location to store data from an incoming message.</p> <p>A Map Descriptor may be defined to store only 'Alarm', 'Fault', 'Trouble' or 'Other events. Alternatively, specify "Any"</p> <p>A table of events vs. categories is provided in Appendix A.1</p>	Any, Other, Fault, Alarm, Trouble
Point Type	This driver uses this parameter to determine the suitability of a Map Descriptor when looking for a location to store data from an incoming message.	Zone, Relay, Loop, Sensor, Module, Panel
Relay/Loop/Zone Number	<p>Ignored when the Point Type is 'Panel'</p> <p>Point Type = Relay 1..4</p> <p>Point Type = Zone 1..8</p> <p>Point Type = Loop 1..2</p> <p>Point Type = Module 1..2</p> <p>Point Type = Sensor 1..2</p>	Whole Numbers 1, 2 , etc
Length	Each Map Descriptor defines storage locations for a series of addresses. This parameter specifies the length of the series.	1,2,3 .etc Whole numbers
Address	This parameter is only considered for those Map Descriptors whose 'Event Type' is Module or Sensor. It specifies the starting module or sensor number. The length parameter the determines the range of the sensor/module numbers	1..99
Store As	<p>By default the driver stores using the 'Index Value' Method.</p> <p>Set this parameter to 'Bit' to have the driver use the primary Data Array to</p>	Bit, Index_Value

	store using the 'Bit Storage' Method. These methods are described in Appendix A.3	
DA_Bit_Name	If the default 'Store As' is specified or if the parameter is omitted then you can specify a secondary array using this parameter - the driver will store event data as 'Bit Storage' in the secondary array (and as 'Index Values' in the primary array.) These methods are described in Appendix A.3	One of the Data Array names from "Data Array" section above

**4.1.3. Timing Parameters**

Column Title	Function	Legal Values
Scan_Interval	This commonly used parameter has no meaning for this driver as the driver does not poll but listens passively for unsolicited messages from the panel.	

1.07 Map Descriptor Example 1 – Panel Events

In this example a map descriptor defines storage locations for messages sent by the panel containing panel events. A panel event is one that is not a relay, loop, zone, sensor or module. Includes events such as battery failure, resets, silences etc. As these messages do not contain any address information all data will be stored at a single location and the length should be 1. Each time a new event is received the driver stores a value indicating the event cause.

**Example:**

FAULT: AC Power SCU 0:00:04 1/01/92

```
// Client Side Map Descriptors
Map_Descriptors
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, Node_Name, Event Type, Point Type, Address, Length, Clear_on_reset
PanelData, DA_PANEL, 0, Passive_Client, Panel-01, Any, Panel, -, 1, Yes
```

It is recommended that you allocate unique MD names.

Tell the driver the Data Array name and starting location that data should be stored.

The driver listens passively for messages from the Panel. It cannot poll for data.

The name of the Node defined in the Node Descriptor section.

The driver will not apply a filter to incoming events. Any event qualifies for storage.

Only 'Panel' events will be stored using this MD.

The driver will clear the 1(=Length) element of the DA called DA\_Panel starting at offset=0 when a Reset message is received.

1.08 Map Descriptor Example 2 – Sensor / Module Events

If messages from Loop 1, Module 1 to 99 are received then the MD in this example will be used for storage. If you have modules on more than one loop then you will need one MD for each loop. In this example the event type was set to 'Alarm'. This means that only 'Alarm' events will be stored using this MD. This could be useful if you are only interested in one category of events. If you want all events stored then change the 'Event Type' to 'Any'.

**Example:**

EEPROM BAD: QZUb L1M01 << Chief's Office >> 5:24 3/03/93

```
// Client Side Map Descriptors
Map_Descriptors
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, Function, Node_Name, Event Type, Point Type, Relay/Loop/Zone Number, Address, Length, Clear_on_reset
ModuleData1, DA_MODULE, 0, Passive_Client, Panel-01, Alarm, Module, 1, 1, 99, Yes
```

In this example, only Alarm events will be stored.

Messages reporting other events will be ignored unless other MD's are defined.

Change this to 'Sensor' for sensors.

The address specifies the starting Module number and the Length tells the driver the range of Modules.

In this example, Module 1 to 99

1.09 Map Descriptor Example 3 – Bit Storage

This example defines storage location for Relay Point events. The example would work for all other point types. In the example, both primary and secondary storage Data Arrays have been specified. The driver stores index values in the primary array. Each new event for a particular relay will overwrite the value stored previously. In the Bit Array, the driver sets the bit corresponding to the event, leaving other bits unchanged – thus the 2ndary storage can be used to determine if more than one event is active at a time.

```
// Client Side Map Descriptors
Map_Descriptors
Map_Descriptor_Name, Data_Array_Name, Data_Array_Offset, DA_Bit_Name, Function, Node_Name, Event Type, Point Type, Relay/Loop/Zone Number, Address, Length, Clear_on_reset
RelayData, DA_RELAY, 0, DB_Relay, Passive_Client, Panel-01, Any, Relay, 1, -, 4, Yes
```

Data\_Array\_Name is where the primary DA is specified. Index values are stored here.

DA\_Bit\_Name is where secondary storage is defined. Events are stored by setting appropriate bits.

Remember that 2 elements per Relay, Module, Sensor, Loop are used.

MD's for storing Relay, Loop, Zone and Panel do not need the address specified.

## 5. Configuring the FieldServer as a FCI 7200 Series Serial Driver Server

The server side of the driver is intended to support FieldServer's Quality Assurance program and is not intended to provide complete emulation of a SCU and is thus not fully documented. Should you require the Server side functionality to be documented and enhanced, please contact FieldServer's sales group.

**Appendix A. Advanced Topics**

Appendix A.1. Events and Event Categories

The driver reports the event cause using the matching index value. There are 4 event categories:

- 1 = Other
- 2 = Fault
- 3 = Alarm
- 4 = Trouble

The message category must match the 'Event Type' parameter specified on a Map Descriptor before that Map Descriptor can be considered for storage of the message data.

Index	Category	Event
1	2	"Fault"
2	1	"Short"
3	1	"Disconnect"
4	1	"Comm Fault"
5	1	"Config Err"
6	1	"Eeprom Bad"
7	1	"Reset"
8	1	"Silence"
9	1	"Cross Zone"
10	1	"Acknwldgd"
11	1	"Walk Test"
12	1	"Alarm Test"
13	1	"SPVSN Test"
14	1	"Fault Test"
15	1	"Fire Drill"
16	1	"Batt Test"
17	1	"PRGM Mode"
18	1	"Action"
19	1	"Loop Break"
20	3	"Alarm"
21	1	"P.A.S."
22	1	"Off-Normal"
23	1	"RZA Fault"
24	1	"Verify"
25	1	"CM SHort"
26	1	"Test Fail"
27	1	"Alert"
28	1	"Dirty"
29	1	"Very Dirty"
30	1	"Missing"
31	1	"Wrong Type"
32	1	"Extra Addr"
33	1	"Clock Err"
34	4	"Trouble"

35	1	"MLT Events"
36	1	"Alrm Ackd"

Appendix A.2. Extending the Event Table

New event causes may be added to the Event Table and the index value or category of existing event causes modified by adding a section to the configuration CSV file. The examples below illustrate this:

Example 1: Index value of 'Trouble' is updated to a new value of 100

Driver_Table	Event_Type_Description,	Event_Type_Index_Value,	Event_Type_Category ,	Protocol
TROUBLE,	100,	4,	FCI_7200	

Example 2 : New Entry is added

Since it has been added as category=3, only MD's with 'Event Type' set to Alarm or ANY will capture messages with this event description

Driver_Table	Event_Type_Description,	Event_Type_Index_Value,	Event_Type_Category,	Protocol
DESTROYED,	51,	3,	FCI_7200	

For categories use the following values

- 'Other' = 1
- 'Fault' = 2
- 'Alarm' = 3
- 'Trouble' = 4

Appendix A.3. How Data is stored

All messages less than 102 characters long are discarded. All other messages are processed as follows:

- The driver determines if the message is a Zone, Relay, Loop, Sensor, Module or Panel message.
- The driver finds all Map Descriptors with matching 'Point Type' parameters.
- The event category is determined.
- Map Descriptor selection is refined based on whether the 'Event Type' matches or has been defined as "Any":
- The driver determines the Loop, Relay, Zone, Sensor and Module numbers from the message and refines its selection of Map Descriptors by selecting those that match the values determined from the message.
- The selected Map Descriptors are now used to determine a data array and offset at which to store the data.

- Finally the driver checks the 'Store As' parameter. If it hasn't been specified then 'Index Value' storage is assumed. If it has been specified as 'Bits' then the driver will perform 'Bit Storage'. In cases where the Map Descriptor has both a primary and secondary Data Array, the driver will use 'Index Value' storage using the primary data array and 'Bit Storage' using the secondary array.

**Example:**

The following fragment is part of a Map Descriptor definition; some parameters have been omitted for the purposes of clarity.

Map_Descriptors								
Data_Array_Name,	Data_Array_Offset,	Event Type,	Point Type,	Relay/Loop/Zone Number,	Address,	Length,	Clear_on_reset,	DA_Bit_Name
DA_MODU	0,	ANY,	Module,	1,	1,	99,	Yes,	DB_MODU
DA_MODU_A,	0,	ALARM,	Module,	1,	1,	99,	Yes,	DB_MODU_A
DA_MODU_F,	0,	FAULT,	Module,	1,	1,	99,	Yes,	DB_MODU_F
DA_MODU_T,	0,	TROUBLE,	Module,	1,	1,	99,	Yes,	DB_MODU_T
DA_MPODU_O,	0,	OTHER,	Module,	1,	1,	99,	Yes,	DB_MODU_O

Message = "FAULT: AC Power SCU 0:00:04 1/01/92"

- This message does not report the status of a Zone, Relay, Loop, Sensor or Module and is therefore assumed to be a panel message. Since there is no MD with "Point Type" Panel, the message is ignored.

Message = "TROUBLE: QZUb L1M22 << Chief's Office >> 5:24:00 3/03/93"

- This message reports status for Loop 1 Module 22. Since all the MD's in the example have a 'Point Type'='Module', they are all considered for storage.
- The driver looks in the Event Table and finds it has an index value of 34 and a category of 4 (Trouble). Only the MD's with "Event Type" set to "Any" and "Trouble" are now considered.
- Since the value of the 'Relay/Loop/Zone' parameter matches the Loop number in the message, these MD's remain in contention.
- The Module number of 22 is compared with the MD's Address and Length Parameters. The Address is the starting number and the length defines the range. Both MD's have addresses of 1 and length of 99 and thus both are still selected because the Module of 22 falls in this range.
- The driver calculates an offset based on the offset specified in the MD and the Module number relative to the MD's address:

MD Offset = 0  
 MD Address = 1  
 Message Module = 22

Module 1's data is stored at offset 0 and hence Module 22's data will be stored at offset 21. The driver stores the value 34 at offset 21 overwriting any data previously stored at that location. This is 'Index Value' Storage.

- Secondary storage has been defined using the 'DA\_Bit\_Name' Data Array. The driver doubles the offset as two locations are used for each address. Then the driver reads the value found in the Data\_Array, modifies it and writes it back. As the index value is 34 the driver modifies the 34<sup>th</sup> bit – or expressed another way, the driver modifies the 2<sup>nd</sup> bit (34-32) at offset+1.
- Thus, driver calculates the offset for Bit Storage as  $2 \times 21 = 42$ . The driver sees that bit 34 is 2<sup>nd</sup> bit in the next offset and so the driver reads DB\_MODU:43, modifies the value by setting the 2nd bit on and then writing the modified value back. During the modification all other bits are left intact. This using the Bit Storage method, a single Module (or sensor...) can keep track of multiple events.

Appendix A.4. Panel Synchronization

Manual synchronization is required. Push the reset button on the panel. This transmits a reset message to the FieldServer, which clears the data in the FieldServer. After a reset the panel sends messages to report all abnormal states. When all these messages have been processed the FieldServer and panel will be synchronized. This process can be repeated at any time.

Appendix A.5. What happens when the panel sends a Reset Messages

When a panel sends a reset message the driver processes every single Map Descriptor, looking at the 'Clear on Reset' parameter (See section 4.1.2). If the parameter is set to yes, then the driver sets all the Data Array elements referenced by the Map Descriptor to zero by looking up the DA Name, the Data Array offset and the length. The driver also clears the relevant sections of a Data Array specified with the DA\_Bit\_Name parameter.

The process can take some time. For this reason, it is suggested that you take care not to set MD length to a value larger than necessary.

**Appendix B. Revision History**

Date	Resp	Format	Driver Ver.	Doc. Rev.	Comment
2/15/05	PMC		0.00	0	Document Created
3/1/05	Meg	Meg	0.00	1	Formatting updated. Some modifications to tables and text. Redrew connection drawing adding wiring detail. Reduced photograph resolution.
3/29/05	Meg	Meg	0.00	2	Minor changes as requested by customer – DUR0571. Repaired title number formatting.