Driver Manual

FS-8700-112 Gamewell FCI 7100 Series

APPLICABILITY & EFFECTIVITY
Effective for all systems manufactured after March 2021.

Driver Revision: 1.02
Document Revision: 2.A

MSA Safety
1000 Cranberry Woods Drive
Cranberry Township, PA 16066 USA
Website: www.MSAsafety.com

U.S. Support Information:
+1 408 964-4443
+1 800 727-4377
Email: smc-support@msasafety.com

EMEA Support Information:
+31 33 808 0590
Email: smc-support.emea@msasafety.com

Contents

1 Description

The FCI 7100 Series System Control Units are manufactured by Fire Control Instruments. A 7100 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 passive Client driver does not poll for data, nor does it send data or commands to the 7100. Messages received from the 7100 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 must be synchronized manually. This typically requires a panel reset.

Since the driver cannot send data or commands to the 7100 it cannot be used to acknowledge, silence, or reset alarms and other events.

The driver can process the single line messages sent form the 7100 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 can process messages from networked panels. The driver connects to the main panel. Subsidiary panels are configured to send event data to the main panel which then sends messages to the FieldServer. If the panel is configured to send 3 line messages, then the source node information is sent in the line preceding the event and the driver uses this to determine the panel at which the event originated and to store data appropriately.

Max Nodes Supported

FieldServer Mode	Nodes	Comments
Client	1	1 Node per serial port. If there is more than one alarm panel they can be networked and configured to send event data to the primary panel. The driver can process messages which identify the node of origin.
Server	32	1 Node per serial port.

2 Driver Scope of Supply

2.1 Supplied by MSA Safety

Part #	Description
23069	RJ45-RJ11/12 Cable assembly for FS connection to FCI panel.

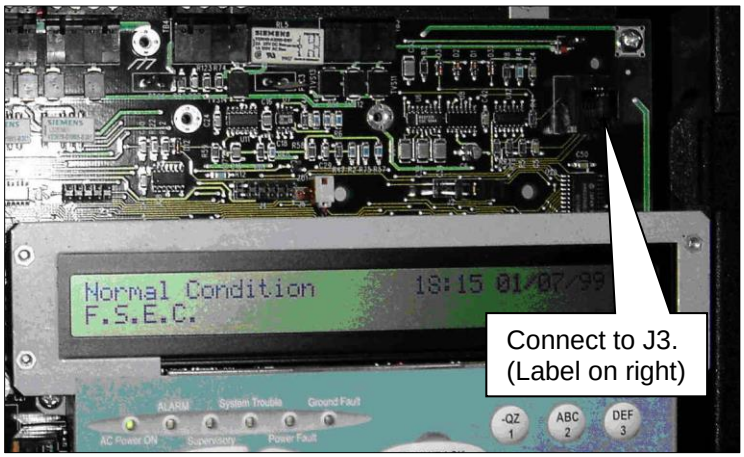
2.2 Provided by the Supplier of 3rd Party Equipment

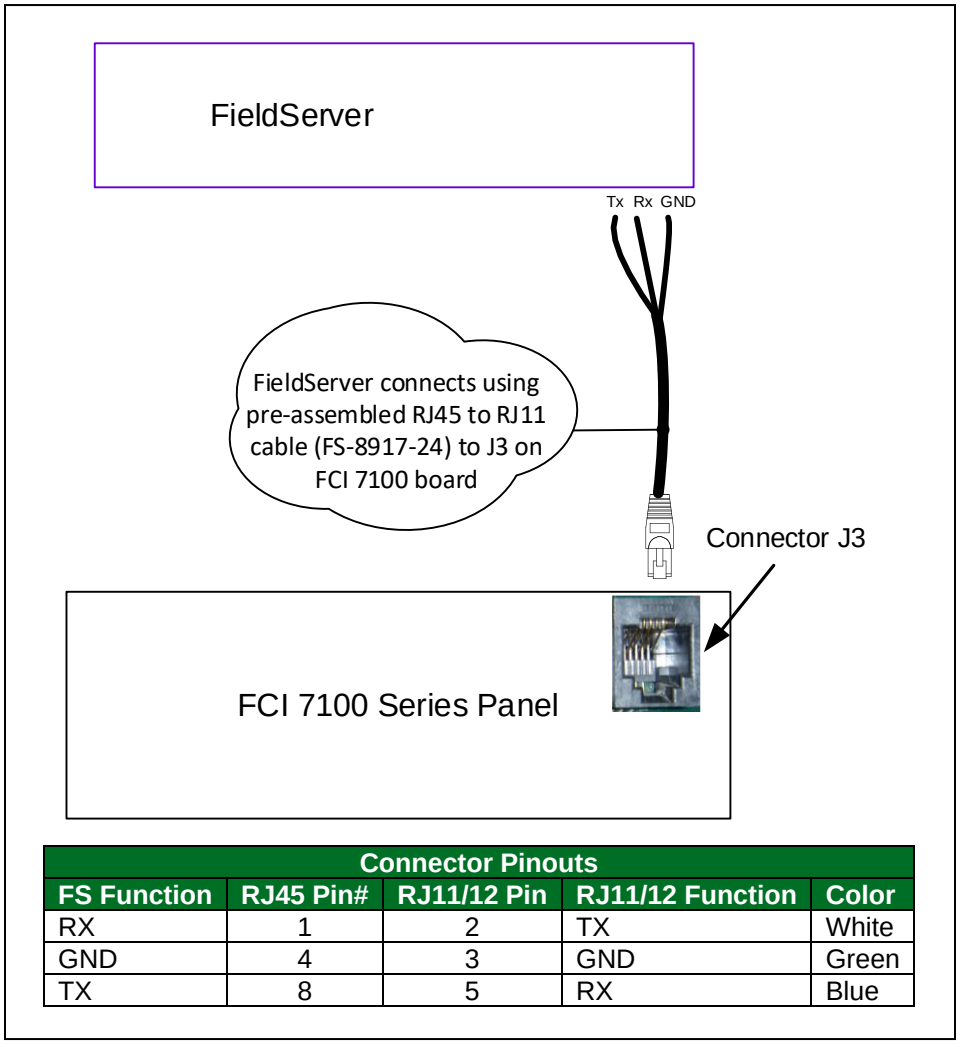
2.2.1 Required 3rd Party Hardware

- FCI Panel must be equipped with a RS-232 Serial Printer Port.
- FCI Panel must be equipped with a PTRN module for isolation of the serial port.

3 Hardware Connections

3.1 Communication Board





Connector Pinouts				
FS Function	**RJ45 Pin#**	**RJ11/12 Pin**	**RJ11/12 Function**	**Color**
RX	1	2	TX	White
GND	4	3	GND	Green
TX	8	5	RX	Blue

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.	Float, Bit, Byte, Uint16, Uint32, Sint16, Sint32
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.	1-10000

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
```

5 Client Side Configuration

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 Gamewell-FCI Series 7100.

NOTE: In the tables below, * indicates an optional parameter, with the bold legal value as 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[1]
Protocol	Specify protocol used.	FCI_7100, FCI_7100_series, FCI7100
Baud*	Specify baud rate.	**1200** (Vendor Limitation)
Parity*	Specify parity.	**None** (Vendor Limitation)
Data_Bits*	Specify data bits.	**8** (Vendor Limitation)
Stop_Bits*	Specify stop bits.	**1** (Vendor Limitation)

Example

```
//    Client Side Connections
Connections
Port     , Protocol      , Baud     , Parity
P1       , FCI_7100      , 1200     , None
```

[1] Not all ports shown may be supported by the hardware. Consult the appropriate Instruction manual for details of the hardware.

5.2 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*	When multiple panels are networked in an FCI network the Node_ID is the panel number. Set the Node_ID of the local panel to zero and create one Node descriptor for each panel setting the Node_ID to the panel number.	In networked configurations, specify whole numbers 0,1,2,3 …
Protocol	Specify protocol used.	FCI_7100, FCI_7100_series, FCI7100
FCI_Reset_Action_ Option*	Only required for networked configurations. This parameter tells the driver what to do with 'Reset' messages. When not specified or set to 'Reset_by_any_Node' then the driver will reset the data array points associated with the given node irrespective of the reset message's origin. When set to 'Reset_by_this_Node_Only' then the driver only resets the data associated with the given node if the reset originated from the same node.	**Reset_by_any_Node,** Reset_by_this_Node_Only
Connection	Specify through which port the device is connected to the FieldServer.	P1-P2, R1-R2[1]

Example

```
//    Client Side Nodes
Nodes
Node_Name     , Protocol      , Connection
Panel-01      , FCI_7100      , 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 "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_Client

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 “Client Node Descriptor” above
Event_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. A Map Descriptor may be defined to store only ‘Alarm’, ‘Fault’, ‘Trouble’ or ‘Other events. Alternatively, specify “Any”. A table of events vs. categories is provided in **Section 8.1**.	**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	Ignored when the Point Type is ‘Panel’ Point Type = Relay 1-255 Point Type = Zone 1-8 Point Type = Loop 1-2 Point Type = Module 1-2 Point Type = Sensor 1-2	Any integer
Length	Each Map Descriptor defines storage locations for a series of addresses. This parameter specifies the length of the series.	Any Integer
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 determines the range of the sensor/module numbers.	1-99, **-**
Store_As*	Set this parameter to ‘Bit’ to have the driver use the primary Data Array to store using the ‘Bit Storage’ Method. These methods are described in **Section 8.2**.	Bit, **Index_Value**
DA_Bit_Name*	If the default ‘Store As’ is specified or if the parameter is omitted then it is an option to 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 **Section 8.2**.	One of the Data Array names from “Data Array” section above
Clear_On_Reset*	If a reset is received, it is an option to prevent the driver resetting the Data Array Points associated with the Map Descriptor by specifying this parameter.	**Yes**, No

5.4 Map Descriptor Examples

5.4.1 Sensor / Module Events

This Map Descriptor will be used to store messages from Loop 1, Module 1 to 99. To store modules on more than one loop, a separate Map Descriptor needs to be defined for each loop. Since the event type was set to 'Alarm', only 'Alarm' events will be stored. To store all events, change the 'Event Type' to 'Any'.

F.S.E.C. :[CR][LF]
FIRST ALARM: UP STAIRS N. ENT Manual Station L1M21 00:37:28 01/01/99[CR][LF]

```
//  Client Side Map Descriptors

Map_Descriptors
Map_Descriptor_Name  , Data_Array_Name  , Data_Array_Offset  , Function        , Node_Name  , Event_Type  , Point_Type
ModuleData1          , DA_MODULE        , 0                 , Passive_Client  , Panel-01   , Alarm       , Module
```

```
, Relay/Loop/Zone_Number  , Address  , Length  , Clear_On_Reset
, 1                       , 1        , 99      , Yes
```

Example comments:

- Map_Descriptor_Name – It is recommended to allocate unique MD names.
- Data_Array_Name & Data_Array_Offset – Tell the driver the Data Array name and starting location that data should be stored.
- Function – The driver listens passively for messages from the Panel. It cannot poll for data.
- Node_Name – The name of the Node defined in the Node Descriptor section.
- Event_Type – In this example, only Alarm events will be stored. Messages reporting other events will be ignored unless other Map Descriptors are defined.
- Point_Type – Change this to 'Sensor' for sensors.
- Address & Length – The address specifies the starting Module number and the Length tells the driver the range of Modules. In this example: Module 1 to 99.
- Clear_On_Reset – The driver will clear the 1(=Length) element of the Data Array called DA_Panel starting at offset=0 when a Reset message is received.

5.4.2 Bit Storage

This example defines storage location for Relay Point events. The example would work for all 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 specific 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 Secondary 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
RelayData             , DA_RELAY          , 0                  , DB_Relay      , Passive_Client   , Panel-01
```

```
, Event_Type   , Point_Type      , Relay/Loop/Zone_Number   , Address   , Length   , Clear_On_Reset
, Any          , Relay           , 1                        , -         , 4        , Yes
```

Example comments:

- Data_Array_Name – Where the primary DA is specified. Index values are stored here.
- Data_Array_Offset – is where secondary storage is defined. Events are stored by setting appropriate bits. Remember that 2 elements per Relay, Module, Sensor, Loop are used.
- Address – Map Descriptors for storing Relay, Loop, Zone and Panel do not need the address specified.

6 Useful Features

6.1 Extending the Event Table

New event causes may be added to the Event Table (**Section 8.1**) 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.

6.1.1 Map Descriptor Parameters

Column Title	Function	Legal Values
Event_Type_Description*	Name (Description) of the new Event Types	Any string – max 19 characters
Event_Type_Index_Value	Provide the value that should be stored for a specific event.	-32768 to 32768, **0**
Event_Type_Category	Specify the category to which the new event should belong	-32768 to 32768, **0**

6.1.2 Example – Index Value "Trouble" Updated to a Value of 100

```
Driver_Table
Event_Type_Description   , Event_Type_Index_Value   , Event_Type_Category   , Protocol
TROUBLE                  , 100                      , 4                     , FCI_7100
```

6.1.3 Example – New Entry Added

Since it has been added as category=3, only Map Descriptors 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_7100
```

For categories use the following values:
'Other' = 1
'Fault' = 2
'Alarm' = 3
'Trouble' = 4

6.2 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.

6.3 Process When the Panel Sends a Reset Message

When a panel sends a reset message the driver processes every single Map Descriptor, looking at the 'Clear on Reset' parameter (see **Section 5.3.2**). 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 be time-consuming. For this reason, it is suggested not to set Map Descriptor length to a value larger than necessary.

Additional considerations apply when panels are connected together in an FCI network. See **Section 6.4**.

6.4 Networked Panels

The driver can processes messages and store data from multiple panels, provided that:

- The panels are connected in an FCI network and the panels are configured to report their events to the main panel.
- The main panel is configured to send the node of origin in a message preceding the event message. Consult with FCI for information on how to achieve this.

An example of a message sent by a networked panel follows. The driver relies on seeing the node of event origin included in parenthesis before each event message.

```
F.S.E.C. (Node02):
MISSING:        PROJ MANG OFFICE      Thermal Det     L1S02  00:40:18 01/01/99
F.S.E.C. (Local):
RESET:                                          00:40:18 01/01/99
```

The main panel is identified as '(Local)'. The driver interprets this as Node_ID=0.

To capture events from multiple networked panels, one Node Descriptor is required for each panel with the appropriate Node_ID. Each Node requires a set of Map Descriptors.

7 Troubleshooting

7.1 Driver Error Messages

Message	Description
FCI7100:#1 FYI. Use a DA called <%s> to expose diagnostic info., FCI_7100_STATS_DA)	Refer to **Section 7.2**.
FCI7100:#2 FYI. Added Event Desc=<%s> Index=%d Categ=%d , new_event_desc , new_event_desc_index_value , new_event_desc_categ) ;	Printed for local info only. No action required if it confirms expectations.
FCI7100:#3 Err. No space. Reject Event Desc=<%s> Index=%d , new_event_desc , new_event_desc_index_value) ;	There is only space for 60 event types.[2]
FCI7100:#4 FYI. Duplicate Event Desc=<%s>. , new_event_desc) ;	Adding an event type that already exists. If updating the category, ignore this message. Otherwise correct the configuration file.[2]
FCI7100:#5 FYI. Duplicate Event Desc=<%s>. , new_event_desc) ;	
FCI7100:#6 Err. Event Index=%d. Too big to set bit., drv_bd->event_index)	If the event index is greater than 64 then the data cannot be stored as bits as only 64 bits are reserved for events.
FCI7100:#7a Err. DA=%s too short. Rqd=%d, dt->buffer_name , offset);	The Map Descriptor in question has a length and offset which makes it run past the end of the Data Array. Message 7b is printed when data is being stored as bits.[2]
FCI7100:#7b Err. DA=%s too short. Rqd=%d, possible_md->data->buffer_name , offset);	
FCI7100:#8 FYI. Reset was rcvd and processed! Stamped %s %s , drv_bd->time , drv_bd->date)	Printed for information only. No action required.
FCI7100:#9 Err. Reset was ignored.	This message is printed when a reset was received but the driver could not reset any data. Ensure that 'Clear_on_Reset' is set to 'no' on all Map Descriptors.
FCI7100:#10 FYI. Reset of DA=%s Off=%d Len=%d, possible_md->data->buffer_name , possible_md->bxi_data_buffer_offset , possible_md->data_length);	Printed for information only. No action required.
FCI7100:#11 Err. Cant reset DA=%s len=%d rqd=%d, possible_md->data->buffer_name , da_get_length_in_items ((DAH_TYP) possible_md-> data) , possible_md->data_length+possible_md->bxi_data_buffer_offset);	The Map Descriptor in question has a length and offset which makes it run past the end of the Data Array.[2]
FCI7100:#12a Err. No MD's to store message data.	A message arrived and the driver could not find a place to store the data. If there is no interest in the data then ignore the message. Otherwise update the configuration file. [2]
FCI7100:#12b Err. No MD's to store message data."	
FCI7100:#13 Err. Msg was ignored. MD Required for Storage.	

[2] Correct the configuration file, download to the FieldServer and restart the FieldServer for the changes to take effect.

Message	Description
FCI7100:#13a Err. Diagnostic 1);	Take a log. Try and repeat the event that caused the message to be printed. Then contact technical support.
FCI7100:#13b Err. Diagnostic 2);	
FCI7100:#13c Err. Diagnostic 3);	
FCI7100:#14 Err. <%s> file not found., md->mapdesc_name) ;	If this error is repeated often it is possible that a FCI firmware update as made the driver unusable. Take a log and contact technical support.
FCI7100:#15 Err. Event Type=<%s> Not recognized." , drv_bd->event_desc)	
FCI7100:#16 Err. Point Type='%c'(%#x) Not recognized. , drv_bd->point_identifier[0] , drv_bd->point_identifier[0]) ;	
FCI7100:#17 Err. Loop=%d < 1. Rejected. , drv_bd->loop)	This message is printed if a byte in a message has been corrupted. If noticed more than once then take a log and contact technical support.
FCI7100:#18 Err. Loop Type='%c'(%#x) Not recognized. , drv_bd->point_identifier[2] , drv_bd->point_identifier[2])	If this error is repeated often it is possible that a FCI firmware update as made the driver unusable. Take a log and contact technical support.
FCI7100:#19 Err. Relay=%d < 1. Rejected. , drv_bd->relay	This message is printed if a byte in a message has been corrupted. If noticed more than once then take a log and contact technical support.
FCI7100:#20 Err. Zone=%d < 1. Rejected. , drv_bd->zone"	
FCI7100:#21 Err. Point Type not recognized	Valid Point Types are listed in **Section 5.3.2**.[3]
FCI7100:#22 Err. Undefined Point Type"	
FCI7100:#23 Err. Event Type not recognized	Valid Event Types are listed in **Section 5.3.2**.
FCI7100:#24 Err. Undefined Event Type	
FCI7100:#25a Err. Address+Length>99. Length Truncated	The maximum value for a sensor/module is 99. The combination of address and length specified produce a number > 99.
FCI7100:#25b Err. Address+Length>99. Length Truncated	
FCI7100:#26 Err. Invalid Module number. Expected 1..99	Correct the configuration file (**Section 5**).
FCI7100:#27a Err. Invalid Loop number. Expected 1..10	
FCI7100:#27b Err. Invalid Loop number. Expected 1..10	
FCI7100:#27c Err. Invalid Loop number. Expected 1..10	
FCI7100:#28 Err. Invalid Sensor number. Expected 1..99	Correct the configuration file (**Section 5**).
FCI7100:#29 Err. Invalid Zone number. Expected 1..255	
FCI7100:#30 Err. Invalid Relay number. Expected 1..255	
FCI7100:#31 Err. Point Type Invalid.	Valid Point Types are listed in **Section 5.3.2**.

[3] Correct the configuration file, download to the FieldServer and restart the FieldServer for the changes to take effect.

Message	Description
FCI7100:#32 Err. No MD Length. Default to 1	Specify the 'length' of each Map Descriptor. Refer to **Section 5.3.2**.
FCI7100:#33 Err. Driver cant poll or write.	The driver can only listen passively for message from the panel. Remove any active Map Descriptors from the configuration file.
FCI7100:#36 Err. Too Short. Bytes=%d , conn->ux_iptr"	An event message is less than 80 bytes long. If this error is repeated often it is possible that a FCI firmware update as made the driver unusable. Take a log and contact technical support.

7.2 Driver Stats Exposed

In addition to the standard FieldServer operating statistics the driver exposes certain key stats in a Data Array if required. A Server Side device can then monitor these stats.

Add the following to the configuration file to activate these stats.

```
// Expose Driver Operating Stats.

Data_Arrays
Data_Array_Name    , Data_Format    , Data_Array_Length
fci-7100-stats     , UINT32         , 200
```

The driver exposes stats based on a port handle. The offset specified in the table below must be added to the handle number multiplied by 100. For example, for a port whose handle is 1, the driver will store the 1st stat at 1+100*1=101.

Stat	Offset	Description
#define FCI_7100_STAT_NO_PLACE_TO_STORE	1	Increments each time point data is received but there is no Map Descriptor to store the data (any)
#define FCI_7100_STAT_NO_PLACE_TO_STORE_ZONE	2	Increments each time point data is received but there is no Map Descriptor to store Zone data
#define FCI_7100_STAT_NO_PLACE_TO_STORE_RELAY	3	Increments each time point data is received but there is no Map Descriptor to store Relay data
#define FCI_7100_STAT_NO_PLACE_TO_STORE_LOOP	4	Increments each time point data is received but there is no MD to store the Loop data
#define FCI_7100_STAT_NO_PLACE_TO_STORE_SENSOR	5	Increments each time point data is received but there is no MD to store the Sensor data
#define FCI_7100_STAT_NO_PLACE_TO_STORE_MODULE	6	Increments each time point data is received but there is no MD to store the Module data

Stat	Offset	Description
#define FCI_7100_STAT_EMPTY_MSG	7	Number of times that a message line was zero bytes long (excluding the terminator)
#define FCI_7100_STAT_SHORT_MSG	8	Number of times that a message line was too short probably a system id tag line
#define FCI_7100_STAT_NO_RESET	9	Increments each time a reset was rcvd but no DA was reset.
#define FCI_7100_STAT_NO_PLACE_TO_STORE_PANEL	10	Increments each time point data is received but there is no MD to store data that cannot be attributed to a zone, relay, loop, sensor, module.
#define FCI_7100_STAT_RCVD_MSGS	11	Increments each time a message is received.
#define FCI_7100_STAT_RCVD_BYTES	12	Increments each time a character is received from the panel. The bytes are only added when a message terminator is received. Thus, this count is equivalent to the byte count in all FCI_7100_STAT_RCVD_MSGS
#define FCI_7100_STAT_PARSED_NO_ERRS_EXCLD_RESET	13	Increments each time a message is parsed without errors. Excludes Reset Messages.
#define FCI_7100_STAT_PARSED_NO_ERRS_RESET	14	Increments each time a reset message is parsed without errors.
#define FCI_7100_STAT_PARSED_NO_ACTION	15	Increments each time a message is parsed with no errors but the nature of the message doesn't require data to be stored (e.g. empty msg lines).
#define FCI_7100_STAT_PARSED_WITH_ERRS	16	Increments each time a message produces an error when parsed.
#define FCI_7100_STAT_INHIBIT_RESET	17	Set to 1 to inhibit resets altogether.
#define FCI_7100_STAT_INHIBIT_RESET_DA_PUT	18	Set to 1 to inhibit resets from clearing arrays.
#define FCI_7100_STAT_INHIBIT_RESET_WHILE	19	Set to 1 to inhibit reset function from looping thru MD's.
#define FCI_7100_STAT_NODE_INFO_MSG	20	Increments each time a message with node information is received.
#define FCI_7100_STAT_NO_PLACE_TO_STORE_NODE	21	Increments each time an event needs to be stored, the event contains node info and the node cannot be found so there is no match and the message was discarded.

8 Reference

8.1 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"

8.2 Data Storage

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 according to the ‘Event Type’ specification.
- 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 Map Descriptor 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 Map Descriptors 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 Map Descriptors 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 Map Descriptors remain in contention.

- The Module number of 22 is compared with the Map Descriptors Address and Length Parameters. The Address is the starting number and the length defines the range. Both Map Descriptors 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 Map Descriptors 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 34th bit – or expressed another way, the driver modifies the 2nd bit (34-32) at offset+1.
- Thus, driver calculates the offset for Bit Storage as 2 x 21 = 42. The driver sees that bit 34 is 2 2nd 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. Thus using the Bit Storage method, a single Module (or sensor) can keep track of multiple events.