



## **FieldServer**

### **FS-8700-69 Interlogix Automation Module Protocol Driver**

**(Supports Advent, Concord, and Concord Express Panels)**

### **Driver Manual**

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

#### **APPLICABILITY & EFFECTIVITY**

**Effective for all systems manufactured after December 2015**

Driver Version: 1.00  
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](mailto:info@sierramonitor.com)

Website: [www.sierramonitor.com](http://www.sierramonitor.com)

**TABLE OF CONTENTS**

**1 Interlogix advent serial driver description ..... 4**

**2 Driver Scope of Supply ..... 6**

2.1 Supplied by Sierra Monitor Corporation for this driver ..... 6

2.2 Provided by the Supplier of 3<sup>rd</sup> Party Equipment..... 6

**3 Hardware Connections..... 7**

3.1 Serial Port Physical Interface ..... 8

**4 Configuring the FieldServer as a Interlogix Advent Serial Driver Client ..... 9**

4.1 Data Arrays..... 9

4.2 Client Side Connection Descriptors..... 10

4.3 Client Side Node Descriptors ..... 10

4.4 Client Side Map Descriptors ..... 11

4.4.1 *FieldServer Specific Map Descriptor Parameters* ..... 11

4.4.2 *Driver Specific Map Descriptor Parameters* ..... 11

4.4.3 *Timing Parameters* ..... 13

4.4.4 *Map Descriptor Examples: Polling for Data.* ..... 14

4.4.4.1 Polling a Panel for a Refresh of Dynamic Data ..... 14

4.4.4.2 Polling a Panel for a List of all its Equipment..... 15

4.4.4.3 Polling a panel for specific data. .... 16

4.4.5 *Map Descriptor Examples: Storing Panel Data*..... 17

4.4.5.1 Zone Status..... 17

4.4.5.2 Arming Level. .... 18

4.4.5.3 Entry Exit Delay..... 19

4.4.5.4 Alarm / Trouble..... 20

4.4.5.5 Siren Setup. .... 21

4.4.5.6 Siren Stop / Go..... 22

4.4.5.7 Touchpad Display ..... 23

4.4.5.8 User Lights. .... 24

4.4.5.9 Zone Data..... 25

4.4.5.10 Partition Data. .... 26

4.4.5.11 Device Data..... 27

4.4.5.12 Superbus Device Capability Data. .... 28

**5 Configuring the FieldServer as a Interlogix Advent Serial Driver Server ..... 29**

5.1 Abandoned Messages..... 29

5.2 Poll Timing Issues..... 32

5.3 Driver Stats ..... 32

5.4 Data Variables & Data Storage Defined..... 34

5.5 Alarm / Trouble General & Specific Types ..... 38

5.6 Text Tokens ..... 42

5.7 Driver Error Messages..... 45

5.8 Device Addresses..... 47

5.9 Device Capabilities ..... 47

**LIST OF FIGURES**

Figure 1 - Generic Connection Diagram..... 7

## **1 INTERLOGIX ADVENT SERIAL DRIVER DESCRIPTION**

The Interlogix Advent Serial Driver allows the Communications FieldServer to transfer data to and from devices over either RS-232 or RS-485 using Interlogix Advent Serial Driver protocol. The FieldServer can emulate either a Server or Client

The Interlogix panels generates unsolicited messages in response to changes in Zone Status, Alarm conditions etc. This driver is capable of parsing and storing data from all these messages and in addition the driver is capable of polling an Interlogix panel for refreshed dynamic data and equipment lists.

Driver configuration (as a client) is required to allow the driver to store data from these incoming messages. Incoming messages which cannot be processed by this configuration are abandoned.

The driver is capable of exposing its communication statistics in a Data Array so that they can be monitored by any downstream device.

This driver is capable of acting like an Interlogix panel in the sense that it can serve data and respond to polls in the same way.

The following table indicates the types of data that can be communicated using the Interlogix Serial Driver. The table indicates the data that is sent as unsolicited data (U) by an Interlogix panel, data that is obtained in response to a poll by the client (F→ Full Equipment List Poll S→ Specific Equipment List Poll) or data that is obtained in response to a client making a Dynamic Data Refresh Request.

**Table 1**

Data Message	Unsolicited	Equipment List Request	Dynamic Data Refresh
Panel Type	U on pwr/comms	F <sup>1</sup>	D <sup>2</sup>
Automation Event Lost	U on buffer o/f		
Clear Automation Dynamic Image`	U on pwr/comm		
Zone Status	U on not normal		D
Arming Level	U on Change		D
Entry/Exit Delay	U on start/end		
Alarm/Trouble	U on event		
Siren Setup	U on beep/siren		D <sup>3</sup>
Siren Synchronize	U on event		
Siren Go	U on event		
Touchpad Display	U on touch		D
Siren Stop	U on event		
Feature State	U on Change		D <sup>4</sup>
Temperature			D <sup>2</sup>
Time and Date			D <sup>4</sup>
Lights State Command	U on Change		D <sup>2</sup>
User Lights Command	U on Event <sup>5</sup>		
Keyfob Command	U on keypress		
Zone Data		F, S	
Partition Data		F, S	
SuperBus Device Data		F, S	
SuperBus Device Capabilities Data		F, S	
Output Data		F, S	
User Data		F, S	
Schedule Data		F, S <sup>2</sup>	
Schedule Event Data		F, S <sup>2</sup>	
Light to Sensor Attachment		F, S <sup>2</sup>	
Keypress	Subject to verification		

<sup>1</sup> Advent Only

<sup>2</sup> Concord Only

<sup>3</sup> In Response to refresh poll if siren active

<sup>4</sup> Not supported by Advent

<sup>5</sup> Concord Express Only

**2 DRIVER SCOPE OF SUPPLY**

**2.1 Supplied by Sierra Monitor Corporation for this driver**

Sierra Monitor Corporation PART #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection
FS-8915-10	UTP cable (7 foot) for RS-232 use
FS-8917-02	RJ45 to DB9F connector adapter
FS-8917-01	RJ45 to DB25M connection adapter
FS-8700-69	Driver Manual.

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

Advent Home Navigation System which includes the RS-232 module.

### 3 HARDWARE CONNECTIONS

The FieldServer is connected to the Interlogix advent, concord or concord express panel's automation module serial port AS SHOWN below.

Configure the panel according to manufacturer's instructions.

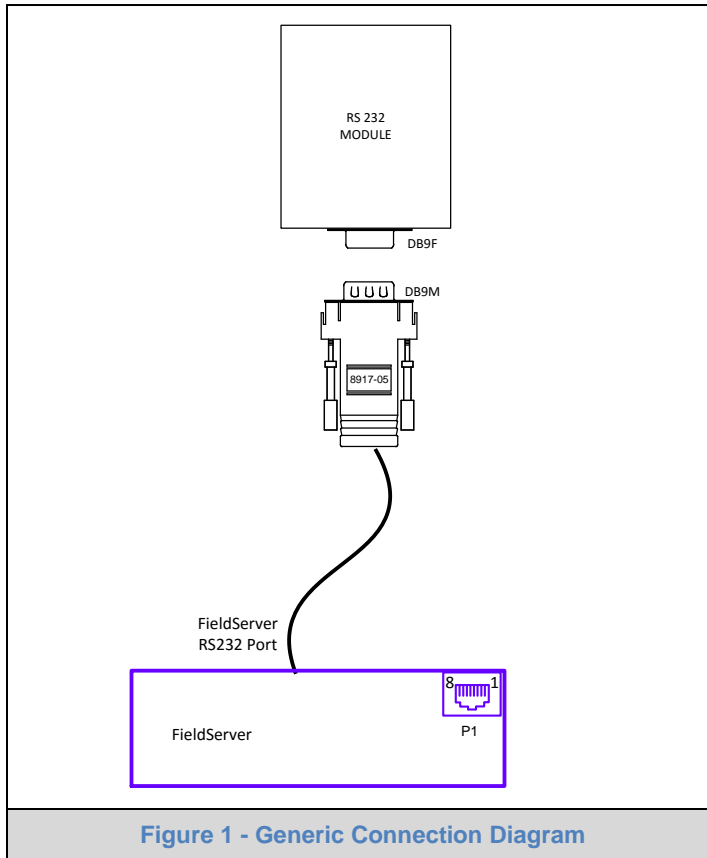


Figure 1 - Generic Connection Diagram

### 3.1 Serial Port Physical Interface

- Serial parameters: 9600 baud, 8 data bits, odd parity, 1 stop bit.
- Serial Port is a RS-232 level, 3-wire Data Communications Equipment (DCE) serial port interface.
- Use a straight-through cable to connect to a PC's 9-pin serial port.

Female 9-pin D-shell connections as follows:

Pin 2	Transmit data (Automation Module's point of view)
Pin 3	Receive data (Automation Module's point of view)
Pin 5	Ground
Remaining pins	No-connect

**WARNING:** The Automation Module's serial port must be connected to an Automation Device in order to work. The level translator IC used on the serial port is configured for a power-saving mode. This IC automatically shuts down if it does not detect a valid RS-232 level on its receiver input. Using a Break Out box alone to monitor the port will not cause the IC to come out of shut down.



## 4 CONFIGURING THE FIELD SERVER AS A INTERLOGIX ADVENT SERIAL DRIVER CLIENT

For a detailed discussion on FieldServer configuration, please refer to the instruction manual for the FieldServer. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See “.csv” files on the driver diskette).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an Interlogix Advent Serial Driver Server.

It is possible to configure the FieldServer in the following modes

- **Passive Client** - The FieldServer parses and stored unsolicited messages generated by the panel.
- **Active Client** -The FieldServer polls the panel for refreshed dynamic data and/or the panel's equipment list.
- **Full Client** - The FieldServer processes unsolicited messages and periodically polls for data.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for Interlogix Advent 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.

### 4.1 Data Arrays

Section Title		
Data_Arrays		
Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_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 for the data being placed in this array.	1-10,000

**Example**

```

// Data Arrays
Data_Arrays
Data_Array_Name      , Data_Format      , Data_Array_Length
DA_AI_01             , UInt16           , 200
DA_AO_01             , UInt16           , 200
DA_DI_01             , Bit              , 200
DA_DO_01             , Bit              , 200
```

## 4.2 Client Side Connection Descriptors

Section Title		
Connections		
Column Title	Function	Legal Values
Port	Specify which port the device is connected to the FieldServer	P1-P2, R1-R2
Baud*	Specify baud rate	96006
Parity*	Specify parity	Odd 6
Data_Bits*	Specify data bits	8 6
Stop_Bits*	Specify stop bits	1 6
Protocol	Specify protocol used	llgx, Advent
Handshaking*	Specify hardware handshaking	None 6
Poll Delay*	Time between internal polls	0-32000 s, 1 s

### Example

```
// Client Side Connections

Connections
Port          , Baud          , Parity          , Protocol          , Poll_Delay
P1            , 9600            , Odd             , llgx              , 0.100 s
```

## 4.3 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	Although this parameter is commonly used in many driver configurations it is not required for the Inetrlogix Serial Driver as the connections are nodeless.	Not Required
Protocol	Specify protocol used	llogix, Advent
Port	Specify which port the device is connected to the FieldServer	P1-P2, R1-R2

### Example

```
// Client Side Nodes

Nodes
Node_Name          , Protocol          , Port
Panel 1            , llgx              , P1
```

<sup>6</sup> The driver can support all std baud rates & port configurations but at the time of printing, Interlogix panels only support 9600, 8, 1 Odd

## 4.4 Client Side Map Descriptors

At least one Map Descriptor is required for every data type you wish to store based on the data messages described in Table 1. Any incoming message that cannot be processed using one of the Map Descriptors you define will be abandoned. If you are polling the panel for equipment lists or a refresh of dynamic data the poll Map Descriptor should be matched with one or more passive Map Descriptors used for storage of the panel's response.

### 4.4.1 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_Location	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor	RDBC, WRBC, WRBX

### 4.4.2 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 "Client Node Descriptor" above
Data_Type*	Data type is required only for client poll Map Descriptors used to read equipment lists from the Interlogix panel. .  When you want to poll for a specific equipment list the Data_Type is used by the driver to request the specific equipment list.	All , Zones, Alarms, Devices, Dev. Capabilities, Outputs , Users, Schedules, Scheduled Events, Lights, Phones, Systems, Partitions
Length*	Length of Map Descriptor. Only some Map Descriptors require that a length be specified. See the examples that follow this section.	1 - 1000
Address*	Starting address of read Descriptor. . Only some Map Descriptors require that a Address be specified. See the examples that follow this section	0, 1, 2...
llgx_function	Used to tell the driver what kind of data to process.	Panel Type Auto Event Loss Zone Status Arming Level Entry Delay Alarm

		<p>Siren Setup  Siren Synch  Siren State  Touchpad  Feature State  Temperature  Date  Lights State  User Lights  Keyfob  Keypress  Zone Data  Partition Data  Superbus Device Data  Device Data  Superbus Device Capability  Device Capability  Output Data  User Data  Schedule Data  Scheduled Event  Light to Sensor Data  Clear Dynamic Data</p>
Option*	Used for poll RDBC Map Descriptors. Tells the driver whether to poll for Dynamic Data or for Equipment Lists	<p><b>Dynamic</b>  Static</p>
Partition*	Some Map Descriptors require this parameter to determine which incoming data should be processed. If the Interlogix panel only has one partition and the partition number is zero then this parameter need not be specified.	<p><b>0, 1, 2...</b></p>
Area*	Some Map Descriptors require this parameter to determine which incoming data should be processed. If the Interlogix panel only has one area and the area number is zero then this parameter need not be specified.	<p><b>0, 1, 2...</b></p>
Group*	Some Map Descriptors require this parameter to determine which incoming data should be processed. If the Interlogix panel only has one group and the group	<p><b>0, 1, 2...</b></p>

	number is zero then this parameter need not be specified.	
DA_Byte_Name*	<p>Some panel data messages have more than one data variable contained in the message. If you do not define additional arrays for storage of these additional data variables then the additional data is discarded.</p> <p>See Table 6.4 for more information on which Panel Data messages use 2ndary data storage.</p> <p>Don't let the names of these parameters mislead you. The arrays can be of any data type, FLOAT, UINT16 ....</p>	One of the Data Array names from "Data Array" section above
DA_Bit_Name*	See Notes above.	One of the Data Array names from "Data Array" section above

#### 4.4.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is polled	>0.1s

#### 4.4.4 Map Descriptor Examples: Polling for Data.

##### 4.4.4.1 Polling a Panel for a Refresh of Dynamic Data

This example polls the panel every 5.0s for a refresh of dynamic data. Use Table 1 to determine what kind of responses the panel will provide. Note that this Map Descriptor, whilst sufficient to generate the poll, requires one or more additional passive Map Descriptors to process and store the data that the panel will respond with.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Option	Scan_Interval	Timeout
Pol1	DA_AI3	0	Rdbc	Node_A	Dynamic	5.0s	10.0s

Identifier. Unique names are not required but may prove useful in diagnosing problems as some error messages report the offending Map Descriptors name.

Interlogix Driver read Map Descriptors do not store data from responses so this Data Array will not be used by this Map Descriptor. It must, however, be specified, to complete a FieldServer configuration.

Driver asks panel to send a refresh of the dynamic data. Table 1 identifies the data that will for the responses.

Ensure that the poll period is sufficiently slow to allow the panel to complete its response.

You cannot know in advance how much data will be returned by this poll so it is important to set the timeout to a large number.

#### 4.4.4.2 Polling a Panel for a List of all its Equipment

This example polls the Interlogix Panel and asks it to report all the equipment associated with the panel. The data that the pane will send as a response is identified in Table 1. Note that this Map Descriptor, whilst sufficient to generate the poll, requires one or more additional passive Map Descriptors to process and store the data that the panel will respond with.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Option	Scan_Interval	Timeout
Poll-1	DA_AI3	0	Rdbc	Node_A	Equipment	5 .0s	10.0s

The driver asks the panel for its equipment list data. See table 1 for a list of data that will for the responses.

As no data type has been specified the driver requests a full list.

Again, ensure that the poll period is sufficiently slow to allow the panel to complete its response which could consist of dozens of messages.

4.4.4.3 Polling a panel for specific data.

This poll is a variation of the poll in 4.4.4.2. It differs in that instead of requesting all equipment list data from the panel, the panel is asked for a specific data set. In this example, Zone Data is requested. . Note that this Map Descriptor, whilst sufficient to generate the poll, requires one or more additional passive Map Descriptors to process and store the data that the panel will respond with.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Option	Data_Type	Scan_Interval	Timeout
Poll-1	, DA_AI3	, 0	, Rdbc	, Node_A	, Equipment	, Zone Data	, 5 .0s	, 10.0s

By specifying a data type the poll is tuned from a request for a full report of equipment/static data to request for a specific data set. In this case Zone Data.

See table 6.4 for a list of variables that are returned by this poll. As most data sets consists of composite data table 6.4 describes how the composite data is stored.

Possible Data Types are: **All**<sup>1</sup>, Zones, Alarms, Devices, Dev. Capabilities, Outputs, Users, Schedules, Scheduled Events, Lights, Phones, Systems, and Partitions  
 Note<sup>1</sup> Default.



### 4.4.5 Map Descriptor Examples: Storing Panel Data

#### 4.4.5.1 Zone Status.

Zone status is different from zone data. The zone status is an unsolicited message sent by a panel when the state of a zone changes. Zone status messages are also sent by the panel in response to a poll for a refresh of dynamic data.

In this example Map Descriptors are provided which are used to process responses from two partitions each of which has 2 areas. A maximum of 100 zones has been defined for each area. The Array\_Offset is adjusted for each Map Descriptor to place all the incoming zone data in one Data Array called ZONE\_STATUS. Messages for other partition/area combinations will be ignored unless additional Map Descriptors are defined. The zone stats number indicates the zone state – consult the Interlogix panel documentation for current definitions. (bit 0: 1 = tripped bit 1: 1 = faulted bit 2: 1 = Alarm bit 3: 1 = Trouble bit 4: 1 = Bypassed)

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Partition	Area	Address	Length
Panel-1-Zone-MD1	, ZONE_STATUS	, 0	, Passive	, Panel-1	, Zone Status	, 0	, 0	, 1	, 100
Panel-1-Zone-MD2	, ZONE_STATUS	, 100	, Passive	, Panel-1	, Zone Status	, 0	, 1	, 1	, 100
Panel-1-Zone-MD3	, ZONE_STATUS	, 200	, Passive	, Panel-1	, Zone Status	, 0	, 2	, 1	, 100
Panel-1-Zone-MD4	, ZONE_STATUS	, 300	, Passive	, Panel-1	, Zone Status	, 1	, 0	, 1	, 100
Panel-1-Zone-MD5	, ZONE_STATUS	, 400	, Passive	, Panel-1	, Zone Status	, 1	, 1	, 1	, 100
Panel-1-Zone-MD6	, ZONE_STATUS	, 500	, Passive	, Panel-1	, Zone Status	, 1	, 2	, 1	, 100

All Map Descriptors used to store incoming messages from the panel are passive.

This keyword tells the driver that these Map Descriptors should be used to process incoming Zone Status messages.

One Map Descriptor is required for each partition, area combination.

The address & length parameters define the range of zones that will be processed by this Map Descriptor.

4.4.5.2 Arming Level.

This example Map Descriptor is used to store arming level data. Consult the Interlogix Panel Manuals for current information on Arming Level. At the time of printing of this manual for Advent Panels (0 = Zone Test 1 = Off 2 = Home/Perimeter 3 = Away/Full 4 = Night 5 = Silent) and for Concord Panels (1 = Off 2 = Stay 3 = Away 8 = Phone Test 9 = Sensor Test)

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Partition	Address	Length
Panel-1-Arm-MD1	ARMING_LVL5	0	Passive	Panel-1	Arming Level	0	1	100

The address & length parameters define the range of areas that will be processed by this Map Descriptor. Area zero's data will be stored at offset 0, area one's data will be stored at offset 1...

#### 4.4.5.3 Entry Exit Delay.

This example is similar to the previous. The Address/Length parameters define the range of areas covered by the Map Descriptor. A new Map Descriptor is required for each different partition.

Entry/Exit delay data consists of more than one variable – The delay Flags and the Delay Time. The driver stores the delay flags in the primary Data Array (called DELAY\_FLAGS in this example) and abandons the delay time unless a secondary Data Array is defined using the da\_byte\_name parameter. If this is defined then the delay time is stored in the corresponding array element

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Partition	Address	Length
Panel-1-Exit-MD1	DELAY_FLAGS	0	Passive	Panel-1	Delay Time	0	1	100

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Partition	Address	Length	DA_Byte_Name
Panel-1-Exit-MD2	DELAY_FLAGS	0	Passive	Panel-1	Delay Tim	0	1	100	DELAY_TIMES

Delay flag data sent by the panel is stored here.

This Map Descriptor does not define a Data Array for the delay time and the data is discarded.

A Data Array called DELAY\_TIME is defined and the driver stores the delay time data in this array. The location is calculated using the same offset as used for the primary Data Array.

4.4.5.4 Alarm / Trouble

One Map Descriptor is required for each partition / area combination defined in the panel. Alarm messages are generated by 4 source types – Bus Devices – Phones – Zones – Systems. One Map Descriptor is required for each type of source type. The Interlogix documentation should be consulted for current alarm/trouble general/specific type and event specific data values and their meanings.

In the example below Map Descriptors are provided for one partition, several area and several source types.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Data_Type	Partition	Area	Address	Length	DA_Byte_Name
Panel-1-Alarm-MD1	DEV_ALARMS	0	Passive	Panel-1	Alarm	Devices	0	1	1	100	DEV_ES_DATA
Panel-1-Alarm-MD2	ZON_ALARMS	0	Passive	Panel-1	Alarm	Zones	0	1	1	100	ZON_ES_DATA
Panel-1-Alarm-MD3	PHO_ALARMS	0	Passive	Panel-1	Alarm	Phones	0	1	1	100	PHO_ES_DATA
Panel-1-Alarm-MD4	SYS_ALARMS	0	Passive	Panel-1	Alarm	Systems	0	1	1	100	SYS_ES_DATA
Panel-1-Alarm-MD5	DEV_ALARMS	100	Passive	Panel-1	Alarm	Devices	0	2	1	100	DEV_ES_DATA
Panel-1-Alarm-MD6	ZON_ALARMS	100	Passive	Panel-1	Alarm	Zones	0	2	1	100	ZON_ES_DATA
Panel-1-Alarm-MD7	PHO_ALARMS	100	Passive	Panel-1	Alarm	Phones	0	2	1	100	PHO_ES_DATA
Panel-1-Alarm-MD8	SYS_ALARMS	100	passive	Panel-1	Alarm	Systems	0	2	1	100	SYS_ES_DATA

Alarm/Troubles are identified by a general and specific type number pair. The driver stores the pair by converting it. The value found at each location is the  $GT \times 100 + ST$ . Thus you must divide by 100 to extract the General Type. See **Table 6.5** for more information.

The offset for the next area's Map Descriptors is determined using the length of the previous area's Map Descriptors so that the storage does not overlap.

One Map Descriptor per partition/area/source type combination. The source type is specified using the data type parameter.

The address / length parameters are used to define the range of devices/zones ... that are processed by the Map Descriptor. In MD1 devices numbered 1 to 100 are processed.

Event Specific Data is stored in these Data Arrays. Location corresponding to the location in the primary Data Arrays are used.

4.4.5.5 Siren Setup.

Use one Map Descriptor for each partition/area combination. The Siren Repeat count is stored in the primary Data Array. A secondary array must be specified using the DA\_Byte\_Name parameter for the driver to store the Siren Cadence data. In these examples 3 Map Descriptors are provided for 3 different areas of partition zero. If you have more partitions and more areas you will need more Map Descriptors. Normally this message is an unsolicited message sent by the panel when the data changes. If an alarm is active then the siren setup message is sent by the panel in response to a dynamic data refresh poll too.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Partition	Area	DA_Byte_Name
Panel-1-Sset-MD1	SIREN_REPEATS	0	passive	Panel-1	Siren Setup	0	1	SIREN_CADENCE
Panel-1-Sset-MD2	SIREN_REPEATS	4	passive	Panel-1	Siren Setup	0	2	SIREN_CADENCE
Panel-1-Sset-MD3	SIREN_REPEATS	8	passive	Panel-1	Siren Setup	0	3	SIREN_CADENCE

Offset by 4 each time because the cadence data requires four consecutive Data Array locations.

This keyword tells the driver to process incoming Siren Setup messages using these Map Descriptors.

One Map Descriptor for each partition / area combination.

4.4.5.6 Siren Stop / Go.

This example provides a Map Descriptor which stores the sirens state. A value of 1 indicates that the siren is running and a value of 0 indicates that the siren has been stopped. One Map Descriptor is required per panel. The siren state is sent as an unsolicited message from the panel.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offse	Function	Node_Name	Ilgx_Func
Panel-1-Sstate-MD1	SIREN_STATE	0	passive	Panel-1	Siren State

4.4.5.7 Touchpad Display

Touchpad display messages are sent by the panel when the data changes as well as in response to a poll for a refresh of dynamic data.

The response contains a message type and a number of text/display tokens. The token list is a variable length list of data terminated by a zero value. Thus sufficient Data Array space must be left to allow all the tokens to be stored. If you do not care about the tokens then do not specify the secondary array and the driver will discard the token data. Use one Map Descriptor for each partition/area combination.

Table 6.6 provides a list of all the tokens.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Partition	Area	DA_Byte_Name
Panel-1-Tpad-MD1	TOUHPAD_DATA	0	passive	Panel-1	Touchpad	0	1	
Panel-1-Tpad-MD1	TOUHPAD_DATA	0	passive	Panel-1	Touchpad	0	1	TOUCHPAD_TOKENS

Token data is stored in this array. The data is variable length. The driver fills the Data Array starting at the location specified by the Data\_Array\_Offset. The first zero element value element indicates the end of the list of tokens.

No secondary array is specified and the token data is discarded.

4.4.5.8 User Lights.

In this example a Map Descriptor is defined to store incoming User Light data from the Interlogix Panel. One Map Descriptor is required per partition/area/source type combination. User Light messages are generated by 4 source types – Bus Devices – Phones – Zones – Systems. The light number and light state data is stored using one value. Value = Light\_Number \* 100 + Light State.

Light Number value: 0 = all lights 1-32 specific light 0 = off

Light State values: 1 = on 3 = bright 5 = dim

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	Ilgx_Func	Data_Type	Partition	Area	Address	Length
Panel-1-Ulight-MD1	USER_LIGHTS	0	Passive	Panel-1	User Lights	Devices	0	1	1	100
Panel-1-Ulight-MD2	USER_LIGHTS	0	Passive	Panel-1	User Lights	Zones	0	1	1	100
Panel-1-Ulight-MD3	USER_LIGHTS	0	Passive	Panel-1	User Lights	Phones	0	1	1	100
Panel-1-Ulight-MD4	USER_LIGHTS	0	Passive	Panel-1	User Lights	Systems	0	1	1	100

Keyword tell driver to process messages sent by the panel that contain user light data.

The data type parameter tells the driver which source type to process using each Map Descriptor.

If you are only interested in data from one source then you only need a Map Descriptor for that source type. Data from other source types will then be discarded.

One Map Descriptor per partition / area combination

Address and length define the range of device / zone ... numbers that can be processed using the Map Descriptor.



4.4.5.9 Zone Data.

Zone data is different from Zone Status although both message types contain zone status information. A zone data message is a response to an equipment poll for zone data. The zone data message contains additional data in that the zone text tokens and the zone type are also provided.

In this example Map Descriptors are provided which are used to process responses from two partitions each of which has 2 areas. A maximum of 100 zones has been defined for each area. The Array\_Offset is adjusted for each Map Descriptor to place all the incoming zone data in one Data Array called ZONE\_STATUS. Messages for other partition/area combinations will be ignored unless additional Map Descriptors are defined. The zone stats number indicates the zone state – consult the Interlogix panel documentation for current definitions. (bit 1: 1 = faulted bit 2: 1 = Alarm bit 3: 1 = Trouble bit 4: 1 = Bypassed ) Note that the zone data message cannot report a zone's tripped status. Bit Zero will always be zero.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	llgx_Func	Partition	Area	Address	Length	DA_Byte_Name	DA_Bit_Name
Panel-1-Zone-MD1	ZONE_STATUS	0	Passive	Panel-1	Zone Data	0	0	1	100	ZONE_TYPES	ZONE_TOKENS
Panel-1-Zone-MD2	ZONE_STATUS	100	Passive	Panel-1	Zone Data	0	1	1	100		

Zone status is stored here. Bit zero is always off.

One Map Descriptor for each partition / area combination.

Address & length define the range of zone numbers that this Map Descriptor can be used to process.

Map Descriptor 1 has 2ndry storage arrays defined. The Zone type and zone tokens are stored in these 2ndary arrays.

MapDesc2 doesn't have 2ndary arrays defined so the zone type and tokens are discarded.

Don't let the names of these parameters mislead you. The arrays can be of any data type, FLOAT, UINT16 ....

If you do store the zone token data then use one Map Descriptor per zone because the token list is variable length. Leave enough space for the max number of tokens.

4.4.5.10 Partition Data.

In this example Map Descriptors are provided to store partition data. The panel sends partition data messages in response to an equipment list poll. The response contains text tokens. If you define a Data Array for 2ndary storage then the partition text tokens will be stored too. Use one Map Descriptor per partition / area combination. Consult the Interlogix Panel manual for current information on arming levels and text tokens (table 6.5). Advent Panels (0 = Zone Test 1=Off 2=Home/Perimeter 3=Away/Full 4=Night 5=Silent ) Concord Panels (1=Off 2=Stay 3=Away 8=Phone Test 9=Sensor Test )

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	Node_Name	llgx_Func	Partition	Area	DA_Byte_Name
Panel-1-partn-MD1	PART_ARMING	0	Passive	Panel-1	Partition Data	0	0	PART_TOKENS
Panel-1-partn-MD2	PART_ARMING	20	Passive	Panel-1	Partition Data	0	1	PART_TOKENS
Panel-1-partn-MD3	PART_ARMING	40	Passive	Panel-1	Partition Data	0	2	PART_TOKENS

20 Data Array locations have been left for each Map Descriptor. This means that if more than 19 tokens are returned for each partition/area then the list from one partition/are will overflow into the data area for the next.

One Map Descriptor per partition / area combination.

No 2ndry storage Data Array is defined for ..MD3 (Area2). This means that if the text tokens for this partition/area will be discarded.

IN Map Descriptors 1 & 2 tokens will be stored in consecutive array locations starting at location Data\_Array\_Offset. The first zero value indicates the end of the list of tokens.

#### 4.4.5.11 Device Data

Superbus Devices are numbered by the factory. Each device has a unique number even if the equipment type is the same. This means that when you define Map Descriptors for this data you will need to change the Map Descriptor if you ever replace field equipment

In this example 'mapdesc1' stores data for devices 451255, 451256, 45127 (address is 451255 and length is 3). The device state 0=Ok 1=Failed is stored. One element of DA\_01 is used for each device. It is likely, given the values of the device numbers (6-8 digits) that you will need one Map Descriptor for each device.

If you wish to store the tokens associated with each device then you can only have one Map Descriptor per device. Specify the DA\_Bit\_Name and the driver stores the tokens, terminating the list with a zero. If there are 20 tokens then 21 elements of the array are used. The device state is stored as described above.

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	node_name	ilgx_func	Partition	Area	Group	Address	length
Mapdsec1	DA_01	0	passive	Node_A	Device Data	3	1	0	451255	3

Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	node_name	ilgx_func	da_bit_name	Partition	Area	Group	Address	length
MapDesc2	DA_01	0	passive	Node_A	Device Data	DB_01	3	1	0	451255	1

Once you specify an array for token storage you must set the length to 1. i.e. One device per Map Descriptor.

Space successive Map Descriptors to leave enough space for the token lists.

Device addresses are set by the factory and are large multi-digit numbers. They are as unique as a serial number. No two devices have the same number.

4.4.5.12 Superbus Device Capability Data.

This data is sent by the panel on power-up and in response to an equipment list poll. A capability number for each device is reported. Some devices have additional optional capability information. This is stored in a secondary array (if specified. ) See the previous example and chapter 6.8 for information in device numbers.

In this example the capability of device 46101 is stored in the array called DA\_DEVCAP and the optional capability is stored in DA\_DEVCAPOP. The meaning of the capability number is defined in Chapter 6.

Map_Descriptors											
Map_Descriptor_Name	Data_Array_Name	Data_Array_Offset	Function	node_name	ilgx_func	DA_Byte_Name	Partition	Area	Group	Address	length
MapDesc2	DA DEVCAP	0	Passive	Node_A	Device Capabilit	DEVCAPOP	3	1	0	46101	1

## 5 CONFIGURING THE FIELDSEVER AS A INTERLOGIX ADVENT SERIAL DRIVER SERVER

For a detailed discussion on FieldServer configuration, please refer to the instruction manual for the FieldServer. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See “.csv” files on the driver diskette).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a Interlogix Advent Serial Driver Client.

**The driver is capable of acting as a limited function Interlogix server but this capability has been developed for testing and simulation. Contact Fieldserver if you require additional information on the server capability**

### 5.1 Abandoned Messages

Every incoming message will be parsed and then a Map Descriptor will be sought to use to store the incoming data contained in the message. If the driver cannot find a Map Descriptor to use for storage, the message is abandoned. This event will be recorded as a statistic which can be seen using RUI Debug and if you have exposed the driver statistics, the statistic will be available in a Data Array which can be monitored by a downstream device.

In addition, an error message will be produced for the error log. Only one occurrence of this message will be logged so that the log is not filled with repetitive messages.

```

ILGX:#4 Err. Incoming data is being abandoned.

Subsequent message suppressed!

Abandoned Message (followed by a hex dump)
    
```

Section 5.3 describes the driver stats that need to be implemented in order to capture abandoned messages. The 1<sup>st</sup> four elements of the stats array provide information on the most recent message to be abandoned. The driver never clears these elements and only the most recent event is shown

**Table 6.1.1 Abandoned Message Indication**

Stats Array Element	Contents
0	Type of message being abandoned. (See Table 6.1.2)
1	Partition
2	Area
3	Zone / Device / Output / User #

**Table 6.1.2 Abandoned Message Types**

No	Description	No	Description
1	Panel Type	16	Lights State Cmd
2	Auto Event Lost	17	User Lights Cmd
3	Clr Auto Dyn Image	18	Keyfob Cmd
4	Zone Status	19	Rqst Equip List
5	Arming Level	20	Send Single Equip List
6	Entry Delay	21	Send Zone Data
7	Alarm Trouble	22	Send Partition Data
8	Siren Setup	23	Send Superbus Data
9	Siren Synch	24	Send Superbus Capabil
10	Siren Go	25	Send Output Data
11	Touchpad Display	26	Send User Data
12	Siren Stop	27	Send Schedule Data
13	Feature State	28	Send Scheduled Event
14	Temperature	29	Send Light To Sensor
15	Time Date	31	Keypress

Once you have determined the type of message being abandoned it is a simple matter to construct a Map Descriptor in the CSV file which can be used to process the abandoned message.

**Example:**

Stats Array Element	Contents
0	7
1	0
2	1
3	11018649

The value 7 in the first element indicates that an alarm/trouble message was ignored. Therefore an alarm/trouble Map Descriptor needs to be constructed. See the example in section 4.4.5.4 of this manual.

The partition number is zero (element 1), the area number is 1 (element 2) and the device address is 11018649 (element 3)

Therefore a suitable Map Descriptor could be:

Map_Desc_Name	Data_Array_Name	Data_Offset	Function	Node_Name	Ilgx_Func	Data_Type	Partition	Area	Address	Length
Pan-1-Alrm-MD1	DEV_ALARMS	0	passive	Panel-1	Alarm	Devices	0	1	11018649	1

Read section 0 on device address to understand why a device could have an address of 11018649

## 5.2 Poll Timing Issues

Polling for data has some serious timing consequences. It is not possible to know in advance how many messages will be sent in response to a poll so it is important to adjust the timeout and poll scan interval by experimentation.

When the driver polls a panel for **equipment lists** the panel sends a final message saying that it has finished sending the equipment list. The driver uses this to release the poll Map Descriptor and to process the next Map Descriptor. The number of messages that may be obtained in response is large and it is suggested that these poll's be done infrequently and that the timeout is set to many seconds. Timeouts of more than 20 seconds have been noted when communicating with Interlogix panels. Ensure that the scan interval is greater than the timeout.

When the driver polls for a **dynamic data refresh** there is no way for the driver to know when the response is complete (depends on active alarm states, zones ...). Thus the driver releases the Map Descriptor and processes the next one as soon as the poll message has been acknowledge by the panel. The consequence of this is that the scan interval must be set at a lower rate than the driver appears to be capable of processing.

## 5.3 Driver Stats

The statistics recorded by the Interlogix Serial Driver are slightly different from the way that Fieldserver driver normally records statistics. This difference arises from the fact that this driver is not a simple poll response driver.

A single poll can generate a large number of response messages. Each poll is recorded as a PLC\_READ/WRITE\_MSG for the Map Descriptor that generates the poll. The bytes received / transmitted are recorded at the connection level and are not updated for the Map Descriptor.

The driver does not record any stats for passive Map Descriptors use for data storage. This driver can expose these and additional statistics by writing data to a Data Array. A special Map Descriptor is required. The driver recognizes the Map Descriptor by its name which must be "**ILGX-stats**".

The following example shows how this special Map Descriptor can be configured.

Nodes			
Node_name		, Protocol	
null_node		, ilgx	
Data_Arrays			
Data_Array_Name		, Data_Format	, Data_Array_Length
ILGX_STATS		, UINT32	, 330
Map_Descriptors			
Map_Descriptor_Name		, Data_Array_Name	, Node_name
ilgx-stats		, ILGX_STATS	, null_node

When the driver sees this Map Descriptor it uses the Data Array ILGX\_STATS (in this example) to store driver specific statistics. Only one of these Map Descriptors may be specified (per tier) per FieldServer.

The driver stores the following stats for each port. The offset into the Data Array can be found by multiplying the port number by 30.



Number	Statistic
0	Not Used. Base offset into array is found by multiplying port number by 30.
1	DRV_DLL_CLIENT_SENDS_MSG
2	DRV_DLL_CLIENT_SENDS_ACKNAK
3	DRV_DLL_CLIENT_SENDS_BYTES
4	DRV_DLL_SERVER_SENDS_MSG
5	DRV_DLL_SERVER_SENDS_ACKNAK
6	DRV_DLL_SERVER_SENDS_BYTES
7	DRV_DLL_CLIENT_RCVS_MSG
8	DRV_DLL_CLIENT_RCVS_BYTES
9	DRV_DLL_SERVER_RCVS_MSG
10	DRV_DLL_SERVER_RCVS_BYTES
11	DRV_DLL_TIMEOUT
12	DRV_DLL_ERROR
13	DRV_DLL_ERROR_CODE
14	ILGX_STAT_DRV_ACT_NODE_OFFLINE_R
15	ILGX_STAT_DRV_ACT_NO_MAPDESC_R
16	ILGX_STAT_DRV_ACT_NO_DATA_R
17	ILGX_STAT_DRV_ACT_NO_NODE_R
18	ILGX_STAT_DRV_ACT_NAK_R
19	ILGX_STAT_DRV_ACT_NORMAL_R
20	ILGX_STAT_CD_UPD_CHECK
21	ILGX_STAT_CD_UPD_FUNCTION
22	ILGX_STAT_CD_UPD_NO_START
23	ILGX_STAT_CD_UPD_PROTO
24	ILGX_STAT_CD_UPD_NO_SLAVE
25	ILGX_STAT_CD_UPD_IC_TIMEOUT
26	ILGX_STAT_CD_UPD_MSG_IGNORED
27	ILGX_STAT_CD_UPD_NAK
28	ILGX_STAT_CD_UPD_STREAMING
29	ILGX_STAT_CANNOT_PARSE_MSG_BAD_FN

5.4 Data Variables & Data Storage Defined.

The Panel generates messages and responses that are categorized in Table 1 of this manual. The notes in this section describe the data variables that are contained in these messages. The notes also describe how these variables are stored.

**Table 6.4 Data Variables & Data Storage**

Data Variable.	Storage / Extraction	
	Default Array	2ndry/tertiary Storage
	Storage location is always referenced to the Data_Array_Offset specified in the Map Descriptor.	Da_byte_name or DA_Bit_Name  Storage location is always referenced to the Data_Array_Offset specified in the Map Descriptor.
<b>Arming Level</b>		
Make one MapDesc per partition. Length determines the number of areas covered. The area and length are used to determine the relative storage position. If the partition is different then a new MapDesc is required.		
Arming Level	Location is based on area number relative to MapDesc area and offset	
User		Same relative location as Arming Level
<b>Automation Event Lost</b>		
Make one MapDesc per device.		
Counts number of this type of message received	0	
Panel Type	0	
Hardware Rev	1	
Software Rev	2	
Serial # Part 1	3	
Serial # Part 2	4	
Serial # Part 3	5	
Serial # Part 4	6	
<b>Zone Status</b>		
Set the address equal to the start zone number and the length equal to the number of zones. If the area number or partition is different then a new MapDesc is required.		
Zone Status	Based on zone number relative to address.	
<b>Zone Data</b>		
Set the address equal to the start zone number and the length equal to the number of zones. If the area number or partition is different then a new MapDesc is required.		
Zone Status	Based on zone number relative to address.	
Zone Type		Based on zone number relative to address.

Data Variable.	Storage / Extraction	
	Default Array	2ndry/tertiary Storage
Zone Tokens		DA_Bit_Name : Based on zone number relative to address and then as many consecutive locations as there are tokens (depends on panel config.) Thus when using this make one zone per MapDesc.
<b>Alarm Trouble</b>		
<p>Set the address equal to the start src number and the length equal to the number of devices.            If the area number or partition is different then a new MapDesc is required.            The data type of the MapDesc is important too.            Use Zones, Devices to differentiate storage for zone / devices / system / Phone            Default is device is no data type is specified.            Device numbers are unique for each device used. They can be considered as the equivalent of a manufacturer's serial number. Thus two identical hardware configurations will require different CSF file configurations as the device numbers will be different. Device numbers are large numbers in the range 0-16,777,216. It may, thus, be necessary to use many Map Descriptors to process alarms to ensure that your arrays are not too sparse.</p>		
Alarm Type = 100 * General_type + Specific_type  Eg if GT=1 and ST=2 Then value stored is 102.	Based on zone number relative to address.	
Event Specific Data		Same relative location as alarm type.
<b>Delay Entry</b>		
<p>Make one MapDesc per partition. Length determines the number of areas covered. The area and length are used to determine the relative storage position. If the partition is different then a new MapDesc is required.</p>		
Delay Time	Location is based on area number relative to MapDesc area and offset	
Delay Flags		Same relative location as delay time.
<b>Siren Setup</b>		
One MapDesc for each partition / area combination.		
Repeats	Location based on MapDesc offset	
Cadence Byte1		MapDesc offset + 0
Cadence Byte2		MapDesc offset + 1
Cadence Byte3		MapDesc offset + 2
Cadence Byte4		MapDesc offset + 3
Siren Go/Stop One MapDesc per node.		
State (1=Go 0=Stop)	Location based on MapDesc offset	
<b>Siren Synch</b>		
One MapDesc per node.		
Count of how many times this message has been	Location based on MapDesc offset	

Data Variable.	Storage / Extraction	
	Default Array	2ndry/tertiary Storage
received.		
<b>Partition Data</b> One mapdesc per partition per area.		
Arming Level	Location based on MapDesc offset	
Tokens		DA_Bit_Name : Based on MapDesc offset and then as many consecutive locations as there are tokens (depends on panel config.)
<b>Superbus Device Data</b> One MapDesc per partition per area. Use address for 1st device and length to define the number of devices to be processed using this MapDesc.		
Device State		Based on device number relative to address.
Tokens		DA_Bit_Name : Based on device number relative to address and then as many consecutive locations as there are tokens (depends on panel config.) Thus when using this make one device per MapDesc.
<b>Superbus Device Capability</b> One MapDesc per range of devices per node. Use address and length to define the devices that will be processed using this MapDesc.		
Capability Number	Based on device number relative to address.	
Option Capability Data		Based on device number relative to address.
<b>Output Data</b> One MapDesc per range of devices per node. Use address and length to define the devices that will be processed using this MapDesc.		
Output State	Based on device number relative to address.	
ID Bytes		DA_Byte_Name: Based on device number relative to address. Uses 5 consecutive locations)
Tokens		DA_Bit_Name : Based on device number relative to address and then as many consecutive locations as there are tokens (depends on panel config.) Thus when using this make one device per MapDesc.
<b>User Data</b> One MapDesc per user. The user number is specified in the address field.		
Tokens	Based on user number relative to address and then as many	

Data Variable.	Storage / Extraction	
	Default Array	2ndry/tertiary Storage
	consecutive locations as there are tokens (depends on panel config.)	
<b>Schedule Data</b>		
One MapDesc per partition per area. Use address to define the schedule that will be processed using this MapDesc. One schedule number per MapDesc.		
Start Hour	0	
Start Minute	1	
Stop Hour	2	
Stop Min	3	
Days of weeks	4	
Schedule Validity	5	
<b>Feature State</b>		
One MapDesc per partition per area.		
Feature Bytes	MapDesc offset and as many consecutive locations as there are features.  Zero ends list.	
<b>Temperature</b>		
One MapDesc per partition per area.		
Temp in °F	0	
Set point		0
<b>Date &amp; Time</b>		
One MapDesc per node		
Hour	0	
Minute	1	
Month	2	
Day	3	
Year	4	
<b>Touchpad Display</b>		
One MapDesc per partition per area		
Display Tokens	Based on MapDesc offset and then as many consecutive locations as there are tokens	
<b>Lights State</b>		
One MapDesc per partition per area		
Light state bits	0 (Based on MapDesc offset)	
<b>User Lights</b>		
One MapDesc per partition per area per range of source numbers. Use address and length to define the range of light source numbers		
(lo Byte) Light Number + (hi Byte) light State	Based on source number relative to MapDesc address & offset	
Source Type		Based on source number relative to

Data Variable.	Storage / Extraction	
	Default Array	2ndry/tertiary Storage
		MapDesc address & offset
<b>Keyfob</b>		
One MapDesc per partition & area. Range of zone to be processed is defined using address and length.		
Key Code	Based on zone number relative to MapDesc address & offset	
<b>Clear Dynamic Data</b>		
Can have many of these per node. When a clear dynamic data message is received then the all array elements starting at offset for length equal to the MapDesc length will be set to zero.		
Zeroized Data	Based on address and length.	

### 5.5 Alarm / Trouble General & Specific Types

The following list should be verified using current Interlogix documentation. The content has been extracted from the Interlogix specifications.

The underlined> events apply only to Advent.

Alarm (General Type = 1)		
1.0	Unspecified	
1.1	Fire	
1.2	Fire Panic	
1.3	Police	
1.4	Police Panic	
1.5	Medical	
1.6	Medical Panic	
1.7	Auxiliary	
1.8	Auxiliary Panic	
1.9	Tamper	
1.10	No Activity	
1.11	Suspicion	
1.12	Not used	
1.13	Low Temperature	
1.14	High Temperature	
1.15	Keystroke Violation (Touchpad Tamper)	
1.16	Duress	
1.17	Exit Fault	
1.18	Explosive Gas	
1.19	Carbon Monoxide	
1.20	Environmental	
1.21	Latchkey	ES = user number
1.22	Equipment Tamper	
1.23	Holdup	
1.24	Sprinkler	
1.25	Heat	
1.26	Siren Tamper	
1.27	Smoke	
1.28	Repeater Tamper	
1.29	Fire Pump Activated	
1.30	Fire Pump Failure	
1.31	Fire Gate Valve	

1.32	Low CO2 Pressure	
1.33	Low Liquid Pressure	
1.34	Low Liquid Level	
1.35	Entry/Exit	
1.36	Perimeter	
1.37	Interior	
<b>Alarm Cancel (General Type = 2)</b>		
2.xx	xx = same as General Type = 1	
<b>Alarm Restoral (General Type = 3)</b>		
3.xx	xx = same as General Type = 1.	
<b>Fire Trouble (General Type = 4)</b>		
4.0	Unspecified	
4.1	Hardwire	
4.2	Ground Fault	
4.3	Device	
4.4	Supervisory	
4.5	Low Battery	
4.6	Tamper	
4.7	SAM	
4.8	Partial Obscurity	
4.9	Jam	
4.10	Zone AC Fail	
4.11	n/u	
4.12	NAC Trouble	
4.13	Analog Zone Trouble	
4.14	Fire Supervisory	
4.15	Pump Fail	
4.16	Fire Gate Valve Closed	
4.17	CO2 Pressure Trouble	
4.18	Liquid Pressure Trouble	
4.19	Liquid Level Trouble	
<b>Fire Trouble Restoral (General Type = 5)</b>		
5.yy	yy = same as General Type = 4.	
<b>Non-Fire Trouble (General Type = 6)</b>		
6.yy	yy = same as General Type = 4.	
<b>Non-Fire Trouble Restoral (General Type = 7)</b>		
7.yy	yy = same as General Type = 4	
<b>Bypass (General Type = 8)</b>		
8.0	Direct Bypass	ES = user number
8.1	Indirect Bypass	
8.2	Swinger Bypass	
8.3	Inhibit	
<b>Unbypass (General Type = 9)</b>		
9.0	Direct Bypass	ES = user number
9.1	Indirect Bypass	ES = user number
9.2	Swinger Bypass	
9.3	Inhibit	
<b>Opening (General Type = 10)</b>		
10.0	Normal Open/Close	ES = user number
10.1	Early Open/Close	ES = user number
10.2	Late Open/Close	ES = user number
10.3	Fail To Open/Close	
10.4	Open/Close Exception	ES = user number

10.5	Open/Close Extension	ES = user number
10.6	Open/Close Using Keyfob/Keyswitch	
10.7	Scheduled Open/Close	
10.8	Remote Open/Close	ES = user number
<b>Closing (General Type = 11)</b>		
11.zz	zz = same as General Type 10	
<b>Partition Configuration Change (General Type = 12)</b>		
12.0	User Access Code Added	ES = user number
12.1	User Access Code Deleted	ES = user number
12.2	User Access Code Changed	ES = user number
12.3	User Access Code Expired	ES = user number
12.4	User Code Authority Changed	
12.5	Authority Levels Changed	
12.6	Schedule Changed	
12.7	Arming or O/C Schedule Changed	
12.8	Zone Added	
12.9	Zone Deleted	
<b>Partition Event (General Type = 13)</b>		
13.0	Schedule On	
13.1	Schedule Off	
13.2	Latchkey On	
13.3	Latchkey Off	
13.4	Smoke Detectors Reset	
13.5	Valid User Access Code Entered	
13.6	Arming Level Changed	ES = user number
13.7	Alarm Reported	
13.8	Agent Release	
13.9	Agent Release Restoral	
13.10	Partition Remote Access	
13.11	Keystroke Violation in Partition	
13.12	Manual Force Arm	
13.13	Auto Force Arm	
13.14	Auto Force Arm Failed	
13.15	Arming Protest Begun	
13.16	Arming Protest Ended	
<b>Partition Test (General Type = 14)</b>		
14.0	Manual Phone Test	
14.1	Auto Phone Test	
14.2	Auto Phone Test with existing trouble	
14.3	Phone Test OK	
14.4	Phone Test Failed	
14.5	User Sensor Test Started	
14.6	User Sensor Test Ended	
14.7	User Sensor Test Completed	
14.8	User Sensor Test Incomplete	
14.9	user Sensor Test Trip	
14.10	Installer Sensor Test Started	
14.11	Installer Sensor Test Ended	
14.12	Installer Sensor Test Completed	
14.13	Installer Sensor Test Incomplete	
14.14	Installer Sensor Test Trip	
14.15	Fire Drill Started	
<b>System Trouble (General Type = 15)</b>		



15.0	Bus Receiver Failure	
15.1	Bus Antenna Tamper	
15.2	Main Low Battery	
15.3	SnapCard Low Battery	
15.4	Module Low Battery	
15.5	Main AC Failure	
15.6	SnapCard AC Failure	
15.7	Module AC Failure	
15.8	Aux. Power Failure	
15.9	Bus Shutdown	
15.10	Bus Low Power Mode	
15.11	Phone Line 1 Failure	
15.12	Phone Line 2 Failure	
15.13	Remote Phone Tamper	
15.14	Watchdog Reset	
15.15	RAM Failure	
15.16	Flash Failure	
15.17	Printer Error	
15.18	History Buffer (almost) Full	
15.19	History Buffer Overflow	
15.20	Report Buffer Overflow	
15.21	Bus Device Failure	
15.22	Failure To Communicate	
15.23	Long Range Radio Trouble	
15.24	Module Tamper Trouble	
15.25	Un-enrolled Module Trouble	
15.26	Audio Output Trouble	
15.27	Analog Module Trouble	
15.28	Cell Module Trouble	
15.29	Buddy 1 Failure	
15.30	Buddy 2 Failure	
15.31	Buddy 3 Failure	
15.32	Buddy 4 Failure	
15.33	SnapCard Trouble	
15.34	Analog Loop Short	
15.35	Analog Loop Break	
15.36	Analog Address 0	
15.37	Un-enrolled Analog Head	
15.38	Duplicate Analog Head	
15.39	Analog Module Initializing	
	Microphone Switch Trouble	
	Microphone Trouble	
	Microphone Wiring Trouble	
	JTECH Premise Paging Trouble	
	Voice Siren Tamper Trouble	
	Microburst Transmit Failure	
	Microburst Transmit Disabled	
	Microburst Module Failure	
	Microburst Not In Service	
	Automation Supervisory Trouble	
	Microburst Module Initializing	
	Printer Paper Out Trouble	
<b>System Trouble Restoral (General Type = 16)</b>		

16.xx	xx = same as General Type = 15	
<b>System Configuration Change (General Type = 17)</b>		
17.0	Program Mode Entry	
17.1	Program Mode Exit Without Change	
17.2	Program Mode Exit With Change	
17.3	Downloader Session Start	
17.4	Downloader Session End Without Change	
17.5	Downloader Session End With Change	
17.6	Downloader Error	
17.7	Downloader Connection Denied	
17.8	Date/Time Changed	
17.9	Module Added	
17.10	Module Deleted	
17.11	Speech Tokens Changed	
17.12	Code Changed	
17.13	Panel First Service (cold reset)	
17.14	Panel Back In Service (warm reset)	
17.15	Installer Code Changed	
<b>System Event (General Type = 18)</b>		
18.0	Callback Requested	
18.1	Output Activity (not used, see 18.5 & 18.6)	
18.2	Buddy Reception	
18.3	Buddy Transmission Request	
18.4	History Buffer Cleared	
18.5	Output On	ES = output number
18.6	Output Off	ES = output number

## 5.6 Text Tokens

Token	Token Value	Token	Token Value
0	00	GARAGE	70
1	01	GALLERY	71
2	02	GOODBYE	72
3	03	GROUP	73
4	04	HALL	74
5	05	HEAT	75
6	06	HELLO	76
7	07	HELP	77
8	08	HIGH	78
9	09	HOURLY	79
#	0C	HOUSE	7A
:	0D	IMMEDIATE	7B
/	0E	IN SERVICE	7C
?	0F	INTERIOR	7D
.	10	INTRUSION	7E
A	11	INVALID	7F
B	12	IS	80
C	13	KEY	81
D	14	KITCHEN	82

Token	Token Value	Token	Token Value
E	15	LAUNDRY	83
F	16	LEARN	84
G	17	LEFT	85
H	18	LIBRARY	86
I	19	LEVEL	87
J	1A	LIGHT	88
K	1B	LIGHTS	89
L	1C	LIVING	8A
M	1D	LOW	8B
N	1E	MAIN	8C
O	1F	MASTER	8D
P	20	MEDICAL	8E
Q	21	MEMORY	8F
R	22	MIN	90
S	23	MODE	91
T	24	MOTION	92
U	25	NIGHT	93
V	26	NORTH	94
W	27	NOT	95
X	28	NUMBER	96
Y	29	OFF	97
Z	2A	OFFICE	98
Space	2B	OK	99
'	2C	ON	9A
-	2D	OPEN	9B
_	2E	OPENING	9C
*	2F	PANIC	9D
AC POWER	30	PARTITION	9E
ACCESS	31	PATIO	9F
ACCOUNT	32	PHONE	A0
ALARM	33	POLICE	A1
ALL	34	POOL	A2
ARM	35	PORCH	A3
ARMING	36	PRESS	A4
AREA	37	QUIET	A5
ATTIC	38	QUICK	A6
AUTO	39	RECEIVER	A7
AUXILIARY	3A	REAR	A8
AWAY	3B	REPORT	A9
BACK	3C	REMOTE	AA
BATTERY	3D	RESTORE	AB
BEDROOM	3E	RIGHT	AC
BEEPS	3F	ROOM	AD
BOTTOM	40	SCHEDULE	AE

Token	Token Value	Token	Token Value
BREEZEWAY	41	SCRIPT	AF
BASEMENT	42	SEC	B0
BATHROOM	43	SECOND	B1
BUS	44	SET	B2
BYPASS	45	SENSOR	B3
BYPASSED	46	SHOCK	B4
CABINET	47	SIDE	B5
CANCELED	48	SIREN	B6
CARPET	49	SLIDING	B7
CHIME	4A	SMOKE	B8
CLOSET	4B	Sn	B9
CLOSING	4C	SOUND	BA
CODE	4D	SOUTH	BB
CONTROL	4E	SPECIAL	BC
CPU	4F	STAIRS	BD
DEGREES	50	START	BE
DEN	51	STATUS	BF
DESK	52	STAY	C0
DELAY	53	STOP	C1
DELETE	54	SUPERVISORY	C2
DINING	55	SYSTEM	C3
DIRECT	56	TAMPER	C4
DOOR	57	TEMPERATURE	C5
DOWN	58	TEMPORARY	C6
DOWNLOAD	59	TEST	C7
DOWNSTAIRS	5A	TIME	C8
DRAWER	5B	TIMEOUT	C9
DISPLAY	5C	TOUCHPAD	CA
DURESS	5D	TRIP	CB
EAST	5E	TROUBLE	CC
ENERGY SAVER	5F	UNBYPASS	CD
ENTER	60	UNIT	CE
ENTRY	61	UP	CF
ERROR	62	VERIFY	D0
EXIT	63	VIOLATION	D1
FAIL	64	WARNING	D2
FAILURE	65	WEST	D3
FAMILY	66	WINDOW	D4
FEATURES	67	MENU	D5
FIRE	68	RETURN	D6
FIRST	69	POUND	D7
FLOOR	6A	HOME	D8
FORCE	6B	CARRIAGE RETURN	F9
FORMAT	6C	PSEUDO SPACE	FA

Token	Token Value	Token	Token Value
FREEZE	6D	CARRIAGE RETURN	FB
FRONT	6E	BACKSPACE	FD
FURNACE	6F	BLINK NEXT TOKEN	FE

## 5.7 Driver Error Messages

Message	Action
ILGX:#1 FYI. The MapDesc called <%s> is too short.	This Map Descriptor is used to expose driver statistics. Increase the length of this Map Descriptor to 330.
ILGX:#2 FYI. You could have used a MapDesc called <%s> to expose diagnostic info.	You can ignore this message. Read section 5.3 for information on how to expose driver statistics.
*ILGX:#3 Err. Response requires too much buffer. oprt=%u iptr=%u	Monitor the ILGX_STAT_BUFFER_OFLOW statistic
*ILGX:#4 Err. Incoming data is being abandoned.	Read section 5.1 for more information.
ILGX:#5 Err. Light to Sensor data abandoned. No storage method	These functions have not been implemented by the driver. Contact FieldServer to discuss your requirements.
*ILGX:#6 Err. Scheduled Event data abandoned. No storage method	
*ILGX:#7 Err. Schedule data abandoned. No storage method	
*ILGX:#8 Err. Incoming Poll is being Ignored. No response MapDesc's.	The Panel may produce more data than you are interested in. For example you may only be monitoring alarms but the panel may send information about the siren setup or about a zone if the zone status changes. If you do not have Map Descriptors to process every possible unsolicited message the panel can send then you will get this message.
*ILGX:#9 Err. Light to Sensor data. Null response sent.	These functions have not been implemented by the driver. Contact Fieldserver to discuss your requirements.
*ILGX:#10 Err. Scheduled Event data. Null response sent.	
*ILGX:#11 Err. Schedule data. Null response sent.	
*ILGX:#12 Err. Command=%x(h) cant be parsed	The driver has received a command from the panel which it cannot parse. This could be a new function/message produced by the panel that is not supported by the driver or it could arise from a corrupt message. If the error occurs repeatedly notify FieldServer. You can monitor this by checking the statistic ILGX_STAT_CANNOT_PARSE_MSG_BAD_FN (See section 5.3). Although these messages have the same origin and are similar they help identify the source of the problem. You cannot take any corrective action.
*ILGX:#13 FYI. Command=%x(h) SubCmd=%x(h) cant be parsed	
*ILGX:#14 FYI. Command=%x(h) SubCmd=%x(h) cant be parsed.	
ILGX:#15 FYI. Write for simulation only. MapDesc=<%s>	The driver has been developed to act as a client. The server mode is for driver testing and simulation only. Do not use the driver to produce data unless you understand the limitations of the implementation.
ILGX:#16 FYI. Slave is a 'Concord' Panel.	These messages are produced as appropriate for each tier of

\* Error messages that are produced once only and then suppressed even if the error recurs. The driver implements some messages like this so that they do not fill the error buffer as they may be produced on every poll.

Message	Action
ILGX:#17 FYI. Slave is a 'Concord Express' Panel.	the FieldServer on which the driver is loaded. They are used to inform you how the driver has been configured when in server mode. No action is necessary in response to these messages.
ILGX:#18 FYI. Slave is NOT an 'Advent' Panel.	
ILGX:#19 FYI. Slave is an 'Advent' Panel.	
ILGX:#20 Err. Changed data type to zone for MapDesc=<%s>	The driver does not understand the data type specified in the CSV file. It has changed the data type to Zone and will continue to operate but this may result in unexpected data being written to the Data Arrays. Review and correct the CSV file, then reset the FieldServer.
ILGX:#21 FYI. Will read dynamic data for MapDesc=<%s>	An invalid 'Option' has been specified. The option may be omitted or set to Dynamic, Static or Equipment. Review and correct the CSV file and then reset the FieldServer. If you do not change the CSV file the driver will operate as if the option had been set to Dynamic.
ILGX:#22 Err. No passive Map Descriptors.	This message is produced if the CSV file has no Map Descriptors capable of being used to process an incoming message. Having a poll (rdbc) Map Descriptor is not sufficient. One or more passive Map Descriptors must be defined in the CSV file to process unsolicited messages and responses to the poll. Read the manual carefully and then edit your CSV file.
*ILGX: Err.23 Data type no suitable for active mapdesc=<%s>.	The Data Type specified using the Data_Type parameter in the Map Descriptor is invalid. Correct the error and reset the FieldServer. See section 4.4.2 for a list of valid keywords.
ILGX: Err.24 Invalid Function=%x(h) for mapDesc=<%s>	The function you have selected cannot be used with a Map Descriptor when the function is 'write'.
ILGX:25 Err. Unknown ilgx_function for mapdesc=<%s>	The function specified in using the ilgx_func parameters is invalid. Correct the CSV and reset the FieldServer. See section 4.4.2 for a list of valid keywords.
*ILGX:26 Err. No Store! Array too short. Mapdesc=<%s> xxxx	This range of error messages is used when the Data Array specified for storage of incoming data is too short to store all the data. The xxxx part of the message is either blank or is used to indicate if the 2ndary storage array (specified by da_byte/bit_name) is also causing the error. The solution is to increase the length of the Map Descriptor in the CSV file and reset the FieldServer. The specific number of the error message indicates the type of Map Descriptor causing the problem. As the message provides the offending Map Descriptor's name it is not necessary to tabulate the error message numbers separately. Ensure that you have used unique Map Descriptor names.
*ILGX:58 Err. No Store! Array too short. Mapdesc=<%s> xxxx	

\* Error messages that are produced once only and then suppressed even if the error recurs. The driver implements some messages like this so that they do not fill the error buffer as they may be produced on every poll.

## 5.8 Device Addresses

Interlogix manufacture SuperBus Devices with unique addresses. The address is fixed at the time of manufacture. Each device has a unique address, thus identical equipment configurations have different addresses. A consequence of this is that you cannot duplicate configuration (CSV) files even if the hardware configurations are identical.

Device addresses are in the range 0 – 16,777,215.

## 5.9 Device Capabilities

Capability number	Description	Optional data
0x00	Power Supervision	
0x01	Access Control	
0x02	Analog Smoke	
0x03	Audio Listen-In	
0x04	Snapcard Supervision	
0x05	Microburst	
0x06	Dual Phone Line	
0x07	Energy Management	
0x08	Input Zones	Number of inputs
0x09	Phast/Automation/System Manager	
0x0A	Phone Interface	
0x0B	Relay Outputs	Number of outputs
0x0C	RF Receiver	
0x0D	RF Transmitter	
0x0E	Parallel Printer	
0x0F		
0x10	LED Touchpad	
0x11	1-Line/2-Line/BLT Touchpad	
0x12	GUI Touchpad	
0x13	Voice Evacuation	
0x14	Pager	
0x15	Downloadable code/data	
0x16	JTECH Premise Pager	
0x17	Cryptography	
0x18	LED Display	