FieldServer Technologies

---

Driver Manual
(Supplement to the FieldServer Instruction Manual)

# FS-8700-69 Interlogix Automation Module Protocol Driver

# (Supports Advent, Concord and Concord  Express Panels.)

**APPLICABILITY & EFFECTIVITY**

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

# TABLE OF CONTENTS

# 1. Interlogix Advent Serial Driver Description

The Interlogix Advent Serial Driver allows the Communications Bridge to transfer data to and from devices over either RS-232 or RS-485 using Interlogix Advent Serial Driver protocol. The Bridge 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 just like an Interlogix panel.

The following table indicates the types of data that are reported by the panel. The table also shows whether the data is generated by the panel or is obtained as the result of a poll.

| Data Message | Unsolicited | Equipment List Request | Dynamic Data Refresh | Notes |
|---|---|---|---|---|
| Panel Type | U on pwr/comms | F | D | F=2 D=3 |
| Automation Event Lost | U on buffer o/f | | | |
| Clear Automation Dynamic Image | U on pwr/comms | | | |
| 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 | 5 |
| 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 | 1 |
| Temperature | | | D | 3 |
| Time and Date | | | D | 1 |

| | | | | |
|---|---|---|---|---|
| Lights State Command | U on Change | | D | 3 |
| User Lights Command | U on event | | | 2,4 |
| 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 | | 3 |
| Scheduled Event Data | | F, S | | 3 |
| Light to Sensor Attachment | | F, S | | 3 |
| Keypress | Subject to verification | | | |
| | | | | |
| **NOTES** | | | | |
| Not supported by Advent | 1 | | | |
| Advent Only | 2 | | | |
| Concord Only | 3 | | | |
| Concord Express Only | 4 | | | |
| In response to refresh poll if siren active | 5 | | | |
| | | | | |

**<u>Table 1.0</u>**

Table of data that can be communicated using the Interlogix Serial Driver. The table indicates the data that is sent as unsoloicted 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.

## 2. Driver Scope of Supply

### 2.1 Supplied by FieldServer Technologies for this driver

| | |
|---|---|
| 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 |
| | Driver Manual. |

### 2.2 Provided by user

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

## 3. Hardware Connections

The bridge 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

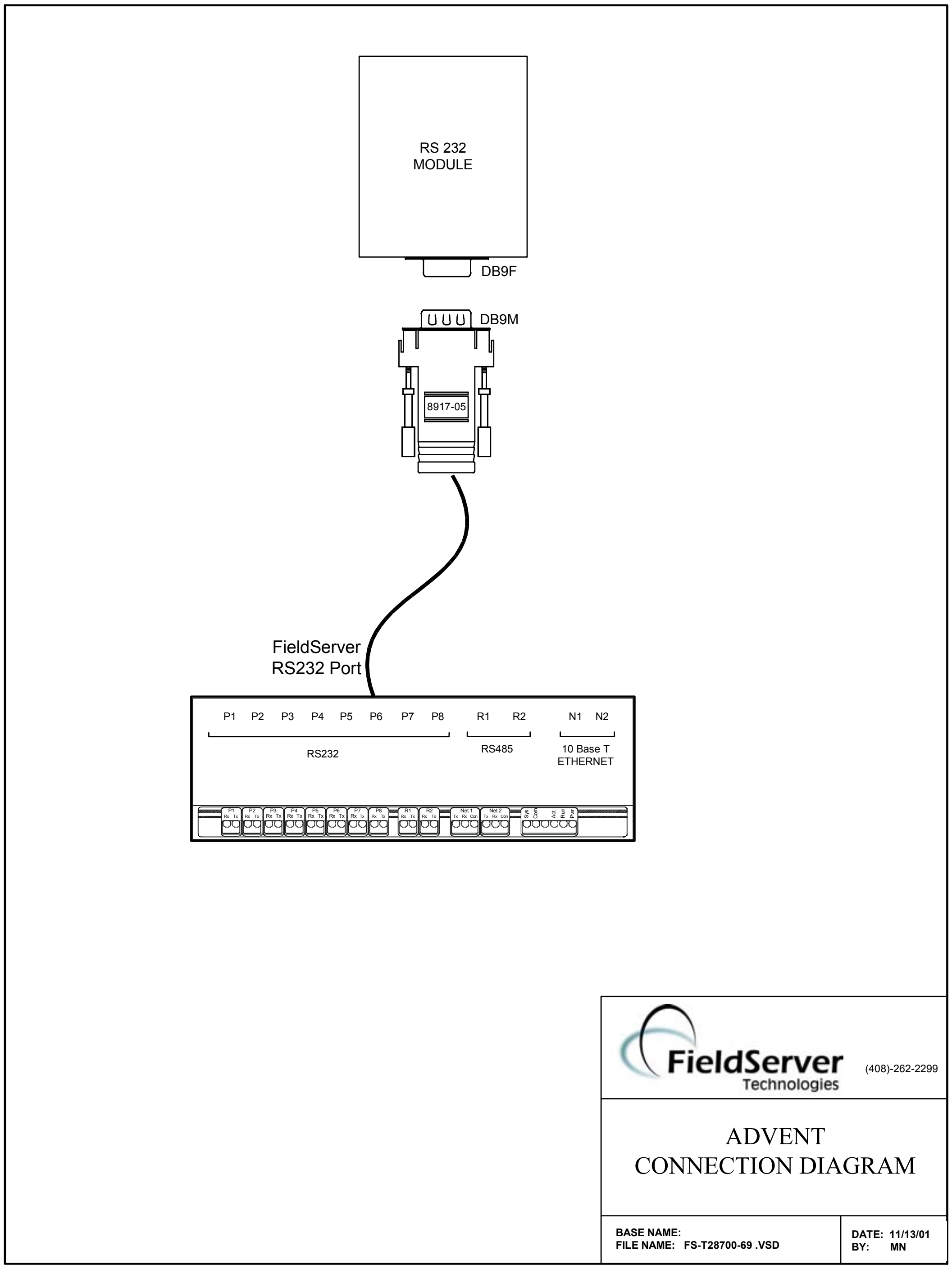

Fig. 3.1

## 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 Bridge as a Interlogix Advent Serial Driver Client

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

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

It is possible to configure the bridge in the following modes

- Passive Client
  The bridge parses and stored unsolicited messages generated by the panel.

- Active Client
  The bridge polls the panel for refreshed dynamic data and/or the panel's equipment list.

- Full Client
  The bridge 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 Connections

| Section Title | | |
|---|---|---|
| Connections | | |
| **Column Title** | **Function** | **Legal Values** |
| Port | Specify which port the device is connected to the FieldServer | P1-P8, R1-R2 |
| Baud* | Specify baud rate | 9600 Note1 |
| Parity* | Specify parity | Odd Note1 |
| Data_Bits* | Specify data bits | **8** Note1 |
| Stop_Bits* | Specify stop bits | **1** Note1 |
| Protocol | Specify protocol used | Ilgx, Advent |
| Handshaking* | Specify hardware handshaking | **None** Note1 |
| Poll Delay* | Time between internal polls | 0-32000 seconds **default 1 second** |

Notes
1. The driver can support all std baud rates & port configurations but at the time of printing, Interlogix panels only support 9600, 8 ,1 Odd

**Example**

```
//    Client Side Connections

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

## 4.3 Client Side Nodes

| 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 | Ilogix, Advent |
| Port | Specify which port the device is connected to the FieldServer | P1-P8, R1-R2 |

**Example**

```
//    Client Side Nodes

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

## 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 descriptor 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. .<br><br>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 block. . Only some map descriptors require that a Address be specified. See the examples that follow this section | 0, 1, 2 .. |

| | | |
|---|---|---|
| Ilgx_function | Used to tell the driver what kind of data to process. | Panel Type<br>Auto Event Loss<br>Zone Status<br>Arming Level<br>Entry Delay<br>Alarm<br>Siren Setup<br>Siren Synch<br>Siren State<br>Touchpad<br>Feature State<br>Temperature<br>Date<br>Lights State<br>User Lights<br>Keyfob<br>Keypress<br>Zone Data<br>Partition Data<br>Superbus Device Data<br>Device Data<br>Superbus Device Capability<br>Device Capability<br>Output Data<br>User Data<br>Schedule Data<br>Scheduled Event<br>Light to Sensor Data<br>Clear Dynamic Data |
| Option* | Used for poll RDBC map descriptors. Tells the driver whether to poll for Dynamic Data or for Equipment Lists | **Dynamic**<br>Static |
| 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 paritiion number is zero then this parameter need not be specified. | **0**,1,2 .. |
| 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. | **0**,1,2 .. |

| | | |
|---|---|---|
| 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 number is zero then this parameter need not be specified. | **0**,1,2 .. |
| DA_Byte_Name* | 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.<br><br>See Table 6.4 for more information on which Panel Data messages use 2ndary data storage.<br><br>Don't let the names of these parameters mislead you. The arrays can be of any data type, FLOAT, UINT16 …. | 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
Poll-1,                  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 bridge 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 the Panel for a List of all its Equipment

*This example poll 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 Poll the 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 aksed 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[1] ,** Zones, Alarms, Devices, Dev. Capabilities, Outputs , Users, Schedules, Scheduled Events, Lights, Phones, Systems, Partitions

Note[1] 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 area. 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 = 0ff 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_LVLS, 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 ,…

### 4.4.6 **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 Time  , 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 rray.

### 4.4.6.1 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-Alrm-MD1  , DEV_ALARMS ,0          , passive , Panel-1   , Alarm   , Devices  , 0   , 1  , 1   , 100  , DEV_ES_DATA
Panel-1-Alrm-MD2  , ZON_ALARMS ,0          , passive , Panel-1   , Alarm   , Zones    , 0   , 1  , 1   , 100  , ZON_ES_DATA
Panel-1-Alrm-MD3  , PHO_ALARMS ,0          , passive , Panel-1   , Alarm   , Phones   , 0   , 1  , 1   , 100  , PHO_ES_DATA
Panel-1-Alrm-MD4  , SYS_ALARMS ,0          , passive , Panel-1   , Alarm   , Systems  , 0   , 1  , 1   , 100  , SYS_ES_DATA
Panel-1-Alrm-MD5  , DEV_ALARMS ,100        , passive , Panel-1   , Alarm   , Devices  , 0   , 2  , 1   , 100  , DEV_ES_DATA
Panel-1-Alrm-MD6  , ZON_ALARMS ,100        , passive , Panel-1   , Alarm   , Zones    , 0   , 2  , 1   , 100  , ZON_ES_DATA
Panel-1-Alrm-MD7  , PHO_ALARMS ,100        , passive , Panel-1   , Alarm   , Phones   , 0   , 2  , 1   , 100  , PHO_ES_DATA
Panel-1-Alrm-MD8  , SYS_ALARMS  ,10        , passive , Panel-1   , Alarm   , Systems  , 0   , 2  , 1   , 100  , SYS_ES_DATA
```

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.

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*100+ST. Thus you must divide by 100 to extract the General Type. See **Table 6.5** for more information.

### 4.4.6.2 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 for3 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.

### 4.4.6.3 Siren Stop / Go.

*This example provides a map descriptor which stores the sirens state. A value of 1 indicates that the siren is running an 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_Offset, Function, Node_Name, Ilgx_Func    ,
Panel-1-Sstate-MD1 , SIREN_STATE      ,0                     , passive , Panel-1     , Siren State  ,
```

### 4.4.6.4 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 lists 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,
Panel-1-Tpad-MD1  , TOUHPAD_DATA ,0                      , passive , Panel-1     , Touchpad   , 0          , 1       ,
```

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

```
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       ,  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.

See table 6.6 for token details.

### 4.4.6.5 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 ad 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 n 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

### 4.4.6.6 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, Ilgx_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.

### 4.4.6.7 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 respone 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=0ff 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, Ilgx_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 ,
```

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.

### 4.4.6.8 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_blocks
Map_block_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_blocks
Map_block_Name, Data_Array_Name, Data_Array_Offset, Function, node_name, ilgx_func      ,da_bit_name, partitior    ea, group, address , length
MapDesc2    ,       DA_01                , 0                          , passive  , Node_A     , Device Data ,  DB_01        , 3                  , 0      , 451255            , 1
```

Once you specify an array for token storage you must set the length to 1. ie. 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 multidigit numbers. They are as unique as a serial number. No two devices have the same number.

### 4.4.6.9 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_blocks
Map_block_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 Capability , DEVCAPOP        , 3          , 1     , 0      , 46101, 1
```

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

For a detailed discussion on bridge configuration, please refer to the instruction manual for the bridge. 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.**

# 6. Driver Notes

## 6.1 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. What happens 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)*

How do you know what map descriptors to construct to capture abandoned data ? The driver provides some help .

Read section 6.3 and implement the driver stats. The 1st four elements of the stats array provided information on the most recent message to be abandoned. The driver never clears these elements and only the most recent event is shown

| 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 6.1.1 Abandoned Message Indication**

| | |
|---|---|
| 1 | Panel Type |
| 2 | Auto Event Lost |
| 3 | Clr Auto Dyn Image |
| 4 | Zone Status |
| 5 | Arming Level |
| 6 | Entry Delay |
| 7 | Alarm Trouble |
| 8 | Siren Setup |
| 9 | Siren Synch |
| 10 | Siren Go |
| 11 | Touchpad Display |
| 12 | Siren Stop |
| 13 | Feature State |
| 14 | Temperature |
| 15 | Time Date |
| 16 | Lights State Cmd |

| | |
|---|---|
| 17 | User Lights Cmd |
| 18 | Keyfob Cmd |
| 19 | Rqst Equip List |
| 20 | Send Single Equip List |
| 21 | Send Zone Data |
| 22 | Send Partition Data |
| 23 | Send Superbus Data |
| 24 | Send Superbus Capabil |
| 25 | Send Output Data |
| 26 | Send User Data |
| 27 | Send Schedule Data |
| 28 | Send Scheduled Event |
| 29 | Send Light To Sensor |
| 31 | Keypress |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

**Table 6.1.2 Abandoned Message Types**

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/trooubl message was ignored. Therefor a alarm / trouble map descriptor needs to be constructed. See the example in section 4.4.6.1 of this manual.

The partition number is zero (element 1), the area number is 1 (element 2) and the deivice 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 6.8 on device address to understand why a device couyld have an address of 11018649

## 6.2 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 release 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.

## 6.3 Driver Stats

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

Bare in mind that 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_blocks
Map_block_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 bridge.

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.

| | |
|---|---|
| 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 |

## 6.4 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 described how these variables are stored.

6.4.1 Table 6.4 Data Variables & Data Storage

| | **Data Variable**<br><br>**Identifies variables that are contained in the data message from the Panel.** | **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<br><br>Storage location is always referenced to the Data_Array_Offset specified in the map descriptor. |

| | | | |
|---|---|---|---|
| **Arming Level** | | | |
| Make one MapDesc per partition.<br><br>Length determines the number of areas covered.<br><br>The area and length are used to determine the relative storage position.<br><br>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 |
| **Automation Event Lost** | | | |
| One MapDesc per device. | | | |
| | Counter of number of message<br>Like this received | 0 | |
| **Panel Type** | | | |
| Make one MapDesc per node. | | | |
| | 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 | |
| **Zone Status** | | | |
| Set the address equal to the start zone number and the length equal to the number of zones.<br><br>If the area number or partition is different then a new MapDesc is required. | | | |
| | **Zone Status** | Based on zone number relative to address. | |
| **Zone Data** | | | |
| Set the address equal to the start zone number and the length equal to the number of zones.<br><br>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. |
| | **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. |
| **Alarm Trouble** | | | |

| | | | |
|---|---|---|---|
| Set the address equal to the start src number and the length equal to the number of devices.<br><br>If the area number or partition is different then a new MapDesc is required.<br><br>The data type of the MapDesc is important too.<br><br>Use Zones, Devices to differentiate storage for zone / devices / system / Phone<br><br>Default is device is no data type is specified.<br><br>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. | | | |
| | Alarm Type = 100 * General_type + Specific_type<br><br>Eg if GT=1 and ST=2<br>Then value stored is 102. | Based on zone number relative to address. | |
| | Event Specific Data | | Same relative location as alarm type. |
| **Delay Entry** | | | |
| 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. | | | |
| | Delay Time | Location is based on area number relative to MapDesc area and offset | |
| | Delay Flags | | Same relative location as delay time. |
| **Siren Setup** | | | |
| 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 | |
| **Siren Synch** | | | |
| One MapDesc per node. | | | |
| | Count of how many times this message | Location based on MapDesc offset | |

| | | | |
|---|---|---|---|
| | has been received. | | |
| **Partition Data** | | | |
| 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.) |
| **Superbus Device Data** | | | |
| One MapDesc per partition per area.<br><br>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. |
| **Superbus Device Capability** | | | |
| One MapDesc per range of devices per node.<br><br>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. |
| **Output Data** | | | |
| One MapDesc per range of devices per node.<br><br>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. |
| **User Data** | | | |
| 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 | |

| | | | |
|---|---|---|---|
| | | consecutive locations as there are tokens (depends on panel config.) | |
| Schedule Data | | | |
| One MapDesc per partition per area.<br><br>Use address to define the schedule that will be processed using this MapDesc.<br><br>One schedule number per MapDesc. | | | |
| | Start Hour | 0 | |
| | Start Minute | 1 | |
| | Stop Hour | 2 | |
| | Stop Min | 3 | |
| | Days of weeks | 4 | |
| | Schedule Validity | 5 | |
| **Feature State** | | | |
| One MapDesc per partition per area. | | | |
| | Feature Bytes | MapDesc offset and as many consecutive locations as there are features.<br><br>Zero ends list. | |
| **Temperature** | | | |
| One MapDesc per partition per area. | | | |
| | Temp in °F | 0 | |
| | Set point | | 0 |
| **Date & Time** | | | |
| One MapDesc per node/ | | | |
| | Hour | 0 | |
| | Minute | 1 | |
| | Month | 2 | |
| | Day | 3 | |
| | Year | 4 | |
| **Touchpad Display** | | | |
| One MapDesc per partition per area | | | |
| | Display Tokens | Based on MapDesc offset and then as many consecutive locations as there are tokens | |
| **Lights State** | | | |
| One MapDesc per partition per area | | | |
| | | | |
| | Light state bits | 0 (Based on MapDesc offset) | |
| **User Lights** | | | |
| One MapDesc per partition per area per range of source numbers.<br><br>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 MapDesc address & offset |
| **Keyfob** | | | |
| One MapDesc per partition & area.<br><br>Range of zone to be processed is defined using address and length. | | | |
| | Key Code | Based on zone number relative to MapDesc address & offset | |
| **Clear Dynamic Data** | | | |
| Can have many of these per node.<br><br>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. | |

## 6.5 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 | |
| Alarm Cancel (General Type = 2) | | |
| 2.xx | xx = same as General Type = 1 | |
| Alarm Restoral (General Type = 3) | | |
| 3.xx | xx = same as General Type = 1. | |
| Fire Trouble (General Type = 4) | | |
| 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 | |
| Fire Trouble Restoral (General Type = 5) | | |
| 5.yy | yy = same as General Type = 4. | |
| Non-Fire Trouble (General Type = 6) | | |
| 6.yy | yy = same as General Type = 4. | |
| Non-Fire Trouble Restoral (General Type = 7) | | |
| 7.yy | yy = same as General Type = 4 | |
| Bypass (General Type = 8) | | |
| 8.0 | Direct Bypass | ES = user number |
| 8.1 | Indirect Bypass | ES = user number |
| 8.2 | Swinger Bypass | |

| | | |
|---|---|---|
| 8.3 | Inhibit | ES = user number |
| Unbypass (General Type = 9) | | |
| 9.0 | Direct Bypass | ES = user number |
| 9.1 | Indirect Bypass | ES = user number |
| 9.2 | Swinger Bypass | |
| 9.3 | Inhibit | ES = user number |
| Opening (General Type = 10) | | |
| 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 |
| Closing (General Type = 11) | | |
| 11.zz | zz = same as General Type 10 | |
| Partition Configuration Change (General Type = 12) | | |
| 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 | |
| Partition Event (General Type = 13) | | |
| 13.0 | Schedule On | ES = schedule number |
| 13.1 | Schedule Off | ES = schedule number |
| 13.2 | Latchkey On | |
| 13.3 | Latchkey Off | |
| 13.4 | Smoke Detectors Reset | |
| 13.5 | Valid User Access Code Entered | ES = user number |
| 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 | ES = user number |
| 13.13 | Auto Force Arm | |
| 13.14 | Auto Force Arm Failed | |
| 13.15 | Arming Protest Begun | ES = arming level attempted |
| 13.16 | Arming Protest Ended | ES = current arming level |
| Partition Test (General Type = 14) | | |
| 14.0 | Manual Phone Test | ES = user number |
| 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 | ES = user number |
| 14.6 | User Sensor Test Ended | ES = user number |
| 14.7 | User Sensor Test Completed | ES = user number |
| 14.8 | User Sensor Test Incomplete | ES = user number |
| 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 | ES = user number |
| System Trouble (General Type = 15) | | |
| 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 | |
| 15.40 | Microphone Switch Trouble | |
| 15.41 | Microphone Trouble | |
| 15.42 | Microphone Wiring Trouble | |
| 15.43 | JTECH Premise Paging Trouble | |
| 15.44 | Voice Siren Tamper Trouble | |
| 15.45 | Microburst Transmit Failure | |
| 15.46 | Microburst Transmit Disabled | |
| 15.47 | Microburst Module Failure | |
| 15.48 | Microburst Not In Service | |
| 15.49 | Automation Supervisory Trouble | |
| 15.50 | Microburst Module Initializing | |
| 15.51 | Printer Paper Out Trouble | |
| System Trouble Restoral (General Type = 16) | | |
| 16.xx | xx = same as General Type = 15 | |
| System Configuration Change (General Type = 17) | | |
| 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 | |
| System Event (General Type = 18) | | |
| 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 |

## 6.6 Text Tokens

| Token | Token Value |
|---|---|
| ------- | -------------- |
| 0 | 00 |
| 1 | 01 |
| 2 | 02 |
| 3 | 03 |
| 4 | 04 |
| 5 | 05 |
| 6 | 06 |
| 7 | 07 |
| 8 | 08 |
| 9 | 09 |
| # | 0C |
| : | 0D |
| / | 0E |
| ? | 0F |
| . | 10 |
| A | 11 |
| B | 12 |
| C | 13 |
| D | 14 |
| E | 15 |
| F | 16 |
| G | 17 |
| H | 18 |
| I | 19 |
| J | 1A |
| K | 1B |
| L | 1C |
| M | 1D |
| N | 1E |
| O | 1F |
| P | 20 |
| Q | 21 |
| R | 22 |
| S | 23 |
| T | 24 |
| U | 25 |
| V | 26 |
| W | 27 |
| X | 28 |
| Y | 29 |
| Z | 2A |
| Space | 2B |
| ' | 2C |
| - | 2D |
| _ | 2E |
| * | 2F |
| AC POWER | 30 |
| ACCESS | 31 |
| ACCOUNT | 32 |

| | |
|---|---|
| ALARM | 33 |
| ALL | 34 |
| ARM | 35 |
| ARMING | 36 |
| AREA | 37 |
| ATTIC | 38 |
| AUTO | 39 |
| AUXILIARY | 3A |
| AWAY | 3B |
| BACK | 3C |
| BATTERY | 3D |
| BEDROOM | 3E |
| BEEPS | 3F |
| BOTTOM | 40 |
| BREEZEWAY | 41 |
| BASEMENT | 42 |
| BATHROOM | 43 |
| BUS | 44 |
| BYPASS | 45 |
| BYPASSED | 46 |
| CABINET | 47 |
| CANCELED | 48 |
| CARPET | 49 |
| CHIME | 4A |
| CLOSET | 4B |
| CLOSING | 4C |
| CODE | 4D |
| CONTROL | 4E |
| CPU | 4F |
| DEGREES | 50 |
| DEN | 51 |
| DESK | 52 |
| DELAY | 53 |
| DELETE | 54 |
| DINING | 55 |
| DIRECT | 56 |
| DOOR | 57 |
| DOWN | 58 |
| DOWNLOAD | 59 |
| DOWNSTAIRS | 5A |
| DRAWER | 5B |
| DISPLAY | 5C |
| DURESS | 5D |
| EAST | 5E |
| ENERGY SAVER | 5F |
| ENTER | 60 |
| ENTRY | 61 |
| ERROR | 62 |
| EXIT | 63 |
| FAIL | 64 |
| FAILURE | 65 |
| FAMILY | 66 |
| FEATURES | 67 |
| FIRE | 68 |
| FIRST | 69 |
| FLOOR | 6A |

| | |
|---|---|
| FORCE | 6B |
| FORMAT | 6C |
| FREEZE | 6D |
| FRONT | 6E |
| FURNACE | 6F |
| GARAGE | 70 |
| GALLERY | 71 |
| GOODBYE | 72 |
| GROUP | 73 |
| HALL | 74 |
| HEAT | 75 |
| HELLO | 76 |
| HELP | 77 |
| HIGH | 78 |
| HOURLY | 79 |
| HOUSE | 7A |
| IMMEDIATE | 7B |
| IN SERVICE | 7C |
| INTERIOR | 7D |
| INTRUSION | 7E |
| INVALID | 7F |
| IS | 80 |
| KEY | 81 |
| KITCHEN | 82 |
| LAUNDRY | 83 |
| LEARN | 84 |
| LEFT | 85 |
| LIBRARY | 86 |
| LEVEL | 87 |
| LIGHT | 88 |
| LIGHTS | 89 |
| LIVING | 8A |
| LOW | 8B |
| MAIN | 8C |
| MASTER | 8D |
| MEDICAL | 8E |
| MEMORY | 8F |
| MIN | 90 |
| MODE | 91 |
| MOTION | 92 |
| NIGHT | 93 |
| NORTH | 94 |
| NOT | 95 |
| NUMBER | 96 |
| OFF | 97 |
| OFFICE | 98 |
| OK | 99 |
| ON | 9A |
| OPEN | 9B |
| OPENING | 9C |
| PANIC | 9D |
| PARTITION | 9E |
| PATIO | 9F |
| PHONE | A0 |
| POLICE | A1 |
| POOL | A2 |

| | |
|---|---|
| PORCH | A3 |
| PRESS | A4 |
| QUIET | A5 |
| QUICK | A6 |
| RECEIVER | A7 |
| REAR | A8 |
| REPORT | A9 |
| REMOTE | AA |
| RESTORE | AB |
| RIGHT | AC |
| ROOM | AD |
| SCHEDULE | AE |
| SCRIPT | AF |
| SEC | B0 |
| SECOND | B1 |
| SET | B2 |
| SENSOR | B3 |
| SHOCK | B4 |
| SIDE | B5 |
| SIREN | B6 |
| SLIDING | B7 |
| SMOKE | B8 |
| Sn | B9 |
| SOUND | BA |
| SOUTH | BB |
| SPECIAL | BC |
| STAIRS | BD |
| START | BE |
| STATUS | BF |
| STAY | C0 |
| STOP | C1 |
| SUPERVISORY | C2 |
| SYSTEM | C3 |
| TAMPER | C4 |
| TEMPERATURE | C5 |
| TEMPORARY | C6 |
| TEST | C7 |
| TIME | C8 |
| TIMEOUT | C9 |
| TOUCHPAD | CA |
| TRIP | CB |
| TROUBLE | CC |
| UNBYPASS | CD |
| UNIT | CE |
| UP | CF |
| VERIFY | D0 |
| VIOLATION | D1 |
| WARNING | D2 |
| WEST | D3 |
| WINDOW | D4 |
| MENU | D5 |
| RETURN | D6 |
| POUND | D7 |
| HOME | D8 |
| CARRIAGE RETURN | F9 |
| PSEUDO SPACE | FA |

| | |
|---|---|
| CARRIAGE RETURN | FB |
| BACKSPACE | FD |
| BLINK NEXT TOKEN | FE |

## 6.7 Driver Error Messages

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 6.3 for information on how to expose driver statistics.

*ILGX:#3 Err. Response requires too much buffer. optr=%u iptr=%u
Monitor the ILGX_STAT_BUFFER_OFLOW statistic

*ILGX:#4 Err. Incoming data is being abandoned.
Read section 6.1 for more information.

ILGX:#5 Err. Light to Sensor data abandoned. No storage method
*ILGX:#6 Err. Scheduled Event data abandoned. No storage method
*ILGX:#7 Err. Schedule data abandoned. No storage method
These functions have not been implemented by the driver. Contact Fieldserver to discuss your requirements.

*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.
*ILGX:#10 Err. Scheduled Event data. Null response sent.
*ILGX:#11 Err. Schedule data. Null response sent.
These functions have not been implemented by the driver. Contact Fieldserver to discuss your requirements.

*ILGX:#12 Err. Command=%x(h) cant be parsed
*ILGX:#13 FYI. Command=%x(h) SubCmd=%x(h) cant be parsed
*ILGX:#14 FYI. Command=%x(h) SubCmd=%x(h) cant be parsed.
The driver has received a command from the panel which it does cannot parse. This could be a new function / message produced by the panel that is not supported by the driver or it could also 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 6.3). 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:#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 an 'Concord' Panel.
ILGX:#17 FYI. Slave is an 'Concord Express' Panel.
ILGX:#18 FYI. Slave is NOT an 'Advent' Panel.
ILGX:#19 FYI. Slave is an 'Advent' Panel.

These messages are produced as appropriate for each tier of the bridge 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:#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 and then reset the bridge.

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 bridge. 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 ore 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 bridge. See section 4.4.2 for a lost 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 bridge. See section 4.4.2 for a lost of valid keywords.

*ILGX:26 Err. No Store! Array too short. Mapdesc=<%s> xxxx
…
*ILGX:58 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 too causing the error. The solution is to increase the length of the map descriptor in the CSV file and reset the bridge. The specific number of the error message indicates the type of map descriptor causing the problem. As the message provide the offending map descriptor's name it is not necessary to tabulate the error message numbers separately. Ensure you have used unique map descriptor names.

- Indicated error messages that are produced once only and then suppressed even if the error occurs again. The driver implements some messages like this so that they do not full the error buffer as they may be produced very poll.

## 6.8 Device Addresses

Interlogix manufacture SuperBus Devices with unique addresses. The address is fixed at the time of manufacture. Each device, even the same type 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 address are in the range 0 – 16,777,215.

## 6.9 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 | |

## 1. Revision History

| Date | Driver Version | Document Revision | Resp | Comment |
|---|---|---|---|---|
|  | 1.00 | 1 |  |  |
| 12/30/03 | 1.00 | 2 | JD | Releasing |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |