Chipkin
Automation Systems

Driver Manual
(Supplement to the FieldServer Instruction Manual)

FS-8705-16
Mircom FX2000 Fire Alarm Panel

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after May 1, 2001

© **Chipkin Automation Systems**, 3495 Cambie St- Box 211, Vancouver, BC, Canada, V5Z 4R3
Tel: (866) 383-1657, **Fax:** (416) 915-4024
Email: dfs@chipkin.com **Website:** www.chipkin.com

TABLE OF CONTENTS

1. Mircom FX2000 Fire Alarm Panel Serial Driver Description

The FX2000 driver is a passive client driver intended for connection to the serial printer port of a FX2000 Panel. A passive client driver waits for messages to be sent to it (by the panel). The driver cannot send messages to the panel and hence it cannot request the state of any point in the panel.

The driver can process alarm and trouble events and the system reset message. All other messages and events are ignored.

The driver can only be used as a client. Minimal server functionality is provided only to support our ongoing quality assurance program by facilitating automated testing of the driver. It is not documented or supported.

Synchronization

A consequence of the fact that this is a passive client driver, is that the FieldServer must be synchronized with the panel by clearing all abnormal states, resetting the panel and then restarting the FieldServer/Sending the FieldServer a command to reset the data.

Driver Functionality

When an alarm occurs the panel reports the alarm with a message which identifies the loop and address. The driver will store a 1 in a Data Array location mapped for that point. When the condition causing alarm is cleared, the panel does not send a message reporting this. Thus the driver will not know the condition has been cleared until a panel reset is performed. When a reset is performed the driver will clear all the alarm states, previsouly stored, to zero. For those points where the alarm condition has not been cleared, the panel sends a new alarm notification which the driver will recognize and store a 1 in a Data Array location mapped for that point.

Thus for the two scenario's

Alarm condition cleared before reset

Alarm Event - Driver Store a 1
Alarm condition cleared - Driver is not informed (no message from panel)
Reset Initiated - Driver clears the 1 to zero (for all alarms)

Alarm condition not cleared before reset

Alarm Event - Driver Stores a 1
Reset Initiated - Driver clears the 1 to zero (for all alarms)
Panel now sends re-notifications of active alarms
Re-notification of Alarm Event - Driver Stores a 1

For troubles, the panel sends a message when the trouble event occurs and it also sends a message when the event is cleared even if a panel reset has not been performed. The driver will store a 1 when the trouble event occurs and a zero when it is cleared. This will not be affected by a reset.

Max Nodes Supported

FieldServer Mode	Nodes	Comments
Passive Client	1	Each port on the FieldServer can only be connected to 1 panel since the message do not report the panel number so message from different panels connected on the same port cannot be differentiated.
Active Server (Simulate a FX2000 Panel)	0	Not supported or documented.

2. Driver Scope of Supply

2.1. Supplied by FieldServer Technologies for this driver

FieldServer Technologies PART #	Description
-	No specific cables are shipped with this driver. A generic RJ45 Ethernet cable must be shipped with this driver.
-	A generic male and Female connector kit must be shipped with this driver.
FS-8705-16	Driver Manual.

2.2. Provided by the Supplier of 3rd Party Equipment

2.2.1. Required 3rd Party Hardware

Part #	Description

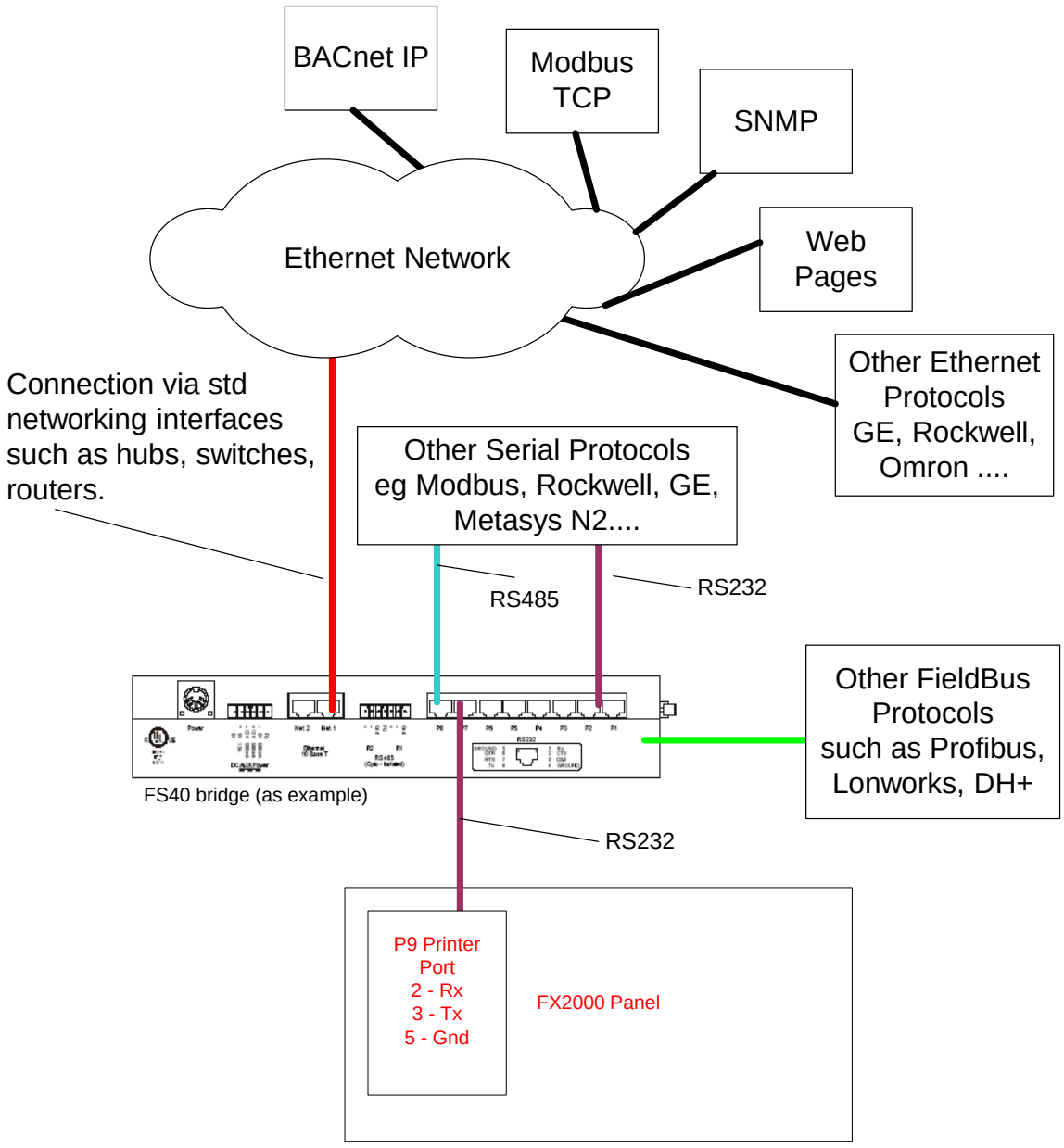
2.2.2. Required 3rd Party Software

2.2.3. Required 3rd Party Configuration

None Known.

3. Hardware Connections

Multiple upstream protocols and connection supported. See list of FieldServer Drivers.



3.1. Hardware Connection Tips / Hints

Blank Section

4. Configuring the FieldServer as a FX2000 Fire Alarm Panel Passive Client

For a detailed discussion on FieldServer configuration, please refer to the FieldServer Configuration Manual. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" sample files provided with the FS).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a FX2000 Panel

4.1. Data Arrays/Descriptors

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for FSC - Electronic Siren Controllers Serial Driver communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

Section Title
Data_Arrays

Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	Float, Bit, UInt16, Uint32, SInt16, Packed_Bit, Byte, Packed_Byte, Swapped_Byte
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10,000

Example

```
//    Data Arrays
Data_Arrays
Data_Array_Name,        Data_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 Descriptions

Create one connection for each FX2000 port. Each connection can only be used to connect to a single FX2000 interface/port.

Section Title		
Connections		
Column Title	**Function**	**Legal Values**
Port	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[1]
Protocol	Specify protocol used	Mircom_FX2000
Baud*	Specify baud rate	Driver Supports : 110; 300; 600; 1200; 2400; 4800; **9600**; 19200; 28800; 38400; 57600 Baud *FX2000* supports: 9600
Parity*	Specify parity	Driver Supports : 7,**8** *FX2000* supports: 8
Data_Bits*	Specify data bits	Driver Supports : **1**,2 *FX2000* supports: 1
Stop_Bits*	Specify stop bits	Driver Supports : Odd, Even, **None** *FX2000* supports: none
Handshaking*	Specify hardware handshaking	**None**
Poll _Delay*	This parameter is not used.	
FX2000_Max_Loops	How many Fire Alarm Panel Loops must be processed. Save execution time by setting this to the maximum loop number	Default=1

[1] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

FX2000_Store_Style	Controls how the driver stores alarm and trouble data.	0 (Default) 1 (Compact)
FX2000_Points_Per_Loop	Controls how the driver stores alarm and trouble data when compact storage is selected	Positive intger.

Example

```
//      Client Side Connections

Connections
Port,       Protocol,          Baud,   Parity,   FX2000_Max_Loops,
P1,         Mircom_FX2000,     9600,   None,     2,
```

4.3. Client Side Node Descriptors

Create one Node per connection only.

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for node	Up to 32 alphanumeric characters
Node_ID	Station address of physical server node This parameter is not used directly by the driver. We recommend that a unique Node ID's be given to each node.	1-258
Protocol	Specify protocol used	FX2000
Connection	Specify which port the device is connected to the FieldServer	P1-P8, R1-R2[2]

Example

```
//    Client Side Nodes

Nodes
Node_Name,          Node_ID,    Protocol,          Connection
SirenNode,          1,          Mircom_FX2000,     P1
```

[2] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

4.4. Client Side Map Descriptors

4.4.1. FieldServer Related Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from "Data Array" section above
Data_Array_Offset	Starting location in Data Array	0 to maximum specified in "Data Array" section above
Function	Function of Client Map Descriptor	RDBC, WRBC, WRBX

4.4.2. Driver Related Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in "Client Node Descriptor" above
Data_Type	Data type This commonly used parameter is not used by this driver.	
Length	Length of Map Descriptor This commonly used parameter is not used by this driver. However, it must be specified with a value of 1.	1
Address	This commonly used FieldServer parameter is not used by this protocol.	

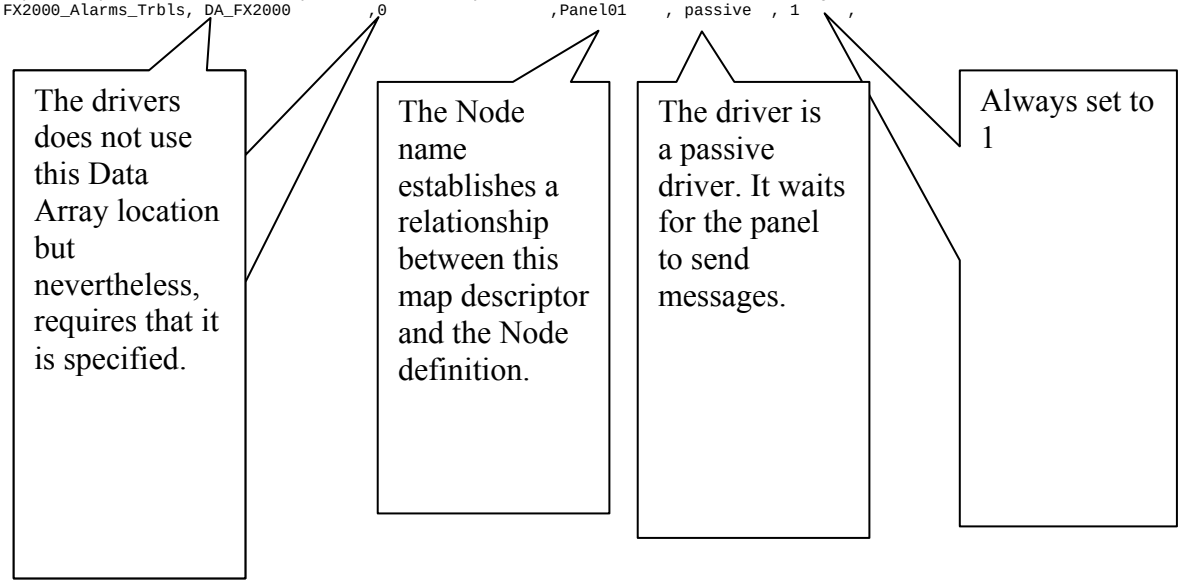
4.4.3. Timing Parameters

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

4.4.4. Map Descriptor Example 1 .

Only one Map Descriptor is required to process all messages from the panel. The driver stores data in specially named Data Arrays.

```
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset ,Node_Name , Function , Length,
FX2000_Alarms_Trbls, DA_FX2000        ,0                 ,Panel01    , passive  , 1     ,
```



4.5. How Point Data is Stored.

There are two styles for Storing Data – Default and Compact

In the default mode each address on each loop is allocated one data element (an integer value) to indicate the status. You check the value to see if the point is normal, alarm, trouble or if alarm and trouble are both active.

In the compact mode the driver allocated two data elements (each element is 1 bit). One of the elements is indicate and alarm, the other trouble so if you wish to know if the device is not in a normal state you have to inspect two items of info

4.5.1. Default Storage Method

For the purposes of this discussion we will call a loop-address combination a point (which represents a field device such as a pull station, a smoke detector etc.).

The driver stores data representing panel events in specially named data arrays. They are named as follows:

DA_FX2000LOOPxx

Where xx = Loop Number
Example: DA_FX2000LOOP01, DA_FX2000LOOP02, DA_FX2000LOOP03 etc.
It is your responsibility to define data arrays for each loop that can report events.

When the driver receives a message reporting an event for a loop-address point it updated the data array named after the loop. The position in the data array that is updated is based on the address.

Example: Event for Loop 2 Address 90

Driver updates DA_FX2000LOOP02[90] ie. Offset 90 of the Data Array called DA_FX2000LOOP02

Each of these location is treated as byte with 8 bits. Which bits get updated indicates the kind of event.

Bit Number	Meaning
0	Point in alarm state
1	Point in trouble state
2	Point in alarm / trouble state (

	summary)

Example: Alarm Event for Loop 2 Address 90
Driver updates DA_FX2000LOOP02[90] = 5
Indicates alarm state is active (bit 0)
Indicates alarm/trouble active (bit 2)

Example: Trouble Event for Loop 2 Address 90
Driver updates DA_FX2000LOOP02[90] = 6
Indicates trouble state is active (bit 1)
Indicates alarm/trouble active (bit 2)

Example: Restored Trouble Event for Loop 2 Address 90
Driver updates DA_FX2000LOOP02[90] = 0
Indicates trouble state is normal (bit 1)
Indicates alarm/trouble not present (bit 2)

4.5.2. Compact Storage Method

With compact storage the driver uses the connection parameter "FX2000_Points_Per_Loop".

For the purposes of this discussion we will call a loop-address combination a point (which represents a field device such as a pull station, a smoke detector etc.).

The driver stores data representing panel events in specially named data arrays. They are named as follows:

DA_FX2000LOOPxx

Where xx = Loop Number
Example: DA_FX2000LOOP01, DA_FX2000LOOP02, etc.

It is your responsibility to define data arrays for each loop that can report events.

When the driver receives a message reporting an event for a loop-address point it updated the data array named after the loop. The position in the

data array that is updated is based on the address AND a consideration of whether the event is alarm / trouble related.

If the event is alarm related the offset into the Data Array is based on the Address

Example: Alarm for Loop 2 Address 90

Driver updates DA_FX2000LOOP02[90] ie. Offset 90 of the Data Array called DA_FX2000LOOP02

If the event is trouble related event the offset into the Data Array is based on the Address and the "FX2000_Points_Per_Loop".

Example: Trouble for Loop 2 Address 90
"FX2000_Points_Per_Loop". = 100 (Eg)

Driver updates DA_FX2000LOOP02[90+100]

ie. Offset 190 of the Data Array called DA_FX2000LOOP02

4.6. How Most Recent Event Data is Stored.

Each time an event is processed the driver stores data about the most recent event is a specially named data array. All you need to in the configuration is define the Data Array with named "DA_FX2000"

Example:

```
Data_Arrays
Data_Array_Name ,Data_Format ,Data_Array_Length
DA_FX2000       ,UINT16      ,20
```

The driver uses offsets 1-10 to store summary data.

Offset	Contents
1	Loop Number Driver stores a zero if the event is a panel reset.
2	Address Driver stores a zero if the event is a panel reset.
3	Event Type 1=Alarm 2=Trouble 3=Trouble Restored 4=Reset event
4	Year eg. 2009
5	Month eg 1 for Jan
6	Day of month
7	Hour
8	Minute
9	Reserved
10	Reserved

The driver looks for this array and stores the following table of data:

4.7. Driver Limitations and Event Processing

The driver can only process the following events

Supported Messages
Alarms Must contain a loop number and address 2nd line of message if present with a description is ignored. Driver recognizes “ALARM ACTIVATED” (case insensitive) or Driver recognizes “PRIORITY ALARM” (case insensitive)
Troubles Must contain a loop number and address 2nd line of message if present with a description is ignored. Driver recognizes “OPEN CIRCUIT TRB” (case insensitive) or Driver recognizes “TRB” (case insensitive) or Driver recognizes “TROUBLE” (case insensitive) or
Trouble Restores Must contain a loop number and address 2nd line of message if present with a description is ignored. Driver recognizes “TROUBLE RESTORED” (case insensitive)
Completed Panel Resets. Driver recognizes “SYSTEM RESET COMPLETE” (case insensitive)

You can change the list of keywords that the driver recognizes. Preapre and download a file called fx2000.ini (case sensitive file name). After the download the driver will load and use the table of keywords.

This is a sample for fx2000.ini. If you were to prepare and use this sample then there would be no change to the driver's default behavior since this list of keywords is the same as the one the driver recognizes by default.

Example 1:

```
TROUBLE RESTORED, 1 , 1 ,
OPEN CIRCUIT TRB, 2 , 1 ,
TRB, 2 , 1 ,
TROUBLE, 2 , 1 ,
MISSING DEVICE, 2 , 1 ,
ALARM ACTIVATED, 3 , 1 ,
PRIORITY ALARM, 3 , 1 ,
SYSTEM RESET COMPLETE, 4 , 0 ,
```

Note the format.

1. Caps for keywords
2. No spaces to left or right of the keywords. Ie no space before comma
3. Column 2 represents the event type.
    a. 1=Restore
    b. 2=Trouble
    c. 3=Alarm
    d. 4=Reset
4. Column 3 tells the driver if a loop/address should be expected. 1=yes. 0=No.

For example. Say you wanted the driver to recognize a new alarm event which will be recognized by the text 'Initial Alarm'. Then 1) Prepare the file with all the lines from example 1 and then add the line from example 2.

Example 2:

```
INITIAL ALARM, 3 , 1 ,
```

The driver processes these lines in order from top to bottom. That is why the 'Trouble Restored' line is placed before the line with 'Trouble'. If it were the other way around the driver would match the word trouble and thus recognize the 'Trouble Restored' message incorrectly as a trouble.

5. Configuring the FieldServer as a FX2000 – Security Intercom System Server

This driver cannot be used to emulate a FX2000 panel.

Appendix 1. Advanced Topics

This section is blank.

Appendix 1.1. Driver Error Messages

Error Message We have shown place holders for the parts of the message which change. %s is a place holder for a text string. %d is a place holder for a number %c is a place holder for an alpha character.	**Explanation and corrective action** ***FYI messages are informational and do not require a corrective action. Simply use them to confirm configuration / behaviors are what you expect.***
FX2000:#01 FYI. Using these keywords	Reports which keywords the driver is using to determine the event type. Read the notes in section 4.7
FX2000:#02 FYI. Table Loaded from fx2000.ini	Reports which keywords the driver is using to determine the event type. Read the notes in section 4.7. This message is printed when the driver defaults are overridden by using the file fx2000.ini You will need to delete this file on the FieldServer to return to the defaults.
FX2000:#03 FYI Max Loops=%d	Reports the maximum number of panel loops that will be processed. Use the connection parameter 'FX2000_MAX_LOOPS' to control this.
FX2000:#04a Err. Cant find DA=%s for store	This message is printed when the driver attempt to store data for a point on a loop but the driver cannot fnd an appropriately named data array. Read section 4.5 to see how Data Arrays should be named. Then correct the configuration, download the corrected configuration and restart the FieldServer for changes to take effect.

FX2000:#05 Err. expected LOOP&ADDRESS (%s)	The driver expects to find a loop and address specified on certain messages sent by the panel. This message is printed when it cant find one/both. You cannot correct this problem without help. Please capture a log (using the FieldSerever Utilities) and then send the log to CAS for analysis.
FX2000:#10 Err. Cant find DA=%s for store FX2000:#11 Err. DA=%s too short. Min Len=10	The driver stores summary data in a Data Array called "DA_FX2000". These messages are printed (and then suppressed) if the Data Array has not been defined or is too small. Read section 4.5 to learn more.

Appendix 1.1. Modbus/TCP Template File

```
//===============================================================================
//
// 1.00aA  10 Mar 10 PMC  Created
//
//===============================================================================*/


//===============================================================================
//
// Notes :  None.
//
//
//===============================================================================


//===============================================================================
//
//    Common Information
//

Bridge
Title
Mircom FX200 ModbusTCP Template

//===============================================================================
//
//    Data Arrays
//

Data_Arrays
Data_Array_Name ,Data_Format ,Data_Array_Length
DA_FX2000       ,Uint16      ,200
DA_FX2000LOOP01 ,UINT16      ,400
DA_FX2000LOOP02 ,UINT16      ,400
```

These arrays have names that have a specific meaning to the driver. There must be one per loop else data for that loop will not be stored.

If you are using the 'Compact' Storage method then the length must twice the max address of the loop.

```
//=========================================================================
//
//    Passive Client Side Connections
//

Connections
Port , Baud , Data_Bits , Stop_Bits , Parity , Protocol      ,FX2000_Max_Loops ,FX2000_Store_Style ,FX2000_Points_Per_Loop
P1   , 9600 , 8         , 1         , None   , Mircom_FX2000 ,2                ,1(Compact)          ,100
```

Data from other loops will not be stored. Set this number accurately. The smaller it is the less processing time is required to process a reset message.

Compact means each point has two storage locations. One to report an alarm state and one to report a trouble state.

When calculating the offset into the data array where point state are calculated, the driver uses this value to calculate the offset to the location where point zero's trouble will be stored.

```
//=========================================================================
//
//    Passive Client Side Nodes
//

Nodes
Node_Name , Node_ID , Protocol
Panel01   , 1       , Mircom_FX2000
```

Node number is not used but must be specified. Use a unique number.

```
//=========================================================================
```

```
//
//    Passive Client Side Map Descriptors
//

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Node_Name , Function ,Length
FX2000_Alarms_Trbls , DA_FX2000       , 0                 , Panel01   , Server   ,1
```

This is the only Map Descriptor required to process messages from the Panel. The 'Recent Event' data is stored in the data array specified here. All other data is stored in the specially named Data Arrays.

```
//=============================================================================
//
//    Server Side Connections
//
Connections
Adapter ,Protocol    ,
N1      ,Modbus/TCP  ,

//=============================================================================
//
//    Server Side Nodes
//
Nodes
Node_Name ,Node_Id ,Protocol
MOD_01    ,2       ,Modbus/TCP
```

```
//================================================================================
//
//    Server Side Map Descriptors
//
Map_Descriptors
Map_Descriptor_Name ,Data_Array_Name ,Data_Array_Offset ,Node_Name ,Function,Address  ,Length
ServLoop1Alms       ,DA_FX2000LOOP01 ,0                 ,MOD_01    ,Server  ,40001    ,200
ServLoop2Alms       ,DA_FX2000LOOP02 ,0                 ,MOD_01    ,Server  ,41001    ,200
ServLoop1Trbls      ,DA_FX2000LOOP01 ,100               ,MOD_01    ,Server  ,42001    ,200
ServLoop2Trbls      ,DA_FX2000LOOP02 ,100               ,MOD_01    ,Server  ,43001    ,200
ServGeneral         ,DA_FX2000       ,1                 ,MOD_01    ,Server  ,44001    ,20
```

The allocation of these addresses is arbitrary. You can use any non-overlapping addresses and you can use 1xxxx addresses too.

The alarms and troubles are stored in the same data array but at different locations. The driver uses the value of "FX2000_Points_Per_Loop" to work out where the trouble must be stored. Alarms are stored at offset 0,1,2…. Troubles are stored at offsets FX2000_Points_Per_Loop+0, FX2000_Points_Per_Loop+1 , FX2000_Points_Per_Loop+2 ….

Blank Page

Revision History

Date	Resp	Format	Driver Ver.	Doc. Rev.	Comment
21 Jul 2009	PMC		1.00a	0	Document Created
10 Mar 2010	PMC		1.01a	1	Added notes about compact storage method.
11 Mar 2010	PMC		1.01a	2	ModbusTCP Template and notes added.