CHIPKIN

# Mircom Pro2000 Serial Driver FS-8705-17

Chipkin - Enabling Integration

salesgroup1@chipkin.com

Tel: +1 866 383 1657



Driver Version: 1.00a
Document Revision: 1

# TABLE OF CONTENTS

# 1 Mircom PRO2000 Fire Alarm Panel Serial Driver Description

The driver provides an interface to monitor a Mircom Pro2000 Fire Alarm Panel. It is able to read status data from the panel – when combined with another protocol driver this data can be served using another protocol such as BACnet, Lonworks, Johnson Controls N2, Rockwell, XML etc. A block diagram showing potential connectivity is provided below.

To avoid manual configuration the driver is provided with a utility that is used to aid in the creation of the FieldServer configuration. This utility reads the PRO2000 Modbus CSV and uses this to make BACnet objects, for each device connected to the system. One set of objects for each device is created – reporting the alarm, trouble and other status info of the device. The Modbus CSV file is created when the panel is configured and is provided by the engineer who configures the fire alarm panel.

This is an active client driver – it sends commands and polls and the panel responds with data.

The driver supports a small subset of the command/query functions and assumes the devices have been configured correctly and appropriately to achieve the control required.

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**

No special steps are required for synchronization. On startup the driver will looked for new flagged events, thereafter it will read the status of every device thus providing an automatic synch.

**Driver Functionality**

The driver continuously polls the panel reading a set of change flags. These flags are set by the panel when a device's status changes. Actually each flag may represent more than one device. The Pro200 refers to these groups of devices as blocks. When a change flag is seen, the driver reads the set of data associated with the device that the change flag represents. This can range from 1 to 100 devices. The quantity (block size) is determined by the panel configuration engineer. When data associated with a change flag is read, the panel unsets the flag.

If more than one change flag is set this is how the driver operates. It reads, the data associated with the 1st one set. In the next cycle, when it reads the change flags again, it will look for a change flag for a higher block number and then read that block's data. It only returns to look at the start of the change flags again when all the change flags have been processed.

If no change flags are set, the driver will read the data for each device by reading the blocks of data associated with each flag. In this mode, with no change flags set, the driver sequences through each block in turn and starts at the beginning again.

Each device has 16 bits allocated to it. These bits report the status of the device. They are documented in this manual but you should check with the manufacturer's documentation in case there have been changes. This data can be mapped onto any other FieldServer protocol such as BACnet or Modbus/TCP.

## Configuration Tool

CAS has developed a configuration tool to assist in the automation of the production of the configuration file. To obtain a copy of the tool send an email to support@chipkin.com and ask for the 'Mircom Pro2000 configuration tool'.

Use of this tool eliminates the need to create the 16 potential objects for each device, eliminates the need to retype the device labels etc. .. ie use it for productivity and accuracy.

The tool uses the data in the Modbus file to produce a portion of a FieldServer configuration. This is then added to the base configuration file and the result is a complete configuration to map the panel data onto BACnet/IP objects.

## Max Nodes Supported

| FIELDSERVER MODE | NODES | COMMENTS |
|---|---|---|
| Active Client | 1 | Only one panel can be connected to a single FieldServer . If this **LIMITATION** has significant impact for your project, contact us, we might be able to change this limitation. |
| Active Server<br>(Simulate a PRO2000 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.<br>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-8705-17 | 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

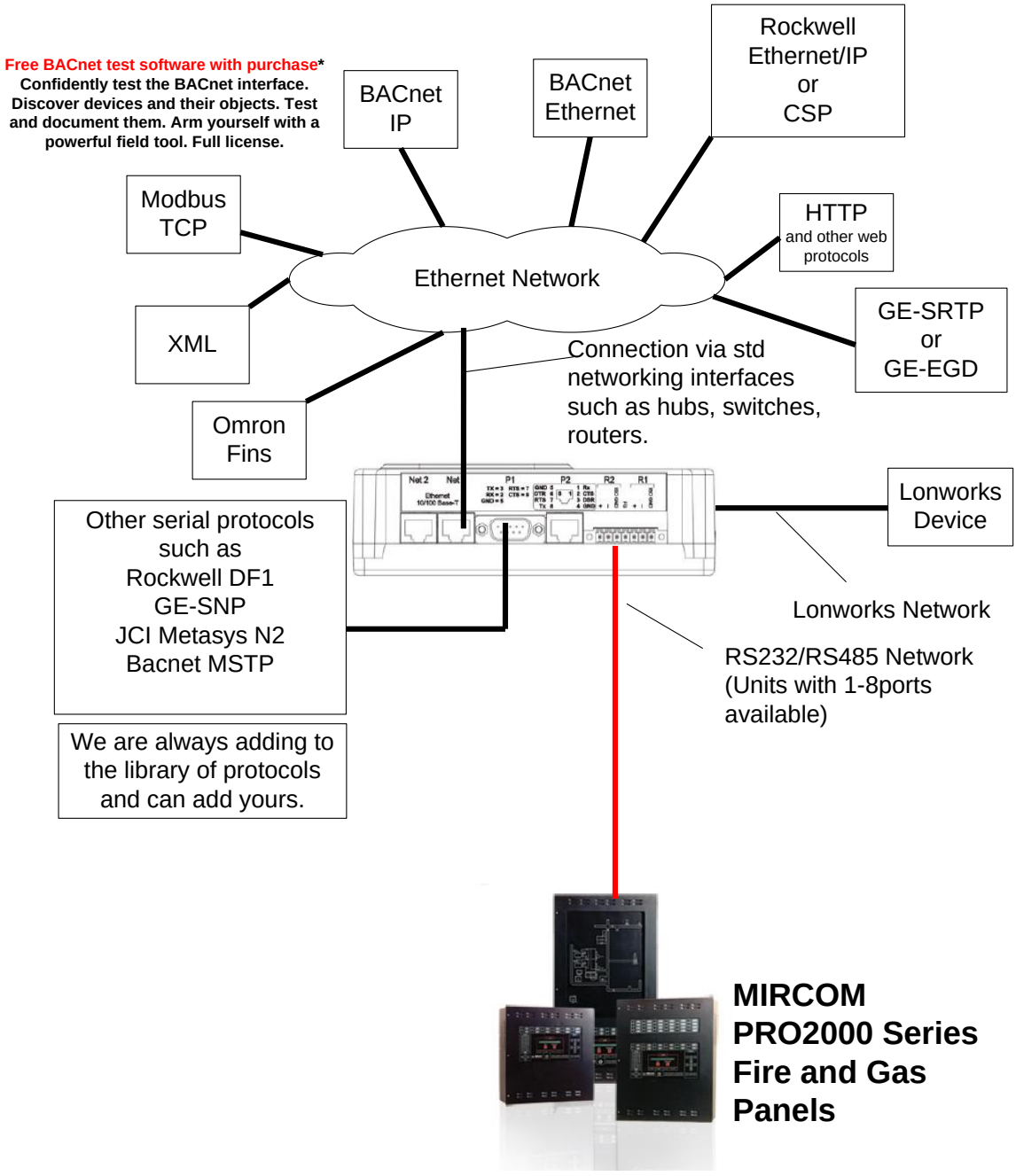
### 2.2.3 Required 3rd Party Configuration

None known.

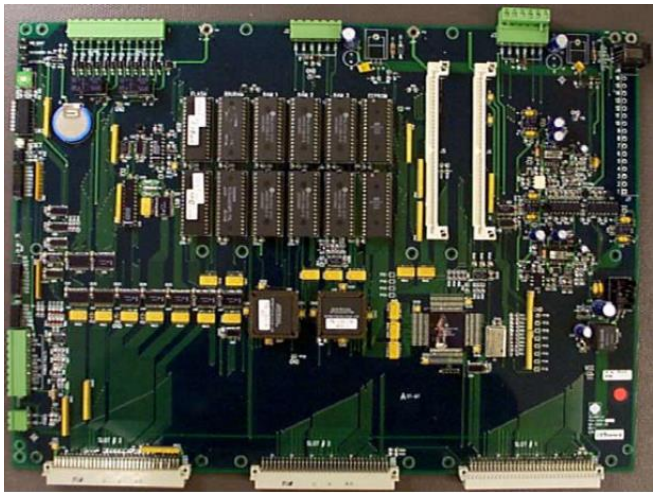
# 3 Hardware Connections

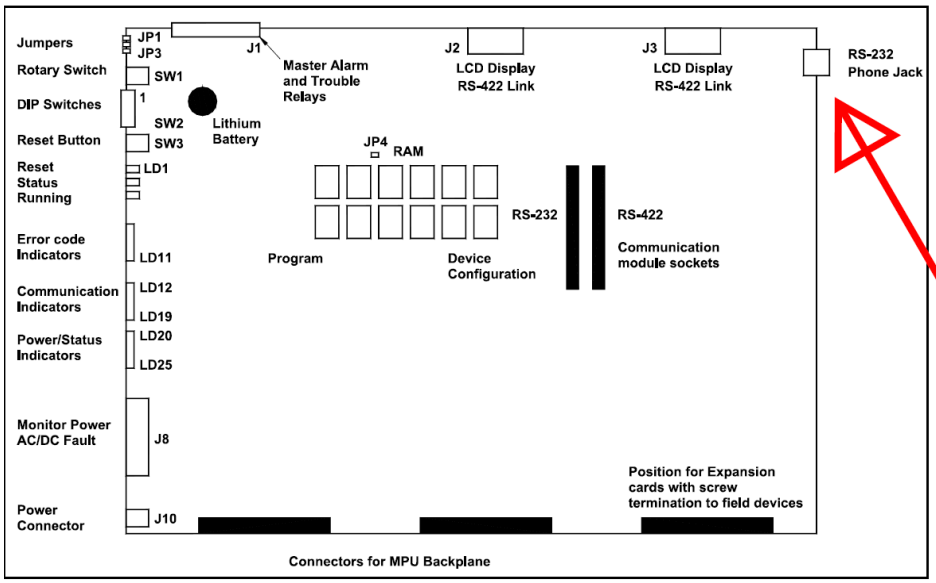
Multiple upstream protocols and connection supported. See list of FieldServer Drivers.

**Monitor and Control Mircom PRO2000 FIre Panels using BACnet, Lonworks or a PLC system**



## 3.1 Hardware Connection Tips / Hints






Jack J4 Pinout

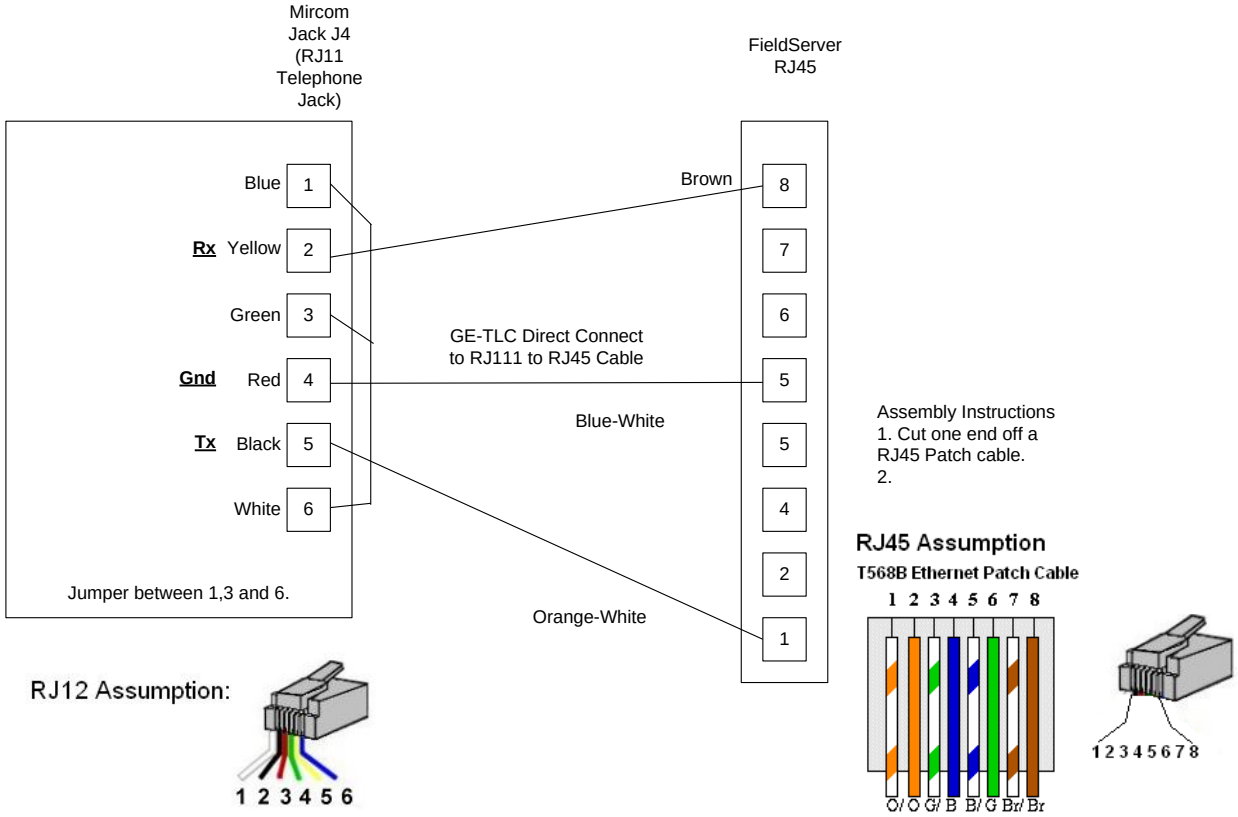
| Signal | Pin |
|---|---|
| TX (out) | 5 |
| RX (in) | 2 |
| RTS* (out) | 3 |
| CTS* (out) | 6 |
| CD* (in) | 1 |
| GND | 4 |

MPU Switch Positions

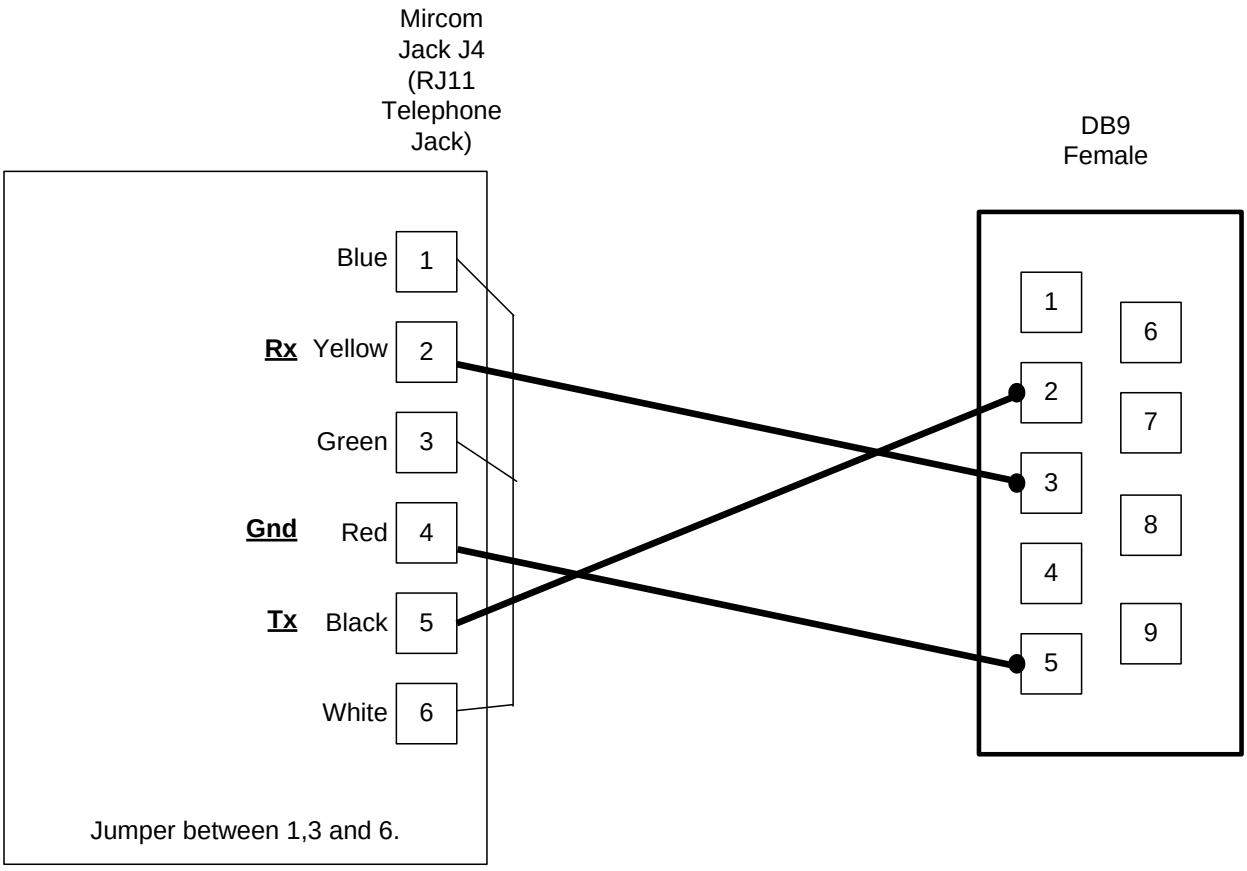
| | |
|---|---|
| **SW1** | 0: Printer, 1: Configuration, 2: Normal/Diagnose; 3: ModBus, 4: LCD, 5: Network, 6: Redundant, E: Firmware upgrade, F: Default configuration |
| **SW2** | Normal position: All OFF |
| **SW3** | Used to reset the MPU card |

Led's for RS232

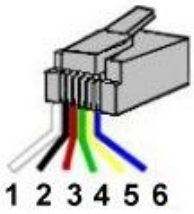
LD16 (RX3) and LD17 (TX3) for RS-232




Cable Assembly – This cable is not provided.




RJ12 Assumption:




Cable Assembly – This cable is not provided.

# 4 Configuring the FieldServer as a PRO2000 Fire Alarm Panel Active 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 PRO2000 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,
PRO2k_DATA,             Packed_Bit,     16000
PRO2k_FLAGS,            Packed_Bit,     1600
pro2000-stats,          UNT16,          200
```

REQUIRED:

These 3 Data arrays are required in all Pro2000 configurations.

## 4.2 Client Side Connection Descriptions

Create one connection for each PRO2000 port. Each connection can only be used to connect to a single PRO2000 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_PRO2000 |
| Baud* | Specify baud rate | Driver Supports : 110; 300; 600; 1200; 2400; 4800; **9600**; 19200; 28800; 38400; 57600 Baud<br>*PRO2000* supports: 2400,4800,9600,91200 |
| Parity* | Specify parity | Driver Supports : 7,**8**<br>*PRO2000* supports: 8 |
| Data_Bits* | Specify data bits | Driver Supports : **1**,2<br>*PRO2000* supports: 1 |
| Stop_Bits* | Specify stop bits | Driver Supports : Odd, Even, **None**<br>*PRO2000* supports: none |
| Handshaking* | Specify hardware handshaking | **None** |
| Poll _Delay* | This parameter is not used. | |

**Example**

```
// Client Side Connections

Connections
Port,    Baud,    Parity,    Protocol,
P1,      9600,    None,      Mircom_PRO2000,
```

[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.

### 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<br><br>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 | PRO2000 |
| Connection | Specify which port the device is connected to the FieldServer | P1-P8, R1-R2[2] |
| Pro2000_Max_Blocks | When there are no change flags set the driver sequences through each block of device information. However, the driver doesn't know how many block there are being used so it could read more information than necessary. Use this parameter to make sure the driver does not read data blocks that do not correspond to installed devices. | **16 , Any whole number Max=32** |

[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.

**Example**

```
//  Client Side Nodes


Nodes
Node_Name,          Node_ID,    Protocol,          Connection
Pro2kPanel,         1,          Mircom_PRO2000,    P1
```

## 4.4 Client Side Map Descriptors

Think of a Map Descriptor as a task. Normally each task read or writes different items of data. For this driver is there is only one map descriptor required – its precense is mandatory.

### 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<br><br>This commonly used parameter is not used by this driver. | |
| Length | Length of Map Descriptor<br><br>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

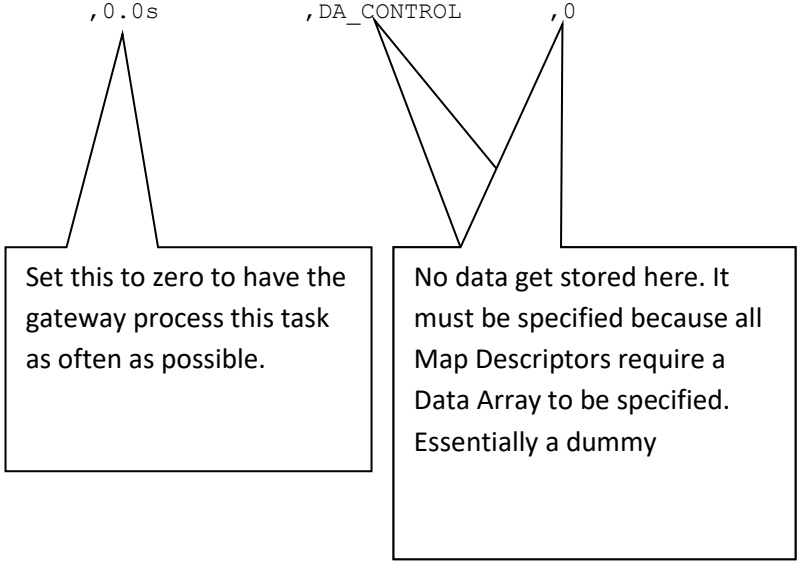
| COLUMN TITLE | FUNCTION | LEGAL VALUES |
|---|---|---|
| Scan_Interval | Rate at which data is polled | ≥0.001s |

### 4.4.4 Map Descriptor Example 1

Only one Map Descriptor is required to process all messages from the panel. The MD is mandatory. This task is used to drive the automatic reading of change flags and device status data.

**Read section 4.5** **How Point Data is Stored**

```
Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Node_Name ,Function ,Address ,Length
Read_Inputs         ,0.0s          ,DA_CONTROL      ,0                  ,Panel01   ,Rdbc     ,1       ,1
```



## 4.5 How Point Data is Stored

Data is always stored in two Data Arrays. The names of the Data Arrays is fixed. Please follow our recommendation of data type.

```
Data_Arrays
Data_Array_Name  ,Data_Format ,Data_Array_Length
PRO2k_DATA       ,Packed_Bit  ,16000
PRO2k_FLAGS      ,Packed_Bit  ,1600
```

Each time change flags are read, they are stored in the Data Array named PRO2k_FLAGS.

Each time a block of device status data is read, it is stored in the Data Array named PRO2k_DATA. One word (16 bits) are used for each device.

Device data starts at offset 21 in the Data Array.

There is no obvious relationship between the position and the device number. Ie. You cannot tell which device stores its status data at offset 21, 22, 23….. The allocation of storage positions is done by the Pro2000 panel configuration software and documented in the Pro2000 Modbus File. Only when you have that file in your possession can you allocate meaning to the data.

## 4.6 How Most Recent Event Data is Stored

**Input Device**

| BIT # | STATE=0 | STATE=1 |
|---|---|---|
| 0 | No Trouble | Trouble |
| 1 | No Alarm | Alarm |
| 2 | Service Mode OFF | Service Mode ON |
| 3 | Isolate Mode OFF | Isolate Mode ON |
| 4 | NA | NA |
| 5 | Not Used | Not Used |
| 6 | Not Used | Not Used |
| 7 | Reserved bit 3 | Reserved bit 3 |
| 8 | Reserved bit 2 | Reserved bit 2 |
| 9 | Reserved bit 1 | Reserved bit 1 |
| 10 | UA bit 1 | UA bit 1 |
| 11 | Device not present in database | Device present in database |

| | | |
|---|---|---|
| 12 | Alarm not cleared | Alarm cleared |
| 13 | Trouble not cleared | Trouble cleared |
| 14 | Alarm not acknowledged | Alarm acknowledged |
| 15 | Trouble not acknowledged | Trouble acknowledged |

**Output Device**

| BIT # | STATE=0 | STATE=1 |
|---|---|---|
| 0 | No Trouble | Trouble |
| 1 | Normal | Off Normal(Supervisory) |
| 2 | Service Mode OFF | Service Mode ON |
| 3 | Isolate Mode OFF | Isolate Mode ON |
| 4 | Automatic | Manual |
| 5 | Not Used | Not Used |
| 6 | Not Used | Not Used |
| 7 | Reserved bit 3 | Reserved bit 3 |
| 8 | Reserved bit 2 | Reserved bit 2 |
| 9 | Reserved bit 1 | Reserved bit 1 |
| 10 | UA bit 1 | UA bit 1 |
| 11 | Device not present in database | Device present in database |
| 12 | NA | NA |
| 13 | Trouble not cleared | Trouble cleared |
| 14 | NA | NA |
| 15 | Trouble not acknowledged | Trouble acknowledged |

## 4.7 Driver Limitations and Event Processing

To obtain a copy of the tool send an email to support@chipkin.com and ask for the 'Mircom Pro2000 configuration tool' (Download from this link: https://www.filesanywhere.com/fs/v.aspx?v=8a696588616173b771a0 )

The tool is used to form the server portion of a FieldServer configuration file. The client side portion of the configuration file is usually standard. A copy and notes are provided in another section of this manual.

The PRO-2000 Configurator will generate a Text file to cross-reference a device ID to a MODBUS register. Each device will have a MODBUS register in the MODBUS database. The MODBUS register will be assigned automatically and sequentially by the PRO-2000 Configurator

At the time of writing the Modbus text file contains the following columns

Column 1 (1st Column) Modbus Holding Register Number.

Column 4 Device ID

Column 5 LCD Message

The tool does not use the contents of any other columns. The tool looks for a line that begins :MODBUS. It processes all the lines starting on the next line, until it finds a line that begins :Local. It then ends processing. The 'finds' are case sensitive.

(You could construct any CSV file that has this format. This would give you an opportunity to use different labels than allocated for the DeviceID and LCD Message)

The tool takes the following parameters

| PARAM | |
|---|---|
| 1 | InputFileName<br><br>The name of the Modbus Text file produced by the pro2000 configurator. . You can specify the full path. |
| 2 | OutputFileName<br><br>The output from this tool is saved in a file. Specify the name of the file here. You can specify the full path. |
| 3 | NameOption<br><br>Valid options are 1,2,3 |

The tool must make a name for each BACnet data object. The name is visible to the remote BACnet system and is usually descriptive.

You can form the BACnet object name using

1. The contents of column5 of the Modbus file only (LCD Message).
2. The contents of column4 of the Modbus file only (DeviceID ).
3. The contents of column4+5 of the Modbus file (DeviceID+LC Label).

Remember that only 30 characters of data can be used because the FieldServer still needs to add a suffix when using the name.

4

objectsOption

There are 16 status bits for each device in the Pro2000. Some of these are used and some are not. Each of these status bits will be represented by a single BACnet Binary object. Thus each fire alarm panel device could have up to 16 BACnet binary objects.

Using this parameter you can control whether this tool make 16 or less objects.

**Output Devices**

| | | If there is a 1 In the column then the corresponding BACnet Object is created when Option= | | |
|---|---|---|---|---|
| **Suffix** | **Description** | **Option=1** | **Option=2** | **Option=3** |
| "Sup" | "Supervisory" | 1 | 1 | 1 |
| "Alm" | "Alarm" | 1 | 1 | 1 |
| "Servce" | "Service Mode" | 1 | 1 | 1 |
| "Iso" | "Isolate Mode" | 1 | 1 | 1 |
| "Auto" | "Auto Manual" | 1 | 1 | 1 |

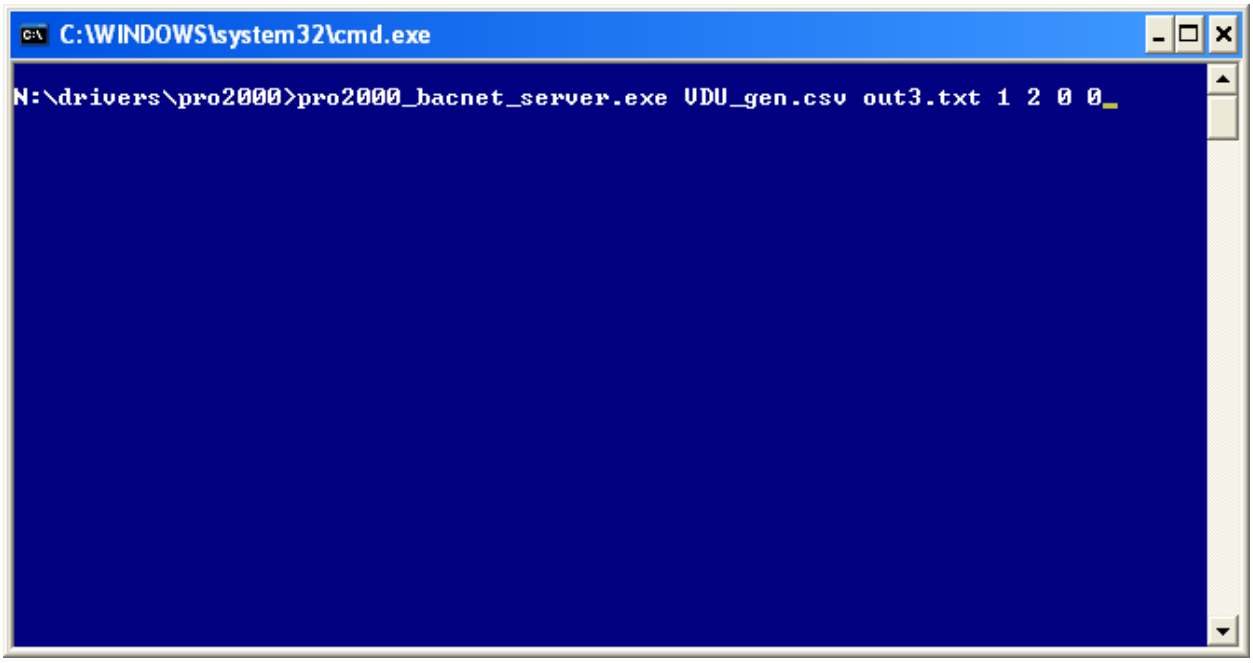
| | | | | | |
|---|---|---|---|---|---|
| | "NU1" | "Not Used" | 1 | 0 | 0 |
| | "NU2" | "Not Used" | 1 | 0 | 0 |
| | "Rsvd3" | "Reserved Bit 3" | 1 | 0 | 0 |
| | "Rsvd2" | "Reserved Bit 2" | 1 | 0 | 0 |
| | "Rsvd1" | "Reserved Bit 1" | 1 | 0 | 0 |
| | "UA1" | "UA Bit 1" | 1 | 1 | 0 |
| | "CfgErr" | "Device not in Dbase" | 1 | 1 | 0 |
| | "N-A" | "Not Applicable" | 1 | 0 | 0 |
| | "Trb-NC" | "Trouble not Cleared" | 1 | 1 | 0 |
| | "N-A" | "Not Applicable" | 1 | 0 | 0 |
| | "TrbAck" | "Trouble Acked" | 1 | 1 | 0 |

| | |
|---|---|
| 5 | columnOffset<br><br>In case the format of the table has changed, some flexibility is provided. Set this parameter to a non-zero value if the DeviceID and LCD Message are no longer columns 4 and 5. If the new positions are columns 5 and 6 then set this parameter to 1. Ie use it to offset the original position. |
| 6 | commentOption<br><br>If set to 1 then an extra line is written in the csv file for each BACnet object created. This extra line is a comment. It contains the 'Description' from the table for parameter 4 as well as the original contents of the DeviceID and LCD Message columns. |
| 7 | headerOption<br><br>If set to 1 then the output file is a complete configuration and will include the default client side. If set to zero the output file contains the BACnet server configuration only. |

Example:

If you were provided the MODBUS text file from the configuration engineer and it is name VDU_gen.csv and you copied it into your temp folder on your C drive.

C:\temp>pro2000_bacnet_server.exe VDU_gen.csv out3.txt 1 2 0 0 0




When this command is executed the tool opens VDI_gen.csv (param1) and starts to read it. At the same time it create a file called out3.txt (if the file previously existed then the contents will be overwritten.) For each device it makes a set of objects. The object names use column5 the LCD Message (controlled by param3=1). It makes 9 objects per output device (controlled by param6=2). The LCD Name is found in column 5 as expected and no offset is required (controlled by param7=0). The tool does not write a comment in the output file for each object created. (Controlled by param8=0).

### 4.7.1 How to use the tool

Obtain the tool

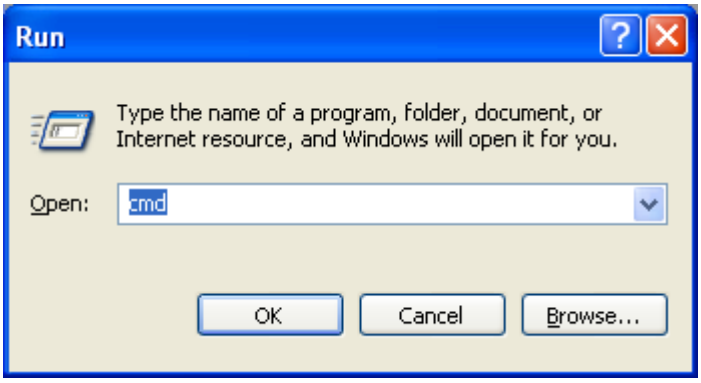
Create a folder called c:\temp\pro2000

Copy the tool application file into that folder. If it is zipped. Unzip into that folder.

Start a command session

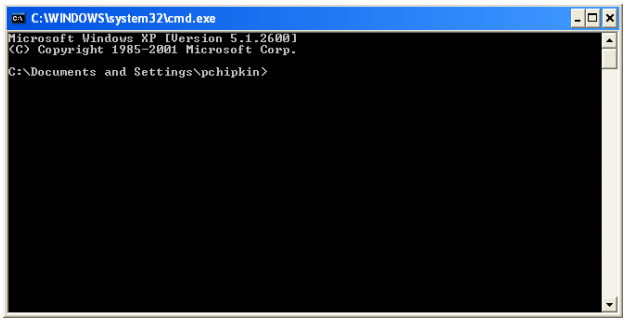
Start Button

Run Button



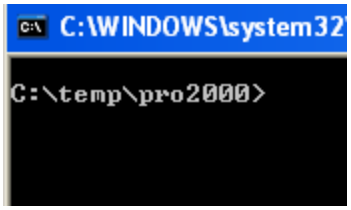

Type cmd and push the OK button

You get a window similar to this one




Change folder by typing

Cd\temp\pro2000<Enter>




Copy the input file (the Modbus text file) into the same folder. For the purposes of this example assume it is called Modbus.txt

Type the following

C:\temp\pro2000>pro2000_bacnet_server.exe Modbus.txt bacnet.csv 1 2 0 0 0<Enter>

Combine the base config file and the BACnet portion

C:\temp\pro2000>Copy client.csv+bacnet.csv config.csv<Enter>

Copy the file config.csv to the download location using explorer

C:\FieldServer Technologies\FieldServer Utilities\Config

Download the configuration and restart the FieldServer to give effect to the new file.

### 4.8 Driver Limitations

The following functions have not been implemented but could easily be added on request.

1. ClearStack The PRO-2000 cancels all previous received commands
2. GetVar Get the content of a PRO-2000 Variable
3. SetVar Set the contents of a PRO-2000 Variable (except date)
4. SetDate Set the Pro-2000 date variable
5. GetAnalog Get the analog values of a detector

# 5 Configuring the FieldServer as a PRO2000 Server

This driver cannot be used to emulate a PRO2000 panel.

# 6 Revision History

| DATE | RESP | FORMAT | DRIVER VER. | DOC. REV. | COMMENT |
|---|---|---|---|---|---|
| 20 Sep 2010 | PMC | | 1.00a | 0 | Document Created |
| 10 May 2021 | YC | | 1.00a | 1 | Updated to latest template |
| | | | | | |
| | | | | | |

# Appendix A. **Advanced Topics**

This section is blank.

## Appendix A.1 **Driver Error Messages**

| ERROR MESSAGE | EXPLANATION AND CORRECTIVE ACTION |
|---|---|
| We have shown place holders for the parts of the message which change.<br><br>%s is a place holder for a text string.<br>%d is a place holder for a number<br>%c is a place holder for an alpha character. | ***FYI messages are informational and do not require a corrective action. Simply use them to confirm configuration / behaviors are what you expect.*** |
| PRO2000:#01 Err Cant Write to device | This driver cannot be configured to write to the panel.<br><br>You will need to review the manual, understand the capabilities, edit the configuration and download the modified configuration to the FieldServer. Once restarted the changed take effect. |
| PRO2000:#02 FYI Registers per block=%d | You may safely ignore this message. When the driver reads this information from the panel it prints this message. |
| PRO2000:#03 FYI Max blocks=max(%d,%d) | You may safely ignore this message. The driver is reporting how many blocks of data it will process. The 1st value if the default. The 2nd is the value loaded from the config. |
| PRO2000:#04 Err. DA=%s not found | This is a fatal error arising from a configuration error. The Data array required to store the flags is not present.<br><br>Read section **Error! Reference source not found. Error! Reference source not found.** |

| | |
|---|---|
| | You will need to review the manual, understand the capabilities, edit the configuration and download the modified configuration to the FieldServer. Once restarted the changed take effect. |
| PRO2000:#05 Err. DA=%s not found | This is a fatal error arising from a configuration error. The Data array required to store the device status data is not present.<br><br>Read section **Error! Reference source not found. Error! Reference source not found.**<br><br>You will need to review the manual, understand the capabilities, edit the configuration and download the modified configuration to the FieldServer. Once restarted the changed take effect. |
| PRO2000:#06 FYI. Use an Array called <%s> to expose diagnostic info | You may safely ignore this message.<br><br>Read section **Error! Reference source not found.** - **Error! Reference source not found.** for more information |
| PRO2000:#07 Err. Bad MD length - defaulting to 1 | The driver expects the single map descriptor to have a length of 1. Review your configuration. |
| PRO2000:#08 FYI Flags 1st=%d next=%d | Ignore this message. It is used during testing. |
| PRO2000:#09 FYI Next block=%d | Ignore this message. It is used during testing. |

# Appendix A.2 **Driver Statistics**

## A.2.1. **Exposing Driver Stats**

The diver makes some of its operating statistics available in a Data Array where they can be read by a remote client. The lines from the example below can be cut and pasted into a configuration file.

```
Data_Arrays,
Data_Array_Name,    Data_Format,    Data_Array_length,
Pro2000_stats,      UINT32,         1000,
```

| OFFSET | DESCRIPTION |
|---|---|
| 1 | Driver sets this value each time it reads the panel to learn the number of registers per block |
| 2 | The driver reports the max number of block it can process |
| 3 | The driver reports the max number of blocks it has been configured to process. If not specified in the configuration then the driver uses the default. (See #2) |
| 4 | Driver report the next block that it will process |
| 5 | The driver increments this stat each time it reads the number of registers per block |
| 6 | The driver increments this stat each time it reads the change flags section |
| 7 | The driver increments this stat each time it reads device status data. |
| 8 | The driver increments this stat each time it sends a read message |
| 9 | The driver increments this stat each time is receives a response. |
| 10 | The driver increments this stat each time it complete a task cycle (ie return the read map descriptor to the queue and waits to be retasked.) |
| 11 | The driver increments this stat each time the response received was valid in length but the checksum is wrong. It is abandoned and the polling cycle begins again. |
| 12 | The driver increments this stat each time it receives a response who length allows it be considered for processing |
| 13 | The driver increments this stat each time it receives a response theta was invalid in structure or length. |

## Appendix A.3 **The Client Side of the configuration**

```
//===============================================================================
//
//
//
//
// 1.00  20 Sep 2010 PMC  Created
//
//===============================================================================*/



Bridge
Title                                ,System_Node_Id
Pro2000 Base config - Client Side Rev=100 ,1


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

Data_Arrays
Data_Array_Name  ,Data_Format ,Data_Array_Length
PRO2k_DATA       ,Packed_Bit  ,16000
PRO2k_FLAGS      ,Packed_Bit  ,1600
DA_CONTROL       ,UINT16      ,1
pro2000-stats    ,uint16      ,100



//===============================================================================
//
//    Client Side Connections
//
//

Connections
Port ,Baud  ,Data_Bits ,Stop_Bits ,Parity ,Protocol        ,
P1   ,19200 ,8         ,1         ,None   ,Mircom_pro2000  ,
```

Mandatory names for data arrays. Please use the same data format.

Check the Baud rate corresponds to the panel

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

Nodes
Node_Name ,Node_ID ,Protocol       ,Port ,Pro2000_Max_Blocks
Panel01   ,1       ,Mircom_pro2000 ,P1   ,100
//

//=========================================================
//
//    Client Side Map Descriptors
//

Map_Descriptors
Map_Descriptor_Name ,Scan_Interval ,Data_Array_Name ,Data_Array_Offset ,Node_Name ,Function ,Address ,Length
Read_Inputs         ,0.0s          ,DA_CONTROL      ,0                 ,Panel01   ,Rdbc     ,1       ,1
```

This is an important variable. If absent or set to high, the driver will waste time reading device status info that correspond to devices that have not been installed.

Despite this, when a state changes, the driver will poll for that device data because the panel sets a change flag and change flags get service priority

Only a single MD is required. The driver knows what work to do. This single map descriptor is used to initiate that work.