

**Omni-Comm™ Example**  
**Caterpillar ADEM III**  
**To**  
**Modbus Master**



# Caterpillar to Modbus MASTER Configuration

## TABLE OF CONTENTS

<b>SECTION 1</b> .....	<b>1</b>
<i>Overview</i> .....	<i>1</i>
1.1 WHAT IS THIS: .....	1
1.2 OTHER FILES YOU WILL NEED: .....	1
<b>SECTION 2</b> .....	<b>2</b>
<i>Concepts</i> .....	<i>2</i>
2.1 GENERAL CONCEPTS .....	2
2.2 CCM COMMUNICATION .....	2
2.3 OMNII-COMM COMMUNICATION .....	2
<b>SECTION 3</b> .....	<b>3</b>
<i>Starting the program</i> .....	<i>3</i>
3.1 GETTING STARTED .....	3
3.2 START OMNII-CONFIG .....	3
From DOS .....	3
From Windows 3.1X .....	3
From Windows 95/98/NT .....	3
<b>SECTION 4</b> .....	<b>5</b>
<i>Typical Configuration Procedure</i> .....	<i>5</i>
4.1 SELECT PROTOCOLS .....	5
4.2 LOAD MOD2M5X.DBA .....	6
4.3 CONFIGURE CONNECTORS .....	6
4.4 COMMON INFORMATION .....	10
4.5 POLL TABLES .....	11
4.5.1 <i>Poll Table #1; Define Command Data</i> .....	<i>12</i>
Poll Table #1, Read .....	13
Poll Table #1, Write .....	14
Poll Table #1, Error .....	15
4.5.2 <i>Define Registers for Return Data</i> .....	<i>15</i>
Poll Table #2 Read .....	16
Poll Table #2 Write/EXT .....	16
Poll Table #2 Error .....	17
4.5.3 <i>Single Parameter Read</i> .....	<i>17</i>
Poll Table #5 Read Section .....	17
Poll Table #5 Write Section .....	18
4.5.4 <i>Define Broadcast Lists</i> .....	<i>19</i>
Poll Table #6 Read .....	19
Poll Table #6 Write .....	20
Poll Table #7 Read .....	21
Poll Table #11 Write .....	22
4.6 CATERPILLAR M5X PROTOCOL EXTENSION TABLE .....	23
4.7 MODBUS REGISTER/BIT ASSIGNMENTS .....	24

## Section 1

### Overview

#### *1.1 What is this:*

This document describes how to connect a Miille Applied Research Co. Omnii-Comm™ module between a Caterpillar CCM and another system that communicates using Modbus protocol. In this example, the Omnii-Comm is configured as a Modbus Master device. In this mode, the Omnii-Comm will configure the CCM with information it reads from a Modbus Slave. Data collected from the Caterpillar engine by the CCM will be written to Holding Registers in the Modbus Slave. This Application Note describes new functions and features available on the newest Caterpillar CCM modules. This CCM is available on engines equipped with the ADEM III controller. You can also use the procedures described here to communicate with an older CCM.

#### *1.2 Other files you will need:*

This document describes a specific Omnii-Comm™ configuration. An almost unlimited number of variations can be generated depending on specific user needs. It is recommended that you download the following files to be used as a starting place for your final configuration. These files should be available for download from the MARC web site at <http://www.miille.com>

MOD2M5XM.DBA	Omnii-Config database file
MOD2M5XM.XLS	Excel spreadsheet detailing register mapping for MOD2M5XM.DBA

## Section 2

### Concepts

#### 2.1 General Concepts

The Miille Applied Research Co. Omnii-Comm™ can be configured in a multitude of different ways depending on the requirements of the user. This example specifically describes a configuration of the Omnii-Comm that enables it to be a Modbus Master device that can define the configuration of a Caterpillar CCM based on information read from a Modbus Slave and also report engine data collected by the CCM to the Modbus Slave.

#### 2.2 CCM Communication

The CCM is a standard Caterpillar product that provides an industry standard RS232 connection to the Caterpillar data bus (CAT BUS). The communication protocol used on the RS232 connection is Caterpillar's standard M50 protocol. The protocol on the CAT BUS is proprietary to Caterpillar. The Omnii-Comm connects to the RS232 port on the CCM and talks to it using the M50 protocol. Depending on its configuration, the CCM will periodically send data collected from the engine via the CAT BUS out its serial port to the Omnii-Comm using the M50 protocol.

On the other side of the CCM, connected to the CAT BUS, are a multitude of sensors that measure various engine parameters such as RPM, Coolant Temperature, Oil Pressure, etc. A full list of the Parameter IDs (PIDs) that are available, via the CCM, and can be found in your CCM Users Manual. The Parameter IDs that are available vary depending on the type of engine and the CCM model number.

The important point here is that the Caterpillar CCM is a **configurable** communication device and based on its configuration, will periodically send (broadcast) data out its serial communication port. The data that the CCM sends is organized into "Lists" and each "List" is broadcast at a specific interval. The CONTENTS of each List is user definable as is the broadcast INTERVAL.

**To get any *meaningful* data out of a CCM it must first be configured!**

#### 2.3 Omnii-Comm Communication

The Omnii-Comm is also a **configurable** module. Not only can you select the communication protocol to be used on each of its serial ports, you can easily control the moving data in one port and out another; changing protocols on the way if you wish. Another way to use the Omnii-Comm is to collect data from outside devices, using the standard protocol of that device, and then make it available to another system using the protocol of the requesting system.

The first mode of operation is what is detailed in this help file. The Omnii-Comm is connected to a Caterpillar CCM and communicates with it using Caterpillar M50 protocol. It is configured to also have a Modbus Master port that will communicate with a Modbus Slave such as a PLC. The Modbus master will READ data from the Slave to control the CCM configuration and will WRITE engine data received from the CCM to Holding Registers in the Slave.

**To make this work, the CCM AND the Omnii-Comm must BOTH be configured!**

## Section 3

### Starting the program

#### 3.1 Getting Started

Copy all of the files from the Configuration Diskette included with your new Omnii-Comm module to a new sub-directory on your computer. If the diskette is missing you may download the latest copy from our web site at <http://www.miille.com>. You can also download a set of example configuration files

#### 3.2 Start Omnii-Config

Omnii-Config will run under DOS, Windows 3.1X, Windows 9X and Windows NT. Depending on your environment, start Omnii-Config using one of the following procedures:

##### From DOS

Change Directory to where you saved the Omnii-Config files.  
Start the MARC Omnii-Comm Configuration program by simply typing Config at the DOS prompt.

##### From Windows 3.1X

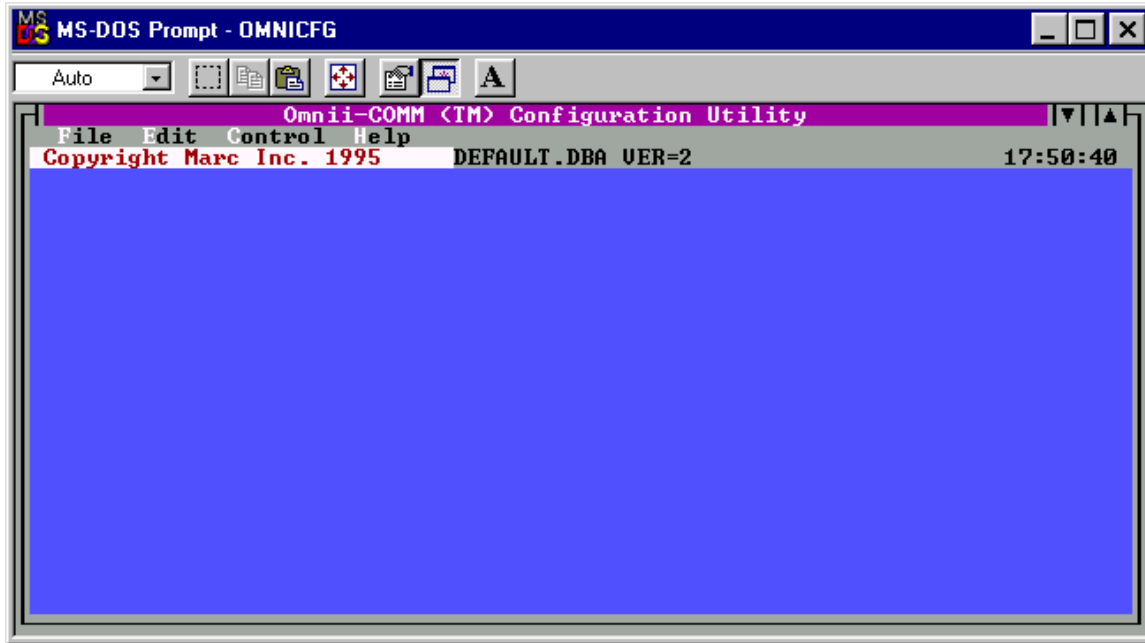
From the Program Manager window, select File, New, Program Item then OK  
Complete the Program Items Properties screen  
Description = Omnii-Config  
Command Line = Config  
Working Directory = name of the subdirectory where you saved the files

##### From Windows 95/98/NT

Right click on the desktop  
Select New then Shortcut  
Browse to the folder where you saved the Omnii-Config files  
Select Omnicfg.exe  
Select a name for the shortcut  
Select an ICON for the shortcut. You may wish to use the CHECKMRK.ICO file supplied on the Omnii Config diskette.  
To start the program, double click on the new shortcut Icon

### 3.3 Main Edit screen

The Omnii-Config program should start and you should see the Main Edit screen.



There are only four basic steps required to build a configuration file and you will start each one of them from the Main Edit screen shown above.

#### The basic steps are:

- Select Protocols to use
- Configure the connectors
- Configure Common items
- Define address mapping between protocols

Once you have built a configuration file, you will save it using the Save or Save As commands under the File menu item. You will use the Control menu item to start another program, Omnii-Talk that is used to download the configuration file to the Omnii-Comm, save it to EEPROM and Start. You can click on the Help menu item at any time to get on-screen help.

Depending on your specific application and the protocols to be used, it is not possible to provide an exact step-by-step procedure that can be used in all cases. The following paragraphs present a typical configuration that can be used as a starting place for your final configuration. If you understand the example configuration, you should have no problems editing it to match your exact requirements.

## Section 4

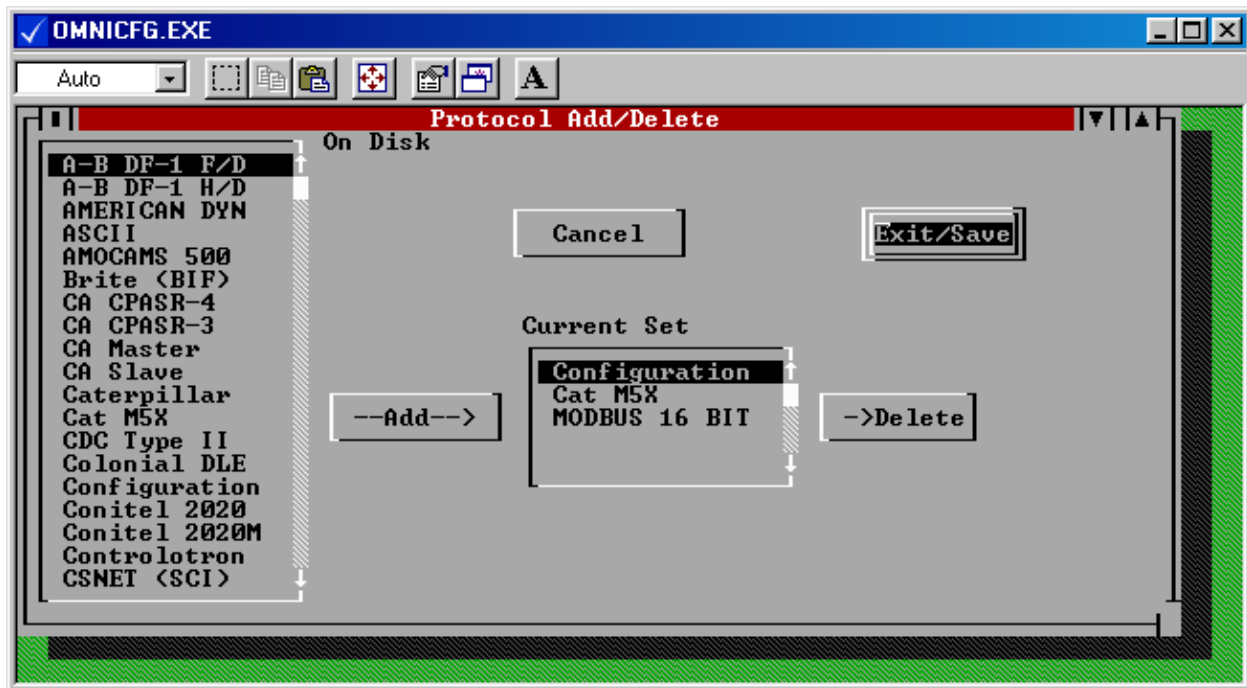
# Typical Configuration Procedure

### 4.1 Select Protocols

Before you attempt to load an existing Configuration file (an Omnii-Config configuration file has a .DBA extension) or to begin a new edit session, you must be sure that you have the correct protocols loaded into your Current Set.

**From the Main Edit screen select Edit then Protocol Maintenance**

You should see a screen like this:



For this example, change your system to make your Current Protocol Set look like the screen above. Delete and add protocols as necessary to get your Current Set protocols the same as shown.

If you need to add a protocol to your current set, highlight it in the On Disk list and click on the Add button. To remove a protocol from the current set, highlight it in the Current Set list and click the Delete button.

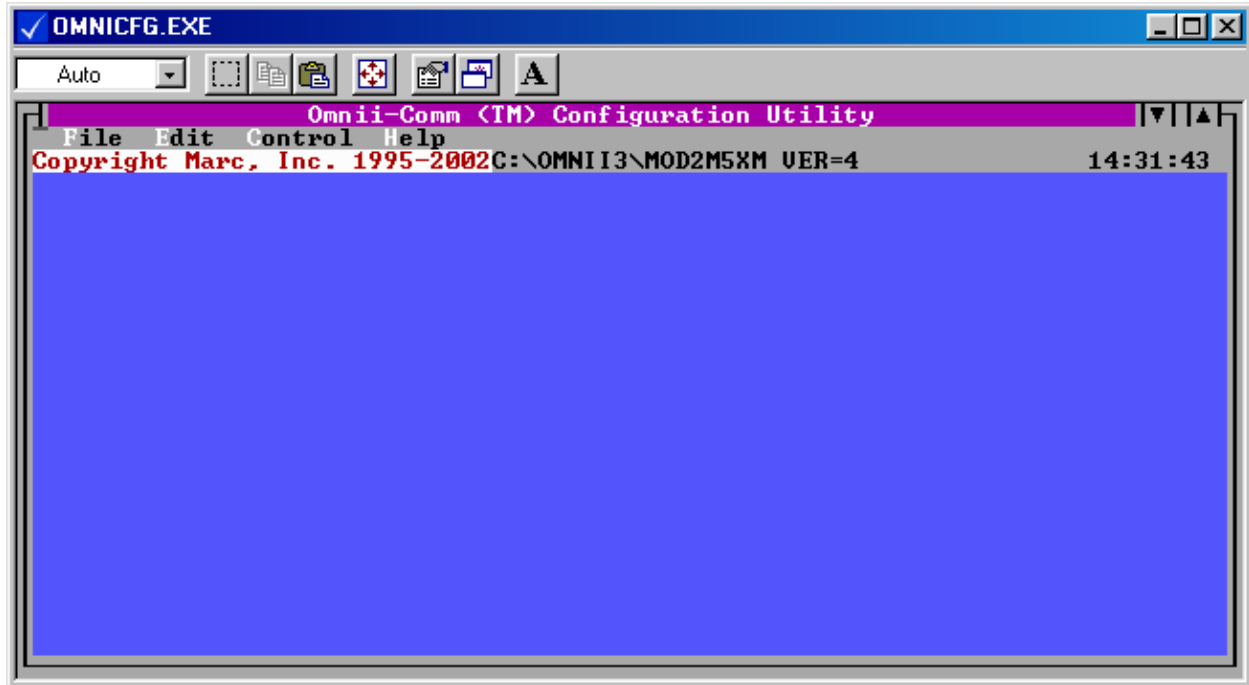
When you have the protocols you need in the Current Set list, click the Exit/Save button to return to the Main Edit screen.

The Current Set as shown above is all that is necessary to load the MOD2M5X.DBA example file.

When the Current Set is correct click on the Exit/Save button to save your changes and return to the Main Edit screen.

## 4.2 Load MOD2M5X.DBA

Load the example file MOD2M5XM.DBA. To do this, first select File, then Open and navigate to the directory that contains the configuration file MOD2M5XM.DBA. Double click on the file name and you should see the Main Edit screen with the current file name displayed on the top line as shown below: The file should load with no errors.



## 4.3 Configure Connectors

You can define the protocol to use on each Omnicomm connector in almost any way you want to! Be aware that there are some minor restrictions and caveats that we will address later in this section but for now, just assume that you can assign any protocol to any connector.

### From the Main Edit screen select Edit then Connector

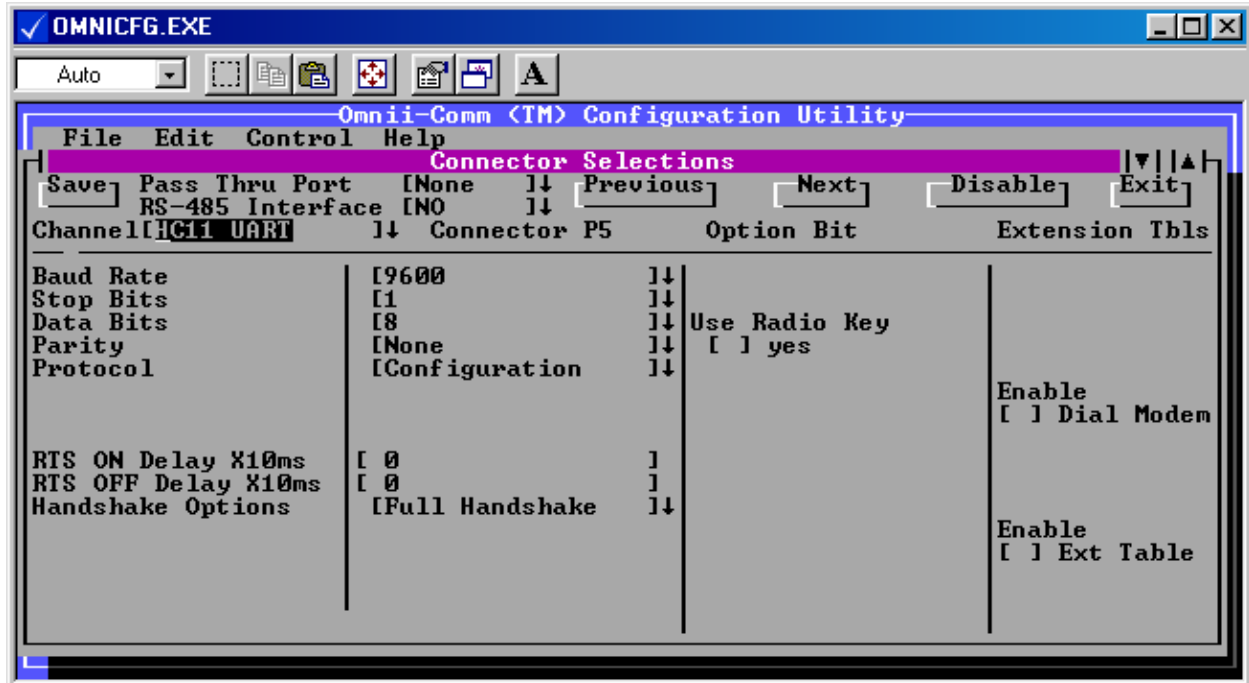
The Connector Selections screen will appear.

Use the Previous and Next buttons to move between Omnicomm connectors. Depending on the model number you have, some connectors will not be valid selections.

**NOTE: The Omnicomm program is used for all Omnicomm models and does not restrict you from configuring a connector that may not be available on your hardware. If you configure a connector that is not available, then you will get an error when you attempt to run the configuration (start polling).**

## Caterpillar M5X to Modbus MASTER Configuration Example

For each connector (P1, P2, P3, P4 and P5) select the protocol you wish to assign and a “Channel” to use for it. Channel assignments are unique; you can only use a channel name on a single connector. If you try to assign the same channel name to two different connectors, you will get an error message when you attempt to save the screen. After you have selected all the options, save the screen by pressing the Save button



One of the connectors on every Omni-Comm is designated as a “Configuration” connector. This is the connector to which the PC is connected, enabling you to download a new configuration file. You can FORCE an Omni-Comm to go to its configuration mode and reconfigure its Configuration connector for download by pressing the RESET button on the module. In all current Omni-Comm models, the “Configuration” connector is either P1 or P5. Consult the documentation for your specific model to determine what connector is used as the configuration connector for your hardware. You should factor the Configuration connector into your connector assignments. It is very convenient to define the Configuration connector (P1 or P5) with the Configuration protocol. This will leave the Configuration connector active when you start the module and provide a “window” into the module for debugging.

When you select a Protocol, you will be presented with additional choices on the Connector Selection screen that are appropriate for the protocol selected. Think of the “Channel” as a communication resource such as COM1, COM2, etc., on your PC. The Omni-Comm has several “Channels” and each provide different features. Some protocols require a specific ‘Channel’ to be selected for operation. If this is the case, it will be noted in the help files for that specific protocol.

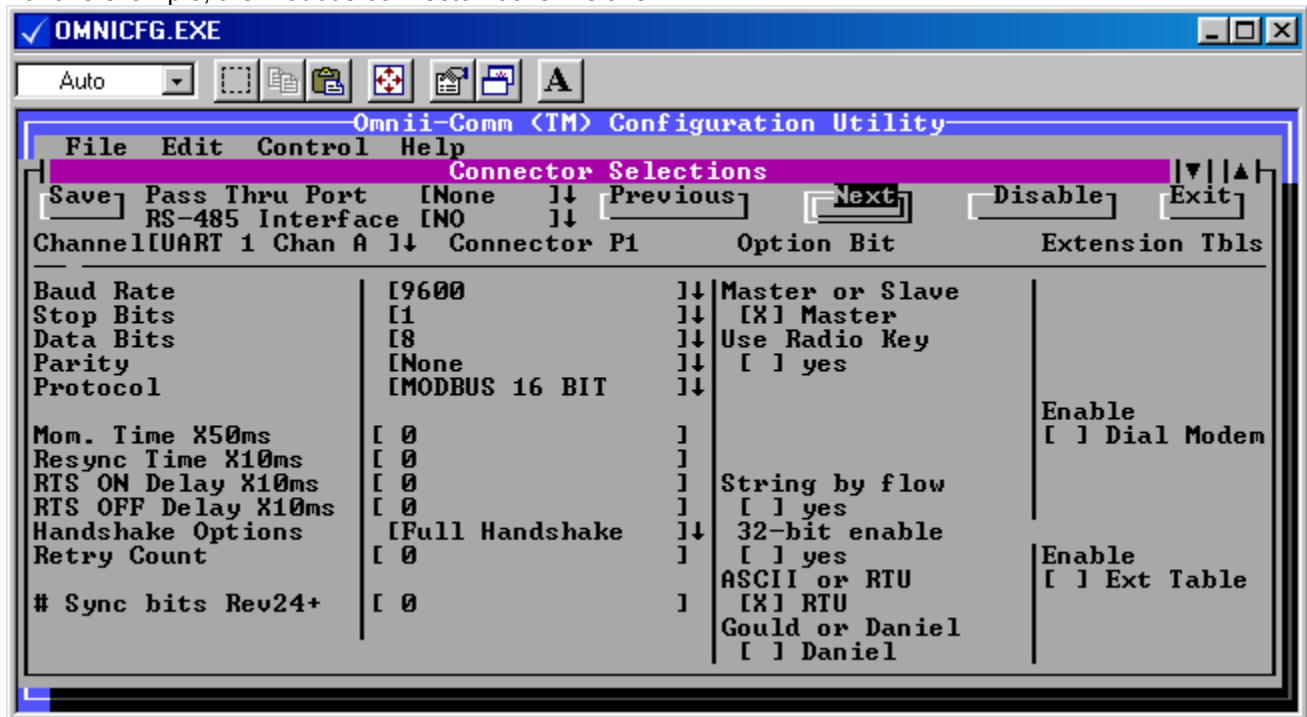
**THIS** example only uses the standard serial resources; HC11 UART, UART1 Chan A, UART1 Chan B.

This example configures the Omni-Comm to use the P5 connector as the default configuration connector, P1 as the Modbus connector and P2 as the Caterpillar connector. This would be appropriate if you have a DIN rail mounted Omni-Comm since it uses the P5 connector as the Configuration connector.

**Again, you are free to choose the protocol for each connector depending on your system constraints.**

## Caterpillar M5X to Modbus MASTER Configuration Example

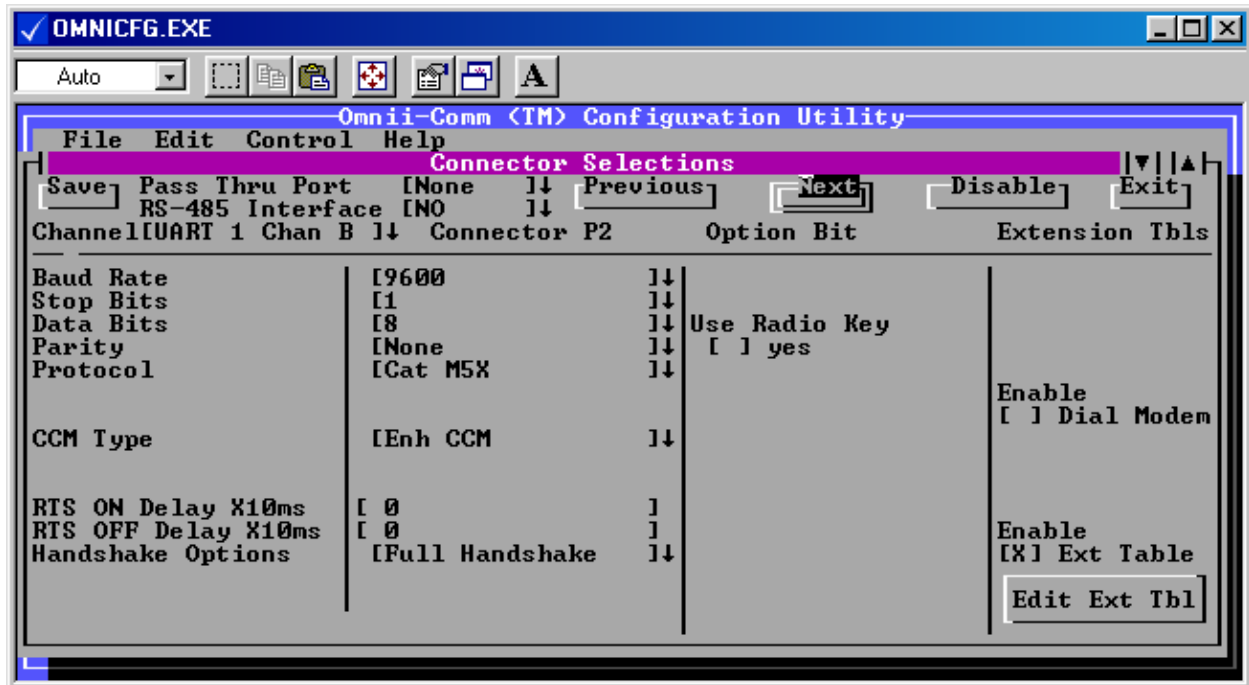
For this example, the Modbus connector looks like this:



The example configures the Modbus connector (P1) for operation at 9600 baud, 8 data bits, 1 stop bit and no parity. We are set to use Modbus RTU protocol with standard addressing. We are a Modbus Master because the Master option box is checked.

## Caterpillar M5X to Modbus MASTER Configuration Example

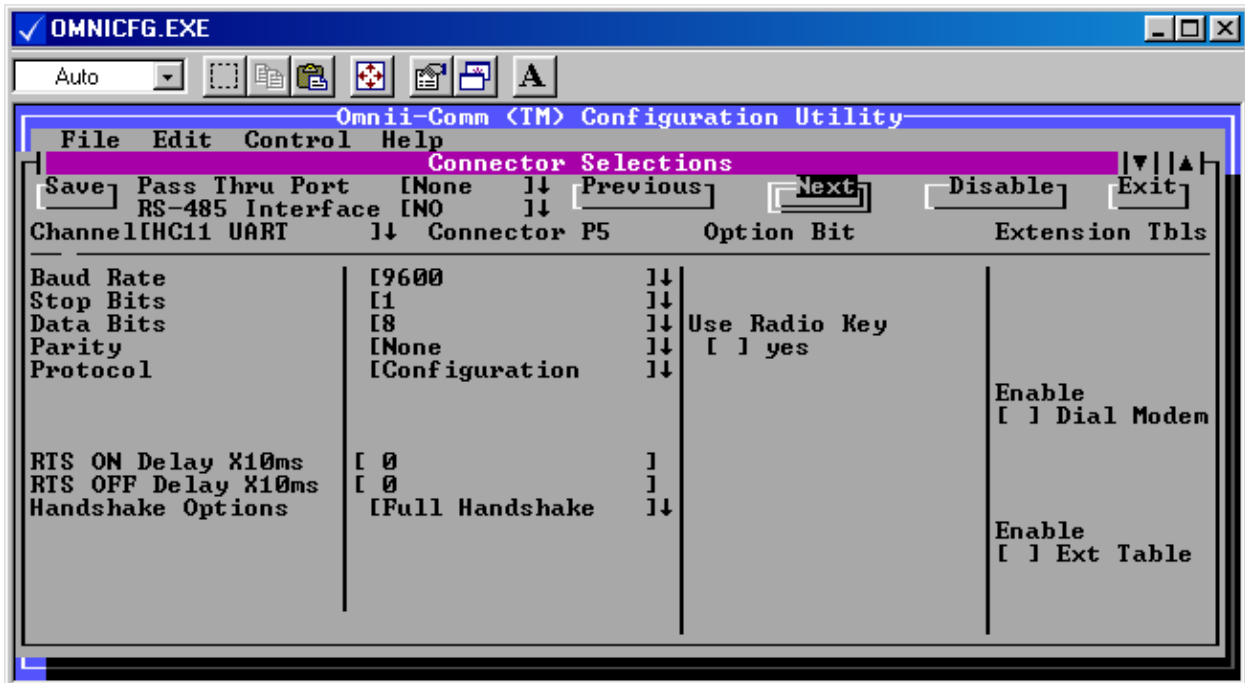
The Caterpillar connector is P2. The configuration screen for this connector looks like this:



We are set up for the standard CCM communications parameters of 9600 baud, 8 data bits, 1 stop bit and no parity. We have selected the Enhanced CCM so we can use the Advanced communication commands which will give us access to all the data from the ADEM III and ICSM controllers. Note that the Protocol Extension table is enabled. The Protocol Extension Table provides some key information which we will review in detail later on in this description.

## Caterpillar M5X to Modbus MASTER Configuration Example

Connectors P3 and P4 are Disabled.



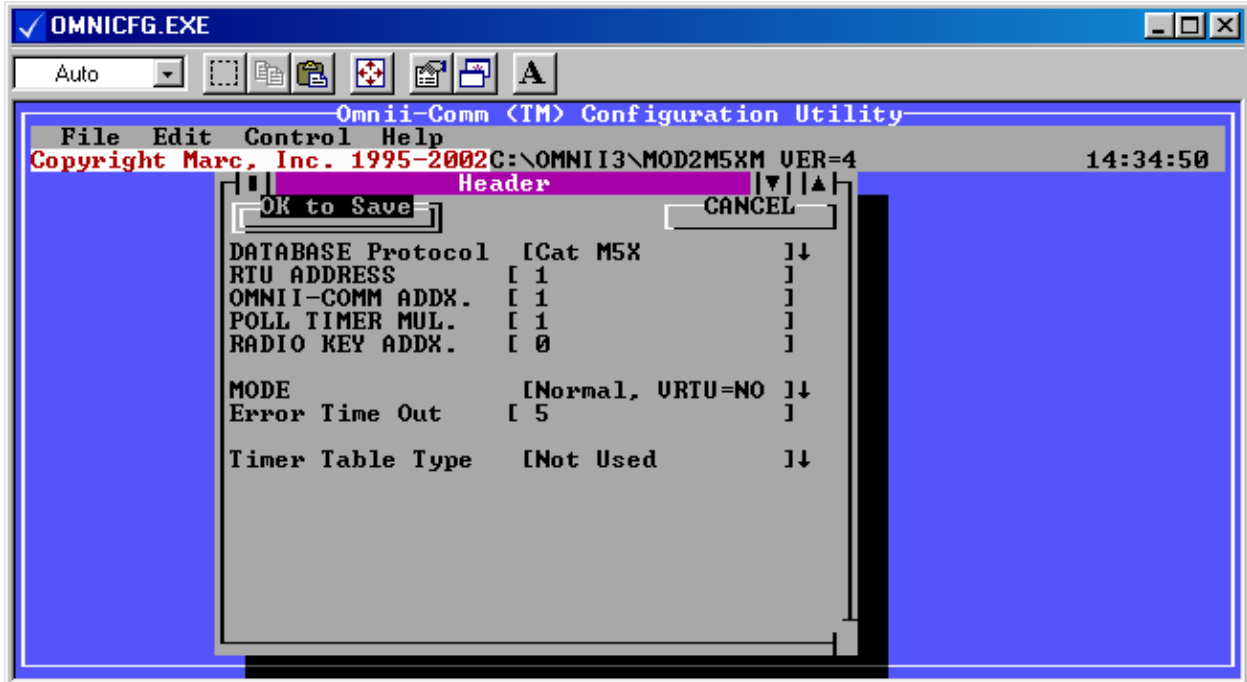
Connector P5 is used as a debug/diagnostics connector when the module is running. This provides a convenient "window" into the Omnii-Comm while it is in operation that can be used during troubleshooting.

### 4.4 Common Information

#### From the Main Edit screen select Edit then Header

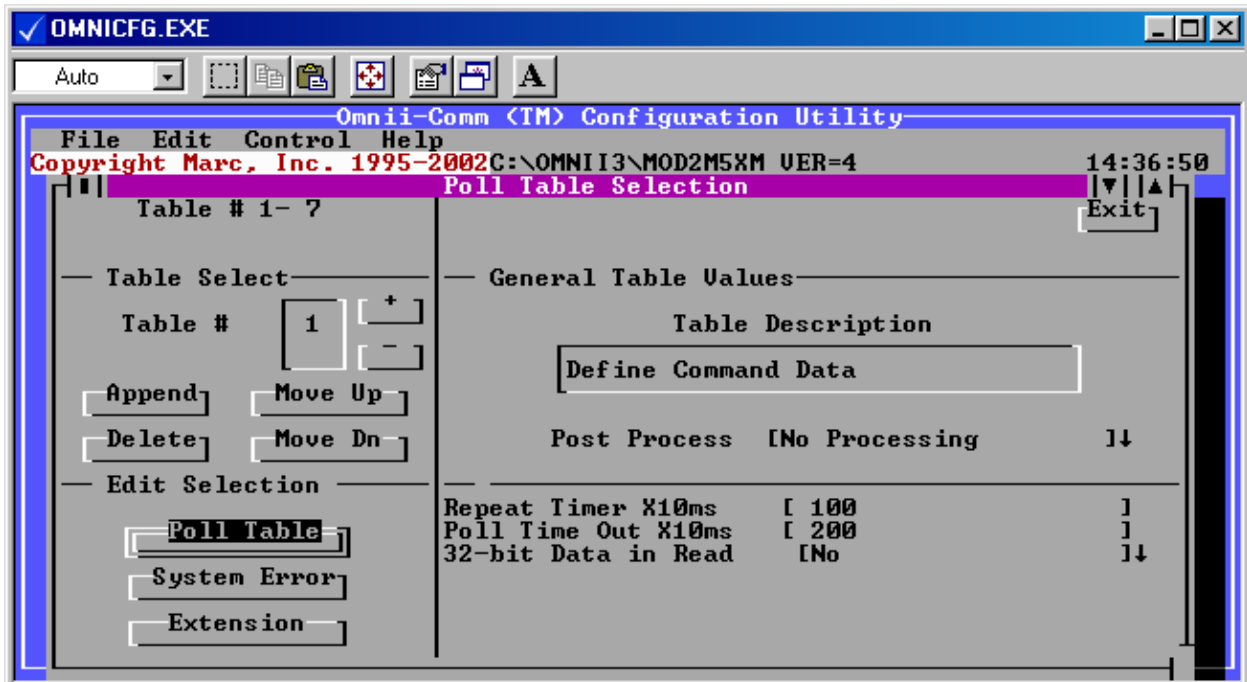
The Header configuration screen will appear. This screen is where you will enter information that is common to all ports on the Omnii-Comm.

The Omnii-Comm database organization is selected as CAT M5X for convenience. The Modbus RTU address is not used. The Omnii-Comm Address, Poll Timer Multiplier, and Radio Key Address are not used in this application. The Operating MODE is selected to be Normal, VRTU=NO and the Error Time Out set to 5 seconds. We are not going to use the Dynamic Poll Table Timers so the data type field is set to Not Used.



## 4.5 Poll Tables

From the Main Edit screen select Edit then Poll Table



The Poll Table Selection screen will appear. Polling tables are where the bulk of the work in an Omnicomm application takes place. We will review the poll tables for this application in detail. First, take a close look at the Poll Table Selection screen. There is a lot of information displayed for you to use.

First, you can see that this configuration has 7 tables (Table # 1-7). Table number 1 is the Current Poll number because it is displayed in the Table # window. You can make another table the current one by clicking on the up and down arrows to the right of the window. There are four buttons in the table select portion of the window that are used for table editing. **Append** makes a copy of the current table # and puts it at the end of the list of poll tables. **Delete** removes the current table number and closes up the list of poll tables. For example, if you select table 2 and click on the Delete key, table 2 will be removed, table 3 will move up to fill the spot where 2 was and table 4 will move to position 3. The order of the poll tables is important and can be easily changed using the **Move Up** and **Move Down** buttons. Clicking on the Move Up button “bubbles” the current table number up one position; Move Down does the opposite.

**NOTE: Do not use the Move Up and Move Down buttons to try to navigate to a new table number. Use the arrows for this. Move Up and Move Down are used only to rearrange the poll table order.**

On the right hand part of the Poll Table Selection screen is the General Table Value information. The **Table Description** is a place to enter a brief description of the function that the table is doing, in this case, Define Command Data. Below the description are the **Repeat Rate** and **Poll Time Out** parameters. The Repeat Rate is how often the Omnii-Comm will attempt to run the poll table. The Poll Time Out is how long the Omnii-Comm will wait for a table operation to complete once it is started.

On the lower left are three buttons that select the section you want to edit or view. **Poll Table** will take you to the Current Poll table edit screens, **System Error** will take you to the common System Error edit screen and **Extension** will take you to the Current Poll tables Extension Table if there is one.

#### 4.5.1 Poll Table #1; Define Command Data

**From the Poll Table Selection screen make Table #1 the Current Poll number and click the Poll Table button**

Before we dive into the specifics for this poll table there are some general items that are important to note. First, every polling table has three sections: A Read section that tells the Omnii-Comm how to go about collecting some data; A Write section that tells us what to do with the data collected in the read section; and an Error section that tells us what to do in case we have problems with either the Read or the Write operation. Polling tables that have Database selected for the Connector in the Write section also have an “Extension” table that tell us how to sort the incoming data into the database. The sections are accessed by clicking on the small R, W, E and Ext buttons located on the top left of the Table Selections screen, just below the Save button. The Current Poll Table Number and Section is shown for reference at the top of the table Selections Screen. Shown at the bottom of the screen is the Channel and Protocol that will be used for the operation. Remember, the Channel is uniquely assigned to a specific Connector on the Connector configuration screens that were completed earlier. The first thing to define on any poll table selection screen is the Connector to use for the operation that you want to perform.

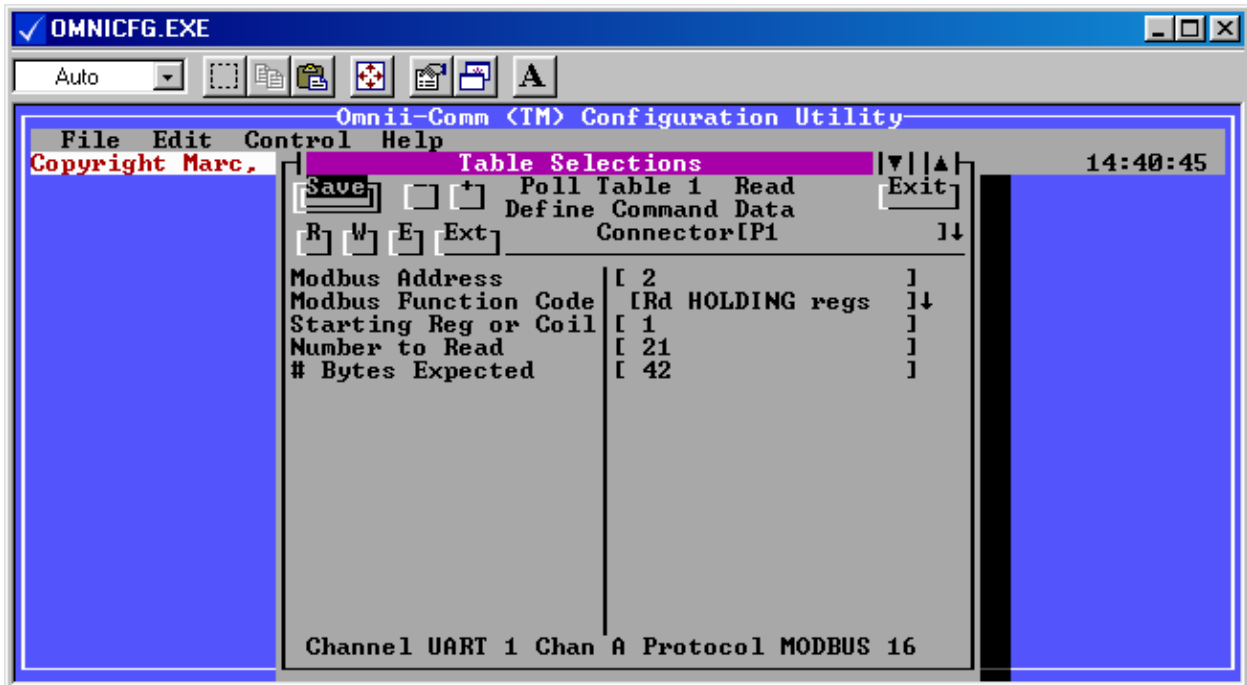
**NOTE: The Omnii-Comm has two “pseudo” connectors that can be enabled. They function like a real hardware connector but do not actually occupy any hardware resources. They only support one protocol each and the protocol cannot be changed. They are called Local RAM (VRAM) and Database. If you select Local RAM or Database in your current protocol set, you will automatically have a connector assigned for them, even if you do not ever reference them in your configuration.**

Now on to the specifics of this example:

## Caterpillar M5X to Modbus MASTER Configuration Example

### Poll Table #1, Read

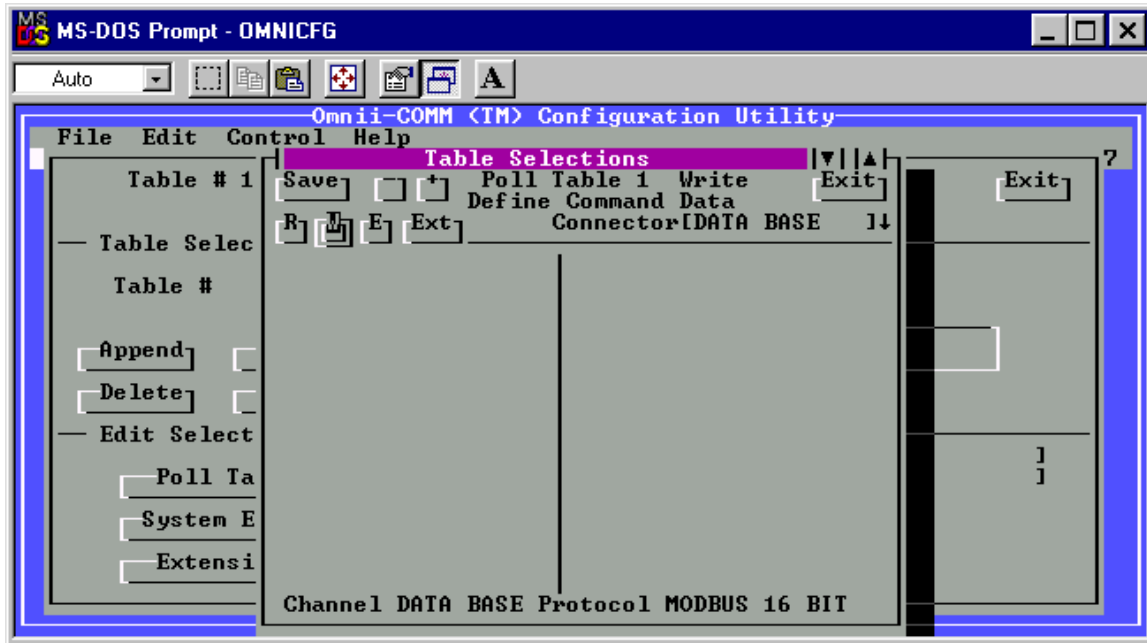
The Read section of Polling Table 1 is automatically opened when you click on the Poll Table button on the Poll Table Selection screen.



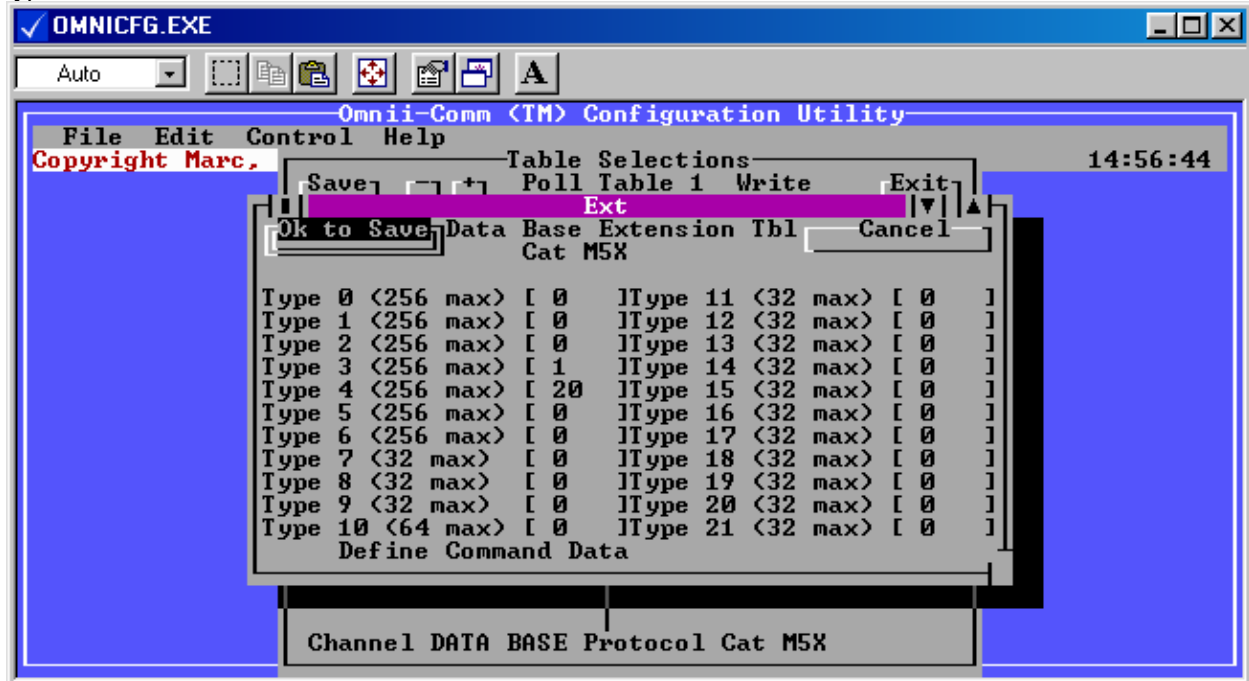
The connector selected is P1, the Modbus Master port. This table reads 21 Holding Registers from Modbus Slave 2. For this application, the first Holding Register will be used for Command Triggers and the remaining 20 Holding Registers for Command Data. This allows the Modbus Slave device to initiate Commands to the CCM. The 21 Holding Registers can come from anywhere in the address space of the Modbus Slave. The write section, explained next will define what type or types of registers are created.

**Poll Table #1, Write**

Clicking on the W button opens the Write section of Poll Table #1.



The Connector selected for the Write operation is Data Base so the important information will be contained in the Extension Table. **Click on the Ext button** to see the Poll Table Extension Table. This table is used to define how to distribute the data from the read operation into the database of the Omnii-Comm. We have selected CAT M5X on the Header screen as our database type so we have CAT data types to select from.

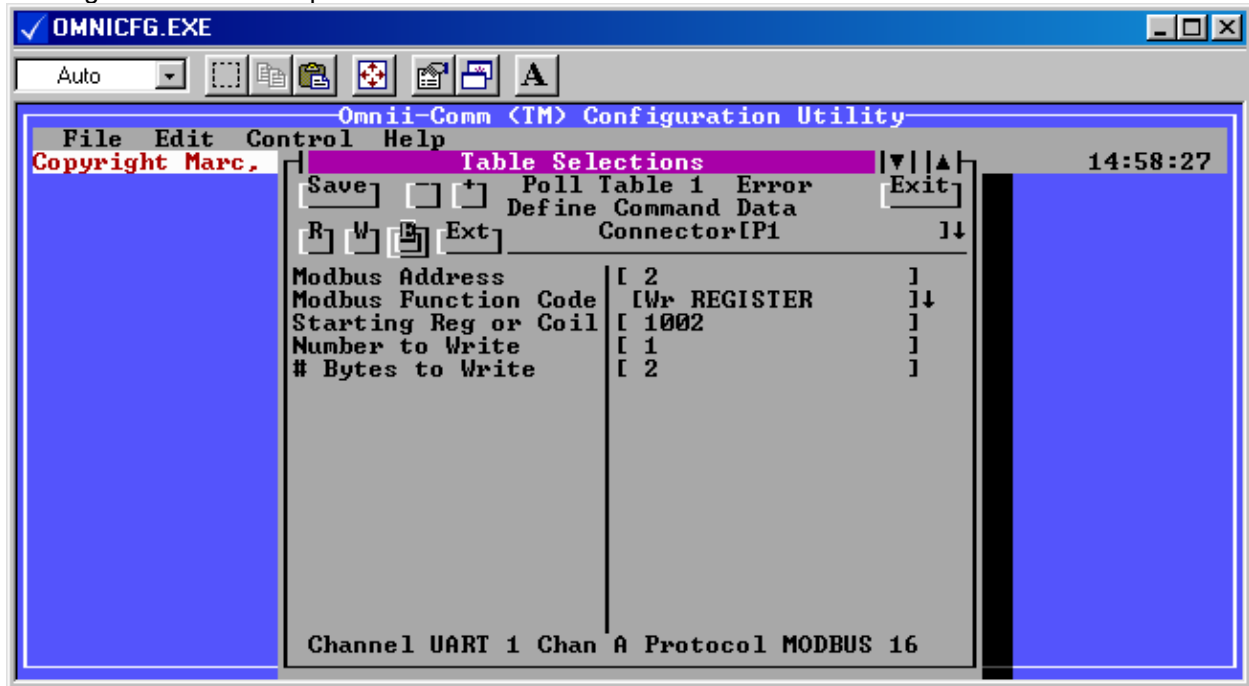


## Caterpillar M5X to Modbus MASTER Configuration Example

The Extension Table says that we are to create 1 word of Type 3 and 20 words of Type 4 data from the 42 bytes we got from the read section. More about this later.

### Poll Table #1, Error

Clicking on the E button opens the Error section of Poll Table #1.



Errors, if they occur are written to Holding Register 1002 for this application. This is purely an arbitrary location. You can change this to any address that is free in your system.

### 4.5.2 Define Registers for Return Data

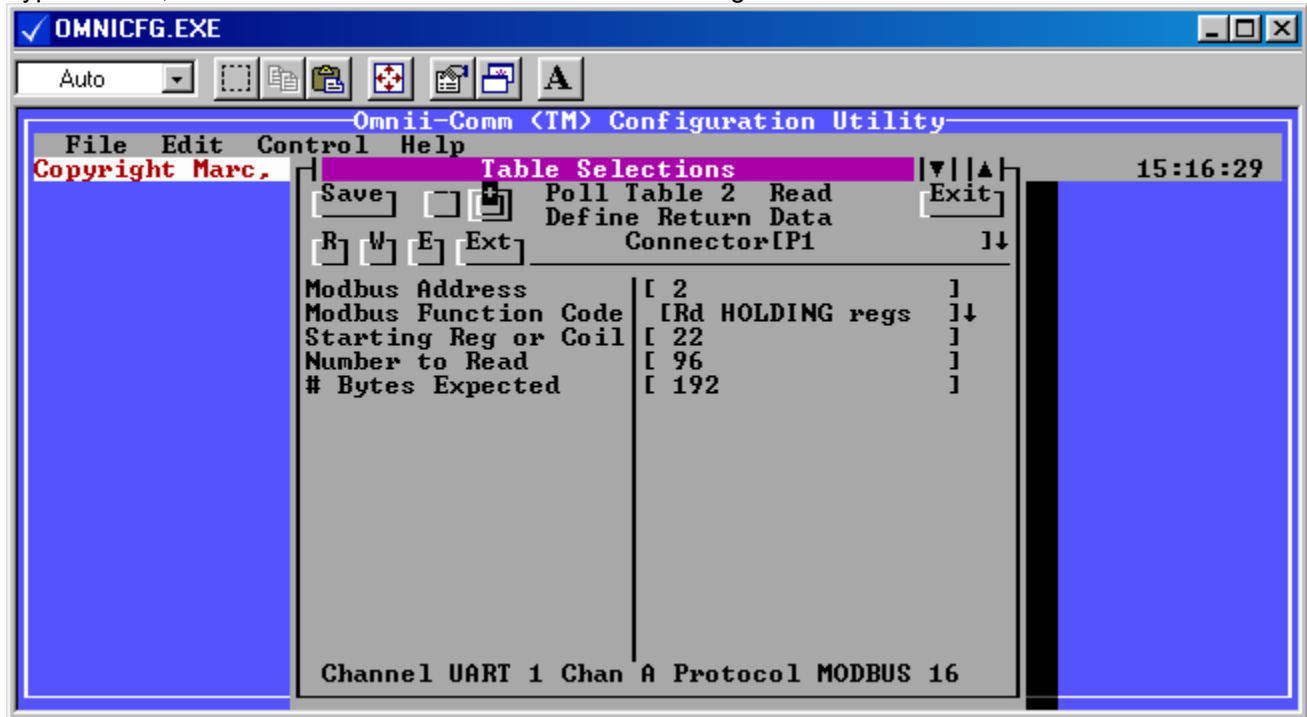
Polling tables # 2 through 4 define Database registers that will be used to store data received from the CCM. Table #2 "creates" 96 registers by reading 192 bytes from the Modbus Slave and Writing to Data Base Type 1. Poll Table #3 makes 96 more and Poll Table #4 makes 63 for a total of 255. When the CCM broadcasts a list to the Omnii-Comm, the information will be written to database to a Data Type and Offset as specified in the Poll Table that configures the broadcast list. The Advanced Broadcast list can return 1 byte, 2 byte and 4 byte data depending on the PID specified. Data is written to the registers in the order received starting with the specified offset. One byte data will be written to the low byte of a register and the high byte set to 0. Two byte data is written to a register in the order received and four byte data is written to two adjacent registers in the order received.

There is a good reason for the "strange" byte count in the polling tables. When a write operation is performed to an Omnii-Comm database the data is automatically written back to the source of the database point. In this case, writes to Data Type 1 write back to Holding Registers in the Modbus Slave. We plan to allocate 16 registers per list to be able to hold up to 8 4 byte values. The Omnii-Comm can only generate a single write operation for each database write. If the registers fall across two polling tables we will be unable to perform the write back operation. The byte counts selected define an even number of 16 register groups per poll table.

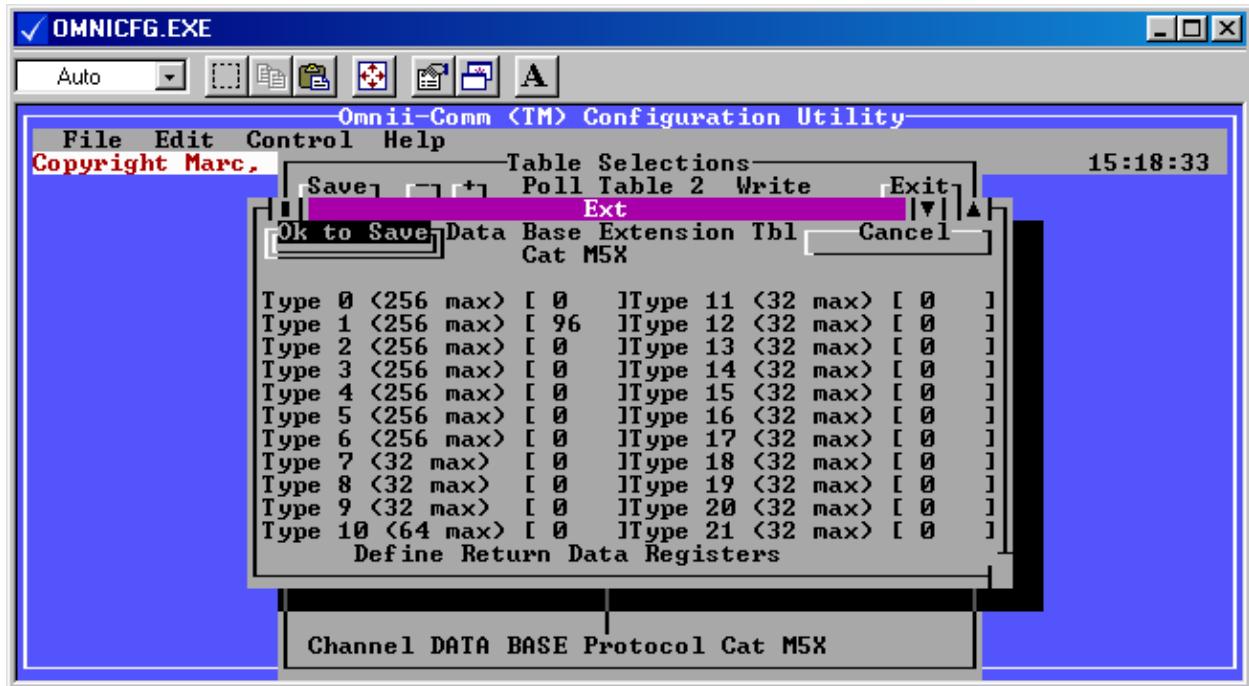
Caterpillar M5X to Modbus MASTER Configuration Example

Poll Table #2 Read

Typical Read, Write and Error sections for Poll Tables 2 through 4.

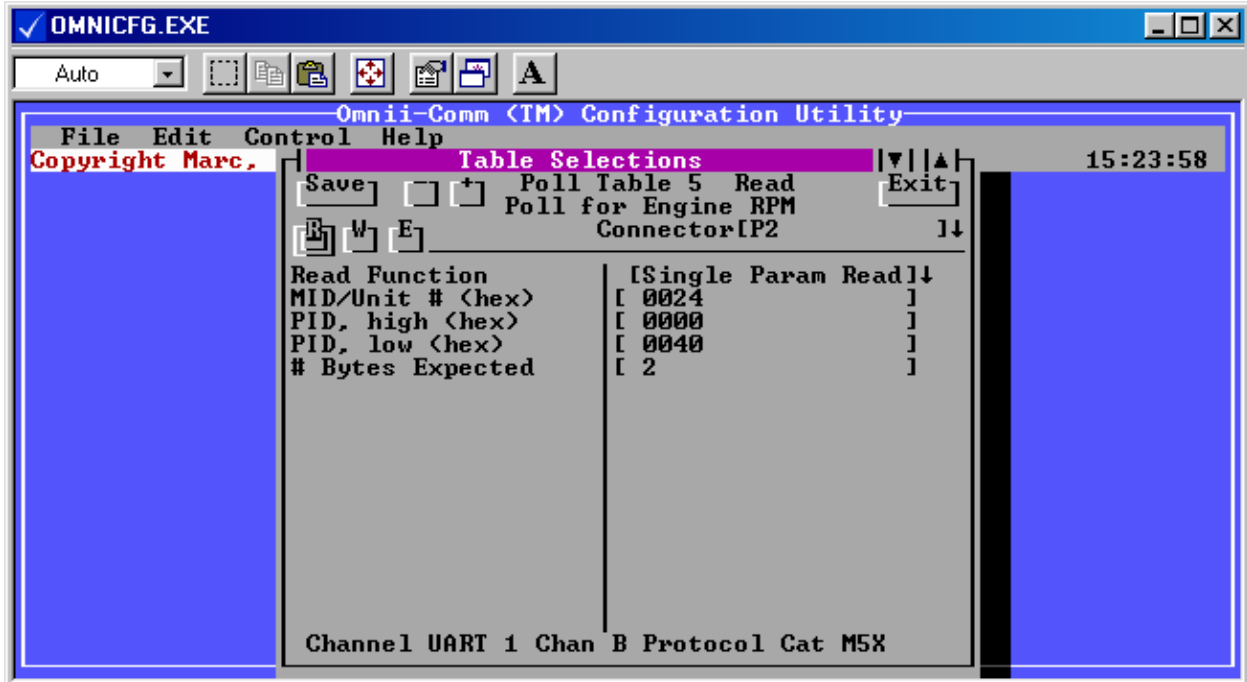


Poll Table #2 Write/EXT



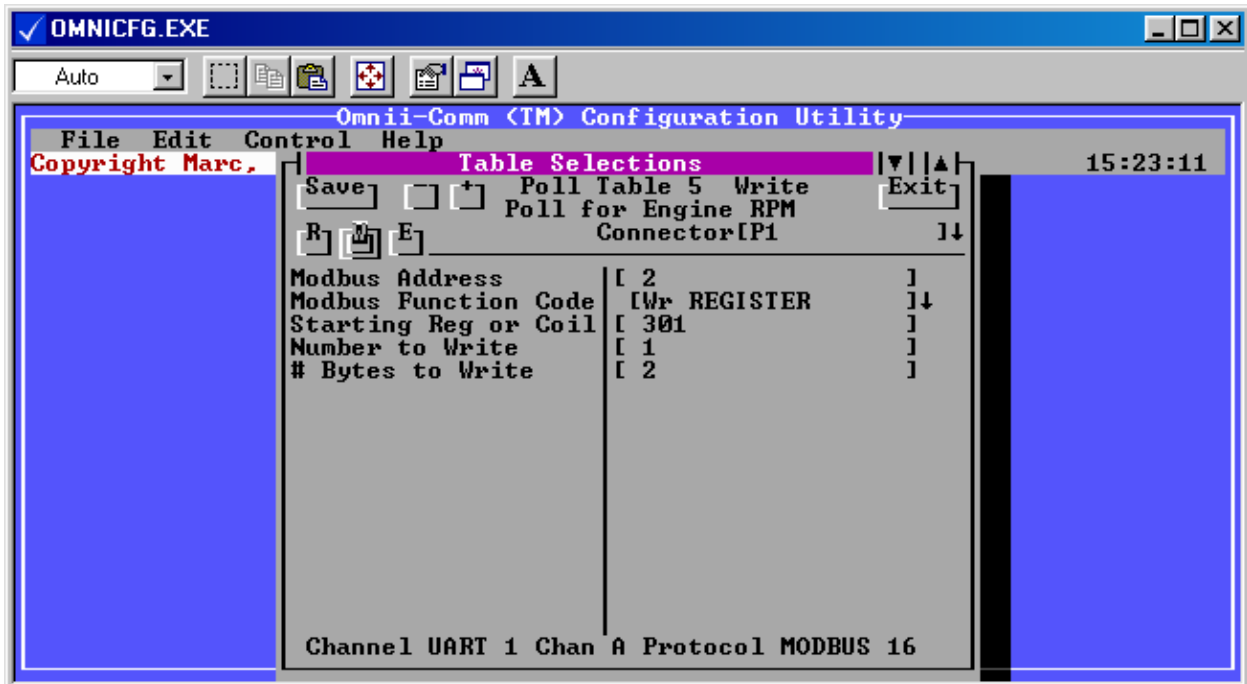


Caterpillar M5X to Modbus MASTER Configuration Example



**Poll Table #5 Write Section**

The Write Section of Poll Table #5 tells us to put the Engine RPM collected in the read section into Holding Register 301 in Modbus Slave 2.

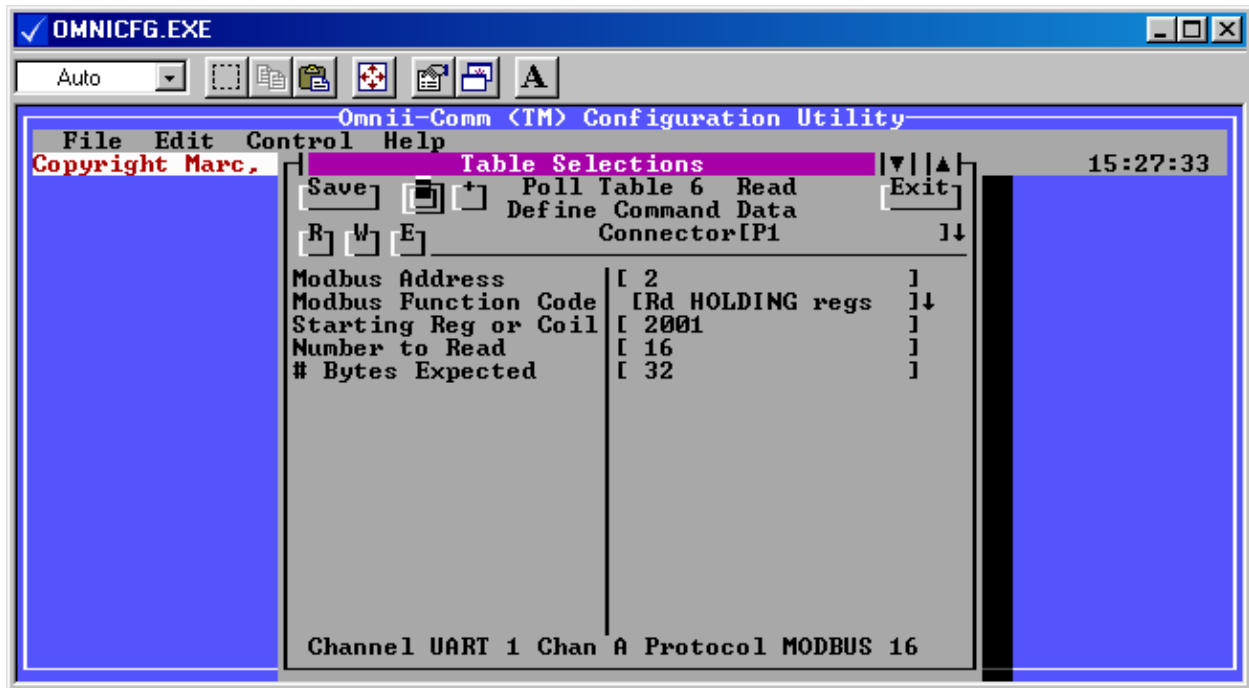


#### 4.5.4 Define Broadcast Lists

You can create from 1 to 16 Polling Tables to configure up to 16 CCM Broadcast lists. This example only sets up the first two lists. Polling Tables #6 and 7 set up Advanced Broadcast Lists #1 and #2.

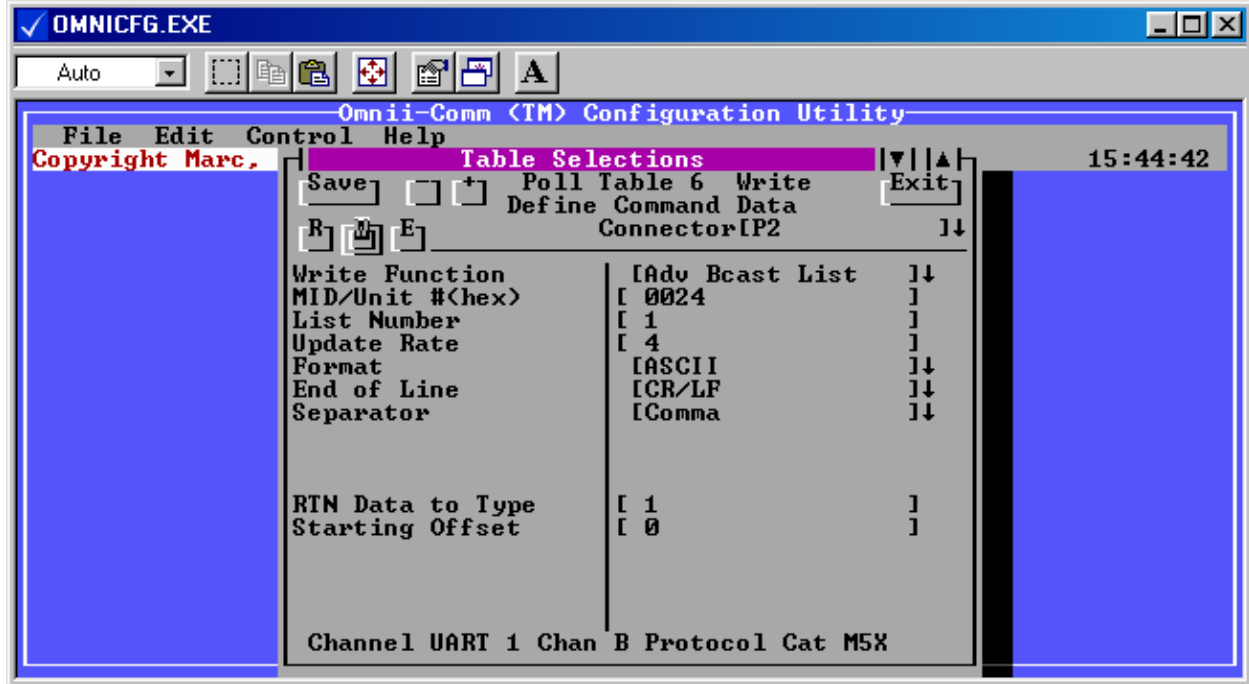
##### Poll Table #6 Read

The Read section of Poll Table #6 tells the Omnii-Comm to get 16 registers (32 bytes) of data starting at Holding Register 2001 of Slave 2. The contents of these registers will be the desired Parameter Ids (PIDs) for the data that will be returned in Broadcast List #1. Parameter Ids are read as two consecutive Holding Registers. The most significant byte of the first register is always 0, the least significant byte is the first PID digit and the next register has the last two PID digits. For instance, Engine RPM would be stored as \$00 \$00, \$00 \$40 in two consecutive registers. Cylinder 1 Detonation Level would be represented by \$00 \$D0 \$00 \$20. Note: All the numbers are in hex format. We will terminate the list when we find two consecutive registers of value \$0000. The 16 Holding Registers can come from any starting address.



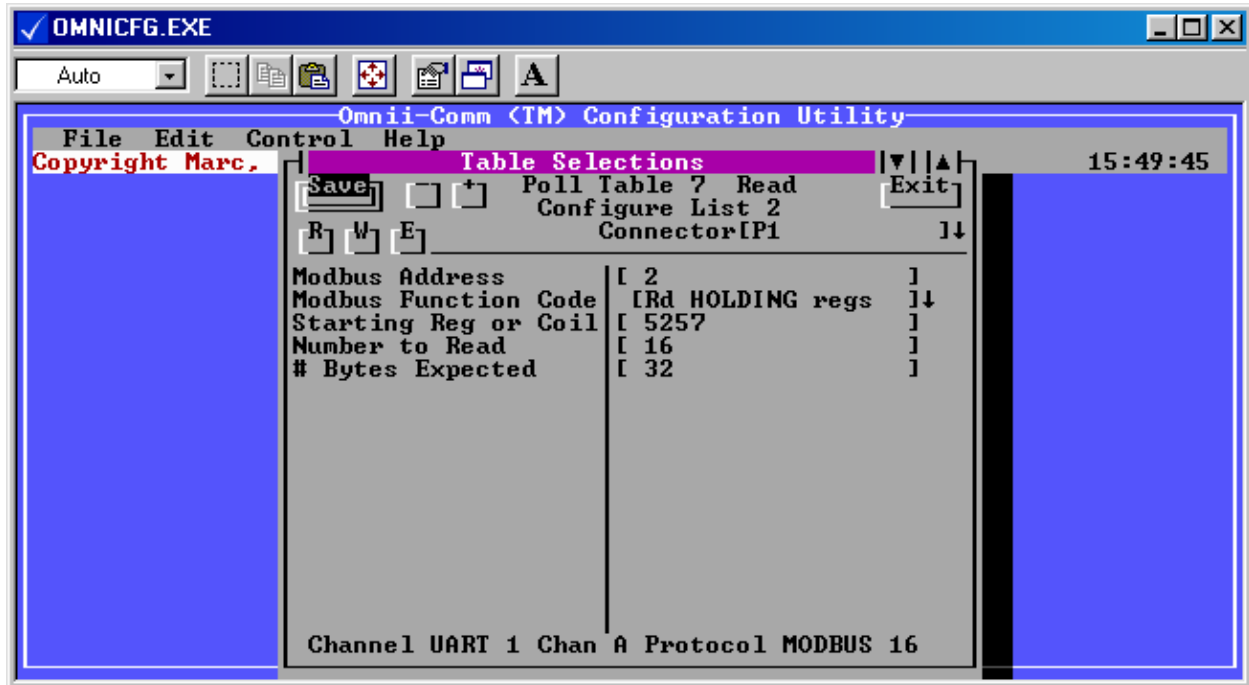
**Poll Table #6 Write**

The Write Section of Poll Table #6 selects P2, the Caterpillar connector, for the write operation. Notice that we have selected Advanced Broadcast List as the function to perform. The data will come from Module ID \$24, the ADEM III controller. We are configuring List 1 with an update rate of 2 seconds (4 X .5). Data will be returned in ASCII format with a CR/Line Feed at the end of each line. PID values will be comma separated. This setting will allow you to use a standard terminal program such as Omnii-Talk to monitor the CCM messages. The data returned in list #1 will be written to Data Type 1 starting at offset 0. This Data Type and offset was defined in Poll Table #2 as coming from Holding Register 22 of Slave 2 so the broadcast list will be written by the Omnii-Comm to Modbus Slave 2 starting at Holding Register 22.



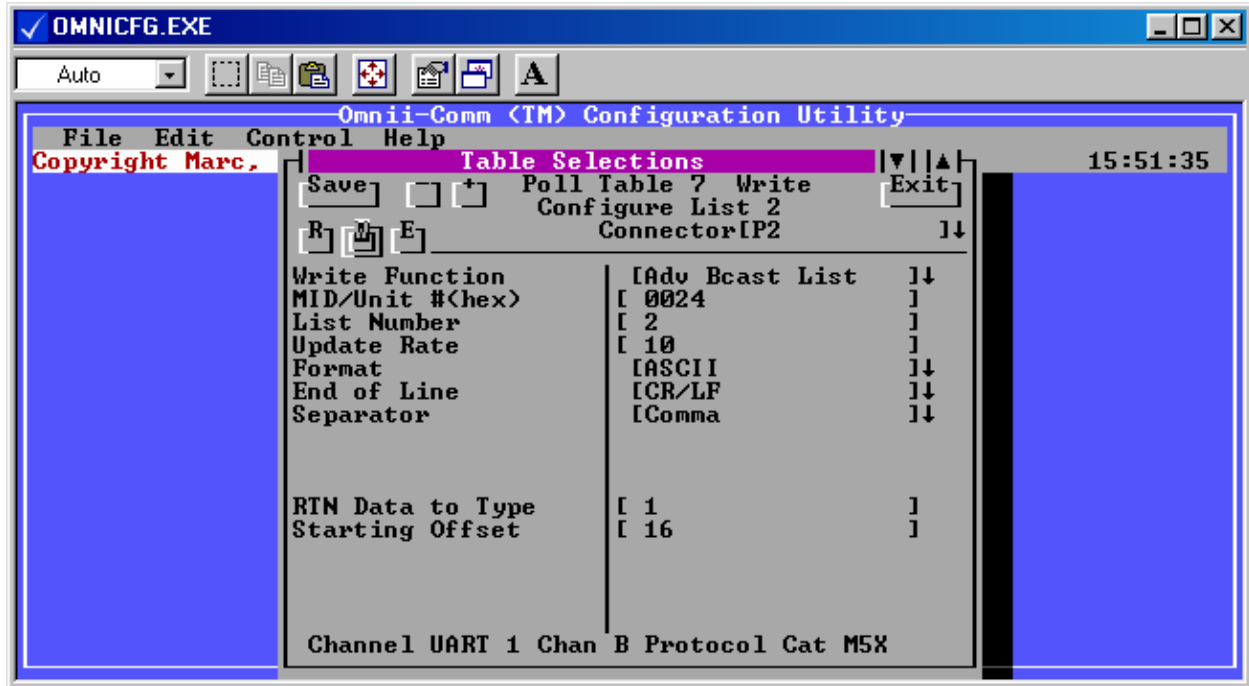
**Poll Table #7 Read**

The Read section of Poll Table #7 tells the Omnii-Comm to get 16 registers (32 bytes) of data starting at Holding Register 5257 of Slave 2. The contents of these registers will be the desired Parameter Ids (PIDs) for the data that will be returned in Broadcast List #2. Parameter Ids are read as two consecutive Holding Registers. The most significant byte of the first register is always 0, the least significant byte is the first PID digit and the next register has the last two PID digits. For instance, Engine RPM would be stored as \$00 \$00, \$00 \$40 in two consecutive registers. Cylinder 1 Detonation Level would be represented by \$00 \$D0 \$00 \$20. Note: All the numbers are in hex format. We will terminate the list when we find two consecutive registers of value \$0000. The 16 Holding Registers can come from any starting address.



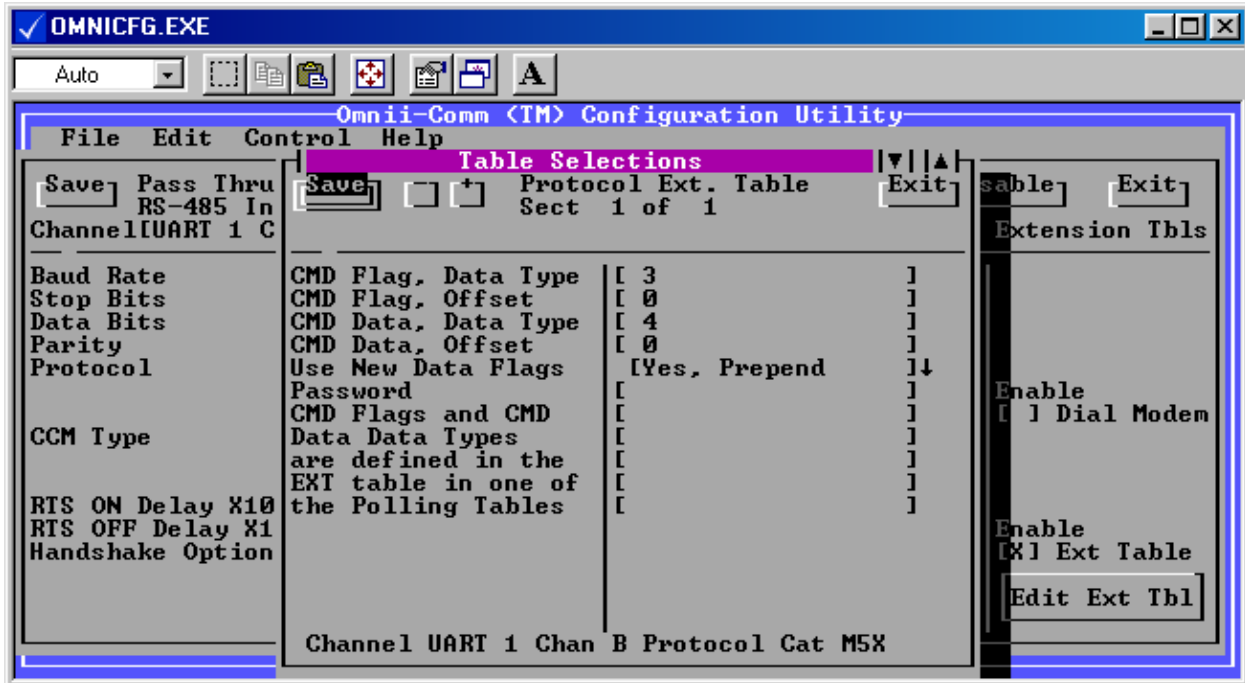
**Poll Table #11 Write**

The Write Section of Poll Table #7 selects P2, the Caterpillar connector, for the write operation. Notice that we have selected Advanced Broadcast List as the function to perform. The data will come from Module ID \$24, the ADEM III controller. We are configuring List 2 with an update rate of 2 seconds (4 X .5). Data will be returned in ASCII format with a CR/Line Feed at the end of each line. PID values will be comma separated. This setting will allow you to use a standard terminal program such as Omnii-Talk to monitor the CCM messages. The data returned in list #2 will be written to Data Type 1 starting at offset 0. This Data Type and offset was defined in Poll Table #2 as coming from Holding Register 38 of Slave 2 so the broadcast list will be written by the Omnii-Comm to Modbus Slave 2 starting at Holding Register 38.



## 4.6 Caterpillar M5X Protocol Extension Table

Remember, back in Section 4.3 of this tutorial, we said that we would be back to talk about the Protocol Extension Table? **Well, pay attention!** This is important!



**From the Main Edit screen select Edit then Connector. Use the Next button to advance to the Caterpillar connector and then click on the Edit Ext Tbl button.**

The Protocol Extension table is unique for each connector. That is, you can define different settings for each connector. This example has only one CCM so we only have to pay attention to the screen capture above. It is possible to have multiple CCM modules connected to other serial ports on an Omnii-Comm. The Extension Tables are unique for each connector. **We use the Protocol Extension table in Caterpillar to link Caterpillar information to the database.**

The first two entries, CMD Flag Data Type and Flag offset is where you tell the Omnii-Comm to look for Command Flags. The Omnii-Comm continuously monitors the Command Flags. When a Command Flag changes state from OFF to ON, a Command is issued **FROM THIS CONNECTOR** to the CCM that it is connected to. The specific command depends on the bit position (Flag) that changes state. See the tables following or the Excel spreadsheet for a definition of what command each bit will initiate. Likewise, the next two fields define the Command Data Type and Offset that will be used by the Omnii-Comm to find information that it needs in order to **build** the command. For this example, we have instructed the Omnii-Comm to monitor Data Type 3, starting at offset 0 for Command Flags and Data Type 4 starting at offset 0 for Command Data. Data Type 3 and 4 are updated by Polling Table #1 by reading Modbus Slave 2, Holding Registers 1 thru 21. Holding Register 1, bits 1 thru 8 are used as command triggers. Holding Registers 2 thru 21 are used for command data.

The Use New Data Flags option has three possible selections, No, Yes, Prepend and Yes, Postpend. Recall that the CCM is broadcasting data to us. It may be useful to know when new data has arrived and what the quality of that data is. If the selection is No, then we will not write a New Data flag. If the selection is Yes then we will add a "flag" word either at the front of the data (prepend) or at the end of the data (postpend). The Flag Word will have \$FF in the high byte. The low byte will be a copy of the

## Caterpillar M5X to Modbus MASTER Configuration Example

Recently Updated Data (RUD) byte from the Advanced Broadcast Response message or \$00 if not from an Advanced Broadcast List. There is 1 bit per PID in the list. 1=not updated, 0=updated. Bit 0, the Least Significant bit (rightmost) represents the status of PID 1, Bit 7 represents the status of PID #8.

The CCM can be optionally password protected. The default password is all blanks. If the password has been changed, then you must place the password in this extension table. Passwords are any combination of up to 8 numbers and letters. No spaces are allowed.

### To summarize:

Commands are sent by writing to Modbus Slave 2 Holding Register 1 bits 1 thru 8 (Bits 9 through 16 when set report the successful completion of the command operation). If an error occurs, the CCM error response will be written to the System Error location, in this example Modbus Slave 2, Holding Register 1001.

Command Data is read from Modbus Slave 2, Holding Registers 2 through 21.

**NOTE that this data must be properly initialized by the Modbus master prior to initiating a command.**

CCM responses are directed to Modbus Input Registers 1 through 85.

### *4.7 Modbus Register/Bit Assignments*

The following table summarizes the Modbus Slave register numbers used for this example. Additional details are shown in the Excel spread sheet MB2M5X.XLS

<b>Holding Register 1</b>	<b>Bit #</b>	<b>Function</b>
	1	Single Parameter Read
	2	Single Parameter Write
	3	Program Composite Response
	4	Request Fault Codes (Older CCMs Only that support IID82)
	5	Auto Init Sequence Start
	6	Log on to CCM
	7	Special Parameter Command
	8	Spare
	9 – 16	Command Complete bits for the above commands

<b>Holding Register</b>	<b>Function</b>
2,3	Command PID
4	Command Flags
5	Command Update Time
6	Command MID/Unite Number
7	Parameter byte count (IID 00, 34)
8-21	Parameter Data for IID 00, 34 (1-27 bytes)

<b>Holding Register</b>	<b>Function</b>
22-276	Broadcast List Return Data
2001-2016	PID List for List 1
5257-5272	PID List for List 2
1001-1008	Error Registers