

Omnii-CommTM Example

Caterpillar

To

Modbus



TABLE OF CONTENTS

SECTION 1.....	1
<i>Overview</i>	1
1.1 WHAT IS THIS:.....	1
1.2 OTHER FILES YOU WILL NEED:	1
SECTION 2.....	2
<i>Concepts</i>	2
2.1 GENERAL CONCEPTS.....	2
2.2 CCM COMMUNICATION	2
2.3 OMNII-COMM COMMUNICATION	2
SECTION 3.....	3
<i>Starting the program</i>	3
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
SECTION 4.....	5
<i>Typical Configuration Procedure</i>	5
4.1 SELECT PROTOCOLS.....	5
4.2 LOAD MOD2CATM.DBA	6
4.3 CONFIGURE CONNECTORS	6
4.4 COMMON INFORMATION	10
4.5 POLL TABLES.....	11
4.5.1 Poll Table #1; Read Command Flags and Data	12
Poll Table #1, Read.....	13
Poll Table #1, Write.....	14
Poll Table #1, Error.....	15
4.5.2 Poll Table #2; Define New Data Available Flags.....	15
4.5.3 Poll Table #3; CAT Data Storage.....	16
4.6 CATERPILLAR PROTOCOL EXTENSION TABLE.....	17
4.7 MODBUS REGISTER/BIT ASSIGNMENTS	19

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 reads data from a Modbus device that is used to configure and control the CCM. Engine data received from the CCM is written to registers in the Modbus device. The Omnii-Comm is a Modbus Master in this example. Full support of all Caterpillar gas and diesel engines is maintained.

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>

MOD2CATM.DBA	Omnii-Config database file
MOD2CATM.XLS	Excel spreadsheet detailing register mapping for MOD2CATM.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 register information read from a Modbus Slave device and also report engine data collected by the CCM to registers in the Modbus Slave device.

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. Single word PIDs are placed in Modbus registers by the Omnii-Comm.

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's M50 protocol. It is configured to also have a Modbus Master port that will communicate with a Modbus slave such as a Modicon PLC. The Omnii-Comm will WRITE data received from the CCM to Modbus registers in the PLC when the CCM broadcasts each list. Other registers will be read from the Modbus Slave device that will be used to configure and control the operation of the CCM.

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

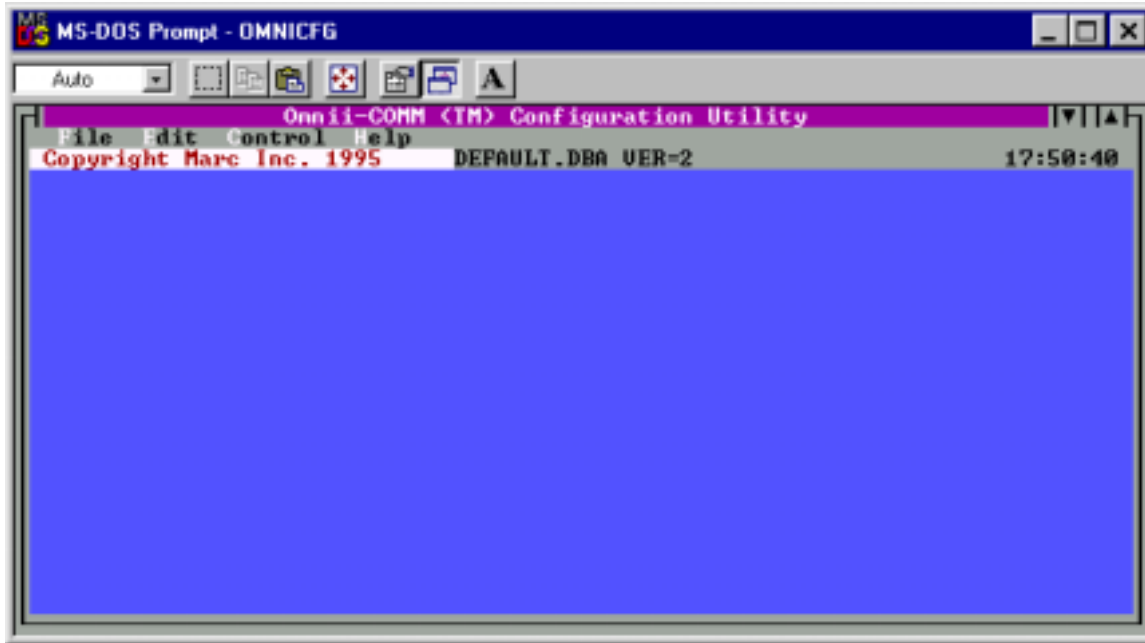
From the Program Manager window, select File, New, Program Item then OK
Complete the Program Items Properties screen
Description = Omnii-Config
Command Line = Config
Working Directory = name of the subdirectory where you saved the files

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 to Start Polling on the Omnii-Comm. 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

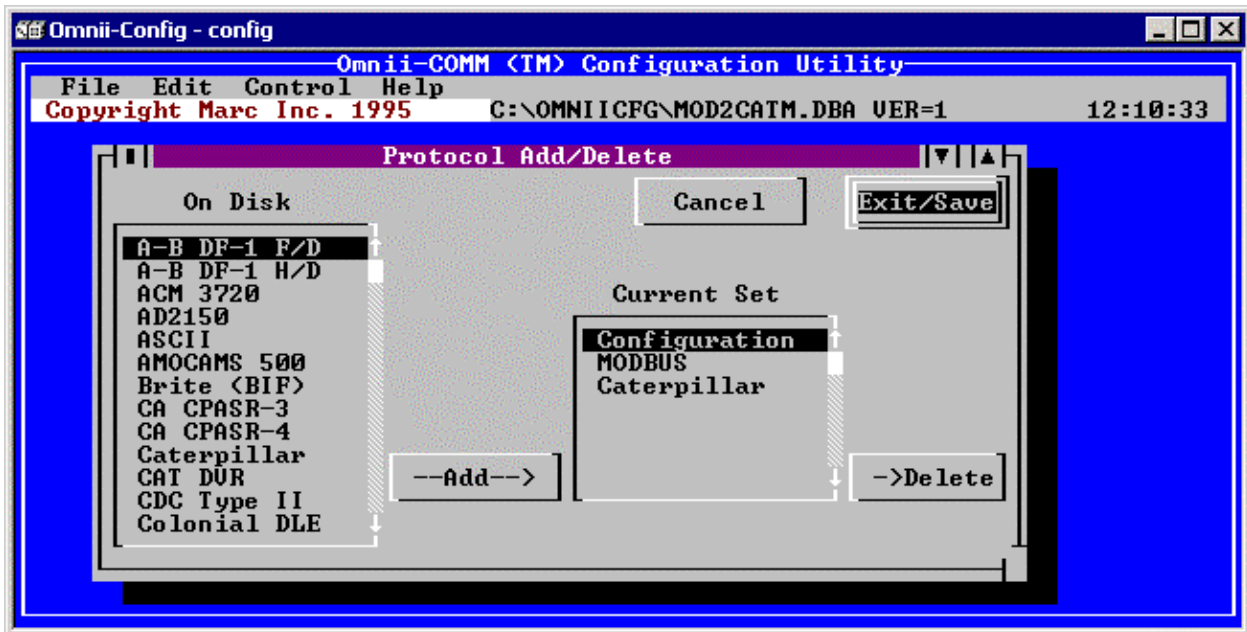
Typical Configuration Procedure

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.

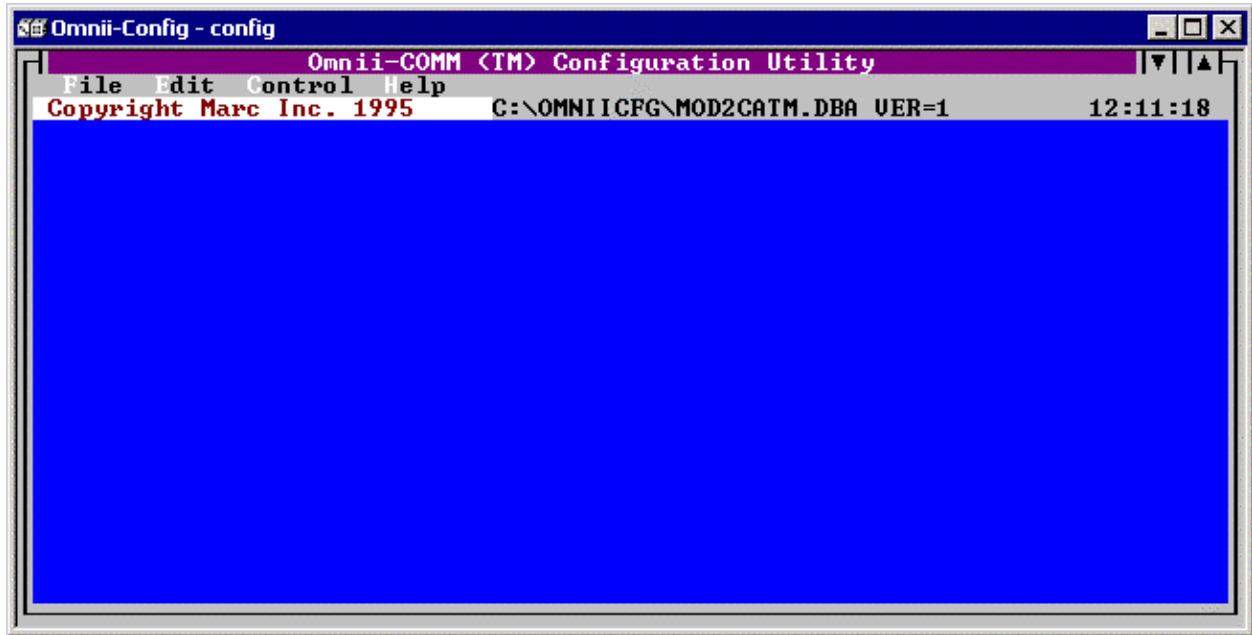
If you need to add a protocol to your current set, highlight it in the On Disk list and click on the Add button. To remove a protocol from the current set, highlight it in the Current Set list and click the Delete button.

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 MOD2CATM.DBA example file.

4.2 Load MOD2CATM.DBA

Load the example file MOD2CATM.DBA. To do this, first select File, then Open and navigate to the directory that contains the configuration file MOD2CATM.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.



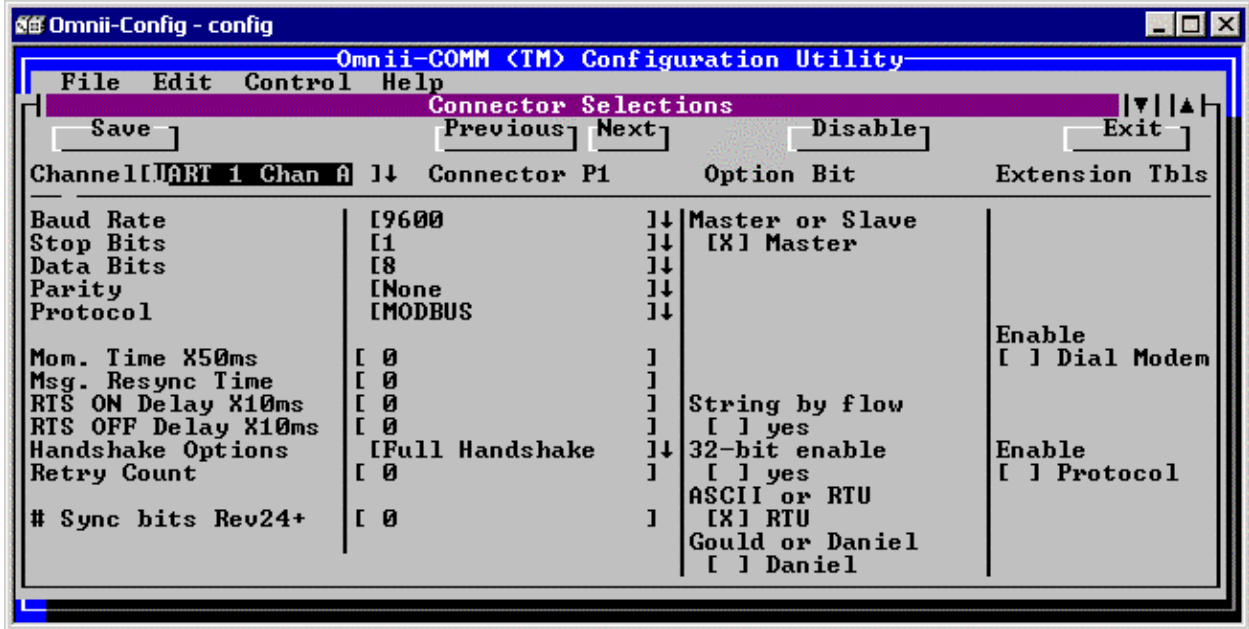
4.3 Configure Connectors

You can define the protocol to use on each Omnia-Comm 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.

Caterpillar to Modbus Configuration Example



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

NOTE: The Omnii-Config program is used for all Omnii-Comm 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).

For each connector (P1, P2, P3, P4 and Modem) 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. Note that 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 Omnii-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.

One of the connectors on every Omnii-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 Omnii-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 Omnii-Comm models, the "Configuration" connector is either P1 or Modem. Consult the documentation for your specific model to determine what connector is used as the configuration connector for your hardware. You should factor the Default Configuration connector into your connector assignment selections. It is very convenient to define the Configuration connector (P1 or Modem) 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. If you need it, you can configure the Default Configuration Connector as another protocol port at run time by simply defining it in you configuration.

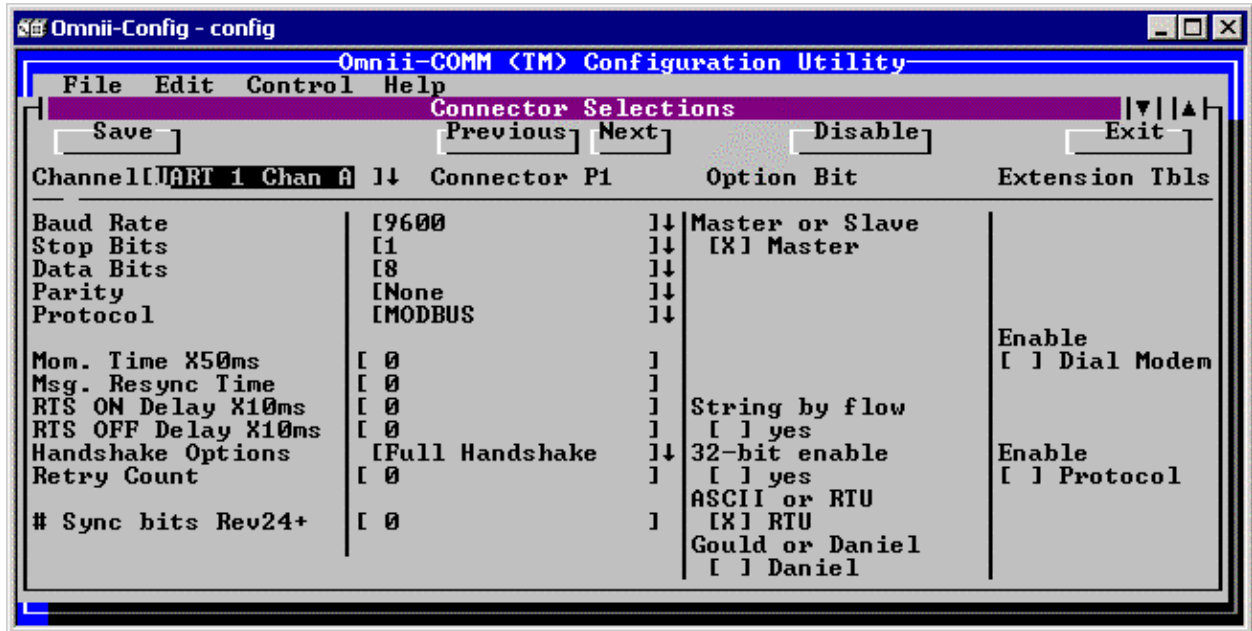
Caterpillar to Modbus Configuration Example

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

This example configures the Omnii-Comm to use the Modem connector as the Run Time configuration connector, P1 as the Modbus connector and P2 as the Caterpillar connector. This would be appropriate if you have a DIN rail mounted Omnii-Comm since it uses its Modem connector as the Default Configuration connector.

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

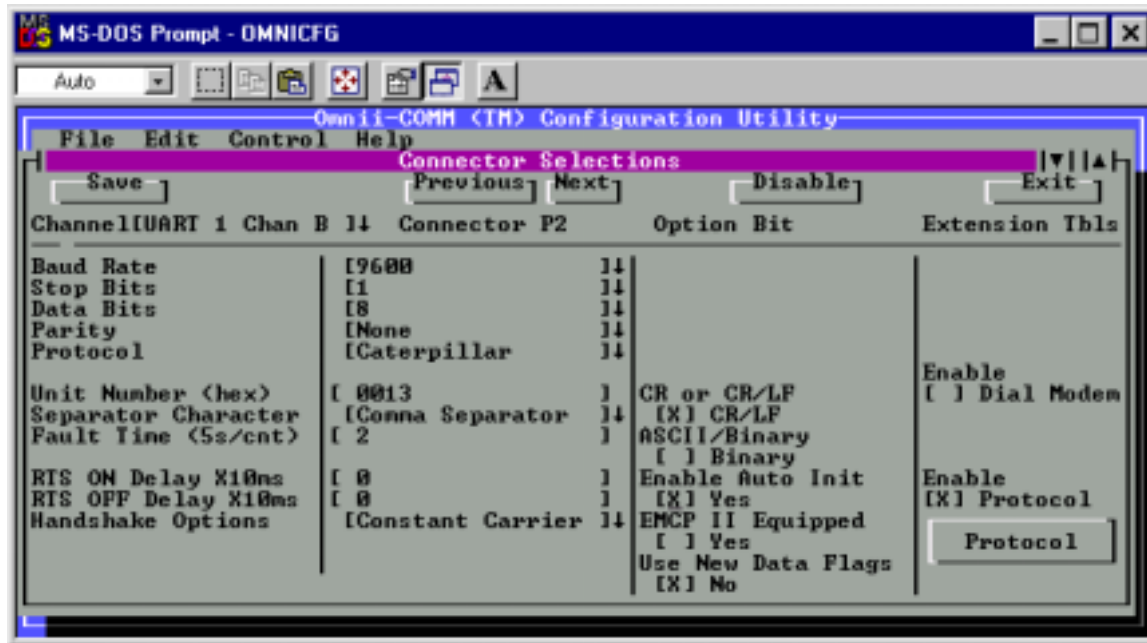
For this example, the Modbus connector looks like this:



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 and have selected UART 1 Chan A as the Channel

Caterpillar to Modbus Configuration Example

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

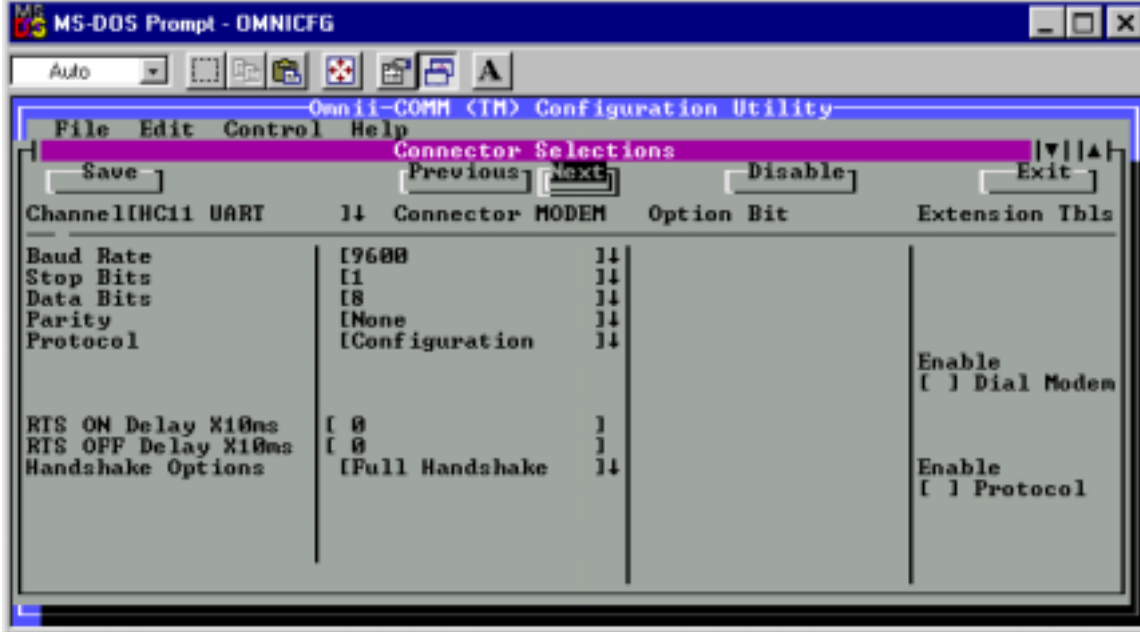


We are set up for the standard CCM communications parameters of 9600 baud, 8 data bits, 1 stop bit and no parity on UART 1 Channel B. The default Caterpillar Unit ID is 13, Auto Init feature is enabled and we want the data to be formatted in ASCII with a comma separator and a CR/LF at the end of each message. We are not going to use the New Data Flags. 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 to Modbus Configuration Example

Connectors P3 and P4 are Disabled.

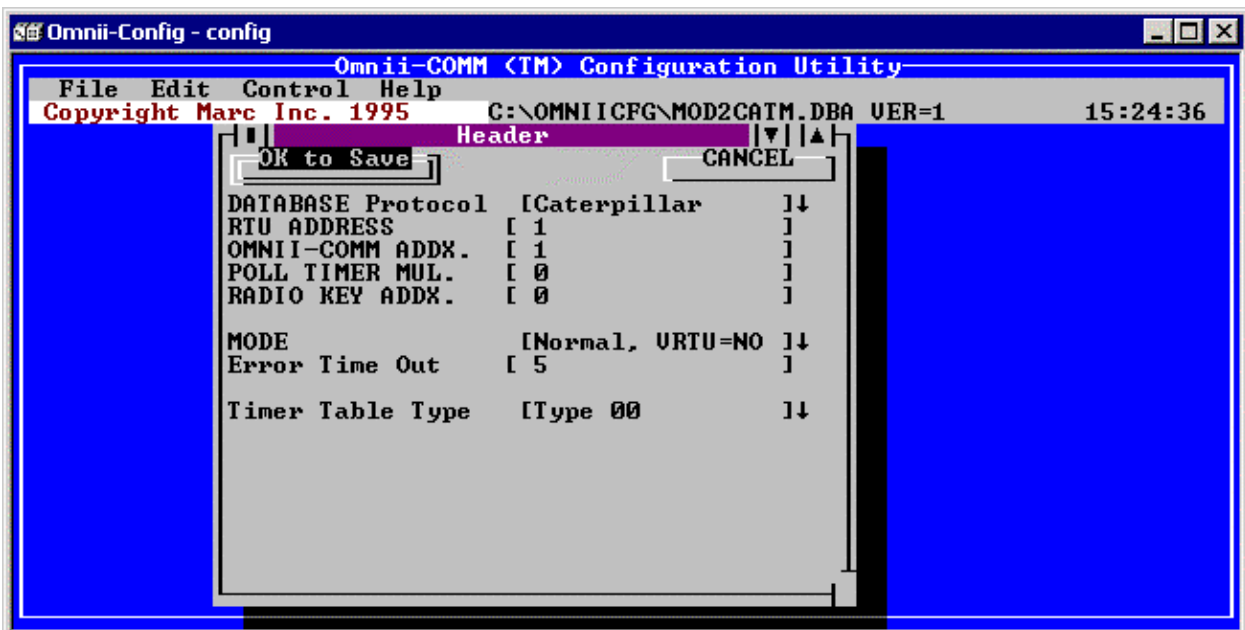
Connector Modem is used as a debug/diagnostics connector when the module is running. This provides a convenient "window" into the Omnii-Comm while it is in operation that can be used during troubleshooting. Always use HC11 UART as the Channel for the Configuration port if you have one.



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.



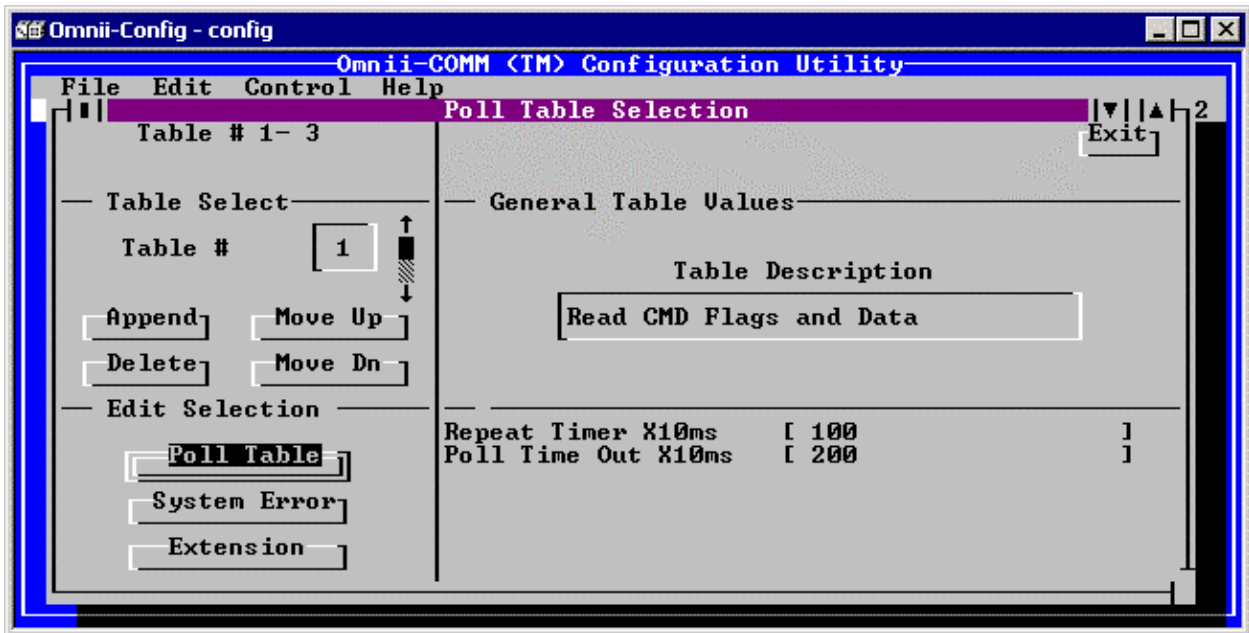
Caterpillar to Modbus Configuration Example

The Omnii-Comm database organization is selected as Caterpillar for convenience. The RTU Address, Omnii-Comm Address, Poll Timer Multiplier, and Radio Key Address are not used in this application. The Operating MODE is selected to be Normal 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 Type 00.

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 Omnii-Comm 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 3 tables (Table # 1-3). 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, Read Command Flags and 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; Read Command Flags and 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 reference the Omnii-Comm Database 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.

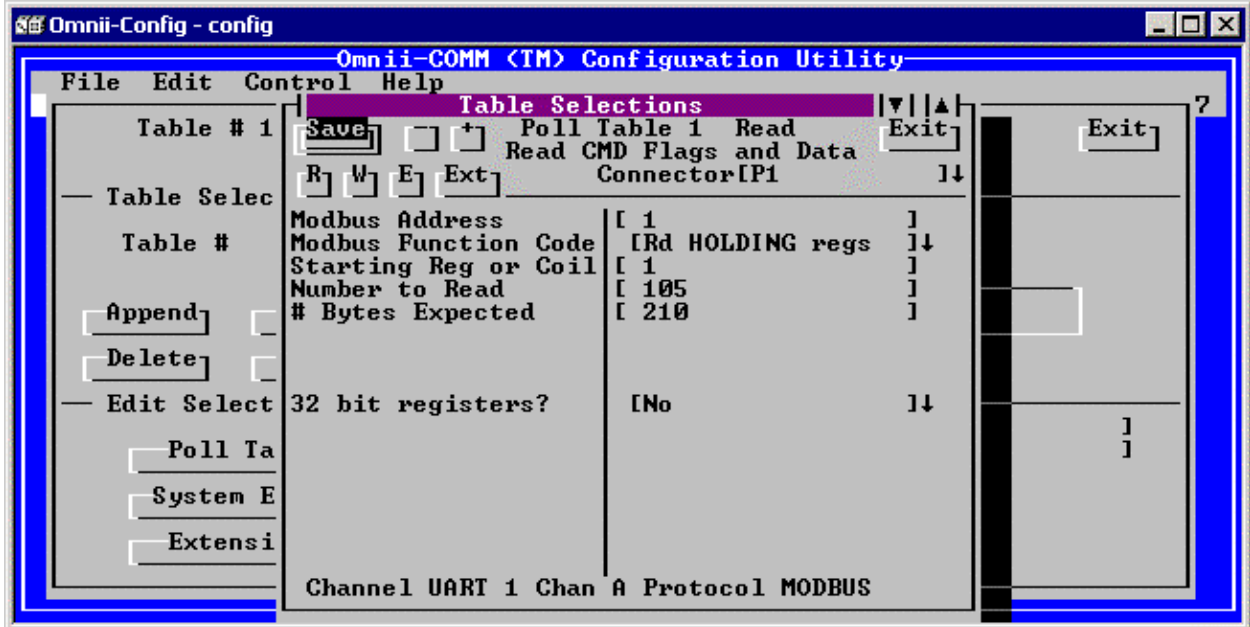
This application makes use of the Local RAM protocol as explained in the following section.

Now on to the specifics of this example:

Caterpillar to Modbus 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.

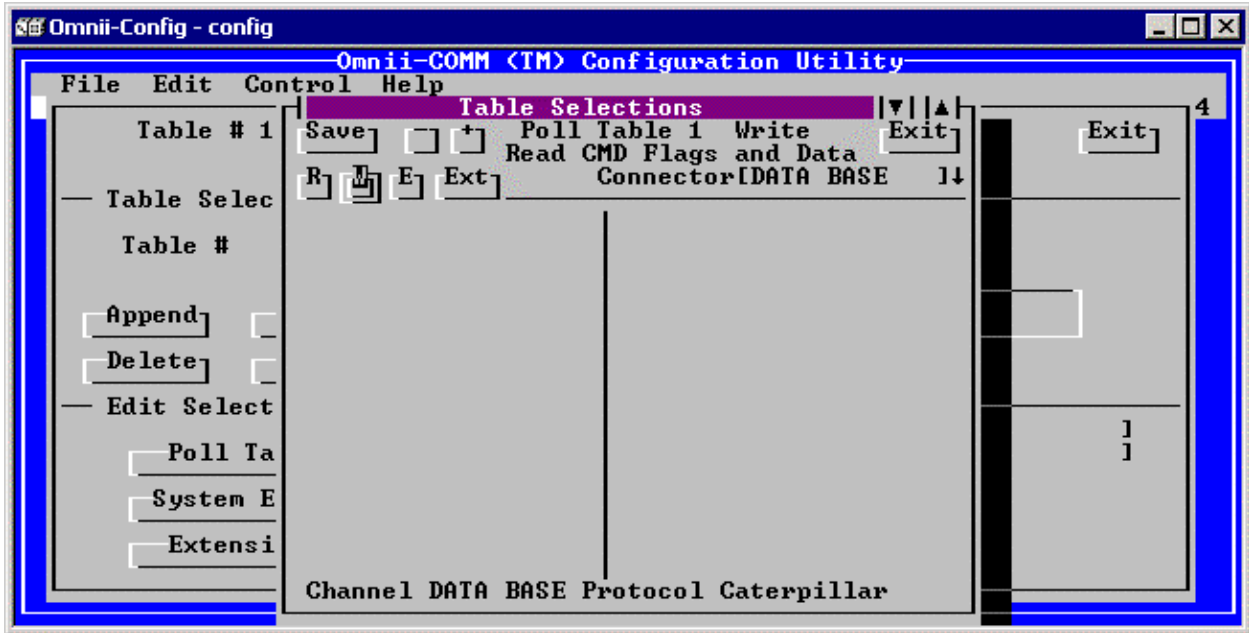


The connector selected is P1 which was configured for Modbus communication. The remainder of the items on this screen tell the Omnii-Comm how to go about reading some data from a Modbus slave device that is connected to P1. This table reads 105 Holding Registers from Modbus address 1 starting with the first register in the device. Holding Registers are Modbus 40000 series registers. The write section, explained next will define what we will do with this data.

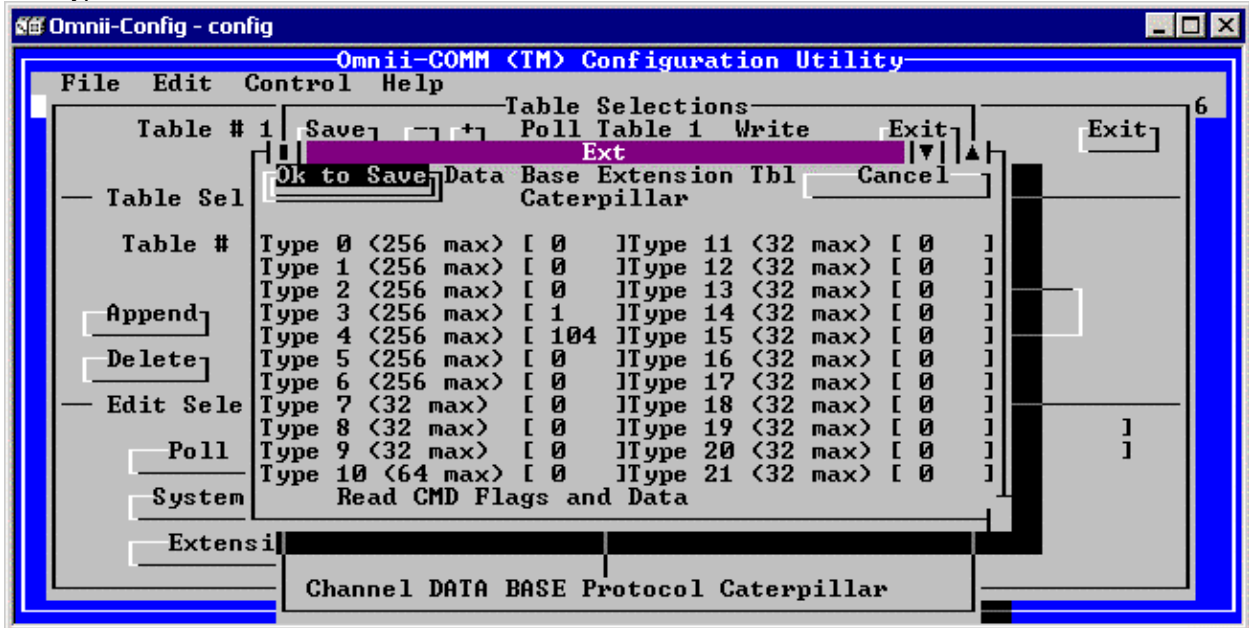
Caterpillar to Modbus Configuration Example

Poll Table #1, Write

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



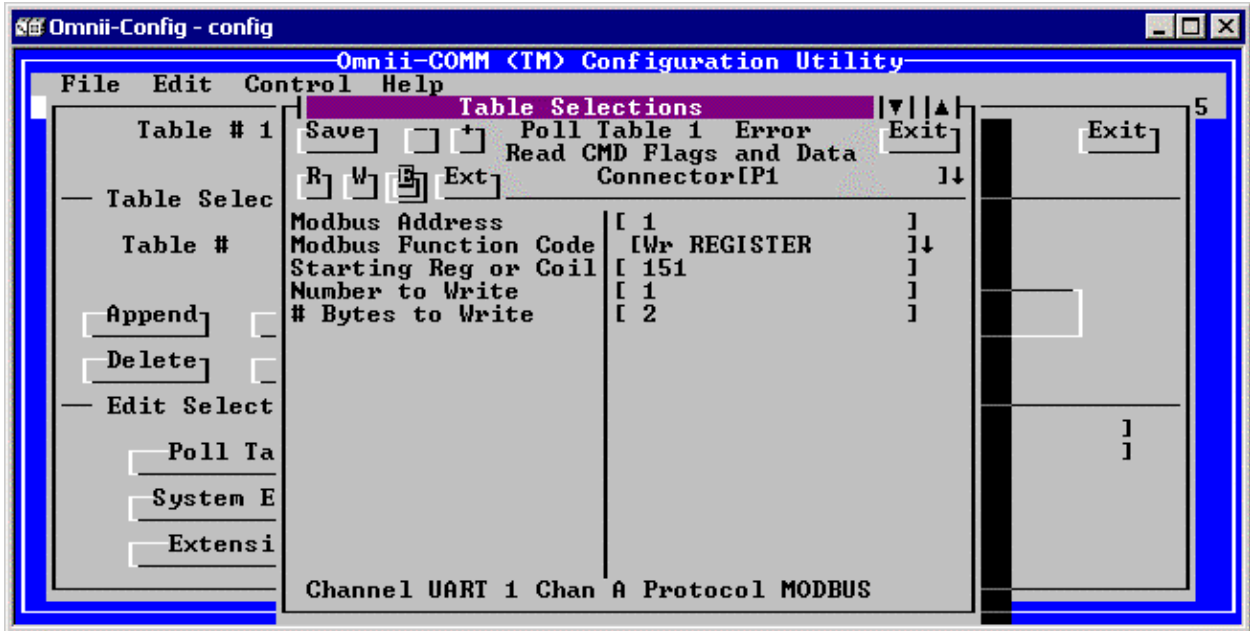
The Connector selected for the Write operation is Data Base so the important information will be contained in the Extension Table. **Click on the Ext button** to see the Poll Table Extension Table. This table is used to define how to distribute the data from the read operation into the database of the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



The Extension Table says that we are to create 1 Type 3 database word and 104 Type 4 database word from the 210 bytes we got from the read section. The contents of the 105 Holding Registers are placed into these registers. More about this later.

Poll Table #1, Error

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



The Error word associated with Poll Table 1 is Holding Register 151 in the Modbus Slave device. The Omnii-Comm will write an error to this Holding Register if an error occurs. All error writes are one word, 2 bytes long. You can of course change this register number as required for your application.

4.5.2 Poll Table #2; Define New Data Available Flags

Poll Table #2 is not strictly needed in our configuration. Table #2 is used to define a place to write “New Data Flags” should you want them. Writing New Data Flags is a user definable option selected on the Connector screen for Caterpillar protocol. The purpose of these Flags or bits is to notify the user that a new message has been received from the Caterpillar CCM module. When a list is broadcast from the CCM, the Omnii-Comm will accept it, put the data where ever it is configured to go and then, optionally, set a New Data Available Flag so the user can determine that the data has been refreshed. In this example we have disabled this option on the Connector screen for Caterpillar.

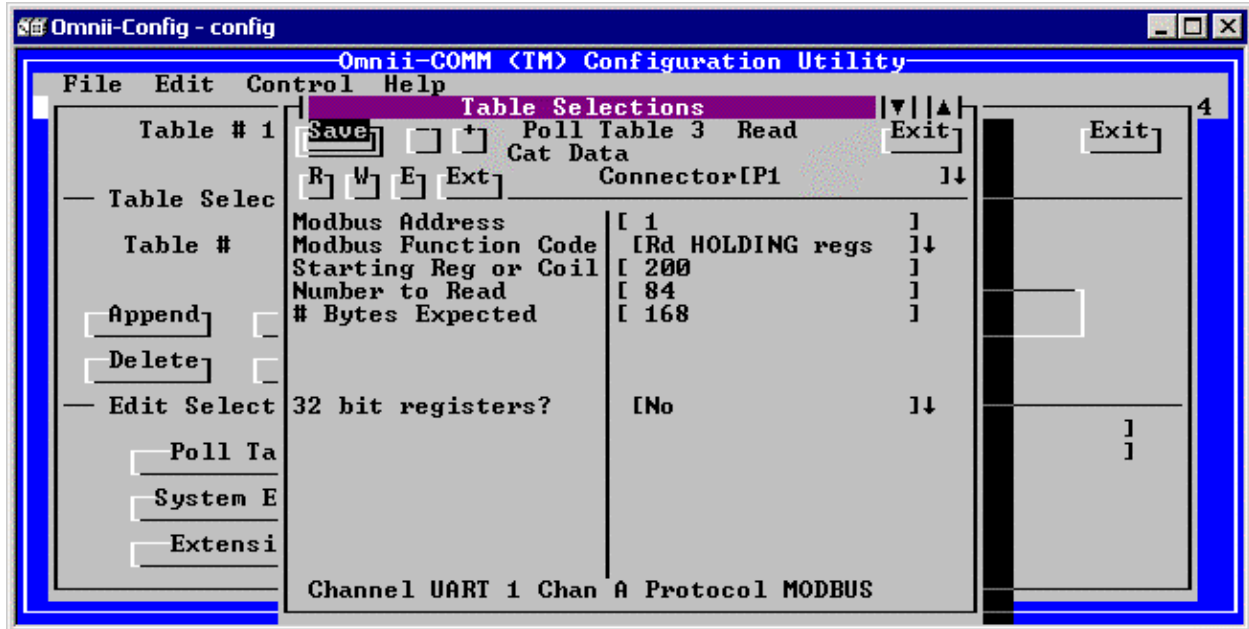
This table allocates Holding Register 106 for the New Data Available Flags.

This table is available so that, should you enable the option, there will be a place to write the New Data Available Flags.

NOTE: The Repeat Rate has been set to zero for this poll. This means that we will reserve the database Word but not waste time doing the actual data transfer.

4.5.3 Poll Table #3; CAT Data Storage

Polling table # 3 Reads 84 Holding Registers from the Modbus Slave starting at register 200 and writes them to database Data Type 00. We are going to use these 84 registers to store information coming in from the CCM.



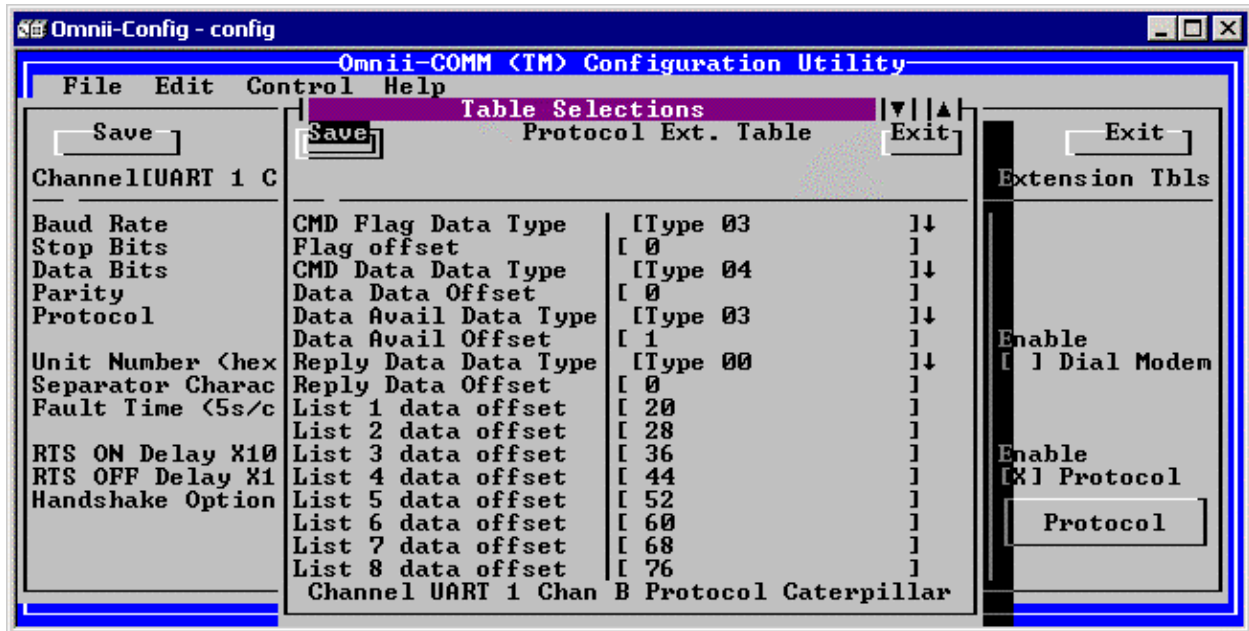
This seems backwards doesn't it? The poll table Reads from the Modbus slave and Writes to Database when we actually want to WRITE to registers in the Modbus slave. You might think that we should Read from the CCM and Write to Modbus registers. The CCM doesn't work like that. It is configured to automatically send (broadcast) blocks of information out its serial port a periodic rate. The block of information is called a "List" and the rate at which it is sent out is called the "List Time" in Caterpillar parlance. The CCM does not support polled operation so we cannot simply Read from Caterpillar CCM and Write to Modbus. To get around this little problem, we need to make use of an Omnii-Comm database property that is called "Write Thru". Changing the value of a database register AUTOMATICALLY changes the source of the information. The Write operation is automatic. It will occur whenever the database is written to. When the CCM broadcasts a list of data we will store it into database. The actual locations that we will write to are defined in Section 4.6 below. The Omnii-Comm will detect the Write operation, calculate the Poll Table # that defined the database registers and then determine where the source of the data is located. It looks at the Read section of the poll table to find out what connector, address, register type and register number was the source of the information and then automatically generates a WRITE operation back to the source.

Note also that the Poll Table Repeat timer has been set to 0 for this poll. This means that we will never actually waste time Reading these 84 registers from the PLC. We only will write to them when a list arrives from the CCM.

4.6 Caterpillar 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 where everything is tied together!

From the Main Edit screen select Edit then Connector. Use the Next button to advance to the Caterpillar connector and then click on the Protocol 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. The Offset value tells us where to start looking in the data base for the Flags.

Likewise, the next two fields define the Command Data 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 These registers are updated with new data every second by poll table #1 that reads from a block of Holding Registers in the Modbus Slave.

Omnii-Comm Data Types are defined in the Polling Tables and the Data Type Number is equal to the POSITION it occupies in the Extension Table. For convenience, we give the positions specific NAMES when we select a Data Base Organization on the Header screen but you can always refer to a data base type by number. In order to allow Caterpillar connections to any database type, we select the Types and Offsets by number rather than by name on this screen.

Caterpillar to Modbus Configuration Example

The next two entries, New Data Available Data Type and Offset define where the New Data Flags will be stored. In this case, Modbus address 1 Holding Register 106 because the Data Type is 3 and the offset is 1 and Polling Table 2 was used to create this database location.

The remainder of the Protocol Extension Table entries define where the Caterpillar data will be stored. Remember, the CCM will BROADCAST information, organized into LISTS after it has been configured. When the CCM sends out a Broadcast List, it includes the List Number in the message along with all the data defined for that list. The List Number is used to determine which entry in the Protocol Extension Table to use to find out where to start writing the data.

Other information can come back from the CCM in addition to list data. For instance, the CCM may send Fault Codes or the response to a command or the answer to a Read Single Parameter command.

The Reply Data Type field determines where these responses will go. In our example, we have selected Data Type 0. Writing to data type 0 generates an automatic write back to the source of the data, in this case Modbus Holding Registers between 200 and 284.

The Omnii-Comm always uses the first 20 words of the Reply Data Type in the following way:

Offset 0	Register 200	Response to Enable List, Disable List or Define List
Offset 1	Register 201	Response to a Single Parameter Read
Offset 2 - 19	Registers 202 thru 219	Fault Codes, 2 registers per fault, 9 maximum

The data broadcast in each list by the CCM is stored at the location defined by the offset for each list; all will be in the Reply Data Type address space. In this example, Cat Data for list 1 will be written to database data type 0 offset 20. This was filled from the Modbus slave device Holding Register 220 so list 1 data will appear in the PLC starting at Holding Register 220, list 2 data at 228 and so on.

To summarize:

Commands are sent by setting bits in Modbus Holding Register 1 in the Slave device. The first 8 bits are called Command Trigger Bits. When a bit changes from OFF to ON a command is sent. You should only set one bit at a time. Bits 1 thru 8 send one of 8 possible commands to the CCM. When the command has been sent, the Omnii-Comm will clear the Command Trigger bit and set the appropriate Command Complete Bit (Bits 9 through 16 of Holding Register 1). If an error occurs, the CCM error response will be written to the first register defined for the response data (in this case Holding Register 200) and the complete bit will not be set. You could use Modbus Coils as the Command Trigger bits by simply adding a Poll Table to read in the Coils and Writing them to Data Type 3. But, this can introduce a phasing error. Since the reading of the Modbus Slave device is asynchronous to any changes that the slave might make. It is possible that we could use old Command Data when the trigger bit is set. By reading both Command Flags and Command Data in the SAME poll table it is certain that the Command Data will be updated when the trigger bit is set as long as you change the command data first.

Command Data is set up in Modbus Holding Registers 1 through 104 in the Slave device.

NOTE: Change the Command Data Registers before setting the Command Trigger bit.

New Data Available Flags are sent to Modbus Holding Register 106 (if enabled).

CCM responses are directed to Modbus Holding Registers 200 through 283

4.7 Modbus Register/Bit Assignments

The following table summarizes the Modbus register numbers used for this example. Additional details are shown in the Excel spread sheet MB2CATM.XLS

Holding Register 1 Bit	Function
1	Activate List
2	Deactivate List
3	Program List
4	Request Single Parameter Read
5	Request Fault Codes
6	Special Parameter Command (ECMP II)
7	Single Parameter Write
8	Start Auto Init Sequence
9 – 16	Command Complete bits for the above commands

Holding Register	Function
2 – 33	Command Data for Commands 1 through 7
34 – 105	List Configuration data 9 registers/list.

Holding Register	Function
200	Command Response Data
201	Single parameter Read Data
203-219	Fault Codes
220-284	List Data from CCM/ 8 registers per list