

# Omni-Comm<sup>TM</sup> Example

CA CPASR V-SAT

To

Allen-Bradley



# TABLE OF CONTENTS

<b>SECTION 1.....</b>	<b>1</b>
<i>Overview</i> .....	1
1.1 WHAT IS THIS:.....	1
1.2 OTHER FILES YOU WILL NEED: .....	1
<b>SECTION 2.....</b>	<b>2</b>
<i>Concepts</i> .....	2
2.1 GENERAL CONCEPTS.....	2
2.2 V-SAT COMMUNICATION.....	2
2.3 PLC COMMUNICATION .....	2
2.4 DATABASE.....	2
<b>SECTION 3.....</b>	<b>3</b>
<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
<b>SECTION 4.....</b>	<b>5</b>
<i>Typical Configuration Procedure</i> .....	5
4.1 SELECT PROTOCOLS.....	5
4.2 LOAD CPATOAB.DBA .....	6
4.3 CONFIGURE CONNECTORS .....	6
<i>Connector P2 Allen-Bradley DF-1</i> .....	8
<i>Connector P3 CPASR V-SAT</i> .....	9
CPASR Extension Table.....	9
<i>Connector P4 Modbus Master</i> .....	10
<i>Connector P1 Configuration</i> .....	11
<i>Connector MODEM</i> .....	11
4.4 COMMON INFORMATION .....	12
4.5 POLL TABLES .....	13
4.5.1 Poll Table #1; Read CA Data From AB.....	14
Poll Table #1, Read.....	15
Poll Table #1, Write .....	16
Poll Table #1, Error.....	17
4.5.2 Read CA Data From Modicon .....	17
4.5.3 Poll Table #3; Set Virtual RTU Size.....	18
4.5.4 Poll Table #4, Define Initial FC 51 Interval.....	18
4.6 SUMMARY OF OPERATION .....	19
4.6.1 Virtual RTU Maximum Table Size File.....	20
Size Table Summary .....	20
Example Configuration Download.....	20
Digital Input Mapping.....	21
Analog Output Mapping .....	22

## Section 1

### Overview

#### *1.1 What is this:*

This document describes how to connect a Miille Applied Research Co. Omnii-Comm™ module between a Satellite Communication DIU and an Allen-Bradley PLC. In this example, the Omnii-Comm is configured to operate as a Teledyne/CA Remote Terminal Unit. Communication on the V-SAT DIU serial port uses the Teledyne/CA protocol with Unsolicited Report (Function Code 51) messages enabled. Virtual RTU support is enabled and Function Code 46 Configuration Download is supported to define the Virtual RTU configurations. Memory on board the Omnii-Comm, called Database, is organized as a standard CA RTU would be. This Database is filled with information read from other devices. In this example the Omnii-Comm reads data from an Allen-Bradley PLC 5 using DF-1 Full Duplex protocol. The PLC could be any Allen-Bradley model from a ControlLogix down to a MicroLogix since they all communicate using the DF-1 protocol. To highlight the versatility of the Omnii-Comm module, a third port on the unit is set to use Modbus protocol. Additional CA database registers are filled with data read from the Modbus device and reported to the CA Host. Control commands from the Host will change locations in the PLCs when they are received.

#### *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>

CPATOAB.PDF This file

CPATOAB.DBA Omnii-Config database file

CPATOAB.XLS Excel spreadsheet detailing register mapping for CPATOAB.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 uses DF-1 Full Duplex protocol to read and write registers in an Allen-Bradley PLC and Modbus protocol to read additional data from a Modicon PLC. Information collected from the PLCs is stored in internal Database memory of the Omnii-Comm. The Database is accessed by a CA Host using the standard CPASR communication function codes.

#### 2.2 V-SAT Communication

Communication to a Teledyne Vector Host is accomplished using the Teledyne/CA communications protocol. The Omnii-Comm supports most of the CA function codes (the list of Function Codes supported is supplied in Appendix A). Most important for V-SAT communication is the support of Function Code 46, Configuration Download and Function Code 51, Unsolicited Report.

When the Omnii-Comm is powered up it will periodically send "Request for Configuration" messages to the Host. A Request for Configuration message is a special case of the Function Code 51 message as described later. When the Host recognizes the Request for Configuration it will send a Configuration Download (Function Code 46) to the Omnii-Comm. The Omnii-Comm will reconfigure itself using information from the download and commence sending Function Code 51 messages with RTU data.

#### 2.3 PLC Communication

Information sent to the CA Host comes from Omnii-Comm memory called Database. The Database is filled by using serial communication ports of the Omnii-Comm to read data from other equipment **using the standard protocol of the device(s) we are connected to**. You could conceivably have as many of 4 different communication networks, each with their own protocol, used to fill the Database. The networks could also be multi-dropped so the actual number of external devices is almost unlimited. This example uses Allen-Bradley DF-1 protocol to read data from a PLC 5 and Modbus protocol to read data from a Modicon PLC.

#### 2.4 Database

The important concept here is that internal Omnii-Comm memory, called Database, is used to hold data collected from other devices. The Database is organized into data types that are standard for CA. It has areas for read only data types such as Digital Inputs, Analog Inputs, Meters, Calculated Integers and Tanks. There are also data types allocated for CA output types Digital Outputs and Analog Outputs. The CA Host does not know how the data was collected or the protocol used to collect it. It can only see the Database. Likewise, when the CA Host needs to do a control operation it can only see the Database. Controls sent to a Database point are automatically sent to the source of the data using the appropriate protocol.

## 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

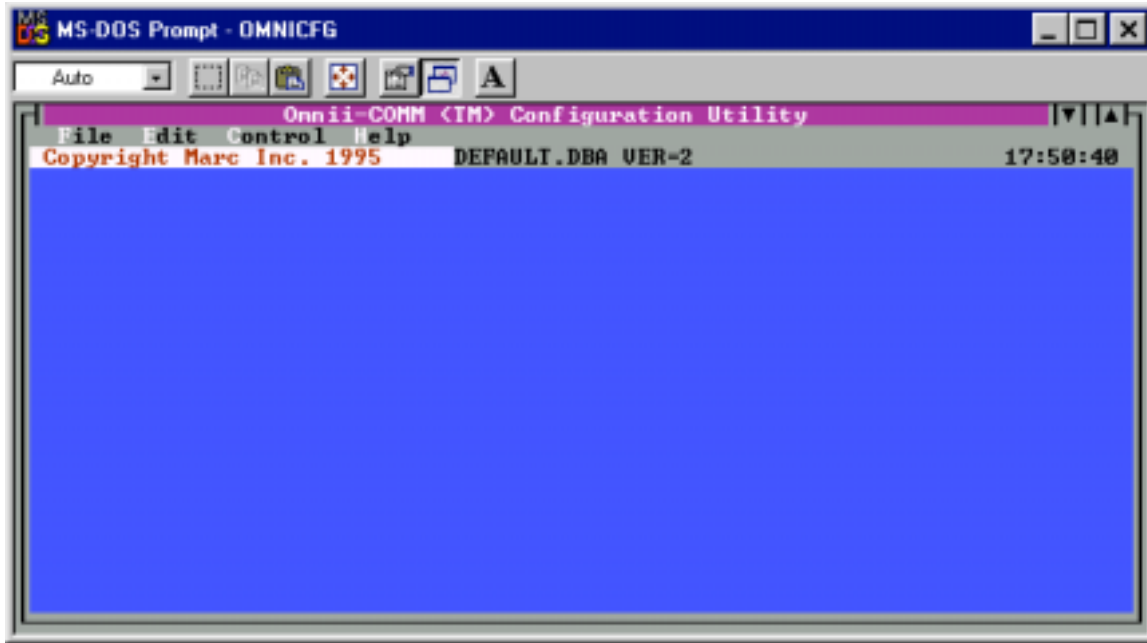
From the Program Manager window, select File, New, Program Item then OK  
Complete the Program Items Properties screen  
Description = Omnii-Config  
Command Line = Config  
Working Directory = name of the subdirectory where you saved the files

##### 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

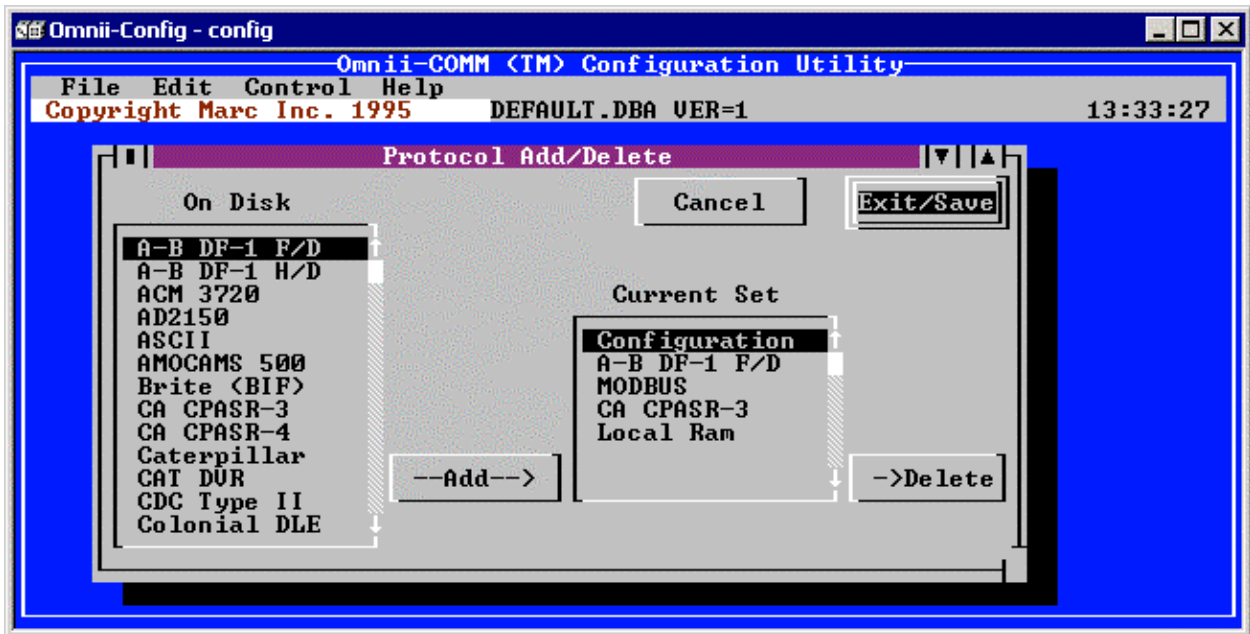
### Typical Configuration Procedure

#### 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.

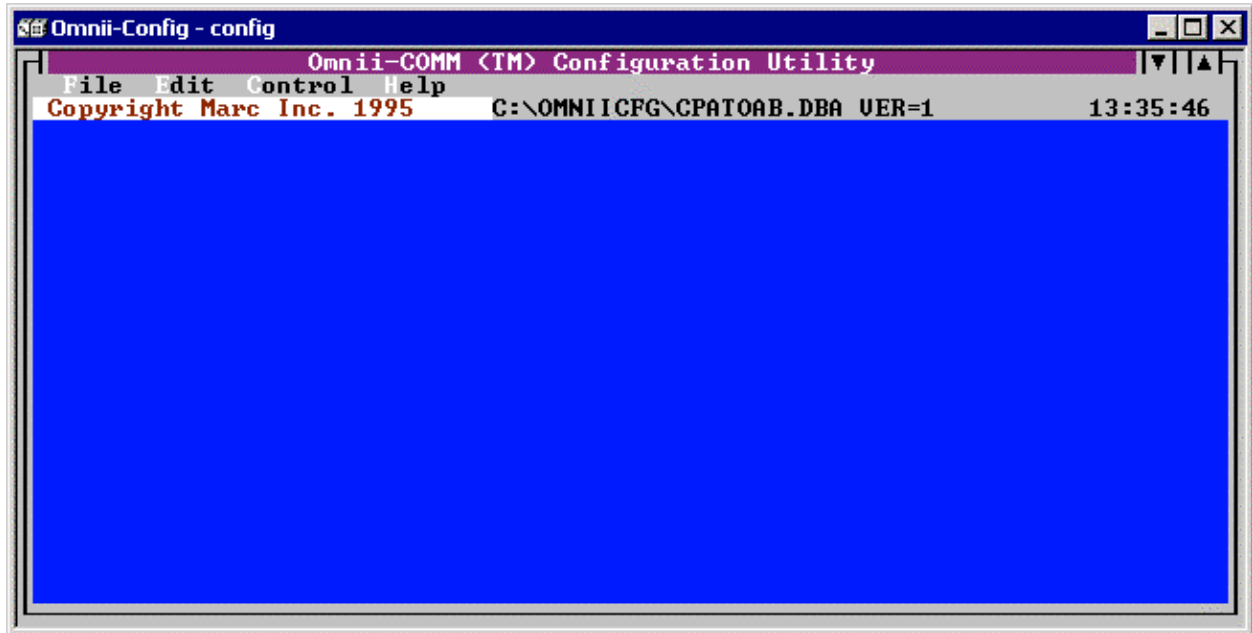
If you need to add a protocol to your current set, highlight it in the On Disk list and click on the Add button. To remove a protocol from the current set, highlight it in the Current Set list and click the Delete button.

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 CPATOAB.DBA example file.

## 4.2 Load CPATOAB.DBA

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

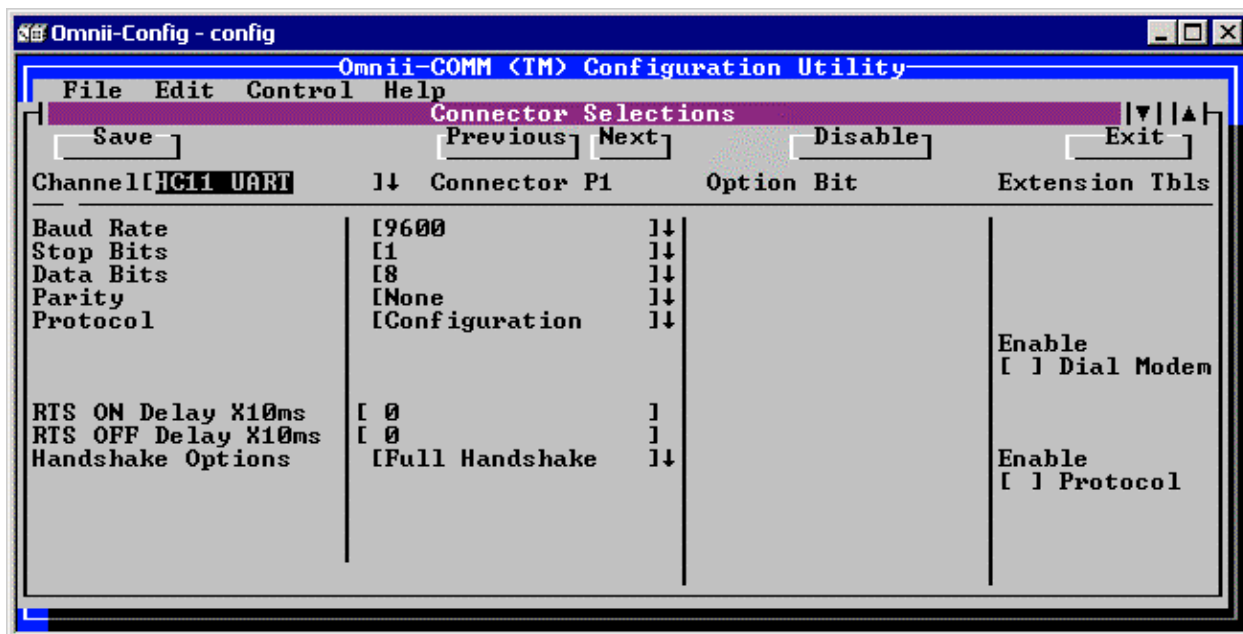


## 4.3 Configure Connectors

You can define the protocol to use on each Omnii-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 first Connector Selections screen will appear.



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 your configuration.

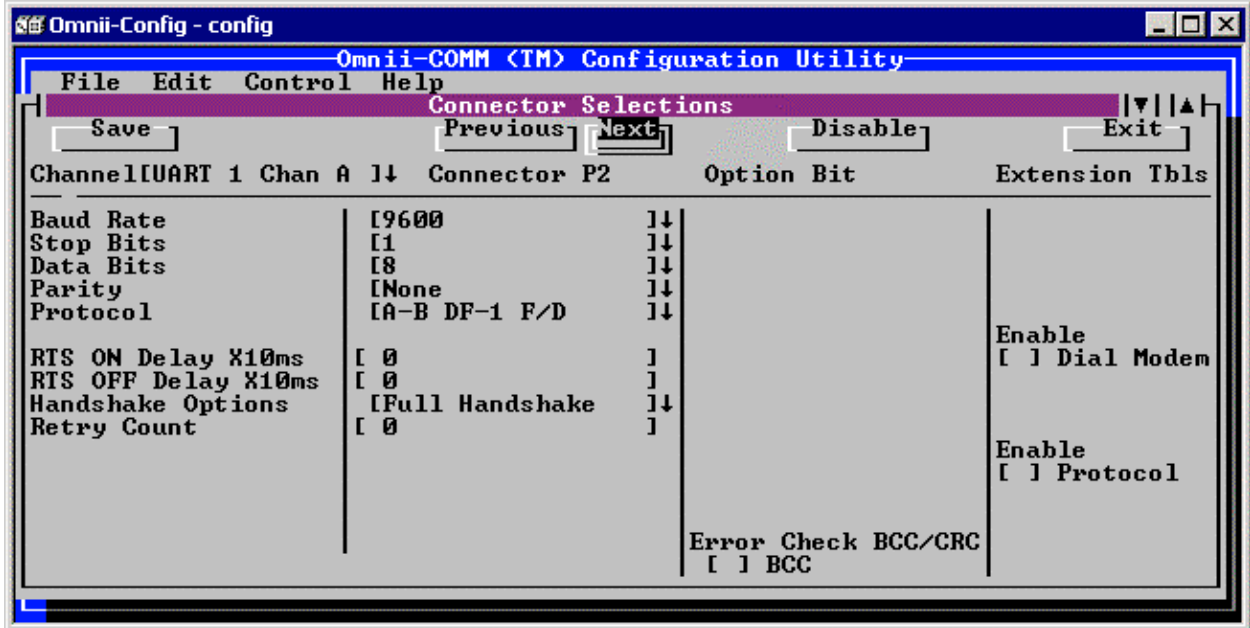
## CPASR to Allen-Bradley Configuration Example

This example configures the Omnii-Comm to use the P1 connector as the Run Time configuration connector, P2 as the DF-1 connector P3 as the CA V-SAT communications port and P4 as a Modbus communications port. This would be appropriate if you have a Slot mounted Omnii-Comm (166-500) since it uses the P1 connector as the Default Configuration connector.

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

### Connector P2 Allen-Bradley DF-1

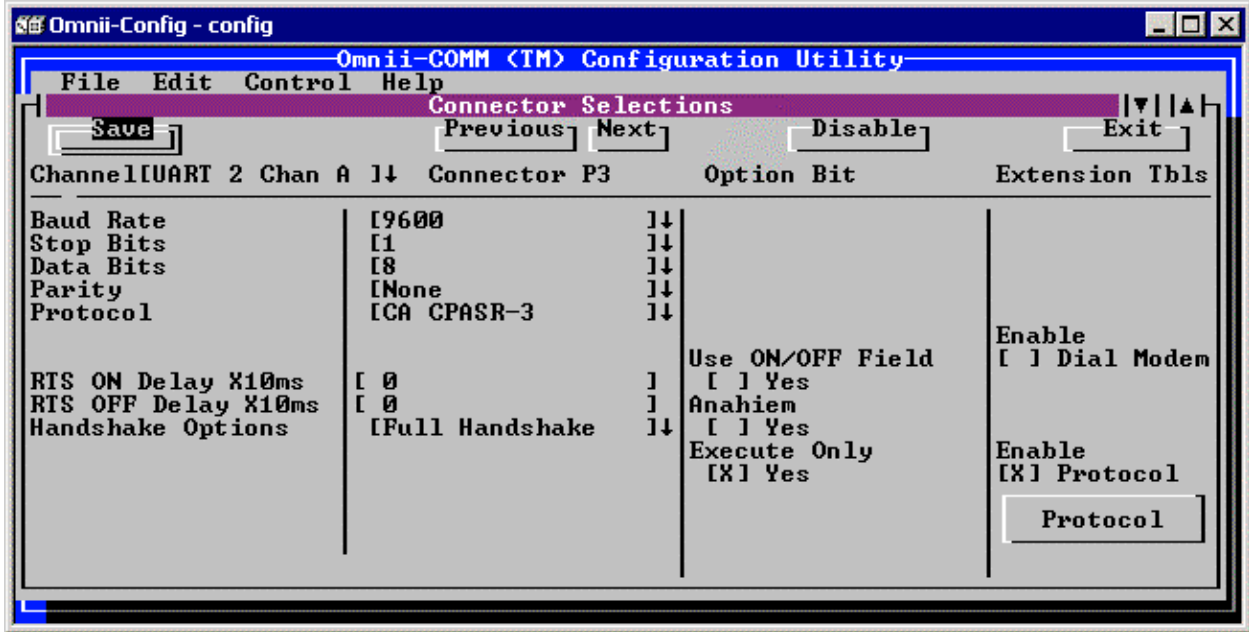
For this example, the Allen Bradley connector looks like this:



stop bit and no parity. We are set to use DF-1 Full Duplex protocol with CRC error checking. We The example configures the Allen Bradley connector (P2) for operation at 9600 baud, 8 data bits, 1 have selected UART 1 Chan A as the Channel.

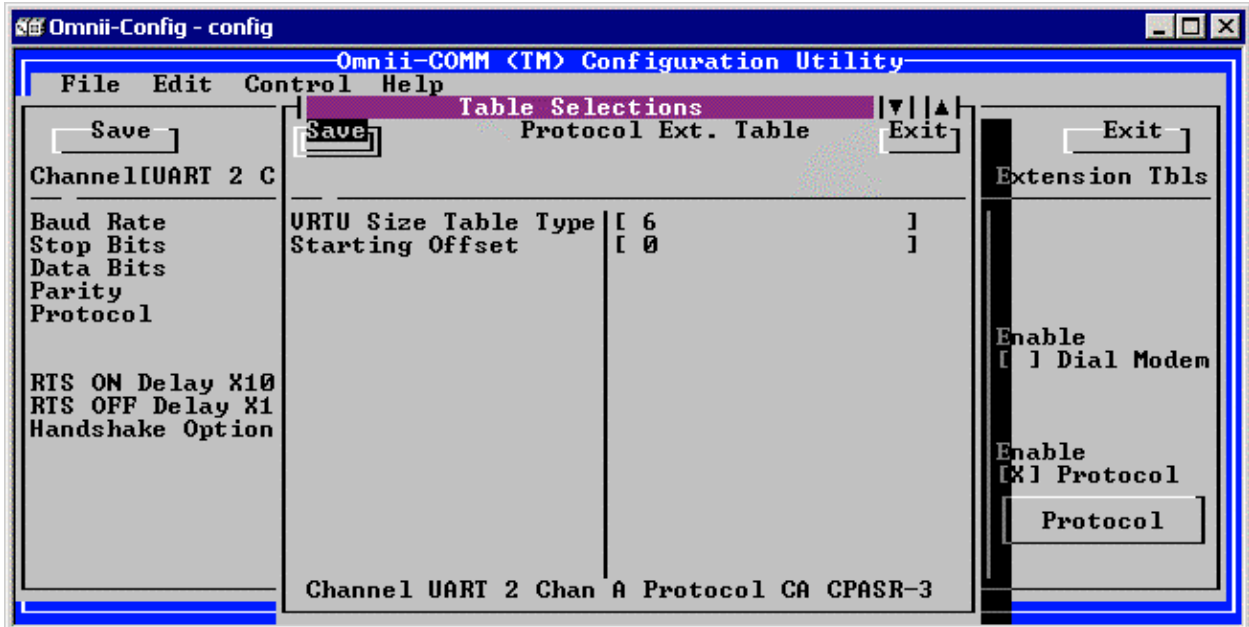
Connector P3 CPASR V-SAT

The CA V-SAT connector is P3. The configuration screen for this connector looks like this:



We are set up for 9600 baud, 8 data bits, 1 stop bit and no parity on UART 2 Channel A. The communications protocol is CA CPASR-3. The -3 in this protocol name refers to the size of the Accumulator data type. CA CPASR-3 has three byte accumulators and CA CPASR-4 has four byte accumulators. The Execute Only option is selected because the normal CA Select/Execute sequence is not used over the satellite. Note that the Protocol Extension table is enabled.

CPASR Extension Table



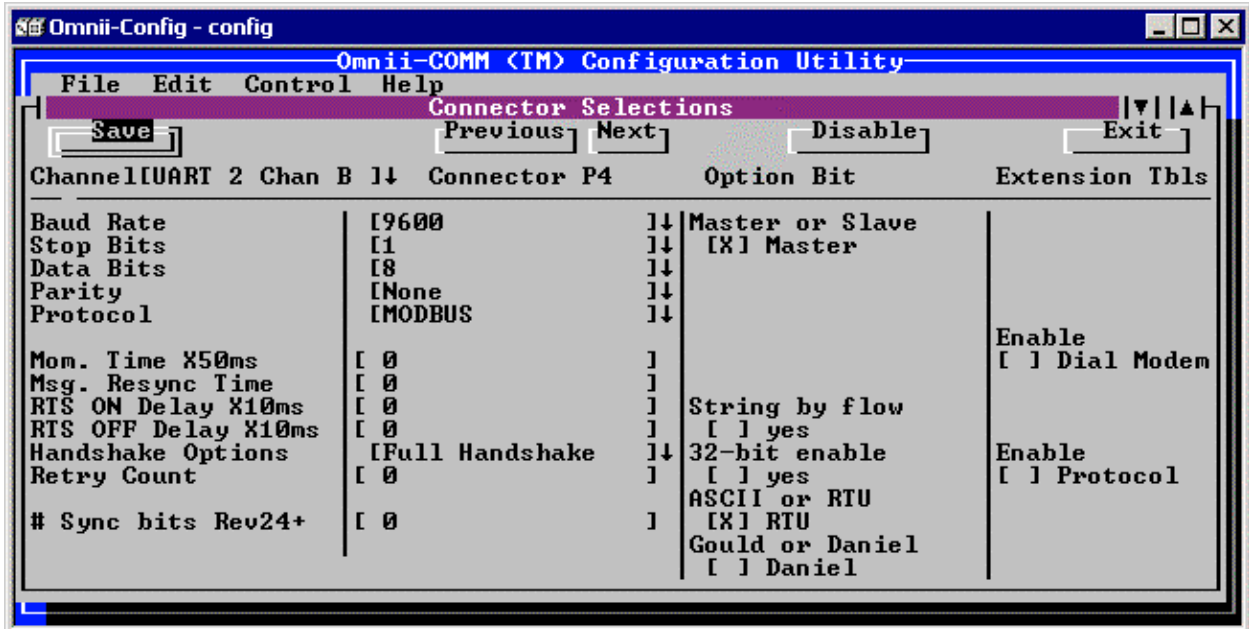
## CPASR to Allen-Bradley Configuration Example

Clicking on the Protocol button in the Extension Table column opens the Protocol Extension Table for CA CPASR protocol. The Extension Table is used to define the location of the Virtual RTU Default size table. The first entry is the Data Type in Omnii-Comm Database memory where the table is located and the second is used to define the Starting Offset in that data type. This example places the table into Data Type 6 at offset 0.

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 CA CPASR connections to any database type, we select the Types and Offsets by number rather than by name on this screen.

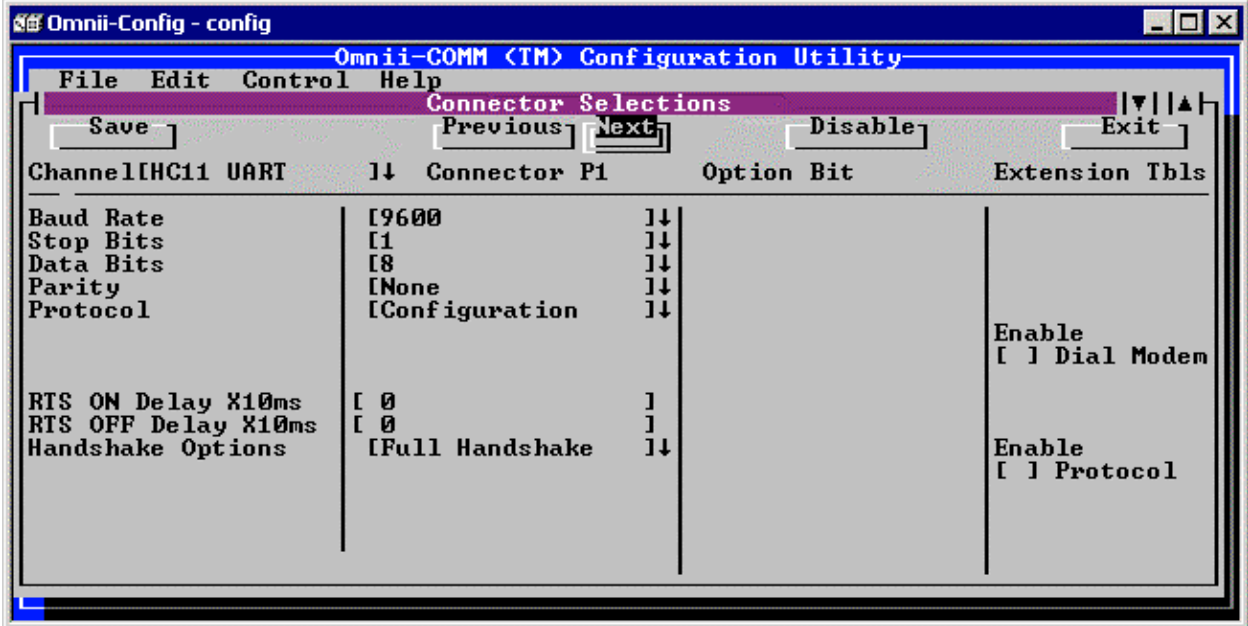
### Connector P4 Modbus Master

Connector P4 is used for Modbus. We are set up for 9600,8,N,1 operation using Modbus RTU protocol. We are a Master port because we will be reading from other Modbus slave devices.



## Connector P1 Configuration

Connector P1 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.



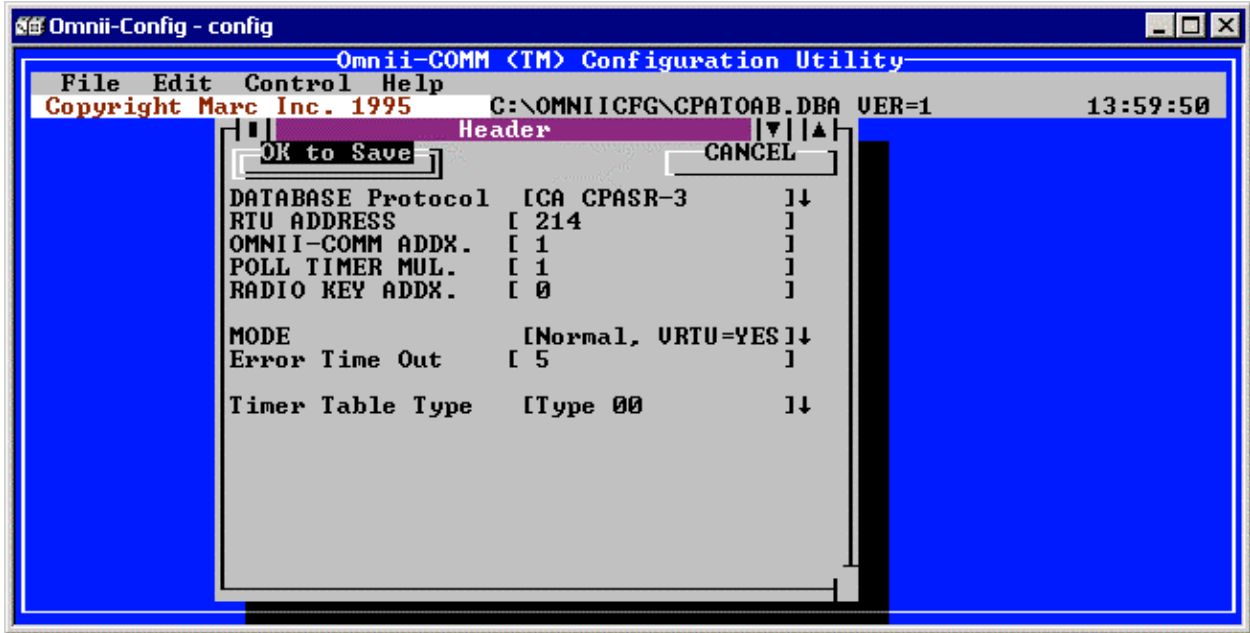
## Connector MODEM

This connector is not used in this example and is disabled. You could enable it with standard CA protocol and have a standard CA port that could be used to access the Database. If you enable the Dial Modem extension table and install an optional onboard modem, then this could be a Backup Dial Port that could be used to access the Omnii-Comm during times that the V-SAT link is down.

## 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.

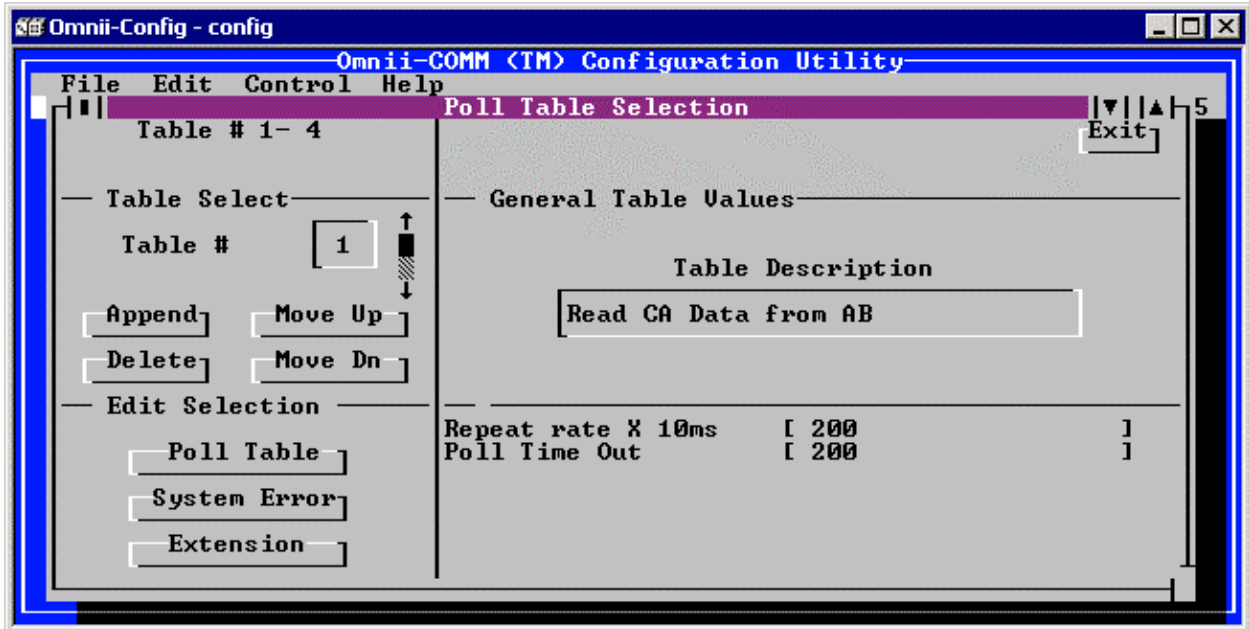


The Omnii-Comm Database organization is selected as CA CPASR with 3 byte accumulators (meters). Our CPASR Address is set to 214. The Omnii-Comm Address and Poll Timer Multiplier are set to 1. The Radio Key Address is not used in this application. The Operating MODE is selected to be Normal with Virtual RTU support enabled. 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 4 tables (Table # 1-4). 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 because that determines the order that the Database locations are filled. The order 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 CA Data from AB. 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 table’s Extension Table if there is one.

#### 4.5.1 Poll Table #1; Read CA Data From AB

**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.

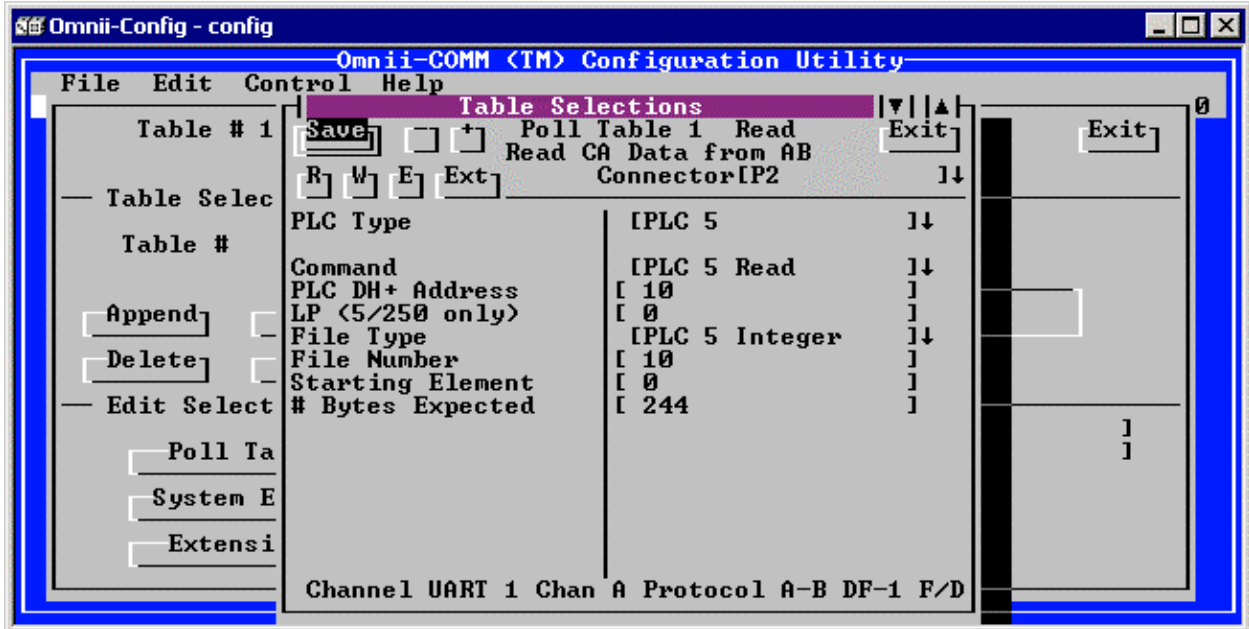
**NOTE: The Omnii-Comm has two "pseudo" connectors that can be enabled. They function like a real hardware connector but do not actually occupy any hardware resources. They only support one protocol each and the protocol cannot be changed. They are called Local RAM (VRAM) and Database. If you select 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 a following section.

Now on to the specifics of this 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 P2 which was configured for Allen-Bradley communication. The remainder of the items on this screen tell the Omnii-Comm how to go about reading some data from a PLC that is connected to P2 using the Allen-Bradley DF-1 Full Duplex protocol. This table reads 122 Integer Registers from a PLC 5, Integer File 10 starting with element 0. The Data Highway address of this PLC is 10. Note that the Data Highway Address entered on this screen is a DECIMAL number. Data Highway Addresses in an Allen-Bradley system are usually numbered in OCTAL. You will need to convert the Octal Data Highway Address into a decimal number before you enter it here. Decimal 10 is Data Highway address 12 octal. The write section, explained next will define what we will do with this data.

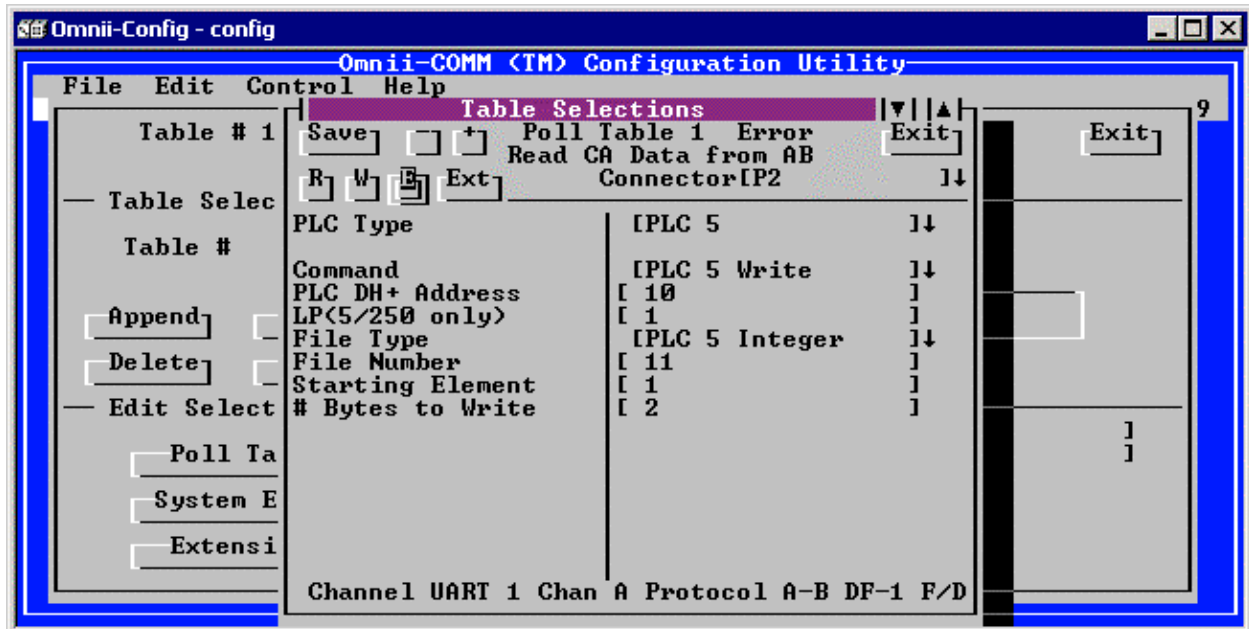


## CPASR to Allen-Bradley Configuration Example

The Extension Table says that we are to create 12 Digital Input Words, 24 Analog Inputs, 12 3-byte Meters, 12 Digital Output Cards, 12 Analog Outputs, 20 Calculated Integers and 12 Tanks from the 244 bytes we got from the read section. Note that Tank Data is 4 bytes per tank. This example shows that you can read multiple data types with a single read. It may be convenient to define multiple poll tables to fill the data base. Digital Inputs from one area in the PLC, Analog Inputs from another, etc. Reading them all in at one time is more efficient but does require that all the data be moved into file N10 by the ladder logic. This also makes it more difficult to expand the number of points should you need to do so in the future.

### 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 N11:1 in the PLC. The Omnii-Comm will write an error to this 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 Read CA Data From Modicon

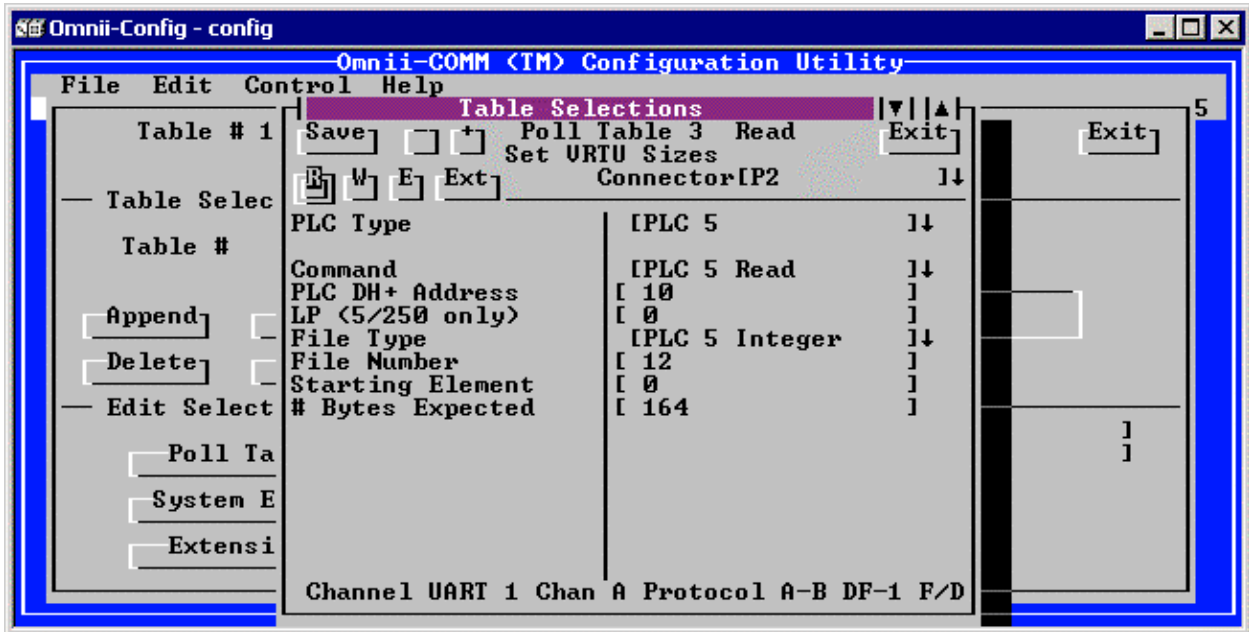
Poll Table #2 reads additional data from a Modicon PLC that is connected to Connector P4. This table reads 10 Modbus Holding Registers from address 10 starting at Holding Register 501. Modbus Holding Registers are Modicon 40000 series registers.

The Extension Table for Poll Table #2 instructs the Omnii-Comm to create 2 more CA Analog Outputs and 8 more Calculated Integers. The order of the poll tables define the order that the Database data types are filled. Since Poll Table #1 had Analog Outputs and Calculated Integer data types, they are placed into the Database first. The points from Poll Table #2 go in after the ones from Poll Table #1. We now have a total of 14 Analog Outputs and 28 Calculated Integers in this CA Database.

The Error section of this Poll Table is located at N11:2 in the PLC 5.

### 4.5.3 Poll Table #3; Set Virtual RTU Size

This configuration supports Virtual RTUs (this was enabled on the Header screen) so we need to define the default size for them. Polling table # 3 Reads 82 elements from the PLC 5 File N12 starting at element 0 and writes them to unused Database Data Type 06. We are going to use these 82 registers to define the Virtual RTU DEFAULT size for 5 VRTUs as explained later. We use the Default size information to determine the MAXIMUM amount of storage in our Database to allocate for each Virtual RTU. The Configuration Download message will usually reduce the maximum size of each data type as required for current operation. By setting up the table of MAXIMUM sizes, you can build in spare locations for each data type in each Virtual RTU. This makes later expansion much easier because adding a new point to one VRTU does not affect the Database addresses of other VRTUs.



This example reads the table from a file in the Allen-Bradley PLC but there are many other ways to get this information. For instance, it could be read from registers in the Modicon PLC or even out of Omnii-Comm memory if you don't have a PLC or don't want to use it to store the files. The table just needs to be read into the Database SOMEWHERE. The Read section of this Poll Table selects the source of the table and the Write section of this Poll Table defines where it will be placed in Database memory. We tell the Omnii-Comm where we put it in the Protocol Extension Table on the connector screen.

The VRTU size definition table format is straightforward. The first word in the table is a hex 5A5A, following the 5A5A is one or more 32 byte sections. Each section defines a Virtual RTU. Byte 0 is reserved for use in other applications and should be set to 0. Byte 1 is the Virtual RTU Address. Following the RTU Address are 22 bytes that define the number of Database elements to reserve for each of the 22 Data Types in an Omnii-Comm Database (CA protocol only uses the first 16); then 8 spare bytes that are reserved for future use. If there are more than one VRTU, then another 32 bytes must follow immediately. After the last VRTU has been defined the table will end with two bytes of 0.

### 4.5.4 Poll Table #4, Define Initial FC 51 Interval

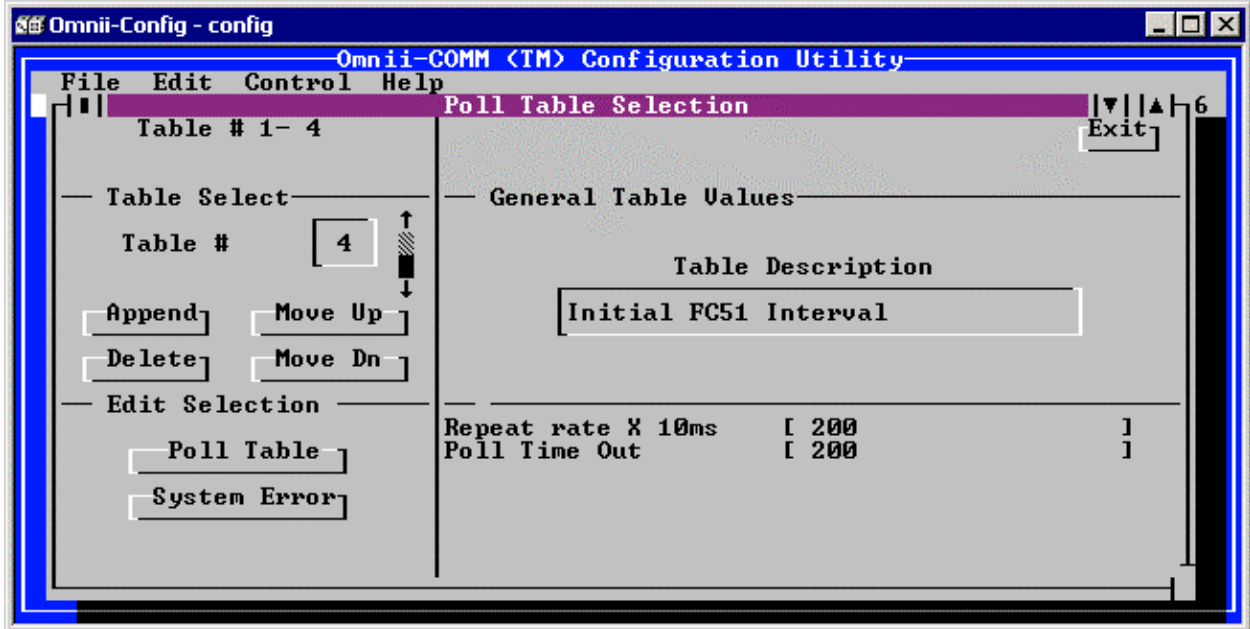
Polling Table #4 defines the rate that the Omnii-Comm will send a Request for Download message when it first powers up. Recall that we will need a Download from the Host to define our Virtual RTU configurations when we first power up. We send a special Function Code 51 message out to request this

## CPASR to Allen-Bradley Configuration Example

download. How often we send the request is defined with this table. When the Host sends a Download, the FC 51 rate will be whatever is defined in the download.

The initial rate will be the Poll Table Repeat rate as set on the Poll Table Selection screen. In this example, we will send it every 2 seconds.

Set the Connector for the Read and Write sections to the CA V-SAT connector. No other information is required. Set the Error section to a valid error location so that we can report errors during the initial request period.



### 4.6 Summary of Operation

OK, the connectors have been defined, we have set up the CPASR address in the Header and have defined the Database. Database locations have been filled with information collected from two PLCs, an Allen-Bradley and a Modicon. We have read in a table of default Virtual RTU sizes and have set up a table that will define the initial FC51 beacon rate. How does this all tie together. **Keep reading.**

The Maximum VRTU size information is read from the PLC (in this example) in Poll Table #3. Set the registers in the PLC 5 to look like this.

### 4.6.1 Virtual RTU Maximum Table Size File

Address	0	1	2	3	4	5	6	7	8	9
N12:0	5A5A	0026	0202	0202	0202	0000	0200	0000	0000	0000
N12:10	0000	0000	0000	0000	0000	0000	0000	0024	0202	0202
N12:20	0202	0000	0200	0000	0000	0000	0000	0000	0000	0000
N12:30	0000	0000	0000	0014	0202	0202	0202	0000	0200	0000
N12:40	0000	0000	0000	0000	0000	0000	0000	0000	0000	0025
N12:50	0202	0202	0202	0000	0200	0000	0000	0000	0000	0000
N12:60	0000	0000	0000	0000	0000	0001	0410	0404	040C	0000
N12:70	0400	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:80	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:90	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:100	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:110	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:120	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:130	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:140	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
N12:150	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

#### Size Table Summary

The following table summarizes the VRTU Maximum sizes defined by the data in N12.

Data Type	RTU 26	RTU 24	RTU 14	RTU 25	RTU 1
DI	2	2	2	2	4
AI	2	2	2	2	16
ACC	2	2	2	2	4
DO	2	2	2	2	4
AO	2	2	2	2	6
CI	2	2	2	11	20
Tanks	2	2	2	2	4

OK, now, just for example, suppose that the Function Code 46 Configuration Download defines the following VRTU configurations:

#### Example Configuration Download

Data Type	RTU 26	RTU 24	RTU 14	RTU 25	RTU 1
DI	1	1	1	1	2
AI	1	1	2	1	7
ACC	0	0	0	0	2
CI	0	0	0	10	15

Here is where a good old Excel spreadsheet can do you good; It is a great help when setting up any new configuration. The complete .XLS file for this example can be found in the CPATOAB.XLS file. Here, we will analyze a couple of sections copied from this file. First, Digital Inputs or status are analyzed. This is how the assignments go for the first Data Type, Digital Inputs. The default size fields for each RTU allocate 2 Digital Input cards for each VRTU except for VRTU #1 which has 4 reserved. All are read from the PLC N10 file. The Configuration Download actually defines only one of the cards for every VRTU except for VRTU #1 which gets 2 cards. The following table shows the source data elements for the Digital Inputs for each VRTU. The other input data types are mapped in a similar fashion as shown in the spreadsheet.

**Digital Input Mapping**

DATA MAP FOR CPATOAB.DBA EXAMPLE FILE								
				VIRTUAL RTU DATA POINTS				
Data Source		CA Data Type	Card #	RTU 26	RTU 24	RTU 14	RTU 25	RTU 1
N10	0	DI	0	DI 0				
N10	1	DI	1	SPARE				
N10	2	DI	2		DI0			
N10	3	DI	3		SPARE			
N10	4	DI	4			DI0		
N10	5	DI	5			SPARE		
N10	6	DI	6				DI0	
N10	7	DI	7				SPARE	
N10	8	DI	8					DI0
N10	9	DI	9					DI1
N10	10	DI	10					SPARE
N10	11	DI	11					SPARE

Digital Outputs are mapped in a similar fashion but there are some important differences. Normally, the Digital and Analog Output Default sizes are not altered in the Configuration Download message so the defaults become the actual sizes. This is normally not a problem because the Host should already know which points are valid for each RTU and will not try to write to any that are not defined. You can, however, alter the size of the DO and AO data types by inserting the changes in the Read All Scan definition of the Configuration Download message if you need to. You do this by setting the appropriate selection bit in the read all message to enable DO and/or AO data types. If this bit(s) are set, then we will look at the size field for that select bit and alter the maximum number of control points based on that entry divided by 2 (the size field is in bytes). The following is a copy of the Analog Output section mapping from the CPATOAB.XLS spreadsheet.

### Analog Output Mapping

N10	66	AO	0	AO0				
N10	67	AO	1	AO1				
N10	68	AO	2		AO0			
N10	69	AO	3		AO1			
N10	70	AO	4			AO0		
N10	71	AO	5			AO1		
N10	72	AO	6				AO0	
N10	73	AO	7				AO1	
N10	74	AO	8					AO0
N10	75	AO	9					AO1
N10	76	AO	10					AO2
N10	77	AO	11					AO3
Modbus #10, Holding Register	501	AO	12					AO4
Modbus #10, Holding Register	502	AO	13					AO5

The outputs are defined in the polling tables just like the inputs. In this example, N10:66 thru 77 and Modbus #10, Holding Registers 501 and 502 are assigned to Omnii-Comm Data Type 04, Analog Outputs as shown in the table above. The first 12 are read in Poll Table #1 and the next 2 in Poll Table #2. These Poll Tables Read from the PLC and Write to the Database.

This seems backwards doesn't it? The poll table Reads from the PLC and Writes to Database when we actually want to WRITE to registers in the PLC. You might think that we should Read from the Database and Write to registers in the PLC. Here we are using a very important Omnii-Comm Database property that is called "Write Thru". Changing the value of a Database register AUTOMATICALLY changes the source of the information. A Write to Database Writes Thru to the source of the data. The Write operation is automatic. It will occur whenever the Database is written to. When the CA Host does a control operation, it is addressed to the Database. The Omnii-Comm detects this Write operation, then computes the source of the data and forms a Write operation back to the source of the data using the appropriate communications port and protocol that was used to collect the data in the first place. At the completion of the current poll (if there is one running), the command will be sent to the source.

**Controls operate almost immediately, there is no delay. Polling tables are not used.**

Digital outputs operate in a similar fashion. This example does not show any Digital Outputs going back to the Modbus device but you could easily add them. You could define Database Digital Output cards by reading either Modbus Holding Registers or Coils. If you define the Digital Outputs by reading Modbus Holding Registers, then a command from the Host that falls into the address space for the Modbus will be performed by doing a "Read-Modify-Write" operation on the Holding Register that contains the addressed bit. If, on the other hand, you defined the Database location by reading Modbus Coils, then the command to the Modbus will be a Write Coil operation.

**Note: Allen-Bradley single bit writes are always performed using the "read-modify-write" operation because most Allen-Bradley PLCs do not support single bit writes.**