January 2006

Midac Interface

1.0 General

The CPP has a driver for interfacing to a Midac FTIR instrument. The interface protocol is Modbus RTU, with the CPP being the master and the Midac being the RTU slave. The CPP polls the Midac instrument and stores the responses. No responses and incorrect responses to polls are logged in an error file. Entering an **E** commands the CPP to print the error file. In normal operation errors should be infrequent. AN **EC** clears the error log.

2.0 Initializing Midac into a standard CPP channel

The Midac is connected to a CPP channel using the standard channel configuration. From the Main Menu select Channel Setup.

Channel # = **01<cr>**
Name = **SO2<cr>**
Units = **ppm<cr>**
Full Scale = **1000<cr>**
Zero = **0<cr>**

Instrument Manufacturer
00 = Other
01 = API
02 = M&C
03 = Midac
03 = R&P
04 = Sick
05 = Siemens
06 = Thermo
07 = Temet

Selection = **03<cr>**
Midac Param ##
01 = Register #01
02 = Register #03
03 = Register #05
04 = Register #07
05 = Register #09
06 = Register #11
07 = Register #13
08 = Register #15
09 = Register #17
10 = Register #19
11 = Register #21
12 = Register #23

13 = Register #25
14 = Register #27
15 = Register #29
16 = Register #31
17 = Register #33

Selection = **01<cr>**

Half Dpx(H), Full Dpx(F) = F **<cr>**
Comm Port (3-00 Thru 3-08) = 3-00 3- **02<cr>**

Baud Rate
1 = 300, 2 = 1200, 3 = 2400, 4 = 4800, 5 = 9600, 6 = 19.2K, 7 = 38.4K = 5 **<cr>**

Instrument ID = 0000 **1001<cr>**
IMD Installed = 00 **<cr>**

Points For Interim Avg = 01 **04<cr>**
Points For Final Avg = 01 **9<cr>**

Unip(U)/Bip(B) = U **<cr>**
Unip(U)/Bip(B) Cals = B **<cr>**

Channel # = **<cr>**

Return to Main Menu.

The above initializes the parameter being stored in the Midac register #01 into CPP channel #01. The CPP will poll the Midac unit at a predetermined rate and store the data in the CPP database. This data can then be used by other features offered by the CPP such as converting the readings to isolated 4-20mA outputs. In the example above the address entered as 1001 will request data from the Midac as a Modbus RTU slave unit with an address of 01. The addressing scheme is described below.

The CPP assumes that the serial protocol is 8 data bits, one stop bit, and even parity. The Baud rate can be selected in the CPP set up.

3.0 Addressing Scheme

Modbus addresses can be in the range of 0-255. The CPP uses the last three digits of the input address as the RTU address. The first digit is used to select the method of converting the 32 bit floating point number received from the Midac into a real number. Modbus registers are 16 bits in length, so a 32-bit floating point number requires two registers. A 32-bit number is represented as shown below.

<Sign, 1 bit> <Exponent, 8 bits> <Field, 23 bits> = 32 bits

The Modbus register hold the 32 bits in two 16 bit registers as follows.

Reg #1 <Sign, 1 bit> <Exponent, 8 bits> <Field bits, 7 bits>	= 16 bits
Reg #2 <Field bits, 16 bits>	= 16 bits

Depending on the set up, either one of these registers can be sent first followed by the other bit. The floating point routine in the CPP expects the registers to be sent as Reg #1, followed by Reg #2. If this is the case then an address for the Midac RTU is entered with the first digit of the four-digit address as a zero, e.g., 0001. If the transmission of the two registers are reversed, (Reg #2 sent first, followed by Reg #1), then the first digit of the four-digit address should be entered as a one, e.g., 1001. This instructs the CPP poller to reverse the order of the received registers before sending to the CPP floating point routine. This is the case with the Midac unit. The first address bit should be set to a one.

In some instruments, the order of the bits in each register can be reversed. In other words, the sign bit could oriented to the right of the register, instead of being left oriented as shown above. In such cases the first address bit should be set to two, e.g., 2001. This instructs the CPP to rotate the bits prior to sending to the floating point routine. These two schemes can be combined if needed by entering the first address bit as a three, e.g., 3001. In this case the CPP would exchange the registers, then rotate the bits prior to sending to the floating point routine.

4.0 Cable Connections

The Midac has a standard PC, DB9 female connector for the Modbus connection. The mating plug is a DB9 male connector.

5.0 Saving Configurations

After the CPP has been set up as desired, the configuration should be saved in the on board EEPROM. This is done as follows over comm port #4.

CFGUL<cr>
This will overwrite any configuration stored in the Compact Flash
Continue Y/N **Y<cr>**

Entering a Y will result in the CPP saving the set up to the compact flash. If a compact flash is not installed, the CPP responds a follows.

No Compact Flash installed, OK to write to EEPROM Y/N **Y<cr>**

This can take several minutes.

At the first question, CFGUL, if an E is entered instead of a Y, the CPP will write to the on board EEPROM, even if a compact flash card is installed.

A list of data being stored in either the compact flash or the EEPROM will scroll across the screen. A prompt is not printed at the end of the configuration save.

At any power up the CPP checks the configuration RAM for parity errors. If any are detected, the CPP clears the associated RAM error, and if a configuration has been stored in the CPP, automatically reloads the configuration.

The stored configuration can be downloaded into the CPP with a CFGDL command and a Yes (Y) answer to the question.

6.0 Initial Set Up Helpers

Initialize the Midac into com. port 3-0. Over comm port #4, enter an M keyin. The CPP will print a list of mnemonics and memory address locations. Two that can be of interest are the transmit register(TX3BP) and the receive registerRX3BP) for comm port #3, which is the comm port indicated above to be used. above.used. Assmue that RX3BP = 8E17 is printed. Entering M 8E17 results in the CPP printing two lines of hexadecimal data starting at memory address $8E17_H$. The first two bytes are the receive buffer pointer, and the bytes that follow are the bytes that were last received. In a similar fashion, if TX3BP = 8DA6, then the string transmitted by the CPP out of comm port #3-0 can be viewed.

The Error Log can also be viewed to provide information on the CPP poll/Midac response.