RMV400NT Programmers Guide For Visual Basic

By John Dreese (www.dreesecode.com)

The RMV400NT is a powerful stepper motor controller board that also has Digital Input/Output as well as analog inputs. This guide assumes that you have a single 400NT board connected with the computer by way of a serial port, allowing control of up to 4 motors.

All of the special functions used to control, read, and write to the 400NT are contained in a special Windows DLL file called st400nt.dll. Make sure that this file is either in your Windows/System directory or inside your application directory. Aside from having this file, the most important necessity is the proper set of Function Declarations inside your program. These are statements that setup all the functions you will need when operating your 400NT board. They can be seen in the Module1.bas file that is included in the RMVdemo. There are a TON of them, so just either import that module into your current project or copy and paste them. If you don't have a Module1 in your program already, select Project/Add-Module from within Visual Basic.

The Special Function Call Format

The special function calls that allow interaction with the 400NT board require a special format. They work by making an arbitrary variable equal to the function call itself. I like to use one called "result". For example, to connect the 400NT board with the second serial port at a baud rate or 9600, I would use:

```
result = PortOpen(2,9600)
```

If your request was successful, then `result` will have a value of 1. If it was a failure, it will have a value of –1. This is an easy way to check if your function call worked correctly. For example, the following code will tell you if the PortOpen function worked:

```
result = PortOpen(2,9600)
If result = -1 Then MsgBox "Error with serial port!"
```

Please note that this format is the format used by all function calls for the RMV400NT controller board, even the motor settings and motor actions.

Chapter 1: Starting Up

Every time you run your program, you will need to use the following commands in your Form_Load event:

```
Dim AddressList As String
result = PortOpen(2,9600)
AddressList = "0,1,2,3,N,N,N,N,N,N,N,N,N,N,N"
result = GetConnectedControllers(AddressList)
result = MotorStationsInit()
```

Let me explain these in detail:

PortOpen

The PortOpen command is used to connect your computer to the RMV400NT controller board via serial port. I recommend using either the first or second serial ports, often referred to as COM1 and COM2. These are represented by the integer of their port number, 1 and 2 for example. Common settings for the baud rate are 9600,19200, 38400, 57600, 115200. However, I recommend using 9600 since most Windows serial ports are configured for that Baud rate by default. The above example of the PortOpen function call shows a sample of how it would be connected to the second serial port at a baud rate of 9600.

If you are tempted to let the port and baud settings be variables, make certain that you define the *Port and BaudRate as Long* because using regular Integers will generate errors. For example,

```
Dim RMVSerialPortNumber As Long
Dim RMVSerialPortBaud as Long
RMVSerialPortNumber = 2
RMVSerialPortBaud = 9600
result = PortOpen(RMVSerialPortNumber, RMVSerialPortBaud)
If result = -1 Then MsgBox "Error experienced!"
```

Note: When you run into errors where everything seems to be right numerically, check to make sure you defined your variables correctly. The 400NT function calls are very picky about this subject.

GetConnectedControllers

This function call connects you with the four stepper motor controllers on board the 400NT. One VERY important thing to remember is that when dealing with the motors, they are represented by a String variable. So to connect to all four motor controllers, you need the following type of command:

```
Dim AddressList as String
AddressList = “0,1,2,3,N,N,N,N,N,N,N,N,N,N,N,N”
result = GetConnectedControllers(AddressList)
If result = -1 Then MsgBox “Error experienced!”
```

Note that the motors are numbered from 0 to 3 instead of from 1 to 4. The 400NT can be networked with three other boards, each controlling four of their own motors for a grand total of 16 motors on the system. Since this guide assumes that you do not have networked cards, we assume no Motor Controllers on them and represent it with an “N”.

I have tried using other ways to represent the AddressList, but they cause errors. Stick with the above representation and you will avoid errors.

MotorStationsInit

This function call initializes the motor controllers on the 400NT board. Below is an example of how to use it.

```
result = MotorStationsInit
If result = -1 Then MsgBox “Error experienced!”
```

Chapter 2: Configuring & Using Digital IO / Digital Logic

The RMV400NT board has three banks of Digital Input/Output. They each consist of 14 pins each located on the board surrounded by light blue plastic. Although there are 14 pins, you can only use the first 8 for data, pin 13 for +5Volts, and pin 14 for Ground. You have to decide in advance what ports will be for input and which will be for output because they can't do both.

To make matters more complicated, you can choose to NOT use some of the pins for output. Since I don't know why you would ever want this, I generally use all 8 pins available for data. The usage is *defined* by either assigning the Digital IO Port a value of zero or 255. Zero forces the port to be used as an 8-bit *input* device. Using a value of 255 forces the port to be used as an 8-bit *output* device. For my use of the 400NT, I configure Port #1 and Port #2 for input and leave Port #3 for output. At the start of your program after you've accomplished those tasks outlined in Chapter 1, you must do the following configuration (in *this* example, Ports 1 & 2 for input, Port 3 for output):

```
Dim UseForInput As Long
Dim UseForOutput As Long
UseForInput = 0
UseForOutput = 255
result = ConfigureDigitalUserIO("1",UseForInput)
result = ConfigureDigitalUserIO("2",UseForInput)
result = ConfigureDigitalUserIO("3",UseForOutput)
```

ReadDigitalUserIO

Digital input is often used to detect if the user has pushed an external button or closed some kind of circuit. When they push the button closing the circuit, it allows voltage to flow from the +5V pin to one of the 8 available data pins on the port, showing a binary value of 1. If the circuit is not closed, then that data pin shows a voltage of about 0V, or a binary value of 0. Sometimes the voltage wanders around so you'll want to use a Pull-Down resister to insure 0V on open circuits.

Now that you've properly set up your three Digital IO ports, you probably want to read them using the ReadDigitalUserIO function call. This special function call returns not only the result variable, but it returns the essentially 8-bit integer as read by the 8 data pins on the port. The following example will give you back an 8-bit integer in the variable called BitValue.

```
Dim BitValue As Long
result = ReadDigitalUserIO(1, BitValue)
```

If none of the 8 data pins have any voltage on them, BitValue will return with some integer where the *first* 8 bits are zero. But let's imagine that you have a switch closed that applies +5V to the first data pin. To test whether or not the pin is "activated", you must use the special AND logic operator. The following subroutine takes the BitValue variable obtained from reading the Digital IO port and tests to see whether or not any of the 8 pins were set to something other than 0:

```
Dim BitValue As Long
result = ReadDigitalUserIO(1, BitValue)
If result = -1 Then MsgBox "Error experienced!"
If (BitValue And 1) > 0 Then
     'Pin 1 was set to +5V
ElseIf (BitValue And 2) > 0 Then
     'Pin 2 was set to +5V
ElseIf (BitValue And 4) > 0 Then
     'Pin 3 was set to +5V
ElseIf (BitValue And 8) > 0 Then
     'Pin 4 was set to +5V
ElseIf (BitValue And 16) > 0 Then
     'Pin 5 was set to +5V
ElseIf (BitValue And 32) > 0 Then
     'Pin 6 was set to +5V
ElseIf (BitValue And 64) > 0 Then
     'Pin 7 was set to +5V
ElseIf (BitValue And 128) > 0 Then
     'Pin 8 was set to +5V
Endif
```

WriteDigitalUserIO

As shown above, the Digital ports can be used to detect some outside action, such as the user pushing a button or closing a circuit attached to one of the 8 data pins. If configured for output, you can use the same 8 data pins to turn ON an external device. For example, if you have some machines' On/Off switch attached to the first pin of output, writing a value of 1 to that port will place +5V on that pin, activating the machine.

In most cases, you want to either turn on or off a single pin on your output port. This is accomplished by using the Or and Xor logic operators. To activate the 4th pin, I would take the current value and Or it with 8 and then write that value to the output port such as:

```
Bitvalue = BitValue Or 8  'Turn On the 4th pin.
result = WriteDigitalUserIO(3,BitValue)

BitValue = BitValue Xor 8 'Turn Off the 4th pin.
result = WriteDigitalUserIO(3,BitValue)
```