

DVT – Fanuc Modbus Communication

Communicating to Fanuc R-J3 Robot Controllers

Version 2.0

3/05

Copyright © 2002-2003, DVT Corporation

Introduction

The MBServer.pc program functions as a server for the Modbus TCP protocol on a FANUC robot. It implements a Class 0 Slave function as defined by the Open Modbus/TCP Specification (Release 1.0, 29 March 1999). As of this release, the only functions supported are “Read Multiple Registers” and “Write Multiple Registers”. MBServer.pc is a program written in KAREL and runs on R-J3 and newer robot controllers. It is intended to be used with the DVT *Modbus Object* Script functions.

Requirements

The following are the minimum hardware and software requirements for the robot and DVT SmartSensor.

FANUC

The FANUC Robot Controller must be a R-J3 or higher with the “PC Interface” option installed.

DVT

The DVT SmartSensor must have FrameWork 2.5.7 or higher installed.

NETWORK

The Robot and the DVT SmartSensor are connected via an appropriate Ethernet Network. The local connections should be through an “Intelligent Switch” rather than a Hub to minimize the amount of other traffic seen by the robot and SmartSensor.

Installation

Installation of the MBServer program includes the following steps.

1. Confirm the presence of the “PC Interface” option on the robot.
2. Configure TCP/IP settings
3. Configure a “Server Tag” for “Socket Messaging”
4. Copy the MBServer.pc program file to the robot
5. Set the MBServer program to run at power up of the robot

R-J3 Controller TCP/IP Setup

TCP/IP Setup

If not already completed, the R-J3 controller should be configured for TCP/IP operation per the instructions in the SYSTEM R-J3 Controller Internet Options Setup and Operations Manual, Section 2 SETTING UP TCP/IP, and 2.5 SETTING UP TCP/IP. These sections describe the settings for the robot's IP address, Subnet Mask, router I/P address and name, and any additional Host I/P addresses and names.

Server Tag Setup

In addition to the basic TCP/IP setup, a Server Tag must be configured per the instructions in the section "Setting up a Server Tag", Section 9.3.1 in hardcopy manuals, Section 11 in the eDoc manuals.

In this case, configure the S8: tag for Socket Messaging.

Note that it is not necessary to set the port number as shown in steps 15-22 as the MBServer.pc program will initialize this variable at run-time.

MBServer File Installation

Install the MBServer.pc program using a memory card, a PC with KFLOPPY, or PC File Services using the methods described in the FANUC Robotics SYSTEM R-J3 Controller Software Installation Manual, Appendix D.3 LOADING OR RESTORING FILES FROM EXTERNAL STORAGE TO CONTROLLER MEMORY.

When re-loading the MBServer.pc file, be sure the program is not running and not selected prior to performing the file transfer. If installing a newer version, delete the file first and also delete the MBServer.vr file that will appear when the .pc file is deleted.

Setup MBServer to run at "Power-up"

Configure the robot to run the MBServer program at power-up by entering the program name "MBSERVER" in the System Variable **\$PWR_NORMAL**. This variable is accessed from the Teach Pendant via Menu>System>Type>Variables selection. It can also be set on newer robots via the Menu>System>Type>Config menu item "Autoexec program for Cold start:"

Once started, the MBServer program cannot be aborted. If it becomes necessary to not have it run, replace or delete it, follow one of the following procedures.

1. To temporarily disable the program, select it (SELECT>Cursor to program>ENTER) then power-down and power-up. Since the

program was selected when the system was powered-down, the program will not be run on power-up. (Note also that the program must not be selected when powering-down if it is to run automatically at power up.)

2. Remove the program name from the System Variable \$PWR_NORMAL. Then power-down and power-up. The program will not run. (Be sure to re-enter the program name in \$PWR_NORMAL when automatic starting is required for production operation.)

Register Mapping

Fanuc Mapping

Register addressing for the MBServer.pc program is as follows.

The FANUC R-J3 controller has two types of registers, Position Registers (PR[n]) and general registers (R[n]). The R-J3 controller can be configured with a maximum of 200 position registers (1-200) and a maximum of 999 general registers (1-999). Writing to general registers above 250 will cause conflicts with the Modbus server mapping of the position registers. (Not many controllers will ever be configured with the maximum numbers.)

Several constraints, mostly in the Modbus Specification, but also in FANUC's and DVT's systems limit the practical maximum number of registers transferred in one block to 4 Position Registers and 30 general registers.

Position Registers are transferred as XYZWPR in real format (6 four byte real values for a total of 24 bytes).

General registers are transferred in real format (1 four byte real for a total of 4 bytes per register).

To differentiate the Position Registers from general registers, 1000 is added to the register number. Therefore, Slave Register Start values from 1 to 999 are for general registers 1 to 999, and values of 1001 to 1200 are for Position Registers 1 to 200.

As previously stated all values transferred are in Real format.

There are no pre-conditions for transferring data to or from the general registers.

Position Registers must be initialized before use and must be in Cartesian Representation. This is easily accomplished by entering zeros in all of the elements of the register from the Teach Pendant.

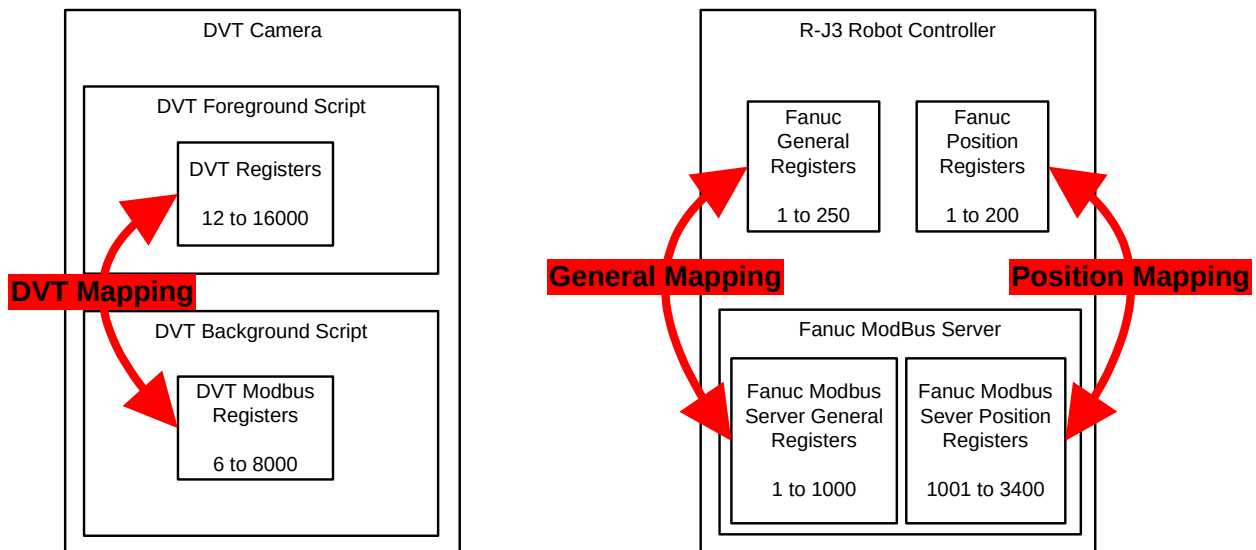
Position Register configuration data (EXT) is not transferred. When writing to a Position Register the MBServer program first reads the existing configuration data for that Position Register and then re-writes it when the position data is written. Since this application was written for offset data, this should not be a problem, as an offset should not include a configuration change.

In some cases, a previously taught position register may need to be re-taught to correct configuration data.

DVT Mapping

The registers are located in DVT general memory.

For a graphical representation of the mapping refer to the diagram below.



Fanuc Position Register	PR[1]	PR[2]	PR[3]	PR[4]	PR[200]	PR[n]
Fanuc Modbus Server	1001-1012	1013-1024	1025-1036	1037-1048	3389-3400	$[12(n-1)+1001] - [12(n-1)+1012]$

Fanuc General Register	R[1]	R[2]	R[3]	R[250]	R[999]	R[n]
Fanuc Modbus Server	1-4	5-8	9-12	997-1000	3993-3996	$[4(n-1)+1] - [4(n-1)+4]$

DVT Register	12	16	20	100	16000	n
DVT Modbus Register	6	8	10	50	8000	n/2

Transfer examples:

Example 1

Suppose you need to transfer data from DVT Modbus register 50 to the Fanuc General register 2.

In a background script we need to establish a connection and then transfer the data. The command string to pass the value to the Fanuc controller is:

```
mb.Write(50,8,2,1000);
```

or

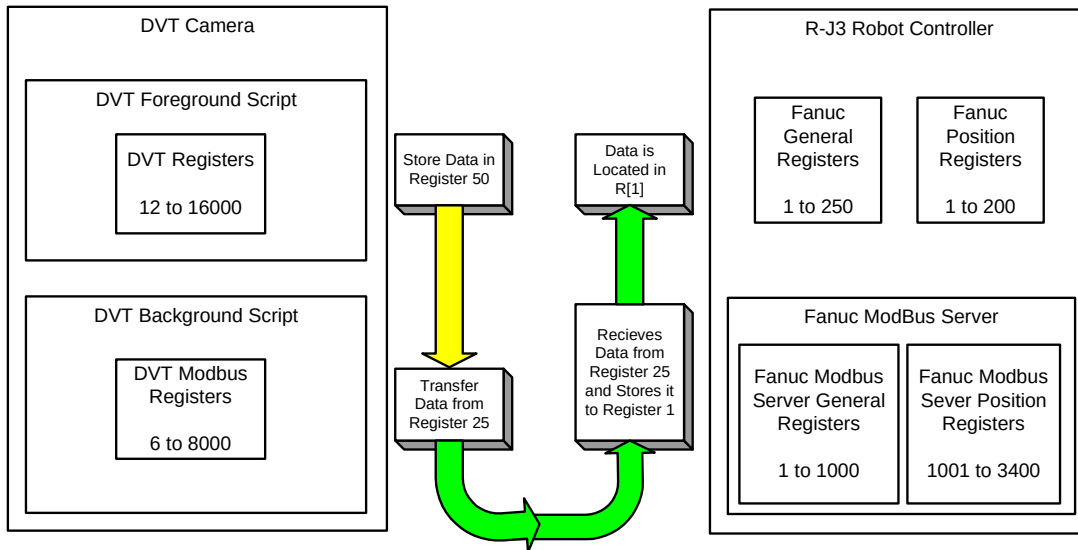
```
mb.Write(DVT Modbus Reg, Fanuc Modbus Reg, Number of Bytes, Timeout (ms));
```

Below is a complete example of the Background script:

```
class CoordinateTransfer
{
    public static void main()
    {
        int res = 0;
        MBTransfer mb = new MBTransfer();    // Declare a modbus
                                           //master object
        // Attemp to connect to the MB server
        res = mb.Connect("192.168.0.10");
        // Change this to the IP of the RJ-3 controller
        if (res != 0)
        {
            DebugPrint(" could not connect.  Error: " + res);
        }
        while (res == 0)
        {
            res = mb.Write(50,8,2,1000);
            // Read Value From DVT Modbus Register 50, and
            //Store in Fanuc General Register [R] 2.
        }
        DebugPrint(" Exited due to Error: " + res);
    }
}
```

Example 2

Transfer data from a blob select tool in a Foreground script, to a Fanuc General Register.



Example 2

Transfer Number of Blobs to Fanuc General Register 1

FOREGROUND SCRIPT

```

class BlobLocationTransfer
{
    public void inspect()
    {
        RegisterWriteFloat(50,BlobPosition.NumBlobs); // Register 50 contains the Number of blobs
    }
}
  
```

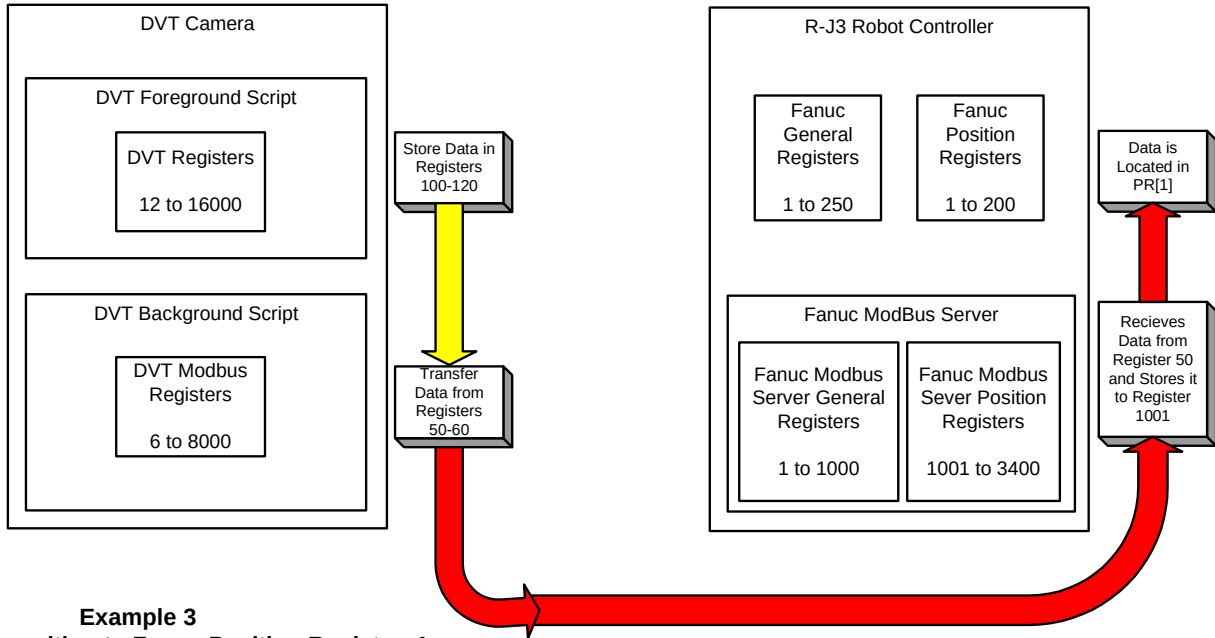
BACKGROUND SCRIPT

```

class CoordinateTransfer
{
    public static void main()
    {
        int res = 0;
        MBTransfer mb = new MBTransfer(); // Declare a modbus master object
        // Attempt to connect to the MB server
        res = mb.Connect("192.168.0.10"); // change this to the IP of the RJ-3 controller
        if (res != 0)
        {
            DebugPrint(" could not connect. Error: " + res);
        }
        while (res == 0)
        {
            res = mb.Write(25,4,2,1000); // Read Value From DVT Register 50, and Store in
            // Fanuc General Register [R] 1.
        }
        DebugPrint(" Exited due to Error: " + res);
    }
}
  
```

Example 3

Transfer position data from a blob select tool in a Foreground script, to a Fanuc Position Register.



Example 3 Transfer Blob position to Fanuc Position Register 1

FOREGROUND SCRIPT

```

class BlobLocationTransfer
{
    public void inspect()
    {
        RegisterWriteFloat(100,BlobPosition.BlobPosition.X[0]); // Register 100 contains the X position of blob 0
        RegisterWriteFloat(104,BlobPosition.BlobPosition.Y[0]); // Register 104 contains the Y position of blob 0
        RegisterWriteFloat(108,0); // Register 108 contains the Z position of blob 0
        RegisterWriteFloat(112,BlobPosition.BlobAngle[0]); // Register 112 contains the angle of blob 0
        RegisterWriteFloat(116,0); // Register 116 contains the Pitch
        RegisterWriteFloat(120,0); // Register 120 contains the Roll
    }
}
  
```

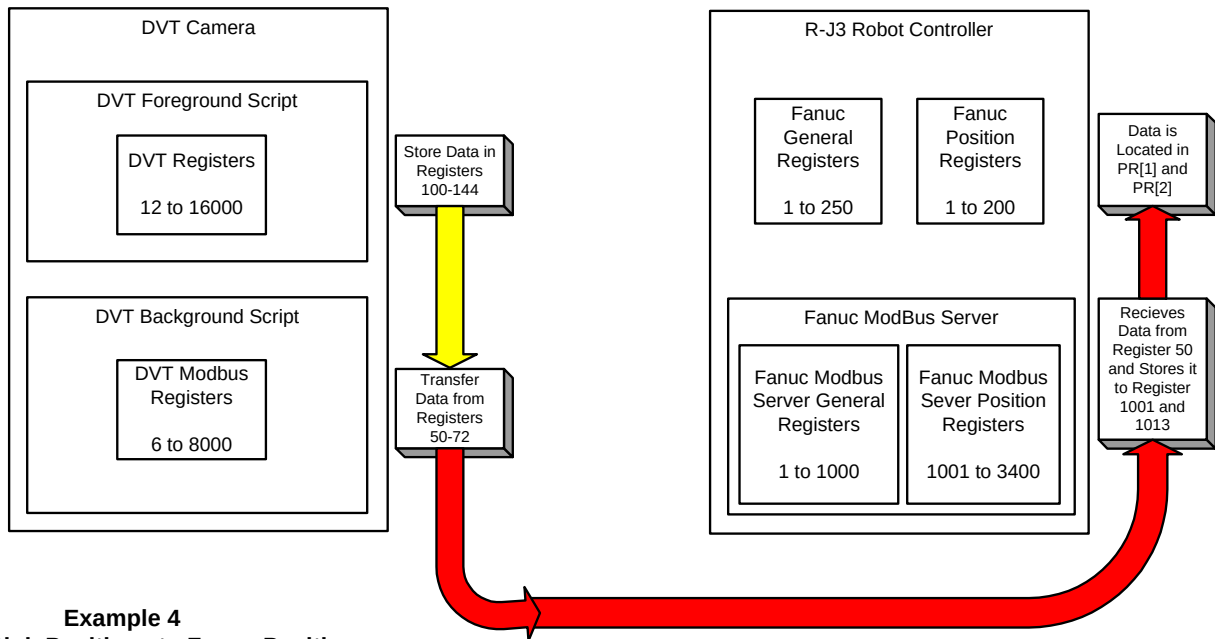
BACKGROUND SCRIPT

```

class CoordinateTransfer
{
    public static void main()
    {
        int res = 0;
        MBTransfer mb = new MBTransfer(); // Declare a modbus master object
        // Attempt to connect to the MB server
        res = mb.Connect("192.168.0.10"); // change this to the IP of the RJ-3 controller
        if (res != 0)
        {
            DebugPrint(" could not connect. Error: " + res);
        }
        while (res == 0)
        {
            res = mb.Write(50,1001,12,1000); // Read Value From DVT Register 100, and
            // Store in Fanuc Position Register [PR] 1.
        }
        DebugPrint(" Exited due to Error: " + res);
    }
}
  
```

Example 4

Transfer position data from a blob select tool in a Foreground script, to two Fanuc Position Registers.



Example 4

Transfer 2 Blob Positions to Fanuc Position Registers 1 and 2

FOREGROUND SCRIPT

```

class BlobLocationTransfer
{
    public void inspect()
    {
        RegisterWriteFloat(100,BlobPosition.BlobPosition.X[0]); // Register 100 contains the X position of blob 0
        RegisterWriteFloat(104,BlobPosition.BlobPosition.Y[0]); // Register 104 contains the Y position of blob 0
        RegisterWriteFloat(108,0); // Register 108 contains the Z position of blob 0
        RegisterWriteFloat(112,BlobPosition.BlobAngle[0]); // Register 112 contains the angle of blob 0
        RegisterWriteFloat(116,0); // Register 116 contains the Pitch
        RegisterWriteFloat(120,0); // Register 120 contains the Roll

        RegisterWriteFloat(124,BlobPosition.BlobPosition.X[1]); // Register 124 contains the X position of blob 1
        RegisterWriteFloat(128,BlobPosition.BlobPosition.Y[1]); // Register 128 contains the Y position of blob 1
        RegisterWriteFloat(132,0); // Register 132 contains the Z position of blob 1
        RegisterWriteFloat(136,BlobPosition.BlobAngle[1]); // Register 136 contains the angle of blob 1
        RegisterWriteFloat(140,0); // Register 140 contains the Pitch
        RegisterWriteFloat(144,0); // Register 144 contains the Roll
    }
}
  
```

BACKGROUND SCRIPT

```

class CoordinateTransfer
{
    public static void main()
    {
        int res = 0;
        MBTransfer mb = new MBTransfer(); // Declare a modbus master object
        // Attempt to connect to the MB server
        res = mb.Connect("192.168.0.10"); // change this to the IP of the RJ-3 controller
        if (res != 0)
        {
            DebugPrint(" could not connect. Error: " + res);
        }
        while (res == 0)
        {
            res = mb.Write(50,1001,24,1000); // Read Value From DVT Register 100-144,
            // and Store in Fanuc Position Register [PR] 1
            // and [PR] 2.
        }
        DebugPrint(" Exited due to Error: " + res);
    }
}
  
```

DVT ModBusMapper

Determining the correct values to put into the script string can become cumbersome. The ModBusMapper program can help out with this issue.

Below is a screenshot of the application.

INSTRUCTIONS

1. Select the DVT Register you want to Access.
2. Select the Fanuc Register you want to Access.
3. Select the Type of Fanuc Register.
4. Select the Number of Items to Transfer.
5. Click the "Generate Script String" Button.
6. Copy and Paste the Desired String to your Background Script.

DVT - Fanuc Modbus Mapper

DVT Register: 100 Range 12 - 16,000

Fanuc Register: 1 Range R[1 - 200] PR[1 - 250]

Fanuc Register Type: General Register [R]

Number of Consecutive Items to Transfer: 1 Range R[1 - 30] PR[1 - 4]

Generate Script String

Write to Fanuc: mb.Write(50,4,2,1000); // Reac

Read from Fanuc: mb.Read(50,4,2,1000); // Read

Once all of the parameters are entered and the "Generate Script String" button is pressed. Two strings are constructed along with comments. The string/strings then will have to be copied and pasted into your Background script.

It is not necessary to use this application but it should make things easier.

*Note: The Timeout delay is assumed to be 1000ms. If another time is desired you can change it after you paste it into your script.

The syntax of the string for the write and read functions are very similar. The variables, values inside the parenthesis are the same.

They are: (DVT Modbus Reg, Fanuc Modbus Reg, Number of Bytes, Timeout (ms))