Modbus Server

Description

The ModBus server allows:

- The cyclic or acyclique interrogation of equipments connected to the serial comport **COM1** to **COM10**.
- Up to 115200 Bauds.
- The communication server refreshes a memory database of variables.
- The communication server get in a memory variable database the values to send to the equipments the communication server places in icon in the task bar of Windows.
- The communication server runs under Windows 98, NT, 2000 & XP.
- The communication server can be connected to programs written in different languages. Read or write variables are made using a library (DLL) or by an object Automation (inprocserver) contain in the VPLCOM.DLL file.

The different programs installed by the Setup.exe program are:

ServModbus.exe: The Modbus configuration.

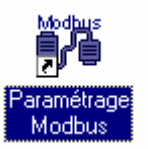
Server.exe: The final communication server functioning with the parameters defined by the configuration.

Visudyna.exe: Debugging grid allowing visualizing global variables in the server.

Recense.exe: A utility to register or unregister the COM object in the Windows register.

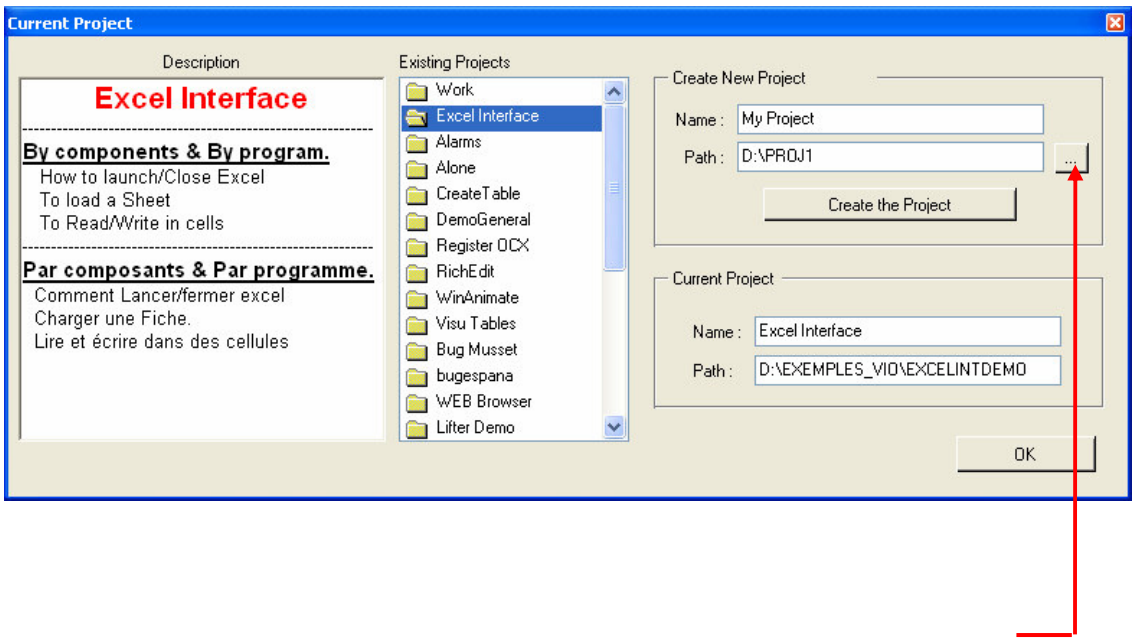
ServModbus.exe

This configuration makes it possible settings Modbus frames on the serial link

Click on the icon 

First Step

If no project is defined you can define it by clicking the button change current project on the top of the form:



I
indicate in the field *Name*, the name of your new project (here My Project)
Indicate in the field *Path*, the complete path where must be located this new project (here D:\Proj1).
If the directory already exists or not, you can navigate, choose or create it while clicking on this button

Then click on the **Create the Project** to record the name and the path of this one in the register of Windows.
This project is listed then in the central list box, it is easy to select it by double-clicking above.

Note: you create a new project, no global variable is defined. You must define them.

The description of the project (optional) is possible on the left edit box in RTF.
To obtain different the graphic Font and graphics effects, constitute your text under Word then to carry out a Copy/Paste command in the zone of Description

Second Step

Define an image table for communications

After defined a new project, you must to define global variables receiving values to transmit to slaves or variables receiving the result of communications from the Modbus. It is the Modbus image table. Normally in the recent installation a project is already created at setup with several global variables.

Type of Variables in the Modbus server

Boolean : Byte (True=1 False=0).
Byte : Byte 8 bits. (Bits from 0 to 7)
Word : Word 16 bits. (Bits de 0 to 15).
Integer : Integer 32 Bits. (Bits de 0 to 31). Range: -2147483648...2147483647
Real : Floating number. Range: $1,9x10^{-4951}$.. $1,1x10^{4932}$
Single : Floating point (on 4 bytes)
String : String of 255 characters. (From 1 to 255).
Array : Array of variables defined below.
Example: Array [0..12] of Boolean;

Addressing

Boolean

```
MBool  : Boolean;
MBool:=True; MBool:=False;    or equivalent Mbool:=1; Mbool:=0;
```

Byte

```
By     : Byte;          By:=12;
```

Word

```
WD     : Word;          WD:=12000;    WD:=$F012;
```

Integer

```
It     : Integer        ;
It:=255; It:=$FF;       Allocation of a numerical value;
It.0:=True; It.4:=False; Allocation of a Bit of an Integer;
```

Real

```
RL     : Real;          RL:=-12.40;
```

String

```
ST     : String;        ST:='Visual PLC';       ST:='C:\WinNt';
```

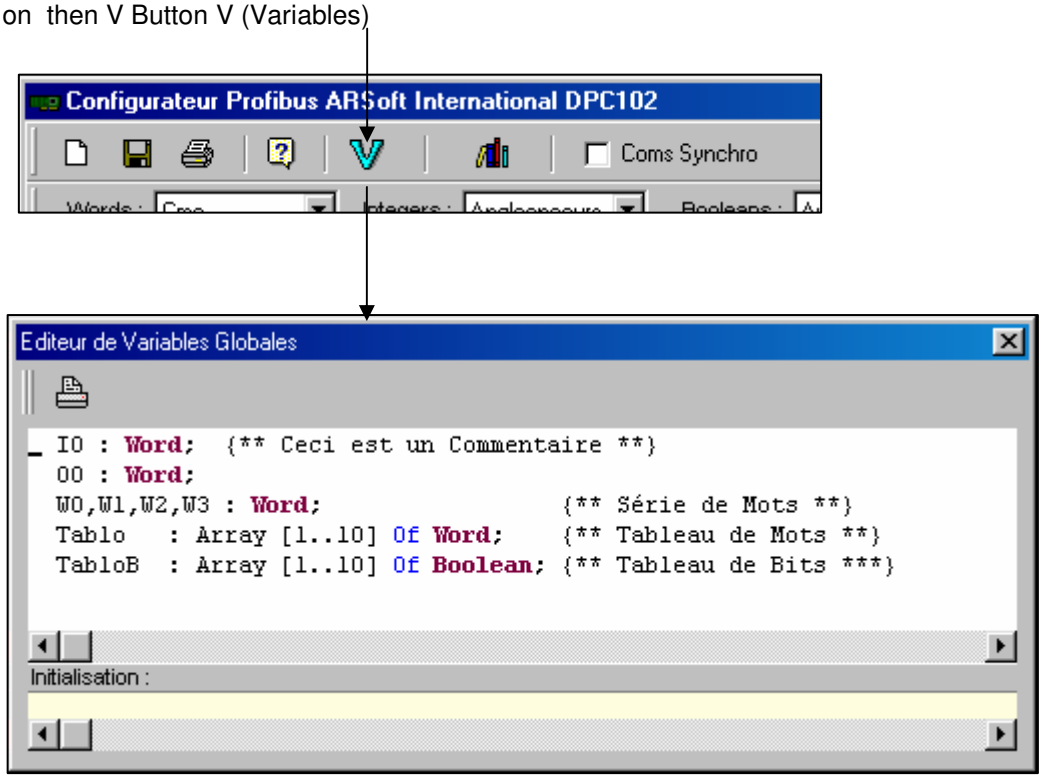
Array

```
AMBOOL: Array [0..10] of Boolean;     AMBool[1]:=True;

ABY    : Array [1..50] of Byte;   ABY[2]:=34;

AWD    : Array [1..50] of Word;  AWD[2]:=12034;

AIT    : Array [1..30] of Integer;          AIT[4]:=-255;   AIT[5].0 :=True;

ARL    : Array [2..14] of Real;   ARL[2]:=-34.6789;

AST    : Array [0..6] of String;   AST[2]:='Bonjour';
```

For example an image table corresponding to a block of 16 physical inputs and a block of 16 physical output. The 16 physical Inputs will be copied by the Modbus Server in the Variable I0 and 16 Outputs will be physically affected by the content of the variable O0 thanks to the server.

Example of declaration :

Click on then V Button V (Variables)

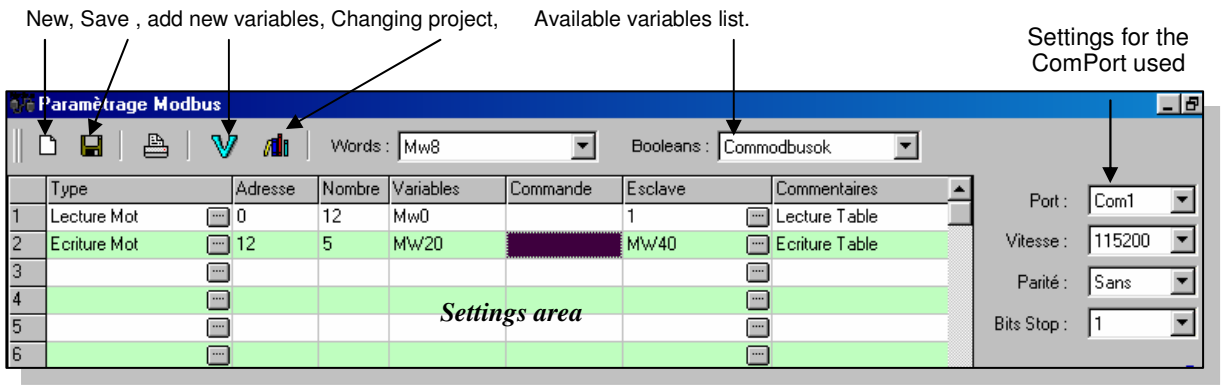


Here I0,O0 is a Word of 16 Bits. W0 to W3 are 4 words 16 bits.
Tablo is a table of 10 Words of 16 Bits. Tablob is a table of 10 Bits (Boolean).

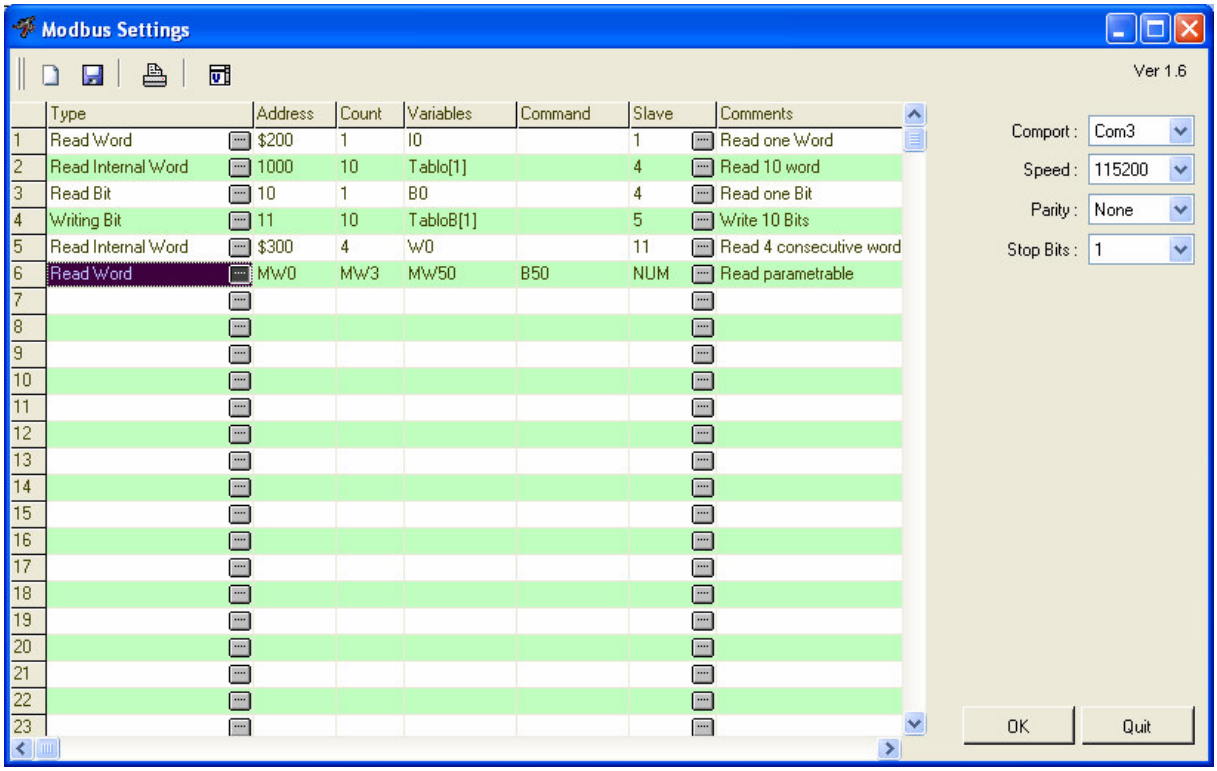
The choice of variable names constituting the image table is left free to the user. We can imagine names such that InputCard1, Consigne1, Temperature10 Etc..
Functions in the DLL or in the object Automation will need these names to address these variables.
The ARSOFT OPC server returns also these names of variables.

Possibility to read or force a bit in a word (ex : W0.0 ; InputCard.0 etc..)

You can not set communication, before to define variables constituting the image table.



Example of settings :



	Type	Address	Count	Variables	Command	Slave	Comments
1	Read Word	$200	1	I0		1	Read one Word
2	Read Internal Word	1000	10	Tablo[1]		4	Read 10 word
3	Read Bit	10	1	B0		4	Read one Bit
4	Writing Bit	11	10	TabloB[1]		5	Write 10 Bits
5	Read Internal Word	$300	4	W0		11	Read 4 consecutive word
6	Read Word	MW0	MW3	MW50	B50	NUM	Read parametrable
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							

Line 1 : Read in **slave 1 one word** to address **200** in hexa ($200) and store the value in variable I0 (Word type 16Bits).

Line 2 : Read in **slave 4 10** words beginning at address **1000** in decimal (100) and store these words in an array names Tablo (tablo[1] to tablo[10]).

Line 3 : Read in **slave 4** one bit at address 10 and store result in variable B0 (Boolean).

Line 4 : Write in **slave 5** of 10 bits at address 11 with the values containing in variables TabloBB[1] to TabloBB[10] (Boolean type).

Line 5 : Read in **slave 11** of **4** words beginning at address **$300** in hexa and storage of these in the variables W0, W1, W2 and W3 (these variables are declared consecutively in the editor, the received words are also store consecutively in these variables).

Line 6 : Read words in the slave with address contained in the variable NUM de **X** words (value of the variable MW3) beginning at the address contained in the variable MW0 et storage of these in variables MW50 (these being declared consecutively, the received words are also stored consecutively in these variables). **The frame send if the Bit B50 is True**. When the communication is perform correctly or when a timeout occurs, **the variable B50 is reset by the server**, allowing to perform an acknowledge. You can test StatusModbus[X] to known the result of the communication.

NB : You can test or force the les bits and the words read or written (ex : Inputcard or outputcard) W0.0, O0.0 I0.0 etc..

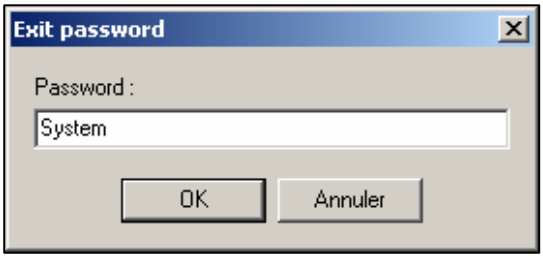
Save these settings and run the Modbus.

Launching the communication server

SERVER.exe

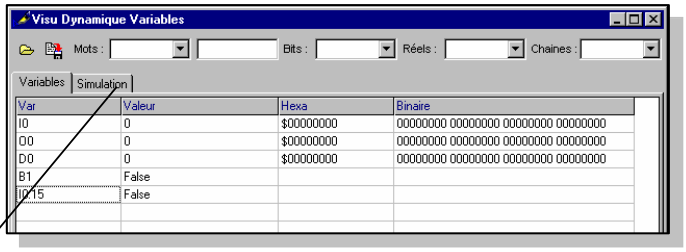
The server with launching comes to position in the taskbar of Windows indicating its presence.

By clicking on the icon in the task bar Windows a window appears :
By clicking on the 'Quit' button, it is possible to stop the real Time engine by striking the password **'System' or '456789'**.



Dynamic visualization of the Variables

This program makes it possible to explore the variables declared moving in the communications server. A dynamic visualization and a forcing numerically, Hexadecimal and binary are possible. This list can be safeguarded on disc on different name.



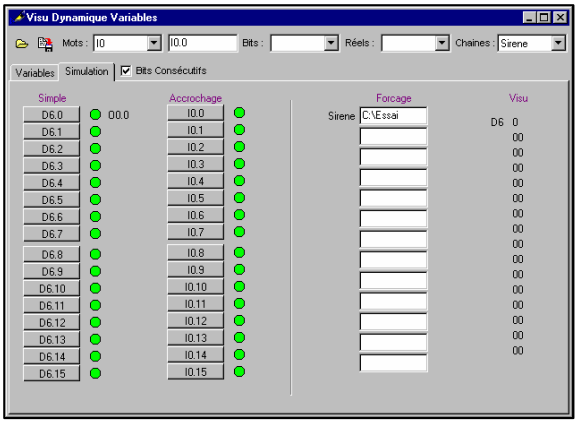
Simulator

A simulator makes it possible to assign to simple or toggle pushbuttons Boolean variables. To affect a button or an indicator, bring by Drag & Drop from the edit box the name of the variable.
If the check box consecutive is checked, the following buttons take the names of the adjacent variables automatically.

To erase the assignment of a variable to a button, erase the text in the edit box on the right of Combo Box '*Words*', then bring by drag & drop the empty text on the button, the indicator or another element of visualization.

In the column **Forcing**, edit boxes are available to directly affect numeric variables and character strings.
In the column **visu**, a simple visualization of variables is possible without forcing of those.

Combo box on top, make it possible to memorize the name of the global variables the Drag & Drop are possible from these.



Other informations

The time out for a Modbus communication is 1 second. In another words if the slave don't reply during 1 second the server cancel the current transaction with this slave and continue its work.

When the frame is conditioned by a command Bit
If any bit is specified the interrogation of the equipment is made regularly by the server. If a bit is specified, the interrogation of the concerned equipment is provided when this bit (Boolean) is set. When the frame has been read (the equipment replied without error) this bit is set to 0 (False) by the server returning thus an acknowledgement of reception.

If the transaction is correctly performed with the equipment, the server set StatusModbus[X] else these bits are reset.
However in all cases the command bit is reset (if ComOk or TimeOut).

StatusModbus is array of Boolean indicating the good working communications.

Ex : StatusModbus[1] is true when the Modbus command is OK (reply ok) corresponding of the first line of the grid .

The configuration adds automatically the following variables :

ComModbusOK : Boolean ;

Bit set if the communication port is initialized correctly. Comport open without error.
If this bit is false, the communication port is already open by another application .

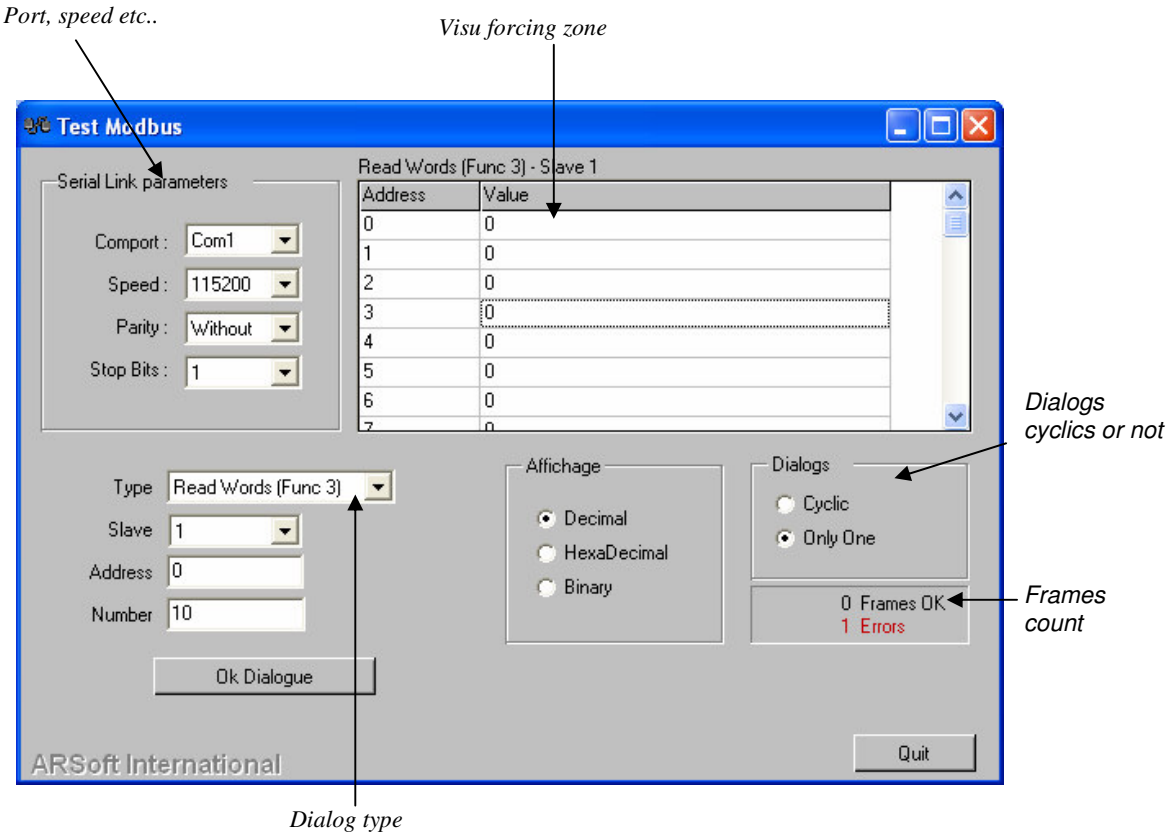
StatusModbus : ARRAY [0..255] Of Boolean;

Each indice of this array reflects the state of the corresponding communication.

If you have 3 communication frames,
The status of the first frame is StatusModbus[1] this bit is true if the communication with the concerning slave replied correctly.
The status of the second communication is in StatusModbus[2].
And so on..

Utility to test the Modbus communication

This utility makes it possible to generate frames cyclic or not on the communication port, in order to test the connected equipment and to visualize all the area of this one.



This utility used the DLL modbus another product of ARSOFT.
The DLL modbus is used to incorporate directly functions in a program written in different languages like VB, Delphi, C++

Programmation

The product is delivered with 2 DLL : **VPC.DLL et VPLCOM.DLL**

VPLC.DLL is a classical DLL allowing the interfacing with different language like Delphi or Visual C++. However a problem with string parameters en procedures and functions (Pointer on chars), It impossible for certain language as Visual Basic to pass directly a simple String. The VB strings are in Unicode format incompatible with pchar.

It possible then to resolve this problem by using VPLCOM.DLL which is a com Object receiving the UNICODE strings.

For Visual Basic programs use the COM object (VPLCOM.DLL) for calling the methods listed below. This object is registered automatically at setup.

Using the DLL :
****
!! Attention case sensitive concerning the name of the DLL and the name of the procedures
****
Using the COM Object:
The methods and the parameters are identical.
The case sensitive is not necessary.

Prototypes et comments :

Function GetVpuInt (Name : PChar) : Integer;
-> Get the "Name" value (Integer type) ;

Procedure SetVpuInt (Name : PChar; Value : Integer);
-> Set the "Value" value to the variable "Name" (Integer type);

Function GetVpuWord (Name : PChar) : Word;
-> Get the "Name" value (Word type) ;

Procedure SetVpuWord (Name : PChar; Value : Word);
-> Set the "Value" value to the variable "Name" (Word type);

Function GetVpuBool (Name : PChar) : Boolean;
➔ Get the "Name" value (Boolean type)
➔ Possibility to get a bit in a word ex : GetVpuBool('W0.0')

Procedure SetVpuBool (Name : PChar; State : Boolean);
-> Set the "Value" value to the variable "Name" (Boolean type);

Function GetVpuByte (Name : PChar) : Integer;
-> Get the "Name" value (Integer type) ;

Procedure SetVpuByte (Name : PChar; Value : Byte);
-> Set the "Value" value to the variable "Name" (Byte type);

Function GetVpuString(Name : PChar) : PChar;
-> Get the "Name" value (String type) ;

Procedure SetVpuString(Name : PChar; Value : String);
-> Set the "Value" value to the variable "Name" (String type);

Function GetVpuReal (Name : PChar) : Single ;
-> Get the "Name" value (Real type) ;
!!Attention the return value is store in 4 bytes its range is limited
Compared to the variables of Visual PLC type Real 10 bytes (Extended).

Procedure SetVpuReal (Name : PChar; Value : Single);
-> Set the "Value" value to the variable "Name" (Real type);
!!Attention the value sent is converting in single decreasing the range possible of this value.

{**** Access to Array **********}
Function GetVpuAInt (Name : PChar; Element : Integer) : Integer;
-> Get the value of an array "Name" at the indice "Element" return in Integer type;

Procedure SetVpuAInt (Name : PChar; Element : Integer; Value : Integer);
-> Set the value "Value" at the indice "Element" of an array "Name" in integer type;

Function GetVpuAWord (Name : PChar; Element : Integer) : Word;
-> Get the value of an array "Name" at the indice "Element" return in Word type;

Procedure SetVpuAWord (Name : PChar; Element : Integer; Value : Word);
-> Set the value "Value" at the indice "Element" of an array "Name" in Word type;

Function GetVpuABool (Name : PChar; Element : Integer) : Boolean;
-> Get the value of an array "Name" at the indice "Element" return in Boolean type;

Procedure SetVpuABool (Name : PChar; Element : Integer; Value : Boolean);
-> Set the value "Value" at the indice "Element" of an array "Name" in Boolean type;

Function GetVpuAByte (Name : PChar; Element : Integer) : Integer;
-> Get the value of an array "Name" at the indice "Element" return in Integer type;

Procedure SetVpuAByte (Name : PChar; Element : Integer; Value : Byte);
-> Set the value "Value" at the indice "Element" of an array "Name" in Byte type;

Function GetVpuAString(Name : PChar; Element : Integer) : PChar;
-> Get the value of an array "Name" at the indice "Element" return in PChar type (Pointer on first chars);

Procedure SetVpuAString(Name : PChar; Element : Integer; Value : PChar);
-> Set the value "Value" at the indice "Element" of an array "Name" in Pchar type (Pointer on first chars);

Function GetVpuAReal (Name : PChar; Element : Integer) : Single;
-> Get the value of an array "Name" at the indice "Element" return in Single type

Procedure SetVpuAReal (Name : PChar; Element : Integer; Value : Single);
-> Set the value "Value" at the indice "Element" of an array "Name" in Real or single type;

{*** Access at a word whatever its type *****}
Function GetVpuValue (Name : PChar) : PChar;
-> Get the value of the variable "Name" whatever its type and returns it in Pchar (pointer on first character)

{*** Set a value to any variable *****}
Procedure SetVpuValue (Name,Value : String);
Example : SetVpuValue ('Counter','10');
Example : SetVpuValue ('State[1]','True');

{*** Set the path containing the MVarglob.VPU File *********}
Procedure SetPathVarglob(Path : String);
Example : SetPathVarglob('C:\Test1\'); //*** Attention '\' at the end of the string

{**** Get the address of the specified variable *********}
Function GetVpuAddr (Nom : String) : Pointer;
Example : Address :=GetVpuAddr (Counter) ; returns the memory address of the variable Counter

Procedure ClearOptimisation;
-> At each function call, the address of the variable "Name" is saved in a temporary array for increase the performances of access to the global variables.
At a certain time, this array is more important than the array of the global variables. It is necessary to reinit the temporary array thanks to this procedure.

Using the DLL

Programmation in Delphi :
declare all prototypes of functions and procedures in var

VAR
Function **GetVpuInt** (Name : PChar) : Integer; StdCall; External 'vplc.dll';
Procedure **SetVpuInt** (Name : PChar; Value : Integer); StdCall; External 'vplc.dll';
Etc..

Use them like normal procedures in your program.

Implementation in Visual Basic

Impossible use by COM Object VPLCOM.DLL

Implementation in Delphi and Visual Pascal

Declare all prototypes (functions & procedures) in var section

```
VAR
  Function  GetVpuInt   ( Name : PChar) : Integer; StdCall; External 'vplc.dll';
  Procedure SetVpuInt   ( Name : PChar; Value : Integer); StdCall; External 'vplc.dll';
Etc..
```

VarVpuD.pas for Visual Pascal and Delphi

The VarVpuD.pas can perform read/write operations with the same procedures and function included in the DLL. The Source (*.pas) is supplied with the product. Recompile with your Delphi version the varvpud.pas.

Implementation in C

```
#define       DWORD       unsigned int     // 32 bits
#define       BYTE        unsigned char    // 8 bits
```

DWord **GetVpuInt** (Char * Name) ;

Use them like normal procedures in your program.

Using GetVpuAddr

Example :
The words declared in the global variables are located consecutively in the memory of the server. It is possible to get the address of the first word then read or write consecutively the bytes.

Example of declared global variables.

```
Var
        Counter         : Integer ;       {** 4 Bytes **}
        State           : Boolean         {**1 Byte **}
        Eana            : Word ;          {** 2 Bytes **}
```

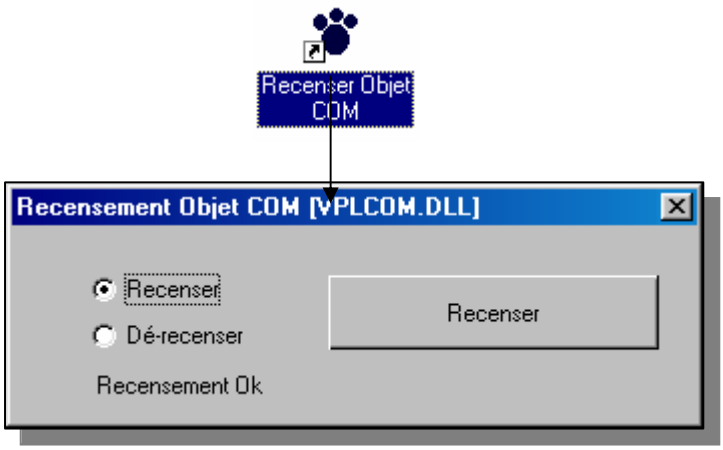
Example in Pascal (Visual Pascal or Delphi)

```
Var
  Pt                 : Pointer ;
  SaveCounter  : Integer ;
  SaveState       : Boolean ;
  SaveEana       : Word ;

Begin
 PT :=GetVpuAddr(Counter) ;       //** Get the address of the first word  (counter variable)***}
 Move(PT,SaveCompteur,4+1+2) ;   {** Move from server memory to local memory (local variable) **}
End ;
```

Using COM object

The object name is **VPLC.OBJ**

Before using the com object (object Automation) VPLCOM.OBJ, you must register it in the registry of Windows. Click on this icon



Choose radio-button Recenser or dé-recenser then click on the button to proceed.
Quit the program, the Com object is now available.

Programmation in Visual Basic

```
Private Sub Command1_Click()
Dim Svr As Object
Dim I   As Long
Set Svr = CreateObject("VPLC.OBJ")      '*** Create Object

Svr.SetVpuInt "W0", 123                 '*** Call Method SetVpuInt
I = Svr.GetVpuInt("W0")                 '*** Call Method GetVpuInt
Label1.Caption = Svr.GetVpuString("TS")  '*** Call Method GetVpuString
End Sub
```

Programmation in Delphi and Visual Pascal
Example :

```
Var
  Svr : Variant;

Procedure TForm1.Button1Click(Sender: TObject);
Var
  I : Integer;
Const
  InitDone : boolean=False;
begin
  IF InitDone =False Then
  Begin
    Svr:=CreateOleObject('VPLC.OBJ');   //*** Create Object

    InitDone:=True;
  End;
  Svr.SetVpuInt('W0',-12);              //*** Call Method SetVpuInt
  I:=Svr.GetVpuInt('W0');               //*** Call Method GetVpuInt
  Label1.Caption:=IntToStr(I);
end;
```