Serveur Modbus

Ver 1.5

Description

Le Serveur ModBus permet :

- L'interrogation cyclique d'esclaves connectées sur les ports série de communication **COM1** à **COM4**.
- A une vitesse inférieure ou égale à 115200 bauds.
- Le serveur de communication **rafraîchit cycliquement** une base de données de variables.
- Le serveur communication puise dans une base de données de variables les valeurs à envoyer vers les esclaves paramétrés.
- Le serveur de communication se place comme une **tache prioritaire** dans la barre de taches de Windows.
- Le serveur de communication fonctionne tout aussi bien sous Windows 95, 98 et NT.
- Le serveur de communication peut être relié à des programmes écrit dans différents langages évolués par le jeu de fonctions contenues dans une DLL (VPLC.DLL) ou par un objet Automation (inprocserver) contenu dans VPLCOM.DLL.

Les différents programmes installés par le programme Setup.exe contenu sur le disque fourni sont :

ServModbus.exe: Le configurateur de trames modbus.

Tmodbus.exe: Un utilitaire pour tester vos équipements en modbus connectés sur la liaison Série.

Server.exe: Le serveur de communication final fonctionnant avec les paramétrages défini par le configurateur.

Visudyna.exe: Le programme de débug permettant de visualiser les variables évoluant au sein du serveur.

Recense.exe: Un utilitaire pour recenser ou dé-recenser l'objet COM dans la base de registre de Windows.

Autres Fichiers :

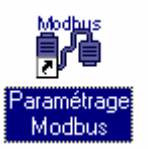
Visuln32.DLL : Communication mémoire à mémoire avec le serveur.
Vplc.dll : Contient les procédures et les fonctions de communication avec le serveur.
Vplcom.dll : Objet COM (Inprocserver) de communication avec le serveur.
VarVPu.Dcu : Librairie pour Delphi 4 et Delphi 5. Contient les mêmes procédures et fonctions que celles implémentées dans la DLL Vplc.dll

Sous Répertoire Test : Contient une application Visual I/O complète de test en communication avec le serveur.
Sous répertoire VB : Contient une application complète en Visual Basic, permettant de lire et écrire des variables dans le serveur Profibus.

ServModbus.exe

Ce programme permet le paramétrage des trames Modbus à envoyer cycliquement sur la liaison série :

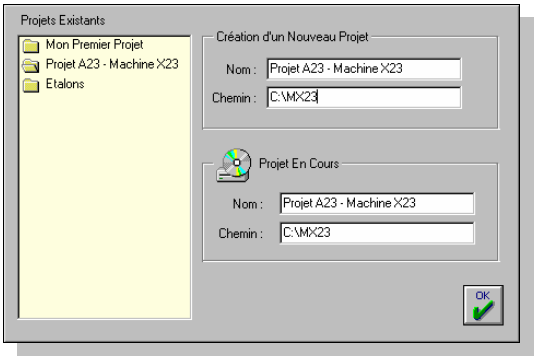
Cliquez sur l'icône



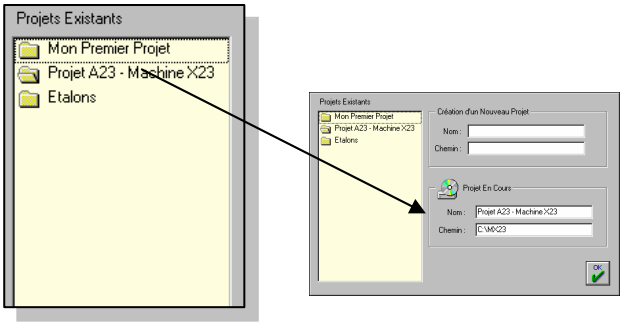
Première Etape

Au premier lancement, une fenêtre apparaît permettant de renseigner un nom de projet ainsi que la directorie dans lequel se trouve celui-ci :

Dans le champ **Nom**, frappez l'intitulé du projet. Ici Projet A23 - Machine X23.
Dans le champ **Chemin**, frappez le chemin complet sur le disque. Celui-ci contiendra tous les fichiers de votre application



Le nouveau projet créé apparaît dans la liste de gauche. A ce stade il n'est toujours pas validé. Double cliquez sur celui-ci pour ce projet nouvellement créé.



Deuxième Etape

Définition de la table image des communications

Après avoir défini un nouveau projet, vous devez définir les variables globales recevant les valeurs à transmettre vers les esclaves ou les variables recevant le résultat des communications en d'autres termes la table image Modbus.

Type de Variables dans le serveur Modbus

Boolean : Octet (True=1 False=0).
Byte : Octet 8 bits. (Bits de 0 à 7)
Word : Mot 16 bits. (Bits de 0 à 15).
Integer : Entier 32 Bits. (Bits de 0 à 31). Portée : -2147483648..2147483647
Real : Nombre en virgule Flottante. Portée : $1,9x10^{-4951}$.. $1,1x10^{4932}$
String : Chaîne de 255 caractères. (De 1 à 255).
Array : Tableau de type de variables définies ci-dessus.
Exemple : Array [0..12] of Boolean;

Adressage

Boolean

```
MBool   : Boolean;
MBool:=True; MBool:=False;      ou équivalent Mbool:=1; Mbool:=0;
```

Byte

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

Word

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

Integer

```
It      : Integer ;
It:=255; It:=$FF;        Affectation d'une valeur numérique;
It.0:=True; It.4:=False; Affectation d'un bit d'un Integer;
```

Real

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

String

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

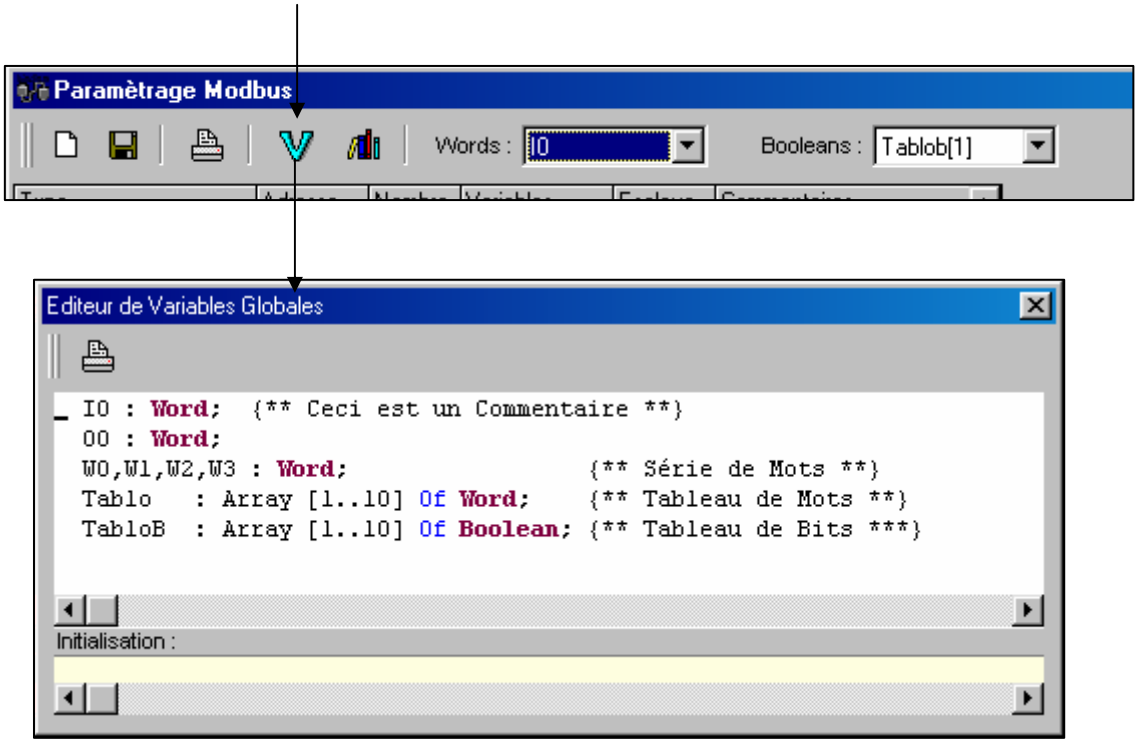
Array

```
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';
```

Prenons comme exemple une table image correspondant à un bloc de 16 entrées physiques et à un bloc de 16 sorties physiques. Ces 16 Entrées physiques seront recopiées par le serveur modbus dans la Variable **I0** et les 16 sorties seront affectées par le contenu de la variable **O0** à travers le serveur.

Exemple de déclaration :

Cliquez sur le Bouton V (Variables)

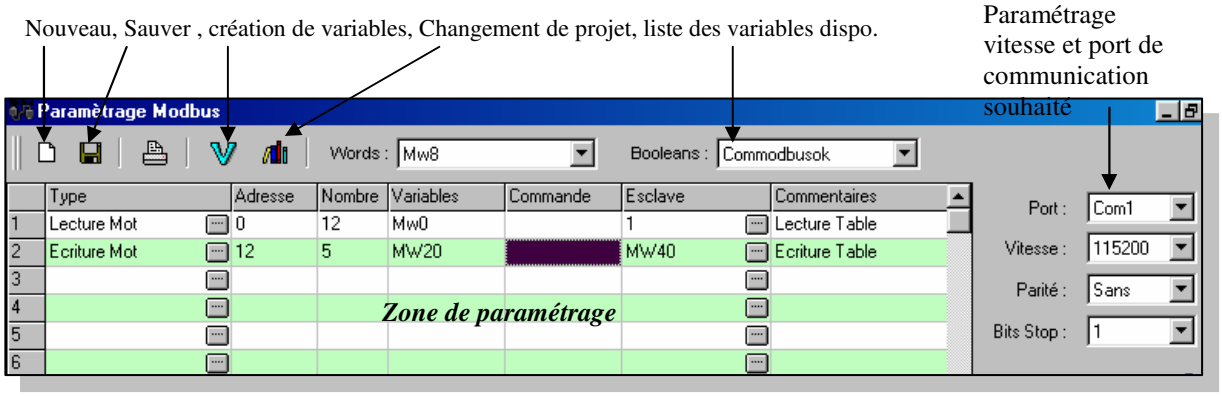


Ici I0,O0 est un Mot de 16 Bits. W0 à W3 sont 4 mots de 16 bits.
Tablo est un tableau de 10 Mots de 16 Bits. Tablob est un tableau de 10 Bits (Boolean).

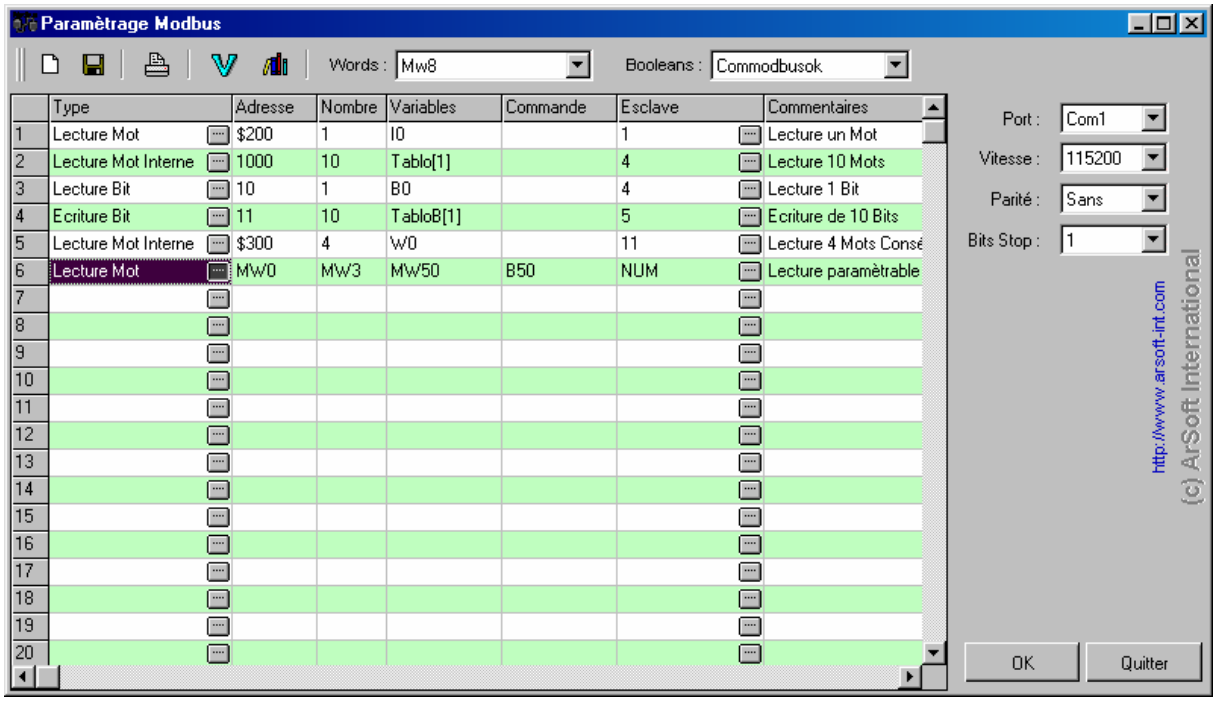
Le choix des noms de variables constituant la table image est laissée libre à l'utilisateur. L'on peut imaginer des noms tel que CarteEntree1, Consigne1, Temperature10 Etc..

Les fonctions de la DLL ou de l'objet Automation auront besoin de ces noms pour adresser ces variables.
Possibilité de récupérer ou de forcer un bit d'un mot (ex : W0.0 ; CarteEnt.0 etc..)

Vous ne pouvez pas paramétrer de communication, avant d'avoir défini des variables constituant la table image.



Exemple de paramétrage :



	Type	Adresse	Nombre	Variables	Commande	Esclave	Commentaires
1	Lecture Mot	$200	1	I0		1	Lecture un Mot
2	Lecture Mot Interne	1000	10	Tablo[1]		4	Lecture 10 Mots
3	Lecture Bit	10	1	B0		4	Lecture 1 Bit
4	Ecriture Bit	11	10	TabloB[1]		5	Ecriture de 10 Bits
5	Lecture Mot Interne	$300	4	W0		11	Lecture 4 Mots Consé
6	Lecture Mot	MW0	MW3	MW50	B50	NUM	Lecture paramètrable

Ligne 1 : Lecture dans l'esclave d'adresse **1 d'un** mot à l'adresse 200 en Hexa ($200) et stockage de celui-ci dans la variable I0 (type Word 16 Bits).

Ligne 2 : Lecture dans l'esclave d'adresse **4** de **10** mots commençant à l'adresse **1000** en décimal (100) et stockage de ceux-ci dans le tableau de nom Tablo (tablo[1] à tablo[10]).

Ligne 3 : Lecture dans l'esclave d'adresse **4** d'un bit à l'adresse 10 et stockage de celui-ci dans la variable B0 (de type Boolean).

Ligne 4 : Ecriture dans l'esclave d'adresse **5** de 10 bits à l'adresse 11 avec les valeurs des variables TabloBB[1] à TabloBB[10] (de type Boolean).

Ligne 5 : Lecture dans l'esclave d'adresse **11** de **4** mots commençant à l'adresse **$300** en hexa et stockage de ceux-ci dans les variables W0, W1, W2 et W3 (ces variables étant déclarées consécutivement, les mots recus sont aussi stockés consécutivement dans celles-ci).

Ligne 6 : Lecture de mots dans l'esclave d'adresse contenue dans la variable NUM de **X** mots (valeur de la variable MW3) commençant à l'adresse contenue dans la variable MW0 et stockage de ceux-ci dans les variables MW50 (ces variables étant déclarées consécutivement, les mots reçus sont aussi stockés consécutivement dans celles-ci). **La trame est lancée si le Bit B50 est à True**. Lorsque la communication est effectuée correctement ou time out, **la variable B50 est remise à zéro par le serveur**, permettant ainsi d'effectuer un accusé de réception. Testez alors StatusModbus[X] pour connaître le résultat de la communication.

Rappel : Vous pourrez tester ou forcer les bits des mots lus et écrits (ex : Carte Entrée ou carte sortie) W0.0, O0.0 I0.0 etc..

Sauvez le paramétrage puis lancez le serveur de communication Modbus ainsi que la Visudynamique des variables.

Autres informations

Le time out sur une communication est de 1 seconde.
Autrement dit si l'équipement modbus n'a pas répondu avant une seconde, le serveur annule la transaction en cours avec celui-ci.

Lorsqu'une trame est conditionnée par un **bit de commande**, celle-ci n'est active que si ce bit à True.
Si la transaction avec l'équipement c'est correctement est effectuée, le serveur positionne StatusModbus[X] à TRUE sinon à FALSE.
Cependant dans tout les cas le Bit de commande est remis à False (soit ComOk ou TimeOut).

StatusModbus est un tableau de bit indiquant le bon déroulement des trames modbus paramètrées.
Ex : StatusModbus[1] passe à true quand la commande modbus paramètrée sur la ligne 1 dans le configurateur est passée correctement et que l'esclave y à répondu positivement.

Le configurateur Modbus ajoute automatiquement les variables :

ComModbusOK : Boolean ;

Bit passant à True si la communication c'est bien initialisée (ouverture du port série avec la vitesse demandée).
Si ce bit reste à False, le port de communication est inexistant ou est déjà occupé par une application.

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

Chaque cellule du tableau reflète l'état de la communication correspondante.

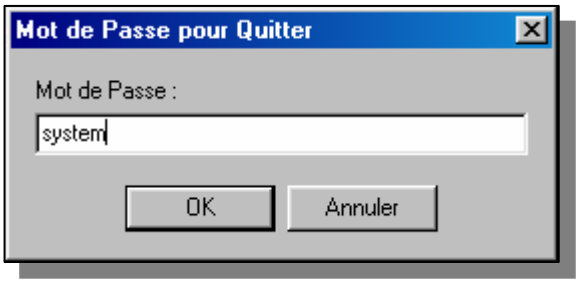
Si vous avez paramétré 3 trames de communication,
Le statut de la première trame se trouve dans StatusModbus[1] le bit passe à True si la communication c'est correctement déroulée avec l'esclave concerné.
Le statut de la deuxième trame se trouve dans StatusModbus[2] le bit passe à True si la communication c'est correctement déroulée avec l'esclave concerné.
Le statut de la troisième trame se trouve dans StatusModbus[3] le bit passe à True si la communication c'est correctement déroulée avec l'esclave concerné.

Etc..

Lancement du serveur de communication

Le serveur au lancement vient se positionner dans la barre de tache de windows indiquant ainsi sa présence.

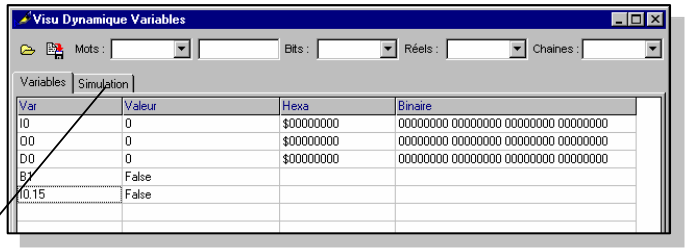
En double cliquant sur cette icône, une fenêtre apparaît permettant d'arrêter celui-ci. En cliquant sur le bouton Quit, une fenêtre demande un mot de passe apparaît :



Frappez le mot de passe **system** pour arrêter le serveur de communication

Visualisation dynamique des Variables du serveur

Ce programme permet d'explorer les variables déclarées évoluant dans le serveur de communication.
Une visualisation dynamique et un forçage en numérique, Hexadécimal et binaire est possible.
Cette liste peut être sauvegardée sur disque et sous un nom donné puis rappelé à tout moment. Permettant ainsi de constituer plusieurs pages de variables.



Simulateur

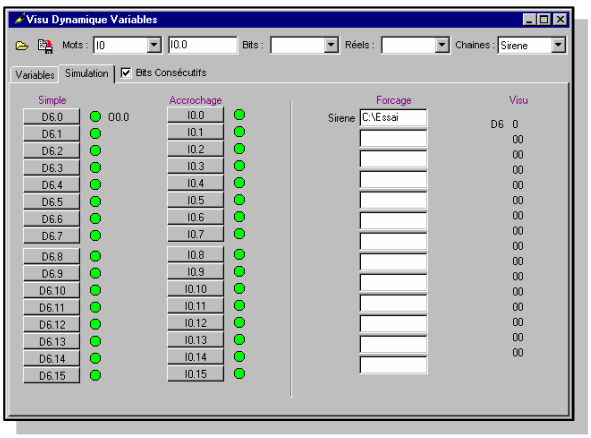
Un simulateur permet d'affecter à des boutons poussoirs simples ou à accrochages des variables booléennes.
Pour affecter un bouton ou un voyant, amenez par Drag & Drop du champ de saisie le nom de la variable concernée sur celui-ci.
Si la case à cocher 'Bits Consécutifs' est valide, les boutons suivants prennent automatiquement les noms des variables adjacentes.

Pour effacer l'affectation d'une variable à un bouton, effacer le texte dans le champ de saisie à droite du ComboBox Mots, puis amenez par drag & drop le texte vide sur le bouton, voyant ou autre élément de visualisation.

Note :
Pour effectuer un '**Drag & Drop**' Cliquez sur le texte avec le bouton gauche de la souris sans relâchez celui-ci, puis faites glissez le curseur de la souris sur l'élément cible souhaité et relâchez le bouton sur celui-ci.

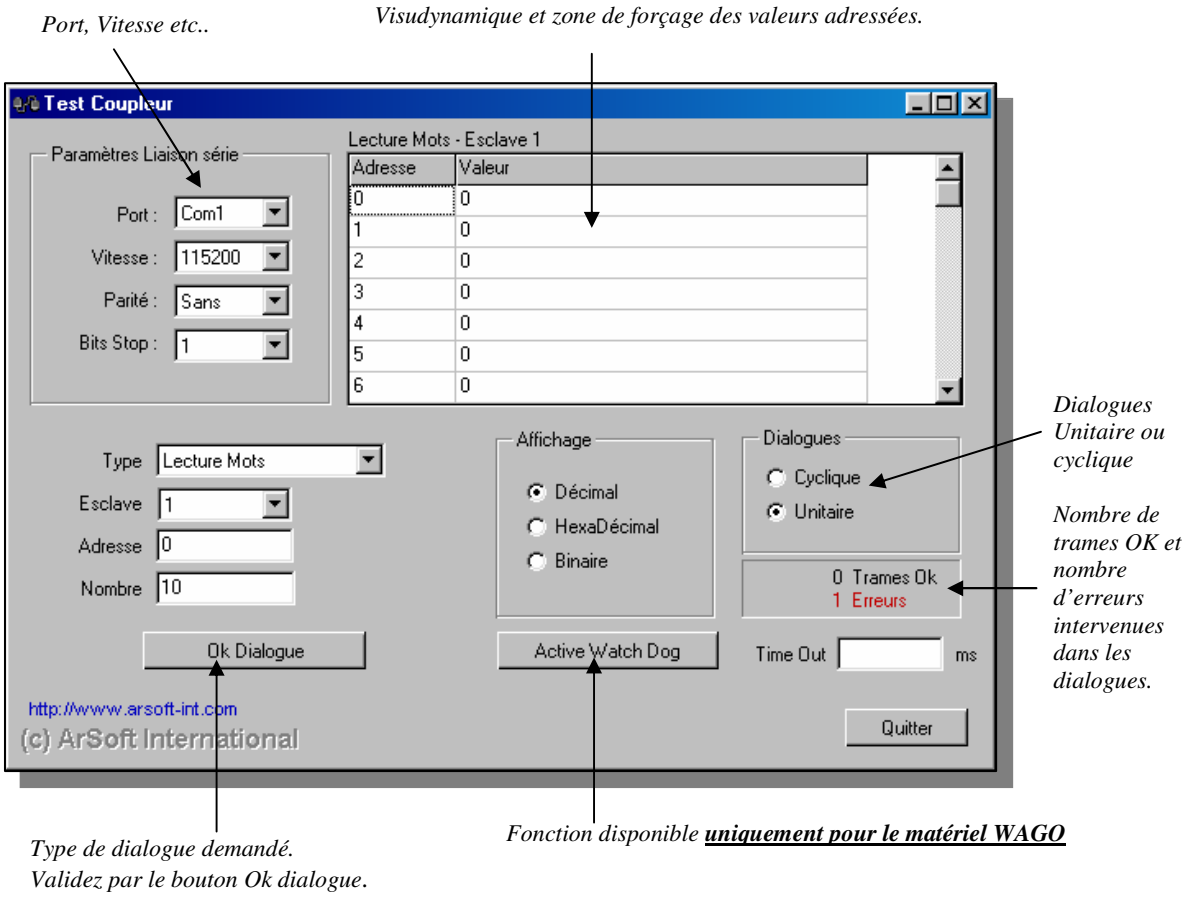
Dans la colonne **Forçage**, des champs de saisie sont disponibles pour affecter directement des variables numériques et chaînes de caractères.
Dans la colonne **visu**, une simple visualisation de variables est possible sans forçage de celles-ci.

Les Combobox supérieurs, permettent de rappeler le nom des variables globales ainsi que le mécanisme de Drag & Drop.



Utilitaire de test de la liaison Modbus.

Cet utilitaire permet de générer des trames unitaires ou cycliques sur un port de communication, afin de tester l'équipement connecté et de visualiser toutes les zones de celui-ci.



Programmation

Le produit est livré avec 2 DLL : **VPC.DLL et VPLCOM.DLL**

VPLC.DLL est une DLL classique permettant l'interfacage avec différents langages évolués comme Delphi ou Visual C++.
Cependant du fait du passage dans les procédures et fonctions d'une String (Pointer sur caractères), il est impossible pour des langages comme Visual Basic de passer directement une chaîne. Les chaînes Visual Basic étant au format UNICODE incompatible avec les pointeurs sur chaînes de caractères.

il est alors possible de résoudre ce problème en utilisant VPLCOM.DLL qui est un objet COM et qui reçoit les chaînes UNICODE.

Pour Visual Basic Utilisez l'objet COM (VPLCOM.DLL) pour l'appel aux procédures et fonctions décrites ci-dessous. Cet objet est recensé automatiquement dans votre base de registres à l'installation du produit.

Pour une utilisation sous forme de DLL :
****
!! Attention respectez les minuscules dans les déclarations concernant le nom de la dll vplc.dll
!! Respectez aussi l'écriture (majuscule & minuscule) dans les noms de procédures et fonctions
****
Pour l'utilisation sous forme d'objet COM:
Les méthodes ainsi que leurs paramètres sont identiques.
Le respect des majuscules/Minuscules n'est pas nécessaire.

Prototypes et commentaires :

Function GetVpuInt (Name : PChar) : Integer;
-> récupère la valeur de la variable "Name" de type Integer ;

Procedure SetVpuInt (Name : PChar; Value : Integer);
-> affecte la valeur "Value" à la variable "Name" de type Integer ;

Function GetVpuWord (Name : PChar) : Word;
-> récupère la valeur de la variable "Name" de type Word ;

Procedure SetVpuWord (Name : PChar; Value : Word);
-> affecte la valeur "Value" à la variable "Name" de type Word ;

Function GetVpuBool (Name : PChar) : Boolean;
➔ récupère la valeur de la variable "Name" de type Boolean ;
➔ Possibilité de récupérer un bit d'un mot ex : GetVpuBool('W0.0')

Procedure SetVpuBool (Name : PChar; State : Boolean);
-> affecte la valeur "Value" à la variable "Name" de type Boolean ;

Function GetVpuByte (Name : PChar) : Integer;
-> récupère la valeur de la variable "Name" de type Byte ;

Procedure SetVpuByte (Name : PChar; Value : Byte);
-> affecte la valeur "Value" à la variable "Name" de type Byte ;

Function GetVpuString(Name : PChar) : PChar;
-> récupère la valeur de la variable "Name" de type String ;

Procedure SetVpuString(Name : PChar; Value : String);
-> affecte la valeur "Value" à la variable "Name" de type String ;

Function GetVpuReal (Name : PChar) : Single ;
-> récupère la valeur de la variable "Name" de type Real ;
!!Attention la valeur retournée est stockée sur 4 octêts
son étendue est limité par rapport aux variables réelles de Visual PLC stockées
sur 10 octêts (Extended).

Procedure SetVpuReal (Name : PChar; Value : Single);
-> affecte la valeur "Value" à la variable "Name" de type Real ;
!!Attention la valeur envoyée limite l'étendue par rapport aux variables réelles de Visual PLC stockées sur 10 octêts (Extended).

{**** Accès au Tableaux **********}
Function GetVpuAInt (Name : PChar; Element : Integer) : Integer;
-> récupère la valeur d'un tableau "Name" à la position "Element" de type Integer ;

Procedure SetVpuAInt (Name : PChar; Element : Integer; Value : Integer);
-> affecte la valeur "Value" à la position "Element" du tableau "Name" de type Integer ;

Function GetVpuAWord (Name : PChar; Element : Integer) : Word;
-> récupère la valeur d'un tableau "Name" à la position "Element" de type Word ;

Procedure SetVpuAWord (Name : PChar; Element : Integer; Value : Word);
-> affecte la valeur "Value" à la position "Element" du tableau "Name" de type Word ;

Function GetVpuABool (Name : PChar; Element : Integer) : Boolean;
-> récupère la valeur d'un tableau "Name" à la position "Element" de type Boolean ;

Procedure SetVpuABool (Name : PChar; Element : Integer; Value : Boolean);
-> affecte la valeur "Value" à la position "Element" du tableau "Name" de type Boolean ;

Function GetVpuAByte (Name : PChar; Element : Integer) : Integer;
-> récupère la valeur d'un tableau "Name" à la position "Element" de type Byte ;

Procedure SetVpuAByte (Name : PChar; Element : Integer; Value : Byte);
-> affecte la valeur "Value" à la position "Element" du tableau "Name" de type Byte ;

Function GetVpuAString(Name : PChar; Element : Integer) : PChar;
-> récupère la valeur d'un tableau "Name" à la position "Element" de type String ;

Procedure SetVpuAString(Name : PChar; Element : Integer; Value : PChar);
-> affecte la valeur "Value" à la position "Element" du tableau "Name" de type String ;

Function GetVpuAReal (Name : PChar; Element : Integer) : Single;
-> récupère la valeur d'un tableau "Name" à la position "Element" de type Real ;

Procedure SetVpuAReal (Name : PChar; Element : Integer; Value : Single);
-> affecte la valeur "Value" à la position "Element" du tableau "Name" de type Integer ;

{*** Accès à un Mot quel que soit son Type *****}
Function GetVpuValue (Name : PChar) : PChar;
-> récupère la valeur de la variable "Name" quel que soit son type et la renvoie sous forme de chaîne de caractère (PChar);

{*** Fixe une valeur à n'importe quel variable *****}
Procedure SetVpuValue (Nom,Value : String);
Exemple : SetVpuValue ('Compteur','10');
Exemple : SetVpuValue ('Etat[1]','True');

{*** Fixe le chemin contenant le Fichier MVarglob.VPU *********}
Procedure SetPathVarglob(Path : String); StdCall;
Exemple : SetPathVarglob('C:\Essai1\'); Attention mettre '\' à la fin

{**** Récupère l'adresse du premier mot spécifié *********}
Function GetVpuAddr (Nom : String) : Pointer;
Exemple : Adresse :=GetVpuAddr (Compteur) ; renvoie l'adresse mémoire de la variable Compteur

Procedure ClearOptimisation;
-> A chaque appel d'une fonction ou d'une procédure, l'adresse de la variable "Name" est stockée dans une table temporaire pour augmenter les performances d'accès aux variables de Visual PLC. Au bout d'un certain temps, cette table est plus importante que la table réelle des variables de Visual PLC ; il est donc nécessaire de réinitialiser la table temporaire à l'aide de cette procédure. Par exemple, utilisez cette fonction à chaque changement de page.

Utilisation sous forme de DLL

Programmation en Delphi :
déclarez tous les prototypes de fonctions et de procédures en var

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

Utilisez les alors comme des procédures normales dans votre programme.

Implémentation en Visual Basic

Impossible passez par l'objet Automation VPLCOM.DLL

Implémentation en Visual C++ et C++Builder.

Utiliser les prototypes des fonctions de la DLL Vplc.dll ou utilisez un objet Com. Voir exemple fourni en Visual Basic.

Implémentation en Delphi et Visual Pascal

déclarez tous les prototypes de fonctions et de procédures en var

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

Implémentation en C

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

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

Utilisez les alors comme des procédures normales dans votre programme

Utilisation de GetVpuAddr

Exemple :
Les mots déclarés dans les variables globales sont rangées consécutivement dans la mémoire du serveur. Il est possible de récupérer l'adresse du premier mot puis de lire ou d'écrire consécutivement des octêts

Exemple de variables globales déclarées.

```
Var
        Compteur        : Integer ;       {** 4 octets **}
        Etat            : Boolean         {**1 octet **}
        Eana            : Word ;          {** 2 octets **}
```

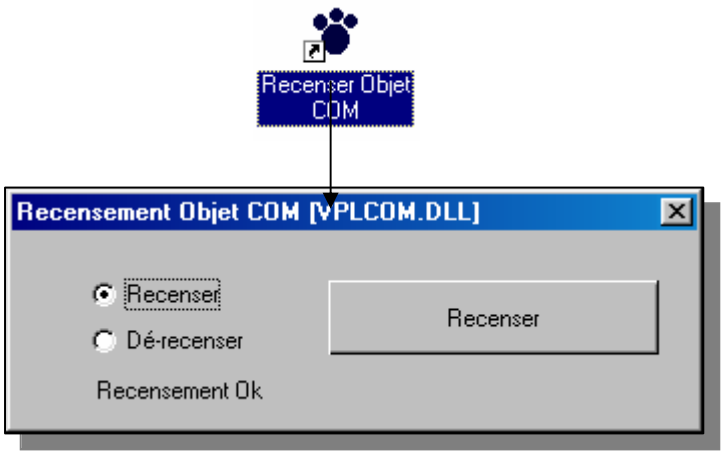
Exemple en Pascal (Visual Pascal ou Delphi)

```
Var
  Pt      : Pointer ;
  SaveCompteur: Integer ;
  SaveEtat      : Boolean ;
  SaveEana      : Word ;

Begin
 PT :=GetVpuAddr( Compteur) ;     {** récupère adresse premier mot dans le serveur ***}
 Move(PT,SaveCompteur,4+1+2) ;  {** Transfert mémoire serveur vers variables locales procédure **}
End ;
```

Utilisation sous forme d'objet COM

Le nom de l'objet est **VPLC.OBJ**

Avant de pouvoir utiliser l'objet Com (objet Automation) VPLCOM.OBJ, vous devez le recenser dans la base de registre de Windows. Cliquez sur l'icône recenser Objet Com



Choisissez le radio-bouton Recenser ou dé-recenser l'objet puis cliquez sur le bouton. Après un recensement correct, le texte *Recensement Ok* apparaît sous les 2 radio-boutons de gauche. Le recensement c'est alors correctement effectué. Quitter alors le programme l'objet COM est disponible.

Programmation en Visual Basic (voir aussi exemple complet fourni sur CD)

```
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 en Delphi et Visual Pascal (voir aussi exemple complet fourni sur CD)
Exemple :

```
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;
```