

XWProtocol User Manual

Revision A

Catalog No. 8U0124



Copyright © 2004 by YET, Yaskawa Eshed Technology Ltd.

XWProtocol User Manual

Cat. No. 8U0124

August 2004

All rights reserved. No part of this publication may be stored in a retrieval system, or reproduced in any way, including but not limited to photocopy, photography, magnetic or other recording, without the prior agreement and written permission of the publisher. Program listings may be entered, stored and executed in a computer system, but not reproduced for publication.

This manual is designed to provide information about the XWProtocol. Every effort has been made to make this book complete and as accurate as possible. However, no warranty of suitability, purpose or fitness is made or implied. YET Ltd. is not liable or responsible to any person or entity for loss or damage in connection with or stemming from the use of **XWProtocol** and/or the information contained in this publication

YET Ltd. bears no responsibility for errors, which may appear in this publication and retains the right to make changes to the software and manual without prior notice.

MAIN OFFICE:

13 Hamelacha St.,

Afeq Industrial Estate

Rosh Ha'ayin 48091

ISRAEL

Tel: +972-3-9004114

Fax: +972-3-9030412

E-mail: info@yetmotion.com

Homepage: <http://www.yetmotion.com/>

USA OFFICE:

YET US Inc.

531 King St.,

Unit 1

Littleton, MA 01460

USA

Tel: +1-866-YET-8080

E-mail: USinfo@yetmotion.com

Homepage: <http://www.yetmotion.com/>

Table of Contents

1. Introduction to XWProtocol	5
1.1. What is XWProtocol?.....	5
1.2. Installing XWProtocol.....	6
1.2.1. System Requirements	6
1.2.2. Installing the XWProtocol Files.....	7
1.3. Uninstalling XWProtocol Control	10
2. Building XWProtocol Applications With Various Languages	11
2.1. Developing Visual Basic Applications	11
2.1.1. Loading XWProtocol into the VB Toolbox	11
2.1.2. Adding XWProtocol to Your Project	13
2.1.3. Using XWProtocol's Methods and Properties.....	14
2.1.4. Using the Object Browser	15
2.2. Developing Visual C++ Applications.....	16
2.2.1. Initializing the ActiveX(OLE) Libraries	16
2.2.2. Importing XWProtocol Using the #import Directive	16
2.2.3. Defining the Global Interface Pointer	17
2.2.4. Create Instance.....	17
3. Main Functions of XWProtocol.....	19
3.1. Generating a Message	19
3.2. Translating the Response Message from the XtraDrive.....	20
3.2.1. XWResponseHeaderData.....	20
3.2.2. Response Data	23
3.2.3. Examples of Using TranslatingMessageRespond	23
4. Operating the XtraDrive Using XWProtocol	25
4.1. Communication Settings	25
4.2. Polling Message	27
4.2.1. Sending the Polling Message	27
4.2.2. Analyzing the Polling Response Message	28
4.3. Parameter Control.....	29
4.3.1. Get Parameter	29
4.3.2. Set Parameter.....	32
4.4. Get Driver Information.....	34
4.4.1. Sending the GetDriverInfo Message.....	34
4.4.2. Analyzing the GetDriverInfo Answer From the Driver .	35
4.5. XtraDrive Programming	36
4.5.1. Downloading a Program.....	36
4.5.2. Uploading a Program.....	38
4.5.3. Parametric Commands	41

5. References	45
5.1. Enumerations	45
5.2. Properties	46
5.3. Structs	46
5.4. Methods	46
5.5. Commands.....	47
Appendix A: Protocol Review	53
Message Data Structure	53
Master Message	54
Response Message	56
Answer Field for Acknowledge (ACK)	56
Answer Field for Data Request Command	57

1. Introduction to XWProtocol

This chapter provides an overview of XWProtocol, lists the XWProtocol system requirements, and describes how to install the XWProtocol files.

Related documents:

TITLE	CATALOG NUMBER
XtraDrive (XD-) SERIES AC SERVO DRIVER User Manual	8U0108
XtraWare User Manual	8U0109

1.1. What is XWProtocol?

XWProtocol is an ActiveX Control that interprets and generates the XtraDrive protocol within any compatible ActiveX control container, such as Visual Basic or VC++.

Using XWProtocol, you can easily

- ◆ Generate a message for any XtraDrive command without needing to know the small details of XtraDrive protocol. Only "XtraDrive language" (commands) knowledge is needed
- ◆ Interpret the response messages received from the XtraDrive.

The communication treatment is not a part of the XWProtocol. The user must have communication configuration and programming knowledge to be able to send and receive XtraDrive messages via a Serial Com port.

1.2. Installing XWProtocol

This section provides detailed instructions for installing XWProtocol onto your computer.

1.2.1. System Requirements

For optimum performance, the XWProtocol requires:

- ◆ Computer: Pentium 166 MHz (Pentium II 350 MHz recommended).
- ◆ At least 32 MB of RAM (64 MB recommended).
- ◆ A hard drive with at least 100 MB of free disk space.
- ◆ Operating System:
 - Windows 98 (Service Pack 2 or later)
 - Windows 2000
 - Windows XP
- ◆ Super VGA or better graphics display, minimum 256 colors (65536 colors recommended).
- ◆ Serial Com port.
- ◆ CD-ROM drive (for installation only).

1.2.2. Installing the XWProtocol Files

The XWProtocol software is supplied on a CD. Before proceeding with the installation procedure, close any open applications. During the installation procedure, XWProtocol and its related files are installed on your hard disk.

► **To install the XWProtocol files:**

1. Insert the XWProtocol installation CD-ROM disc into the CD-ROM drive on your computer.
2. The installation program should start automatically. If auto-run is not enabled on your computer, use your Windows Explorer or the Windows Run command to execute setup.exe on the XWProtocol installation CD-ROM disc (assume "D" is the letter of your CD-ROM disc drive): D:\Install\setup.exe.

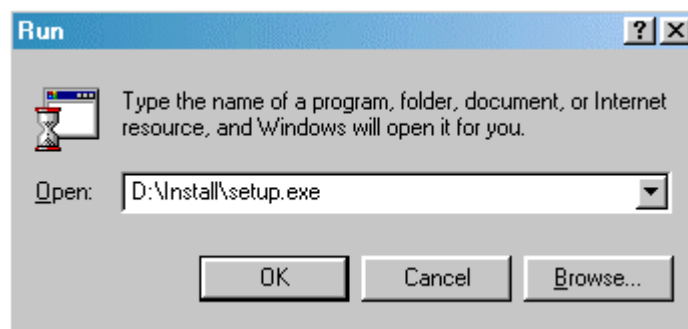


Figure 1: Run setup.exe From the Installation CD-ROM Disc

The program may take a couple of minutes to load. Follow the instructions in the installation wizard.

3. The XWProtocol installation program loads.

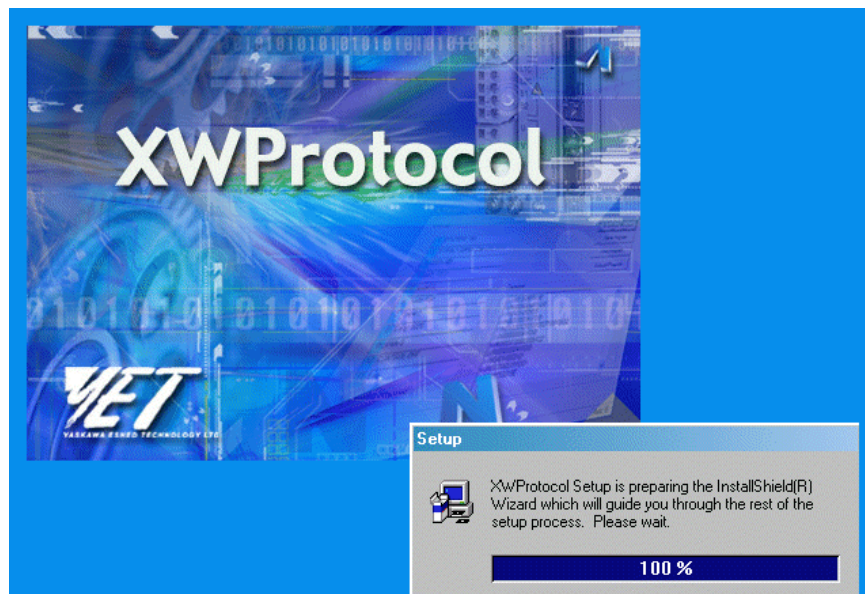


Figure 2: Loading the XWProtocol Setup Program

4. In the Choose Destination Location window, specify a file path on your local computer to install the XWProtocol files. The default path is:
C:\Program Files\YET\XWProtocol.

You can choose a different directory path by clicking the **Browse** button.

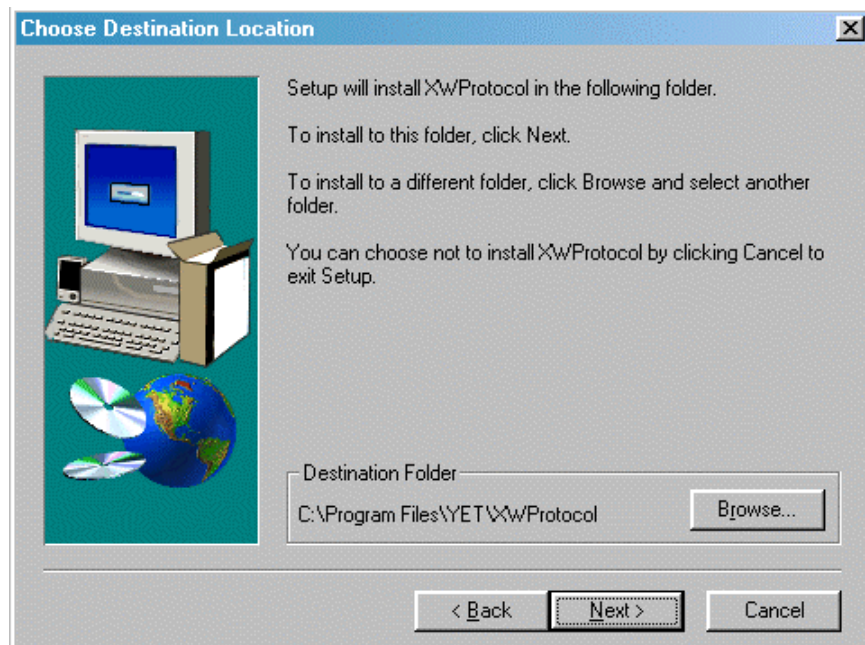


Figure 3: Choose Destination Directory for Program Files

5. Click **Next**. The files required to run XWProtocol will be installed in the folder you selected.

When the installation program finishes copying all the program files to your hard disk, the Setup Complete window will be displayed.

- Click **Finish** to close the installation program and restart your computer if necessary.

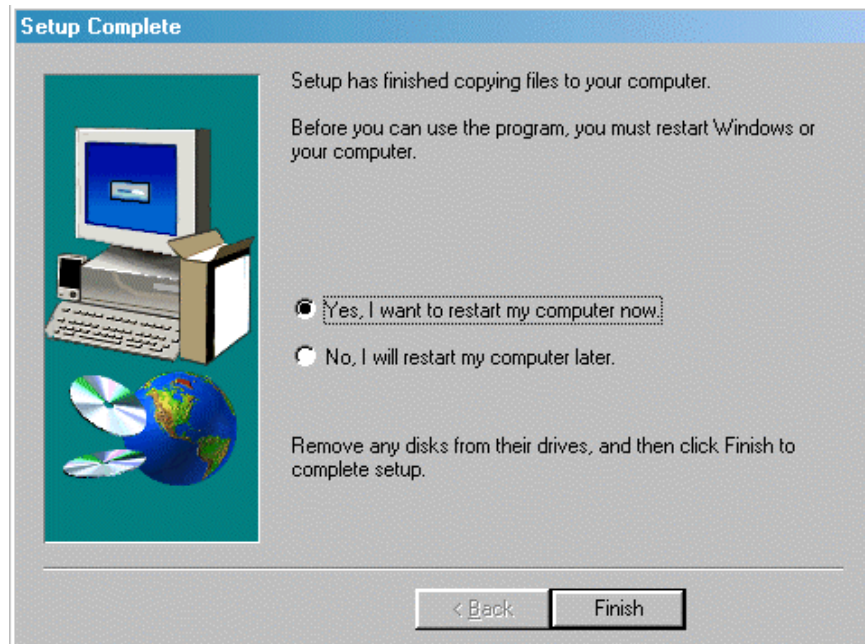


Figure 4: Installation Program Setup Complete Window

- The installation of the XWProtocol control is now finished and the installation program will close.

You will notice that the XWProtocol program shortcuts have been copied to your Windows Start menu so that you can easily launch the program and view the related documentation.

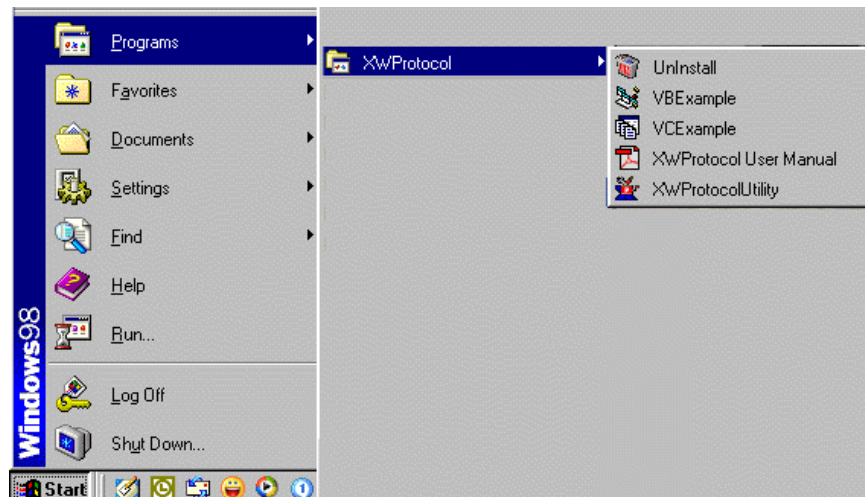


Figure 5: XWProtocol Shortcuts on the Windows Start Menu

1.3. Uninstalling XWProtocol Control

XWProtocol includes an uninstall utility to remove the files from your computer.

► **To uninstall XWProtocol control and its related files:**

Either

1. From the Windows Start menu click **Programs | XWProtocol | UnInstall.**

or

1. Choose **Settings | Control Panel** from your Windows Start menu.
2. Double-click on the icon labeled "Add/Remove Programs".
3. Select the item labeled "XWProtocol" and click **Add/Remove.**

2. Building XWProtocol Applications With Various Languages

This chapter describes how can you use the XWProtocol control with the following development tools:

- ◆ Microsoft Visual Basic
- ◆ Microsoft Visual C++

This chapter assumes that you are familiar with the basic concepts of using Visual Basic and Visual C++, including selecting the type of application, designing the form, and placing the control on the form.

2.1. Developing Visual Basic Applications

The following subsections explain how to:

- ◆ Load XWProtocol into the VB toolbox.
- ◆ Add XWProtocol to your project.
- ◆ Use XWProtocol's methods and properties.
- ◆ Open the Object Browser.

2.1.1. Loading XWProtocol into the VB Toolbox

Before using the XWProtocol control to build an application, you need to add it to the Visual Basic toolbox.

► **To load XWProtocol to the Visual Basic toolbox:**

1. Select **Project | Components**.
2. In the Controls tab, scroll down to the XWProtocol control.
3. Select **XWProtocol ActiveX Control module**. If XWProtocol ActiveX Control module is not shown in the list, click **Browse** to select the desired controls from the \WINDOWS\SYSTEM(32) directory.

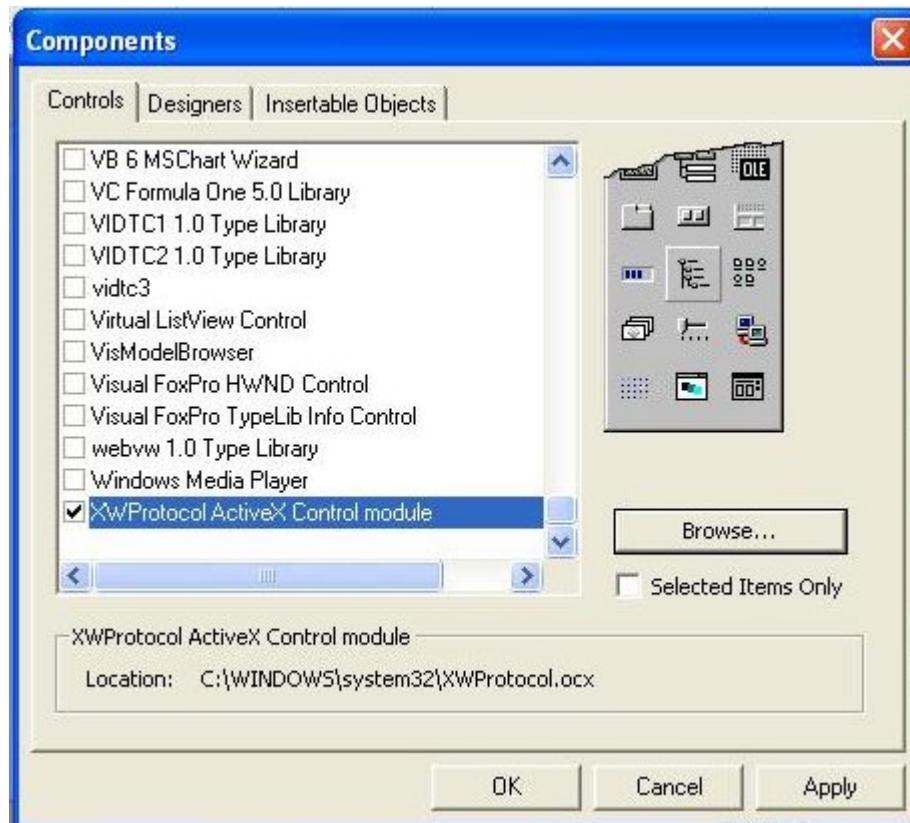


Figure 6: Visual Basic Components dialog box showing the XWProtocol ActiveX Control module

2.1.2. Adding XWProtocol to Your Project

After you add the XWProtocol control to the Visual Basic toolbox, place the XWProtocol icon on a Visual Basic form.

► **To add XWProtocol to your project:**

1. Select the XWProtocol icon  from the toolbox and place it on the Visual Basic form.

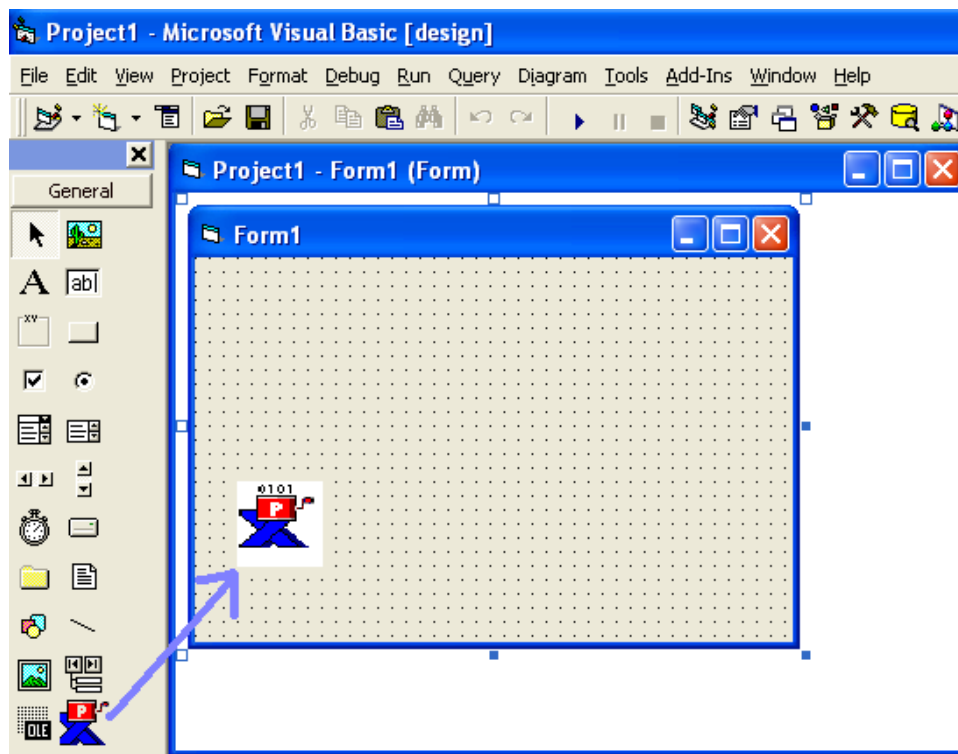


Figure 7: Placing the XWProtocol icon on a Visual Basic Form

2. Add the following code line to your Form Code:

```
Dim XW As New Protocol
```



NOTE:

Instead of performing the above procedure, you can add the following code to your Module Code:

```
Public XW As New XWPROTOCOLLib.Protocol
```

2.1.3. Using XWProtocol's Methods and Properties

To call a method or get/set a property of the XWProtocol control, add the name of the method or property after the name of the control.

The following figure shows how the method and property list is shown.

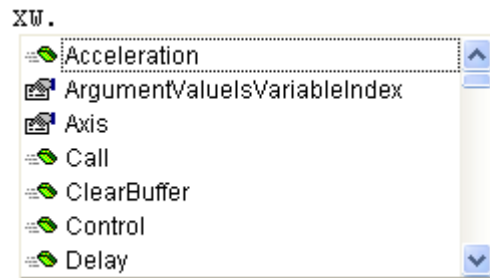


Figure 8: The Method and Property List



NOTE:

Each XtraDrive command has a corresponding XWProtocol method.

2.1.4. Using the Object Browser

The VB Object Browser helps you create Visual Basic code. It can display a simple description of XWProtocol's available properties and methods.

To open the Object Browser, select **View | Object Browser**.

An example of XWProtocol's properties and methods in the Visual Basic Object Browser is shown below.

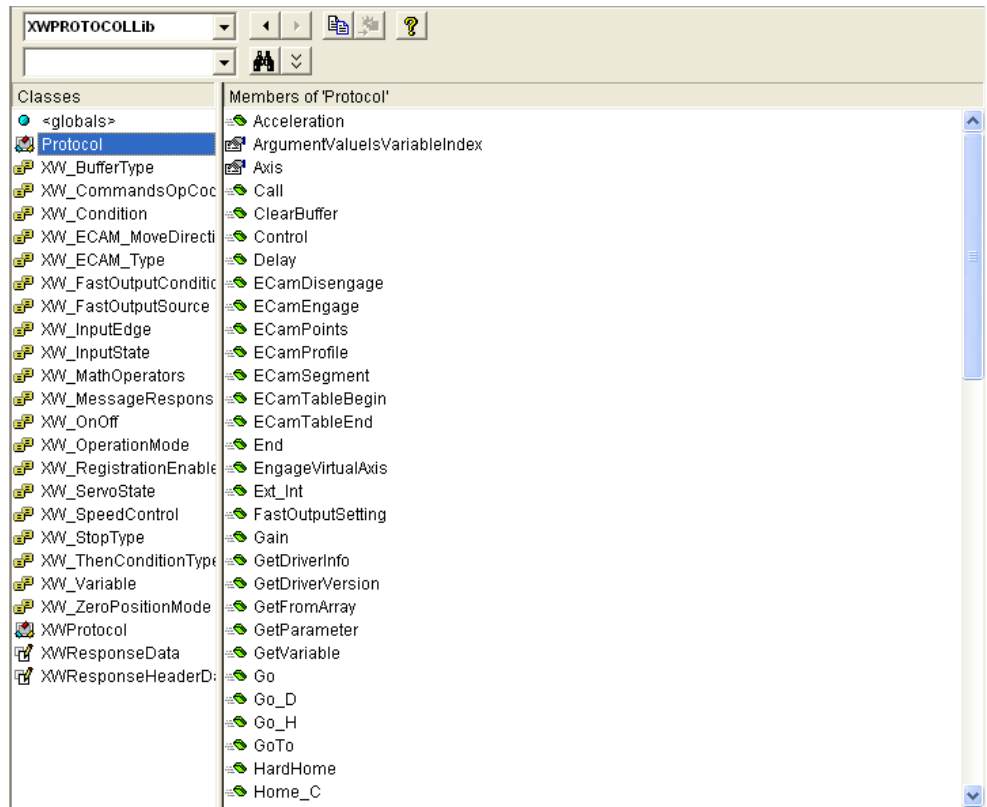


Figure 9: XWProtocol in the Visual Basic Object Browser

2.2. Developing Visual C++ Applications

To create a Visual C++ application working with XWProtocol, you can use the Visual C++ MFC AppWizard to create a skeleton project and program. After you create a project with ActiveX control support, use the following procedure to develop an application using the XWProtocol control. A dialog-based application is used below to illustrate the procedure.

2.2.1. Initializing the ActiveX(OLE) Libraries

Before using XWProtocol control to build an application, you must initialize the ActiveX(OLE) libraries in the `InitInstance` overridable function `CMyApp::InitInstance()`.

To initialize the ActiveX(OLE) Libraries, enter the following code:

```
if ( !AfxOleInit() )
{
    AfxMessageBox ("OLE Initialization failed");
    return FALSE;
}
```

2.2.2. Importing XWProtocol Using the `#import` Directive

The `#import` directive is used to incorporate information from XWProtocol. The content of XWProtocol is converted into C++ classes describing the interfaces.

Import XWProtocol.ocx in the StdAfx.h header file:

```
#import "XWProtocol.ocx"
using namespace XWPROTOCOLLib;
```

The `import` directive creates two header files (XWProtocol.tlh and XWProtocol.tli) that reconstruct the type library contents in C++ source code.

You can find the XWProtocol.tlh file and the XWProtocol.tli files by navigating to your build output directory (probably `\Debug`). These are the C++ header files that the compiler created from the XWProtocol at build time.

2.2.3. Defining the Global Interface Pointer

To define the global interface pointer to `Iprotocol` enter the following code line:

```
IProtocolPtr pProtocolPtr = NULL;
```

**NOTE:**

The `IProtocolPtr` is a smart pointer generated automatically by the `#import` directive.

2.2.4. Create Instance

In order to create an instance of the `XWProtocol`, call the `CreateInstance` function of the `IProtocolPtr` class and pass the GUID for the `XWProtocol Protocol` class (using the `__uuidof` keyword).

For example, add the following code in `OnInitDialog()`:

```
HRESULT hr = pProtocolPtr.CreateInstance(__uuidof( Protocol) );
if( hr != S_OK )
{
    AfxMessageBox( "Cannot initialize Protocol object!" );
    return FALSE;
}
```


3. Main Functions of XWProtocol

This chapter explains the main functions of XWProtocol: generating a message and translating the response message from the XtraDrive.

3.1. Generating a Message

XWProtocol enables you to generate a message for any XtraDrive command.

► **To generate a message:**

1. Set the axis.
2. Set the message ID.
3. Choose an operation mode.
4. Choose the method for the specific command and assign the argument values.



NOTE:

The message string ends with Carriage Return (CR- ASCII Code 0x13).

Sample VB Code For the Move Command

```
XW.Axis = 0
XW.MessageID = 0
XW.ModeOfOperation = OpSequential
Dim Message As String
Dim MoveDistance as long
Dim MotionTime as long
MoveDistance=1000
MotionTime=100
Message = XW.Move(MoveDistance, MotionTime)
```

Sample C++ Code For the Move Command

```
pProtocolPtr->Axis=0;
pProtocolPtr->MessageID=0;
pProtocolPtr->ModeOfOperation=opSequential;
bstr_t Message=pProtocolPtr->Move(100,50);
CString strMess((char*)Message);
```

3.2. Translating the Response Message from the XtraDrive

The XWProtocol enables you to translate the response message from the XtraDrive.

For translating the response message, use the `TranslateMessageRespond` function.

The input for the function `TranslateMessageRespond` is the response message. The function returns the translation results in the `XWResponseHeaderData` structure and the `ResponseData` array of the `XWResponseData` structure.

3.2.1. XWResponseHeaderData

`XWResponseHeaderData` is a structure that includes relevant information about the response message received from the driver.

Members of 'XWResponseHeaderData'

-  `AxisNumber`
-  `ChecksumError`
-  `ErrorOpCode`
-  `ErrorString`
-  `MessageID`
-  `MessageOpCode`
-  `MessageType`
-  `ParameterNumber`
-  `ProgramCmdOpCode`
-  `s_AlarmON`
-  `s_EmergencyOFF`
-  `s_InternalLimitActive`
-  `s_NeedRestart`
-  `s_OperationModeSpecific_Bit12`
-  `s_OperationModeSpecific_Bit13`
-  `s_ProgramRun`
-  `s_QuickStop`
-  `s_ReadyForStart`
-  `s_ServoON`
-  `s_TargetReached`
-  `s_Warning`
-  `StatusWordStringBits`
-  `StringMessage`
-  `VariableOpCode`

Figure 10 : Members of XWResponseHeaderData in Visual Basic



NOTE:

Not all of the data members are relevant to every response message.

The `XWResponseHeaderData` members are the following:

- ◆ `Axis Number (Byte)`: Represents the axis number.
- ◆ `ChecksumError (Boolean)`: Indicates if there was a checksum error in the response message.
- ◆ `ErrorOpCode (Byte)`: When an error occurs, indicates the `OpCode` of an error.
- ◆ `ErrorString (String)`: The error description.
- ◆ `MessageID (Byte)`: When an error occurs, indicates the ID of the command message or program line that causes the error. In the case of a program upload/download, the message ID is the line number of the program.
- ◆ `MessageOpCode (Byte)`: Indicates the response message `OpCode`.
- ◆ `MessageType (XW_MessageResponseType)`: Indicates the type of the response message. The options are:
 - `rtCommand`-The response message of the Get Parameter, Get Variable, Get Driver Version and Get Driver Information answers.
 - `RtErrorAndInfo` - The response message of an error.
 - `RtOk` - The response message is acknowledged (ACK).
 - `RtProgramUpload` - The response message for a program upload.
 - `RtWatchVariable` - The response message for a variable watch.
- ◆ `ParameterNumber (Integer)`: The parameter number in decimal form. This member is relevant only for the Get Parameter answer.
- ◆ `ProgramCmdOpCode (XW_CommandsOpCode)`: Indicates the `OpCode` of the uploaded command line. Relevant only in the case of a program upload.
- ◆ `StatusWordStringBits (String)`: Indicates the XtraDrive status word. Relevant only in `rtOK` and `rtErrorAndInfo` messages.
- ◆ `s_AlarmON, s_EmergencyOFF, s_InternalLimitActive, s_NeedRestart, s_OperationModeSpecificBit12, s_OperationModeSpecificBit13, s_ProgramRun, s_QuickStop, s_ReadyForStart, s_ServoON, s_TargetReached, s_Warning (Boolean)`: Each `s_` member represents a specific bit in the XtraDrive status word.
- ◆ `StringMessage (String)`: Contains the XtraDrive string answer. Relevant when you get a response from the Get Driver Information command.

- ◆ `VariableOpCode` (`XW_Variable`): The variable OpCode in decimal form. This member is relevant only for the Get Variable answer.

**NOTE:**

A list of OpCode errors, OpCode commands, and OpCode variables can be found in the *XtraWare User Manual*.

3.2.2. Response Data

`ResponseData` is an array of the `XWResponseData` structure. The array size is change based on the response type.

3.2.2.1. XWResponseData

The `XWResponseData` members are:

- ◆ `Value (Long)`: The value that can be a parameter value, a variable value or an argument of a command.
- ◆ `VariableOpCode (Boolean)`: Indicates whether the data in the `Value` member is actually a value or a variable `OpCode`. Used in case of a parametric command argument.

3.2.3. Examples of Using TranslatingMessageRespond

Sample VB Code

```
Dim ResponseHeader As XWResponseHeaderData
Dim ResponseData() As XWResponseData
Call XW.TranslateMessageRespond ("N050171000003E8FFFFFFFFFA2",
    ResponseHeader, ResponseData)
```

Sample C++ Code

```
XWResponseHeaderData RHD;
SAFEARRAY *pResponseData=NULL;
CString MessageResponse("N050171000003E8FFFFFFFFFA2");
pProtocolPtr->TranslateMessageRespond(
    MessageResponse.AllocSysString(), &RHD, &pResponseData)
```


4. Operating the XtraDrive Using XWProtocol

This chapter provides detailed instructions for operating the XtraDrive servo driver using the XWProtocol.

4.1. Communication Settings

The communication is not part of the XWProtocol; it is your responsibility to add a communication control. In this section we will use the MSComm Control, which provides Serial communication by allowing the transmission and reception of data through a serial port in order to establish communication between the XtraDrive and the PC.

► **To use the MSComm Control:**

1. Load the MSComm component into the VB toolbox.
2. Place the MSComm on the form.
3. Set the MSComm properties as shown in the following figure.

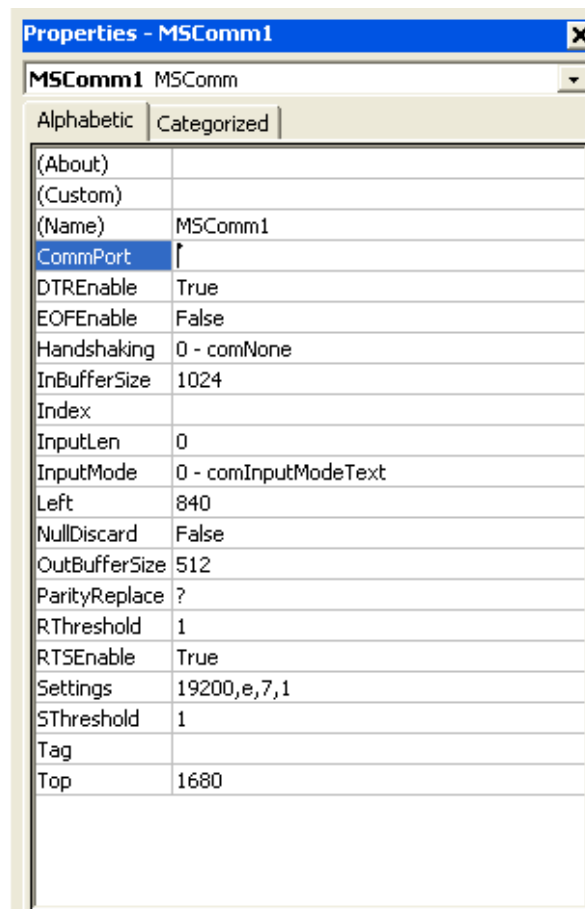


Figure 11: Setting the MSComm Properties in VB



NOTE:

Choose the relevant CommPort number according to your needs.

► **To use the MSComm for communication:**

1. Open the port.
2. Send the required message to the driver.
3. Wait for data to come back to the serial port. Each message ends with the carriage return symbol (CR).

Sample VB Code

Here is sample code for sending and receiving a message using the MSComm control:

```
Dim MessageRespond as String

Sub Send(Command As String)
    MSComm1.Output = Command
End Sub

Sub Receive()
    Dim Buffer As String
    MessageRespond = ""
    Do
        DoEvents
        Buffer = MSComm1.Input
        MessageRespond = MessageRespond + Buffer
    Loop Until InStr(MessageRespond, vbCr)
End Sub
```



NOTE:

The following sections use the Send and Receive functions in the VB sample codes that were defined here.

The Receive function assigns the response message in the MessageRespond global variable.

4.2. Polling Message

This section describes how to send the Polling message and analyze the Polling answer received from the driver.

4.2.1. Sending the Polling Message

► **Perform the following to send the Polling message:**

1. Set `Axis` to the relevant axis.
2. Set `MessageID` to 0.
3. Set `ModeOfOperation` to `opPolling`.
4. Use the `Polling` function to generate the polling message.
5. Send the message via communication port.

Sample VB Code

```
Sub SendnPolling ()  
    Dim Message As String  
    XW.Axis = 0  
    XW.MessageID = 0  
    XW.ModeOfOperation = opPolling  
    Message = XW.Polling  
    Call Send(Message)  
End Sub
```

4.2.2. Analyzing the Polling Response Message

► **After receiving the response message, perform the following:**

1. Translate the message using the method
`TranslateMessageRespond.`
2. Verify that
`XWResponseHeaderData.MessageTypeMessage = rtOK`
OR
`XWResponseHeaderData.MessageTypeMessage = rtErrorAndInfo.`
3. Check the following:
 - All members of `XWResponseHeaderData` with the prefix `'s_'` (for example `XWResponseHeaderData.s_ServoON`)
 - `XWResponseHeaderData.StatusWordStringBits`
 - If `MessageTypeMessage = rtErrorAndInfo`, the following members are also relevant:
`XWResponseHeaderData.ErrorOpCode`
`XWResponseHeaderData.ErrorString`

Sample VB Code

```
Dim ResponseHeader As XWResponseHeaderData
Dim ResponseData() As XWResponseData
Dim ServoON as Boolean
Dim EmergencyOFF as Boolean

Call XW.TranslateMessageRespond (MessageRespond,
                                ResponseHeader, ResponseData)

Select Case ResponseHeader.MessageType
    Case rtOk, rtErrorAndInfo:
        ServoON=ResponseHeader.s_ServoON
        EmergencyOFF=ResponseHeader.s_EmergencyOFF
    Case rtCommand:
    Case rtProgramUpload:
End Select
```



NOTE:

More information about the status word can be found in the *XtraWare User Manual*.

4.3. Parameter Control

This section explains how to use the Get Parameter and Set Parameter commands.

4.3.1. Get Parameter

The Get Parameter command is used to read the value of a parameter from the XtraDrive.

4.3.1.1. Sending the Get Parameter Command

► **Perform the following to send the Get Parameter command:**

1. Set `Axis` to the relevant axis.
2. Set `MessageID` as required.
3. Set `ModeOfOperation` as required.
4. Use the `GetParameter` function to generate the Get Parameter message string. The argument is your requested parameter number.
5. Send the message string via communication port.

Sample VB Code

```
Dim Message As String
Dim ParNumber As Long
XW.Axis = 0
XW.MessageID = 1
XW.ModeOfOperation = opImmediate
ParNumber = &H0 'Parameter Number000
Message = XWP.GetParameter(ParNumber)
Send(Message)
```

4.3.1.2. Analyzing the GetParameter Answer from the Driver

- **After receiving the response message, perform the following:**
 1. Translate the message using the method `translateMessageRespond`.
 2. If `XWResponseHeaderData.MessageTypeMessage` is not `rtCommand`, send a Polling Message until it is `rtCommand`.
 3. Check that `XWResponseHeaderData.MessageOpCode` is `opcGET_PAR`.
 4. Verify that `XWResponseHeaderData.ParameterNumber` is your requested parameter. The parameter value is in `ResponseData(1).Value`.



NOTE:

The index of the first element in the array depends on the array definition.

Sample VB Code

```
Sub ReceiveGetParResponse()
    Dim Count As Integer
    Dim Message As String
    While ReceiveResponse() <> rtCommand
        XW.ModeOfOperation = opPolling
        Message = XWP.Polling
        Call Send(Message)
        Count = Count + 1
    Wend
End Sub

Private Function ReceiveResponse() As Integer
    Dim ResponseHeader As XWResponseHeaderData
    Dim ResponseData() As XWResponseData
    Call Receive()
    Call XW.TranslateMessageRespond (MessageRespond
        ResponseHeader, ResponseData)
    Select Case ResponseHeader.MessageType
    Case rtOk:
        ReceiveResponse = rtOk
    Case rtErrorAndInfo:
        ReceiveResponse = rtErrorAndInfo
    Case rtCommand:
        ReceiveResponse = rtCommand
    End Select
End Function
```

```
Select Case ResponseHeader.MessageOpCode
Case opcGET_PAR:
    Dim strParNumber As String
    Dim ParNum As Long
    ParNum = ResponseHeader.ParameterNumber
    strParNumber = Hex(ParNum)
End Select
End Select
Dim Index As Integer
Dim ParValue As Long
For Index= 1 To UBound(ResponseData)
    ParValue = ResponseData(iParNum).Value
Next iParNum
End Function
```

4.3.2. Set Parameter

The Set Parameter command is used to set the value of an XtraDrive parameter.

4.3.2.1. Sending the Set Parameter Command

► **Perform the following to send the Set Parameter command:**

1. Set `Axis` to the relevant axis.
2. Set `MessageID` as required.
3. Set `ModeOfOperation` as required.
4. Use the `SetParameter` function to generate the Set Parameter message string. The arguments are your requested parameter number and the value.
5. Send the string via the communication port.

Sample VB Code

```
Dim Message As String
Dim ParValue As Long
Dim ParNumber As Long
XW.Axis = 0
XW.MessageID = 1
XW.ModeOfOperation = opImmediate
ParNumber = 100 'Pn100
ParValue = 40
Message = XW.SetParameter(ParNumber, ParValue)
Call Send(Message)
```

4.3.2.2. Analyzing the Set Parameter Answer from the Driver

- **After receiving the response message, perform the following:**
 1. Translate the message using the method `translateMessageRespond`.
 2. Verify that `XWResponseHeaderData.MessageTypeMessage = rtOK`.

Sample VB Code

```
Dim ResponseHeader As XWResponseHeaderData
Dim ResponseData() As XWResponseData
Call Receive()
Call XW.TranslateMessageRespond (MessageRespond,
    ResponseHeader, ResponseData)
Select Case ResponseHeader.MessageType
Case rtOk:
    'The Parameter was updated ok
Case rtErrorAndInfo:
    'There is an error
End Select
End Function
```

4.4. Get Driver Information

This section describes how to send the Get Driver Information message and analyze the Get Driver Information answer received from the driver.

4.4.1. Sending the GetDriverInfo Message

► **Perform the following to send the GetDriverInfo message:**

1. Set `Axis` to the relevant axis.
2. Set `MessageID` as required.
3. Set `ModeOfOperation` as required.
4. Use the `GetDriverInfo` function to generate the `GetDriverInfo` message.
5. Send the message via communication port.

Sample VB Code

```
Dim Message As String
XW.Axis = 0
XW.MessageID = 0
XW.ModeOfOperation = opImmediate
Message = XW.GetDriverInfo
Call Send (Message)
```

4.4.2. Analyzing the GetDriverInfo Answer From the Driver

► **After receiving the response message, perform the following:**

1. Translate the message using the method `translateMessageRespond`.
2. If `XWResponseHeaderData.MessageTypeMessage` is not `rtCommand`, send a Polling Message until it is `rtCommand`.
3. Check that `XWResponseHeaderData.MessageOpCode = opcGET_DRIVER_INFO`.

The relevant data is

`XWResponseHeaderData.StringMessage`.

Sample VB Code

```
Dim ResponseHeader As XWResponseHeaderData
Dim ResponseData() As XWResponseData
Dim DriverInfo as String
Call Receive()
Call XW.TranslateMessageRespond (MessageRespond
    ResponseHeader, ResponseData)
Select Case ResponseHeader.MessageOpCode
    Case rtCommand:
        Select Case ResponseHeader.MessageOpCode
            Case opcGET_VAR:
            Case opcGET_PAR:
            Case opcGET_DRIVER_INFO:
                DriverInfo = ResponseHeader.StringMessage
        End Select
    Case rtOk
End Select
```

4.5. XtraDrive Programming

This section explains how to download a program to the driver and upload a program from the driver.

4.5.1. Downloading a Program

► **Perform the following to download a program to the driver:**

1. Clear the program stored in the driver using the `ClearBuffer(UPB)` function.
2. Set `Axis` to the relevant axis.
3. Set `MessageID` to the program line number you want to download. The first line of the program is 1.
4. Set `ModeOfOperation` to `opProgramDownload`.
5. For each line you want to download:
 - 5.1. Use the function corresponding to the program command line you want to download to generate the message string.
 - 5.2. Increase the message ID for the next program line by 1.
 - 5.3. Send the message via the communication port.
6. Set `ModeOfOperation` to `opImmediate`.
7. Use the function `SavePrgEcam` to save the program in the driver.

Sample VB Code

This sample shows how to download the following program:

Line Number	Command Line
1	Control On
2	End

```
Private Sub DownloadProgram()
    Dim msgID As Integer
    Dim message As String
    msgID = 1
    XW.Axis = 0
    'clear buffer
    XW.ModeOfOperation = opImmediate
    message = XW.ClearBuffer(UPB)
    Call Send(message)
```

```
XW.ModeOfOperation = opProgramDownload
    'line #1
XW.MessageID = msgID
message = XWP.Control(xwServoON)
Call Send(message)
    'line #2
msgID = msgID + 1
XW.MessageID = msgID
message = XWP.End()
Call Send(message)
    'save program
XW.ModeOfOperation = opImmediate
message = XW. SavePrgEcam ()
Call Send(message)
End Sub
```

4.5.2. Uploading a Program

4.5.2.1. Sending the Upload Instruction

- ▶ **Perform the following to send the Upload instruction:**
 1. Set `Axis` to the relevant axis.
 2. Set `MessageID` to the program line number you want to upload.
 3. Set the `ModeOfOperation` to `opProgramUpload`.
 4. To upload the program line, use the `Polling` function.
 5. Send the string via communication port.

4.5.2.2. Analyzing the Upload Answer from the Driver

- ▶ **After receiving the response message, perform the following:**
 1. Translate the message using the method `translateMessageRespond`.
 2. Check that `XWResponseHeaderData.MessageType` is `rtProgramUpload`.

Each answer is a program line.

- `ResponseHeader.MessageID` represents the program line number.
- `ResponseHeader.ProgramCmdOpCode` is the program line command `OpCode`.
- The `ResponseData` array contains the command's arguments.



NOTE:

Stop sending the polling message when you get
`ResponseHeader.ProgramCmdOpCode=0;`

4.5.2.3. Sample VB Code

```
Dim UploadComplete as Boolean
Private Sub UploadProgram()
    Dim msgID As Integer
    Dim Message As String
    UploadComplete=False
    XW.Axis = 0
```

```

msgID = 1
XW.ModeOfOperation = opProgramUpload
While Not UploadComplete
    XW.MessageID = msgID
    Message = XW.Polling
    Call Send(Message)
    msgID = msgID+1
    Call ReceiveUploadResponse()
Wend
End sub

Sub ReceiveUploadResponse()
    Dim ResponseHeader As XWResponseHeaderData
    Dim ResponseData() As XWResponseData
    Call Receive()
    XWP.TranslateMessageRespond (MessageRespond, ResponseHeader,
        ResponseData)
    Dim line As Integer
    If ResponseHeader.MessageType = rtProgramUpload Then
        If ResponseHeader.ProgramCmdOpCode=0 then
            UploadComplete=True
            Exit Sub
        End if
        line = ResponseHeader.MessageID
        Select Case line
            Case 1:
                If ResponseHeader.ProgramCmdOpCode = opcCONTROL
                    And ResponseData(1).Value = xwServoON then
                    MsgBox "Line #1 is: Control On"
                End If
            Case 2:
                If ResponseHeader.ProgramCmdOpCode = opcEND Then
                    MsgBox "Line #2 is: End"
                End If
            End Select
        End If
    End Sub

```

4.5.2.4. Sample C++ Code

```
void CProtocolDemoDlg::OnTranslateResponse()
{
    XWResponseHeaderData RHD;
    SAFEARRAY *pResponseData=NULL;

    if(pProtocolPtr->TranslateMessageRespond(
        m_ResponseMessage.AllocSysString(),
        &RHD,&pResponseData))

        return;
    switch(RHD.MessageType)
    {
        case rtOk: break;
        case rtError: break;
        case rtCommand: break;
        case rtProgramUpload:
        {
            PVOID pvData;
            if(S_OK==SafeArrayAccessData(pResponseData, &pvData))
            {
                char CommandOpCode= RHD.ProgramCmdOpCode
                //Get the command argument data from the SAFEARRAY
                XWResponseData * pRD;
                pRD = (XWResponseData *)pvData;
                for(int iParNum=0;iParNum<
                    (int)pResponseData->rgsabound->cElements;iParNum++)
                {
                    if(pRD->VariableOpCode)
                    {
                        //Parametric argument
                    }
                    else
                    {
                        //the argument is value
                    }
                    pRD++;
                }
                SafeArrayUnaccessData(pResponseData);
            }
        }
    }
}
```

4.5.3. Parametric Commands

In some commands, instead of entering a number to specify the value of an argument you can set the argument equal to one of the variables. The variables that can be selected depend on the command.

Use the `IsParametricCommandArgument` function to check if you can set the command argument to one of the variables. The `IsParametricCommandArgument` function gets the Command Opcode and the argument number.

4.5.3.1. Downloading a Program Line

This section explains how to download a program line when the command has a parametric argument.

► **To download a program line:**

1. Set `Axis` to the relevant axis.
2. Set `MessageID` to the line number you want to download.
3. Set `ModeOfOperation` to `opProgramDownload`.
4. Use the `IsParametricCommandArgument` function to verify if the argument of the program line command can be parametric.
5. Set `ArgumentValueIsVariableIndex (index)` to `True`. The `index` is the argument's position in the command.
6. Use the function corresponding to the program command line you want to download in order to generate the message string.
7. Send the message via the communication port.

Sample VB Code

This sample shows how to download the following program:

Line Number	Command Line
1	Move_D var_01,-1

```
sub DownloadParamtricLine()  
    Dim msgID as Integer  
    Dim message as String  
    MsgID=1  
    XW.MessageID = msgID  
    If XW.IsParametricCommandArgument(opcMOVE_D,0) Then  
        XW.ArgumentValueIsVariableIndex(0) = True  
        message = XW.Move_D(XW_Variable.Var_01, -1)
```

```

        Call Send(message)
    End If
End Sub

```

4.5.3.2. Uploading a Program Line

This section explains how to upload a program line when one of the commands has a parametric argument. The program line that will be uploaded is: `Move_D var_01,-1`.

To upload a program line, use the same procedure as sending the upload instruction. For more information, refer to section 4.5.2.1. *Sending the Upload Instruction*.

► **After receiving the response message, perform the following:**

1. Translate the message using the method `translateMessageRespond`.
2. Check that `XWResponseHeaderData.MessageType` is `rtProgramUpload`.

Each answer is a program line.

- `ResponseHeader.MessageID` represents the program line number.
- `ResponseHeader.ProgramCmdOpCode` is the program line command Opcode.
- The `ResponseData` array contains the command's arguments.
- Use `ResponseData().VariableOpCode` to know if the argument is parametric.

Sample VB Code

```

Dim ResponseHeader As XWResponseHeaderData
Dim ResponseData() As XWResponseData
Call Receive()
Call XW.TranslateMessageRespond (MessageRespond,
    ResponseHeader, ResponseData)
Dim line As Integer
If ResponseHeader.MessageType = rtProgramUpload Then
    line = ResponseHeader.MessageID
    Select Case line
        Case 1:
            If ResponseHeader.ProgramCmdOpCode = opcMOVE_D Then
                If ResponseData(1).VariableOpCode then
                    ' the argument is parametric
                    ' ResponseData(1).Value is the Var_01 OpCode

```

```
        EndIf
    End If
    Case 2:
    End Select
End If
```


5. References

5.1. Enumerations

ENUMERATION
XW_BufferType
XW_CommandsOpCode
XW_Condition
XW_ECAM_MoveDirection
XW_ECAM_Type
XW_FastOutputCondition
XW_FastOutputSource
XW_InputEdge
XW_InputState
XW_MathOperators
XW_MathSetOperation
XW_MessageResponseType
XW_OnOff
XW_OperationMode
XW_RegistrationEnableCondition
XW_ServoState
XW_SpeedControl
XW_StopServoState
XW_StopType
XW_ThenConditionType
XW_Variable
XW_ZeroPositionMode

5.2. Properties

PROPERTY	SYNTAX
Axis	HRESULT Axis([in,range(0,15)] unsigned char newVal)
	HRESULT Axis([out, retval] unsigned char * pVal)
MessageID	HRESULT MessageID([in] unsigned char newVal)
	HRESULT MessageID([out, retval] unsigned char * pVal)
ModeOf Operation	HRESULT ModeOfOperation([in] XW_OperationMode newVal)
	HRESULT ModeOfOperation([out, retval] XW_OperationMode * pVal)
ArgumentValue IsVariableIndex	HRESULT ArgumentValueIsVariableIndex([in] long index, [in] VARIANT_BOOL newVal)
	HRESULT ArgumentValueIsVariableIndex([in] long index, [out, retval] VARIANT_BOOL * pVal)

5.3. Structs

STRUCT	DESCRIPTION
XWResponseData	The data of the translation answer
XWResponseHeaderData	The header of the translation answer

5.4. Methods

METHOD	SYNTAX
MessageCheck Sum	HRESULT MessageChecksum([in] BSTR Message, [out, retval] unsigned char * pVal)
Translate Message Respond	HRESULT TranslateMessageRespond([in] BSTR Message, [out] XWResponseHeaderData * pResponseHeaderData, [out] SAFEARRAY(XWResponseData) * pResponseData)
IsParametric Command Argument	HRESULT IsParametricCommandArgument([in] XW_CommandsOpCode Code, ([in] Byte index [out, retval] VARIANT_BOOL * pVal)

5.5. Commands


NOTE:

You can find more information about each command in the *Command Reference* section in the *XtraWare User Manual*.

COMMAND	SYNTAX
Acceleration	HRESULT Acceleration([in]long Acceleration, [out, retval] BSTR* Message)
Call	HRESULT Call([in]unsigned char Label, [out, retval] BSTR* Message)
Clear Buffer	HRESULT ClearBuffer([in]XW_BufferType Buffer, [out, retval] BSTR* Message)
Control	HRESULT Control([in] XW_ServoState Servo, [out, retval] BSTR* Message)
Delay	HRESULT Delay([in] long Time, [out, retval] BSTR* Message)
Ecam Disengage	HRESULT EcamDisengage([out, retval] BSTR* Message)
Ecam Engage	HRESULT EcamEngage([in] unsigned char ProfileID, [in] XW_ECAM_Type Mode, [out, retval] BSTR* Message)
Ecam Points	HRESULT EcamPoints([in] unsigned char NumOfPoints, [in] short DeltaPos1, [in] short DeltaPos2, [in] short DeltaPos3, [in] short DeltaPos4, [out, retval] BSTR* Message)
Ecam Profile	HRESULT EcamProfile([in] unsigned char ProfileID, [out, retval] BSTR* Message)
Ecam Segment	HRESULT EcamSegment([in] long DeltaMaster, [in] long MasterStep, [in] unsigned char SegmentType, [out, retval] BSTR* Message)
EcamTable Begin	HRESULT EcamTableBegin([out, retval] BSTR* Message)
EcamTable End	HRESULT EcamTableEnd([out, retval] BSTR* Message)
End	HRESULT End([out, retval] BSTR* Message)

COMMAND	SYNTAX
Engage Virtual Axis	HRESULT EngageVirtualAxis([in] unsigned char ProfileID, [in] XW_ECAM_MoveDirection Direction, [out, retval] BSTR* Message)
Ext_Int	HRESULT Ext_Int([in] unsigned char IntPriority, [in] unsigned char InputNumber, [in] XW_InputEdge Edge, [out, retval] BSTR* Message)
FastOutput Setting	XW_FastOutputCondition Condition, [in] long Value, [out, retval] BSTR* Message)
Gain	HRESULT Gain([in] short Gain, [out, retval] BSTR* Message)
GetDriver Version	HRESULT GetDriverVersion([out, retval] BSTR* Message)
GetFrom Array	HRESULT GetFromArray([in] short ArrayIndex, [out, retval] BSTR* Message)
Get Parameter	HRESULT GetParameter([in] short ParNumber, [out, retval] BSTR* Message)
Get Variable	HRESULT GetVariable([in] XW_Variable VarIndex, [out, retval] BSTR* Message)
Go	HRESULT Go([in] long Target, [in] long MotionTime, [out, retval] BSTR* Message)
Go_D	HRESULT Go_D([in] long Target, [in] long MotionTime, [out, retval] BSTR* Message)
Go_H	HRESULT Go_H([in] long Target, [out, retval] BSTR* Message)
GoTo	HRESULT GoTo([in] unsigned char LabelNumber, [out, retval] BSTR* Message)
HardHome	HRESULT HardHome([in] short Torque, [in] long Speed, [out, retval] BSTR* Message)
Home_C	HRESULT Home_C([in] long Speed, [out, retval] BSTR* Message)
Home_SW	HRESULT Home_SW([in] long SpeedToSwitch, [in] long ReturnSpeed, [out, retval] BSTR* Message)
Home_SW_C	HRESULT Home_SW_C([in] long SpeedToSwitch, [in] long SpeedToCpulse, [out, retval] BSTR* Message)

COMMAND	SYNTAX
IF	HRESULT IF([in] XW_Variable VarIndex, [in] XW_Condition Condition, [in]long Value, [in] XW_ThenConditionType Then, [in]unsigned char LabelNumber, [out, retval] BSTR* Message)
IF_INPUT	HRESULT IF_INPUT([in]unsigned char VarIndex, [in]XW_InputState InputState, [in] XW_ThenConditionType Then, [in]unsigned char LabelNumber, [out, retval] BSTR* Message)
INPUT_CASE	HRESULT INPUT_CASE([in]long InputsMask, [in]long InputsState, [out, retval] BSTR* Message)
Int	HRESULT Int([in]unsigned char Priority, [in] XW_Variable VarIndex, [in]XW_Condition Condition, [in]long Value, [out, retval] BSTR* Message)
IntReturn	HRESULT IntReturn([in]unsigned char LabelNumber, [out, retval] BSTR* Message)
JerkTime	HRESULT JerkTime([in]long Time, [out, retval] BSTR* Message)
Label	HRESULT Label([in]unsigned char LabelNumber, [out, retval] BSTR* Message)
Latching Trigger	HRESULT LatchingTrigger([in]XW_RegistrationEnableCondition RegistrationEnableCondition, [out, retval] BSTR* Message)
LOOP	HRESULT LOOP([in]unsigned char LoopLevel, [in]long NumberOfCycles, [in]unsigned char LabelNumber, [out, retval] BSTR* Message)
Math	HRESULT Math([in]XW_Variable SetVarIndex, [in]XW_Variable VarIndex, [in] XW_MathOperators Operators, [in]long Value, [out, retval] BSTR* Message)
Move	HRESULT Move([in]long distance, [in]long Time, [out, retval] BSTR* message)
Move_D	HRESULT Move_D([in]long distance, [in]long Time, [out, retval] BSTR* Message)
Move_H	HRESULT Move_H([in]long Distance, [out, retval] BSTR* Message)
Move_R	HRESULT Move_R([in]long Distance, [out, retval] BSTR* Message)

COMMAND	SYNTAX
Polling	HRESULT Polling([out, retval] BSTR* Message)
ReadFrom Array	HRESULT ReadFromArray([in] short ArrayIndex, [in]unsigned char VarIndex, [out, retval] BSTR* Message)
Registration Distance	HRESULT RegistrationDistance([in] long Distance, [out, retval] BSTR* Message)
Return	HRESULT Return([out, retval] BSTR* Message)
Run	HRESULT Run([in]unsigned char LabelNumber, [out, retval] BSTR* Message)
SavePrgECam	HRESULT SavePrgECam([out, retval] BSTR* Message)
SetOutput	HRESULT SetOutput([in]unsigned char OutputNumber, [in]XW_OnOff OutputState, [out, retval] BSTR* Message)
SetOutputs	HRESULT SetOutputs([in]long OutputsMask, [in]long OutputsState, [out, retval] BSTR* Message)
Set Parameter	HRESULT SetParameter([in]short ParNumber, [in]long ParValue, [out, retval] BSTR* Message)
SetVariable	HRESULT SetVariable([in] XW_Variable VarIndex, [in]long VarValue, [out, retval] BSTR* Message)
SetZero Position	HRESULT SetZeroPosition([in]XW_ZeroPositionMode ZeroPositionMode, [out, retval] BSTR* Message)
Slide	HRESULT Slide([in]long Speed, [out, retval] BSTR* Message)
SlideAnalog	HRESULT SlideAnalog([out, retval] BSTR* Message)
Speed	HRESULT Speed([in]long Speed, [out, retval] BSTR* Message)
Speed Control	HRESULT SpeedControl([in]XW_SpeedControl Source, [out, retval] BSTR* Message)
Start	HRESULT Start([out, retval] BSTR* Message)
Stop	HRESULT Stop([in]XW_OnOff ServoState, [out, retval] BSTR* Message)
StopEX	HRESULT StopEX([in]XW_StopType StopType, [in]XW_OnOff ServoState, [out, retval] BSTR* Message)

COMMAND	SYNTAX
StopMotion	HRESULT StopMotion([out, retval] BSTR* Message)
Torque	HRESULT Torque([in] short Torque, [out, retval] BSTR* Message)
Torque Analog	HRESULT TorqueAnalog([out, retval] BSTR* Message)
Torque Limits	HRESULT TorqueLimits([in]short FWR, [in]short REV, [out, retval] BSTR* Message)
WaitExact	HRESULT WaitExact([in]long Time, [out, retval] BSTR* Message)
WaitForStart	HRESULT WaitForStart([out, retval] BSTR* Message)
WaitInput	HRESULT WaitInput([in]unsigned char InputNumber, [in]XW_InputState InputState, [in] long Time, [out, retval] BSTR* Message)
WaitStop	HRESULT WaitStop([in] long Time, [out, retval] BSTR* Message)
WaitVar	HRESULT WaitVar([in]XW_Variable VarIndex, [in]XW_Condition Condition, [in] long Time, [out, retval] BSTR* Message)
WriteTo Array	HRESULT WriteToArray([in] short ArrayIndex, [in]long Value, [out, retval] BSTR* Message)

Appendix A: Protocol Review

For your convenience, this appendix is an overview of some of the main points regarding XtraWare. For additional information on these topics, please refer to the *XtraWare User Manual*.

In this master/slave protocol, a PC (or other device) is the master and the XtraDrive is the slave. The master sends a request or a Polling message, and the XtraDrive answers with a response message. The master can only send a new message after receiving an answer or ACK (acknowledge) message, or after the timeout has expired.

The master can control up to 15 XtraDrive units using addresses. When broadcast messages are sent, the master does not wait for an ACK.

When there is no command to send, the master can continue sending polling messages; the XtraDrive responds with an ACK.

The diagram below illustrates the communication protocol between a PC (master) and a single XtraDrive.

PC Master				XtraDrive Slave
	1 st Message	→		
		←	Response	
	2 nd Message	→		
	:			
	:			
	:			
Timeout	3 rd message	→		
		←	Response	

Figure 1: Master-Slave Communication Protocol

Message Data Structure

- ◆ A message consists of bytes where each byte holds one digit of hexadecimal data in ASCII code representation.
- ◆ The data can be signed or unsigned according to the Command Operational Code argument type. For signed data the leftmost bit (msb) determines the sign.
- ◆ Negative number representation is according to standard hexadecimal representation and to the size of data.
- ◆ Each messages is a string structured according to one of the formats specified in the sections that follow, where each block in the format represents a byte.
- ◆ Every message in this protocol starts with "N" and terminates with CR (Carriage Return).

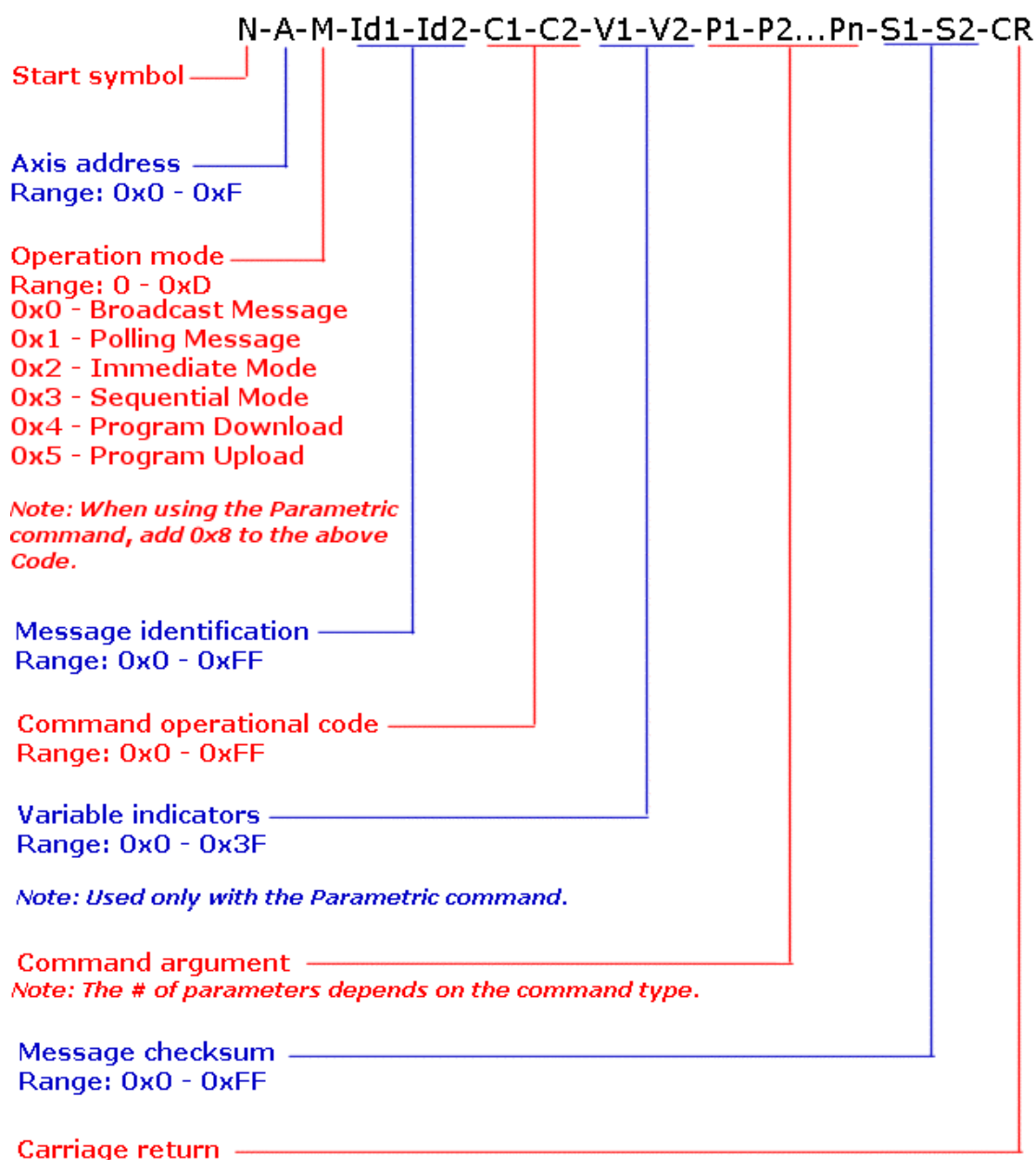


NOTE:

0x## represents a hexadecimal number.

Master Message

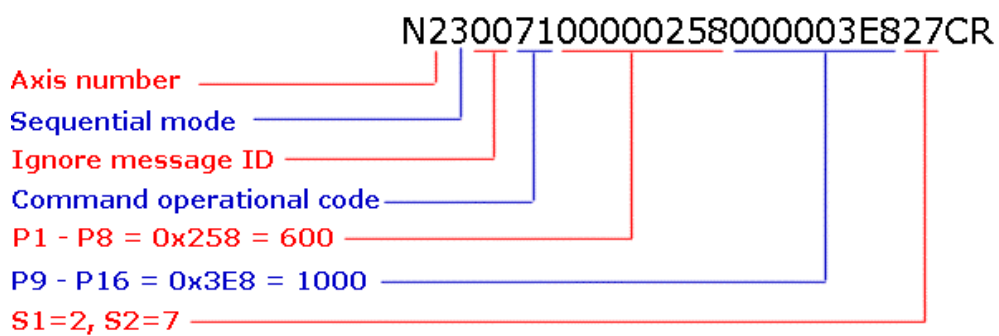
Format:



Example: MOVE Command

This is an example of the `MOVE` command (600uu in 100msec), of axis 2 in Sequential mode:

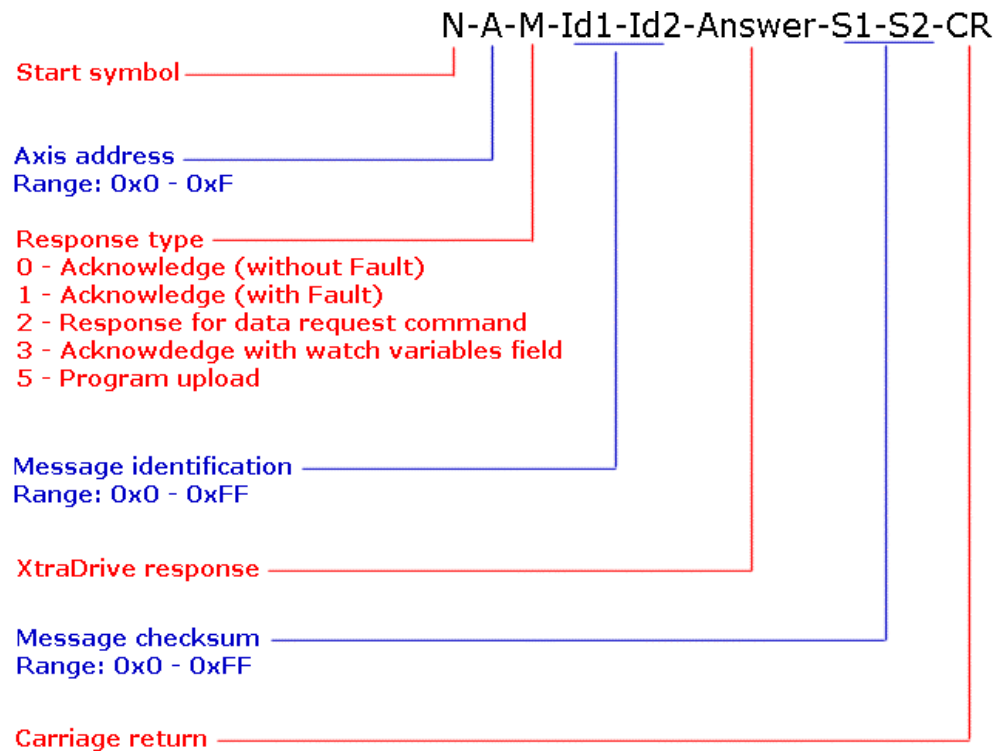
Format:



Response Message

All master messages, except broadcast messages, are responded by an XtraDrive response message.

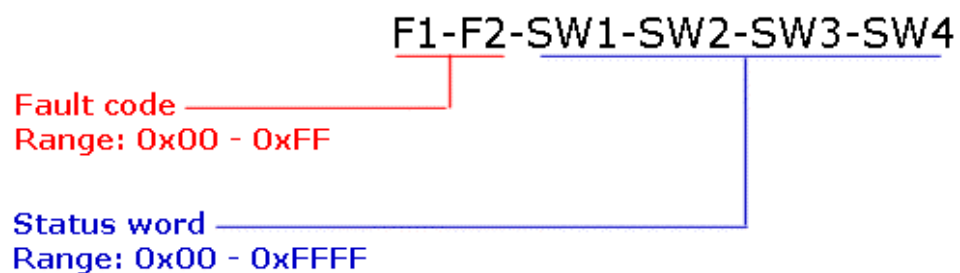
Format:



Answer Field for Acknowledge (ACK)

A response message is sent either in response to a Data Request command or as a response to other commands such as ACK. ACK accepts only when Response type m=0,1,3.

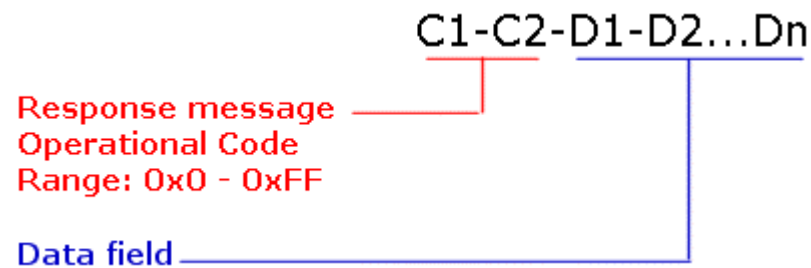
Format:



Answer Field for Data Request Command

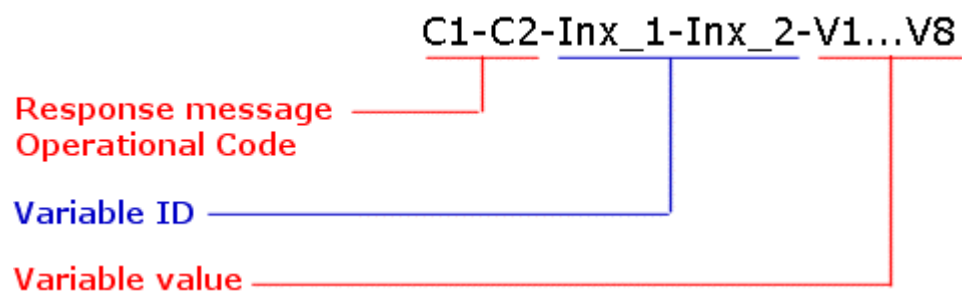
A response message is sent either in response to a Data Request command or as a response to the other commands such as ACK. The answer to a Data Request command is accepted only when Response type m=2 (0xA). The Answer format depends on the specific command.

Format:



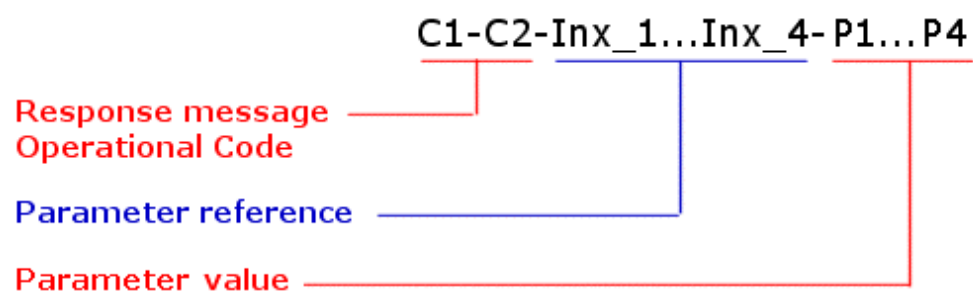
Answer Field for GET_VAR Command

Format:



Answer Field for GET_PAR Command

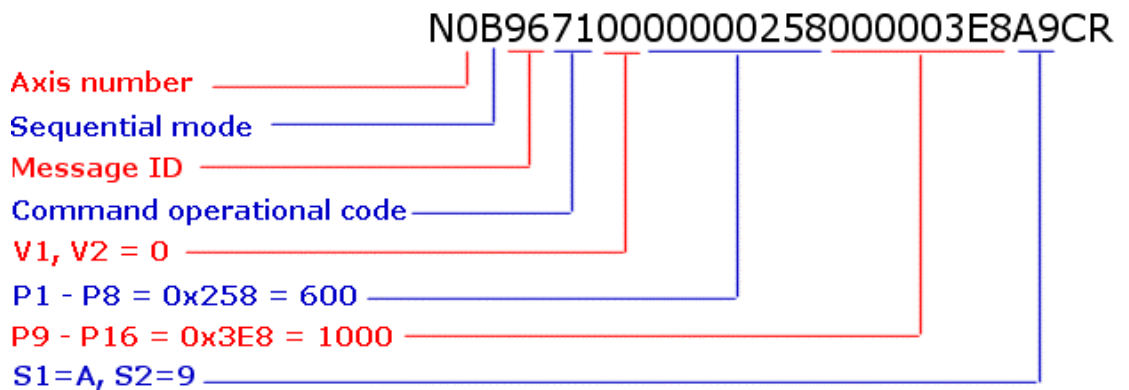
Format:



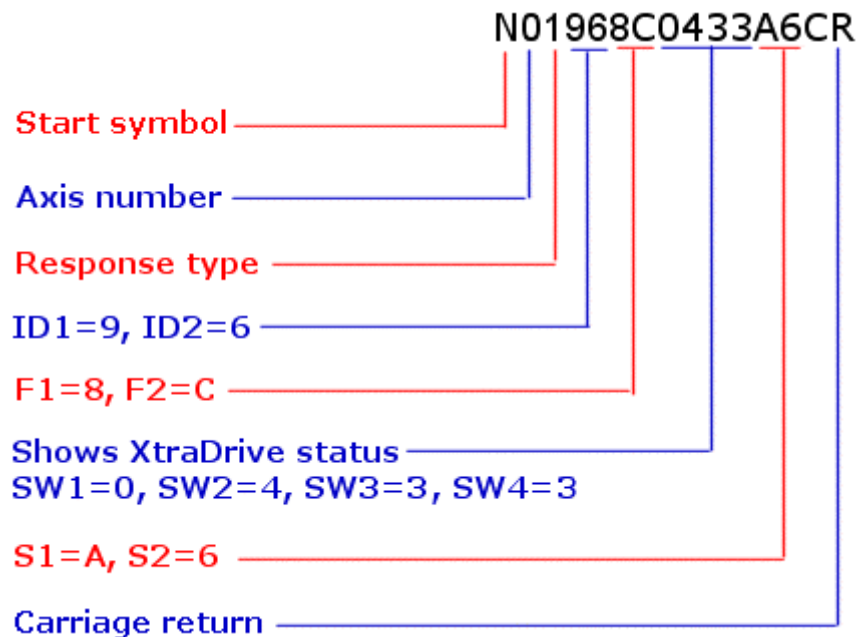
MOVE Command

This is an example of a response message to the
 MOVE <600> <1000> command (600uu in 1000ms) of axis 0 in
 Sequential mode, with message identification of 0x96 when the
 control is off. When the control is off, a motion cannot be
 executed. A 0x8C fault occurs.

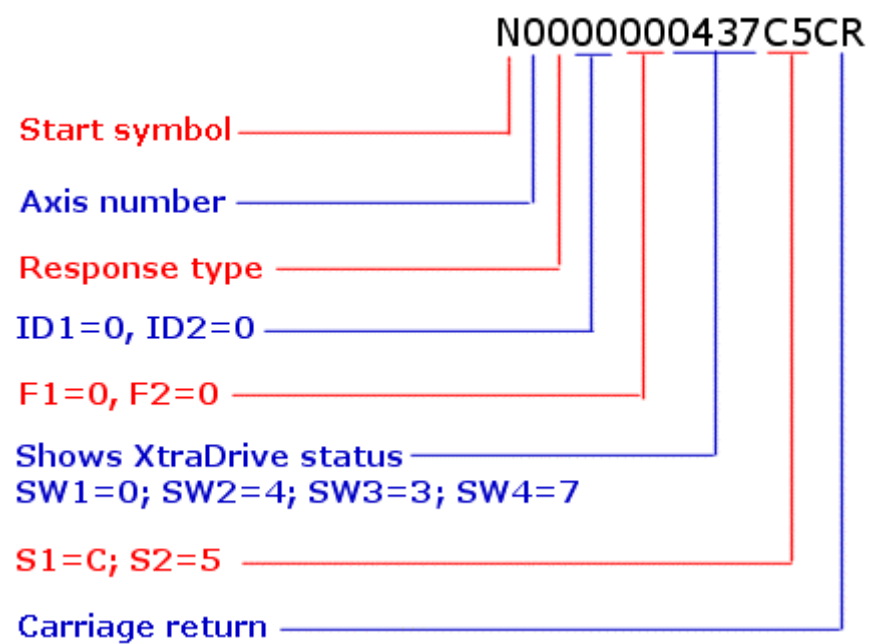
Master Message Format:



Response Message Format in case of fault:



Response message format in case of no fault:



GET_VAR Command

This is an example of a response message to the `GET_VAR` command using the variable `Position_Actual_value` (0x09) to axis 0 in Immediate mode with message identification of 0x7F.

Master Message Format:

Start symbol — NOA6548000940CR
Axis number —
Immediate mode —
Message ID —
Command operational code —
V1, V2 = 0 —
P1=0, P2=9 —
S1=4, S2=0 —
Carriage return —

Response message format in cases of no fault:

Start symbol — NO2654809FFFFFF060FACR
Axis number —
Immediate mode —
Message ID —
Response message operational code C1, C2 —
Variable Position_Actual_valueID
Inx_1=0, Inx_2=9
Variable Value V1-V8 —
S1=F, S2=A —
Carriage return —

MAIN OFFICE

13 Hamelacha St.,
Afeq Industrial Estate
Rosh Ha'ayin 48091
ISRAEL
Tel: +972-3-9004114
Fax: +972-3-9030412
info@yetmotion.com

USA OFFICE

YET US Inc.
531 King St.,
Unit 1
Littleton, MA 01460
USA
Tel: +1-866-YET-8080
USinfo@yetmotion.com

For more information refer to our web site:
www.yetmotion.com



Specifications are subject to change without notice due to ongoing product modifications and improvements.

XWProtocol User Manual

Catalog No. Rev. A