MacComm OCX Control documentation
Version 1.01 Beta 8

Interface overview:

Methods:

OpenPort()
ClosePort()
ReadParameter([in] Address, [in] ParamNum, [out] Value)
ReadParameterAlternate([in] Address, [in] ParamNum, [out] Value)
WriteParameter([in] Address, [in] ParamNum, [in] Value)
WriteParameterAlternate([in] Address, [in] ParamNum, [in] Value)
GetParamNumFromName([in] ParamNum)
GetLastError()
GetLastErrorStr([in] long ErrorCode)
GetParameterType([in] ParamNum)
Reset([in] Address)
ResetWait([in] Address)
WriteToFlash([in] Address)
WriteToFlashWait([in] Address)
SetFactors([in] PositionFactor, [in] AccelerationFactor, [in] VelocityFactor)
AboutBox()

Properties:

ComPort
Retries

Method Descriptions:
In the examples MacComm is an instance of the MacComm OCX.
NOTE: All methods will block the calling thread until completed.

Name: OpenPort()
Return type: Boolean Returns true if open was successful
Description: Use this method to open the port

Example(s):
C++:
Opening the port
`bool Result=MacComm.OpenPort();`
 	
BASIC:
Opening the port
`Dim Result As Boolean`
`Result = MacComm.OpenPort`

Name: ClosePort()
Description: Use this method to close the port

Example(s):
C++:
Closing the port
`MacComm.ClosePort();`
 	
BASIC:
Closing the port
`MacComm.ClosePort`

Name: ReadParameter([in] Address, [in] ParamNum, [out] Value)

Parameters:

Type	Name	Description
16 bit signed integer	Address	Address of the MAC motor (Use 255 to broadcast)
16 bit signed integer	ParamNum	Parameter number
32 bit signed integer (pointer)	Value	Value to be written (Pointer)

Return type: Boolean
Returns true if read was successful
It will try the amount of times the property “Retries” has been set to before returning false.

Description:
Use this method to read a parameter from a Macmotor register

Value is one of the following types cast to a long integer:

Word:	16 bit unsigned integer
Integer:	16 bit signed integer
LongInt:	32 bit signed integer
Fixed4:	16 bit signed fixed point (Unit: 1/4096)
Fixed8:	16 bit signed fixed point (Unit: 1/256)

Example(s):
C++:

Getting operation mode (Parameter number 2)
```
long Value;
bool Result=MacComm.ReadParameter(255,2,&Value);
```

Getting position (Parameter 10: P_IST)
```
long Value;
bool Result=MacComm.ReadParameter(255,3,&Value);
```

BASIC:

Common dim statements:
```
Dim LocalValue As Long
Dim Result As Boolean
```

Getting operation mode (Parameter number 2)
```
Result = MacComm.ReadParameter(255, 2, LocalValue)
```

Getting position (Parameter 10: P_IST)
```
Result = MacComm.ReadParameter(255, 10, LocalValue)
```

Name: ReadParameterAlternate([in] Address, [in] ParamNum, [out] Value)

Parameters:

Type	**Name**	**Description**
16 bit signed integer	Address	Address of the MAC motor (Use 255 to broadcast)
16 bit signed integer	ParamNum	Parameter number
32 bit floating point (pointer)	Value	Value to be written (Pointer)

Return type: Boolean
Returns true if read was successful
It will try the amount of times the property "Retries" has been set to before returning false.

Description:
Use this method to read a parameter from a Macmotor register
This method uses the factors for Acceleration, Position and Velocity-registers, which can be set by calling SetFactors.
For the other registers the value just passes through
Types are handled automatically by this method

Example(s):
C++:
```
    Getting operation mode (Parameter number 2)
        float Value;
        bool Result=MacComm.ReadParameterAlternate(255,2,&Value);

    Getting position (Parameter 10: P_IST) multiplied with Positionfactor
        float Value;
        bool Result=MacComm.ReadParameterAlternate(255,3,&Value);
```

BASIC:
```
    Common dim statements:
        Dim LocalValue As Single
        Dim Result As Boolean

    Getting operation mode (Parameter number 2)
        Result = MacComm.ReadParameterAlternate(255, 2, LocalValue)

    Getting position (Parameter 10: P_IST) multiplied with Positionfactor
        Result = MacComm.ReadParameterAlternate(255, 10, LocalValue)
```

Name: WriteParameter([in] Address, [in] ParamNum, [in] Value)

Parameters:

Type	**Name**	**Description**
16 bit signed integer	Address	Address of the MAC motor (Use 255 to broadcast)
16 bit signed integer	ParamNum	Parameter number
32 bit signed integer	Value	Value to be written

Return type: Boolean

Returns true if write was successful
It will try the amount of times the property "Retries" has been set to before returning false.

Description:

Use this method to write a parameter to a Macmotor register

Value is one of the following types cast to a long integer:

Word:	16 bit unsigned integer
Integer:	16 bit signed integer
LongInt:	32 bit signed integer
Fixed4:	16 bit signed fixed point (Unit: 1/4096)
Fixed8:	16 bit signed fixed point (Unit: 1/256)

Example(s):

C++:

Setting operation mode (Parameter number 2) to Position mode (Value 2)

```
bool Result=MacComm.WriteParameter(255,2,2);
```

Setting Position (Parameter 3: P_SOLL) to 4096 (Value 4096)

```
bool Result=MacComm.WriteParameter(255,3,4096);
```

BASIC:

Setting operation mode (Parameter number 2) to Position mode (Value 2)

```
Dim Result As Boolean
Result = MacComm1.WriteParameter(255, 2, 2)
```

Setting Position (Parameter 3: P_SOLL) to 4096 (Value 4096)

```
Dim Result As Boolean
Result = MacComm1.WriteParameter(255, 3, 4096)
```

Name: WriteParameterAlternate([in] Address, [in] ParamNum, [in] Value)

Parameters:

Type	**Name**	**Description**
16 bit signed integer	Address	Address of the MAC motor (Use 255 to broadcast)
16 bit signed integer	ParamNum	Parameter number
32 bit floating point	Value	Value to be written

Return type: Boolean
Returns true if write was successful
It will try the amount of times the property “Retries” has been set to before returning false.

Description:
Use this method to write a parameter to a Macmotor register
This method uses the factors for Acceleration, Position and Velocity-registers, which can be set by calling SetFactors.
For the other registers the value just passes through
Types are handled automatically by this method

Example(s):
C++:

```
Setting operation mode (Parameter number 2) to Position mode (Value 2)
    bool Result=MacComm.WriteParameterAlternate(255,2,2);

Setting Position (Parameter 3: P_SOLL) to 4000 divided by Positionfactor (Value 4000)
    bool Result=MacComm.WriteParameterAlternate(255,3,4000);
```

BASIC:

```
Setting operation mode (Parameter number 2) to Position mode (Value 2)
    Dim Result As Boolean
    Result = MacComm1.WriteParameterAlternate(255, 2, 2)

Setting Position (Parameter 3: P_SOLL) to 4000 divided by Positionfactor (Value 4000)
    Dim Result As Boolean
    Result = MacComm1.WriteParameterAlternate(255, 3, 4000)
```

Name: GetParamNumFromName([in] ParamName)

Parameters:

Type	**Name**	**Description**
String	ParamName	Parameter name

Return type: 16 bit signed integer
Returns parameter number or 0 if not found.

Description:
Use this method to retrieve the parameter number from the name

Example(s):

C++:

Getting last error code:

```
unsigned short ParamNum=MacComm.GetParamNumFromName(“P_IST”);
```

BASIC:

Getting last error code:

```
Dim ParamNum As Integer
ParamNum=MacComm.GetParamNumFromName(“P_IST”)
```

Name: AboutBox()

Description:
Shows a dialog about the program

Example(s):

C++:

Show the about box

```
MacComm.AboutBox();
```

BASIC:

Show the about box

```
MacComm.AboutBox
```

Name: GetLastError()

Return type: 32 bit signed integer
Returns an error code like the Windows GetLastError(), but with some additions

Description:
Use this method to retrieve the error code for the last error

Example(s):
C++:
Getting last error code:
```
unsigned short ErrorCode=MacComm.GetLastError();
```

BASIC:
Getting last error code:
```
Dim ErrorCode As Integer
ErrorCode = MacComm.GetLastError
```

Name: GetLastErrorStr([in] ErrorCode)

Parameters:

Type	Name	Description
32 bit signed integer	ErrorCode	Errorcode to be converted to a string

Return type: String
Returns an error code description like the Windows GetLastError(), but with the same additions as GetLastError()

Description:
Use this method to retrieve a description of an error code

Example(s):
C++:
Get description of passed error code
```
CString Text=MacComm.GetLastErrorStr(MacComm.GetLastError());
```

BASIC:
Getting last error code:
```
Dim Description As String
Description = MacComm.GetLastErrorStr(MacComm.GetLastError)
```

Name: GetParameterType([in] ParamNum)

Parameters:

Type	**Name**	**Description**
16 bit signed integer	ParamNum	Parameter number

Return type: 16 bit signed integer

Return value indicates what type the parameter is stored as internally in the MAC Motor

-1	Invalid	Invalid parameter!
0	Word:	16 bit unsigned integer
1	Integer:	16 bit signed integer
2	LongInt:	32 bit signed integer
3	Fixed4:	16 bit signed fixed point (Unit: 1/4096)
4	Fixed8:	16 bit signed fixed point (Unit: 1/256)

Description:

Use this method to determine how a parameter should be sent.
The integer types should just be used as parameters.
The Fixed4 type should be converted to an integer by multiplying with 4096
The Fixed8 type should be converted to an integer by multiplying with 256

Example(s):

C++:

```
Get Parameter 100s type
    short Type=MacComm.GetParameterType(100);
```

BASIC:

```
Get Parameter 100s type
    Dim ParameterType As Integer
    ParameterType = MacComm.GetParameterType(100)
```

Name: Reset([in] Address)

Parameters:

Type	Name	Description
16 bit unsigned short integer	Address	Address of the MAC motor (Use 255 to broadcast)

Return type: Boolean
Returns true if reset was successful
It will try the amount of times the property "Retries" has been set to before returning false.

Description:
Resets MAC motor to last flashed values.
Returns as soon as the Reset command has been sent to the MAC motor.

Example(s):
C++:
Reset MAC motor
```
bool Result=MacComm.Reset(255);
```

BASIC:
Reset MAC motor
```
Dim Result As Boolean
Result=MacComm.Reset(255)
```

Name: ResetWait([in] Address)

Parameters:

Type	Name	Description
16 bit unsigned short integer	Address	Address of the MAC motor (Use 255 to broadcast)

Return type: Boolean
Returns true if reset was successful
It will try the amount of times the property "Retries" has been set to before returning false.

Description:
Resets MAC motor to last flashed values
Returns when MAC motor is ready.

Example(s):
C++:
Reset MAC motor
```
bool Result=MacComm.Reset(255);
```

BASIC:
Reset MAC motor
```
Dim Result As Boolean
Result=MacComm.Reset(255)
```

Name: WriteToFlash([in] Address)

Parameters:

Type	**Name**	**Description**
16 bit unsigned short integer	Address	Address of the MAC motor (Use 255 to broadcast)

Return type: Boolean

Returns true if flashing was successful
It will try the amount of times the property “Retries” has been set to before returning false.

Description:

Writes MAC registers to Flash memory
Returns as soon as the Flash command has been sent to the MAC motor.

Example(s):

C++:

Write registers to flash

```
bool Result=MacComm.WriteToFlash(255);
```

BASIC:

Write registers to flash

```
Dim Result As Boolean
Result=MacComm.WriteToFlash(255)
```

Name: WriteToFlashWait([in] Address)

Parameters:

Type	**Name**	**Description**
16 bit unsigned short integer	Address	Address of the MAC motor (Use 255 to broadcast)

Return type: Boolean

Returns true if flashing was successful
It will try the amount of times the property “Retries” has been set to before returning false.

Description:

Writes MAC registers to Flash memory
Returns when MAC motor is ready.

Example(s):

C++:

Write registers to flash

```
bool Result=MacComm.WriteToFlashWait(255);
```

BASIC:

Write registers to flash

```
Dim Result As Boolean
Result=MacComm.WriteToFlashWait (255)
```

Name: SetFactors([in] PositionFactor, [in] AccelerationFactor, [in] VelocityFactor)

Parameters:

Type	**Name**	**Description**
16 bit floating point	Pos	Position Factor
16 bit floating point	Acc	Acceleration Factor
16 bit floating point	Vel	Velocity Factor

Description:
Sets factors used by ReadParameterAlternate and WriteParameterAlternate
The defaults are

Name	**Factor**	**Resulting unit**
PositionFactor	1/4096	Pulses
AccelerationFactor	~248.3	RPM/s
VelocityFactor	~0.4768	RPM

The following registers are also converted, but these factors are fixed:

7 (T_SOLL)	100/1023	Percent
8 (P_SIM)	1/16	Encoder counts
16 (I2T)	1/22	Percent (assuming I2TLIM is 2200)
18 (UIT)	1/6	Percent (assuming UITLIM is 600)
41 (T_HOME)	100/1023	Percent
77-80 (T1-4)	100/1023	Percent
121 (VF_OUT)	100/1023	Percent
122 (ANINP)	10/1023	Volts
123 (ANINP_OFFSET)	10/1023	Volts
124 (ELDEGN_OFFSET)	360/2048	Degrees
125 (ELDEGP_OFFSET)	360/2048	Degrees
143 (ELDEG_IST)	360/2048	Degrees
151 (U_SUPPLY)	0.0537	Volts

Example(s):

C++:

Set Position factor to 1/4096 (Converts Pulses to revolutions), and disable the other factors

```
MacComm.SetFactors((float)1/4096,1,1);
```

BASIC:

Set Position factor to 1/4096 (Converts Pulses to revolutions), and disable the other factors

```
MacComm.SetFactors 1/4096,1,1
```

Installation

The MacComm OCX and required DLLs are installed automatically by running Setup.exe and following the onscreen prompts.
You have the option to install a Visual Basic sample and a LabVIEW along with the OCX.

It can also be done manually by copying the following Microsoft redistributable DLLs to the Windows\System folder:

- OLEAUT32.DLL
- OLEPRO32.DLL

MacComm.OCX should be placed in a directory called MacComm in the Windows folder, and registered with RegSvr32 i.e. “Regsvr32 C:\Windows\MacComm\MacComm.ocx”

Adding MacComm OCX to the program

Visual Basic 6

1. In the menu Projects click Components.
2. Make sure the "Selected Items Only" checkbox is NOT selected
3. Find "MacComm OCX Control module", and put a checkmark besides it, and click OK

The MacComm OCX is now available in the controls bar
When put on a form the properties page of the object can be used to set the startup values for the 2 properties (Retries and ComPort)

Visual C++ 6

1. In the menu "Projects" choose "Add To Project" and click "Components and Controls…"
2. Go into the folder "Registered ActiveX Controls" and click "MacComm Control"
3. Click Insert, and two times OK followed by a Close

The MacComm OCX is now available in the controls bar
When put on a dialog the properties page of the object can be used to set the startup values for the 2 properties (Retries and ComPort)

Visual .NET

1. In the menu "Tools" click "Customize Toolbox…"
2. Find "MacComm OCX Control module", and put a checkmark besides it, and click OK

The MacComm OCX is now available in the Toolbox
When put on a form the properties page of the object can be used to set the startup values for the 2 properties (Retries and ComPort)

Borland C++ Builder 6.0

1. In the menu "Component" click "Import ActiveX Control…"
2. Select "MacComm ActiveX Control module…" in the lists of components.
3. Press the "install…" button.
4. On the page "Into existing package" select the dclusr.bpk file (This should be default) and click "OK".
5. Select "yes" to rebuild the package.
6. The ActiveX should now be available in the tool palette on the ActiveX page.

LabVIEW 7.0

1. Place an ActiveX container on your Front Panel.
2. Right click it and select "Insert ActiveX object…"
3. Select MacComm Control from the list.
4. Connect it to a "Property node" and use this to setup the properties.
5. Connect it to an "Invoke node" and use this to call the methods.

Custom Errors:

Hex value:	**Description**
2000 0000	Serial port could not be initialized
2000 0001	Serial port is not open
2000 0002	Could not write required Bytes to serial port
2000 0003	Answer is not of expected length
2000 0004	Invalid accept from mac motor
2000 0005	Writesync error in reply
2000 0006	Address mismatch in reply
2000 0007	Parameter number mismatch in reply
2000 0008	Reply length mismatch
2000 0009	Inversion check failed on value
2000 000A	Endsync error in reply