

MDF4 Writer Documentation

Version 1.0.0.1 May 14 2014

Methods

CreateMDF4(BSTR strToolName, BSTR strToolVendor, BSTR strToolVersion, BSTR strComment, LONG lVersion);	Creates an MDF4.x file Name of tool which generated the file Vendor of the tool Tool version Comment if applicable MDF4 version (400 for 4.00 or 410 for 4.10)
get_strPathName(BSTR* pVal);	Returns the path name
put_strPathName(BSTR newVal);	Sets the path name
AddGroup(BSTR strComment, LONG ExpNValues, LONG* iNo);	Add a signal group to the file Group comment Expected number of records (values per signal) Returns the group id
AddGroup64(BSTR strComment, LONGLONG ExpNValues, LONG* iNo);	Add a signal group to the file (64 Bit version) Group comment Expected number of records (values per signal) Returns the group id
AddSignal(LONG iGroupNo, BSTR strName, BSTR strLabel, LONG lDataType, LONG lFirstBit, LONG lnBits, BSTR strUnit, DOUBLE yFactor, DOUBLE yOffset, LONG bHasNovalues, DOUBLE Novalue, LONG invalPos, LONG* iNo);	Add a signal to the group The group id where to add the signal Signal name (unique inside group) Signal label Data type (see below) First bit used, -1 = don't care Number of bits used SI-unit or SI-derived unit for signal Factor for linear transformation Offset for linear transformation 1 if signal has novalues, 0 if not Nvalue Position of invalid bit, -1 = don't care Returns the signal id
AddTimeInfo(LONG iGroupNo, BSTR strName, BSTR strUnit, DOUBLE Factor, DOUBLE Offset);	Adds a virtual time signal (equidistant time) The group id where to add the signal Signal name (unique inside group) SI-unit or SI-derived unit for signal Factor for linear transformation Offset for linear transformation
AddTimeSignal(LONG iGroupNo, BSTR strName, LONG lDataType, LONG lnFirstBit, LONG lnBits, BSTR strUnit, DOUBLE Factor, DOUBLE Offset, LONG* iNo);	Adds time signal (non- equidistant time) The group id where to add the signal Signal name (unique inside group) Data type (see below) First bit used, -1 = don't care Number of bits used SI-unit or SI-derived unit for signal Factor for linear transformation Offset for linear transformation Returns the signal id

<code>MakeGroups(void);</code>	Create the groups/channels in the file. After this do not add any groups or signals
<code>SetSignalValue(LONG iGroupNo, LONG iSignalNo, DOUBLE Value, LONG bIsNoValue);</code>	Set the value for a signal in the current record Group id Signal id Value 1 = this is a novalue
<code>WriteRecord(LONG iGroupNo);</code>	Write the current record
<code>FlushGroup(LONG iGroupNo);</code>	Flush data buffers (required for MDF4.10)
<code>Close(void);</code>	Close the file
<code>void CreateSRBlock(long lGroupNo, double dt, double xrange, double OldXFactor, double OldXOffset)</code>	Creates a signal reduction block Group number delta t for SR intervals Time range of group Original time increment Original time offset
<code>void SetFileTime(__int64 FileTime)</code>	Sets the file time A FILETIME object
<code>void FileDescription(LPCTSTR strHDComment, LONG lTimerQualityClass, LONGLONG start_time_ns, WORD tz_offset_min, WORD dst_offset_min, BYTE time_flags)</code>	Sets meta data for the file Comment in header Timer class Start time in ns since 1.1.1601 Time zone offset in minutes Daylight saving time in minutes Time flags (cf. below)
<code>void GroupDescription(long iGroupNo, LPCTSTR strAcqName, LPCTSTR strCGComment, LPCTSTR strSIName, LPCTSTR strSIPath, LPCTSTR strSICComment)</code>	Sets meta data for the group Group number Acquisition name Comment of channel group Source information name Source information path Source information comment
<code>void SignalDescription(long iGroupNo, long iSignalNo, LPCTSTR strSIName, LPCTSTR strSIPath, LPCTSTR strSICComment)</code>	Sets meta data for the signal Group number Signal number Source information name Source information path Source information comment
<code>void SetSignalDiscrete(long iGroupNo, long iSignalNo, LONG bDiscrete, LPCTSTR strN2T,</code>	Sets the signal discrete bit Group number Signal number !0 = discrete A string with numerical-to-text conversion, Format: <val>=text <val>=text ... Example: 0=ON 1=OFF
<code>long get_RecordSize(long lGroupID)</code>	Returns the record size Group number
<code>void SetRecord(long lGroupNo, unsigned char * pBuffer)</code>	Sets the record size Group number Data
<code>void SetMinMax(long iGroupNo, long iSignalNo, double min, double max)</code>	Sets min/max of channel Group number Signal number Minimum value Maximum value

long iGroupNo,	Group number
long iSignalNo,	Signal number
double Min,	Minimum value (raw value)
double Max)	Maximum value (raw value)
long get_InvalidBytes(long lGroupID)	Return the number of invalid bytes Group number

Notes:

- AddGroup() exists as a 64-Bit Version (AddGroup64()), because Visual Basic supports LONGLONG only in 64-Bit-Excel. Other languages should use AddGroup64().
- Signal groups in MDF4 have a common time axis, which may be either equidistant or non-equidistant. In the first case a virtual signal is defined by factor and offset. A non-equidistant time signal receives its value through SetSignalValue().
- Data types supported are:


```
// enumeration for member cn_data_type
#define CN_D_UINT_LE    0  // Unsigned Integer LE Byte Order
#define CN_D_SINT_LE    2  // Signed Integer LE Byte Order
#define CN_D_FLOAT_LE   4  // Float (IEEE 754) LE Byte Order
```
- Time flags:


```
// enumeration for member hd_time_class
#define HD_TC_LOCALPC    0  // local PC reference time (Default)
#define HD_TC_EXTERN    10  // external time source
#define HD_TC_EXTABS    16  // external absolute synchronized time
```
- Linear transformation: The values of a signal may be scaled using a linear transformation. This allows to use small data storage for signals. The formula is $\text{physVal} = \text{rawval} * \text{factor} + \text{offset}$
- You must register the COM module (regsvr32 MDF4Writer.dll). This requires Administrator rights.

Programming Sequence

It is important to use a certain programming sequence when using the lib. The sequence is:

1. Create the object
2. Set the path name
3. Call CreateMDF4()
4. Add groups using either AddGroup() or AddGroup64()
5. Add signals to the group using AddSignal
6. Add a time definition to the group (AddTimeInfo() or AddTimeSignal())
7. It is allowed to define an additional group after a group has been fully described.
8. Call MakeGroups(). This will write the data groups and channel definitions to the physical file. After this, you must not define additional groups and/or signals
9. Fill a record of a group with data. Call SetSignalValue() for every signal of the group including a non-equidistant time signal.
10. Call WriteRecord() to write the record to the file.
11. Repeat 9 and 10 for all groups and records
12. Call FlushGroup() for all groups to make sure that non-written blocks are compressed and written to disk.

13. Call Close() to close the file.

Example in C++ (Microsoft Visual Studio 2010)

```
// enumeration for member cn_data_type
#define CN_D_UINT_LE 0 // Unsigned Integer LE Byte Order
#define CN_D_SINT_LE 2 // Signed Integer LE Byte Order
#define CN_D_FLOAT_LE 4 // Float (IEEE 754) LE Byte Order

void WriteMDF4Example(void)
{
    CMDF4Writer m4;
    long idGroup[2], idSignal[7], i;
    __int64 i64N = 100;

    CoInitializeEx(NULL, 0); // don't forget this

    //if (!m4.CreateDispatch(_T("{891BCB49-095B-417C-9235-564194E85533}")))
    if (!m4.CreateDispatch(_T("MDF4WriterLib.1")))
    {
        DWORD dwErr = GetLastError();
        _tprintf(_T("Cannot create dispatch interface\n"));
        return;
    }

    // if file exists, delete it first
    if (_taccess("C:\\Temp\\M4Test.mf4",0)==0)
        _tunlink("C:\\Temp\\M4Test.mf4");
    // Set the file name before you create the file
    m4.put_strPathName( "C:\\Temp\\M4Test.mf4");
    m4.CreateMDF4( "Caller", "Lego", "1.0", "No comment", 410); // Version MDF4.10

    // Add a group with an equidistant, virtual time signal with 10 Hz sampling rate
    // and 10 s offset
    m4.AddGroup64( "Group 1 Test", i64N, &idGroup[0]);
    m4.AddTimeInfo( idGroup[0], "Time", "s", 0.1, 10);
    // Square wave, unsigned int 8 bit
    m4.AddSignal( idGroup[0], "Square", "Signal with square wave", CN_D_UINT_LE, -1, 8,
        "V", 1.0, 0.0, 0, 0.0, -1, &idSignal[0]);
    // Sawtooth wave, signed int 8 bit
    m4.AddSignal( idGroup[0], "Sawtooth", "Signal with sawtooth wave", CN_D_SINT_LE, -1, 8,
        "A", 1.0, 0.0, 0, 0.0, -1, &idSignal[1]);
    // Rampe, double
    m4.AddSignal( idGroup[0], "Ramp", "Signal with ramp wave", CN_D_FLOAT_LE, -1, 64, "m",
        1.0, 0.0, 0, 0.0, -1, &idSignal[2]);

    // Add a group with a non-equidistant time signal with approx. 10 Hz sampling rate
    // and 0 s offset
    m4.AddGroup64( "Group 2 Test", i64N, &idGroup[1]);
    m4.AddTimeSignal( idGroup[1], "Time", CN_D_FLOAT_LE, -1, 64, "s", &idSignal[3]);
    m4.AddSignal( idGroup[1], "Square", "Signal with square wave", CN_D_UINT_LE, -1, 8,
        "V", 1.0, 0.0, 1, -100.0, -1, &idSignal[4]);
    m4.AddSignal( idGroup[1], "Sawtooth", "Signal with sawtooth wave", CN_D_SINT_LE, -1, 8,
        "A", 0.1, -5.0, 0, 0.0, -1, &idSignal[5]);
    m4.AddSignal( idGroup[1], "Ramp", "Signal with ramp wave", CN_D_FLOAT_LE, -1, 64, "m",
        1.0, 0.0, 0, 0.0, -1, &idSignal[6]);

    // Now create the groups and channels in the MDF4 file
    m4.MakeGroups();
    // After this, do not change/add groups or signals

    // Write the data
    for (i=0; i<i64N; i++)
    {
```

```

// Group 1
m4.SetSignalValue( idGroup[0], idSignal[0], i<i64N/2 ? 0.0 : 255.0, 0);
m4.SetSignalValue( idGroup[0], idSignal[1], (double)(i%10)-5, 0);
m4.SetSignalValue( idGroup[0], idSignal[2], (double)i, 0);
m4.WriteRecord( idGroup[0] );

// Group 2
m4.SetSignalValue( idGroup[1], idSignal[3],
    (double)i/10 + (double)(rand()-16384)/163840., 0);
if (i==50)
    m4.SetSignalValue( idGroup[1], idSignal[4], i<i64N/2 ? 0.0 : 255.0, 1);
else
    m4.SetSignalValue( idGroup[1], idSignal[4], i<i64N/2 ? 0.0 : 255.0, 0);
m4.SetSignalValue( idGroup[1], idSignal[5], (double)(i%10)-5, 0);
m4.SetSignalValue( idGroup[1], idSignal[6], (double)i, 0);
m4.WriteRecord( idGroup[1] );
}
// Flush records, close open data blocks
m4.FlushGroup( idGroup[0] );
m4.FlushGroup( idGroup[1] );

// Close the file
m4.Close();
}

```

Example in VBA (Microsoft Excel 2010)

```

Sub WriteMDF4()
    Dim TSL As Object
    Dim id As Long
    Dim IsNoval As Long
    Dim i64N As Long
    Dim idGroup(2) As Integer
    Dim idSignal(7) As Integer
    Dim val As Double

    Const CN_D_UINT_LE = 0 ' Unsigned Integer LE Byte Order
    Const CN_D_SINT_LE = 2 ' Signed Integer LE Byte Order
    Const CN_D_FLOAT_LE = 4 ' Float (IEEE 754) LE Byte Order

    i64N = 100

    Set m4 = CreateObject("MDF4WriterLib.1")
    ' Make sure the file does not exist! (sorry, I don't know how to unlink a file in VB)
    m4.strPathName = "C:\Temp\M4Test.mf4"
    m4.CreateMDF4 "Caller", "Lego", "1.0", "No comment", 410
    ' Add a group with an equidistant, virtual time signal with 10 Hz sampling rate
    ' and 10 s offset
    m4.AddGroup "Group 1 Test", i64N, id
    idGroup(0) = id
    m4.AddTimeInfo idGroup(0), "Time", "s", 0.1, 10
    ' Square wave, unsigned int 8 bit
    m4.AddSignal idGroup(0), "Square", "Signal with square wave", CN_D_UINT_LE, 8, "V",
        1#, 0#, 0, 0#, id
    idSignal(0) = id
    ' Sawtooth wave, signed int 8 bit
    m4.AddSignal idGroup(0), "Sawtooth", "Signal with sawtooth wave", CN_D_SINT_LE, 8,
        "A", 1#, 0#, 0, 0#, id
    idSignal(1) = id
    ' Rampe, double
    m4.AddSignal idGroup(0), "Ramp", "Signal with ramp wave", CN_D_FLOAT_LE, 64, "m", 1#,

```

```

        0#, 0, 0#, id
idSignal(2) = id

' Add a group with a non-equidistant time signal with approx. 10 Hz sampling rate and
' 0 s offset
m4.AddGroup "Group 2 Test", i64N, id
idGroup(1) = id
m4.AddTimeSignal idGroup(1), "Time", CN_D_FLOAT_LE, 64, "s", id
idSignal(3) = id
' This signal has a novalue of -100
m4.AddSignal idGroup(1), "Square", "Signal with square wave", CN_D_UINT_LE, 8, "V",
    1#, 0#, 1, -100#, id
idSignal(4) = id
' Use a factor/offset scaling for this signal: Factor = 0.1, Offset -5
m4.AddSignal idGroup(1), "Sawtooth", "Signal with sawtooth wave", CN_D_SINT_LE, 8, "A",
    0.1, -5#, 0, 0#, id
idSignal(5) = id
m4.AddSignal idGroup(1), "Ramp", "Signal with ramp wave", CN_D_FLOAT_LE, 64, "m", 1#,
    0#, 0, 0#, id
idSignal(6) = id

' Now create the groups and channels in the MDF4 file
m4.MakeGroups
' After this, do not change/add groups or signals

' Write the data
For i = 0 To i64N - 1
    ' Group 1
    If i < i64N / 2 Then val = 0 Else val = 255
    m4.SetSignalValue idGroup(0), idSignal(0), val, 0
    val = (i Mod 10) - 5
    m4.SetSignalValue idGroup(0), idSignal(1), val, 0
    m4.SetSignalValue idGroup(0), idSignal(2), i, 0
    m4.WriteRecord idGroup(0)

    ' Group 2
    val = i / 10 + (Rnd() / 10)
    m4.SetSignalValue idGroup(1), idSignal(3), val, 0
    If i < i64N / 2 Then val = 0 Else val = 255
    If i = 50 Then
        m4.SetSignalValue idGroup(1), idSignal(4), val, 1 ' missing value
    Else
        m4.SetSignalValue idGroup(1), idSignal(4), val, 0
    End If
    val = (i Mod 10) - 5
    m4.SetSignalValue idGroup(1), idSignal(5), val, 0
    m4.SetSignalValue idGroup(1), idSignal(6), i, 0
    m4.WriteRecord idGroup(1)
Next i

' Flush records, close open data blocks
m4.FlushGroup idGroup(0)
m4.FlushGroup idGroup(1)

' Close the file
m4.Close
End Sub

```


License

Copyright 2014 Michael Bühler & Bernd Sparrer. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY Michael Bühler & Bernd Sparrer ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL Michael Bühler OR Bernd Sparrer OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of Michael Bühler & Bernd Sparrer.