MDF4 Writer Documentation

Version 1.0.0.28 Mar 30 2020

The Lib has two interfaces: A standard COM interface and an ordinary C interface (DLL).

The following describes the COM objects methods. The DLL interface is listed at the end of the document.

Methods

```
CreateMDF4(                       Creates an MDF4.x file
       BSTR strToolName,          Name of tool which generated the file
       BSTR strToolVendor,        Vendor of the tool
       BSTR strToolVersion,       Tool version
       BSTR strComment,           Comment (if applicable) for file history
       LONG lVersion);            MDF4 version (400 for 4.00 or 410 for 4.10)

get_strPathName(                  Returns the path name
       BSTR* pVal);

put_strPathName(                  Sets the path name
       BSTR newVal);

AddGroup(                         Add a signal group to the file
       BSTR strComment,           Group comment
       LONG ExpNValues,           Expected number of records (values per signal)
       LONG* iNo);                Returns the group id

AddGroup64(                       Add a signal group to the file (64 Bit version)
       BSTR strComment,           Group comment
       LONGLONG ExpNValues,       Expected number of records (values per signal)
       LONG* iNo);                Returns the group id

AddSignal(                        Add a signal to the group
       LONG iGroupNo,             The group id where to add the signal
       BSTR strName,              Signal name (unique inside group)
       BSTR strLabel,             Signal label
       LONG lDataType,            Data type (see below)
       LONG lFirstBit,            First bit used, -1 = don’t care
       LONG lnBits,               Number of bits used
       BSTR strUnit,              SI-unit or SI-derived unit for signal
       DOUBLE yFactor,            Factor for linear transformation
       DOUBLE yOffset,            Offset for linear transformation
       LONG bHasNovalues,         1 if signal has novalues, 0 if not
       DOUBLE Novalue,            Novalue
       LONG invalPos,             Position of invalid bit, -1 = don’t care
       LONG* iNo);                Returns the signal id

AddSignalEx(                      Add a signal to the group
       LONG iGroupNo,             The group id where to add the signal
       BSTR strName,              Signal name (unique inside group)
       BSTR strLabel,             Signal label
       LONG lDataType,            Data type (see below)
       LONG lFirstBit,            First bit used, -1 = don’t care
       LONG lnBits,               Number of bits used
       BSTR strUnit,              SI-unit or SI-derived unit for signal
       DOUBLE yFactor,            Factor for linear transformation
       DOUBLE yOffset,            Offset for linear transformation
       LONG bHasNovalues,         1 if signal has novalues, 0 if not
       DOUBLE Novalue,            Novalue
       LONG invalPos,             Position of invalid bit, -1 = don’t care
       LONG* iNo,                 Returns the signal id
       LONG default_x_signal);    ID of x default signal

AddTimeInfo(                      Adds a virtual time signal (equidistant time)
       LONG iGroupNo,             The group id where to add the signal
       BSTR strName,              Signal name (unique inside group)
       BSTR strUnit,              SI-unit or SI-derived unit for signal
       DOUBLE Factor,             Factor for linear transformation
       DOUBLE Offset);            Offset for linear transformation

AddTimeSignal(                    Adds time signal (non- equidistant time)
       LONG iGroupNo,             The group id where to add the signal
       BSTR strName,              Signal name (unique inside group)
       LONG lDataType,            Data type (see below)
       LONG lnFirstBit,           First bit used, -1 = don’t care
       LONG lnBits,               Number of bits used
       BSTR strUnit,              SI-unit or SI-derived unit for signal
       DOUBLE Factor,             Factor for linear transformation
       DOUBLE Offset,             Offset for linear transformation
       LONG* iNo);                Returns the signal id

MakeGroups(void);                 Create the groups/channels in the file. After this
                                  do not add any groups or signals

SetSignalValue(                   Set the value for a signal in the current record
       LONG iGroupNo,             Group id
       LONG iSignalNo,            Signal id
       DOUBLE Value,              Value
       LONG bIsNovalue);          1 = this is a novalue

WriteRecord(LONG iGroupNo);       Write the current record

FlushGroup(LONG iGroupNo);        Flush data buffers (required for MDF4.10)

Close(void);                      Close the file

void CreateSRBlock(               Creates a signal reduction block
       long lGroupNo,             Group number
       double dt,                 delta t for SR intervals
       double xrange,             Time range of group
       double OldXFactor,         Original time increment
       double OldXOffset)         Original time offset

void SetFileTime(                 Sets the file time
       __int64 FileTime)          A FILETIME object

void FileDescription(             Sets meta data for the file
       LPCTSTR strHDComment,      Comment in header
       LONG lTimerQualityClass,   Timer class
       LONGLONG start_time_ns,    Start time in ns since 1.1.1970
       WORD tz_offset_min,        Time zone offset in minutes
       WORD dst_offset_min,       Daylight saving time in minutes
       BYTE time_flags)           Time flags (cf. below)

void FileDescriptionEx(           Sets meta data for the file
       LPCTSTR strHDComment,      Comment in header (no changes to xml text)
       LONG lTimerQualityClass,   Timer class
       DOUBLE start_time_s,       Start time in s since 1.1. 1970
       WORD tz_offset_min,        Time zone offset in minutes
       WORD dst_offset_min,       Daylight saving time in minutes
       BYTE time_flags)           Time flags (cf. below)

void GroupDescription(            Sets meta data for the group
       long iGroupNo,             Group number
       LPCTSTR strAcqName,        Acquisition name
       LPCTSTR strCGComment,      Comment of channel group
       LPCTSTR strSIName,         Source information name
       LPCTSTR strSIPath,         Source information path
       LPCTSTR strSIComment)      Source information comment

void SignalDescription(           Sets meta data for the signal
       long iGroupNo,             Group number
       long iSignalNo,            Signal number
       LPCTSTR strSIName,         Source information name
       LPCTSTR strSIPath,         Source information path
       LPCTSTR strSIComment)      Source information comment

void SetSignalDiscrete(           Sets the signal discrete bit
       long iGroupNo,             Group number
       long iSignalNo,            Signal number
       LONG bDiscrete,            !0 = discrete
       LPCTSTR strN2T,            A string with numerical-to-text conversion,
                                  Format: <val>=text|<val>=text …
                                  Example: 0=ON|1=OFF

long get_RecordSize(              Returns the record size
       long lGroupID)             Group number

void SetRecord(                   Sets the record data
       long lGroupNo,             Group number
       unsigned char * pBuffer)   Data

void SetMinMax(                   Sets min/max of channel
       long iGroupNo,             Group number
       long iSignalNo,            Signal number
       double Min,                Minimum value (raw value)
       double Max)                Maximum value (raw value)

void SetTimeStat(                 Sets the time signal properties
                                  Must be called if later SetRangeTime() is called.
       long iGroupNo              Group number
       double xMean               Mean of time delta
       double xMin                Minimum of time signal
       double xMax                Maximum of time signal
       double xMinDelta           Minimum of time signal delta
       double xMaxDelta)          Maximum of time signal delta

long get_InvalidBytes(            Return the number of invalid bytes
       long lGroupID)             Group number

void SetRecordValues(             Fills a records with values
       long iGroupNo              Group number
       double * pValues           Array of values (double)
       long nRecords)             Number of records in array

void SetRangeTime (               Sets min/max and raster vales of time channel
                                  To be used after data is written
       long iGroupNo,             Group number
       long iSignalNo,            Signal number
       double Min,                Minimum value (raw value)
       double Max,                Maximum value (raw value)
       double minDelta,           Minimum delta value (phys. Value)
       double maxDelta,           Maximum delta value (phys. Value)
       double raster)             Mean delta value (phys. Value)

void SetRange (                   Sets min/max of channel (not master)
                                  To be used after data is written
       long iGroupNo,             Group number
```

```
    long iSignalNo,             Signal number
    double Min,                 Minimum value (raw value)
    double Max)                 Maximum value (raw value)
```

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 an 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 physVal = rawval * factor + offset
- You must register the COM module (regsvr32 MDF4Writer.dll). This requires Administrator rights.

Programming Sequence

It is import 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 graup hase been fully described.
8. Call MakeGroups(). This will write the data groups and channel definitions to the physical file. Aifter 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
```

The C interface

The C interface has some additional functions to load and initialize the library.

A C++ class wrapper is provided (CMDF4WriterLib.h and CMDF4WriterLib.cpp). The usage of the wrapper class is shown in the Caller example (DLLWriteMDF4Example()).

The exported functions (the first argument for all function is the handle (INT_PTR iHandle)):

```
INT_PTR M4WRInitDLL (void);       Return a pointer to the writer class.

void M4WRExitDLL(void);           Frees the writer object, unload the DLL.

M4WRCreateMDF4             same as COM
M4WRSetPathName            same as COM
M4WRAddGroup               same as COM
M4WRAddGroup64             same as COM
M4WRAddSignal              same as COM
M4WRAddTimeInfo            same as COM
M4WRAddTimeSignal          same as COM
M4WRMakeGroups             same as COM
M4WRSetSignalValue         same as COM
M4WRWriteRecord            same as COM
M4WRFlushGroup             same as COM
M4WRClose                  same as COM
M4WRCreateSRBlock          same as COM
M4WRSetFileTime            same as COM
M4WRGroupDescription       same as COM
M4WRSignalDescription      same as COM
DWORD M4WRGetRecordSize(INT_PTR hHandle, LONG lGroupID);
M4WRSetRecord              same as COM
M4WRSetMinMax              same as COM
LONG M4WRGetInvalidBytes (INT_PTR hHandle, LONG lGroupID);
M4WRFileDescription        same as COM
M4WRSetSignalDiscrete      same as COM
M4WRSetTimeStat            same as COM
M4WRSetRecordValues        same as COM
M4WRCreateSRBlocks         same as COM
M4WRSetFileTimeFraction    same as COM
M4WRSetNanoTimeUTC         same as COM
BOOL M4WRWriteRecords (INT_PTR hHandle, LONG iGroupNo, void *pBuffer, LONG nRecords)
```

License