MDF4 Reader Documentation

Version 1.0.0.16 Jul 26 2017

Methods

```
OpenMDF4(                       Opens an existing MDF4.x file
      LPCTSTR strPathName)      Full path name of the file

long get_Version ()             Returns the file version (310 = 3.10, 410 = 4.10)

long get_FileTime()             Returns the file time as UNIX time

double get_TimeFraction()       Returns the remainder of file time as
                                fraction of [s]

long get_NGroups()              Returns the number of valid groups

GetGroupName (                  Returns the group name
      BSTR *strGroupName,       Returns the group name
      LONG iGroupIndex)         Index of group

LoadGroup(                      Loads the group information (this is then the
                                "current group")
      LONG iGroupIndex)         Index of group

long get_NSignals()             Returns the number of signals in the current group

long get_NSamples()             Returns the number of samples (records) in the
                                current group

double get_FirstTimeStamp ()    Returns the first time stamp in the current group

double get_LastTimeStamp ()     Returns the last time stamp in the current group

TimeToIndex(                    Calculates the sample index from a time stamp
      double TimeStamp,         Time stamp to convert
      long lStartIndex,         Index to start search (0 -> from beginning)
      long *lIndex)             Returns the index

LoadTimeSignal(                 Loads the time signal information
      BSTR *strTimeName)        Returns the name of the time signal

LoadSignal(                     Loads the signal information ("current signal")
      BSTR *strTimeName,        Returns the name of the signal
      long iSignalIndex)        Index of signal

GetTimeSignal(                  Gets information from the current time signal
      BSTR * strUnit,           Returns the unit
      long * monotony,          Returns the monotony of the time signal
      double * rmin,            Returns the minimal delta t
      double * rmax,            Returns the maximal delta t
      double * raster)          Returns delta t for equidistant signals

GetSignal(                      Gets information from the current signal
      BSTR *strDisplayName,     Returns the display name
      BSTR *strAliasName,       Returns the alias name
      BSTR *strUnit,            Returns the unit
      BSTR *strComment,         Returns the comment text
      long *discrete)           Returns if a signal is a logical signal

GetData (                       Gets data from the current signal
```

```
       long bTime,               !=0 if from time signal, 0 if from current signal
       long lFirstIndex,         First index to read from
       long lLastIndex,          Last index to read from
       VARIANT *pBuffer,         a safearray of doubles
       long *nValuesRead)        Return the number of values read

GetDataEx (                      Gets data from the current signal (called by ref)
       long bTime,               !=0 if from time signal, 0 if from current signal
       long lFirstIndex,         First index to read from
       long lLastIndex,          Last index to read from
       VARIANT *pBuffer,         a safearray of doubles
       long *nValuesRead)        Return the number of values read

CacheData (                      Caches data from the current signal
       long bTime,               !=0 if from time signal, 0 if from current signal
       long lFirstIndex,         First index to read from
       long lLastIndex,          Last index to read from
       long *nValuesCached)      Return the number of values cached

double get_CachedValue (         Gets a data value from cache
       long bTime,               !=0 if from time signal, 0 if from current signal
       long lIndex)              Index to read from (original index in group)

string get_Comment (             Gets a text comment form a block
       long nBlock,              Block ID (see below)
       long nElement)            Element ID (see below)

long get_NoOfSRBlocks()          Returns the number of SR-Blocks for the current
                                 group

double get_SRdt(long nBlock)     Returns delta t of SR Block nBlock

long get_SRCycleCount(long nBlock) Returns number of cycles of SR Block nBlock

void CacheSRData(                Caches SR data from the current signal
       long nBlock,              Index of SR block
       long bTime,               !=0 if from time signal, 0 if from current signal
       long lFirstIndex,         First SR index to read from
       long lLastIndex,          Last SR index to read from
       long *nValuesCached)      Return the number of values cached

void CachedSRValues(             Gets SR data values from cache
       long bTime,               !=0 if from time signal, 0 if from current signal
       long lIndex,              SR index to read from (original index in group)
       double * Min,             Minimum value, master: start value of interval
       double * Max,             Maximum value, master: minimum raster value
       double * Mean)            Mean value, master: maximum raster value

unsigned __int64 get_NSamples64() Returns number of cycles of current group

unsigned __int64 get_MDF4File()  Returns a pointer to the internal object.

void GetSignalDetail(            Returns signal detail information
       long * lDataType,         Data type (cf. enumeration below)
       long * nFirstBit,         First bit in reords
       long * nBits,             Number of bits used by signal
       double * Factor,          Scaling factor
       double * Offset,          Scaling offset
       double * RawMin,          Minimum as raw value
       double * RawMax,          Maximum as raw value
       long * bHasNoValues,      True if signal has novalues
       long* invalPos)           Bit position in invalid bits

long get_RecordSize()            Return the record size of the current group.
```

```
void GetRecord(                  Read a raw record
       __int64 i64StartIndex,    Start index to read from
       __int64 i64EndIndex,      Last index to read from
       unsigned char * pBuffer)  Buffer for data

long get_InvalidBytes()          Returns the number of invalid bytes in record

long get_TimerQualityClass()     Return the timer quality class
// 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

// Block types
#define ID_HEADER       1
#define ID_FILEHISTORY  2
#define ID_DATAGROUP    3
#define ID_CHANNELGROUP 4
#define ID_CHANNEL      5
#define ID_SI_GROUP     6
#define ID_SI_CHANNEL   7

// Elements (enumeration for text/comment members)
// Header
#define hd_md_comment 5
// File History
#define fh_md_comment 1
// Data Group
#define dg_md_comment 3
// Channel Group
#define cg_tx_acq_name 2
#define cg_md_comment  5
// Channel
#define cn_tx_name     2
#define cn_md_comment  7
// Channel / Channel Group: SI Block
#define si_tx_name    0
#define si_tx_path    1
#define si_md_comment 2
```

Notes:

- There is always a “current group”. This is the group which has been loaded by LoadGroup().
- There is always a “current time signal”. This is the time signal of the current group. Before you call get_FirstTimeStamp() or get_LastTimeStamp(), you must call LoadTimeSignal()!
- There is always a “current signal”. This is one of the signals of the current group.
- 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. The raster information is contained im some MDF4 files and may be queried by GetTimeSignal().
- There are two methods to read the data:
  1: Use a SafeArray to read the data into a buffer
  2: Cache data in DLLs memory and retrieve values one-by-one. The caches of the time signal and the current signal are distinct, i.e., you may keep data od both signals in memory.
- To read SR blocks: Read the number of SR blocks, use the delta t to find out the appropriate block, then cache the data and read the value triples.

- You must register the COM module (regsvr32 MDF4Reader.dll). This requires Administrator rights.

Programming Sequence

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

1. CoInitialize() to initialize the COM interface
2. Create the object
3. Call OpenMDF4()
4. Call get_NGroups() to obtain the number of valid groups (valid means more than 0 samples)
5. Call LoadTimeSignal() to get the time signal
6. Get samples, time range and number of signals in group
7. Call LoadSignal() to get a signal
8. Get signal info of that signal
9. Determine first and last index to read from using TimeToIndex().
10. If method 1 is used: Create a SafeArray and warp it in a VARIANT. Call GetData() to transfer the values.
11. If method 2 is used: Call CacheData() for both, time and signal, and the read the values per get_CachedValue() in free order.

Example in C++ (Microsoft Visual Studio 2010)

```
// Block types
#define ID_HEADER       1
#define ID_FILEHISTORY  2
#define ID_DATAGROUP    3
#define ID_CHANNELGROUP 4
#define ID_CHANNEL      5
#define ID_SI_GROUP     6
#define ID_SI_CHANNEL   7

// enumeration for text/comment members
// Header
#define hd_md_comment 5
// File History
#define fh_md_comment 1
// Data Group
#define dg_md_comment 3
// Channel Group
#define cg_tx_acq_name 2
#define cg_md_comment  5
// Channel
#define cn_tx_name     2
#define cn_md_comment  7
// Channel / Channel Group: SI Block
#define si_tx_name    0
#define si_tx_path    1
#define si_md_comment 2


void ReadMDF4Example(void)
{
  CMdf4Reader m4; // The COM object
```

```
CString str;
int iGrp, nGroups, iSig, nSignals;
LONG l,n;
double val;
long mon, nValues=10, idx1, idx2;
double xmin, xmax, rmin, rmax, raster;
// Some BSTRs
BSTR t,tTime;

CoInitializeEx(NULL, 0); // don't forget this
// Create the object
if (!m4.CreateDispatch(_T("{A5D406EA-0508-415E-B5E2-E868370D3721}")))
{
  DWORD dwErr = GetLastError();
  _tprintf(_T("Cannot create dispatch interface\n"));
  return;
}

// Get an MDF4 file
CFileDialog fdlg(TRUE,_T(".mf4"));
if (fdlg.DoModal() != IDOK)
  return;
printf("File %s\n", fdlg.GetPathName());
m4.OpenMDF4(fdlg.GetPathName());

// Get file time and other infos
CTime ti(m4.get_FileTime());
val = m4.get_TimeFraction()*1000; // ms
str = ti.Format("%d:%m:%Y %H:%M:%S");
printf("FileTime = %s.%03ld\n",str,(int)val);

str = m4.get_Comment(ID_HEADER, hd_md_comment);
if (!str.IsEmpty())
  printf("Header comment = %s\n",str);
str = m4.get_Comment(ID_FILEHISTORY, fh_md_comment);
if (!str.IsEmpty())
  printf("File history = %s\n",str);

nGroups = m4.get_NGroups();
printf("No. of groups = %ld\n",nGroups);
// walk through all groups
for (iGrp=0; iGrp<nGroups; iGrp++)
{
  // Calling a COM function which uses BSTR:
  t=NULL; tTime=NULL; // BSTR must be NULL
  m4.GetGroupName(&t, iGrp); // allocates t
  printf("Group %ld = %s\n",iGrp+1, (char *)_bstr_t(t));
  ::SysFreeString(t); // we must free this string

  // Load the current group
  m4.LoadGroup(iGrp);
  tTime=NULL;

  // Load the time signal of the current group
  m4.LoadTimeSignal(&tTime);

  nSignals = m4.get_NSignals();
  l = m4.get_NSamples();
  printf("  No. of signals = %ld, %ld samples\n",nSignals,l);

  // Time range of group measurement:
  xmin = m4.get_FirstTimestamp();
  xmax = m4.get_LastTimestamp();
  printf("  Time %lf to %lf\n",xmin,xmax);
  printf("  Time signal %s\n", (char *)_bstr_t(tTime));
   ::SysFreeString(tTime);

  // Get time raster information
  // raster = delta t between data points
  // rmin   = minimal delta
  // rmax   = maximal delta, same as rmin for equidistant data
  // mon    = Monotony, defined by MDF4 as
  // #define CN_MON_NOTDEFINED       0
  // #define CN_MON_DECREASE         1
  // #define CN_MON_INCREASE         2
  // #define CN_MON_STRICT_DECREASE  3
```

```
// #define CN_MON_STRICT_INCREASE  4
// #define CN_MONOTONOUS           5
// #define CN_STRICT_MON           6
// #define CN_NOT_MON              7
t = NULL;
m4.GetTimeSignal(&t, &mon, &rmin, &rmax, &raster);
printf("    Unit = %s  Raster = %lf (%lf - %lf)  %ld\n", (char *)_bstr_t(t),raster,rmin,rmax,mon);
::SysFreeString(t);

// Get indexes of time range xmin + 8s, xmin + 8.5s
if (xmax >= xmin+8.5)
{
  m4.TimeToIndex(xmin+8.0, 0, &idx1);
  m4.TimeToIndex(xmin+8.5, idx1, &idx2);
}
else // other time range
{
  m4.TimeToIndex(xmin, 0, &idx1);
  m4.TimeToIndex(xmax, idx1, &idx2);
}
nValues = idx2-idx1+1; // Calc number of data points ^

// walk thru signals
for (iSig=0; iSig<nSignals; iSig++)
{
  t = NULL;
  m4.LoadSignal(&t, iSig);
  printf("  Signal %s:\n", (char *)_bstr_t(t));
  ::SysFreeString(t);

  // Get signal description
  LONG discrete; // signal has discrete values (logical signal)
  BSTR tDisplayName=NULL, tAliasName=NULL, tUnit=NULL, tComment=NULL;
  m4.GetSignal(&tDisplayName, &tAliasName, &tUnit, &tComment, &discrete);
  printf("    Displayname %s\n", (char *)_bstr_t(tDisplayName));
  printf("    Aliasname %s\n", (char *)_bstr_t(tAliasName));
  printf("    Unit %s\n", (char *)_bstr_t(tUnit));
  printf("    Comment %s\n", (char *)_bstr_t(tComment));
  ::SysFreeString(tDisplayName);
  ::SysFreeString(tAliasName);
  ::SysFreeString(tUnit);
  ::SysFreeString(tComment);

  // Read data from time signal and the signal itself:
  // Create 2 SafeArrays
  CComSafeArray<double> *pData;
  CComSafeArray<double> *pTimeData;
  CComSafeArrayBound bound;
  bound.SetCount(nValues); // nValues has been claculated above
  bound.SetLowerBound(0);
  pData = new CComSafeArray<double>(&bound,1);
  pTimeData = new CComSafeArray<double>(&bound,1);

  // Wrap safearrays by a VARIANT
  VARIANT vData, vTimeData;
  vData.parray = *pData->GetSafeArrayPtr();
  vData.vt = VT_ARRAY;
  pData->Detach();
  vTimeData.parray = *pTimeData->GetSafeArrayPtr();
  vTimeData.vt = VT_ARRAY;
  pTimeData->Detach(); // do not lock before call

  // Get the data form the time signal
  m4.GetData(TRUE, idx1, idx2, &vTimeData, &n);
  // Get the data from the signal
  m4.GetData(FALSE, idx1, idx2, &vData, &n);
  pData->Attach(vData.parray);
  pTimeData->Attach(vTimeData.parray);
  // Access the data:
  for (int i=0; i<n; i++)
  {
    printf("    %.3lf  %.3lf\n",pTimeData->GetAt(i), pData->GetAt(i));
  }
  // free memory
  delete pTimeData;
  delete pData;
```

```
      printf("=============================================\n");

      // Method 2: Use cached data and values one-by-one
      m4.CacheData(TRUE, idx1, idx2, &n);
      m4.CacheData(FALSE, idx1, idx2, &n);
      // Access the data:
       for (int i=idx1; i<idx1+n; i++)
      {
        printf("    %.3lf  %.3lf\n",m4.get_CachedValue(TRUE, i),
          m4.get_CachedValue(FALSE, i));
      }
     }
   }
   // Release the COM object and free memory
   m4.ReleaseDispatch();
}
```

License