BINLOG DLL

Manual

Version 1.5

Vector Informatik GmbH, Ingersheimer Straße 24, D-70499 Stuttgart
Tel. +49 (711) 8 06 70-0, Fax + 49 (711) 8 06 70-5 55,
Email can@vector-informatik.de, Internet http://www.vector-informatik.de

Subsidiaries

Frankreich

Vector France SAS

168, Boulevard Camélinat
F-92240 Malakoff

Tel.: +33 1 4231 4000
Fax: +33 1 4231 4009

http://www.vector-france.com

Japan

Vector Japan Co., Ltd.

Nishikawa Bldg. 2F
3-3-9 Nihonbashi, Chuo-ku
J-103-0027 Tokyo

Tel.: +81 3 3516 7850
Fax: +81 3 3516 7855

http://www.vector-japan.co.jp

Schweden

VecScan AB

Fabriksgatan 7
S-41250 Göteborg

Tel.: +46 031 79901 35
Fax: +46 031 79903 05

http://www.vecscan.com/

USA

Vector CANtech, Inc.

Suite 550
39500 Orchard Hill Place
USA-Novi, Mi 48375

Tel.: +1 248 449 9290
Fax: +1 248 449 9704

http://www.vector-cantech.com

Addresses of our distributors can be found on our website:

http://www.vector-informatik.com

Contents

1 Introduction

This document describes the usage of the binlog.dll provided with CANoe/CANalyzer.

The BL package is installed in the folder `Programming\BLF_Logging` of the CANoe/CANalyzer installation.

Besides the binlog header file in the `Include` folder, a sample VisualStudio project is provided in the subfolder `VS_Project`, which demonstrates the usage of the binlog.dll. The resulting sample program `bl.exe` creates the BL file `test.blf`.

In the subfolder `Demo` CANoe/CANalyzer configurations are provided, which make use of the generated sample BL file.

2 BL Functions

2.1 Overview

```
BLAPI( HANDLE) BLCreateFile(  LPCTSTR lpFileName,
                              DWORD dwDesiredAccess);

BLAPI( BOOL)   BLCloseHandle( HANDLE hFile);

BLAPI( BOOL)   BLWriteObject( HANDLE hFile,
                              VBLObjectHeaderBase* pBase);

BLAPI( BOOL)   BLPeekObject(  HANDLE hFile,
                              VBLObjectHeaderBase* pBase);

BLAPI( BOOL)   BLSkipObject(  HANDLE hFile,
                              VBLObjectHeaderBase* pBase);

BLAPI( BOOL)   BLReadObject(  HANDLE hFile,
                              VBLObjectHeaderBase* pBase);

BLAPI( BOOL)   BLReadObjectSecure( HANDLE hFile,
                                   VBLObjectHeaderBase* pBase,
                                   size_t expectedSize);

BLAPI( BOOL)   BLFreeObject(  HANDLE hFile,
                              VBLObjectHeaderBase* pBase);

BLAPI( BOOL)   BLSetApplication(   HANDLE hFile, BYTE appID,
                                   BYTE appMajor,
                                   BYTE appMinor,
                                   BYTE appBuild);

BLAPI( BOOL)   BLSetWriteOptions( HANDLE hFile,
                                   DWORD dwCompression,
                                   DWORD dwReserved);
```

```
BLAPI( BOOL)   BLSetMeasurementStartTime
                            ( HANDLE hFile,
                              const LPSYSTEMTIME lpStartTime);

BLAPI( BOOL)   BLGetFileStatistics
                           ( HANDLE hFile,
                             VBLFileStatistics* pStatistics);

BLAPI( BOOL)   BLGetFileStatisticsEx
                          ( HANDLE hFile,
                            VBLFileStatisticsEx* pStatistics);

BLAPI( BOOL)   BLFlushFileBuffers( HANDLE hFile,
                                   DWORD dwFlags);
```

2.2 BLCreateFile

Syntax	`BLAPI( HANDLE) BLCreateFile( LPCTSTR lpFileName, DWORD dwDesiredAccess)`
Description	Use this function to open a BL file with the desired access.

Parameters	`LPCTSTR lpFileName`
	Pointer to a null-terminated string that specifies the name of the file to create or open.
	`DWORD dwDesiredAccess`
	Specifies the type of access to the file. An application can obtain read access or write access. This parameter can be GENERIC_READ or GENERIC_WRITE.

Return values	If the function succeeds, the return value is an open handle to the specified file. If the function fails, the return value is INVALID_HANDLE_VALUE.

2.3 BLCloseHandle

Syntax	`BLAPI( BOOL) BLCloseHandle( HANDLE hFile)`
Description	Use this function to close a BL file opened with BLCreateFile.
Parameters	`HANDLE hFile` The file handle returned by BLCreateFile.
Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.4 BLWriteObject

Syntax	`BLAPI( BOOL) BLWriteObject( HANDLE hFile, VBLObjectHeaderBase* pBase)`
Description	Use this function to write a BL object to the file.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_WRITE access to the file.
	`VBLObjectHeaderBase* pBase`
	Pointer to a BL object structure containing the data to be

	written to the file.
Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.5 BLPeekObject

Syntax	`BLAPI( BOOL) BLPeekObject( HANDLE hFile, VBLObjectHeaderBase* pBase)`
Description	Use this function to read the base header part of a BL object.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLObjectHeaderBase* pBase`
	Pointer to a BL object structure that receives the object header description.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.6 BLSkipObject

Syntax	`BLAPI( BOOL) BLSkipObject( HANDLE hFile, VBLObjectHeader Base* pBase)`
Description	Use this function to skip a BL object.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLObjectHeaderBase* pBase`
	Pointer to a BL object structure that describes the object to be skipped.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.7 BLReadObject (Obsolete)

Obsolete: This function has been replaced by BLReadObjectSecure.

Syntax	`BLAPI( BOOL) BLReadObject( HANDLE hFile, VBLObjectHeaderBase* pBase)`
Description	Use this function to read a BL object.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLObjectHeaderBase* pBase`
	Pointer to a BL object structure that describes the object to be read.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.8 BLReadObjectSecure

Syntax	`BLAPI( BOOL) BLReadObjectSecure( HANDLE hFile, VBLObjectHeaderBase* pBase, size_t expectedSize)`
Description	Use this function to read a BL object.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLObjectHeaderBase* pBase`
	Pointer to a BL object structure that describes the object to be read
	`size_t expectedSize`
	Size of BL object structure which is provided by pointer pBase.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.9 BLFreeObject

Syntax	`BLAPI( BOOL) BLFreeObject( HANDLE hFile, VBLObjectHeaderBase* pBase)`
Description	Use this function to free the memory which has been allocated for a previously read BL object. Altough this is only required for dy-

	namic sized objects such as environment variables it doesn't harm to call this method for fixed sized objects like CAN messages as well.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLObjectHeaderBase* pBase`
	Pointer to a BL object structure that describes the object to be freed.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.10 BLSetApplication

Syntax	`BLAPI( BOOL) BLSetApplication( HANDLE hFile,`
	`BYTE appID,`
	`BYTE appMajor,`
	`BYTE appMinor,`
	`BYTE appBuild)`

Description	Use this function to specify the application which writes the file.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_WRITE access to the file.
	`BYTE appID`
	The application identifier.
	`BYTE appMajor`
	The application major version number.
	`BYTE appMinor`
	The application minor version number.
	`BYTE appBuild`
	The application build version number.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.11 BLSetWriteOptions

Syntax	`BLAPI( BOOL) BLSetWriteOptions( HANDLE hFile, DWORD dwCompression, DWORD dwReserved)`
Description	Use this function to set the compression.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_WRITE access to the file.
	`DWORD dwCompression`
	The compression to be used during write. Valid values range from 0 (no compression) to 10 (maximum compres sion).
	`DWORD dwReserved`
	Reserved. Must be zero.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.12 BLSetMeasurementStartTime

Syntax	`BLAPI( BOOL) BLSetMeasurementStartTime ( HANDLE hFile, const LPSYSTEMTIME lpStartTime);`
Description	Use this function to set the measurement start time

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_WRITE access to the file.
	`LPSYSTEMTIME lpStartTime`
	The pointer to the windows system time structure

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.13 BLGetFileStatistics

Syntax	`BLAPI( BOOL) BLGetFileStatistics( HANDLE hFile, VBLFileStatistics* pStatistics)`
Description	Use this function to retrieve the file statistics.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLFileStatistics* pStatistics`
	The pointer to the file statistics structure.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.14 BLGetFileStatisticsEx

Syntax	`BLAPI( BOOL) BLGetFileStatisticsEx ( HANDLE hFile, VBLFileStatisticsEx* pStatistics)`
Description	Use this function to retrieve the extended file statistics.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`VBLFileStatisticsEx* pStatistics`
	The pointer to the extended file statistics structure.

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.15 BLFlushFileBuffers

Syntax	`BLAPI( BOOL) BLFlushFileBuffers( HANDLE hFile, DWORD dwFlags)`
Description	

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_WRITE access to the file.
	`DWORD dwFlags`

	Flag indicating how to flush. Valid values are: BL_FLUSH_STREAM - flushes all internal streams BL_FLUSH_FILE - flushes the file and combinations thereof.
Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

2.16 BLSeekTime

Syntax	`BLAPI( BOOL) BLSeekTime ( HANDLE hFile,`
	`ULONGLONG timeStamp,`
	`void* arg,`
	`BOOL (*pProgressCallback)(void*, FLOAT),`
	`WORD callbackRate)`

Description	Use this function to seek forward in a BLF file to the first object with a certain time stamp.

Parameters	`HANDLE hFile`
	The file handle returned by BLCreateFile. The file handle must have been created with GENERIC_READ access to the file.
	`ULONGLONG timestamp`
	The time stamp value you are searching for.
	`void* arg`
	Argument which is passed back to the pProgressCallback call. It can be used as a bridge between the C-Style binlog interface and C++ (by passing the class this pointer).
	`BOOL (*pProgressCallback) (void*, FLOAT)`
	Callback function, which passes back the arg pointer and the progress value (between 0 and 1.0).
	`WORD callbackRate`
	Rate how often pProgressCallback is called (in ms).

Return values	If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

3 BL structures

3.1 Overview

VBLObjectHeaderBase

VBLFileStatistics

3.2 VBLObjectHeaderBase

```
typedef struct VBLObjectHeaderBase_t
{
     DWORD mSignature;   /* signature (BL_OBJ_SIGNATURE) */
     WORD  mHeaderSize;  /* sizeof object header */
     WORD  mHeaderVersion;    /* header version (1) */
     DWORD mObjectSize;       /* object size */
     DWORD mObjectType;  /* object type (BL_OBJ_TYPE_XXX) */
} VBLObjectHeaderBase;
```

3.3 VBLFileStatistics

```
typedef struct VBLFileStatistics_t
{
     DWORD mStatisticsSize;   /* sizeof (VBLFileStatistics) */
     BYTE  mApplicationID;    /* application ID */
     BYTE  mApplicationMajor; /* application major number */
     BYTE  mApplicationMinor; /* application minor number */
     BYTE  mApplicationBuild; /* application build number */
     ULONGLONG mFileSize;     /* file size in bytes */
     ULONGLONG mUncompressedFileSize;
                         /* uncompressed file size in bytes */
     DWORD mObjectCount;      /* number of objects */
     DWORD mObjectsRead;      /* number of objects read */
} VBLFileStatistics;
```

3.4 VBLFileStatisticsEx

```
typedef struct VBLFileStatisticsEx_t
{
    DWORD mStatisticsSize; /* sizeof (VBLFileStatisticsEx) */
    BYTE mApplicationID;   /* application ID */
    BYTE mApplicationMajor; /* application major number */
    BYTE mApplicationMinor; /* application minor number */
    BYTE mApplicationBuild; /* application build number */
    ULONGLONG mFileSize; /* file size in bytes */
    ULONGLONG mUncompressedFileSize;
                    /* uncompressed file size in bytes */
    DWORD mObjectCount; /* number of objects */
    DWORD mObjectsRead; /* number of objects read */
    SYSTEMTIME mMeasurementStartTime;
                         /* measurement start time */
    SYSTEMTIME mLastObjectTime; /* last object time */
    DWORD      mReserved[18]; /* reserved */
} VBLFileStatisticsEx;
```

4 Additional informations

4.1 Extended CAN identifiers

The following structure is used to write CAN frames:

```
typedef struct VBLCANMessage_t
{
    VBLObjectHeader mHeader;    /* object header */
    WORD            mChannel;   /* application channel */
    BYTE            mFlags;     /* CAN dir & rtr */
    BYTE            mDLC;       /* CAN dlc */
    DWORD           mID;        /* CAN ID */
    BYTE            mData[8];   /* CAN data */
} VBLCANMessage;
```

The member mID is used for the numeric identifier of the frame. If you want to write an extended frame identifier, you have to set the highest bit of the mID field. E.g. if you want to write a frame with the extended identifier 0x100, you have to do the following:

message.mID = 0x80000100

For the same frame with a standard identifier you would use the field mID in the following way:

message.mID = 0x00000100

5 License

5.1 Acknowledgement

The compression routines used in the BL library derive from the zlib library.

5.2 The zlib Software License

zlib (http://www.gzip.org/zlib/) Copyright (C) 1995-2002 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

- The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.

- Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.

- This notice may not be removed or altered from any source distribution.

Jean-loup Gailly (jloup@gzip.org) Mark Adler (madler@alumni.caltech.edu)

The data format used by the zlib library is described by RFCs (Request for Comments) 1950 to 1952 in the files ftp://ds.internic.net/rfc/rfc1950.txt (zlib format), rfc1951.txt (deflate format) and rfc1952.txt (gzip format).