Ethernet

BLF Logging Format

Specification

Version 1.3 of 2020-02-07

Status	Completed
Publisher	Vector Informatik GmbH

Document Management

Revision list

Version	Date	Editor	Section	Changes, comments
1.0	2016-06-22	Jr	All	Moved from CAN_and_General_BLF_Format to own document Added hardware channel Added extended Ethernet logging events
1.1	2016-07-04	Ft	All	Editorial changes
1.2	2016-07-25	Jr	All	Fixed description of hardware channel
1.2.1	2017-02-24	Mom	All	CI and layout
1.3	2020-02-07	vsn	3	Public API uses standard types, e.g. uint32_t instead of DWORD. Mentioned Linux libbinlog.so

VECTOR

Contents

1 Disclaimer

Severability clause - Restrictions for the usage of Vector logging data formats outside of Vector products

The format specification / access functions for the Vector BLF and ASC logging data formats are made available under the restrictions and conditions cited hereafter.

Please note that Vector Informatik GmbH neither gives any guarantee nor assumes any liability beyond compulsory legal regulations for the BLF or ASC logging format respectively as well as for the access functions to the single objects.

Vector Informatik GmbH disclaims all liability for errors which might be contained in the access functions or the format specification itself.

Vector Informatik GmbH does neither provide support for the integration into your software nor for problems occurring inside your software on the customer side.

Beyond that Vector Informatik GmbH reserves the right to change the BLF or ASC data format respectively anytime without prior notification. Therefore, the compatibility of the format is not ensured.

2 Overview

The document specifies the format of Ethernet events in the CANoe/CANalyzer BLF logging. The described structures can be used to read and write BLF logging files using the binlog.dll or libbinlog.so, which can be found in the CANoe/CANalyzer User Data folder:

<UserDataFolder>\Programming\BLF_Logging

3 Format Description

3.1 VBLEthernetFrame

Description: Ethernet frame.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_FRAME`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mSourceAddress[6]	uint8_t	Ethernet (MAC) address of source computer (network byte order).
mChannel	uint16_t	The channel of the frame.
mDestinationAddress[6]	uint8_t	Ethernet (MAC) address of target computer (network byte order).
mDir	uint16_t	Direction flag: 0 = Rx, 1 = Tx, 2 = TxRq
mType	uint16_t	Ethernet Type Field which indicates the protocol for Ethernet payload data See protocol specifications for valid values.
mTPID	uint16_t	TPID when VLAN tag valid, zero when no VLAN. See Ethernet standard specification.

Parameter	Type	Description
mTCI	uint16_t	TCI when VLAN tag valid, zero when no VLAN. See Ethernet standard specification.
mPayLoadLength	uint16_t	Length of Ethernet payload data in bytes. Max. 1582 Byte (without Ethernet header)
mPayLoad	uint8_t*	Ethernet payload data (without Ethernet header)

Note: *The size of a VBLEthernetFrame object depends on the length of the payload data. To set the size correctly you have to add the payload data length to the object size:*

```
VBLEthernetFrame ethFrame;

ethFrame.mHeader.mBase.mObjectSize = sizeof(VBLEthernetFrame) +
ethFrame.mPayLoadLength;
```

The following piece of code shows how to save an Ethernet packet to a BLF file.

```
BLHANDLE hFile;
  // open / create the BLF file
  // ...
VBLEthernetFrame packet
uint8_t             payload[1500];
uint32_t            payloadLength  = 0;
  // source Mac 00:01:02:03:04:05; target Mac 06:07:08:09:0A:0B
uint8_t             sourceMacId[6] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05 };
uint8_t             targetMacId[6] = { 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B };
  // fill payload, payload length
  // ...
packet.mChannel       = 1;
packet.mDir           = 0;
packet.mType          = 0x0800;
packet.mTPID          = 0;
packet.mTCI           = 0;
packet.mPayLoadLength = payloadLength;
packet.mPayLoad       = payload;
memcpy( &packet.mSourceAddress     , sourceMacId, sizeof(sourceMacId) );
memcpy( &packet.mDestinationAddress, targetMacId, sizeof(targetMacId) );

packet.mHeader.mObjectTimeStamp  = 0;
packet.mHeader.mBase.mObjectSize = sizeof(VBLEthernetFrame) +
packet.mPayLoadLength;

::BLWriteObject( hFile, &packet.mHeader.mBase );
```

The following piece of code shows how to read in Ethernet packets:

```
BLHANDLE hFile;
  // open the BLF file
  // ...
VBLObjectHeaderBase base;
VBLEthernetFrame packet;

while( BLPeekObject( hFile, &base ))
{
  switch( base.mObjectType)
  {
  case BL_OBJ_TYPE_ETHERNET_FRAME:
    packet.mHeader.mBase = base;
    BLReadObject( hFile, &packet.mHeader.mBase );
    // process Ethernet packet
    // ...
    BLFreeObject( hFile, &packet.mHeader.mBase );
```

```
      break;
    default:
      BLSkipObject( hFile, &base);
      break;
    }
  }
```

3.2 VBLEthernetStatus

Description: Ethernet status event.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_STATUS`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mChannel	uint16_t	The channel of the event.
mFlags	uint16_t	Valid fields: Bit 0 – Link Status Bit 1 – Bit rate Bit 2 – Ethernet Phy Bit 3 – Duplex Bit 4 – MDI Type Bit 5 – Connector Bit 6 – Clock Mode Bit 7 – Pairs Bit 8 – Hardware Channel
mLinkStatus	uint8_t	0 – Unknown 1 – Link Down 2 – Link up 3 – Negotiate 4 – Link error
mEthernetPhy	uint8_t	0 – Unknown 1 – IEEE 100BASE-TX, IEEE 1000BASE-T 2 – IEEE 100BASE-T1 (OABR[1])
mDuplex	uint8_t	0 – Unknown 1 – Half Duplex 2 – Full Duplex
mMdi	uint8_t	0 – Unknown 1 – Direct 2 – Crossover
mConnector	uint8_t	0 – Unknown 1 – RJ4 2 – D-Sub

[1] OABR = OPEN Alliance BroadR-Reach

Parameter	Type	Description
mClockMode	uint8_t	0 – Unknown 1 – Master 2 – Slave
mPairs	uint8_t	0 – Unknown 1 – one pair 2 – two pairs 3 – four pairs
mHardwareChannel	uint8_t	Hardware channel (if network interface supports more than one hardware channel per application channel)
mBitrate	uint32_t	Bitrate in [kbit/sec]

3.3 VBLEthernetStatistic

Description: Ethernet statistic event.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_STATISTIC`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mChannel	uint16_t	The channel of the event.
mRcvOk_HW	uint64_t	Rx-frames detected by HW in epoch
mXmitOk_HW	uint64_t	Tx-frames detected by HW in epoch
mRcvError_HW	uint64_t	Rx-error frames detected by HW
mXmitError_HW	uint64_t	Tx-error frames detected by HW
mRcvBytes_HW	uint64_t	Rx-bytes detected by HW in epoch
mtXmitBytes_HW	uint64_t	Tx-bytes detected by HW in epoch
mRcvNoBuffer_HW	uint64_t	Rx-frames with lost/dropped data buffer in epoch
mSQI	Int16_t	Value for the Quality of the 100BASE-T1 (OABR) connection 0 – ErrorOccuring 1 – NoMargin 2 – Marginal 3 – Acceptable 4 – Good 5 – Excellent 6 – not available
mHardwareChannel	uint16_t	Hardware channel (if network interface supports more than one hardware channel per application channel)

3.4 VBLEthernetRxError

Description: Ethernet Rx/Tx error frame.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_RX_ERROR`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mStructLength	uint16_t	Length of this structure, without `sizeof(VBLObjectHeader)` and without raw data length
mChannel	uint16_t	The channel of the frame.
mDir	uint16_t	Direction flag: 0 = Rx
mHardwareChannel	uint16_t	Hardware channel (if network interface supports more than one hardware channel per application channel) 0 = invalid
mFcs	uint32_t	Ethernet frame checksum.
mFrameDataLength	uint16_t	Number of valid raw Ethernet data bytes, starting with Target MAC ID.
mError	uint32_t	Error code: 1 - Data Length Error 2 - Invalid CRC 3 - Invalid Data received 4 - Collision detected
mFrameData	uint8_t*	Raw Ethernet frame data.

3.5 VBLEthernetFrameEx

Description: Ethernet frame for Ethernet extended logging.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_FRAME_EX`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mStructLength	uint16_t	Length of this structure without VBLObjectHeader and without frame data.
mFlags	uint16_t	Valid Fields: Bit 0 – reserved Bit 1 – mHardwareChannel valid Bit 2 – mFrameDuration valid Bit 3 – mFrameChecksum valid Bit 4 – mFrameHandle valid
mChannel	uint16_t	The channel of the frame.

Parameter	Type	Description
mHardwareChannel	uint16_t	Hardware channel (if network interface supports more than one hardware channel per application channel)
mFrameDuration	uint64_t	Transmission duration in [ns]
mFrameChecksum	uint32_t	Ethernet frame checksum
mDir	uint16_t	Direction flag: 0 = Rx, 1 = Tx, 2 = TxRq
mFrameLength	uint16_t	Length of Ethernet frame inclusive Ethernet header. Max. 1612 Byte
mFrameHandle	uint32_t	Frame handle
mReserved	uint32_t	Reserved
mFrameData	uint8_t*	Ethernet data

Note: *The size of a VBLEthernetFrameEx object depends on the length of the payload data. To set the size correctly you have to add the payload data length to the object size:*

```
VBLEthernetFrameEx ethFrame;
ethFrame.mHeader.mBase.mObjectSize = sizeof(VBLEthernetFrameEx) +
ethFrame.mFrameLength;
```

3.6 VBLEthernetFrameForwarded

Description: Ethernet frame for Ethernet extended logging which was forwarded by the Ethernet hardware interface.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_FRAME_FORWARDED`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mStructLength	uint16_t	Length of this structure without VBLObjectHeader and without frame data.
mFlags	uint16_t	Valid Fields: Bit 0 – reserved Bit 1 – mHardwareChannel valid Bit 2 – mFrameDuration valid Bit 3 – mFrameChecksum valid Bit 4 – mFrameHandle valid
mChannel	uint16_t	The channel of the frame.
mHardwareChannel	uint16_t	Hardware channel (if network interface supports more than one hardware channel per application channel)
mFrameDuration	uint64_t	Transmission duration in [ns]

Parameter	Type	Description
mFrameChecksum	uint32_t	Ethernet frame checksum
mDir	uint16_t	Direction flag: 0 = Rx, 1 = Tx, 2 = TxRq
mFrameLength	uint16_t	Length of Ethernet frame inclusive Ethernet header. Max. 1612 Byte
mFrameHandle	uint32_t	Frame handle
mReserved	uint32_t	Reserved
mFrameData	uint8_t*	Ethernet data

Note: *The size of a VBLEthernetFrameForwarded object depends on the length of the payload data. To set the size correctly you have to add the payload data length to the object size:*

```
VBLEthernetFrameForwarded ethFrame;
ethFrame.mHeader.mBase.mObjectSize = sizeof(VBLEthernetFrameForwarded) +
ethFrame.mFrameLength;
```

3.7 VBLEthernetErrorEx

Description: Ethernet Rx/Tx error frame for extended logging.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_ERROR_EX`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mStructLength	uint16_t	Length of this structure, without sizeof(VBLObjectHeader) and without raw data length
mFlags	uint16_t	Valid Fields: Bit 0 – reserved Bit 1 – mHardwareChannel valid Bit 2 – mFrameDuration valid Bit 3 – mFrameChecksum valid Bit 4 – mFrameHandle valid
mChannel	uint16_t	The channel of the frame.
mHardwareChannel	uint16_t	Hardware channel (if network interface supports more than one hardware channel per application channel) 0 = invalid
mFrameDuration	uint64_t	Transmission duration in [ns]
mFrameChecksum	uint32_t	Ethernet frame checksum
mDir	uint16_t	Direction flag: 0 = Rx, 1 = Tx, 2 = TxRq

Parameter	Type	Description
mFrameLength	uint16_t	Number of valid raw Ethernet data bytes, starting with Target MAC ID.
mFrameHandle	uint32_t	Frame handle
mError	uint32_t	Error code: 1 - Data Length Error 2 - Invalid CRC 3 - Invalid Data received 4 - Collision detected
mFrameData	uint8_t*	Raw Ethernet frame data. Max. 1612 Byte

3.8 VBLEthernetErrorForwarded

Description: Ethernet Rx/Tx error frame for extended logging which was forwarded by the Ethernet hardware interface.

Corresponding object type: `BL_OBJ_TYPE_ETHERNET_ERROR_FORWARDED`

Parameter	Type	Description
mHeader	VBLObjectHeader	Common header type.
mStructLength	uint16_t	Length of this structure, without `sizeof(VBLObjectHeader)` and without raw data length
mFlags	uint16_t	Valid Fields: Bit 0 – reserved Bit 1 – mHardwareChannel valid Bit 2 – mFrameDuration valid Bit 3 – mFrameChecksum valid Bit 4 – mFrameHandle valid
mChannel	uint16_t	The channel of the frame.
mHardwareChannel	uint16_t	Hardware channel (if network interface supports more than one hardware channel per application channel) 0 = invalid
mFrameDuration	uint64_t	Transmission duration in [ns]
mFrameChecksum	uint32_t	Ethernet frame checksum
mDir	uint16_t	Direction flag: 0 = Rx, 1 = Tx, 2 = TxRq
mFrameLength	uint16_t	Number of valid raw Ethernet data bytes, starting with Target MAC ID.
mFrameHandle	uint32_t	Frame handle

Parameter	Type	Description
mError	uint32_t	Error code: 1 - Data Length Error 2 - Invalid CRC 3 - Invalid Data received 4 - Collision detected
mFrameData	uint8_t*	Raw Ethernet frame data. Max. 1612 Byte