CAN, Log & Trigger

ASC Logging Format

Specification

Version 1.4.6 of 2016-11-30

Status	Completed
Publisher	Vector Informatik GmbH © 2016 All rights reserved. Any distribution or copying is subject to prior written approval by Vector. Note: Hardcopy documents are not subject to change management.

Document Management

Revision list

Version	Date	Editor	Section	Changes, comments
1.0.0	2005-03-04	Gia	All	Creation
1.1.0	2005-03-07	Gia	3.4	Added Begin Triggerblock Event and End Triggerblock Event
1.1.1	2006-11-10	Gia	3.3	Added CAN events with symbolic names >32 characters
1.1.2	2007-05-25	Nb	3.7	Macros: Record Signals
1.1.3	2007-07-10	Tn	3.3	CAN-DLC > 8
1.1.4	2007-10-04	Gia	3.3	Special flags for CAN messages (Spyflag, Wake-Up & etc.)
1.1.5	2007-10-09	Sn	3.1.1	Added version number into header
1.1.6	2007-10-17	Nb	3.5, 3.6	Added system variables and environment variables
1.1.7	2008-06-18	Msc	3.1.2	Added split information into header
1.1.8	2008-09-08	Tn	3.8	GPS event added
1.1.9	2009-03-27	Gia	3.6	Correction in SystemVariableEvent
1.2.0	2009-05-18	Gia	1	Added Disclaimer
1.2.1	2010-10-26	Sha	3.3; 3.9	New fields added to CAN Messages
1.2.2	2010-11-29	Hb		Changes in CAN message and CAN error frame
1.2.3	2011-04-07	Mp	3.9	Comment Event (for comments in Trace Window) added
1.2.4	2011-04-12	Sha	3.3	Hint 2 added
1.2.5	2011-09-08	Sha	3.10	Clarification to CAN frame message length and message duration
1.2.6	2011-10-24	Mp	3.11	Event for Global Markers added
1.2.7	2012-04-20	Hb	3.3	CAN error frame description
1.2.8	2012-04-23	Hb	3.3	Numerical ID-field in CAN frame

Version	Date	Editor	Section	Changes, comments
1.2.9	2012-08-24	Chk	3.4, 3.12	Add CAN FD event description. Add symbol description for CAN FD and modify CAN FD specifics for some symbols
1.3.0	2013-04-11	Fsi		New CAN FD message and error frame description with 64 byte support.
1.3.1	2013-10-15	Chk	3.4	Add chapter 3.4.1 and 3.4.2, for CAN message logging on CAN FD channel
1.3.2	2014-03-13	Chk	3.3, 3.4.1	Format description for remote frames updated Wrong channel description under 3.4.1 CAN Extended Message Event changed
1.3.3	2014-04-15	Srj/Vrd	3.3.1, 3.3.12	Added milliseconds to *<FullTime>* and update of header section.
1.3.4	2014-05-12	Srj	3.3	Changed order of *<MessageFlags>* and *ID* for CAN Events.
1.3.5	2014-10-07	Vrd	3	Defined newline representation
1.3.6	2014-12-09	Vrd	3.3	Remote Frame DLC
1.3.7	2014-12-09	Hb	3.3, 3.4	Extended CAN FD frame- and error frame format
1.3.8	2015-01-26	Rue	3.7	Documented flags in system variables
1.3.9	2015-02-13	Chk	3.4	Insert FLAG and CRC description
1.4.0	2015-04-07	Chk	3.4, 3.4.2	Add flag description for extFlags of CAN FD error frame, and replace EDL by FDF
1.4.1	2015-04-23	Lke	3.12	Test structure events added
1.4.2	2015-06-22	Vrd	3.13	Channel ranges
1.4.3	2015-09-01	Lke	3.12	Corrected description for Test structure event fields execID, elementID.
1.4.4	2015-11-24	Chk	3.3, 3.4	Change description for CAN Error Event
1.4.5	2016-11-30	Mom	all	CI and Layout
1.4.6	2017-03-07	Vrd	3.5, 3.13	Trigger Condition added

Contents

1 Disclaimer

2 Overview

The document specifies the format of CAN, Log and Trigger events in the CANoe/CANalyzer ASC logging.

3 Format

Newline is coded in the CR+LF representation.

3.1 Header

A log file in ASCII format starts with a header. The header contains general information about the logging file. See also section 0 for an explanation of the symbols.

Format	```date <WeekDay> <Month> <Date> <Fulltime> <Year>```
	```base <hex|dec> timestamps <absolute|relative>```
	```<""|no> internal events logged```

Example	```date Wed Apr 16 09:21:13.159 am 2014```
	```base hex timestamps absolute```
	```internal events logged```

- **base** indicates the number system in which values are logged. It can be in hexadecimal or decimal notation.

- **timestamps** indicates whether the timestamps are written absolute to the start of the measurement or relative to the preceding event.

- **internal events logged** indicates whether internal events were logged or not.

Hint: Starting with CANalyzer/CANoe v8.2 SP2 the <Fulltime> is stored in milliseconds. This can be disabled by setting the flag ASCII_Format_Milliseconds = 0 in section [SYSTEM] of the CAN.INI file.

3.1.1 Version number

Starting with CANalyzer/CANoe v7.0 a version number is written after the header in form of a comment:

Format Since v7.0	```// version <major>.<minor>.<patch>```
Example	```// version 8.2.1```

<major>.<minor> denotes the CANalyzer/CANoe version number excluding the build number (e.g. v7.0, v7.1, etc…) and *<patch>* denotes changes version **within** a CANalyzer/CANoe main release (e.g. changes made in a service pack).

The *<major>.<minor>* numbers are generally increased with each new main release of CANalyzer/CANoe **regardless** whether changes have been made to the ASCII format or not. The *<patch>* number of the main release is always zero.

In service packs the *<patch>* number is increased if and only if changes and/or additions have been made to the ASCII format.

3.1.2 Split information

Starting with CANalyzer/CANoe v7.1 split information is written in all subsequent files if a logging block is configured to split ASCII files. It is written as comment after the version number.

Format Since v7.1	```// <time> previous log file: <filename>```
Example	```// 60.0000 previous log file: Inc_L1.asc```

<time> is the last absolute time stamp of the previous log file and <filename> is the filename of the previous log file without path information. This time stamp is always absolute even if the events are logged with relative time stamps. With this information it is possible to restore the original time stamps of events logged with relative time stamps if a subsequent file is replayed.

3.2 Example

The following logging contains CAN Message events, CAN Extended Message events, contains CAN FD Message events, CAN FD Extended Message events, Errorframes, CAN Bus Statistic descriptions, Log Trigger events, Log Direct Start events, CAN Status events, CAN Remote Frame events and CAN Error events.

```
date Die Dez 21 11:29:01 2004
base hex  timestamps absolute
internal events logged
// version 7.0.0
Begin Triggerblock Mon Mar 7 01:21:51 pm 2005
   0.0000 Start der Messung
   0.0006 CAN 1 Status:chip status error active
   0.0006 CAN 2 Status:chip status error active
   1.0100 1  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.0%
   1.0100 2  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.0%
   2.0000 log direct start (0ms)
   2.0000 log trigger event
   2.5009 1  64            Tx   d 8 00 01 02 03 04 05 06 07
   2.5010 2  C8x           Rx   d 8 09 08 07 06 05 04 03 02
   2.5010 1  200           Tx   r
   0.010460 CANFD 1  101            Tx   1 0 d 8 8 F1 F1 F1 F1 F1 F1 F1 F1  Length
= 209000 BitCount = 128 ID = 257
   0.014165 CANFD 1  1C4D80A7x      Tx   1 0 d 8 8 88 88 88 88 88 88 88 88  Length
= 278000 BitCount = 140 ID = 474841255x
   0.010235 CANFD 1  100            Tx   1 0 r 8  Length = 138000 BitCount = 57 ID
= 256
   2.5010 1  ErrorFrame
   2.5010 CAN 1 Status:chip status error active  - TxErr: 0 RxErr: 1
   2.5010 CAN 2 Status:chip status error active  - TxErr: 0 RxErr: 1
   2.7000 log trigger event (this trigger was in post trigger time of last block)
   3.0100 1  Statistic: D 1 R 0 XD 0 XR 0 E 1 O 0 B 0.2%
   3.0100 2  Statistic: D 1 R 0 XD 0 XR 0 E 1 O 0 B 0.2%
   4.0000 log direct stop (0ms)
   4.0000 log trigger event
End TriggerBlock
```

3.3 CAN Events on a Classic CAN bus

The section lists all CAN events in CANoe/CANalyzer ASC logging. See section 0 for an explanation of the symbols.

CAN Message Event	
Simple CAN Message received or transmitted on a CAN channel.	
Format up to v7.2	`<Time> <Channel> <ID> <Dir> d <DLC> <D0> <D1>...<D8> <MessageFlags>`
Format since v7.5	`<Time> <Channel> <ID> <Dir> d <DLC> <D0> <D1>...<D8> Length = <MessageDuration> BitCount = <MessageLength> <MessageFlags>`
Format since v8.0	`<Time> <Channel> <ID> <Dir> d <DLC> <D0> <D1>...<D8> Length = <MessageDuration> BitCount = <MessageLength> ID = <IDnum> <MessageFlags>`
Example	`0.003040 1 123  Tx   d 2 00 00    Length = 768000 BitCount = 67 ID = 291`

Hint 1:

The format of <ID> differs when using symbolic logging. In this case the symbolic names are written instead. In this form: `<Time> <Channel> <ID>     <Dir> d <DLC> <D0> <D1>...<D8>`

Symbolic logging is also valid for CAN Extended Message Events & CAN Remote Frame Events.

Hint 2:

The Length, BitCount, and numerical ID-fields can be suppressed by setting the flag ASCII_Format_7_2=1 in section [CAN] of the CAN.INI file. This flag is available from CANoe/CANalyzer version 7.5 SP3.

CAN Extended Message Event	
CAN Message with extended identifier received or transmitted on a CAN channel.	
Format up to v7.2	`<Time> <Channel> <ID>x <Dir> d <DLC> <D0> <D1>...<D8> <MessageFlags>`
Format since v7.5	`<Time> <Channel> <ID>x <Dir> d <DLC> <D0> <D1>...<D8> Length = <MessageDuration> BitCount = <MessageLength> <MessageFlags>`
Format since v8.0	`<Time> <Channel> <ID>x <Dir> d <DLC> <D0> <D1>...<D8> Length = <MessageDuration> BitCount = <MessageLength> ID = <IDnum>x <MessageFlags>`
Example	`4.876870 1  54C5638x       Tx   d 8 00 00 00 00 00 00 00 00    Length = 1704000 BitCount = 145 ID = 88888888x`

CAN Remote Frame Event	
A CAN Remote Frame received or transmitted on a CAN channel.	
Format up to v7.2	`<Time> <Channel> <ID> <Dir> r`

CAN Remote Frame Event

Format since v7.5	`<Time> <Channel> <ID> <Dir> r Length = <MessageDuration> BitCount = <MessageLength> ID = <IDnum>x`
Format since v8.5	`<Time> <Channel> <ID> <Dir> r <DLC> Length = <MessageDuration> BitCount = <MessageLength> ID = <IDnum>x`
Example	`2.5010 1  200   Tx   r 8 Length = 1704000 BitCount = 145 ID = 88888888x`

CAN Error Frame

A CAN Error Frame received on a CAN channel.

Format up to v7.2	`<Time> <Channel> ErrorFrame`

Format since v7.5	CANcardXL, CANcaseXL, CANboardXL, and all other interfaces with SJA1000:
	
	`<Time> <Channel>   ErrorFrame ECC:<ECC>`
	
	Interfaces with CAN-Core:
	
	`<Time> <Channel> ErrorFrame Flags = <flags> CodeExt = <codeExt> Code = <code> ID = <ID> DLC = <DLC> Position = <Position> Length = <Length>`
	
	Flags:
	
	Bit field defining the validity of the parameters Code, CodeExt, ID, DLC, Position, and Length.
	Bit Meaning
	
	0 SJA 1000 ECC is valid
	1 Vector CAN Core Error Code is valid
	2 Vector CAN Core Error Position is valid
	4 Vector CAN Core Frame Length in ns is valid
	
	Code:
	
	Content of Philips SJA1000 Error Code Capture (ECC) register, or the Vector CAN-Core error register (see also mFlags).
	SJA1000-ECC
	See documentation of Philips SJA1000 CAN Controller.
	Vector CAN-Core
	Bit Meaning
	0-5 0: Bit Error
	　　　 1: Form Error
	　　　 2: Stuff Error
	　　　 3: Other Error
	　　　 4: CRC Error
	　　　 5: Ack-Del-Error
	　　　 7: Ack-Error
	6-7 0: RX-NAK-Error
	　　　 1: TX-NAK-Error
	　　　 2: RX-Error
	　　　 3: TX-Error

CAN Error Frame

CodeExt:

Extended error flags (see also the remarks below the table).

Bit	Meaning
0-4	Segment (only SJA1000)
5	Direction, 1=RX
6-11	Error Code 0 Bit Error 1 Form Error 2 Stuff Error 3 Other Error 4 CRC Error [1] 5 ACK-DEL Error [1] 7 ACK Error [1]
12-13	Extended Direction [1] 0 RX NAK 1 TX NAK 2 RX 3 TX
14	1 = The error frame was send from the application

Remarks

1 Only valid for interfaces with Vector CAN-Core (CANcardXLe, VN7600, VN16xx, VN89xx, and others).

2 The validity of ID, DLC, and the data field depends on the type and position of the disturbance. Example: If the message is disturbed in the ID-field, then only the first bits of the ID may be valid, but not the DLC and the data field. The error position is not the position of the disturbance. Example: If the error position is located in the CRC-field, then the message may have been disturbed in any other field, and the erroneous CRC is the result of that disturbance.

Examples	

	`1.592186 1  ErrorFrame  ECC: 10100010`
	`1.592186 2  ErrorFrame  Flags = 0xe  CodeExt = 0x20a2  Code = 0x82  ID`
	`= 0 DLC = 0 Position = 5 Length = 11300`

CAN Bus Statistics Event

CAN Statistic event, which contains statistic information about the CAN channels.
"D" stands for CAN Data Frames
"R" stands for CAN Remote Frames
"XD" stands for CAN Extended Data Frames
"XR" stands for CAN Extended Remote Frames
"E" stands for Error Frames
"O" stands for Overload Frames
"B" stands for Busload

CAN Bus Statistics Event

Format	`<Time> <Channel> Statistic: D <StatNumber> R <StatNumber> XD`
	`<StatNumber> XR <StatNumber> E <StatNumber> O <StatNumber> B`
	`<StatPercent>%`

Example	`  1.0100 1  Statistic: D 1000 R 15 XD 0 XR 0 E 0 O 0 B 0.0%`

CAN Error Event

An event that provides CAN error information.

Format	`<Time> CAN <Channel> Status:<Error>`
	`<Time> CAN <Channel> Status:<Error> - TxErr: <TxCount> RxErr: <RxCount>`
	
	TxCount:
	The value of transmit error count register
	RxCount:
	The value of receive error count register

Example	`0.0006 CAN 2 Status:chip status error active` `2137.317027 CAN 2 Status:chip status error active - TxErr: 8 RxErr: 0`

CAN Overload Frame Event

An Overload Frame received on a CAN channel.

Format	`<Time> <Channel> OverloadFrame`
Example	`  2.5158 1  OverloadFrame`

3.4 CAN FD

The section lists all Classic CAN events and CAN FD events on CAN FD channels in CANoe/CANalyzer ASC logging. See section 0 for an explanation of the symbols.

Note:

`<D1>` … `<D64>` refer to the data fields of the CAN FD message, opposed to standard CAN log format, this field is not fixed to 8 bytes, instead the length is depending on the value of the field `<Datalength>`. For example, when `<Datalength>` is 12, `<D1>` … `<D12>` is logged. For CAN remote frames the `<Datalength>` is 0.

`<Flags>` Bit field

   Bit 0: Reserved, must be 0
   Bit 1: Reserved, for internal use
   Bit 2: 1=NERR (1=single wire on low speed CAN)
   Bit 3: 1=High voltage wake up
   Bit 4: 1=Remote frame (only CAN)
   Bit 5: Reserved, must be 0
   Bit 6: 1= Tx Acknowledge
   Bit 7: 1= Tx Request
   Bit 8: Reserved, must be 0
   Bit 9: SRR (CAN FD)
   Bit 10: R0
   Bit 11: R1
   Bit 12: FDF bit   0: CAN frame 1: CAN FD frame
   Bit 13: BRS bit (CAN FD)
   Bit 14: ESI bit (CAN FD)
   Bit 15: Reserved, must be 0
   Bit 16: Reserved, must be 0
   Bit 17: 1= Frame is part of a burst
   Bit 18: Single shot mode: Frame could not be transmitted
   Bit 19: Single shot mode: If bit 18 set to 1, then this bit reports the reason.
       0 = arbitration lost 1=frame disturbed
   Bit 20: Reserved, for internal use
   Bit 21: Reserved, for internal use
   Bit 22 -31: Reserved, must be 0

`<CRC>` Checksum of the message.

   For CAN FD ISO-frames the stuff count and additional flags are stored in the field
   Bit 0 – 20: CRC
   Bit 21 – 26: Reserved for internal use
   Bit 27 – 29: Stuff count field
   Bit 30: Stuff count field parity
   Bit 31: ISO format. If set to 1, then the message is CAN FD ISO format, and the stuff count is valid.

`<BitTimingConfArb>`, `<BitTimingConfData>` refer to the bit timing parameters used for arbitration and data phase. These fields can be interpreted 32 bit long bit fields:

   Bit 0-7: Quartz Frequency in kHz
   Bit 8-15: Prescaler
   Bit 16-23: BTL Cycles
   Bit 24-31: Sampling Point
   These values are only available if supported by the hardware/driver, otherwise
   `<BitTimingConfArb>` and `<BitTimingConfData>` are 0.

> `<BitTimingConfExtArb>, <BitTimingConfExtData>`
>
> *CANoe/CANalyzer 8.5 and newer versions are supporting CAN FD ISO with extended bit timing capabilities. The bit timing will be stored in this fields. If the bit timings can not be stored in the old format, then* `<BitTimingConfArb>, <BitTimingConfData>` *will be set to 0.*
>
> *The extended format is:*
> *Bit 0 – 7: TSEG1-1*
> *Bit 8 – 15: TSEG2-1*
> *Bit 16 – 27: Prescaler*
> *Bit 28 – 31: Quartz Frequency (enumeration). Supported values: 0: 16 MHz, 1: 32 MHz, 2: 80 MHz*
> *These values are only available if supported by the hardware/driver, otherwise* `<BitTimingConfExtArb>` *and* `<BitTimingConfExtData>` *are 0.*

3.4.1 CAN Events on CAN FD channel

CAN Message Event

Simple CAN Message received or transmitted on a CAN FD channel.

Format since v8.1	`<Time> CANFD <Channel> <Dir> <ID> <SymbolicName> <BRS> <ESI> <DLC> <DataLength> <D1> … <D8> <MessageDuration> <MessageLength> <Flags> <CRC> <BitTimingConfArb> <BitTimingConfData> <BitTimingConfExtArb> <BitTimingConfExtData>`

Example	`0.105364 CANFD   1 Tx        1                                           0 0`
	 `1  1 01    112000    59    200040      39b5 46500250 460a0250 20011736`
	 `20010205`

CAN Extended Message Event

CAN Message with extended identifier received or transmitted on a CAN FD channel.

See CAN-FD Message description for information about the individual fields.

Format since v8.1	`<Time> CANFD <Channel> <Dir> <ID> <SymbolicName> <BRS> <ESI> <DLC> <DataLength> <D1> … <D8> <MessageDuration> <MessageLength> <Flags> <CRC> <BitTimingConfArb> <BitTimingConfData>> <BitTimingConfData> <BitTimingConfExtArb>  <BitTimingConfExtData>`

Example	`0.100995 CANFD   2 Rx     10001x                                      0 0`
	 `1  1 01    156000    82    200000      149a 46500250 460a0250 20011736`
	 `20010205`

CAN Remote Frame Event

A CAN Remote Frame received or transmitted on a CAN channel. The `<DataLength>` is set to 0 and the data bytes `<Dn>` are not available.

Format	`<Time> CANFD <Channel> <Dir> <ID> <SymbolicName> <BRS> <ESI> <DLC> <DataLength> <MessageDuration> <MessageLength> <Flags> <CRC> <BitTimingConfArb> <BitTimingConfData>> <BitTimingConfData> <BitTimingConfExtArb>  <BitTimingConfExtData>`

Example	`0.300981 CANFD   1 Tx     50005x                                      0 0`
	 `5  0    140000    73    200050      7a60 46500250 460a0250 20011736`
	 `20010205`

3.4.2 CAN FD Events

CAN FD Message Event	
CAN FD Message received or transmitted on a CAN FD channel.	
Format since v8.1	`<Time> CANFD <Channel> <Dir> <ID> <SymbolicName> <BRS> <ESI> <DLC> <DataLength> <D1> … <D64> <MessageDuration> <MessageLength> <Flags> <CRC> <BitTimingConfArb> <BitTimingConfData> <BitTimingConfExtArb> <BitTimingConfExtData>`
Example	`0.151061 CANFD   1 Tx            2                           0 0 2 2 02 03    150000   78    301040    10151 46500250 460a0250 20011736 20010205`

CAN FD Extended Message Event	
CAN FDMessage with extended identifier received or transmitted on a CAN channel. See CAN-FD Message description for information about the individual fields.	
Format since v8.1	`<Time> CANFD <Channel> <Dir> <ID> <SymbolicName> <BRS> <ESI> <DLC> <DataLength> <D1> … <D64> <MessageDuration> <MessageLength> <Flags> <CRC> <BitTimingConfArb> <BitTimingConfData> <BitTimingConfExtArb> <BitTimingConfExtData>`
Example	`0.105537 CANFD   2 Rx      10001x                            0 0 1 1 01    174015   91    301000       bfd9 46500250 460a0250 20011736 20010205`

CAN FD Error Frame
An Error Frame received on a CAN-FD channel. In case of certain errors (NACK Error, CRC Error) the hardware/driver may provide further information (ID, DLC, Data …) about the partial frame preceding the actual Error Frame, otherwise these values are 0. For further restrictions regarding the validity of these fields and a detailed description of the <flags>, <code> and <codeExt> fields, refer to the description of CAN Error Frame.

CAN FD Error Frame

Format	<Time> CANFD <Channel> <Dir> ErrorFrame <ErrorText> <flags> <code> <codeExt> <Phase> <Position> <ID> <BRS> <ESI> <DLC> <DataLength> <D1> … <D64> <MessageDuration> <extFlags> <CRC> <BitTimingConfArb> <BitTimingConfData> <BitTimingConfExtArb> <BitTimingConfExtData> <extFlags>:

Extended error flags (see also the remarks below the table).

Bit	Meaning
0	0: FDF is 0 (CAN Error Frame) 1: FDF is 1 (CAN FD Error Frame)
1	0: BRS is 0 1: BRS is 1
2	0: ESI is 0 1: ESI is 1
3	0: Error in Arbitration Phase 1: Error in Data Phase
4	0: Error is on a CAN channel 1: Error is on a CAN FD channel
all others	reserved, must be set to 0

| Example: Classic CAN | ```
0.051203 CANFD   1 Rx    ErrorFrame Stuff Error
fffe 82     20a2 Arb.   5        0  0 0 0  0      11984         10            0
46500250 460a0250 20011736 20010205
``` |
|---|---|
| Example: CAN FD | ```
10.006898 CANFD   1 Tx    ErrorFrame Not Acknowledge error, dominant
error flag       fffe c7     31ca Arb. 556       44   0 0 f 64 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00   1331984        11          0
46500250 460a0250 20011736 20010205
``` |

CAN FD Bus Statistics Event

Not defined yet, see CAN Statistics Event	
Format	
Example	

CAN FD Error Event

Not defined yet, see CAN Error Event	
Format	
Example	

CAN FD Overload Frame	
Not defined yet, see CAN Overload Frame	
Format	
Example	

3.5   Log and Trigger Events

The section lists all Log and Trigger events in CANoe/CANalyzer ASC logging. See section 0 for an explanation of the symbols.

Log Trigger Event (deprecated since v10.0)	
A Log Trigger event. There can be additional information appended at the end of the line, e.g. " (this trigger was in post trigger time of last block)" or " (ignored)".	
Format	`<Time>    log trigger event`
Example	`  2.0000 log trigger event`

Trigger Condition	
A Trigger condition event signals that an event caused a Trigger Block action (start, stop)	
Format	`<Time>    TriggerEvent: TriggerBlock[<Trigger Block Name>] <Trigger State> <Trigger Condition>`
Example	`11.400916 TriggerEvent: TriggerBlock[Logging] Start/Stop SingleTrigger`

Log Direct Start Event	
An event that is written if the logging was started directly by the button in the measurement setup or by the CAPL function StartLogging().	
Format	`<Time>    log direct start (<PreTrigger>ms)`
Example	`  2.1100    log direct start (2000ms)`

Log Direct Stop Event	
An event that is written if the logging was stopped directly by the buttons in the measurement setup or by the CAPL function StopLogging().	
Format	`<Time>    log direct stop (<PostTrigger>ms)`
Example	`  2.1100    log direct stop (1000ms)`

Begin Triggerblock Event	
An event that is written when a trigger block begins.	
Format	`Begin Triggerblock <WeekDay> <Month> <Date> <FullTime> <Year>`
Example	`Begin Triggerblock Mon Mar 7 01:21:51 pm 2005`

End Triggerblock Event	
An event that is written when a trigger block ends.	
Format	`End TriggerBlock`
Example	`End TriggerBlock`

3.6 Environment variables

The section lists the environment variable event in CANoe/CANalyzer ASC logging. See section 0 for an explanation of the symbol <time>. See this chapter for an explanation of the other symbols. The setting hex/dec affects the format of <value> for environment variables from type integer and data.

Environment Variables Event	
An event that is written if the value of a environment variable changed. `<evname>`: a string which contains the environment variable name `<value>`: the environment value as number, string or databytes (depend on variable type) OR a string from the value description table (if exists; only for integer variable type)	
Format	`<Time> <evname> := <value>`
Int	`2.130000    Int_Ev     := 1`
Float	`2.567000    Float_Ev   := -1.125`
String	`3.830000    String_Ev  := "Radio SWR3"`
Data	`2.250000    Data_Ev    := [ 41 41 41 41 ]`

3.7 System Variables

The section lists the system variable event in CANoe/CANalyzer ASC logging. See section 0 for an explanation of the symbol <time> and symbol <svtype>. See this chapter for an explanation of the other symbols. The setting hex/dec affects the format of <value> for system variables from type integer.

System Variables Event	
An event that is written if the value of a system variable changed. `<svtype>`: a number which represents the variable data type `<symbolic>`: whether the value may be the description from an assigned value table `<signed>`: whether the value is signed (only for integer type; ignored for other types) `<path>`: the full path (name with namespace) of the system variable `<value>`: the value as number or string (depend on variable data type). `<valuetype>`: (only for arrays) value type of elements: D: array of doubles, A: array of longs `<count>`: (only for arrays) size of array	
Format	`<Time> SV: <svtype> <symbolic> <signed> <path> = <value>`
Format (Arrays)	`<Time> SV: <svtype> <symbolic> <signed> <path> = <valuetype><count> <value>`

System Variables Event

Example int	`1.200000    SV: 2 0 1 ::NS1::IntVar = 4`
Float	`1.370000    SV: 1 0 1 ::NS1::FloatVar = 4.1`
String	`1.580000    SV: 3 0 1 ::NS1::StringVar = "Value: 4"`
Int array	`1.690000    SV: 5 0 1 ::NS1::IntArray = A3 4 5 2`
Float array	`2.000000    SV: 4 0 1 ::NS1::FloatArray = D3 4.1 2.9 6`

3.8 Macros: Signalevents

The section lists all Signal events for macros in CANoe/CANalyzer ASC logging. See section 0 for an explanation of the symbol <time>. See this chapter for an explanation of the other symbols.

Macro Signal Event: CAN, LIN and FlexRay

An event that is written if the user change a signal value with a panel control, and the macro recording is on.
`<bussystem>:` F = FlexRay / L = Lin / nothing = CAN
`<channel>:` the number of the CAN/LIN/FlexRay channel
`<node>:` a string which contains the node name of the signal
`<message>:` a string which contains the message name of the signal
`<signal>:` a string which contains the signal name
`<value>:` the signal value as number OR a string from the value description table (if exists)

Format	`<Time> <bussystem> <channel> <node>::<message>::<signal> = <value>`
CAN (value as number)	`2.350000 1 Node::aCANMessage::aBitSignal = 1`
CAN (symbolic value)	`2.350000 1 Node::aCANMessage::aBitSignal = Eins`
LIN	`5.000000 L1 L_Slave::aLINMessage::aLINBitSignal = 0`
FlexRay	`1.110000 F1 FR_ECU::aFlexRayMessage::aFRSignal = 3`

3.9 GPS events

The section lists the GPS event in CANoe ASC logging. See section 0 for an explanation of the symbol <time>. See this chapter for an explanation of the other symbols.

GPS Event

An event that is written if an event is received on the GPS channel.
<channel>: the number of the GPS device on which the event is received
<latitude>: the latitude value of the GPS event
<longitude>: the longitude value of the GPS event
<altitude>: the altitude value of the GPS event
<speed>: the speed value of the GPS event
<course>: the course value of the GPS event

Format	`<Time> GPS device: <channel> La:<latitude> Lo: <longitude> Alt:` `<altitude> Sp: <speed> Co: <course>`

GPS Event	
Example	`2.097603 GPS-Device: 1    La: 48.825100    Lo: 9.091267    Alt:` `325.399994    Sp: 29.686400    Co: 87.099998`

3.10 Comment events

The section lists the Comment event in CANoe ASC logging. See section 0 for an explanation of the symbol <time>.

Comment Event	
A comment event that is written before another event that was commented in Trace Window. Commenting events is supported only in Trace Window, so this event can be written only during the export from Trace window. <type>: the type of the commented event <comment text>: the text of the comment	
Format	`<Time> Comment: <type> <comment text>`
Example	`1.593770    Comment: 105 testComment`

3.11 Global marker events

The section lists the Global marker event in CANoe ASC logging. See section 0 for an explanation of the symbol <time>.

Comment Event	
A global marker event that is written if global marker is defined for a time stamp or for another event. If a global marker event is assigned to another event (set in Trace Window) it has to be written before that event. Global marker events can be written only during the export from Trace window. <type>: the type of the commented event <background color>: background color of the marker group <foreground color>: foreground color of the marker group <relocatable>: defines whether the marker can be moved <group name>: the name of the marker group <marker name>: the name of the marker <description> marker description	
Format	`<Time> <type> <background color> <foreground color> GMGroup: <group name> GMMarker: <marker name> GMDescription: <description>`
Example	`2.200804    0 16777215 0  1 GMGroup: Marker Group GMMarker: [1] GMDescription: description`

3.12   Test structure events

This section lists the Test Structure event in CANoe ASC logging. See section 0 for an explanation of the symbol <time>.

Test Structure Event	
Events produced during execution of Test Modules or Test Configurations in CANoe. For each start or end of a structural element in the test, e.g. TestCase or TestGroup, a matching event is produced. An additional abort event is written when a test is aborted due to its verdict impact setting or a user stop.	
When the ASC log file is exported from the CANoe Trace Window, only the events visible there are exported, i.e. start/end of Test Groups etc. are not written.	

<execID>:	Unique ID for the executing Test Configuration or Test Module. This ID can be used to correlate all VBLTestStructure events during one measurement; it's not persistent across measurements.
<elementID>:	Unique ID for the structure element (Test Case etc.), can be used to correlate events belonging to the same element and to disambiguate between elements with the same name. 8-digit hex value. 0 for Test Module and Test Configuration root elements. This value is not persistent across measurements. (Currently not supported for Elements not visible in the CANoe GUI, e.g. Test Cases inside a Test Sequence, these always have a value of 0xFFFFFFFF)
<event text>:	Text of the event as shown in the CANoe Trace Window.
<verdict>:	For end events the overall verdict of the element.

Format begin event	`<time> TFS: [<execID>,<elementID>] <event text>`
Example	`2.679062 TFS: [00000001,00000003] Test configuration 'MyTest', Test unit 'TU1': Test case 'TC1' started.`
Format end event	`<time> TFS: [<execID>,<elementID>] <verdict> <event text>`
Example	`4.465270 TFS: [00000001,00000003] Passed: Test configuration 'MyTest', Test unit 'TU1': Test case 'TC1' finished.`
Format abort event	`<time> TFS: [<execID>,<elementID>] <verdict> <event text>`
Example	`4.462292 TFS: [00000003,00000000] Failed: Test module 'Tester: Test execution aborted due to stop of the test module. Test is incomplete!`

3.13 Symbols

Symbol	Width in chars (hex)	Width in chars (dec)	Meaning	Range	Example	Special
`<Time>`		>=9	absolute or relative time in seconds		1234.5678	usually 4 decimal places
`<Channel>`		1 – 3	Number of the CAN channel.	1..255		
		5 – 10	CAN/CAN FD channel	"CAN "1..255 "CAN FD "1..255		
`<ID>` (numeric logging)	15	15	Numeric identifier	Normal mode: 0x0…0x7FF Extended Mode: 0x0…0x1FFFFF		In extended mode only a width of 14 characters, because of 'x' at the end. CAN FD representation only in hex.
`<ID>` (symbolic logging)	>= 32	>=32	Symbolic identifier Differences to numeric format: 1. Four spaces instead of one after the Identifier 2. Length of <ID> is >=32. For short symbolic names (<32) the remaining positions are filled with spaces.			
`<Dir>`		4	direction of transmission	Rx, Tx, TxRq	Rx	
`<DLC>`	1	2	data length code Note: the DLC field is written as one hex digit. In case DLC is greater 8, there are max. 8 data bytes written/read in.	Hex: 0…F Dec: 0…15	B 12	CAN FD representation is always dec
`<MessageFlags>`	2	2	Some special message flags that are written at the end of a logging	TE, WU, XX	TE	

Symbol	Width in chars (hex)	Width in chars (dec)	Meaning	Range	Example	Special
			line. Possible values are: - "TE": Transmission Error (NERR signal). Indicates whether a line has failed during a two-wire operation. Especially available on Single-Wire mode. - "WU": Wake-Up. Indicates whether a message was transmitted with overvoltage with the purpose of waking up the CAN controller. - "XX": Both, "TE" and "WU" occurred.			
`<StatNumber>`		1-10	the number of received statistic events	0…4294967295	1056	
`<StatPercent>`		3-6	the busload in percent	0.0 … 100.0	11.94	
`<Trigger Block Name>`			name of the Trigger which generated the Trigger Condition Event.		"Logging"	
`<Trigger State>`			State or action resulting from the Trigger Condition		"Start", "Stop", "Start/ Stop"	
`<Trigger Condition>`			Trigger Condition description		"SingleTri gger"	
`<PreTrigger>`		1-10	the pre trigger time in ms	0 … 1316134911	2000	
`<PostTrigger>`		1-10	the post trigger time in ms	0 … 1316134911	2000	
`<Error>`		0-undefin ed	the error message of the CAN error event		"rx queue overrun"	
`<Dx>`	3	2	data byte x	0…255	1E	CAN FD only hex representati

Symbol	Width in chars (hex)	Width in chars (dec)	Meaning	Range	Example	Special
						on
`<WeekDay>`	3	3	a string that represents a day of the week	Mon, Tue, Wed, Thu, Fri, Sat, Sun	Mon	Different range in German version: Mon, Die, Mit, Don, Fre, Sam, Son
`<Month>`	3	3	a string that represents a month	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec	Nov	Different range in German version: Jan, Feb, März, Apr, Mai, Jun, Jul, Aug, Sep, Okt, Nov, Dez
`<Date>`	1-2	1-2	a number that represents the date.	1…31	15	
`<FullTime>`[1]	15	15	a string that represents a time in the current format. hh:mm:ss.ms am|pm	00:00:00.000 … 12:60:60.999	01:13:17.123 pm	Different range in German version: 00:00:00.000 … 23:60:60.999 In German version 'am' and 'pm' are not used. Therefore the width is only 12 chars.
`<Year>`	4	4	a string that represents a year.		1999	
`<svtype>`	1	1	a number that	1…5	2	

[1] **Starting with CANalyzer/CANoe 8.2 SP2 the <Fulltime> is stored in milliseconds. This can be disabled by setting the flag ASCII_Format_Milliseconds = 0 in section [SYSTEM] of the CAN.INI file. In that case the string-width is 4 characters smaller.**

Symbol	Width in chars (hex)	Width in chars (dec)	Meaning	Range	Example	Special
			represents the variable data type of system variable: 1 = Float 2 = Integer 3 = String 4 = Array of Floats 5 = Array of Integers			
`<Message-Duration>`			Message duration [in nanoseconds]. Not including 3 Interframe Space bit times and by Rx-messages also not including 1 End-Of-Frame bit time		768000	
`<Message-Length>`			Total number of bits of the message including EOF and Interframe Space [in bits]		67	
`<BRS>`	1		Bit rate switch flag for CAN FD. Indicate bit rate switch is enabled or disabled.	0-1	0	
`<ESI>`	1		Error state indicator flag for CAN FD. Indicate a transceiver is in error active or error passive mode.	0-1	0	
`<DataLength>`		3	Valid length of the message in bytes.	0-64	20	
`<BitTiming-ConfArb>`	8		Bit timing information for CAN-FD frames.			May be 0, if not supported by the CAN-controller
`<BitTiming-ConfData>`	8		Bit timing information for CAN-FD frames			May be 0, if not supported by the CAN-controller