Telemotive AG

blue PiraT / blue PiraT 2
Specification Telemotive ASCII Format

Version: 1.4.1

Date: 29 January 2014

Author: Markus Heininger
Helmut Sochor
Robert Schwabe
Markus van Pinxteren

Change history

Version /Date	Section	Change	Author	Version relevance		
				1	2	3
1.0.0 04.05.2011		First version	MHe	**x**		
1.0.1 10.05.2011		Added MOST	HSo RSc		**x**	
09.08.2012		Bugfixes, Layoutupdate New FlexRay frame types	Mvp		**x**	
04.09.2012		**Document release 1.2.1 for Client 1.7.2**				
27.02.2013	3.4	Added error field for CAN messages with valid data.	mvp		**x**	
05.03.2013	3.11	Added DLC to LIN-Table	mvp		**x**	
08.03.2013	3.23	Added Light and Lock information	mvp		**x**	
06.06.2013		**Document release 1.3.1 For Client 1.8.1**				
01.10.2013	3.11	Added pure LIN status frame Added wakeup frame	mvp		**x**	
04.12.2013	3.4	Set CAN-Id field width to 8 (with leading zeros)	mvp			**x**
13.12.2013	3.13	Fixed time stamp format in MarkerMessage payload	mvp			**x**
29.01.2014		**Document release 1.4.1 For Client 1.9.1**				

Contents:

1. The document's purpose

1.1. General

This document describes the composition of a blue PiraT / blue PiraT 2 Telemotive trace file with ASCII format.

1.2. Differences between blue PiraT and blue PiraT 2

There may be differences between the message structures of blue PiraT traces and blue PiraT 2 traces. Where it is possible the implementation of the Telemotive ASCII format is kept equal for both product, where not, the differences are listed in the appropriate message section of this document.

2. File Structure

A Telemotive ASCII trace file contains the ASCII representation of trace messages from converted tmt/xtmt files. When converting to the Telemotive ASCII format the messages from the binary files (tmt/xtmt) can be filtered on the messages' types to get only a subset of all messages.

Telemotive ASCII files converted with the blue PiraT 2 Client software always contain a "SYSTEM MSG" with information of the used format version as first message in the file. After that the messages of the converted binary files follow in chronological order one message per line. As separator the OS's common line ending is used during conversion ('\r\n' for MSW and '\n' for Linux)

3. Message structure

3.1. General message structure

A message has the following general structure:

```
<time stamp> <message type> <channel no.> <payload>
```

`<channel no.>` and `<payload>` are optional depending on the message type.

Example: "09.08.2012 10:57:03.7591 CAN #2 | Rx   005 4 31 32 33 34"

Note: *The number of whitespaces between different parts of the message is not strictly defined.*

3.1.1. Time stamp

The time stamp contains always the local time and is written in this format:
`dd.mm.yyyy hh:mm:ss.xxxx`

Example: 09.08.2012 10:57:03.7591

3.1.2. Message type

The message type indentifies the message with capital letters and can consist of several words.

Example: "SYSTEM MSG"

3.1.3. Channel No.

The channel number is optional. It starts with the '#' character followed by the channel index.

Examples: #2 or #1A (FlexRay)

3.1.4. Payload

The payload is optional. When it exists it is separated from the message's front part by the '|' character.

Example: "| Rx   005 4 31 32 33 34"

The following chapters describe the composition of the messages depending on the message's type.

3.2. Analog Trace Message

Examples:

```
04.05.2011 07:32:06.3194  ANALOG DATA | port = 1, direction = In, data = 123

04.05.2011 07:32:06.3194  ANALOG DATA | (port = 1, direction = In, data =
1,2V) (port = 2, direction = In, data = 0,8A)
```

Message type:

```
ANALOG DATA
```

Payload:

```
port = <port>, direction = <dir>, data = <data><unit>
```

Field	Possible values	Description
<port>	*Numeric value (dec)*	Port number
<dir>	In Out Unkown	Direction
<data>	*Numeric value (hex)*	Measured value
<unit>	 V A	No unit Volt Ampere

3.3. CAN Trace Message

Examples:

```
09.08.2012 10:57:03.7591  CAN #2       | Rx   005  4 31 32 33 34
09.08.2012 10:57:03.7591  CAN #2       | Error Frame [error= ACKNOWLEDGE]
```

Message type:

```
CAN
```

Payload:

```
<direction> [error= <errtype>] <canID> <dlc> <data>
```

or

```
Error Frame [error= <errtype>]
```

Field	Possible values	Description
<direction>	Rx Tx TxRq	received message transmitted message Tx request
<dlc>	*Numeric value (dec)*	Data length code
<canID>	*Numeric value (hex)*	CAN identifier
<data>	*Numeric values (hex) (separated by space)*	CAN data bytes
<errtype>	NO STUFF FORMAT ACKNOWLEDGE BIT1 BIT0 CRC OVERRUN	Status ok Stuff error Format error Acknowledge error Bit 1 error Bit 0 error CRC error Overrun error

The field `[error=<errtype>]` is optional for standard frames. For error frames it is mandatory

3.4. CAN Extended Trace Message

Examples:

```
09.08.2012 10:57:03.7591  CANExt #3   | EXTENDED Rx   15070055  4 12 34 56 78
09.08.2012 10:57:03.7591  CANExt #2   | EXTENDED Error Frame [error=STUFF]
```

Message type:

```
CANExt
```

Payload:

```
EXTENDED <direction> [error= <errtype>] <canID> <dlc> <data>
```

or

```
EXTENDED Error Frame [error= <errtype>]
```

Field	Possible values	Description
<direction>	Rx Tx TxRq	received message transmitted message Tx request
<dlc>	*Numeric value (dec)*	Data length code
<canID>	*Numeric value (hex)*	CAN identifier
<data>	*Numeric values (hex) (separated by space)*	CAN data bytes
<errtype>	NO STUFF FORMAT ACKNOWLEDGE BIT1 BIT0 CRC OVERRUN	Status ok Stuff error Format error Acknowledge error Bit 1 error Bit 0 error CRC error Overrun error

The field `[error=<errtype>]` is optional for standard frames. For error frames it is mandatory

3.5. ECLMessage

Example:

```
30.08.2011 13:06:54.8102  ECL MESSAGE | [ECL_STP] Parameter: 0x02 - 550066
30.08.2011 13:06:50.2601  ECL MESSAGE | [ECL_STWU] 199956
30.08.2011 13:06:55.5101  ECL MESSAGE | [ECL_STR] Node: 0x03; E: 0; O: 0 -
499966
```

Message type:

```
ECL MESSAGE
```

Payload:

```
Type: ECL_EWU, ECL_STWU, ECL_UNDEF_PULSE
[<type>] <txTime>

Type: ECL_STP
[<type>] Parameter: <param> - <txTime>

Type: ECL_STR
[<type>] Node: <node>; E: <e>; O: <o> - <txTime>
```

Field	Possible values	Description
<type>	ECL_EWU ECL_STWU ECL_STP ECL_STR ECL_UNDEF_PULSE	 Electrical WU symbol System test WU symbol System test paramters System test result

		Undefined pulse
<txTime>	*Numeric value (dec)*	Transmission time of Symbol
<param>	*Numeric value (hex)*	System Test Parameters Sequence P1-P5
<node>	*Numeric value (hex)*	Node Class
<e>	1 0	Alive Result
<o>	1 0	MOST Signal Result

3.6. End Of File Message

Examples:

```
15.12.2011 09:36:58.3082  EOF          | CRC = 0x00000000
```

Message type:

```
EOF
```

Payload:

```
CRC = <crc>
```

Field	Possible values	Description
<crc>	*Numeric value (hex), preceded by 0x*	The file's checksum, currently not used – always 0

3.7. Ethernet Trace Message

Examples:

```
09.08.2012 10:57:03.7591  ETHERNET #1 | TX [RAW] - 01 02 03
09.08.2012 10:57:03.7591  ETHERNET #1 | RX [RAW] - 01 02 03
09.08.2012 10:57:03.7591  ETHERNET #3 | RX [DLT-BMW] - ecuID=ABCD, 01 02 03
09.08.2012 10:57:03.7591  ETHERNET #3 | TX [DLT-BMW] - ecuID=ECU1, 01 02 03
```

Message type:

```
ETHERNET
```

Payload:

```
<rxtx> [<protocol>] - <data>
```

Field	Possible values	Description
<rxtx>	RX TX	received message transmitted message
<protocol>	GNLOGGER RAW UTF8 DLT-BMW UDPSERVER EsoTrace SpyMode	
<data>	*Numeric values (hex) (separated by space)*	Protocol specific raw data as byte array For DLT protocol preceded by `ecuID=<ecuid>` with <ecuid> = four character ID

3.8. FlexRay Trace Message

Examples:

```
09.08.2012 10:57:03.7591  FLEXRAY #1B     | status= Frame Data , bits=5,
slot=7, len=120, hCRC=0x0009, cycle=10, payload:
0000,0004,0008,000c,0010,0014,0018,001c,0020,0024,0028,002c,0030,0034,0038

09.08.2012 10:57:03.7591  FLEXRAY #1B     | type = Invalid, data: 05 07 F0 02
4A 00 00 00 04 – raw: 08 15 a5 44 06 1f 12 08 04

09.08.2012 10:57:03.7591  FLEXRAY #1B     | type = <type>
```

Message type:
FLEXRAY

Channel no.:
1A | 1B | 2A | 2B

Payload:

```
status= <status> , bits=<bits>, slot=<slot>, len=<length>, hCRC=<hCRC>, cy-
cle=<cycle>, payload: <payload>

type = Invalid, data: <data> - raw: <raw>
type = <type>
```

Field	Possible values	Description
<status>	Frame Data FIFO A Ovfl FIFO B Ovfl Error Sync No Sync Startup RAW	Frame status
<bits>	*Numeric value (dec)*	frame indicator
<slot>	*Numeric value (dec)*	slot identifier
<length>	*Numeric value (dec)*	frame length
<hCRC>	*Numeric value (hex - preceded by '0x')*	header CRC
<cycle>	*Numeric value (dec)*	cycle identifier
<payload>	*Numeric values (hex – separated by ',')*	data in 16 bit presentation
<type>	Invalid WUS CAS MTS CAS/MTS WUS or CAS/MTS Undefined Low	Invalid message received Wakeup Symbol Collisions Avoidance Symbol Media Access Test Symbol CAS or MTS received (ambiguous) WUS, CAS or MTS received (ambiguous) Undefined Low Pulse (neither symbol nor TSS)
<data>	*Numeric values (hex – separated by space)*	The valid part of the received frame data (data transmission was conform to flexray specification)
<raw>	*Numeric values (hex – separated by space)*	The invalid part of the received frame data (data transmission violated flexray specification)

3.9. GpioTraceMessage

```
22.08.2012 20:12:56.5260  GPIO DATA   | port =   2, dir = In , mask = 0xffff,
data = 000000
```

```
22.08.2012 20:12:56.5260  GPIO DATA   | (port =   1, dir = In , mask =
0xffff, data = 000000) (port =   2, dir = In , mask = 0xffff, data =
0x000012)
```

Message type:
GPIO DATA

Payload:
port = <port>, direction = <dir>, mask = <mask>, data = <data><unit>

Field	Possible values	Description
<port>	*Numeric value (dec)*	Port number
<dir>	In Out Unkown	Direction
<mask>	*Numeric value (hex)*	
<data>	*Numeric value (hex)*	Measured value

3.10. GPS Message (only blue PiraT 2)

Example:
```
04.06.2012   14:47:31.9935       GPS                                  |   $GPG-
GA,144952.0,4811.226298,N,01136.152502,E,1,09,0.9,504.3,M,47.0,M,,*5F
```

Message type:
GPS

Payload:
<nmea>

Field	Possible values	Description
<nmea>		NMEA data set

3.11. LIN Trace Message

Example:
```
09.08.2012 10:57:03.7591  LIN #2      | [status=2, bitTime=3]

09.08.2012 10:57:03.7591  LIN #2      | [status=1, bitTime=3, wakeUpPulse=52]

09.08.2012 10:57:03.7591  LIN #2      | [status=2, bitTime=3, frameTime=4,
breakTime=5, delimiterTime=6, headerTime=7, linId=8, len=8]  f0 e1 d2 c3 b4
a5 96 87
```

Message type:
LIN

Payload:
```
For pure status frames:
[status=<status>, bitTime=<bitTime>]

For wakeup frames:
[status=<status>, bitTime=<bitTime>, wakeUpPulse=<wakeUpPulse>]

For standard LIN frames:
[status=<status>, bitTime=<bitTime>, frameTime=<frameTime>, break-
Time=<breakTime>, delimiterTime=<delimiterTime>, headerTime=<headerTime>, li-
nId=<linId>, len=<len>] <data>
```

Field	Possible values	Description
<status>	*Numeric value (dec)*	Bit 0: wakeup frame Bit 3: invalid data Bit 4: LIN_ERR_BREAK (error in break field) Bit 5: LIN_ERR_SYNC (error in sync pattern) Bit 6: LIN_ERR_IDENTIFIER (error in protected identifier) Bit 7: general error bit – invalid LIN telegram
<bitTime>	*Numeric value (dec)*	time of one bit (1/Baudrate) in µs
<wakeUpPulse>	*Numeric value (dec)*	wake up pulse time in µs
<frameTime>	*Numeric value (dec)*	total frame duration in µs
<breakTime>	*Numeric value (dec)*	sync break duration in µs
<delimiterTime>	*Numeric value (dec)*	break delimiter duration in µs
<headerTime>	*Numeric value (dec)*	header duration in µs
<linId>	*Numeric value (dec)*	Protected Identifier Bit 0…5: ID Bit 6…7: Parity
<len>	*Numeric value (dec)*	Number of following data bytes
<data>	*Numeric values (hex) (separated by space)*	Payload as byte array

3.12. LostMessage

Example:

```
15.12.2011 09:36:58.3082  LOST SEND   | [MOST150] [CTRL] Start time:
15.12.2011 09:36:56.1457  Stop time: 15.12.2011 09:36:58.0024  Number of
failed Send-Msg: 16
```

Message type:

```
LOST SEND
```

Payload:

```
[<device>] [<type>] Start time: <start> Stop time: <Stopp> Number of failed
Send-Msg: <num>
```

Field	Possible values	Description
<device>	MOST50 MOST150	
<type>	CTRL NET MDP MEP	Control message Net state MOST Data Packet MOST Ethernet Packet

3.13. Marker Message

Example:

```
09.08.2012 10:57:03.7591  MARKER      | #466  08-09-2012 10:57:03.759132
```

Message type:

```
MARKER
```

Payload:

```
#<markerId> <timestamp>
```

Field	Possible values	Description
<markerId>	*Numeric value (dec)*	Marker number

<timestamp>	*string*	Marker time stamp, same as message time stamp but with µs resolution

3.14. MOST 25 Trace Status Message

Examples:

```
04.05.2011 07:32:06.3194  MOST STATUS | Light on - Lock - SBC: 14 - MPR: 22
```

Message type:

```
MOST STATUS
```

Payload:

blue PiraT:

```
<light> - <lock> - SBC: <sbc> - MPR: <mpr> - MDR: <mdr> - NPR: <npr> - NDR
<ndr>
```

blue PiraT 2:

```
<light> - <lock> - SBC: <sbc> - MPR: <mpr>
```

blue PiraT: The parameters are only written if the value changed. If nothing changed the payload is "`[reserved for future use]`".
blue PiraT 2: The parameters are always written.

Field	Possible values	Description
<light>	Light on Light off	
<lock>	Lock Unlock	
<sbc>	*Numeric value (dec)*	Synchronous Bandwidth Control
<mpr>	*Numeric value (dec)*	Maximum Position register
<mdr>	*Numeric value (dec)*	Maximum delay register
<npr>	*Numeric value (dec)*	Node Position Register of the previous Node
<ndr>	*Numeric value (dec)*	Node Delay Register

3.15. MOST 25 Trace Control Message

Examples:

```
04.05.2011 07:32:06.3194  MOST CTRL | [0100 -> 0400] Type = 04 (00 7F C0 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00)
04.05.2011 07:32:06.3194  MOST CTRL | [0100 -> 0401] . 01.01 . 000.1 . 0 0 ()
04.05.2011 07:32:06.3194  MOST CTRL | [0100 -> 0401] [CRC_ERROR] . 01.01 .
000.1 . 0 0 ()
```

Message type:

```
MOST CTRL
```

Payload:

Standard messages (Message Type = 0):

```
[<source> -> <target>] [<err>]. <FBlockID>.<InstID> . <FctID>.<OpCode> .
<MsgType> <length> (<data>)
```

Other messages:

```
[<source> -> <target>] [<err>] Type = <MsgType> (<data>)
```

The error field `[<err>]` is optional and only available for blue PiraT traces. blue PiraT 2 doesn't support this value.

Field	Possible values	Description
`<source>`	*Numeric value (hex)*	Source address
`<target>`	*Numeric value (hex)*	Target address
`<err>`	INVALID_ARBITRATION WRONG_PARITY CRC_ERROR	Error type, optional and only for blue PiraT. If no error occurred the surrounding brackets[ ] will not be written.
`<FBlockID>`	*Numeric value (hex)*	Function block ID
`<InstID>`	*Numeric value (hex)*	Instance ID
`<FctID>`	*Numeric value (hex)*	Function ID
`<OpCode>`	*Numeric value (hex)*	OpCode
`<MsgType>`	*Numeric value (hex)* 1 – RemoteRead 2 – RemoteWrite 3 – Alloc 4 – Dealloc 5 – RemoteGetSource	Type
`<length>`	*Numeric value (hex)*	Number of following bytes
`<data>`	*Numeric values (hex) separated by space char*	Data bytes Standard: up to 12 bytes Other: all bytes beginning at FBlockID up to CRC

3.16. MOST25 Asynch

For blue PiraT 2 the payload length of asynchronous messages can be limited by the logger configuration. Such shorted messages are marked in their payload with the prefix "Cut message - " and are described separately here.

3.16.1. MOST25 Trace Asynch Message

Example:

```
04.05.2011 07:32:06.3194  MOST ASYNCH | [0405 -> 0102] . 0A (06 07 08 09 0A
0B 0C 0D 0E 0F)
```

Message type:

```
MOST ASYNCH
```

Payload:

```
[<source> -> <target>] [<err>] . <length> (<data>)
```

The error field `[<err>]` is optional and only available for blue PiraT traces. blue PiraT 2 doesn't support this value.

Field	Possible values	Description
`<source>`	*Numeric value (hex)*	Source address
`<target>`	*Numeric value (hex)*	Target address
`<err>`	INVALID_ARBITRATION WRONG_PARITY CRC_ERROR	Error type, optional and only blue PiraT. If no error occurred the surrounding brackets[ ] will not be written.
`<length>`	*Numeric value (hex)*	Number of following bytes
`<data>`	*Numeric values (hex) separated by space char*	Data bytes

3.16.2. Shorted MOST25 Trace Asynch Message (only blue PiraT 2)

Example:

```
04.05.2011 07:32:06.3194  MOST ASYNCH | Cut message - target(0100),
source(0400), data(31.32), numCutBytes(45)
```

Message type:

```
MOST ASYNCH
```

Payload:

```
Cut message - target(<target>), source(<source>), data(<data>), num-
CutBytes(<ncb>)
```

Field	Possible values	Description
<target>	*Numeric value (hex)*	Target address
<source>	*Numeric value (hex)*	Source address
<data>	*Numeric values (hex) separated by space char or "-"*	Data bytes "-" if not available

3.17. MOST25 Allocation Map Message (only blue PiraT)

Example:

```
27.05.2007 15:04:39.1421  MOST ALLOC  | 00 00 00 00 70 70 70 70 70 70 70 70
70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70
70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 70 06 06 00 00
```

Message type:

```
MOST ALLOC
```

Payload:

```
[<err>] <allocmap>
```

The allocation map message is only available for blue PiraT traces. blue PiraT 2 doesn't support this message type.

Field	Possible values	Description
<err>	INVALID_ARBITRATION WRONG_PARITY CRC_ERROR	Error type, optional. If no error occurred the surrounding brackets[ ] will not be written.
<allocmap>	*Numeric values (hex) separated by space char*	Allocation Map

3.18. MOST 50 Trace Status Message (only blue PiraT)

Examples:

```
09.02.2012 18:45:42.2651  M50 STATUS  | MPR(04), Boundary(04), status1(03),
freeBytes(00), channelWidth(00), channelLabel(00)09.02.2012 18:45:42.2651

M50 STATUS  | [02e9, b485] MPR(04), Boundary(04), status1(03), freeBytes(00),
channelWidth(00), channelLabel(00)
```

Message type:

```
M50 STATUS
```

Payload:

```
[<msgCnt>, <msgCrc>] MPR(<mpr>), Boundary(<bound>), status1(<stat1>), free-
Bytes(<fb>), channelWidth(<cw>), channelLabel(<cl>)
```

MOST50 messages are only available for blue PiraT traces. blue PiraT 2 doesn't support MOST50.

Field	Possible values	Description

`[<msgCnt>, <msgCrc>]`
 	
`<msgCnt>`
`<msgCrc>`
	
	Numeric value (hex)
	Numeric value (hex)
		Telemotive internal message counter and CRC, the entire field is optional (inclusive brackets).

`<mpr>`	*Numeric value (hex)*	Maximum Position register
`<bound>`	*Numeric value (hex)*	Boundary value
`<stat1>`	*Numeric value (hex)*	Network status
`<fb>`	*Numeric value (hex)*	Number of free bytes
`<cw>`	*Numeric value (hex)*	Channel width
`<cl>`	*Numeric value (hex)*	Channel Label

3.19. MOST50 Trace Control Message (only blue PiraT)

Examples:

```
09.02.2012 18:45:42.4830  M50 CTRL    | [02f3, 7374] prio(0e), target(0101),
preAck(04), packetNum(03), source(0100), crc(8dd2), cack(04), ack(0a), da-
ta(31.01.10.12.0c.00.00.00.00.00.02.08.09.0a.0b.0c.0d)

09.02.2012 18:45:42.4830  M50 CTRL    | prio(0e), target(0101), preAck(04),
packetNum(03), source(0100), crc(8dd2), cack(04), ack(0a), da-
ta(31.01.10.12.0c.00.00.00.00.00.02.08.09.0a.0b.0c.0d)
```

Message type:

```
M50 CTRL
```

Payload:

```
09.02.2012 18:45:42.4830  M50 CTRL    | [<msgCnt>, <msgCrc>] prio(<prio>),
target(<target>), preAck(<preAck>), packetNum(<packetNum>), source(<source>),
crc(<crc>), cack(<cack>), ack(<ack>), data(<data>)
```

MOST50 messages are only available for blue PiraT traces. blue PiraT 2 doesn't support MOST50.

Field	Possible values	Description

`[<msgCnt>, <msgCrc>]`
 	
`<msgCnt>`
`<msgCrc>`
	
	Numeric value (hex)
	Numeric value (hex)
		Telemotive internal message counter and CRC, the entire field is optional (inclusive brackets).

`<prio>`	*Numeric value (hex)*	Priority
`<target>`	*Numeric value (hex)*	Target address
`<preAck>`	*Numeric value (hex)*	Preemptive Acknowledge
`<packetNum>`	*Numeric value (hex)*	Packet counter

<source>	*Numeric value (hex)*	Source address
<crc>	*Numeric value (hex)*	CRC
<cack>	*Numeric value (hex)*	CRC Acknowledge
<data>	*Numeric values (hex) separated by '.'*	Data bytes

3.20. MOST50 MDP (only blue PiraT)

The payload length of MOST50 data messages can be limited by the logger configuration. Such shorted messages are marked in their payload with the prefix "Cut message - " and are described separately here.

3.20.1. MOST50 Trace Data Message

Examples:

```
04.05.2011 07:32:06.3194  M50 DATA   | [02f8, 2f6f] target(0100), preAck(06),
packetNum(07), source(0400), crc(cdef), cack(10), ack(00), da-
ta(31.32.33.34.35.36.37.38)

04.05.2011 07:32:06.3194  M50 DATA   | target(0100), preAck(06), pack-
etNum(07), source(0400), crc(cdef), cack(10), ack(00), da-
ta(31.32.33.34.35.36.37.38)
```

Message type:

```
M50 DATA
```

Payload:

```
[<msgCnt>, <msgCrc>] target(<target>), preAck(<preAck>), pack-
etNum(<packetNum>), source(<source>), crc(<crc>), cack(<cack>), ack(<ack>),
data(<data>)
```

MOST50 messages are only available for blue PiraT traces. blue PiraT 2 doesn't support MOST50.

Field	Possible values	Description
[<msgCnt>, <msgCrc>] <msgCnt> <msgCrc>	 *Numeric value (hex)* *Numeric value (hex)*	Telemotive internal message counter and CRC, the entire field is optional (inclusive brackets).
<target>	*Numeric value (hex)*	Target address
<preAck>	*Numeric value (hex)*	Preemptive Acknowledge
<packetNum>	*Numeric value (hex)*	Packet counter
<source>	*Numeric value (hex)*	Source address
<crc>	*Numeric value (hex)*	CRC
<cack>	*Numeric value (hex)*	CRC Acknowledge
<ack>	*Numeric value (hex)*	Acknowledge
<data>	*Numeric values (hex) separated by '.'* *or "-"*	Data bytes "-" if not available

3.20.2. Shorted MOST50 Trace Data Message

Examples:

```
04.05.2011 07:32:06.3194  M50 DATA   | [02f8, 2f6f] Cut message - tar-
get(0100), preAck(06), packetNum(07), source(0400), crc(-),cack(-) da-
ta(31.32), numCutBytes(45)

04.05.2011 07:32:06.3194  M50 DATA   | Cut message - target(0100),
preAck(06), packetNum(07), source(0400), crc(-), cack(-), data(31.32), num-
CutBytes(45)
```

Message type:
`M50 DATA`

Payload:
`[<msgCnt>, <msgCrc>] Cut message - target(<target>), preAck(<preAck>), pack-etNum(<packetNum>), source(<source>), crc(<crc>), cack(<cack>), data(<data>), numCutBytes(<ncb>)`

MOST50 messages are only available for blue PiraT traces. blue PiraT 2 doesn't support MOST50.

Field	Possible values	Description

`[<msgCnt>, <msgCrc>]`
 	
`<msgCnt>`
`<msgCrc>`
	
	Numeric value (hex)
	Numeric value (hex)
		Telemotive internal message counter and CRC, the entire field is optional (inclusive brackets).

`<target>`	*Numeric value (hex)*	Target address
`<preAck>`	*Numeric value (hex)*	Preemptive Acknowledge
`<packetNum>`	*Numeric value (hex)*	Packet counter
`<source>`	*Numeric value (hex)*	Source address
`<crc>`	*Numeric value (hex)* *or "-"*	CRC "-" if not available
`<cack>`	*Numeric value (hex)* *or "-"*	CRC Acknowledge "-" if not available
`<data>`	*Numeric values (hex) separated by '.'* *or "-"*	Data bytes "-" if not available
`<ncb>`	*Numeric value (dec)*	Number of bytes removed from the end of the message

3.21. MOST50 Trace Synchronous Message (only blue PiraT)

Example:

```
15.12.2011 09:37:03.8551  M50 SYNC    | [6b8a, c765] channel(98.4-66.2), da-
ta(04.05.02.03.08.09.06.07. /.../ .0c.0d.0a.0b.10.e8)
```

Message type:
`M50 SYNC`

Payload:
`[<msgCnt>, <msgCrc>] channel(<cl>.<cw>-...-<cl>.<cw>), data(<data>)`

The sequence of `<cl>.<cw>` can occure up to 20 times, depending on the number of channels being allocated on the MOST bus.

MOST50 messages are only available for blue PiraT traces. blue PiraT 2 doesn't support MOST50.

Field	Possible values	Description
[<msgCnt>, <msgCrc>] <msgCnt> <msgCrc>	 *Numeric value (hex)* *Numeric value (hex)*	Telemotive internal message counter and CRC, the entire field is optional
<cl>	*Numeric value (dec)*	Channel label
<cw>	*Numeric value (dec)*	Channel width
<data>	*Numeric values (hex) separated by '.' or "-"*	Data bytes "-" if not available

3.22. MOST150 Trace Control Message

Example:

```
04.05.2011 07:32:06.3194  M150 CTRL   | prio(01), target(2345), preAck(06),
packetNum(07), source(89ab), crc(cdef), cack(10), da-
ta(31.32.33.34.35.36.37.38)
```

Message type:

```
M150 CTRL
```

Payload:

```
<preterm> prio(<prio>), target(<target>), preAck(<preAck>), pack-
etNum(<packetNum>), source(<source>), crc(<crc>), cack(<cack>), data(<data>)
```

Field	Possible values	Description
<preterm>	*PRE-TERMINATED,*	Keyword, only for pre-terminated messages.
<prio>	*Numeric value (hex)*	Priority
<target>	*Numeric value (hex)*	Target address
<preAck>	*Numeric value (hex)*	Preemptive Acknowledge
<packetNum>	*Numeric value (hex)*	Packet counter
<source>	*Numeric value (hex)*	Source address
<crc>	*Numeric value (hex)*	CRC, optional – not for pre-terminated messages
<cack>	*Numeric value (hex)*	CRC Acknowledge, optional – not for pre-terminated messages
<data>	*Numeric values (hex) separated by '.' or "-"*	Data bytes "-" if not available

3.23. MOST150 Trace Status Message

Example:

```
04.05.2011 07:32:06.3194  M150 STATUS | MPR(1234), MDC(5678), status1(09),
status2(0a), nodePos(0000), ts(0b0c), Light on, Lock
```

Message type:

```
M150 STATUS
```

Payload:

```
<preterm> MPR(<mpr>), MDC(<mdc>), status1(<status1>), status2(<status2>), no-
dePos(<nodePos>), ts(<ts>), <light>, <lock>
```

Field	Possible values	Description

<preterm>	*PRE-TERMINATED,*	Keyword, only for pre-terminated messages.
<mpr>	*Numeric value (hex)*	Maximum Position Register
<mdc>	*Numeric value (hex)*	MOST data channel, asynchronous bandwidth
<status1>	*Numeric value (hex)*	Network state, low byte: Bit 0: System Lock Flag Bit 1: Shutdown Flag Bit 2: Ring Lock Flag Bit 3: Open Ring / Multimaster Flag Bit 7-4: Free
<status2>	*Numeric value (hex)*	Network state, high byte: Bit 7-0: Reserved
<nodePos>	*Numeric value (hex)*	Node position
<ts>	*Numeric value (hex)*	SpyNIC time stamp
<light>	Light on Light off	
<lock>	Lock Unlock	

3.24. MOST150 MDP

The payload length of MOST150 data messages can be limited by the logger configuration. Such shorted messages are marked in their payload with the prefix “Cut message - ” and are described separately here.

3.24.1. MOST150 Trace Data Message

Examples:

```
04.05.2011 07:32:06.3194  M150 DATA   | target(0100), preAck(06), pack-
etNum(07), source(0400), crc(cdef), cack(10), data(31.32.33.34.35.36.37.38)

04.05.2011 07:32:06.3194  M150 DATA   | [6b8a, c765] target(0100),
preAck(06), packetNum(07), source(0400), crc(cdef), cack(10), da-
ta(31.32.33.34.35.36.37.38)

04.05.2011 07:32:06.3194  M150 DATA   | PRE-TERMINATED, target(0100),
preAck(06), packetNum(07), source(0400), crc(cdef), cack(10), da-
ta(31.32.33.34.35.36.37.38)
```

Message type:

```
M150 DATA
```

Payload:

```
<preterm> [<msgCnt>, <msgCrc>] target(<target>), preAck(<preAck>), pack-
etNum(<packetNum>), source(<source>), crc(<crc>), cack(<cack>), data(<data>)
```

Field	Possible values	Description
<preterm>	PRE-TERMINATED,	Keyword, only for pre-terminated messages.
[<msgCnt>, <msgCrc>] <msgCnt> <msgCrc>	 *Numeric value (hex)* *Numeric value (hex)*	Telemotive internal message counter and CRC, the entire field is optional.
<target>	*Numeric value (hex)*	Target address
<preAck>	*Numeric value (hex)*	Preemptive Acknowledge
<packetNum>	*Numeric value (hex)*	Packet counter

<source>	*Numeric value (hex)*	Source address
<crc>	*Numeric value (hex)*	CRC, this field incl. label is not available for pre-terminated messages
<cack>	*Numeric value (hex)*	CRC Acknowledge, this field incl. label is not available for pre-terminated messages
<data>	*Numeric values (hex) separated by '.'*	Data bytes

3.24.2. Shorted MOST150 Trace Data Message

Example:

```
04.05.2011 07:32:06.3194  M150 DATA   | Cut message - target(0100),
preAck(06), packetNum(07), source(0400), crc(-), data(31.32), numCutBytes(45)
```

Message type:

```
M150 DATA
```

Payload:

```
<preterm> Cut message - target(<target>), preAck(<preAck>), pack-
etNum(<packetNum>), source(<source>), crc(<crc>), data(<data>), num-
CutBytes(<ncb>)
```

Field	Possible values	Description
<preterm>	PRE-TERMINATED,	Keyword, only for pre-terminated messages.
<target>	*Numeric value (hex)*	Target address
<preAck>	*Numeric value (hex)*	Preemptive Acknowledge
<packetNum>	*Numeric value (hex)*	Packet counter
<source>	*Numeric value (hex)*	Source address
<crc>	*Numeric value (hex) or "-"*	CRC, "-" if crc not available
<data>	*Numeric values (hex) Separated by '. or "-"'*	Data bytes "-" if not available
<ncb>	*Numeric value (dec)*	Number of bytes removed from the end of the message

3.25. MOST150 MEP

The payload length of MOST150 ethernet packet messages can be limited by the logger configuration. Such shorted messages are marked in their payload with the prefix "Cut message - " and are described separately here.

3.25.1. MOST150 Trace Ethernet Message

Examples:

```
04.05.2011 07:32:06.3194  M150 ETH    | target(5a5a5a5a5a5a), preAck(06),
crc(cdefcdef), cack(10), data(31.32.33.34.35.36.37.38)
```

Message type:

```
M150 ETH
```

Payload:

```
<preterm> target(<target>), preAck(<preAck>), crc(<crc>), cack(<cack>), da-
ta(<data>)
```

Field	Possible values	Description

<preterm>	PRE-TERMINATED,	Keyword, only for pre-terminated messages.
<target>	*Numeric value (hex)*	Target address
<preAck>	*Numeric value (hex)*	Preemptive Acknowledge
<crc>	*Numeric value (hex)*	CRC, this field incl. label is not available for pre-terminated messages
<cack>	*Numeric value (hex)*	CRC Acknowledge, this field incl. label is not available for pre-terminated messages
<data>	*Numeric values (hex) separated by '.'*	Data bytes

3.25.2. Shorted MOST150 Trace Ethernet Message

Examples:

```
04.05.2011 07:32:06.3194  M150 ETH    | Cut message - target(5a5a5a5a5a5a),
preAck(06), crc(cdefcdef), data(31.32.33.34.35.36.37.38), numCutBytes (0)
```

Message type:

```
M150 ETH
```

Payload:

```
Cut message - target(<target>), preAck(<preAck>), crc(<crc>), data(<data>),
numCutBytes(<ncb>)
```

Field	Possible values	Description
<target>	*Numeric value (hex)*	Target address
<preAck>	*Numeric value (hex)*	Preemptive Acknowledge "-" if not available
<crc>	*Numeric value (hex)*	CRC, "-" if crc not available
<data>	*Numeric values (hex) separated by '.'* *or "-"*	Data bytes "-" if not available
<ncb>	*Numeric value (dec)*	Number of bytes removed from the end of the message

3.26. Serial Data Trace Message

Examples:

```
04.05.2011 07:32:06.3194  SERIAL #1   | [None] ABCDEFGH
04.05.2011 07:32:06.3194  SERIAL #2   | [Mask Client] IJKLMNOP
04.05.2011 07:32:06.3194  SERIAL #3   | [Generic Logger] QRSTUVWX
04.05.2011 07:32:06.3194  SERIAL #4   | [DLT BMW] [ECU2] 12345678
```

Message type:

```
SERIAL
```

Payload:

```
[<status>] [<protocol>] <data>

[<status>] is optional and only written for error states
```

Field	Possible values	Description
<status>	OVERRUN PARITYERROR FRAMINGERROR BREAK	Error state, optional

<protocol>	None Mask Client Generic Logger DLT BMW	Protocol
<data>	*arbitrary*	Serial data, non-printable values are written escaped. Data with DLT protocol is preceded by [<ecuID>], where <ecuID> is a four character string.

3.27. Shutdown Message (only blue PiraT)

Examples:

```
15.12.2011 09:36:58.3082  SHUTDOWN    | SD_NOW
```

Message type:

```
SHUTDOWN
```

Payload:

```
<type>
```

Field	Possible values	Description
<type>	SD_NOW SD_REQUEST SD_CANCEL	

3.28. Startup Message (only blue PiraT)

Examples:

```
15.12.2011 09:36:58.3082  SYSTEM STARTUP
```

Message type:

```
SYSTEM STARTUP
```

Payload:

No payload

3.29. System Configuration Message

Examples:

```
15.12.2011 09:35:56.9727  SYS CONFIG  |   name=bluePiraT
```

Message type:

```
SYS CONFIG
```

Payload:

```
<cfg>
```

Field	Possible values	Description
<cfg>	*string*	Mostly key/value pair

3.30. System Message

Example:

```
09.08.2012 10:57:03.7591  SYSTEM MSG  | [WARNING] ABC
09.08.2012 10:57:03.7591  SYSTEM MSG  | [SUCCESS] DEF
```

Message type:

```
SYSTEM MSG
```

Payload:

```
[<type>] <data>
```

Field	Possible values	Description
<type>	INFO VERSION USAGE DEBUGMSG ACTION COUNTER SUCCESS EVENT TERMINAL ETHERNET ERROR_LOG ERROR_SEND ECU FILE SEPARATOR LOGGER_STATUS LOGGER_NETWORK WARNING ERROR	general purpose info message data contains version information data contains HD fill level (0-100%) general purpose debugging message data contains a command like "ERROR LED ON" data contains a counter (used for Trigger List on RC) something went well (i.e. download to FlexRay MPC) an info message to be logged into Events file display Terminal activities on RC info message from ethernet device error log protocol format: [\<num>] \<text> number of the failed msg general purpose message for ECUs data contains a filename separates header from trace data in trace file status of logger, format: \<logger_status_t> \<text> network info, format: \<num> \<text> warning, not really serious something bad happend
<data>	*arbitrary*	Depends on <type>

3.31. Temperature Message

Example:

```
09.08.2012 10:57:03.7591  TEMPERATURE | -12 °C
```

Message type:

```
TEMPERATURE
```

Payload:

```
<temperature> °C
```

Field	Possible values	Description
<temperature>	*Numeric value (dec)*	Temperature in degree Celsius

3.32. Testtools SDK Configuration Message

Example:

```
30.08.2011 13:06:15.7455  TTSDK CONF  | type=0, interface=Serial #0,
name=Serial1, containerId=-1, protocol=SP_SERIAL_NO
```

Message type:

```
TTSDK CONF
```

Payload:

```
type=<type>, interface=<channel>, name=<name> containerId=<id>, proto-
col=<proto>, debug level=<level>, device ip=<ip>, devicePort=<port>, host
ip=<host>
```

Field	Possible values	Description
<type>	*Numeric value (dec)*	Currently always 0
<channel>	*One of:* CAN Serial MOST25 FlexRay LIN	

	Ethernet Camera MOST150 MOST50 Analog CCP_XCP GPIO *Followed by:* #<index> with <index>= numeric value (dec)	
<name>	*arbitrary*	The configured channel name
<id>	*Numeric value (dec)*	
<proto>	*For serial interfaces:* *SP_SERIAL_NO* *TRACE CLIENT* *GNLOGGER* *DLT_BMW* *For Ethernet interaces:* GNLOGGER RAW UTF8 DLT UDPServer	Optional, only for serial and Ethernet interfaces. For all other interfaces this field (label + value) is not written.
<level>	*Numeric value (dec)*	Optional, Only for Ethernet interfaces with GNLOGGER protocol. For other interfaces this field (label + value) is not written.
<ip>	*string*	The configured device IP address. Optional, Only for Ethernet interfaces. For other interfaces this field (label + value) is not written.
<port>	*Numeric value (dec)*	The configured device port. Optional, Only for Ethernet interfaces. For other interfaces this field (label + value) is not written.
<host>	*string*	The configured host IP address. Optional, Only for Ethernet interfaces. For other interfaces this field (label + value) is not written.

3.33. Time Jump Message (only blue PiraT)

Examples:

```
15.12.2011 09:36:58.3082  TIME JUMP
```

Message type:

```
TIME JUMP
```

Payload:

No payload

3.34. Trace File Meta Info Messages (only blue PiraT 2)

3.34.1. Trace File Meta Info Message – Time Span

Example:

```
12.09.1918 00:32:54.7052  META INFO   | [TIME SPAN] start time = 25.05.2012
15:29:49.271974, end time = 25.05.2012 15:31:21.705295
```

Message type:
```
META INFO
```

Payload:
```
[TIME SPAN] start time = <start>, end time = <end>
```

Field	Possible values	Description
<start>	*string*	First time stamp of the TMT file's data
<end>	*string*	Last time stamp of the TMT file's data

3.34.2. Trace File Meta Info Message – Time Zone

Example:
```
12.09.1918 00:32:54.7052  META INFO   | [TIME ZONE] WEuropeStandardTime-1DST-
2,M3.5.0/2:0:0,M10.5.0/3:0:0
```

Message type:
```
META INFO
```

Payload:
```
[TIME ZONE] <tz>
```

Field	Possible values	Description
<tz>	*string*	Posix time zone string, see http://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html

3.34.3. Trace File Meta Info Message – Mainboard

Example:
```
12.09.1918 00:32:54.7052  META INFO   | [MAINBOARD] 1013014
```

Message type:
```
META INFO
```

Payload:
```
[MAINBOARD] <mb>
```

Field	Possible values	Description
<mb>	*Numeric value (dec)*	Mainboard number of the origin data logger

3.34.4. Trace File Meta Info Message – Configuration Backup

Example:
```
12.09.1918 00:32:54.7052  META INFO   | [CFG_BCKP] bpng_[1014327]_[2012-08-
23_13-01-45].zip
```

Message type:
```
META INFO
```

Payload:
```
[CFG_BCKP] <fn>
```

Field	Possible values	Description
<fn>	*string*	The file name of the logger's configuration backup for the recording time of the TMT file.

3.34.5. Trace File Meta Info Message – Cascading

Example:

```
12.09.1918 00:32:54.7052  META INFO   | [CASCADING] type = NONE, offset = 0
```

Message type:

```
META INFO
```

Payload:

```
[CASCADING] type = <type>, offset = <off>
```

Field	Possible values	Description
<type>	NONE MASTER SLAVE MERGED	The cascading mode of the logger that recorded the corresponding TMT file “MERGED” is reserved for future use
<off>	*Numeric value (dec)*	The time offset of a slave device to the master device in µs

3.34.6. Trace File Meta Info Message – Data Mix Mode

Example:

```
12.09.1918 00:32:54.7052  META INFO   | [DATA_MIX_MODE] data mix mode:
ALL_DATA
```

Message type:

```
META INFO
```

Payload:

```
[DATA_MIX_MODE] data mix mode: <mode>
```

Field	Possible values	Description
<mode>	ALL_DATA FILTERED_DATA UNDEF (<x>)	ALL_DATA: The corresponding TMT/XTMT file contains the data of all logger interfaces of the file’s enclosed time span. FILTERED_DATA: The corresponding TMT/XTMT file contains a subset of the data of all logger interfaces of the file’s enclosed time span.

3.34.7. Trace File Meta Info Message – Channel

Example:

```
12.09.1918 00:32:54.7052  META INFO   | [CHANNEL] CAN #1 (CAN1) : DATA
```

Message type:

```
META INFO
```

Payload:

```
[CHANNEL] <ch> <chName> : <available>
```

Field	Possible values	Description
<ch>	*string*	A combination of bus and channel number (e.g. CAN #1) or bus and sub channel (e.g. MOST150 CTRL)
<chName>	*string*	The configured channel name
<available>	DATA NO DATA	Marks whether the TMT file contains data of this channel or not.

3.34.8. Trace File Meta Info Message – Event

Example:

```
12.09.1918 00:32:54.7052  META INFO   | [EVENT] STARTUP 31.08.2010
10:27:00.000000 StartUp set by RdbHandler (first startup)
```

Message type:
`META INFO`

Payload:
`[EVENT] <type> <idx> <time> <comment>`

Field	Possible values	Description
`<type>`	STARTUP SHUTDOWN MARKER INFO SLAVE_OFFSET SLAVE_TO_MASTER DATA_DELETED SUDDEN_DEATH TIME_SET NEW_TIME	The event type
`<idx>`	*#<x>* *With <x>=Numeric value (dec)*	The event index, only for MARKER events
`<time>`	*string*	The event time stamp
`<comment>`	*string*	An optional comment

3.35. Trigger Clear Message (only blue PiraT)

Examples:

```
15.12.2011 09:36:58.3082  TRIGGER CLEAR
```

Message type:
`TRIGGER CLEAR`

Payload:
No payload