MVB System User's Guide

Comments & Principles of MVB Engineering

This User's Guide is written for the application engineer dealing with MVB in railway vehicle applications.

Duagon Data Sheet Preamble

On having purchased products described in this data sheet, the customer acquires the right to use the products according to its specified purpose and in accordance with all operation, service and maintenance instructions. All other rights to the product, Duagon's intangible assets rights in particular, belong solely to Duagon and may not be deemed to have been assigned along with the sale of the products.

All product properties are fully described in the data sheet under express exclusion of any warranty for other properties. Of decisive relevance is the data sheet valid at the time of the order being placed. Duagon provides a warranty that the product properties are retained during the period of warranty. Evidence that the properties of the product have been retained will be brought, always and exclusively, on Duagon premises by means of a test construction pursuant to the type test.

The customer is obliged to inspect whether the products themselves are suitable for the application intended. In particular, that inspection must include the integration of the products into the intended system configuration and a check on whether the properties as per data sheet can be fulfilled once integrated into the system configuration as planned by the customer. Since the products are not certificated for operation with security applications, the customer must take appropriate measures to ensure that any malfunctions that may occur in a system configuration with other products will be absorbed by supplementary security measures.

The period of warranty for the products is 24 months and it begins on the date the products are shipped from the factory.

The warranty that Duagon assumes for the products will, at Duagon's discretion, be limited either to the repair of or the replacement of the products at the Duagon factory. The warranty solely covers the products or parts thereof which, despite professional handling, have become defective or unusable and which arrive at the Duagon factories for repair or replacement during the period of warranty. The extent of Duagon's warranty is fully set out in this data sheet. Duagon cannot be held liable for consequential damage caused by a defect or for indirect damage or for consequential damage of any kind. Therefore the customer bears all and any costs that occur due to production downtime, for example, or due to the installation or dismantling of products or due to their transportation to Duagon and back.

Duagon's liability and warranty do not obtain if evidence cannot be brought that the products were being operated according to its specified purpose and in accordance with all operation, service and maintenance instructions as issued by Duagon.

These provisions form an integrated part of the product properties. Duagon products cannot be acquired with other or more extensive degrees of warranty and liability on the part of Duagon.

This data sheet is to be evaluated in accordance with **Swiss** law. The court of jurisdiction is the **seat of the vendor**. The applicability of the UN agreement as to international sales of goods (also known as "Viennese Purchasing Convention") is herewith expressly excluded.

duagon GmbH, Badenerstrasse 21, CH-8953 Dietikon, Switzerland

Phone: +41 1 7434012, Fax +41 1 7434015, www.duagon.com

Document history

Rev.	Date	Author	Comments	Ident-Number
1	August,14th 2004	Schlage	First release	d000656-002181
2	May, 27th 2005	Schlage		d000656-003145
3				
4				
5				
6				
7				
8				

Table of contents

Introduction

This User's Guide is written for the application engineer dealing with MVB in railway vehicle applications.

Three main areas are covered:

- Understanding
  The basics of the MVB world are presented in a condensed way.
- Implementation
  Build your own network: start from network topologies, optimise your own decisions about e.g. engineering effort, bandwidth performance, redundancy, cost, time to complete.
- Practical work
  Take measurements to verify the network performance; analyse and interpret failures in a correct way. Special attention was put on interoperability issues.

This document reflects the personal experience of the author, based on daily business within the company “duagon GmbH”. This is not an advertisement or a commitment for technical data, though – it is much more a collection of issues beyond the scope of usual product documentation.

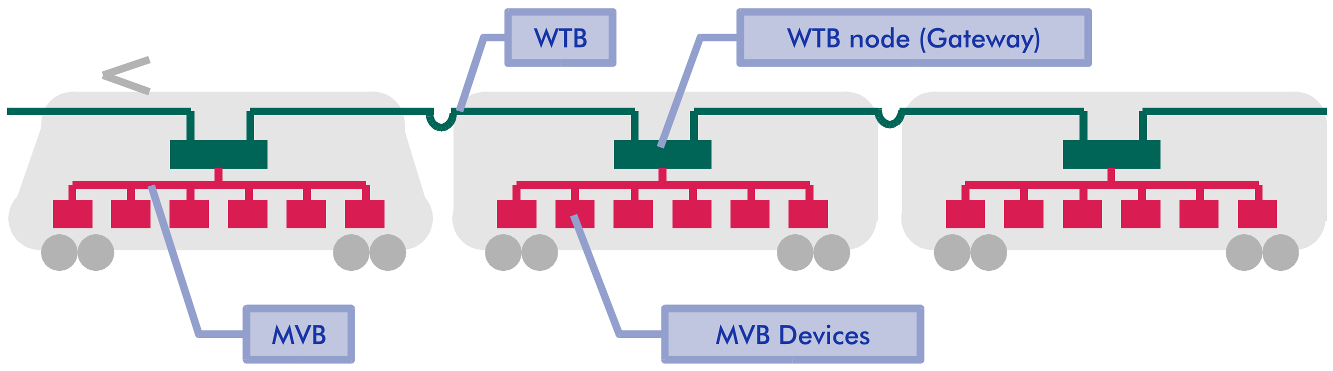
This user's guide shall be a general help for the application engineer, and hopefully this “knowledge snapshot” will be enhanced in future by feedback arriving from the field: It is understood to be a living document; i.e. comments and contributions are invited: contact the author thomas.schlage@duagon.com.

General Network Architecture

TCN – Train communication network

The "TCN" Train Communication Network is a network with two hierarchical layers:

- The WTB connects the vehicles within a train
- The MVB connects the devices within one vehicle



WTB - Wire Train Bus

The technical key specifications of the WTB are:

- Main application: open trains with variable composition such as UIC trains
- Covered distance: 860m
- Number of nodes: 32 max.
- Data rate: 1 Mbits/second over shielded, twisted wires
- Response time: 25ms
- Inauguration: assigns to each node its sequential address and orientation

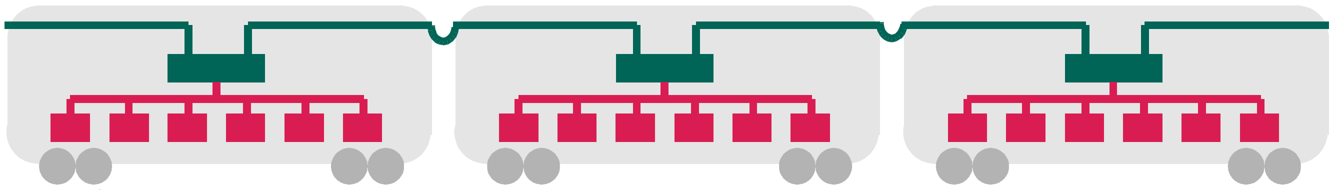
The **key issue** is the **interoperability** between different vehicles:

- vehicles of different type: with and without traction, specific types like sleeping or dining coach
- vehicles from different manufacturers
- vehicles passing the border from one railway operator to the next one

If all these facettes of interoperability shall be fulfilled, a standardisation beyond the communication network takes place:

- The TCN network with WTB and MVB defines how information is exchanged
- Other standards define, what information content is exchanged.

For example, various UIC leaflets define the meaning of exchanged information.



MVB Multifunction Vehicle Bus

The technical key specifications of the MVB are:

- Data rate 1.5 Mbit/s , data frame size 16-256 bits
- Cyclic transmission of process data (broadcast)
- Event driven message data transfer (point-to-point)
- Full support of redundancy
- Three physical layers:

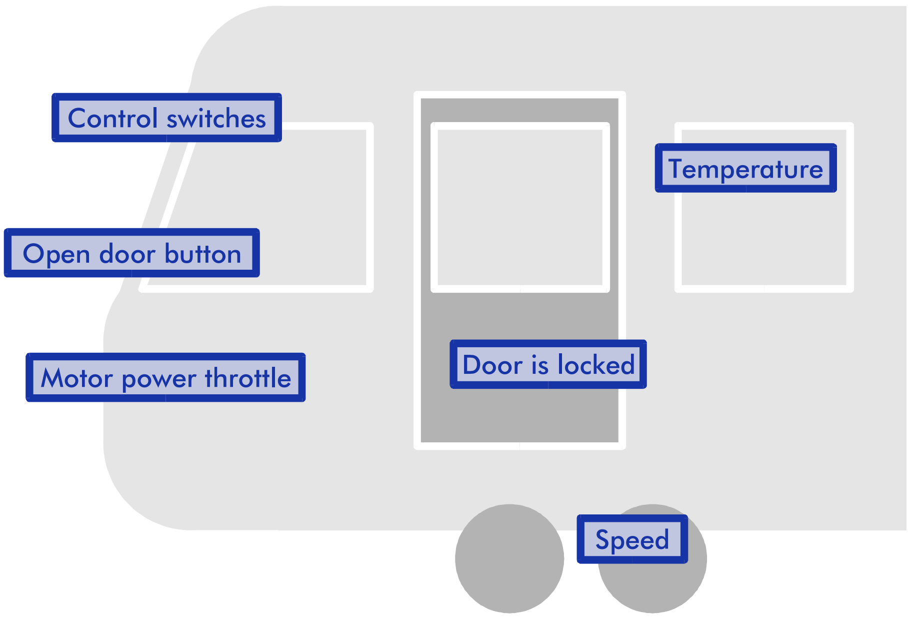
• OGF optical fibres	2000m
• EMD twisted pair	200m, 32 nodes
• ESD twisted pair	20m, 32 nodes
with galvanic insulation	200m

The **key issue** is the **cost versus performance optimisation** within a vehicle:

- Integrate communication to various subsupplier´s devices into one vehicle; e.g. traction equipment, door controllers, displays, driver´s cab control equipment.
- The focus of optimisation follows the type of vehicle. The network may be more or less complex; the requirements about availability are different.
- The cost optimisation of the network nodes takes place on complete units. The network equipment is a small part within the subsuppliers scope of delivery; the competition works on complete units like power converters, doors, passenger information systems, vehicle control systems.
- For the vehicle supplier, the cost of integration, i.e. application engineering, is dominant. The MVB matches railway specific requirements, and the standardisation ensures reusability of engineering work.
- The support cost during years of operation drives the use of standardised communication equipment: one tool set for the whole vehicle; support of later retrofit options from an open supplier market.

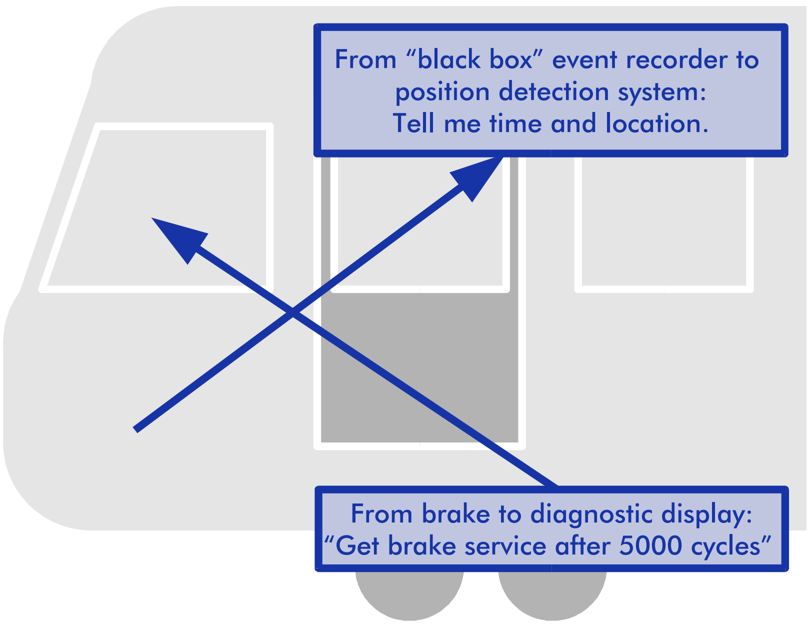
MVB Bus Traffic

State Variables



- The Variables reflect the state of the plant.
- The information content is “small”: From a single bit up to a several bytes
- Their typical response time is in the range of 16 to 256 ms.
- Examples are driver commands, speed or temperature information.
- The variables are periodically transmitted.
- Spurious data losses will be compensated at the next periodic cycle.
- The use of the variables may be at multiply points wide spread over the vehicle

Messages


From "black box" event recorder to position detection system: Tell me time and location.
From brake to diagnostic display: "Get brake service after 5000 cycles"

- Messages contain the events of the plant.
- On-demand transmission from one point to another point.
- The size of a single message may be "big": practical size of up to several 100kBytes.
- Response is at human speed: >0.5s.
- There is a flow control & error recovery protocol for catching all events.
- Examples during normal operation are on-line diagnostics, passenger information, event recorder.
- During commissioning or field service, messages are used for software download and collection of debugging data.

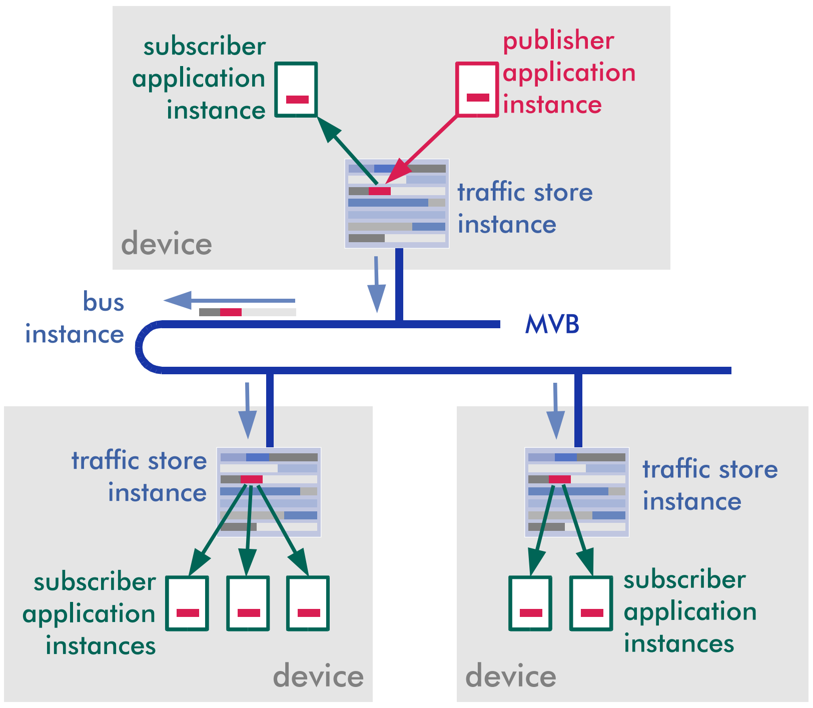
Process Variables

Distribution of Process Variables over the bus

One “publisher application” generates the process variable and places it in the traffic store of the generating device.

From the traffic store of the generating device, the process variable is distributed over the MVB. After transmission over MVB, the process variable is available in all traffic stores.

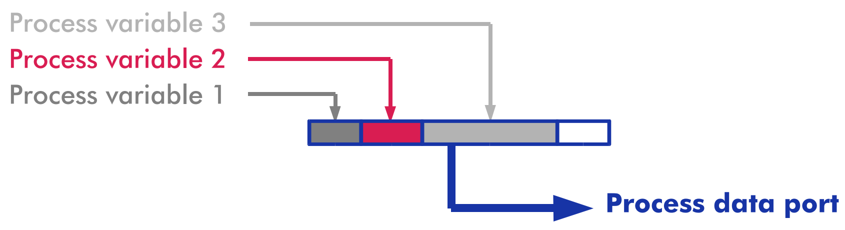
A “subscriber application” reads the process variable from its local traffic store.


subscriber application instance
publisher application instance
traffic store instance
device
bus instance
MVB
traffic store instance
subscriber application instances
device
traffic store instance
subscriber application instances
device

- There exist a number of instances of the same process variable.
- These instances are inconsistent to each other. The distribution from the publisher application to the subscriber application takes time, even if both are located on the same device.

Process Data Ports

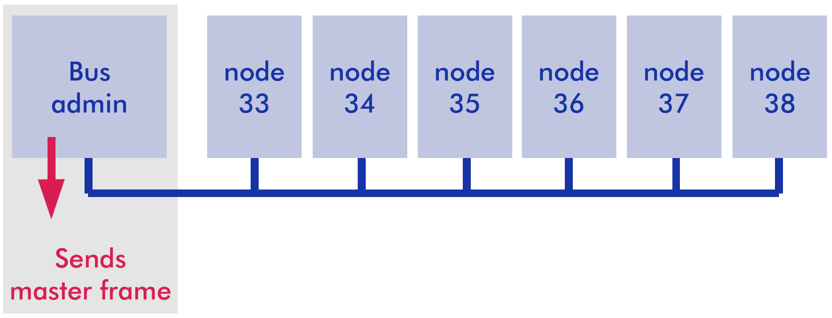
Several process variables, which are intended to be distributed from the same device, are collected together to form a “dataset”. A dataset is transmitted within one “process data port”.



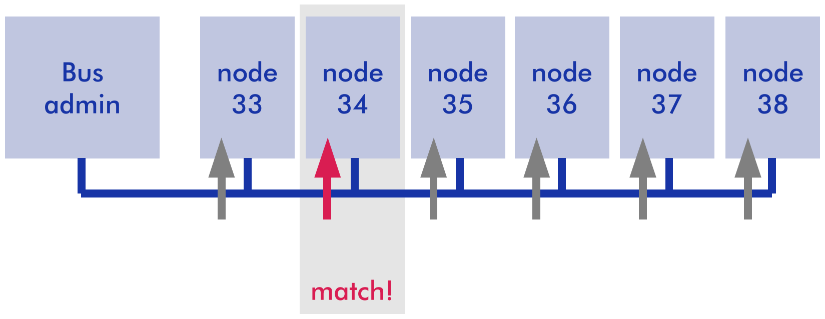
- The process data port as a whole is actually sent over the MVB, not the single variables.
- The definition, how the variables are collected together, is done by the application engineer. This definition will follow practical considerations like source device, repetition frequency over the bus, size of variables, clustering of variables with a functional relationship.
- Obviously, the definition of the variables must be the same for all devices on the bus:
    - Preferably the variable definition should be done with the focus on the overall MVB traffic, together with the MVB bus administrator configuration. A tool then generates different configuration files for all involved devices.
    - Sometimes, the import of such a variable definition is not possible; e.g. due to not matching engineering tools. Then, the definition must be maintained separately: Be sure to keep track of changes!

Process Data: Bus Broadcasting

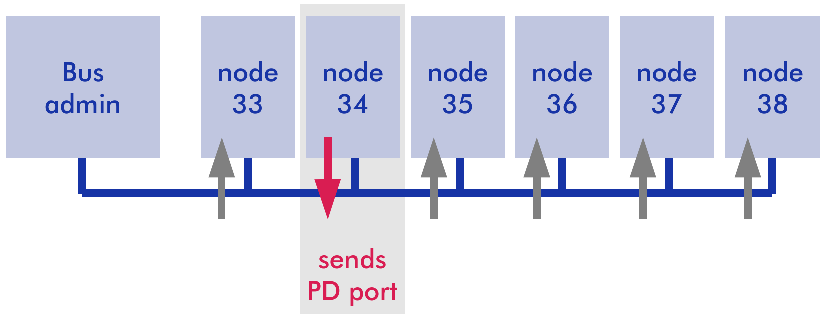
Step 1: The bus administrator sends out a request for a PD port, which is located on node 34. On the bus, this is represented by a "master frame".



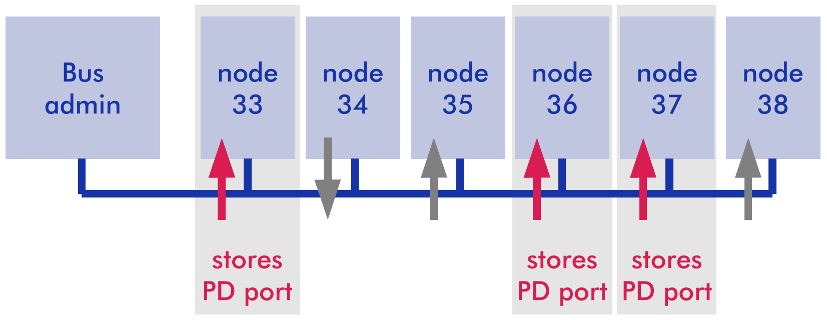
Step 2: Reading the master frame from the bus most devices realise, that they are not involved so far. However, node 34 is alarmed to send out the requested PD port.



Step 3: Node 34 sends out the PD port, and everybody else listens to it (= broad-casting). On the bus, this is represented by a "slave frame".



Step 4: Some of the listening nodes are interested in the PD port sent over the bus. They will store this PD port in their local traffic store; for later use by their ap-plication.



This way of PD port distribution has some consequences:

- The bus administrator initiates the PD transfer, but it does not need to understand any data. Data is received on PD- nodes, only. However, the administrator function can be integrated within a normal node; then all activities are executed transparent to each other.
- The bus administrator addresses "logical ports", not the device addresses of the involved nodes. By this way, each node may have more than one source port.
- For one "logical port address" there is exactly one node having this port defined as a "source".
- The time, when PD data is received or transmitted, is not influenced by the slave nodes; the have simply to react. By this way, there is no need to configure the nodes with any timing related parameters. All timing configuration is to be done on the administrator.
- The slave nodes have to be configured with the following information:
    - The "source" ports (the ports to be transmitted).
    - The size of the ports: Even though this information could be derived from the master frame, it is used for an additional plausibility check during reception or transmission of ports.
    - For simple applications, an individual node may need only a few PD ports. Thus it may make sense to identify a reduced set of PD "sink" ports (ports to be received). By this way, a cost effective small traffic store may be applied.
      More complex nodes will typically implement the whole PD port address space: The theoritical maximum for PD is: 4096 ports * 32 bytes = 128 kBytes. Such nodes will not need "sink port configuration", since they will store all possible logical ports, anyway.
- The bus administrator has to be configured with the following information:
    - All used logical port addresses.
    - The size of all used ports.
    - The repetition rate of all used ports.

    This information is sufficient for a fully functional bus administrator. However, some administrator implementations require some more data (which is separated from the PD traffic):

    - A list of device addresses (existing nodes) together with their MD and BA capabilities. Note: This information can also be dynamically derived from watching the bus traffic.

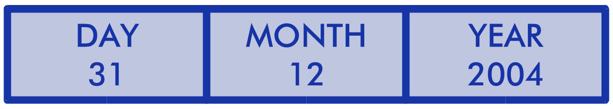
Data Consistency

In some cases, different variables, being defined at a common point in time, must be kept together.

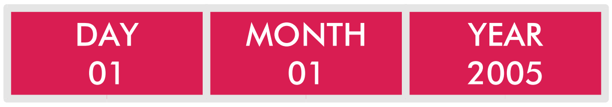
The feature “consistency” is mandatory for variables within one PD port (“dataset”). Note: Any additional “clustering” of variables does not extend the consistency beyond the port/ dataset.

Example:

The source distributes the variables DAY, MONTH, YEAR at the same time, within one source port, representing a “date”:

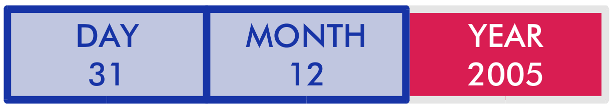


A certain time later, the source distributes the variable DAY, MONTH, YEAR again; but now with changed contents:



Since the distribution of variables over the MVB takes some time, any subscriber of these variables will get the new data somehow later.
However, there is a strong requirement of “data consistency”: Any subscriber of these variables will receive either the first or the second set of variables, but never a mix of both: This means “consistent data contents”.

Imagine, a **wrong** working subsriber node would receive the first variables DAY and MONTH from the first transmitted set; then it is interrupted by the MVB transmission of the second variable set and gets YEAR from the new set:



Obviously, the result is completely wrong. It does not reflect neither the first nor the second set of variables, but something different.

Invalid Variables, received ...

There may be several reasons, why variables could be invalid. Of course, any device receiving variables should be aware of this and act in a reasonable way. This could be for example to switch off all outputs and enter a safe state or to use data from a redundant source:

- The source device worked fine, but now fails completely (like power off) or the communication to the source device is interrupted (e.g. broken cable):
  Every variable includes information about the "freshness". This "age of the variable" can be used to decide, whether it is still to be considered OK or not. A reasonable level is for example twice the repitition time for PD; or 2seconds at most.
  By this way, a single loss of a data frame is still not detected as an error; an error requires two or more consecutive losses.
  Please note, that this limit as well as the reasonable reaction upon data loss is defined by the application engineer. The MVB driver software just delivers the freshness information, nothing more.

Freshness

The freshness information comes with every received variable.

An UNSIGNED16 represents the time being elapsed from the last valid reception up to now: 0 is "just born"; 2000 is 2 seconds, which is fairly old; and 65535 is the outmost value, which does not "age" any more.
When you get a 65535, the port has most probably never been received.

- The source device is still working, but some specific variables are not valid any more:
  This could be one defective sensor among others in normal operation. Or a gateway collects variables from different sources and one of these is missing.
  For this purpose, the application engineer should introduce "check variables". These declare variables to be either valid or erroneous. One check variable may be used for several variables.
  The MVB driver software does not recognise invalid variables in between valid ones: the MVB communication by itself was successful.
  The application engineer has to take care about the introduction, configuration and use of check variables.

- The source device has sent correct data, but the sink receivers get single or multiple bit errors.
  This case is checked by the MVB controller according to several methods. This includes a plausibility check of the Manchester coding, start and end delimiters, cyclic redundancy checking, parity bit, check of the timing between master and slave frame. The result of this check is very reliable; and the result from finding an error is always the same: the data is not presented to the upper software layers. As a result, the user will face missing data, and may react as described before.

Check_Variable

The validity of a variable is confirmed by a Check_Variable. It is defined in the same dataset, type ANTIVALENT2:

0b00:	**the protected variables are erroneous or suspicious;**
0b01:	**the protected variables are assumed to be correct;**
0b10:	**the protected variables have been forced to an imposed value (set to a certain value on purpose e.g. for debugging; to be understood as correct)**
0b11:	**the protected variables are undefined**

...or transmitted

Transmission of invalid variables is absolutely forbidden! It is up to the user to actively avoid some critical cases:

- The application software must make use of a watchdog. In case, the software is locked in an eternal loop, the PD is still actively distributed over the MVB, taking data from the traffic store. This keeps "old" data perfectly fresh on the receiver´s side.
  After a while, the watchdog fires and the complete CPU system dives through a reset. Be sure, that the watchdog reset really arrives at the MVB interface: When reset, the interface has to definitively shut off any MVB activity.
  Afterwards, the software may come back to life again and configure the MVB interface another time.
- If an application is able to detect an invalid state of a variable, it must pass this information with the variable. This could be by the use of check variables, but also by stopping the transmission of source ports, or by stopping MVB traffic at all (software reset).

Life Sign Reception

Quite often, a different method for variable verification is used: "Life_Sign" or " heart beat". This method has been proven by experience, but please note some basic drawbacks:

- The life sign does not prove the valid transmission of process variables in the sense of MVB transmission. There is no particular rule, that "undefined" PD data in the traffic memory must be "constant". For this purpose, use in any case the freshness information.
- The life sign may prove valid execution of the source device´s internal task structure. Of course, this is an important information. However, have a look into using a watchdog on the source side: It would serve the same purpose, too and has the benefit of restarting the complete device.

Life_Sign

A single bit is toggled from 0 to 1 and vice versa; with a rate of e.g. once per second.

On sink side, the toggling bit gives confidence about the health of the source device; this applies also to the received process variables.

Message Data MD

Functions and Devices

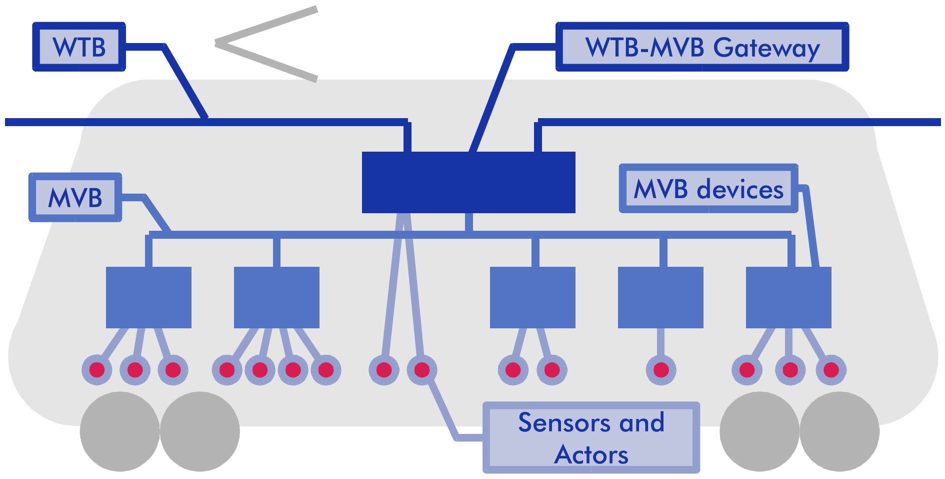
Each vehicle supports a number of standardised functions.

The Train Bus accesses vehicles without knowing their internal structure.

The train bus accesses functions rather than devices.

These functions are implemented by one or several vehicle bus devices, or even by the gateway itself.

The gateway deduces the device from the function and routes messages.



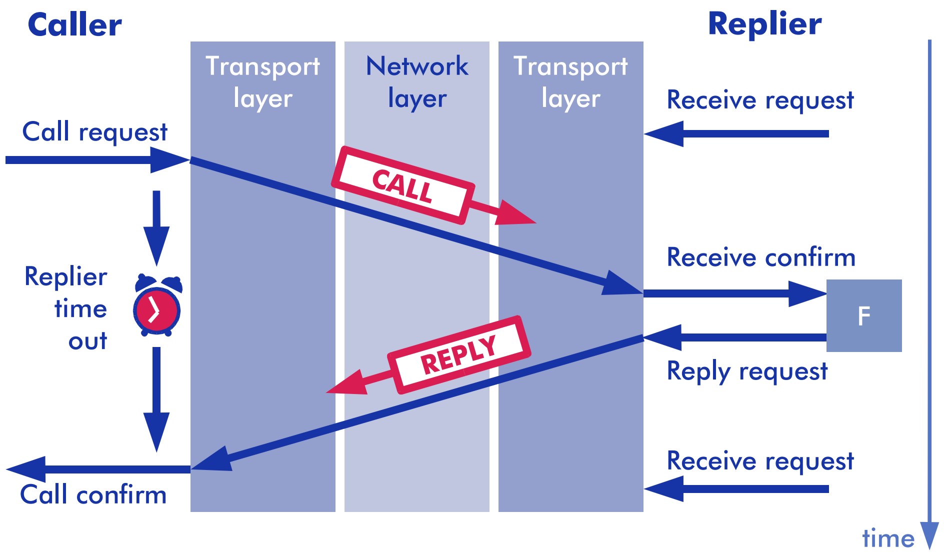
Client- Server Service

The application interface for messages provides a "call with reply" service.

Applications communicate among themselves on a Client- Server basis.

Tasks use the same communication scheme:

- within the same processor
- within the same vehicle bus
- within the Train Communication Network.



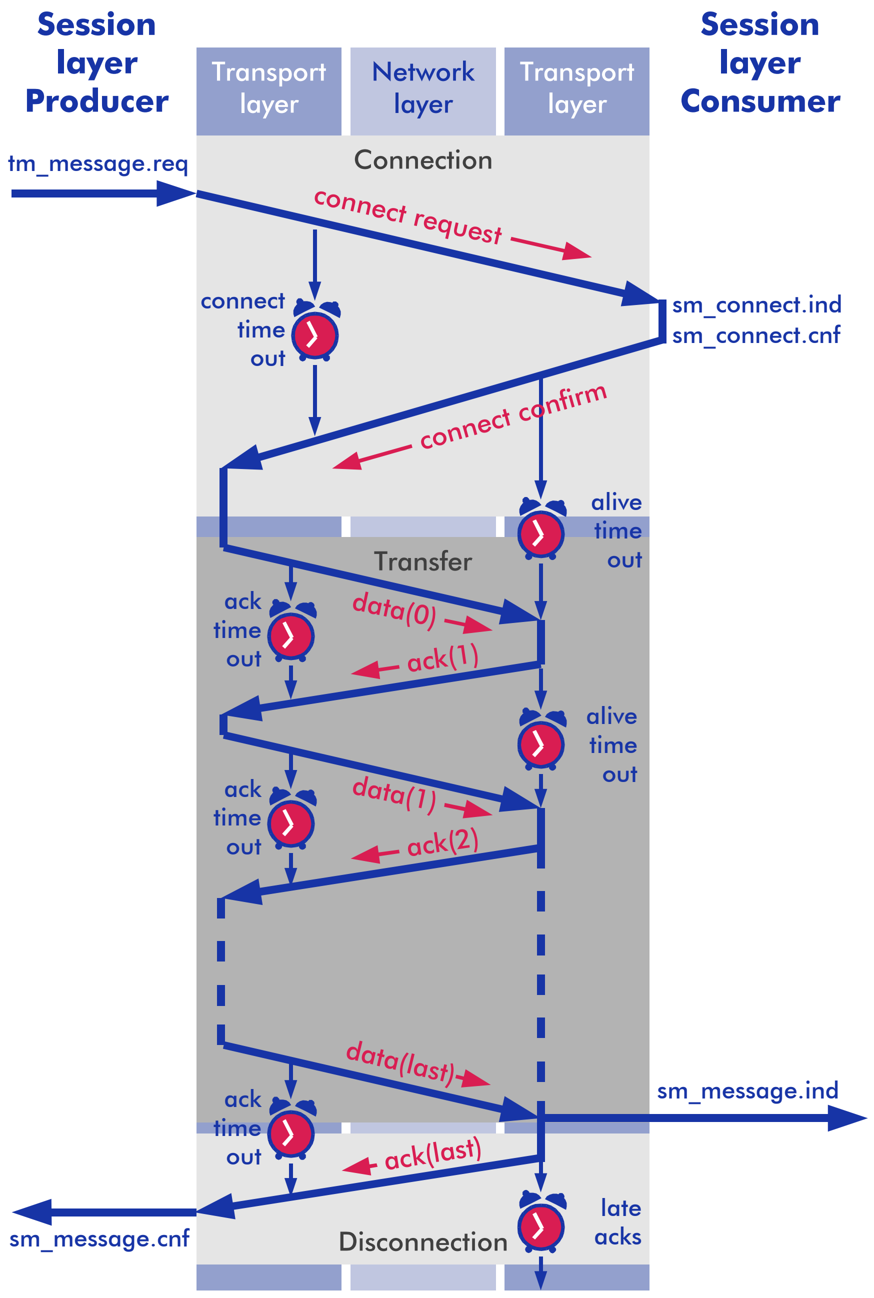
Example: The Caller issues a Call request. To a high extent, this looks like "invoking a function" as it is used in usual software source code. According to the kind of function, parameters are attached to the "call" as well as to the "reply".

In contrast to usual software, where the calling and the replying code is executed on the same processor, the function is executed on a remote node, the "server".

The server may not be available or the communication may be somehow disturbed. This requires the use of some administrative effort: Timers will mark a function as being "unavailable", which unblocks the Caller´s software execution.

Transport: Frame Exchange in one Direction (Call or Reply)

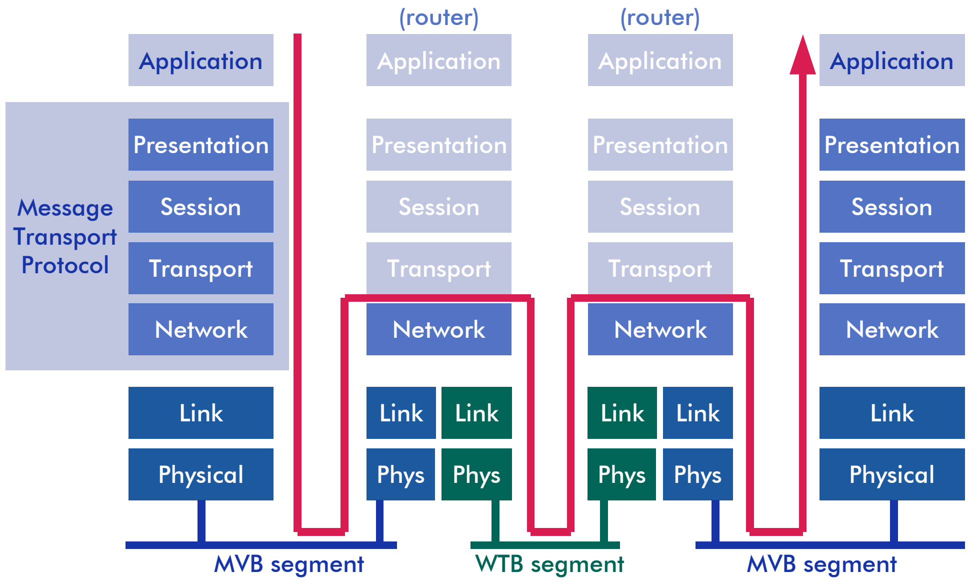
A transport exchange consists of three phases: connection, transfer and disconnection.



In this example, the transfer takes place with window size = 1. In case, there was a higher window size negotiated, the producer´s transport layer will send a higher number of data frames before waiting for an acknowledge.

Various time-outs ensure, that any spurious losses of frames will lead to a repeated transmission. In this sense, the execution flow in the drawing is simplified: the retry paths are not shown.

End- to- End Transport Protocol



The “Message Transport Protocol” is able to route between different links. The example above demonstrates the routing from one vehicle via the WTB train bus to another vehicle.

Foreign Devices & Bus Systems

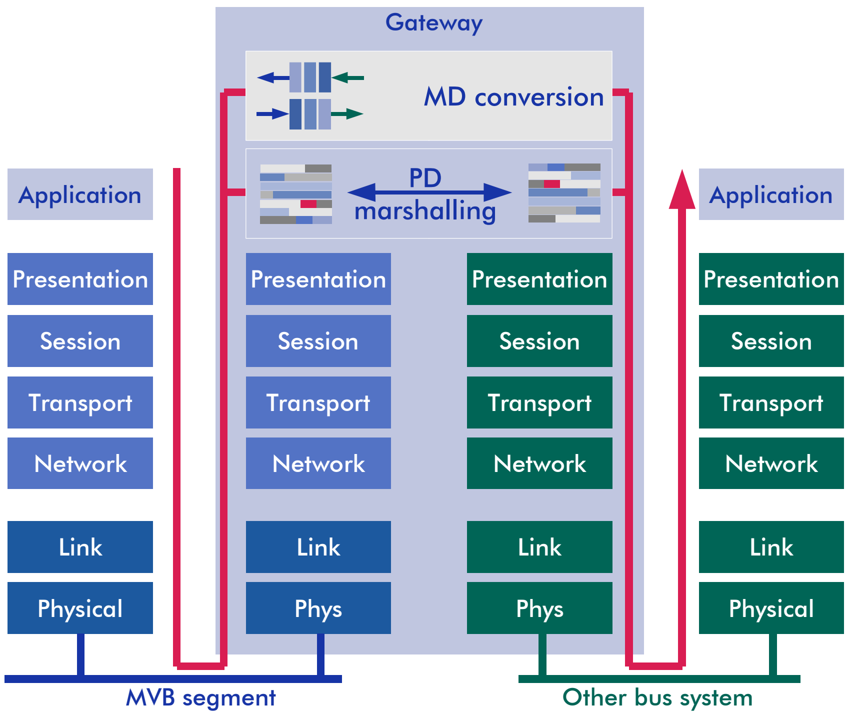
It is possible to link other bus systems to the MVB bus.

This is done by the use of gateways:

- Since the Network Layers are not compatible to each other, the routing of message data has to be done via the application layer.
- The PD marshal picks out data from one side and copies it to the other side; and vice versa.

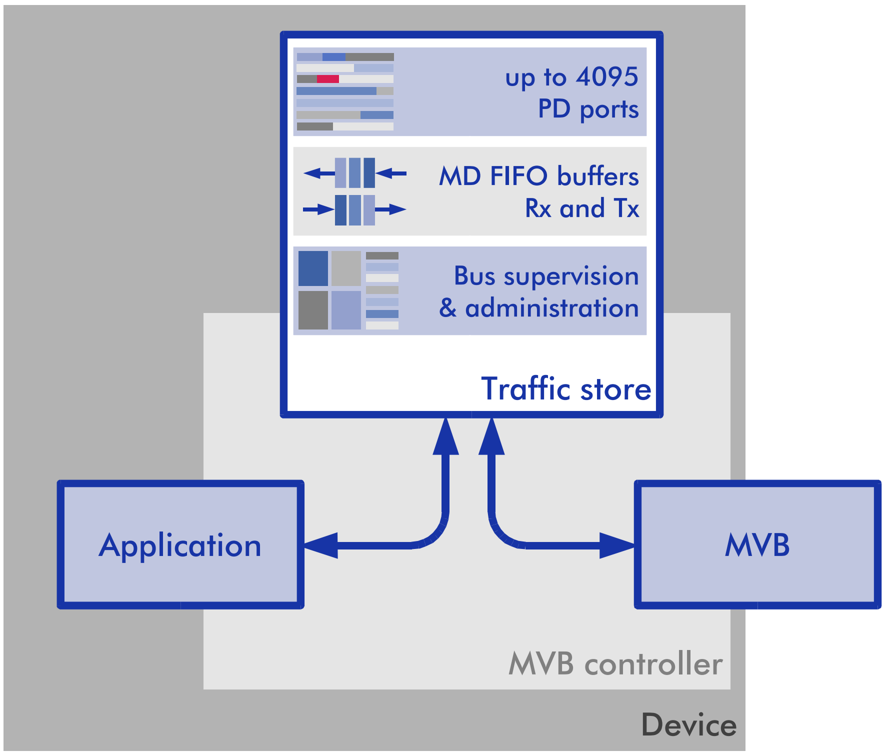
In order to ensure successful operation:

- The protocol conversion requires a common object address space.
- Common data representation and semantics is important.
- Therefore, a standard object description is needed.



Traffic Store Interface

The interface between the bus and the application is the "**traffic store**". This is a shared memory, which is accessed by the application and the bus traffic.



The **MVB- controller** handles the interface of the **application** and the **MVB bus** into the **traffic store**. This function is implemented in a highly integrated logic chip.

The accesses from the application and the bus into the traffic memory are not synchronised to each other: usually, both sides will follow their own timing, independent from each other.

If the accesses occur by coincidence at the same time, an arbitration logic within the controller chip has to decide, which access is to be processed first. During this time, the other side is delayed.

Note: There is no direct interaction from the application to the MVB. All communication has to pass the traffic store, and the timing of bus traffic is determined from the bus administrator on the outside.

How to access the traffic store

For full complexity nodes, the market is divided into two major controller families:

- The "MVBC01" was the first available ASIC chip; originally developped by ABB (now Bombardier) and Siemens. In the meantime, some proprietary new family members appeared.
- The "UART emulation" based MVB controller from duagon. Several versions with application specific optimisation are available.

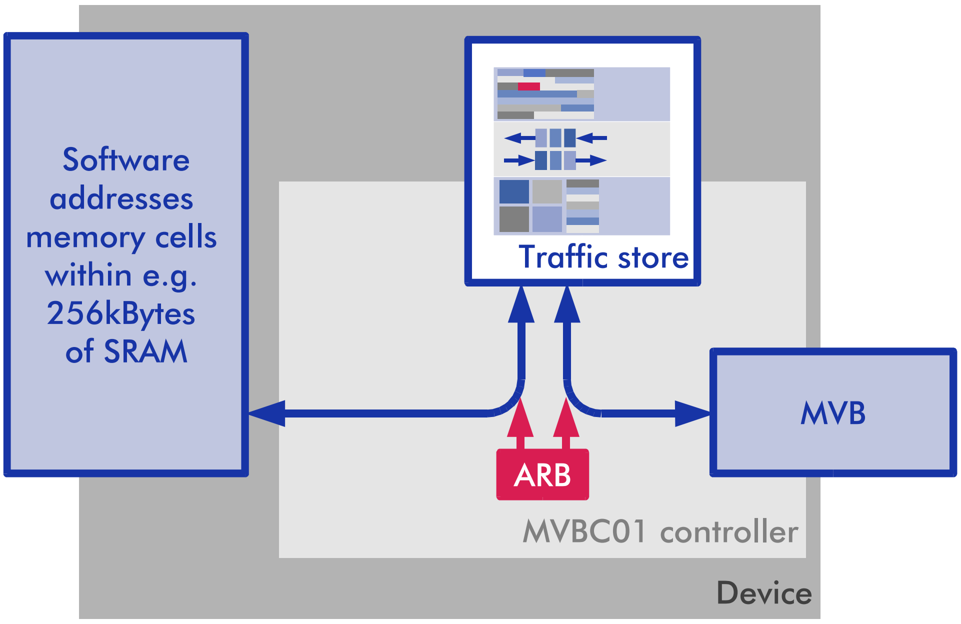
The traffic store interface has to handle the full set of MVB functionality from Class 1 to 4: PD, MD, BA. This requires memory space of at least 256kByte. Since this amount of memory is usually not included within a logic chip, a separate SRAM chip is used.

In both cases, the controller arbitrates the accesses from the application and the MVB side. However, there is a difference about the organisation of data, which is transferred between application and traffic store.

Dual Port RAM Traffic Store Access

The software addresses each single word in the traffic memory. In order to perform a complete Link Layer function like a PD port read, the software will perform a sequence of several memory accesses.

The MVB- controller- internal arbitration works in a word- by- word- manner; and a single access may be delayed (dynamic wait states, up to several 10 μs).


Software addresses memory cells within e.g. 256kBytes of SRAM
Traffic store
MVB
ARB
MVBC01 controller
Device

In order to keep data consistency, the MVB controller has implemented a special protocol to avoid write actions during a read sequence of the same port (called " two page protocol"). While a read of a port takes place in one page, a simultaneous write to the same port takes place on the second page. By this way, data consistency is perfectly kept, as long as the read of a port is completed within the PD repetition time on MVB.

This real time requirement is usually kept by "real time operating systems", for example for PD port repetition rates of 128ms and higher. If faster repetition rates have to be supported, a software expert may introduce an "interrupt disable" during the PD port read. Then, any other interrupt (like the OS tick timer) is delayed until the end of PD port read. This delays the overall interrupt latency figure in the range of several 10 to 100 microseconds.

A problem occurs, when a non- real-time- OS is to be used. A good example is Microsoft Windows in its various derivatives. Several tools are on the market, intended to enhance the RT- behaviour, but these are not very comfortable, since they rely on a specific Windows version; an update may be expensive, uncomfortable or not possible.

To avoid this difficulty between Windows and the MVBC01, a wide spread compromise is to prohibit any use of data consistency beyond one word.

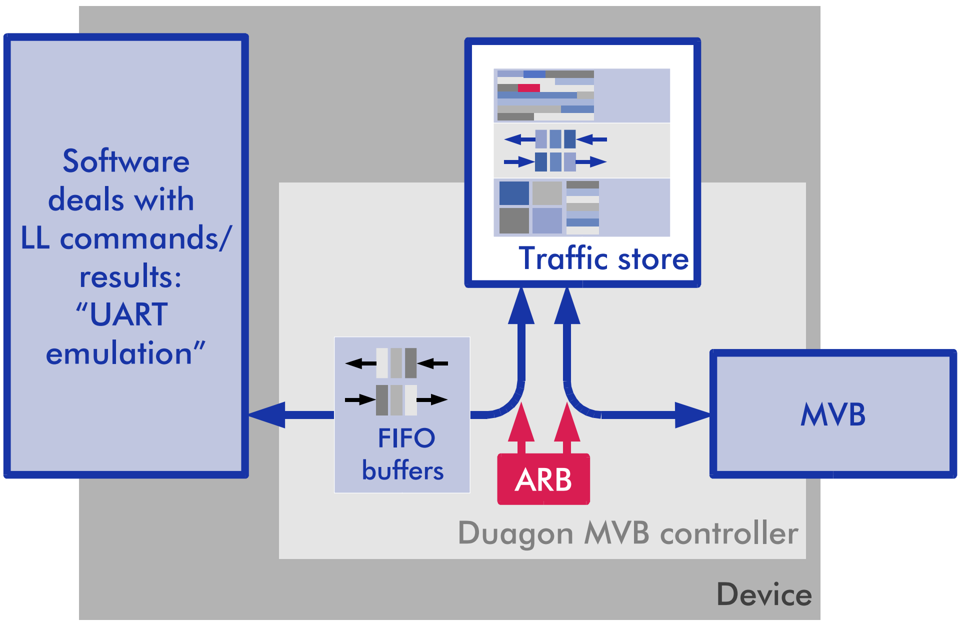
The first representative of this architecture is the MVBC01. Newer derivatives implement "semaphores" in the traffic memory. They temporarily disable accesses from the MVB- side. By this way, data consistency is kept without real time requirements; however newer data is lost during the disable time, which yields additional latency.

Encapsulated Traffic Store Access

The software issues “commands and parameters” into a “serial line” and waits for “results” being sent back. The command packets are defined close to what is defined in the link layer; e.g. they include the handling of complete PD ports.

The transferred data packets contain complete sets for the required action and are stored in FIFOs of the controller chip.

The arbitration to the traffic store is done “packet by packet”. The single access to the UART emulation is of a fixed timing, i.e. dynamic wait states are not required.



No real time requirements are left for the traffic store handling; even the use of Microsoft Windows is no problem any more: Data consistency is always kept.

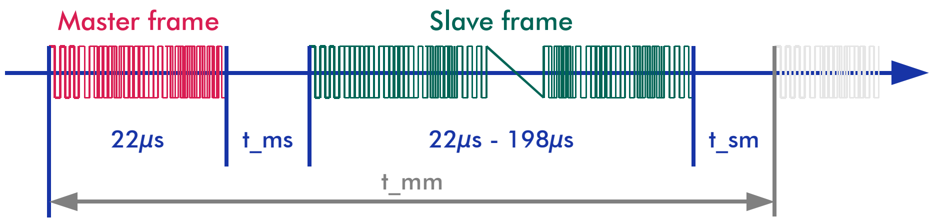
Note: There are still real time requirements; for example application imposed requirements or for the message data protocol stack. However, the involved timings are much more relaxed in terms of absolute numbers: several 100 milliseconds to several seconds.

Note: The expressions “UART” or “serial line” may sound like being slow. This is not true: The UART emulation just uses the simple register model of a UART; the data transmission itself is very fast. The worst case delay from issuing a command to the reply is below 10μs; reading the whole FIFO in single bytes takes approx. 25μs (widely depending on host CPU and local bus speed).

Timing Structure

Frames

All transmission of data is organised within a "master frame – slave frame" sequence. The master frame is always issued by the bus administrator; the slave frame is issued by various devices, depending on which device was addressed by the master frame.



The picture shows the most important timing relationships:

- The length of the master frame (MF) is fixed to 22µs; this represents 16 bits of transmitted data (net data, without start & end delimiter).
- The time between a master frame and its related slave frame (**t_ms**) is from 2µs to 42.7µs. Any slave node has to reply within 6µs; the rest is delay for the frames travelling along the line and crossing repeaters.
- The length of the slave frame depends on the type of data being transferred; with a data contents of 16 to 256 bits.
- The time from the end of a slave frame to the beginning of the next master frame (**t_sm**) is at least 3µs.

The master frame is structured in two bit fields:

Master Frame Structure

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	▨	▨	▨												
F-Code				Address											

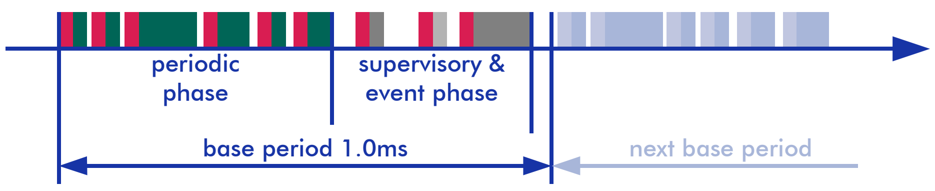
Note: The numbering of the individual bits within a frame reflects the sequence how the data is transmitted over the MVB lines. In usual microprocessors, the numbering of data bits within a word is in the opposite direction.

The table lists all possible MF- SF combinations:

Master frame			Slave frame			
F-code	Address	Request	Source	Size (bits)	Response	Destina-tion
0	Logical	Process data	Single device subscribed as source	16	Process data	All de-vices sub-scribed as sink
1				32		
2				64		
3				128		
4				256		
5	reserved		reserved			
6						
7						
8	Device	Mastership transfer	Proposed master	16	Mastership transfer	Master
9	All devices	General event (pa-rameters)	One or more devices	16	Event identifier	Master
10	reserved		reserved			
11						
12	Device	Message data	Single device	256	Message data	Selected device(s)
13	Device group	Group_ event	One or more devices	16	Event identifier	Master
14	Device	Single event	Single device	16	Event identifier	Master
15	Device	Device status	Single device	16	Device status	Master or monitor device

Basic Period

The basic period is divided into three major sections: periodic phase, supervisory phase and event phase.



- During the **periodic phase**, process data is transferred. The sequence of several MF+SF- frame pairs per base period is defined by the bus administrator list: it may contain a mix of all PD frame lengths, with 16 to 256 net data bits.
  The contents of the periodic phase is determined during the engineering phase and can not be changed dynamically during operation.
- **Supervisory** data is the "device status scan" and the "bus mastership transfer".
  The contents of this phase is automatically generated by the bus administrator and will follow certain deterministic rules. But, there are some aspects being decided dynamically during operation: e.g. the issue of a "bus mastership transfer" depends on the availablity of other BAs.
- The **event phase** basically performs the transfer of message data from one slave node to another. This includes -
  - the identification of devices being ready to send out MD. The bus administrator will perform a search for such devices in an endless loop. Since more than one device may have message data to be sent, the BA must also handle the concurrent reply of several nodes, which is a "collision" of frames on the bus. Following a certain strategy, the "**event arbitration**", the BA takes care that all devices will be able to send out data after a while, and that the transfer of MD is performed within a certain time (real time behaviour).
  - the MD transfer itself. The BA invokes any identified nodes, and these will send out an MD data frame. The destination address of the MD frame is included within the transferred data bits; each listening node will compare the frame contents with respect to its own device address.

  Events, or the transmission of message data, occur with a random intensity: absence of events leads the BA to unsuccesfully scan for MD; during high activity bursts, MD transfer may take place "from everybody to everybody".

A fourth phase, the "guard phase", simply fills up the time until the fixed end of the basic period. Since it does not transmit data, it is neglected here.

From the standard, the base period may also have another length, e.g. 2ms; however this is almost never used in practice.

MF&SF Time Requirements

In order to estimate the amount of MF- SF- combinations which fit into one basic period, the following table shows the worst case time being required for the individual combinations:

Master frame "periodic phase"	F-Code	Max. time required
PD 1 word	0	93 µs
PD 2 word	1	103 µs
PD 4 word	2	125 µs
PD 4 word	3	173 µs
PD 16 word	4	269 µs

The timings shown here assume the worst case delay between the master and slave frame. On a normal working bus, the required amount will be smaller, since the majority of slave frames will have significantly shorter reply times; e.g. for a single cable segment 8.4µs instead of 42.7µs, which is 34.3µs faster.

Master frame "supervisory and event phase"	F-Code	Time required
Bus mastership offer General Event Group Event Single Event Device Status	8 9 13 14 15	93 µs
MD 16 word	12	269 µs

Usually, these frames are automatically generated by the bus administrator. Therefore, the application engineer does not have to organise them (this is done by the MVB controller chip). However, he must leave sufficient time for these frames when allocating time for PD.

Basic period time requirements

The application engineer may influence the available bandwidth for PD and MD within certain limits. Since the overall bandwidth is fixed, this is basically done by moving a trade-off between PD and MD bandwidth.

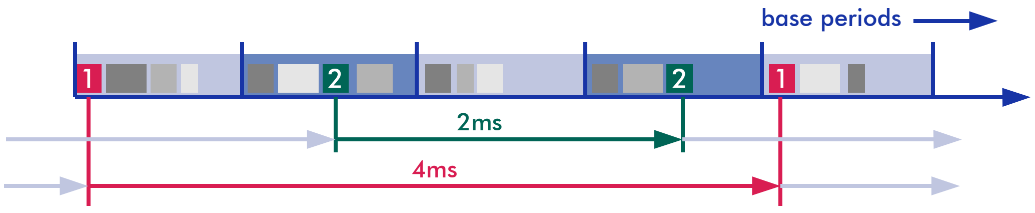
- **Event phase**: The recommended minimum time, being reserved for the event phase, is one "MD transfer" (269μs). The same allocated time can also be used by "general event requests", "group event requests" and "single event requests".
  - **More bandwidth for MD**: For the "classic MVB controller implementations", e.g. MVBC01, the event arbitration will proceed step by step in 1ms- base period increments, with the MD transfer at the end. It did not make sense to allocate more bandwidth for an MVBC01- based bus administrator, since this chip required software interaction for the execution of the event arbitration strategy.
    New MVB- controller implementations (duagon) have the event arbitration completely working in hardware. For them, it is possible to perform even several event requests and MD-transfers within one basic period. This way, additional bandwidth for MD is not wasted. Note: this is also true for saved time due to fast PD- transfers (see previous paragraph).
  - **Less bandwidth for MD**: Of course, it could be possible to allocate the "MD transfer" (268μs) e.g. every fourth base period, only. However, the real time requirements for the messenger software have to be calculated accordingly. Be sure, that the worst case requirements for your application can still be met.
- **Supervisory phase**: The "device status" scan rate is recommended by the standard to be 64 devices per 512ms, which means one "device status request" within eight base periods. This leads to an overall amount of approximately 33 seconds for a complete scan of all 4095 devices. A personal recommendation from the author is to double this rate (16 seconds is still a long time, and one device status request within four base periods does not hurt, yet).
  The "bus mastership transfer" occurs fairly seldom and does not play a role for bandwidth considerations.
- **Periodic phase**: The number, the size and the repetition frequency of PD ports is defined by the application engineer. Any available time for MD is indirectly defined from this, i.e. the application engineer does not have to define the bandwidth for MD, instead he has to allocate PD in a cautious way.

The typical situation will be, that the allocation of the periodic phase is done by a tool, i.e. the application engineer does not have to take care of the individual frames.

For practical reasons, the bandwidth calculation is very often handled in a "statistical" way. Be aware, that the granularity of the required amounts of time may void a statistical calculation: 20% bandwidth for MD sounds nice, but if this margin is equally allocated to the base periods, the remaining 20% of 1ms = 200μs may not allow a single MD transfer!

PD Periodic Polling

The repetition frequency of an individual PD port can be selected from the steps of 1, 2, 4, 8, 16, 32, 64,128, 256, 512 and 1024ms. Due to the fact, that the fast rates will occupy a lot of bandwidth, the repetition rates below 16ms are rarely used.



Example: the PD port 1 has a repetition time of 4ms, and port 2 has 2ms.

The situation for a typical application is, that there will be several hundreds of ports to be served:

- Some “fast” ports, with a repetition rate of 32ms or less, will handle the jobs with critical reaction times.
- “Slow” ports, with 512ms or 1024ms will handle slow moving signals (temperature...), configuration data for remote IO-modules, etc.
- The majority of ports will be somewhere in between, mainly allocated according to general bandwidth considerations.

The application engineer will select a repetition rate for each port. The allocation to particular base periods and the phase relationship between several PD ports is done by a tool.

MD Event Arbitration

"Event arbitration" is an algorithm executed by the bus administrator, which searches for devices having MD data ready for transmission.
The classic event arbitration, as it is described in the TCN- standard and as it is implemented e.g. in the MVBC01 chip, can be described in a simplified way like this:

- The BA starts a new event poll by issuing a "general event request". Three possible reactions can happen now:
- If there is a clearly readable frame from one device, this device will be served, then another "general event request" is issued, this time "not a new poll".
- If there is no reply at all, the "poll" is terminated and everything starts from the very beginning: Obviously, there is no device left, which applies for event service.
- If there is a reply from more than one device, a collision occurs on the bus lines. This situation can be detected by the decoder. The BA will now issue a subsequent "group event request", which invokes half of the devices from before. With repetitive "group event requests" and "single event requests" at the very end, all involved nodes can be found out and served after a while.

As a rule for all participating devices, devices will not start to apply for events during a "poll", a running event round. They will wait until there is a "new round" started. Therefore it is ensured, that every event seeking device will be served before the new round is started. This method leads to a deterministic maximum time, until a random node is served (real time behaviour).

The event arbitration, as it is described in the TCN standard, is "recommended", but not "specified". As written in paragraph 3.6.3.2.4.of the IEC61375, "other algorithms are permitted. ... they may take advantage of the fact, that the identity of participating devices is known, and that events tend to occur in bursts for message data transfers". Newer MVB controller implementations (duagon UART emulation) take up these ideas and implement a new event arbitration strategy. In addition, the new strategy is implemented in hardware (does not need software interaction). Both points together lead to the fact, that the net bandwidth for the application engineer is increased by a factor of 5 to 10.

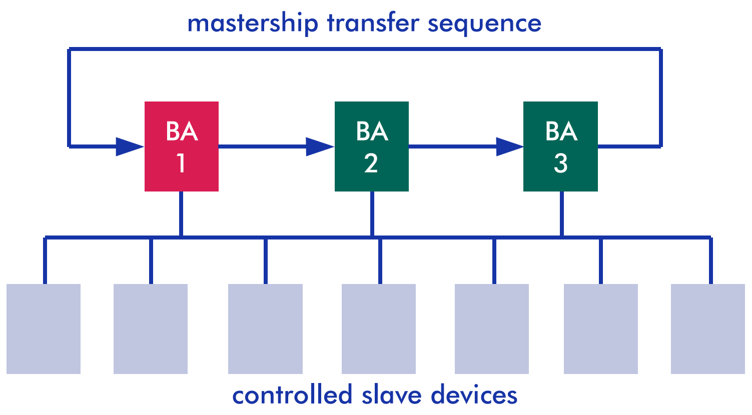
Macro Period

After 1024 base periods, the PD polling list is completely executed and will start from the begin afterwards. This is called a “macro period”.

Turn - BA Mastership Transfer

The medium access is controlled by one of the bus administrators, the master, which remains in control of the bus for the duration of a “turn”.

At the end of the turn, the master tries to pass mastership to another bus administrator (or remain itself master if no other valid BA is found).



The time for a turn is always a multiple of the macro period, i.e. a multiple of 1.024s. A reasonable turn time will allow the device status scan to work at least once through the complete address space: With one device status scan per four base periods, this ends up at a turn time of 16s. This is sufficiently far away from the maximum turn time of 256s.

The “BA mastership” circulates preferably in ascending order of the device addresses. The list of available BAs can be defined in the BA configuration, or it can dynamically be derived online from the device status scan. The latter principle helps to insert new devices into an existing application, e.g. for a diagnostic device.

During power up, a bus administrator waits for a predefined time before starting bus activity. This time is influenced by the device address. Since the device address is unique for all devices on the bus, one of several redundant bus administrators starts first, when all are switched on at the same time.
Note: due to this behaviour, it makes sense to allocate “small device adresses” to the BAs. If the highest possible address is given to a BA, it will wait several seconds before becoming an active master on the bus – try to avoid this unneccessary idle time.

For availability and reliability reasons, the application must monitor the regular BA mastership transfers. Otherwise, a non- operating BA may not be found, leading to a complete bus breakdown if the single remaining BA becomes defective.

MVB System Engineering

Selection of Classes

When planning a new MVB- based control system, it makes sense to review the capabilities of the involved MVB- nodes.

- Some capabilities may be skipped for simple nodes
- It is commercially reasonable to select the required capabilities, only.

In order to simplify the discussion about this issue, some “Classes” have been defined in the IEC standard. Please note, that sometimes there is a more detailed classification used in the market (which is not standardised):

MVB Device Classes

The devices participating at the MVB- traffic, are divided into several classes of “communication capabilities”:

Class 0:	Repeaters and bus converters; working on physical layer with no data interaction.
Class 1:	Process data IO, very simple, often as pure hardware solutions, sometimes without any communication software.
Class 2:	Process + message data, local processing of data, communication software with LinkLayer and real time protocols (RTP).
Class 3:	Same as Class 2, additional application specific software variations, downloadable software configurations.
Class 4:	Most complex bus stations including functions like MVB bus administrator
Class 5:	WTB gateway (nodes with several link layers)

Class 0 and Class 5 devices are usually defined by their function, i.e. there is no headroom left for the application engineer to decide about capabilities.

Class selection

When thinking about your application, try to specify the requirements in the following sequence:

Class 1

Class 1 devices are the cheapest in terms of hardware as well as software and software integration cost. Use Class 1 for as many as possible nodes.

Class 2

The step from Class 1 to 2 is usually reflected in a certain cost increase for the MVB interface implementation. The reason for this is mainly the additional RTP package for Message Data transfer, which contains quite a high complexity.

- For low volume (less than 1000 MVB nodes), it pays off to use MVB interface boards, which have a local microprocessor on board (called the "MVB server"). This one will run the RTP software, and the host CPU has reduced requirements about software integration cost (software integration is the dominating cost factor).
- For high volume (more than 10´000 MVB nodes), it may pay off to port the RTP protocol stack to the application CPU. Now the hardware cost becomes the dominating factor; the high amount of software integration cost is justified by hardware cost reduction.

Class 3

Class 2 and 3 are almost the same for up-to-date control systems. Today, the download of configuration data and software updates is standard; which achieves "Class 3", already.

Class 4

The former cost for the additional bus administrator (the step from Class 3 to 4), is now decreased to an almost invisible level. The reason for this is, that newer BA implementations do not need high speed real time software any more. This makes the BA a "friendly guest" on application CPUs running under tough load. Following this, the BA functionality will be preferably moved to an existing Class 3 node, and not be on own node.

Porting Software into Host Systems

Class 1 and the PD part of Class 2,3

The integration of process data driver software is very simple and cheap: the effort is in the range of 1 to 2 days. Usually, the driver software can be taken (or derived) from an open software; available in source code.

Put some emphasis on clarification of the following issues:

- Does the type of MVB controller require real time behaviour in order to achieve data consistency? If yes: implement e.g. interrupt blocking mechanisms or skip “data consistency” at all. See page 14 for data consistency in general and page 22 for the implications from the traffic memory interface.
- Are application accesses required to happen from several tasks? If yes: The use of semaphores has some impact on bandwidth and/or latency of PD reception. For the practical impact, see page 39.
- Does the application use the Layer 2 or Layer 7 interface?
  In simple cases, it may make sense to use the Layer 2 interface (PD ports): the application reads or writes complete PD ports; the PD port contents is resolved by the application software into several variables. Sometimes it is helpful, to distribute the allocation of variables centralised for the whole MVB traffic. Then, the particular node software tool has to import the variable definition, which delivers the variable name, the variable type, and the location of the variable within the PD port. In this case, you will work preferably on Layer 7 interface (PV variables).

Sometimes, the lower hardware cost of pure Class 1 nodes generates the idea to implement functionalities of the higher Classes via PD and some user written software. Be aware, that this approach may turn out to be very tricky: You will have to imlement an own protocol for it. In particular, we **do not recommend** the following issues for pure Class 1 devices:

- Download of data into a local flash memory.
- Communication with data structures with a size far beyond 32 bytes; for example long text, diagnostic data sets, complex configuration data sets.

Class 2,3 Message data

The RTP real time protocols package is a very complex software. Today, there is still significant software license cost and an NDA contract attached to it. The complexity of the software itself as well as the integration effort into products usually requires at least one engineer, working on this topic all year round.

These questions are in the foreground:

- The operating system must support real time behaviour. If the MVB interface is an “MVB server”, this requirement is restricted to this unit, only. If the RTP are ported to the main application CPU, then this one has to support real time.
- The RTP require support of semaphores, which are preferably to be used from the OS.
- The RTP require a number of (software) timers.

All together, skilled engineers will require in the range of some man months for a complete porting of the RTP on another platform.

For the MVB server approach, the effort is reduced to a two day work: only a slim client software has to be ported, with simple requirements related to the OS: no real time, no timers, maybe just semaphores (in case of multitasking applications).

Class 4 Bus Administrator

Previous controllers required a BA software with very fast interrupt service routines. This is of course a clear interference with demanding applications.

Newer designs include the BA completely in hardware. The remaining requirements upon software are quite trivial: The BA must be started by software, requiring some memory space for the configuration file. After the BA has started, absolutely no software interaction is required.

How to put a new node into operation

After having the software ported, the whole device has to be tested first time with MVB. The proposed sequence shown below is just a recommendation, to be adapted according to the particular project requirements.

Hardware Integration of MVB controller

Try to initialise some PD ports, read and write the same port. This activity takes up the contact with the MVB controller. When yo can read a PD port with a value (which should be the same you have written previously), then the hardware integration is obviously correct (at least the most important parts). For this test, you do not need any communication over the MVB cable.

RTP software porting

The same you can do with message data, too. Call a function, which is implemented on the same node. This does not need MVB interaction, too. It is just a test of the successful software porting.

Hardware of MVB attachment

Now connect the new node with a BA bus administrator and a diagnostic device (most diagnostic tools already contain a BA). The new node must be recognised with its “Device Status” on the preconfigured address.

Check the contents of the device status word:

The RLD- bit must reflect your physical layer choice:

- If you use Line_A only, RLD will be 1 (and LAT=1, too).
- With redundant lines, RLD must be 0.
  Now make a short circuit on MVB Line_A (it is the one between pin 1 and 2 of the Sub-D- connector): The result must be RLD=1 and LAT=0.
  Then make a short circuit on Line_B (between pin 4 and 5), the result must be RLD=1 and LAT=1.
  Be sure, to really get a new DSW from the bus – do not use old values (use freshness information!).

Typical DSW structure:

Bit positions															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Capabilities				specific				common flags							
PD	BA	GW	MD	Ax1	AX0	ACT	MAS	**LAT**	**RLD**	SSD	SDD	ERD	FRC	DNR	SER
0	0	0	1	0	0	0	0	x	x	0	0	0	0	0	0

This test sequence demonstrates, that the MVB connection is working in general, and your device is able to communicate on both Lines, in receiving as well as in transmitting direction.

BA attachment

Now use the bus administrator with a configuration close to what is required in your application project.

Check the ports, that are defined as “source” on your device. You may again use a diagnostic tool to receive these ports via the MVB cable. Be sure to have the BA switched off on the diagnostic tool; otherwise the regular BA may be out of service.

Add a device, that generates with an active source the PD ports, which are “sink” on your device. You may use the diagnostic tool for this, or the finally intended MVB devices of your application project.
Now the PD sink ports on your device must show low freshness values, indicating the successful reception of new data.

The test demonstrates so far, that the configuration of the BA and your PD addresses match to each other. If they do not, communication will not take place (freshness indicates antique data).

Check list for missing frames

When you enter the fact, that PD frames do not arrive at the intended device, several potential reasons should be checked:

- No BA: The BA is not existing, not configured, not active.
- Wrong configuration: The configuration of the BA, the source of PD, or the sink may be wrong. A high potential for errors was found by misunderstanding of hexadecimal versus decimal numbers...
- There is a second source node on the bus, which send out data for the same PD port. This leads to repetitive collisions on the bus, which is of course illegal.
- Wrong physical layer: all nodes must have the same physical layer, EMD or ESD.
- Device internal noise and/or the ESD end delimiter issue inhibits successful reception of data (less probable for lab setups).
- The cable attenuation does not match the real line extension.

System test

Sometimes it sounds a little bit boaring, but it really makes sense:

Check all the details of your system in an isolated way.

- Are variables understood in the same way all around the MVB system? No byte swapping in "integer" formats? Are all PV variable definitions correct, i.e. all bits at the right place? Does your device start with the intended variable defaults or does it take a while until they settle?
- Is the start up and fail safe behaviour of your device correct?
    - A hardware reset must stop MVB transmission immediately.
    - The same is true for a watchdog reset.
    - Before starting the regular MVB communication, the device must not be active on the bus. For "power up", this will be true in most cases, but also after reset without power-down?

Several companies have introduced "test suites", which actively scan all possible aspects of the TCN standard. Usually it pays off to make use of this, even though it costs a few days of additional work. Finding e.g. misunderstandings in variables is sometimes a very time consuming job, especially when you are already on a vehicle (and everybody picks on you...).

Final Performance test

Check your application in a complete system environment (like on the vehicle). Use test sequences, that emulate the real world as close as possible.

This will give you confidence about the more complex properties, like bandwidth on application level, latency times, etc.

Real world performance

What are realistic performance levels from the point of view of the application software?

- Of course, this question depends to a high extent on the used components, i.e. the application CPU, operating system, the driver software architecture, the hardware interface to the MVB controller, the MVB controller itself.
- Anyway, some very general numbers help to learn something at least about the "order of magnitude" for MVB communication.

The base for the examples shown below is an ARM7 RISC CPU with approximately 20 MIPS, running an eCos real time operating system.

The application is a specific type test software, with an emphasis on generating a high load in PD and MD, i.e. the values do not represent optimistic calculations of single accesses, but a sustainable average.

Name	Value	Comment
PD_access time	124 μs	Access of one PD port via API function call, average value over 1000 accesses.
	approx. 250μs	Same test as before, now access via local CPU on MVB interface (MVB Server)
	500 μs to 800 μs	Same test as before, now access of a PC (AMD 500MHz) via a standard Windows COM driver to an MVB Server interface (see note below)
	up to 300μs	Same PC as before, now using a specific Windows driver.
OS_delay_semaphore	32 μs	According to the customer's software architecture, the use of semaphores may become neccessary. Average value over 1000 accesses; ARM7 platform
MD_net_bandwidth Total value for all MD activity	11 kBaud	Tested under heavy load with 26 nodes sending and receiving data. Note: for the individual MD link remains in this case 421 Baud. PD allocated to yield an approximate PD:MD relationship of 70:30 (usually recommended for high PD load). "Classic" BA and RTP implementation.
	100 kBaud	Same test as before, but now with new hardware based BA and optimised RTP software. Now the individual MD link (1 from 26) gets 3.8 kBaud.

Note: Using standard Windows drivers is very convenient, since it does not require to apply a Windows driver yourself. However, the performance is less than perfect; which is OK in a lot of applications. For demanding applications, we therefore recommend to use a specific Windows driver.

Failures, Failures, Failures...

The railway business asks for -

- **Availability of operation**. The absolute level of the required availability is higher than what simple equipment can deliver.
  The usual strategy to overcome this situation is the application of redundancy, i.e. some devices may fail, but the overall function is still delivered since a second device takes over the job.

Availability

Probability of a system to be ready for operational use. Includes properties coming from "failure rate", but also service and repair times.

- **Safety of operation**. If any part of the equipment fails, it must not lead to a safety critical situation.
  In order to tackle this problem, the failure effects of each part of equipment have to be investigated and the internal structure of devices has to be built in a way, that dangerous failure effects are avoided.

Safety

Statistically determines the extent, how far adverse events are avoided.

Usually the failure mechanisms are well known. For each device, the related failure modes can be estimated by statistical methods. For example, the failure of a light bulb is in most cases the break of the filament, less often it may be the loss of vacuum or contact problems.

This knowledge leads to methods, how to avoid unwanted effects. Some successful strategies are:

- **Enhance the internal structure of the equipment.**
  Example: Using LED indicator lights instead of light bulbs is more reliable (LEDs simply live longer...) and more safe (LEDs usually fail by slow reduction of performance, i.e. their complete failure can be avoided in time by replacement; light bulbs fail suddenly).
  Another example is simplicity and/or integration density of equipment. This follows the simple rule: The more you build in, the more will fail. Older electronic equipment usually requires a lot of components to fulfill a certain function. Following the line of technological progress, newer high integration approches will deliver the same function more reliable.
- **Avoid early failures by burnin.**
  "Early failures" arise from hidden defects. The defects are there from the very beginning, but they are hidden, i.e. they are not visible for test methods. The usual way to find these defects is to stress the device under heavy load conditions and watch the behaviour. When this "burn in" is done in the controlled situation of a manufacturing plant, the failures can be repaired in an easy way, requiring less cost than repair in the final operation.
- **Avoid "wear" failures by controlled replacement.**
  A good example is the replacement of various parts of a brake before they fail. Knowing the typical life time of the parts, a "service plan" is installed, applying regular checks and replacements.

- **Apply functional redundancy:**
  Add more parts to the system in a way, that the additional parts take over the function of the other parts in case of failure. Example: the doors are usually opened by interaction with electronic equipment. If this fails, it is good to have an additional, mechanical mechanism which can be used in emergency cases.
  A special application of functional redundancy is the multiplication of existing equipment. If you use just one light bulb, you will sit in the dark upon failure. If you apply two or more light bulbs, then the failure of a single light bulb will not be relevant.
- **Apply check redundancy:**
  Add more parts/ functionality in a way, that wrong conclusions can be avoided.
  Example: A "Pt100"- resistor can be used for temperature measurement; a certain resistance relates to a certain temperature. If the cabling fails with e.g. short circuit or with broken wires, then a wrong temperature will be measured. If you add an additional detector for "resistance close to short circuit" and "resistance close to open circuit", then the cable failures can be found. An "invalid" signal will make any users of "temperature" aware of the fact, that they have to work without "temperature". They will select a "fallback strategy"; e.g. switch off the air condition.

Integrity

Determines, to what extent a systems delivers correct data when it sends data.

Applying Redundancy in general

If “redundancy” is to be applied, some basic principles should be in mind:

Are failures independent from each other?

Example: Usually, having two light bulbs works very fine. This is due to the fact, that the most probable failure mode of a light bulb is the break of the filament, which is an “open circuit”. Assume, the most probable failure mode would be a “short circuit” (there have been such interesting products on the market...). Then, the failure of one bulb would make the fuse trip, i.e. the second bulb is dark, too: Adding a second bulb makes no sense, unless you add a second fuse circuit.

The conclusion is: redundancy works only, if the failure of one part is **independent** from the failure of the redundant parts.

How to use redundant software?

As a matter of fact, also software can fail. There are two ways for successful use of redundant software:

- With “software diversity”, i.e. two redundant software implementations, being developped by different teams, with different tools, with different operating system, checked by several type test approaches, the outmost level of low “failure rate” is reached. Of course, this is very expensive, since the implementation is doubled.
- With no software diversity, i.e. the same software, just used in two different instances, the independence between the nodes is lost. The same failure will occur on both devices at the same time.
  Therefore this approach makes sense for getting the benefits from redundant hardware, only.

Please note: Software does not **become** defective, software **is** defective: Whatever the failure is, it is present from the very beginning of operation. There is no way, that software becomes defective during run time; this is possible for hardware, only (loss of memory, wrong calculation results, etc,).

From this point of view, the quality of the type testing during the engineering process directly influences the “software failure rate” later on. It is highly recommended to invest reasonable time here!

What is the probability for a failure?

It is a common situation, that the failure rate of a certain component is considered to be too high, but adding a second component of the same type reduces the failure rate to an acceptable level.

Example: Assume, the probability of failure for a light bulb is p_1=0.0001 failures per hour. The second bulb will have the same probability, $p_2=p_1$. Since the both failures are independent from each other, the probability that both fail at the same time is $p_1 * p_2$ = 0.000 000 01 failures per hour (once in 11000 years, like hitting a jack pot). The result is, that there is still a probability for a complete failure, but the absolute probability for occurence is very small, i.e. you can forget it for practical considerations.

Conclusion: be aware of -

- the absolute height of the **failure rate** for your equipment and
- the **requirements** from your customer about the absolute height of failure rate. The match of both will determine how far you should go in applying redundancy, which in turn defines cost.

What happens after the first failure?

Example: When you have two light bulbs, the first one failed already, but it is not replaced, then the probability for the second one to fail is p2=0.0001. This is after one year, which is not to be neglected!

Conclusion: A process must be installed, which **detects** failures and **replaces** failed components. Otherwise, there is no real benefit from redundancy.

Are there "single points of failure"?

The usual "redundancy business" is to arrange components in a way, that their failure rate p_1 is not simply added to the overall failure rate. If the component is in redundant relationship with other components, the procuct p_1*p_2 is added; which is usually lower by several orders of magnitude.

Single point of failure

Situation, where the failure of a single component will stop overall functionality of a whole device/ system.

This leads to the more practical rule of thumb, that there will not be multiple failures at the same time. This relies on the assumption, that the time to repair is much faster than the time to fail for the second, redundant component.

However, it is sometimes not this simple to create structures without single points of failure. Very often, they are forgotten during the engineering process.

As a check, it is reasonable to actively search for such unwanted situations of "**single points of failure**":

- Usually, they should be avoided by application of clever architectures.
- Sometimes, it is not possible to avoid them. In these cases, the components have to be of extremely low failure rate; which usually leads to the use of very simple mechanisms.

Applying Redundancy in MVB Based Control Systems

There are several areas within an MVB system which allow redundancy:

- Wire redundancy: please refer to the description of the physical layer, page 57.
- The chapter about “Network Topology” includes information about the use of redundant bus segments, see page 49.
- The “bus adminstrator” may be redundant, too. See page 32.

In this chapter, lets have a closer look into “safety” and “redundancy of bus nodes”:

Safety

The basic principles of the MVB allow a “quite safe” operation:

- The integrity of the transmitted data is very high
- The transmission principles are deterministic, i.e. the do not contain statistical approaches like the collisions in Ethernet
- The API software interface supports and encourages the use of check mechanisms for variables
- The whole concept integrates redundancy for availability and integrity at various places.

From this point of view, the MVB includes the basic prerequisites for safety relevant applications. At least, the MVB carries much more “safety features” than other field busses.

Approval

Unfortunately, the approval for such kind of application is very hard to get: It would mean, that the complete system with all nodes and all application software has to be analysed for deterministed behaviour. This is quite complex, especially since a high amount of the functionality is implemented in software. This requires to analyse also the operating system and the more general hardware platform with CPU, timers, memory, etc.

- Therefore, the **high complexity makes a complete certification almost impossible**.

Once having achieved such an approval, it would be void if only one small part changes. This is a clear contradiction to the required flexibility during the engineering process. Another example: for practical reasons, some people need to use a Microsoft Windows operating system. Can this be certified for safety? How to deal with updates once per year? My personal recommendation: better do not use it for safety critical applications.

However, under certain conditions, the MVB can very well be used in safety critical applications. Example: The braking system may use the MVB for the operational braking. As long as there is an emergency brake in parallel, the MVB does not neccessarily need to be approved for safety.

This means basically, the MVB can be used in functional redundancy with other devices. The sense of this arrangement lies in the fact, that the MVB delivers a lot of additional features:

- The safety relevant parts (outside of MVB) will most probably be very simple. They are reduced to the absolute minimum.

- The MVB will bring a lot of "comfort", "availability" and "cost reduction" functions. As long as the MVB works, it will be the preferred way of communication. The MVB will help to solve additional questions like: How many emergency brakings are allowed until the train has to go into the repair work shop? How the condition monitoring data is collected and delivered to the outside?

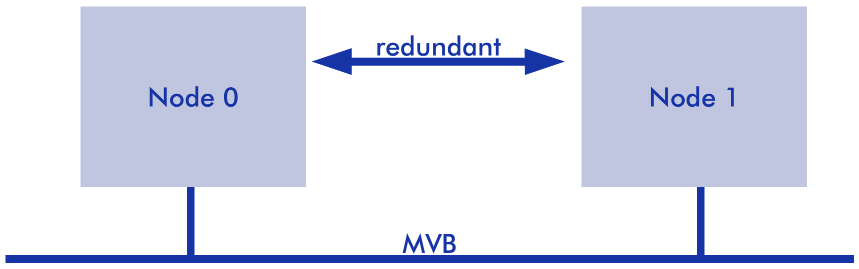
Personal note from the author

According to my knowledge, there was never an approval for e.g. life critical applications of the MVB. Under certain conditions, like limited systems with very low software content, experts have indicated to me, that such an approval could be possible under reasonable effort. At least as you consider an effort of six man months to several years "reasonable". Comments invited!

Redundancy of Bus Nodes

Duplicating bus nodes is simple:

- Bus nodes in a redundant relationship to each other can be connected to the same MVB segment. The MVB hardware connection almost perfectly separates the failure effects from one node to the other.
- Since PD process variables are distributed by broadcasting over the bus, both nodes will see the identical "sink" ports, on identical addresses.
- Both nodes have to use different addresses for PD "source" ports and for device addressing in order not to collide during transmission and to be recognised as different nodes on the bus.
  The different source ports allow a transparent verification of both redundant units, i.e. both units can be checked for failure or disappearance.
- For message data, the involved nodes are "device addressed", i.e. they are distinguished from each other.

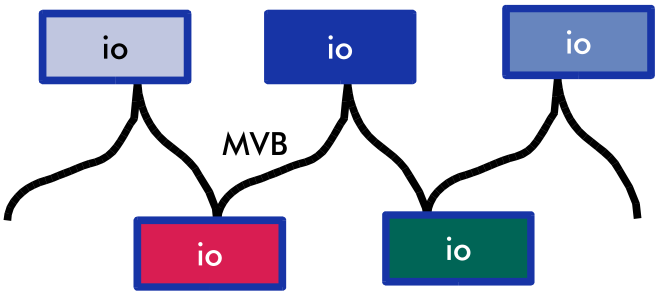


Note: In some applications, application engineers decide that the "source" ports are not on separate addresses. Instead, they dynamically enable and disable their local source configuration in order to become the "active" of the two redundant nodes.

Geographical Addressing

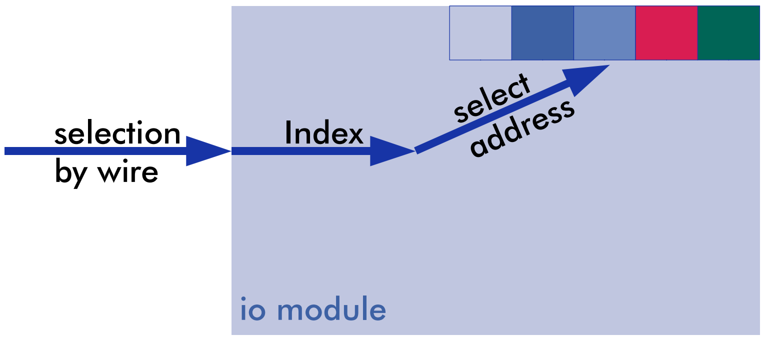
The method of “geographical addressing” is very helpful for applications with several identical devices as bus nodes. This may be for the purpose of “bus node redundancy”, but also for the more general situation of multiple devices. The example below describes the use of several remote IO devices:

When there are several identical modules in one MVB system, their MVB addresses must be somehow different.

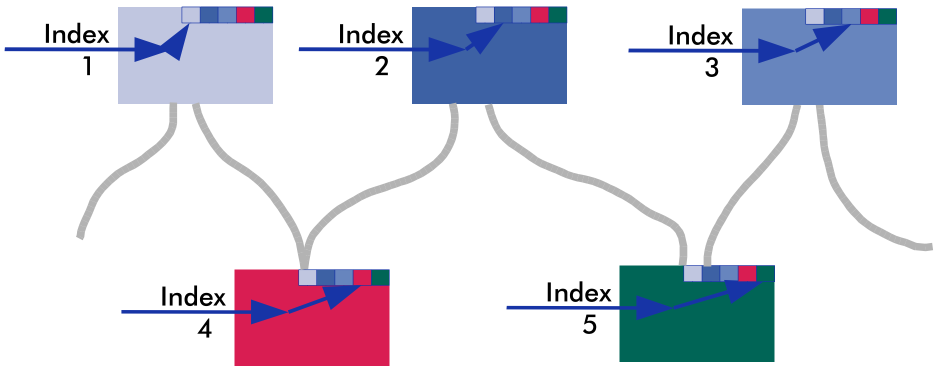


However, it is very useful to have only one type of module within a vehicle; i.e. all modules can be exchanged with each other and the correct function remains at the right place.

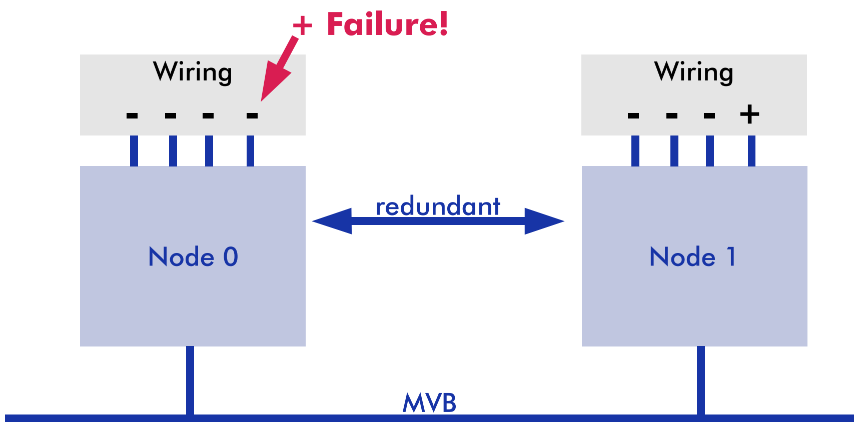
Each module must “know”, where it is (i.e. which function is to be performed). The external wiring tells the module on some special “code” pins, what “**index**” it has. The index has a direct influence on the MVB- addresses.



When exchanging modules from one location to the other, the wiring remains at the right place and selects the correct functionality:

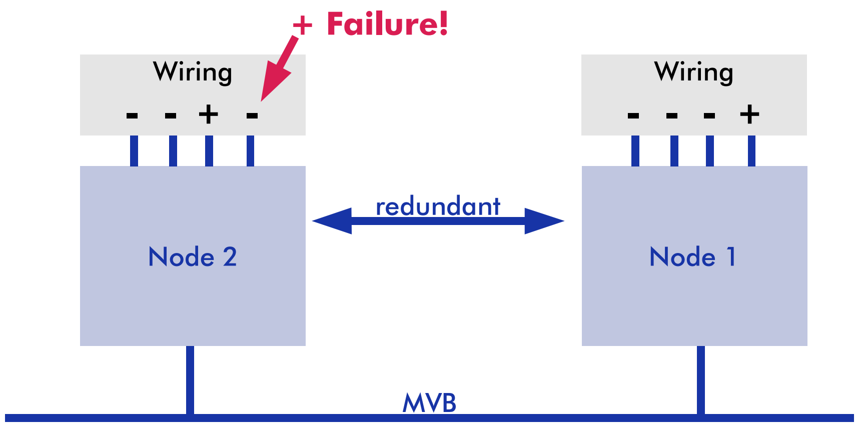


There is a remaining single point of failure, with a relatively small, but existing probability:



A failure of the input may make both devices operate on the same addresses. This causes usually a loss of functionality on both devices.

In order to avoid this single point of failure: define the geographical addressing in a way, that a single failure (wire break, short circuit) does not select an existing other node:



In this example, the failure on the input pin causes Node 2 to work on address “3” instead of “2”. This makes the device inoperable. However, the other node is not disturbed and may work on address “1”.

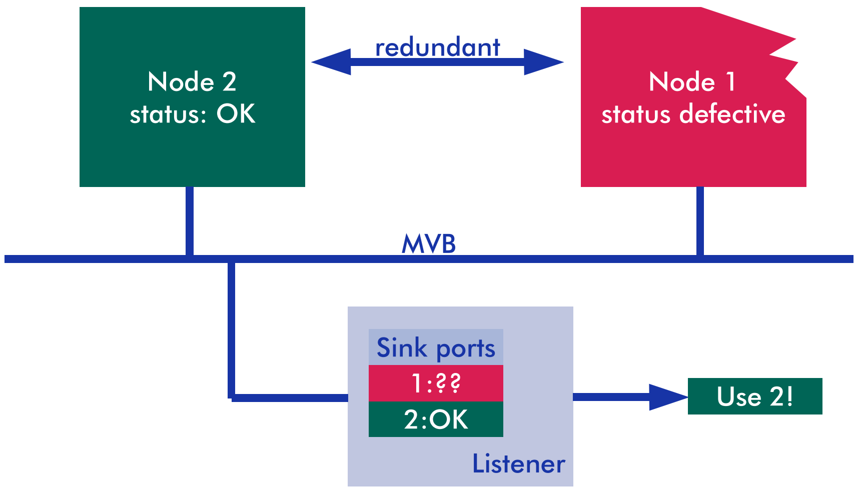
Decision making

Each of the two redundant bus nodes will distribute its own source ports. From the functional point of view, both ports deliver the same information.

Everybody, who is listening to these ports, has to take a decision: which one to take?

Example:
Two vehicle control units are redundant to each other. If one of them fails, the one still being good shall be used for valid output data.

Under normal conditions, i.e. when both vehicle control units are up and working, they will send PD telegrams to the listener. They will have different PD source port addresses, called in this example 1 and 2.



The listener must decide now, which one of two different PD telegrams controls its outputs.

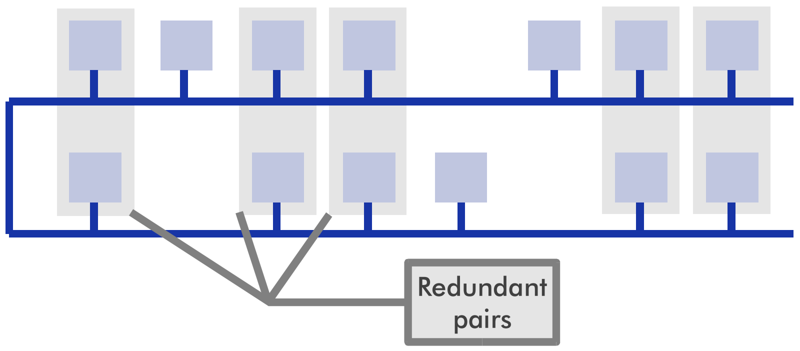
- The most simple approach takes the “freshness” from the sink time supervision. This approach assumes, that a failure of a redundant node is expressed in its missing source ports. This is a natural fact for popular hardware errors like missing power, broken internal logic and similar.
- More elaborate approaches may additionally use check variables and the like. By this way, the listener may distingiush more information about the kind of failure.

In any case, the listener has to report an error, if one of the redundant nodes is missing: This should lead to repair activity for the defective node.

Network Topology

Single Segment

The most simple MVB system is just one segment, and a number of nodes connected to it. Some of the nodes will be in a redundant relationship to each other, some not.



The redundant device pairs are used to implement critical functions. Example: Minimum of control elements on the driver´s desk in order to “limp home”.

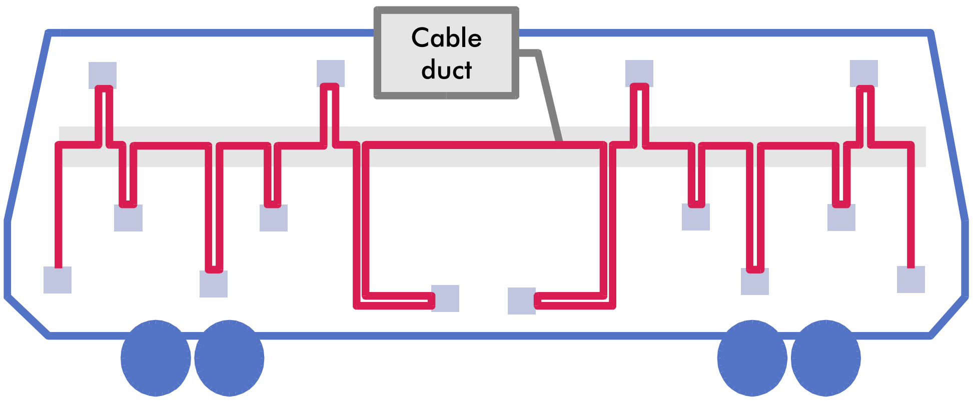
Nodes without a redundant partner will implement functions not being vital: comfort equipment, diagnostic devices, devices with the ability for manual interaction.

This configuration requires the MVB wire redundancy to be used: otherwise, the one MVB line would be a single point of failure.

Note: All devices have to be of the same type, ESD or EMD (since there is one segment, only).

Cable Length Limitation

The single segment length may hit its allowed maximum: when going in and out to all device mounting locations, a high amount of cable length may become neccessary. This is particularly true when using a “cable duct”: Real examples have shown a cable length of up to 200m within a vehicle of 20m length!

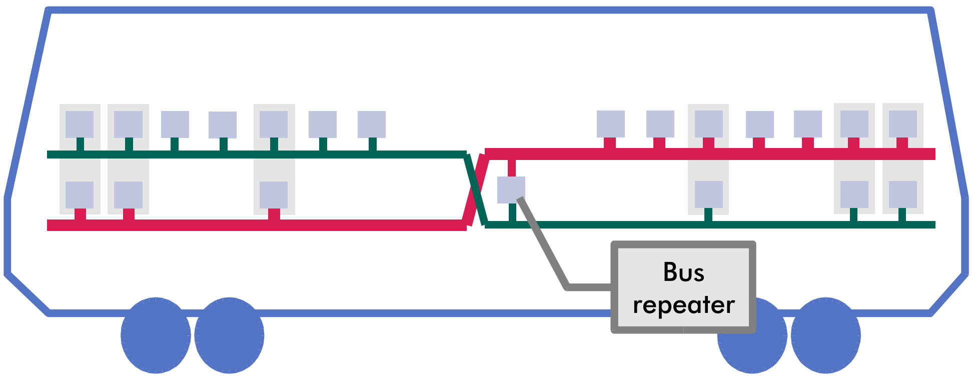


Therefore a very general hint:

- When having more than 10 MVB nodes distributed over a vehicle, check the length of the cable segment.
- For more than 20 MVB nodes, it may become difficult to implement the cabling within the 200m- limit for one cable segment.
- The full number of up to 32 nodes is possible with optimised cable routing and “clustering” of nodes, only.

Double Segment

This topology is very often used, since it relates very close to the properties of symmetric vehicles:



There is one bus repeater, which connects the both segments.

This approach has a very interesting property: The two segments may be implemented without wire redundancy. Since all of the "redundant pairs" are connected to different bus segments, never both to the same segment, a complete segment may fail and the vehicle is still operational: the remaining "good" segment reaches all important and half of the remaining devices.

The bus repeater has the job to -

- pass information flow between the segments during regular operation,
- block any destructive influence from a defective segment to the good one.

Interesting: The bus repeater itself does not need to be redundant. In case of a failure, both segments may even work on their own, completely independent.

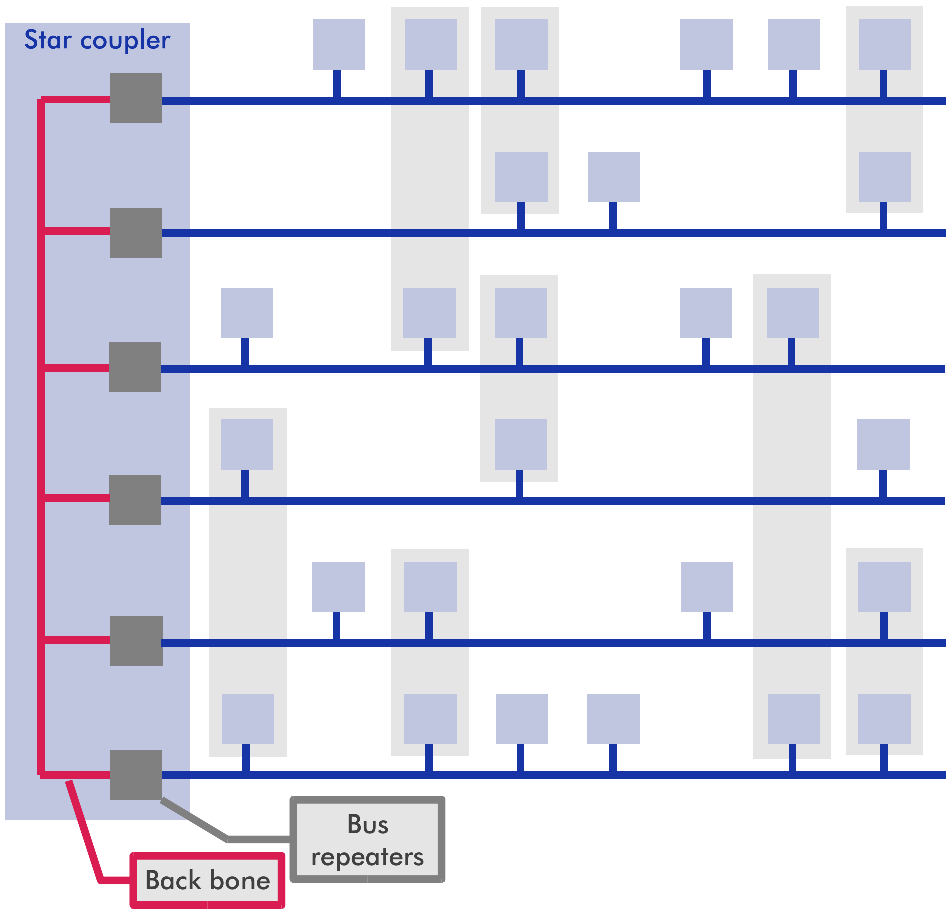
Note: Theoretically, the devices of the two segments could be of different type, ESD and EMD. However, this is obviously a not very favorable solution: It is better to have one type of bus and for repair purposes keep all nodes exchangable with their redundant partner.

The cable length limitations are a little bit more relaxed compared to the single segment structure; two segments may serve twice as many nodes. However, for complex vehicles, it may not be sufficient: follow the recommendations of the "single segment" approach; now for each segment separately.

Star Topology

This structure is a very general approach and supports a huge amount of bus segments.

- Usually, the number of the bus repeaters (and therefore the number of segments) is minimised in order to reduce their cost.
- The segments are easily adapted to different physical layers (ESD or EMD).
- Since the single segment cable is shorter compared to a single segment, it may become a cheaper cable (more attenuation, operation less sensitive to EMI).



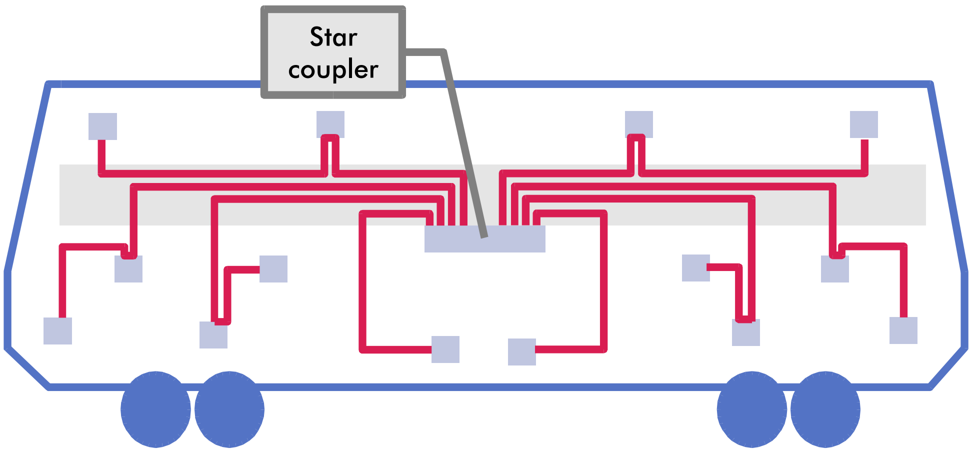
The single segments may have wire redundancy or not. The bus repeaters will follow this decision on their segment side.

The back bone and the repeaters may be concluded into one device, a “star coupler”. In this case, the back bone may disappear completely within the new device.
Please note, that this more general star coupler may serve several nodes on each segment. A classically pure “star coupler” would serve just one node on each segment; as it is the case for the OGF (Optical Glass Fibre).

The application will most probably require, that the various segments are not separated in case of failure. Following this, the back bone must have wire redundancy, or the complete star coupler must be redundant.

This approach puts the emphasis on the cabling in a different way:

- A single segment is routed from the star coupler to a mounting location, where one or more nodes are served.
- Defects in the cable can be found more deterministic, since the failure is already allocated to one segment with a reduced number of nodes.
- Segment cable length limitations usually do not play a role with this approach.



It depends upon the particular application, whether the star approach pays off or not. Costs for cables, commercial availability of EMD versus ESD+- nodes, cable diagnostic efforts during life time, cost and failure rate for the star coupler itself account to a complex situation, which should be resolved for each project separately.

Physical Layers

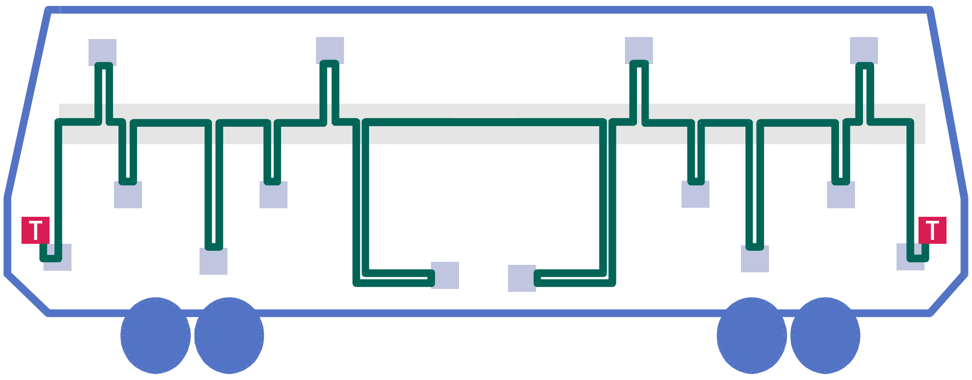
There are three separate physical layers available:

- ESD: Electrical short distance bus,
- EMD: Electrical middle distance bus, and
- OGF: Optical glass fiber bus. Due to the fact, that the OGF is very rarely used, it is not described in detail, here.

All these physical layers can be mixed within an MVB system by the use of re-peaters, switches or gateways. However, nodes of different physics cannot mixed within one bus segment.

Termination

A common property of the wire- based physical layers is their need for termination in order to keep clear signal wave forms (what happens if the terminators are forgotten show the measurements on page 98):



- The transmission medium is linear, i.e. it does not allow to have branches, where the signal may travel into several separate directions.
- The transmission line has to be terminated at both ends in order to avoid reflection from an open end. The termination must match the characteristic impedance of the transmission line.

Please note: Depending on the cabling concept, it is typically required to have two types of terminators: one with a male SUB-D- connector and one with a female connector.

Bias voltage

For the ESD bus, the terminator takes over an additional role: it delivers a bias voltage to the transmission line. „Bias“ is a voltage, which is applied between the signal lines at any time, i.e. also during idle times.

The basic idea of the bias voltage is to reject noise: the receivers detect signal level changes in the area around the differential zero voltage. With a bias voltage, any noise must be higher than this bias in order to be recognised. Of course, the signal itself must be higher than the bias voltage, too.

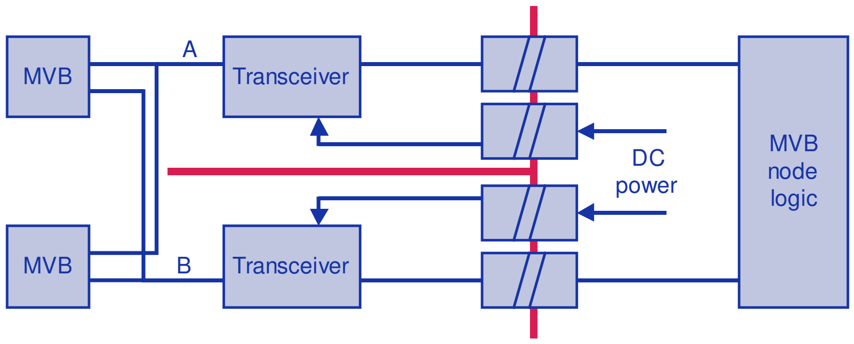
The „bias“ is not applicable for the EMD bus. The transformers, being intended to deliver the galvanic insulation to the node logic, do not conduct DC. This leads to the fact, that noise rejection has to be done in a different way.

Insulation Concepts

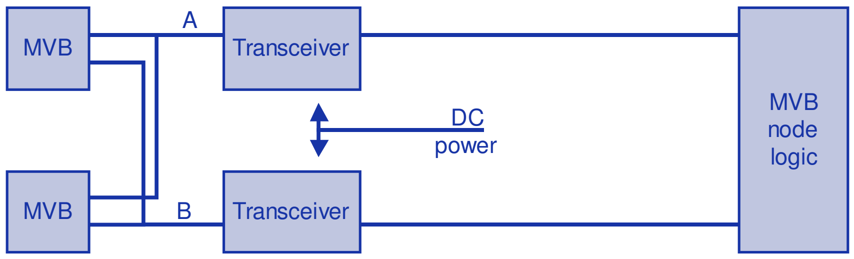
In general, the maximum insulation withstand voltage is 500 V_{AC} for any galvanic insulation.

Insulation ESD+

The insulation separates the node logic (the MVB controller chips, CPU, other electronics) and the RS485- transceivers. Usually, the insulation must be crossed by power (a small DC/DC- converter) and signal lines (opto couplers).
One additional galvanic insulation separates the two MVB lines A and B and their related terminator power supply lines, too. This is not required by the TCN standard, but may be of benefit under certain conditions.

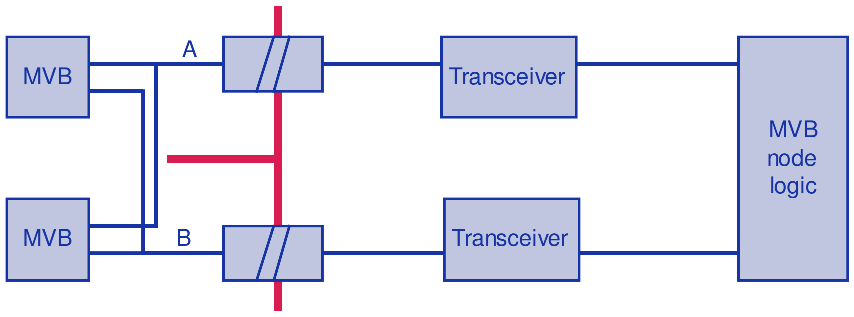


Note: There are some devices on the market, which do not separate the Line_A and Line_B. This still complies to the standard and is a cost effective solution. We recommend that application engineers do not require the galvanic insulation between the MVB line A and B. By this way, exchangeability with all products is kept and the cost figures may become optimised for high volume projects.
Note: The classic ESD- version without galvanic insulation (see below) is intended for use within small areas (e.g. within a rack) or with external interfaces to the OGF.



Insulation EMD

The two MVB lines are insulated to the logic <u>and to each other</u>.



Shielding Concept

There are basically two different shielding concepts used in applications.

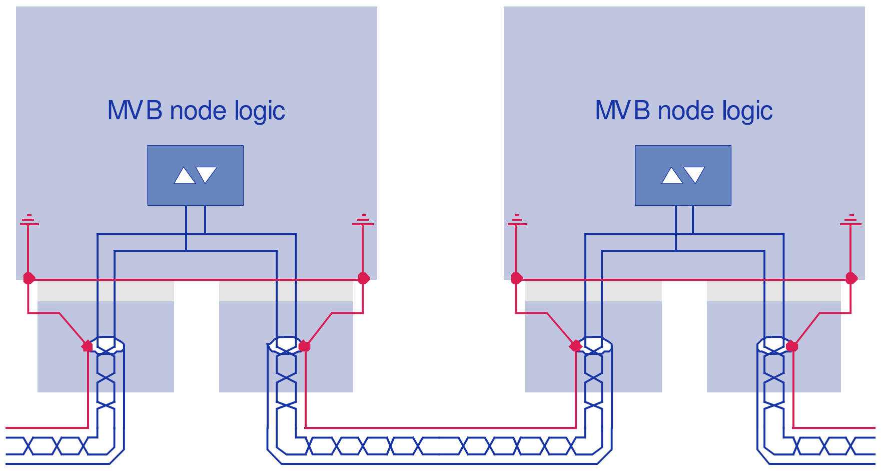
- The most popular concept connects the shield to the cases. This is easy to implement, the shell of the used SUB- D- connectors are directly attached to the device´s housing.
- An alternate concept allows to have potential differences between the housings. This requires to insulate the shield from the node´s housing – which is sometimes difficult to achieve.

The shielding concept has an influence to the way, how the nodes must be built into housings; i.e. the device integration must adapt to the shielding concept used for the particular application.

"Shielding on case" Concept

The main properties of this concept are:

- The cable shield is connected *to the device housing*.
- The cable shield has *in all nodes* the same connection to the device housing.

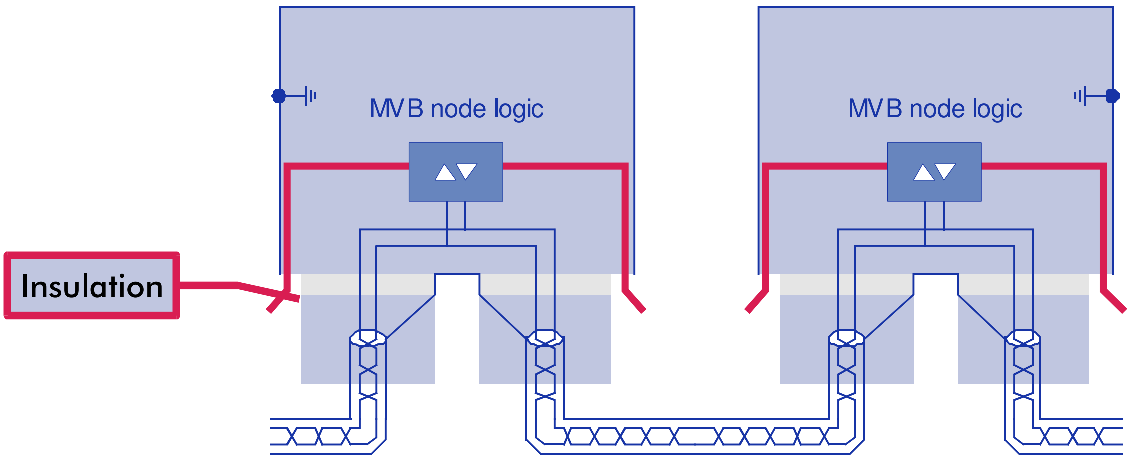


This has as a consequence, that all housings are connected together. Within the vehicle concept it must be ensured, that there are no ground potential differences that may harm the cable shield or the connectors.

The connection between the cable shield and the device housing is done via the cable connector housing and the fixing screw / cable lock (as prescribed by the TCN standard).

"Insulated Shielding" Concept

This concept separates the cable shield from the device housing:



This concept has the benefit, that

- there is no ground potential difference problem any more (assumed, the ground potential differences are below the insulation withstand voltage within the MVB nodes).

This concept has the disadvantage, that

- the SUB-D- connectors must be mounted insulated with respect to the device housing and
- there must be means to connect the cable shield well- defined in one point to ground.

Wires & Redundancy

Introduction to wire redundancy

The railway market asks for solutions with a high availability; a single wire break must not stop the train.

Usual field bus approaches use just the minimum of electrical conductors required for their function. If only one conductor break occurs, communication is stopped, which in turn will stop the train operation. This is not accepted by the train operators!

Being the most probable failure model, the break of a conductor was focused on: this could be the break of a real wire, a contact or any attachment technique (crimp, solder, etc.).

In contrast, the loss of a complete cable (with several wires included), was considered to be less probable. A complete cable loss would be e.g. the cut of all wires at the same time.

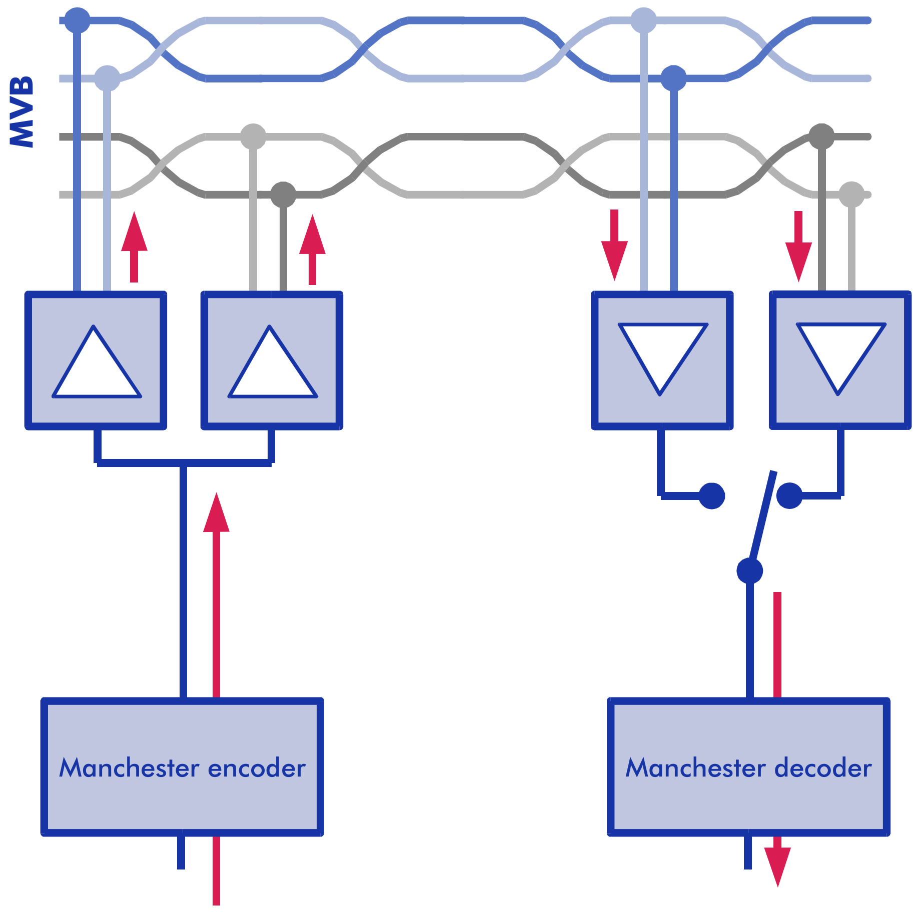
Note: this consideration is valid for the MVB with its static installation within one vehicle. In contrast, for the WTB Train Bus, the cut of a complete cable is considered to be of statistical relevance; there we have redundant cables, not just redundant wires.

As a conclusion, it was decided for the IEC61375 standard to operate the MVB with redundant wires and accept, that the redundant wires are routed within one cable.

The wire redundancy is available for all physical layers, in principle: ESD, EMD, and OGF. Due to the specific properties of the OGF, optical wire redundancy was almost never used in the market. The reason is, that the OGF segment is not a real "bus", but more a point- to- point connection. When compared to the electrical layers, a single point of failure has less influence to the failure of the complete system.

Wire redundancy in general is optional from the point of the standard. In principle, highly available control systems may also be built with non-redundant wires: in this case, e.g. two segments of non-redundant MVB may be used (see page 50 "double segment topology). However, the majority of end customers like to have wire redundancy.

General working principle



- When transmitting data, every node sends out data on **both** Line_A and Line_B.
- When receiveing data, every node is able to receive data **either** on Line_A **or** Line_B.

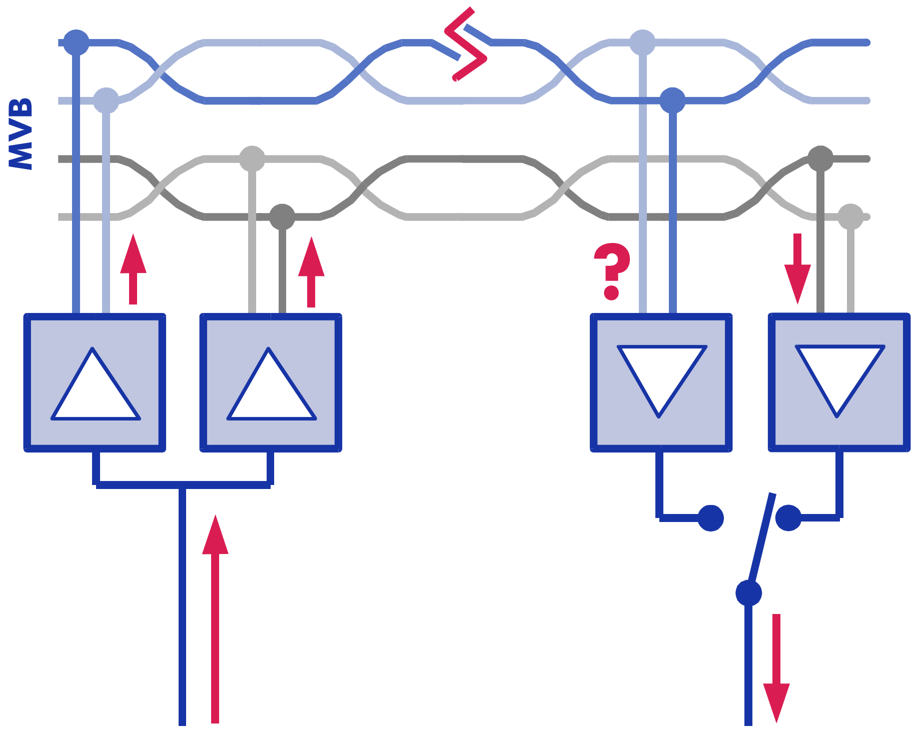
By this way, either Line_A or Line_B may be inoperable. The other line is still able to carry the full amount of communication over the MVB.

The process of "select the line which is to be taken" is called "line switchover".

Any failures must be reported, even if the communication is still working over the redundant line. This enables a repair before the second line may also fail.

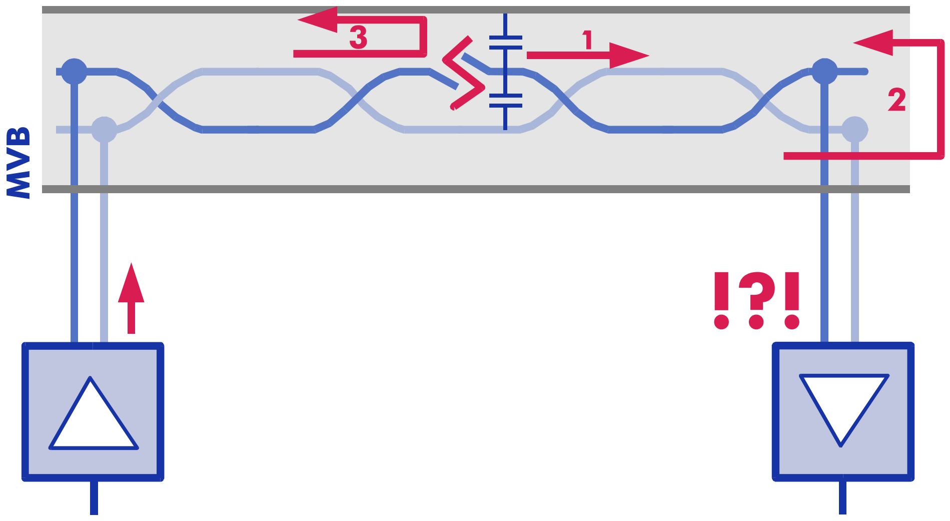
Failure model

In the early days of standardisation, the failure model was pretty simple: If a break of a wire occurs, then communication passing over this failure location is not possible.



The idea was, that a receiver will not receive anything after a wire break. “Nothing” is easily detected; and the second line, receiving “something”, gets priority very easily.

Obviously, this point of view is a little bit too simple. In practical life, more or less the opposite is true: when a single break of a wire occurs, then some devices can communicate over the failure point, others cannot:



The reason for this behaviour is mainly the fact, that the second wire of the twisted pair of wires is still intact and generates a signal on the far side, which is very difficult to keep under "theoretical control".

- **Effect 1** is a capacitive voltage divider from the intact wire to the shield. This induces a signal waveform on the defective wire, travelling to the receiving nodes. This signal is usually big enough to be recognised.
- **Effect 2** lets the signal waveform on the intact wire run through the termination resistor on the far end, attenuate it there, and come back to the break: There it is reflected at an "open end". This yields a distorted signal, which may confuse the receiver.
  Interesting: If the cable length on the right side of the failure is rather small, the termination resistor shifts the properties of the capacitive divider (Effect 1) a little bit, but the frames are still readable.
- **Effect 3** is a simple reflection at the "open end". This may cause other receivers to fail (on the left side of the failure, not shown here in the picture).

Will the receiver on the right side understand the original waveform or not? Sometimes yes, sometimes no. The situation is mainly dominated by the geographical situation of all involved nodes and the signal reflections at the failure point.

To keep control, one would have to calculate the communication ability for all node- to- node combinations. This relates not only to the communication through the point of failure, but also between the nodes on the same side.

Since this kind and amount of work is not possible in practical life, obviously nobody makes use of this theoretically deterministic method.

Line switchover

Unfortunately, several details of the MVB specification have been defined following the early "simple failure model":

The Line A-B switchover is defined in a way, that the receiver recognises either Line A or Line B, but not both at the same time. It is not possible to dynamically switch to the other line during frame reception. This way, validly present data may be lost.

The receiver is supposed to detect various general states of a line: "Weird" waveforms from a cable break should be rejected in order to identify the cable break. "Idle" and "Collisions" must be distinguished from each other in order to allow the recommended event arbitration follow the correct sequence. "Valid frames" must be recognised in presence of noise. It is quite a challenge to fulfill all these classifications reliably. The recognition is always a statistical aproach, it cannot be deterministic.

Personal note from the author

Did **you** fully undestand the specification in the standard about line switchover? **I did not**. I tried of course, and my feeling of having it understood was not bad. But later, when I had funny effects from my real world measurements, I failed to explain it from the specification architecture. The result was, that I decided to solve my problem in a different way, not by using information from the line switching. This would have costed several additional days of work, and time is what we all do not have...

MVBC01 and "line redundancy"

The MVBC01 was the first MVB controller chip available for the MVB market. It was designed before the finalisation of the IEC standard, and therefore has some "differences" to the standard, especially in relationship with "line redundancy":

It has a different "line switchover" strategy. There have been cases reported, where certain failures on one line could also impair communication on the other line: This is an absolute "must not" for high availability control systems! If you need to debug MVBC01- systems, anyway: Make sure, you use the MVBC01 datasheet, not the IEC standard.

When the MVBC is working as BA, and an external Line is short circuited (the MVBC01 cannot read back what it sends out), it may happen that the MVBC01 stops operation completely. This hang-up may be recognised by software and the chip may be restarted, but depending on your software latency time, this may leave the MVB in an idle state for more than 1.3ms. This may start other, redundant BAs, which later on yields collisions on the bus. Therefore: when doing debugging with an MVBC01 as BA, be sure to have only one of them or be prepared to get funny phenomena.

When being the BA, the information of the RLD, LAT- bits has to be moved by software from the received ports to the own Device Status Report. This procedure, if it would have to be perfect, would require software latency times in the range of down to a few microseconds, which is not possible for usual CPUs.

Implementing a better failure model

The early implementations according to the “simple failure model” have led to quite confusing situations during the debugging of real vehicles. As a consequence, the next generation of MVB controller chips implemented a different behaviour. The result was a much better behaviour in practical life, but a partial non- compliance to the standard.

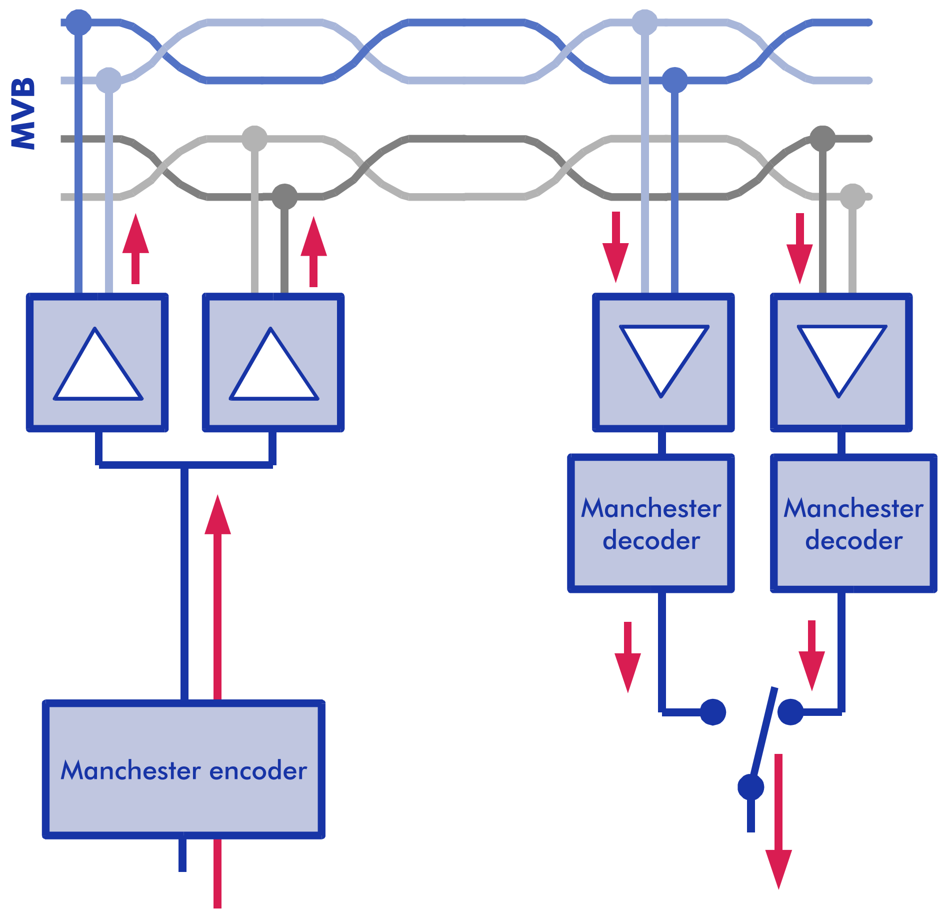
The original architecture was a node containing exactly one decoder for receiving data. This decoder could be switched to watch either the Line_A, or Line_B. The specifications for how to switch between the two lines are quite complicated. Quite a lot of trouble arises from the fact, that when the decoder watches Line_A, then incoming data on Line_B will not be recognised. The decoder will usually switch to Line_B afterwards, which is too late for this frame.

This situation is in particular tricky, when the engineers try to locate a failure by removing nodes from the bus. This is an adequate technique for a lot of other situations. On MVB, the missing of some process data sources will completely change the dynamic behaviour on the bus and thus the line switching. The result was often enough perfect confusion...

Therefore, the new failure model must allow for each received frame a new decision between Line_A and Line_B. By this way, the result seen on the node (successful reception or not) can be used as an independent input for debugging; i.e. it is not dependent from earlier activities.

Since the decision for Line_A or Line_B is taken at a quite late point in time, this dynamic redundancy switching has also a positive effect on noise robustness (beyond the specs from the standard).

The status bits LAT “Line A trusted” and RLD “Redundant Line disturbed” are still used to mark the status of the node. However, due to their tight relationsship to the old line switching technique, the underlying definition has changed.
These bits are distributed on MVB by the “Device Status Requests” from time to time; with some few or up to thousands of other frames in between. By the old way, it was impossible to draw reasonable conclusions from them: which one of the thousand activities was the failing one?



The result from this situation was a new design of the decoder and the line switching. Now, there are basically **two decoders** (instead of only one), and they watch both lines at the same time. By this way, a valid telegram on either line will be recognised in any case. The decision, which line is “the prettiest one”, is taken dynamically and does not depend on any state from before of the telegram under question.

Now, LAT and RLD are the result from the reception of the “Device Status Request” Master Frame for the Device Status Report being sent, including RLD and LAT.

This way, there is a clear geographic relationship of the result: the location of the BA and the scanned device is known, and there is no hidden info included.

Since the BA scans all devices, the list of all DSWs gives an overview about the general situation. This overview is available on all nodes (if they are able to receive DSWs at all).

Personal note from the author

In principle, it would make sense to propose these changes to the IEC working group in order to get the situation simplified for new releases. However, this would make a lot of “political noise”: it may happen, that existing systems loose their “standard compliance”. Due to the fact, that (at least to my knowledge, comments invited) there is no perfect complying solution on the market, I will relax and see what comes…

How to find cable faults by wire redundancy information

The "wire redundancy" as described in the previous section has the task to ensure successful communication even if one line fails.

There is a second task to be performed: In case a failure exists, you should find it and fix it. It makes sense to use the information gained from "wire redundancy handling" and generate from it a statement about the health of the two lines.

Cable diagnostic according to IEC61375

Following the rules of the line switching, a failure of one line will set the RLD bit in the Device Status Report. The LAT bit indicates which line is still "trusted", therefore the other must be the failing one.

In addition, two error counters accumulate errors; seperately for both lines.

Both facts together allow some conclusions:

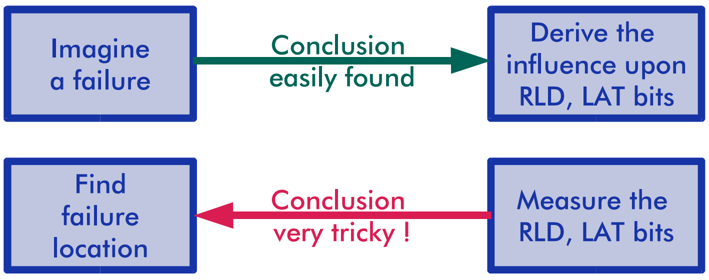
- If there is a failure, it will be detected by RLD=1.
- If there is a failure, it will be detected by all devices in a similar way. If only one node reports errors, then this node has probably lost one of its line receivers.

LAT	RLD	Global statement
0	0	OK. There is no difference between the two combinations!
1	0	
0	1	Failure Line_A
1	1	Failure Line_B

The two combinations for OK will toggle from time to time according to the line switching rules; but this does not include any new information.

Unfortunately, some interesting conclusions are hard to find:

- If a single node is failing (one line transmitter defect), there is no way to identify it, e.g. by its device address.
- The absolute height of the error counters does not really give some information about the kind of failure: the definition is made in a way, that severe wiring errors have the same effect as one single failing node.
  IEC61375 is a little bit fuzzy in the definition, which of the two counters is to be incremented. Obviously, incrementing is intended only in case, the respective line is the trusted one. Since the line switching will move the trusted line away from the found failure, the failure will not generate additional counting. Thus, there is no additional information beyond the "RLD" bit.
  Intensive counting is possible only with more than one single failure (which is not probable) or with collisions found. The main reason for collisions is a wrong address configuration of a node, which should be excluded by type testing beforehand.



Cable diagnostics, next step

In order to give more information to the field service people, another approach was choosen for newer MVB controller implementations (e.g. Duagon products; see also "error counters for duagon products" on page 105):

- The RLD and LAT bits are generated during reception of the Device Status Request Master Frame and transmitted during the Device Status Report Slave Frame immediately following it.
- During reception of both DSR and PD slave frames, an information about both lines is stored individually. For the duagon products, the two bits are called SINK_A and SINK_B.
- The error counters are incremented during the application controlled read of DSR or PD sink ports, using the mentioned SINK_A and SINK_B bits.
  This has the effect, that during regular applications the error counter will measure faults relevant to the own device, only. Nodes, which are not involved in any trouble, will not report failures due to error counters.

LAT	RLD	The DSR-MF was...
0	0	OK. There is no difference between the two combinations!
1	0	
0	1	...failed on A
1	1	...failed on B

Usually, there is no toggling between the two OK- combinations any more.

SINK		PD or DSR slave frame was ...
A	B	
0	0	...not received since last read
1	0	...not received on B
0	1	...not received on A
1	1	...successfully received over both lines.

The conclusions to be taken from this:

- Since the RLD and LAT bits are derived from the transmission of exactly one Master Frame, the information contained is related to a clear geographic situation, since the location of the BA and the node in question is known.
- This is also true for the nodes watching the DSR slave frame. They may use the SINK_A- and SINK_B- bit and there is a defined geographic situation, too: Between the node sending out the device status report and the watching node.

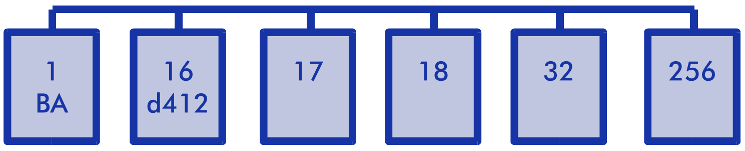
Based on these facts, diagnostic software may take various different conclusions:

- **Consumer oriented** (recommended for all nodes)
  A simple check of the error counters allows to state, whether or not the node is directly involved in trouble for the incoming data.
- **Global focus 1** (a must for at least one node)
  Combine all DSRs together: If at least one RLD bit is set, or if the error counter>0, then there is a failure on a global perspective.
  Note: reading the RLD bit will have the side effect, that any trouble found by the SINK_A/SINK_B information is accumulated in the error counters.
  Check also, whether the number of intended devices equals the number of validly found DSRs.
- **Cable diagnostics 1** (information derived from "Global focus 1")
  Diagnostic software may perform an automatic identification of failing nodes via the device address.
- **Global focus 2** (recommended for at least one node)
  The measures taken in "Global focus 1" may take a while due to the scan rate of DSRs on MVB. The scan rate depends on the BA's scan strategy and may take up to 32s (following the IEC recommendations).
  A faster failure detection (less than 1s) relies on PD sink ports:
  - Scan the error counters in a faster sequence than for "Global focus 1". This will find errors quite fast.
  - The focus may be set to the PD ports used in the local application anyways (this is achieved automatically) or the focus may be extended to all PD ports being defined vehicle wide (simply add dummy reads of these extra ports).
- **Cable diagnostics 2** (information derived from "Global focus 2")
  In order to achieve the same speed for the identification of single failing devices, a mapping from PD ports to Devices may be implemented.
- **Cable diagnostics 3** (may be implemented in software running on a node, running on a seperate PC with a diagnostic tool, or even by hand)
  Collecting all the RLD, LAT, SINK_A and SINK_B information from all nodes, a pattern can be derived for "critical communication partners", which enables the user to develop probable failure models for the specific case.
- **Cable diagnostics 4** (implemented in software running on a node or on a seperate PC with a diagnostic tool)
  Elaborate diagnostic software may even focus the attention on one particular transmission path (repetetively read a specific PD port, only; and nothing else: Watch the error counter). By the height of the error counters you will get a fast reaction upon mechanical influences. This way you will find loose contacts or other dynamic failures.

Practical example: How to find cable faults by DSRs

The procedure proposed here is basically valid for up- to- date duagon products after 2002. Be aware, that other nodes (especially with old MVB controller chips) may not fully support this kind of taking conclusions.

For this example, the bus structure was as follows:



First step: Collect all Device Status Reports from all existing devices.

You may get this info from every node, which is able to receive DSRs at all. This could be the Bus Administrator or any other node. This procedure does not need neccessarily the correct BA list for your application; it requires simply the DSR scan of an almost "empty" BA list.

Another way is to use a diagnostic tool. The printout below was generated by a "d412 Diagnostic Tool" from duagon; the command was "List all devices found during device scan on MVB":

```
mvbMON>svd -a
number of devices: 5
-------------------------------------------------------------
|              |         |         | C C C C | L R S S E F D S |
| device       | device  | S B G M | S S S S | A L S D R R N E |
| address      | status  | P A W D | 0 1 2 3 | T D D D D C R R |
-------------------------------------------------------------
| 0x001 (   1) | 0x5380  | 0 1 0 1 | 0 0 1 1 | 1 0 0 0 0 0 0 0 |
| 0x010 (  16) | 0x1080  | 0 0 0 1 | 0 0 0 0 | 1 0 0 0 0 0 0 0 |
| 0x012 (  18) | 0x1080  | 0 0 0 1 | 0 0 0 0 | 1 0 0 0 0 0 0 0 |
| 0x020 (  32) | 0x1040  | 0 0 0 1 | 0 0 0 0 | 0 1 0 0 0 0 0 0 |
| 0x100 ( 256) | 0x0042  | 0 0 0 0 | 0 0 0 0 | 0 1 0 0 0 0 1 0 |
-------------------------------------------------------------
mvbMON>
```

Everybody online?

Compare the number of received DSRs with the intended number of available nodes. By this way you may find nodes being off-line.

In this example, obviously one device (address= 17) is missing. A very common reason is that this device is switched off.

After having it switched on, the DSR scan looks like this:

```
mvbMON>svd -a
number of devices: 6
-------------------------------------------------------------
|              |         |         | C C C C | L R S S E F D S |
| device       | device  | S B G M | S S S S | A L S D R R N E |
| address      | status  | P A W D | 0 1 2 3 | T D D D D C R R |
-------------------------------------------------------------
| 0x001 (   1) | 0x5380  | 0 1 0 1 | 0 0 1 1 | 1 0 0 0 0 0 0 0 |
| 0x010 (  16) | 0x1080  | 0 0 0 1 | 0 0 0 0 | 1 0 0 0 0 0 0 0 |
| 0x011 (  17) | 0x1080  | 0 0 0 1 | 0 0 0 0 | 1 0 0 0 0 0 0 0 |
| 0x012 (  18) | 0x1080  | 0 0 0 1 | 0 0 0 0 | 1 0 0 0 0 0 0 0 |
| 0x020 (  32) | 0x1040  | 0 0 0 1 | 0 0 0 0 | 0 1 0 0 0 0 0 0 |
| 0x100 ( 256) | 0x0042  | 0 0 0 0 | 0 0 0 0 | 0 1 0 0 0 0 1 0 |
-------------------------------------------------------------
mvbMON>
```

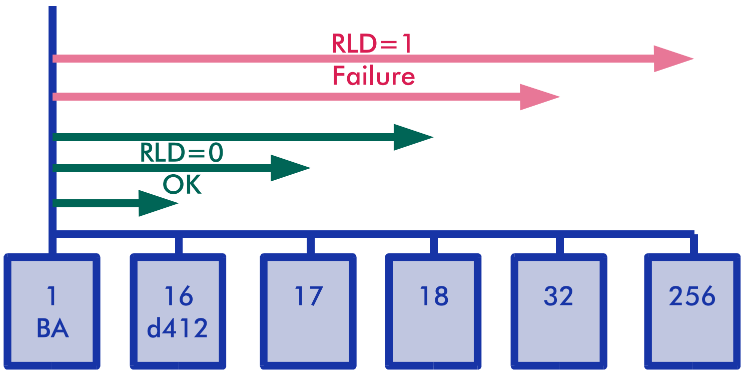
Now, all six devices from the bus structure can be found in the DSR list, i.e. no device is offline.

Are any RLDs set?

Check, whether all DSRs show perfect results: Good devices should deliver valid DSRs, RLD=0; LAT is not important, it may be 0 or 1.

Example
Some of the existing devices have RLD set: try to locate the fault. Make a drawing with the BA and all other devices on the bus with the correct sequence as they really have on the bus. This may give you a closer impression about where the fault must be located: always consider the path between the BA and the device in question.

LAT	RLD	
0	0	OK.
1	0	
0	1	Failure Line_A
1	1	Failure Line_B

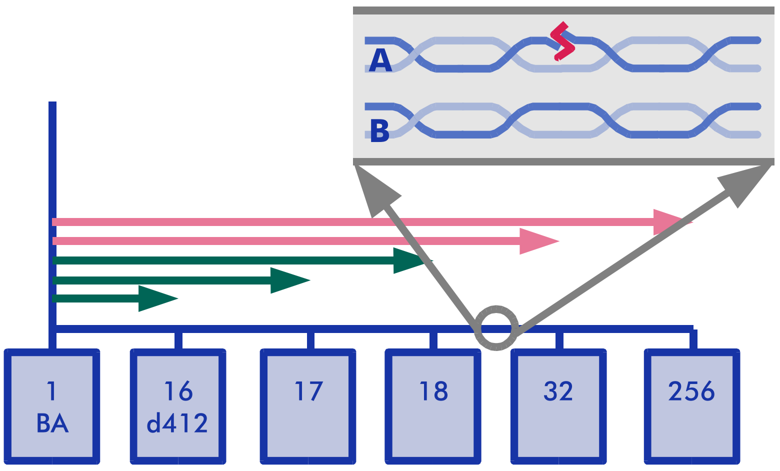


Not the case in our example and not very probable, but still possible:

- Sometimes it is possible to contact nodes passing through a faulty location.
- It may also be, that the cabling between the BA and a failing node is ok, but the communication is still not possible (effect of reflections).

Repair the fault !

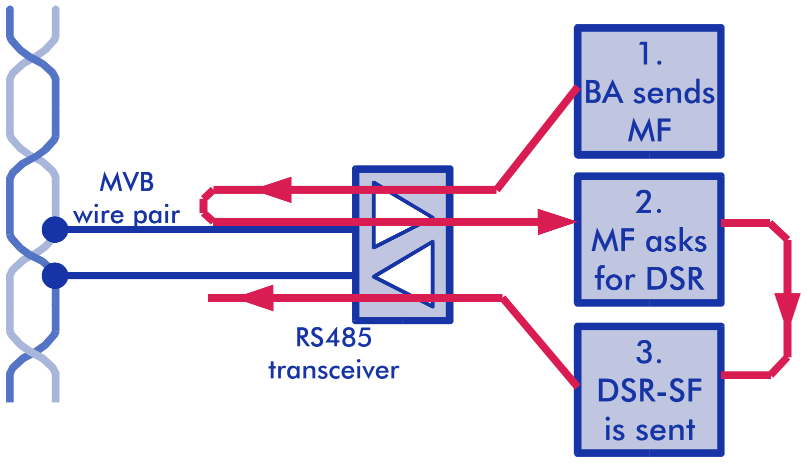
You now may replace MVB cables and/ or short nodes (take them away from the MVB by shorting the MVB- cables just in front of them). Watch, whether the failure disappears and take your conclusions.



Bus Administrators and their own DSR

Sometimes, bus administrators perform a little bit unexpected with regard to their own DSR . In order to understand this, lets have a look into the details of how a BA issues a DSR request to itself and how it replies with a DSR report.

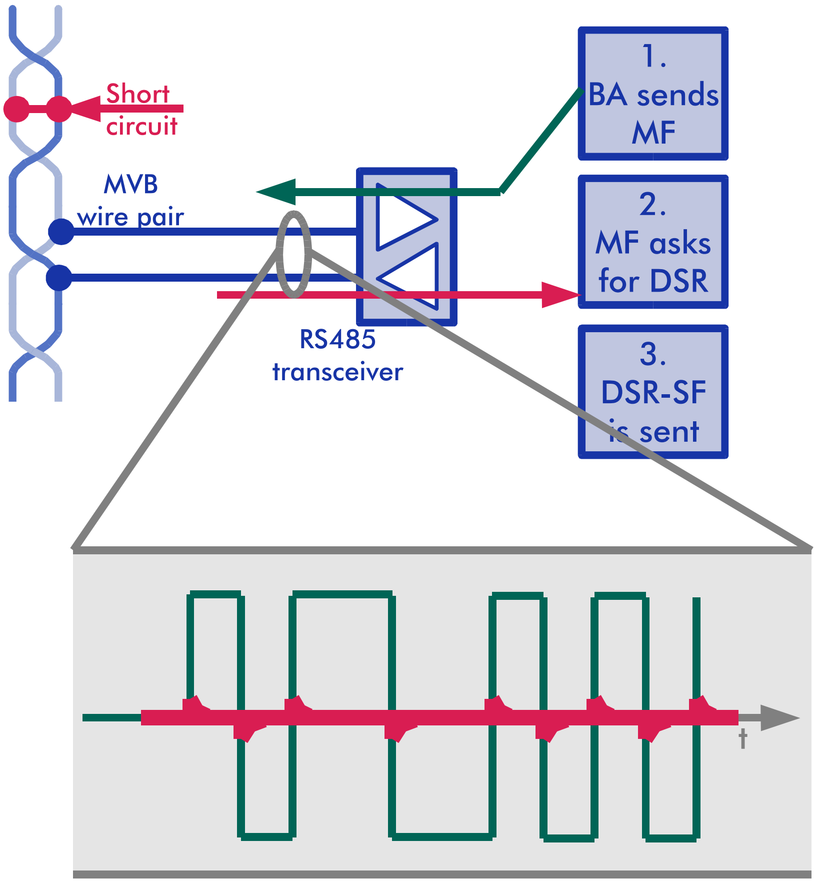
Successful DSR sequence



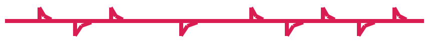
- In a first step, the BA issues a master frame, containing a DSR request with its own address. This frame is placed on the MVB by the RS485 drivers.
- In a second step, the MF, which was sent from the BA node, is received from the MVB lines through the RS485 receiver. The node recognises that it is addressed with the own device address and has to reply.
  Note: There is no shortcut within the BA node, which routes the MF "silently" to the DSR response logic.
- Now the DSR response logic prepares the DSR report, which includes generation of the LAT and RLD bits. The completed report is then distributed over the MVB lines.

What happens with failures

As long as there is no failure on the wires, the MF sent out has a reasonable waveform on MVB (green shape):



In case there is a short circuit on the outside, this waveform will be distorted. Close to the short, the voltage will be zero (by definition), which would deliver a flat zero line. However, back at the RS485 transceiver chip, a small remainder of the original waveform can be seen (red shape).



In addition to a short circuit, also broken wires may introduce distorted wave-forms. For an example, see page 98. However it is not very probable, that these interferences will make the transmission of one device unreadable by the same device.

Will the red waveform be recognised?

The short spikes of the red waveform may exceed the RS485- receivers internal hysteresis or not, depending on the involved physical parameters like line impedances, speed and strength of driver chip, absolute level of receiver threshold and the like. **Therefore the MF may be recognised or not**.

This is where the surprise starts:

- If the MF is recognised, then the DSR response logic will prepare an OK (RLD=0).
  The expected behaviour for a short circuit would be to have **all devices** report an error, but the BA is excluded because he is still able to communicate with himself.
- If the MF on the shorted line is not recognised, and usually the second line successfully receives the MF, then the DSR response logic will prepare a failure report (with RLD=1), as expected.
  This failure reports is successfully distributed over the MVB (the second line has no short-circuit!). The BA and all other devices will be able to receive it.

In the unusual situation, when both lines have a short:

- If the MF is recognised on one or both lines, then the own BA- node will send a DSR- SF, which is received on the BA- node.
  All other nodes will not see all this.
- If both lines are not recognised, then also the BA- node will not receive the own DSR.

Conclusion: If you are sitting on a BA- node, never use your own DSR to take conclusions about cable failures.

Note: The older MVBC01 has some additional confusing properties which have an influence on LAT/RLD generation; see page 61.

How to find cable faults by other checks

The previously described procedure to find cable faults "by redundancy information" assumes that at least certain parts of the MVB cables are working correct.

Sometimes, an MVB installation does not work at all or the redundancy information does not lead to any idea about the fault. In these cases, it is time to go back to the basics and do some checks with the cable.

This is a short (and uncomplete) list of simple practical measurements to be taken:

All single cables checked?

See also the general cable structure on page 82.

- Each cable must show a connection between the two sides on pin 1,2,4,5 and Shield.
- The pins 6 and 7 must have a connection on ESD (potential equalisation lines); they must not have a connection on EMD (otherwise the terminators are disturbed by extra wires).
- The pins 3, 8, 9 must not have a connection.

Measure the bias voltage with a multimeter (ESD only)

Switch the equipment on (at least the two end nodes) and measure the bias voltage.

- It must be 0.75V between pins 1,2 with being pin 2 being the more positive one.
- For redundant wires: Same for pins 4,5 with 5 being the more positive one.

Measure the two termination resistors with a multimeter (ESD only)

Switch off all equipment and measure the DC resistance between pins 1 and 2 (4 and 5). As result, you must see the approximate 120 Ohms of the terminator on each side. You may have to add some extra for the wire copper resistance (realistic order of magnitude up to 20 Ohms).

Measure the two termination resistors with a multimeter (EMD only)

If you measure DC resistance of an EMD wire pair, you will most probably see almost zero Ohms. This is due to the transformers, which have at DC just the copper resistance of the primary winding. This does not give very much information.

You may pull out the cables at all nodes and connect the cables in a sequence. Then, the resistance must be an "open" line, since there is no terminator at the end.

Measure the cable length with a multimeter

Sometimes, the cable length is not known in a project because of last-minute changes, due to e.g. routing in ducts. However, with the length being too high, a limit may be exceeded.

Switch off all equipment. On one end, connect all wires together and measure the capacitance against shield. Dependent on the cable type, you will find a capacity of very roughly 75pF/m. You should calibrate your measurement with a piece of cable with known length (e.g. 10m).

See also "cable characteristics" on page 83 and "cable attenuation calculation" on page 84.

Measure the cable length with TDR equipment

Cable length measurement is much easier to measure with TDR equipment (which is by far more costly).

Please see also the chapters about “reflections” on page 88 and about “cable attenuation calculation” on page 84.

Check the cable attenuation

By using a wrong cable type, the signal levels may be attenuated in an excessive way. This has more or less the same effect as using a too long cable.

See also “cable attenuation calculation” on page 84.

Check wire allocation

During cable manufacturing, sometimes the allocation of wires to pairs is made in the wrong way. This will lead to hard- to- detect effects like crosscoupling between the wire pairs. If you have simple tools, only: Open up the connector housing and check the allocation of wires compared to the cable data sheet.

Please note, that this type of manufacturing fault may be hidden for a long time on a running vegicle: The high crosscoupling between the lines may not appear as a problem during normal operation. However, the way to find an extra fault, e.g. a single wire break, may become extremely tricky.

See also “wire allocation” on page 84.

All nodes EMD or ESD?

Of course, all nodes must comply to the same physical layer.

However, under certain conditions, nodes of different type may exchange data with each other. Usually, it is a “one way”: ESD devices understand EMD telegrams (they check for the short end delimiter, only). EMD devices will not understand ESD telegrams, since the NH part of the end delimiter is missing.

If you have such one-way phenomena, check the EMD/ESD- type of all nodes.

All nodes same redundancy level?

It is not allowed to add a non-redundant node to an MVB- segment, which is supposed to run “redundant”. However, there are MVB products on the market, which support non-redundant configurations, only.

- Duagon products would be tolerant to such a situation; they would simply point to the node which is not able to transmit on both lines (see the previous paragraphs). There is no loss of other functionality: all other nodes will be recognised as redundant and the dynamic line switching handles subsequent faults on either line.
- Nodes complying to the standard with respect to line switching will encounter a general “Line B fault” and seriously loose data by any additional failure.

Since a vehicle contains always a mixture of different MVB controller manufacturers, we highly recommend to keep the principle “all node on same redundancy level”.

Note: adding a redundant node to a non-redundant MVB- segment is usually no problem: no other node will care about this fact; and on the added node you can configure the node by software in a way, that it behaves like a non-redundant device.

Comparison EMD – ESD+

The decision for one of the physical layers has an influence on the hardware and the MVB controller chip; but for all upper layers, it does not make any difference. From this point, the application engineer is free to decide between EMD and ESD+ as long as he is able to purchase all equipment of the selected type. However, there are some small, but sometimes important issues, that make a difference between the two physical layer choices.

Please find below a table with the conclusions. The remarks and explanations for the detailed items are in the following paragraphs.

Issue	EMD	ESD+
Max.bus length	200m+ *)	200m- *)
Robustness against noise	Yes *)	Yes *)
End delimiter	NL + NH symbol	NL or less *)
Cable wires (redundant)	1 pair (2 pairs)	1 pair + 1 GND *) +(2 pairs + 2 GND)

* = see comments below.

Bus length EMD

The requirement from the IEC standard is to have at maximum 8dB attenuation between any two nodes: 1.5V minimum driver amplitude minus 8dB delivers approx. 0.6V. Since the sensitivity of usual receiver chips is much better (200mV), there is plenty of safety margin. See also measurements for "EMD full bus load" on page 100.

Bus length ESD+

Due to the biasing of the line with 0.75V, the attenuated driver signal is going to disappear at the far end of the line. Thus, the required 8dB seems not to be ensured in all cases. See also measurements for "ESD full bus load" on page 99.

Noise rejection

The IEC61375 paragraph 3.2.5.10.6 specifies the "noise rejection" of an EMD receiver. This feature is type tested on the more modern MVB controller chips (e.g. duagon). During the early times of EMD, especially the common mode noise spec was not fulfilled.

There is no paragraph in the IEC61375 about ESD noise rejection. This issue is believed to be covered by the line biasing (the 0.75V on an idle line). By this way, any noise must be higher than this bias in order to become a detectable noise signal.
A drawback of this concept is the fact, that it requires both end nodes to be validly powered (see measurements on page 102). If one end node is switched off, the terminator will not deliver the bias voltage any more, i.e. the bias voltage will drop to 50%. If also the second end node is powered down, then there is no bias voltage any more and the receiver is now open for any noise being higher than the 50mV receiver hysteresis (which is the same situation as for the EMD).
However, this is a problem for old MVB controller designs, only: e.g. MVBC01.
This design relies on the fact, that the noise is filtered out before the digital logic.
Newer designs, as for example the duagon products, apply the same digital filter-

ing algorithms in internal logic for EMD and ESD. By this way, they do not require a bias voltage any more.
Conclusion: if it is required from the application to work reliably also with un-powered end nodes, be sure to have all nodes equipped with digital filter algo-rithms that are able to fulfill the EMD- noise- rejection- type tests.

ESD end delimiter

The specification of the ESD end delimiter leaves room for trouble in practical life. See also the chapter “MVBC01 and ESD end delimiter” on page 93.

Cables

The cable specification for ESD requires to have additional “potential equalisation lines, which makes the cables slightly more expensive and less handy. See also pages 82, 101 for “potential equalisation lines” and page 77 for “terminator sup-ply”.

Electrical Short Distance Bus ESD, ESD+

Introduction to ESD

The ESD physical layer equals basically RS485, with some minor differences. When run with the 1.5MBit/sec, the length of one segment is up to 20m (including a reasonable safety margin for noise, following RS485).By this way, the name "short distance" was introduced.

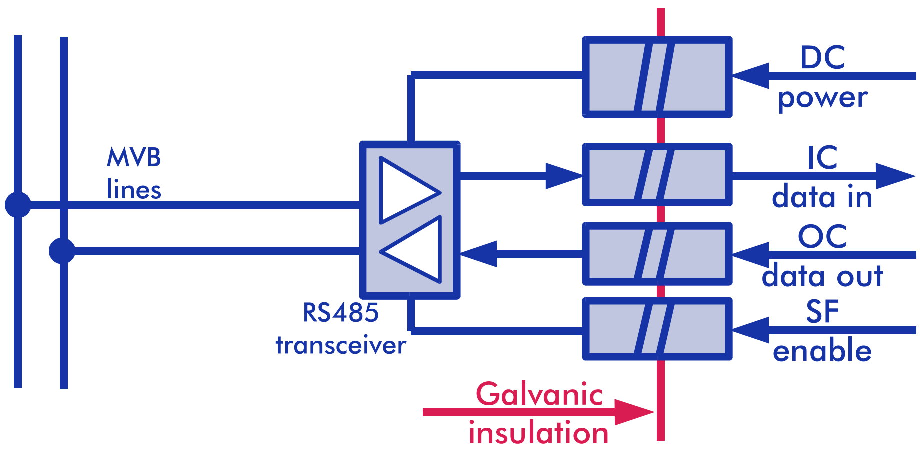
Historically, this was the very first physical layer proposal before the IEC standardisation was finalised; the original idea was to use it in small areas, only: e.g. within one rack of electronic equipment or within the driver´s cab. As a complement, the OGF was intended to be used for longer distances.

The specification for the ESD in the IEC61375 essentially reflects this early phase of MVB development. In a later phase, the ESD was enhanced by introducing galvanic insulations for all nodes. This helped to reduce the common mode noise on the line by breaking up ground loops. Since the definition of an idle bias voltage reduced differential noise compared to RS485, the practical use was extended to applications of up to 200m of segment length. This is the same application range as for the EMD!

Today, the ESD is available in this enhanced version, only. It is very often called "ESD+", and it reflects the IEC standard with additional galvanic insulation of the node transceivers. Due to this fact of the market situation, the "ESD+" is often understood to be the generic "ESD".

The move from ESD to ESD+ (and EMD) had also another effect: the OGF, previously intended for the long distance (up to 2000m) is more or less disappeared from the market. Even though the optical concept has some very nice technical features, there is almost no market volume for it.

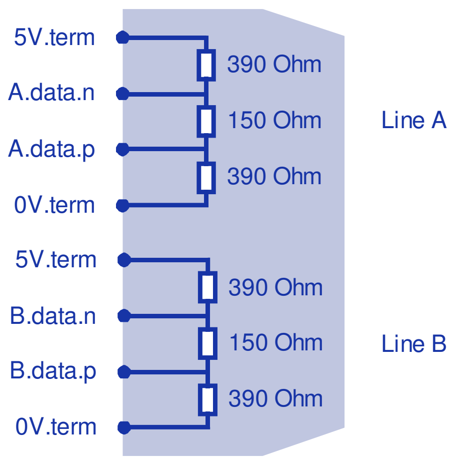
Typical Node Structure ESD+



The picture shows the basic elements of a typical ESD+ - node:

- The RS485 transceiver chip is directly coupled to the MVB lines.
- All signals, as well as electrical power, have to be delivered passing a galvanic insulation. This insulation is not required for ESD, it essentially makes the "+" within "ESD+".

Terminators ESD+



When the MVB line is linearly built up with cables between the various nodes, two connectors, each at the very end of the line, remain open.

The two terminators are mounted on these connectors and delivers the correct line termination.
The resistors inside will bias an „idle" line when not driven by a node transmitter. The effective line termination matches the recommended cable impedance of 120 Ohms.

Pin definition ESD+

SUB-D Pin #	Pin shortcut	Input / Output as seen from the node	Description
1	A.data.P	bidirectional	non- inverted MVB bus line, with RS485- level
2	A.data.N		inverted MVB bus line, with RS485- level
3	NC	-	Not connected
4	B.data.P	bidirectional	non- inverted MVB bus line, with RS485- level
5	B.data.N		inverted MVB bus line, with RS485- level
6	A.0V.term	Power output	Terminator supply: Power from node to external terminator
8	A.5V.term		
7	B.0V.term		
9	B.5V.term		
shell	Shield	-	Connection to shield resp. housing.

According to the TCN standard, pin 3 may optionally be used for a "TxE signal". This signal is intended for controlling bus couplers (for example interface from ESD+ to EMD). However, this function has not found introduction to the market.

Within a node, all signals and the shield of the two connectors are connected to each other.

Terminator Supply ESD+

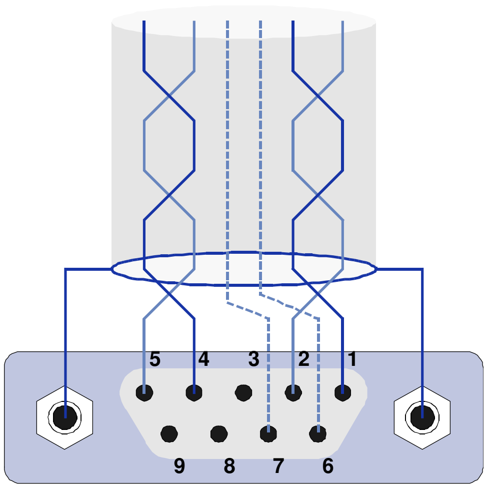
Item	Value	Unit	Remarks
Output voltage	5	V	+/-5% tolerance
Short circuit current limitation	300	mA	See note
Maximum output current	70	mA	when terminator driven „active“

The main intention for the short circuit current limitation is to avoid any fire hazard in case due to overheated wires.

Note: The Terminator supply pins 8 and 9 must not be connected within a cable. This may lead to back- powering the node´s local supply from another node and may impair proper functionality or even damage devices!

Note: There are cases, where the output current specification is not sufficient. Some OGF interfaces (Optical Glass Fiber) require a supply current of approx. 100mA. For this application, we recommend to bridge the normal galvanic insulation and directly use the Vcc power line from the local logic power supply (ESD instead of ESD+). This leads to a higher possible current (close to what the power supply can deliver), but the “short circuit current limitation” spec (see IEC61375) is not kept any more.

MVB Cable Attachment ESD+



The figure shows the recommended cable attachment.

The dotted lines are the “potential equalisation lines”. They have the intention to connect all MVB nodes together.

Please note: Pin 3, 8 and 9 are not connected on the outside. Do not use cables where all pins are connected, this may impair proper functionality or even damage devices!

Electrical Middle Distance Bus EMD

Introduction to EMD

The EMD specification was derived from the early ESD definition. The intention was to introduce a galvanic insulation of the node electronics by a transformer.

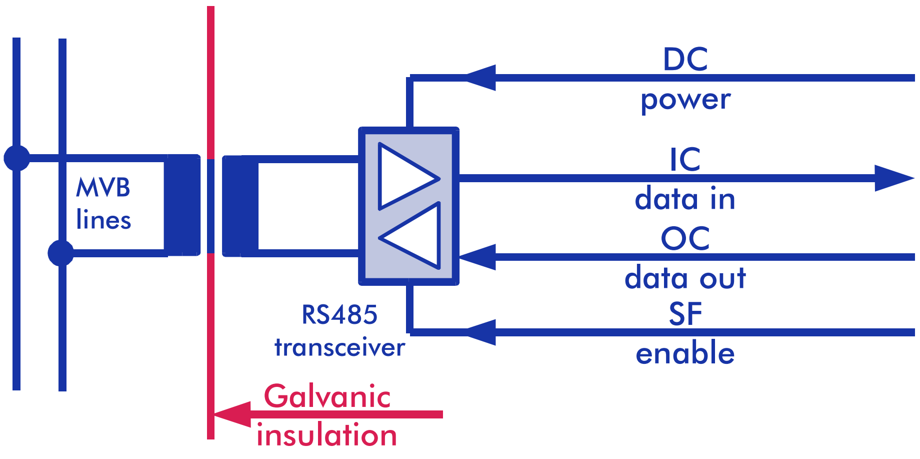
During standardisation, there was a political fight about the implementation of the galvanic insulation: by transformer (EMD) or by DC/DC-converter and opto couplers (ESD+). In order to resolve this conflict, the standard was finally written in a way, that it will allow both parties to use their preferred solution. This is why we have today ESD+ and EMD in the same application area (remark: usually, it is the intention during standardisation to prefer one technical solution).

From the technical situation, the EMD is supposed to be the cheaper solution (a transformer is usually cheaper than a DC/DC- converter plus opto couplers).

However, what makes EMD more tricky to implement: the EMD cannot make use of a bias voltage on the line. On the ESD+, the bias voltage delivers an implicit noise reduction, being understood to work this good, that there is no need for additional specification. Unfortunately, transformers do not conduct bias voltages (DC!); i.e. the noise robustness of the EMD had to be proven by other means. This was the reason, why in the standard there are detailed specifications about noise behaviour for the EMD, but not for the ESD.

During the early days of EMD, it was difficult to fully comply to the standard, especially to the "common mode noise" specification. This issue was resolved by a new generation of controller chips several years ago. Today, the EMD seems to be the more attractive solution, even though both ESD+ and EMD are found to be strong in the market.

Typical Node Structure EMD



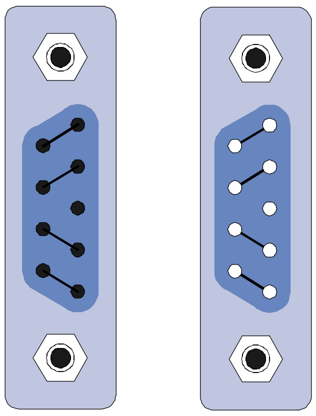
The picture shows the basic elements of a typical EMD- node:

- The RS485 transceiver chip is coupled to the MVB lines via a transformer.
- All signals are directly connected to the MVB controller logic; the electrical power is identical to the power for the local CPU.

Terminators EMD

The termination resistors itself are located inside the node; and they define the effective line termination of 120 Ohms.

The "terminators" for the EMD contain small connection wires from the signal lines to the termination resistors, which are located within the node.



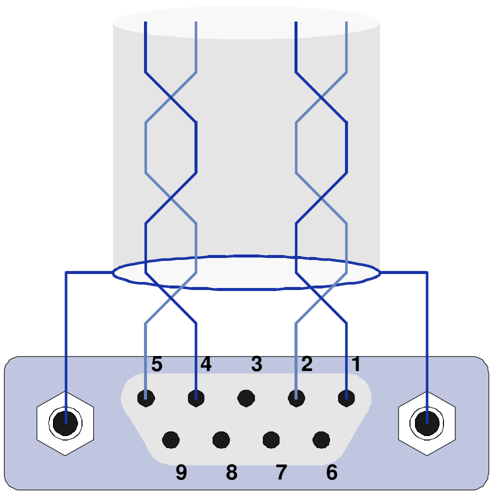
Pin definition EMD

SUB-D Pin #	Pin shortcut	Input / Output as seen from the node	Description
1	A.data.P	bidirectional	non- inverted MVB bus line, with RS485- level
2	A.data.N		inverted MVB bus line, with RS485- level
3	NC	-	Not connected
4	B.data.P	bidirectional	non- inverted MVB bus line, with RS485- level
5	B.data.N		inverted MVB bus line, with RS485- level
6	A.term	Passive resistor (120 Ohm) within node	Termination resistor between the two pins.
8			
7	B.term	Passive resistor (120 Ohm) within node	
9			
shell	Shield	-	Connection to shield resp. housing.

According to the TCN standard, pin 3 may optionally be used for a "TxE signal". This signal is intended for controlling bus couplers (for example interface from ESD+ to EMD). However, this function has not found introduction to the market.

Within a node, all signals and the shield of the two connectors are connected to each other.

MVB Cable Attachment for EMD



The figure shows the recommended cable attachment.

Please note: Pin 3 and 6 to 9 are not connected on the outside. Do not use cables where these pins are connected, this may impair proper functionality or even damage devices!

Cables

The technical properties of the wire- based EMD and ESD- cable are defined within IEC 61375, part 3.

Cable Structure

Dependent from the decisions ESD or EMD respectively line-redundancy or not, there are four different choices for the internal cable structure:

		EMD-S	EMD-R	ESD-S	ESD-R
Description	Signal name	EMD single line	EMD redundant lines	ESD single line	ESD redundant lines
Line_A twisted pair	A.data.P	X	X	X	X
	A.data.N	X	X	X	X
potential equalisation line A	A.bus_gnd			X	X
Line_B twisted pair	B.data.P		X		X
	B.data.N		X		X
potential equalisation line B	B.bus_gnd				X
Shield	shield	X	X	X	X

Personal note from the author

The "potential equalisation lines" for the ESD are required by the standard. When you "forget" them, your system will work anyway; refer also to the measurements on page 101. In fact, for several years of the standardisation process, these lines were not included; they appeared at a relatively late stage. For me, it is not really clear, why these lines are included; but: Standard is standard, so I recommend to include them. Comments invited!

Technical data

These technical requirements are applicable to the twisted pair conductors of all four different cable structures:

Issue	Value
Characteristic impedance measured with sinusoidal signal at 750kHz and 3MHz equals...	120 Ohm +/- 10%
Cable attenuation measured at 1.5MHz less than ...	15 dB/km
Cable attenuation measured at 3.0MHz less than ...	20 dB/km
Distributed capacitance measured at 1.5MHz not more than ...	46 pF/m
Capacitive unbalance measured at 1.5MHz to shield not more than ...	1.5 pF/m
Crosstalk rejection from Line_A to Line_B measured at 750kHz to 3MHz higher than...	45 dB
Shield quality measured at 20 Mhz see IEC 60096-1 (IEC 60096-1 mainly addresses coax cables, but it may be used for twisted pairs considering the short circuited line pair as the inner conductor and the shield as the outer conductor. Transfer impedance less than ...	0,02 Ohm/m
Differential transfer impedance less than...	0.002 Ohm/m
Twisted pair	12 twists/m
Cross- sectional area of wire recommended...	between 0.34mm2 (AWG22) and 0.56mm2 (AWG20)

Personal note from the author

To be honest: I have never seen a cable, that really fulfills all of these specs (You did? Please tell me...).

Most twisted pair cables on the market do not fulfill the attenuation figures, some are not "double twisted pair" but "quadruples", some do not comply to general railway requirements (what happens to the insulation at -40° Celsius?).

In general, the "cable attenuation" is tough to achieve and this leads to fairly stiff and heavy cables (I am always joking about "indoor plumbing"). From this point of view, I understand the wish from the application engineers for slim and flexible cables.

Please take the following paragraph as a guide for those cases, where you do not have to fulfill the IEC61375 in perfection. Dependent from the segment length and the number of nodes, you may get an optimised approach for your specific application.

Attenuation calculation

The maximum attenuation for the signal travelling between any two devices is 8dB. This is a requirement for the frequency range between 750kHz and 3MHz.

The total attenuation is usually determined by the attenuation from the cable length plus the attenuation from the number of nodes (0.15dB per node).

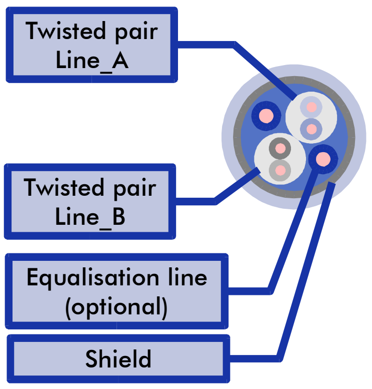
Example: Does a cable with an attenuation of 22dB/km work with 12 nodes and a cumulated bus cable length of 80m?

Attenuation from the cable: 1.76dB = 22dB/1000m * 80m
Attenuation from the nodes: 1.8dB = 0.15dB/node * 12 nodes
Total attenuation = 3.56dB; which is less than 8dB: Yes, this will work.

Wire allocation

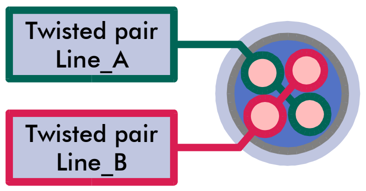
Cables with 2x2 wires ...

The classic cable assembly consists of two pairs of wire; each of them having the two wires twisted around each other. As an option, the power equalisation is routed as separate wires. The shield covers all wires together.



...or with 1x4 wires

Also widely used is the "wire quadruple". This assembly method usually yields smaller cables and is also available with optional equalisation lines (not shown in the picture on the right side).



Note: be sure to have the wires allocated to Line_A and Line_B "over cross"; as shown in the picture. With this arrangement, the unwanted cross coupling between the two wire pairs is minimised.
If this rule is not fulfilled, there may be problems in the field, which are hard to locate: the maximum bus length is significantly reduced, and/or the behaviour with a cable break is completely confusing.

SUB-D- connectors

There is a wide variety of different manufacturers for “sub-D” cable connectors. For the cable connector itself, the user may choose a supplier to his own requirements: the various versions are selected according to obvious quality level versus cost considerations. There are no big issues left being special with the use of the MVB.

However, for the connector hoods, duagon recommends to check some issues more deep:

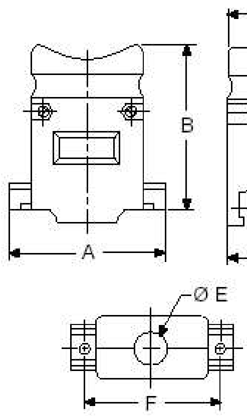
- **Shield continuity**: The MVB requires to have a shield being routed through the nodes. Therefore the user has to make sure, that the connector has a good conducting path between the cable shield and the connector shell.
  In this sense, a metallic hood is the best solution. Plastic hoods with metallisation are less than perfect; pure plastic is recommended for special purposes, only (e.g. in-rack- cabling, lab use).
  Some connectors have “dimples” with the intention to install a conductive path between the two metal shells. These are of benefit, but it is good design practice, not to rely completely on these contacts (almost never, they are specified with e.g. contact resistance). Always consider the screw cable locks as the main shield contact.
- **Mechanical compatibility**: The hood must not interfere with the node´s housing. The main questions are:
  - Does the cable connector hood allow to fully mate on the fixed Sub-D- connector?
  - Does the size of the hood allow to plug in both MVB connectors at the same time? The distance of the two MVB-connectors may be too close to each other.
- **Cable locks**: The MVB uses M3- thread for the cable locks. Be sure not to apply the UNC4/40 thread to it; it will damage the thread.

Cable hood examples

The list of cable connectors below is not complete. This relates both to the number of manufacturers as well as to the number of different connector types. Since there are this many numerous options, our best experience is to contact the local dealer and get a recommendation for your application needs.

Note: the text below contains quoted “keywords”. This is a reasonable way to start the search on the respective home page (everyone has a different name for “hood”).

www.ittcannon.com



“Die cast zinc metal backshell” Part number for the hood e.g. 980-2000-345.

www.erni.com

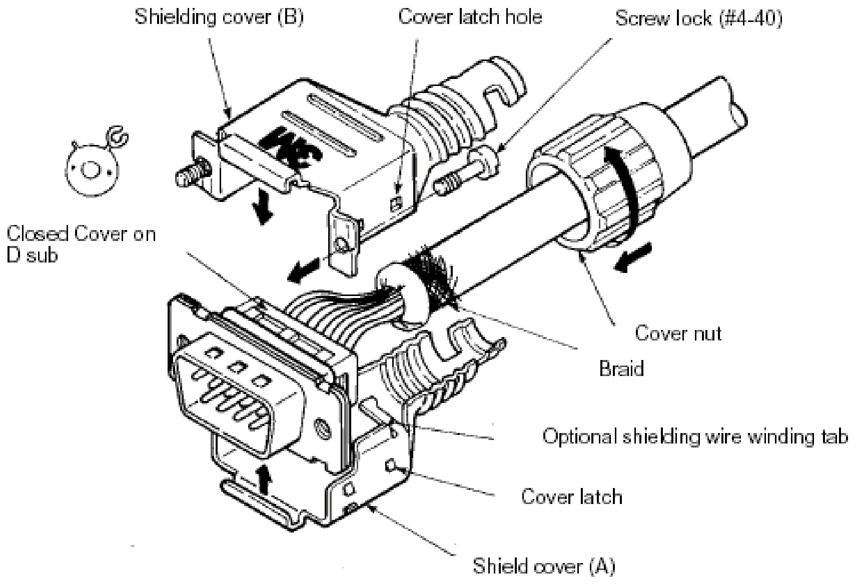
The “cable connector housings” of the series KSG200 are an interesting mix of plastic hood with metal shield inserts.

www.harting.com

Full metal and metallised thermoplastic “hoods”.
Top entry and side entry versions. Some of the side entry versions have only one lock screw: please check, whether the reduced mechanical stability suits your application.

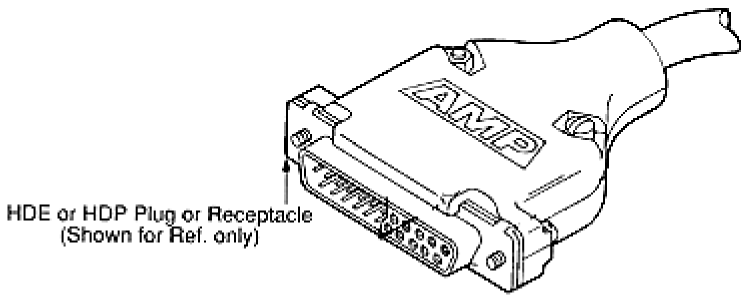
www.3m.com

"Metal EMI junction shell", 3357-92xx series.



connect.amp.com

Look for "cable clamps", they have a wide variety. AMP is now within "Tyco Electronics".



www.mhconnectors.com

For example "Mhulti Sub D Hood System".

Reflections

What they are useful for...

“Reflections” are used in TDR “time domain reflectometers” in order to measure the impedance of a transmission line over its length. As a measurement result, a chart is generated showing either the “characteristic impedance” or the “reflection factor” over the length of the line.

In general, it is good to have a smooth 120Ohm as specified for the cable, all over the whole line, including all connectors and node transceivers.

However, due to the generic properties of connectors and mainly RS485-transceiver chips, it is not possible to get a perfect transmission line: real world nodes always show a certain mismatch into the “capacitive” direction.

Using a TDR makes sense for checking the cable integrity in general. You will see the length of the transmission line, the nodes distributed on it and the amount of reflections/ impedance mismatch.

A practical question: Where is the threshold to decide between “still good” and “already bad”? Unfortunately, the absolute height of the “impedance mismatch” depends on the bandwidth of the TDR equipment: a perfect device, with unlimited bandwidth, would find an impedance of 0 Ohm (capacitive behaviour!) respectively 100% reflection. From this point, it is recommended to collect some experience with the particular TDR equipment in use, and define your own limits between “good” and “bad”.

Unfortunately, it is not this easy to get an idea, where this limit should be. Practical life experience shows, that even examples with excessive mismatch (80 Ohm / 20% reflection) still work fine.

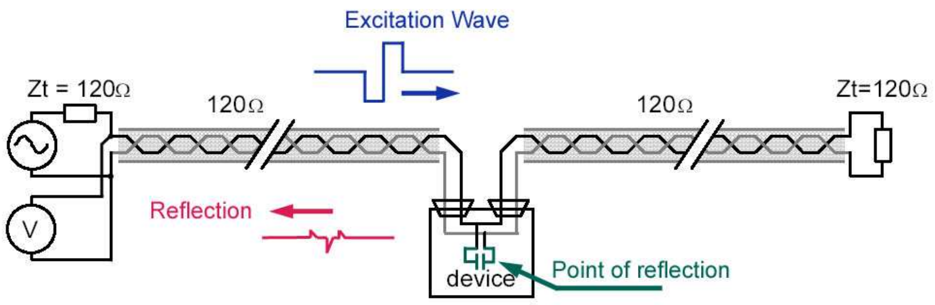
... and what they are bad for...

“Reflections” are also generated during normal operation by the MVB signal frames. In principle, reflections could also disturb the communication. But before going to the conclusions, let´s have the basics first:

What are reflections?

- A signal transmitter drives a transmission line with a defined wave form.
- The signal wave travels along the transmission line.
- When there is an impedance mismatch, then the wave is reflected, travelling backwards in direction to the transmitter.
- The reflected wave depends upon the type of impedance mismatch.

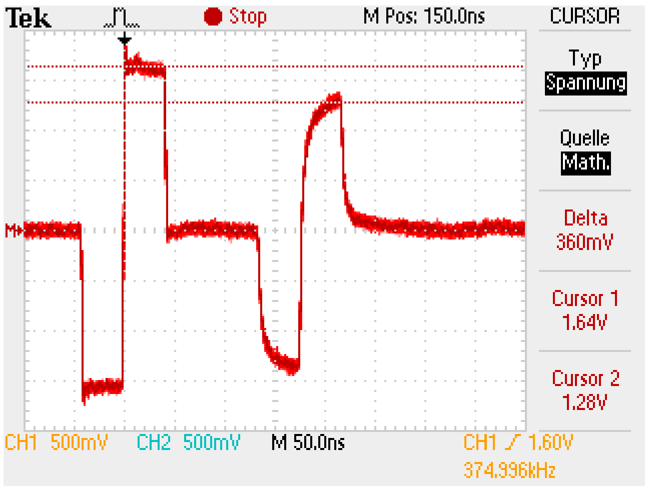
Examples for reflections are „open end“, „short circuit end“ and „capacitive“, as found on typical MVB nodes.



This picture shows a simplified MVB with the TDR (=transmitter and receiver) on the left side; the point of reflection in the middle is the MVB device under test. The far end, shown here as a termination resistor, could also be an open end: this has no influence on the measurement of the MVB device, the reflection from the far end can be distinguished by its later arrival back at the TDR receiver.

Please note, that all the pictures shown in this text are made with a waveform being 8 times faster than normal MVB waveforms (42ns instead of 333ns). This was chosen for better visibility of the effects under discussion.

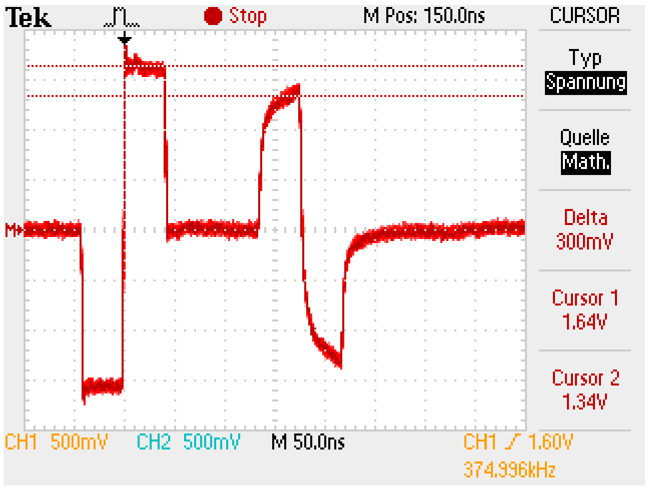
Open end



Original wave and its reflection from an open end, i.e. reflection = 100%.
Cable length 20m.

The cable shows a clear low pass behaviour. The edges are smoothed significantly. The reason for this is the increasing cable attenuation with frequency; main contributor is the skin effect on conductors.

Short circuit

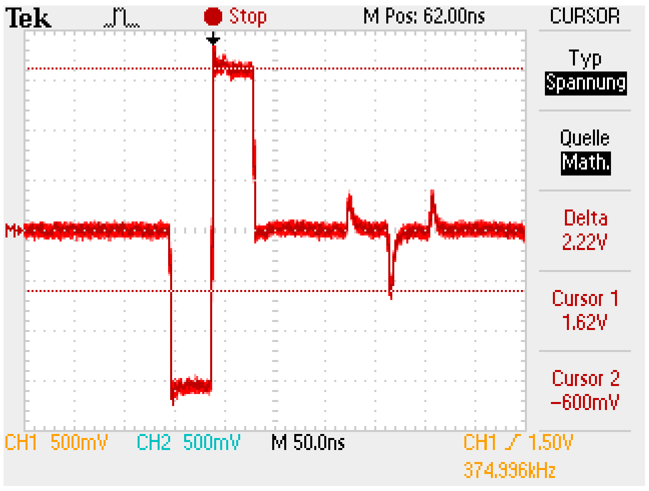


Original wave and its reflection from a short circuited end; 100% reflection.
Cable length 20m.

As a main difference to the open end, the polarity of the reflection has changed.

Note: Independent from “open” or “short” test setup: This simple test is also a nice way to find out the cable attenuation: As you can see, the reflected wave is a little bit smaller than the exciting waveform. In this example, it is very roughly -2dB on 40m of cable. However, be aware that this attenuation relates to the fairlay high frequencies (beyond 12MHz) used in this test. The MVB- standard defines attenuation at 0.75 to 3 MHz.

Capacitive Mismatch

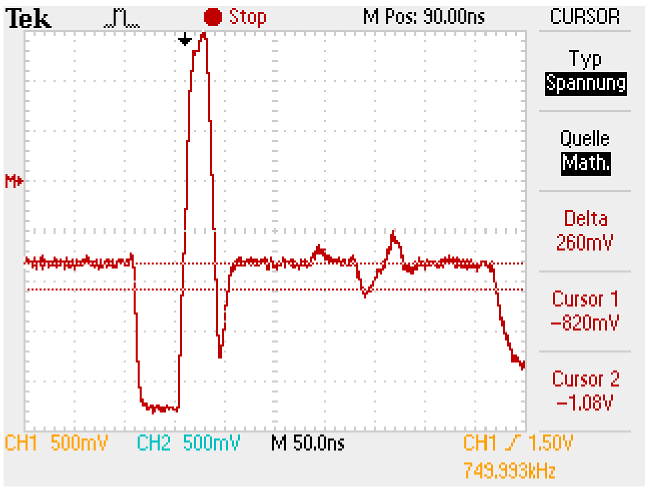


Measuring an MVB device with fairly high reflections.

The exciting wave form comes from a high bandwidth source, like a TDR device.

The reflection has the same polarity as short circuit, this relates to a capacitive behaviour (short circuit for high frequencies).

The reflection clearly shows high pass behaviour: Just the edges of the original waveform are reflected. The main spectral contribution of the exciting waveform (here: 12MHz) is not reflected at all.



Same setup as before, but now the exciting wave form is generated by a real-world- RS485- tranceiver chip (MAX3088): Due to the reduced bandwidth of the sender, the reflection is smaller: 0.260V instead of 0.600V.

TDR measurement conclusions

- TDR equipment will find the MVB node transceivers and the T-stubs from the cable attachment. This is an implicit property of the MVB architecture with multiple nodes on one cable segment.
- TDR equipment checks the wiring at high frequencies: Usual TDR devices are located in the range of 300MHz and beyond. This leads to the fact, that the MVB nodes with their capacitive behaviour will yield more "impressive mismatches" than applicable.
- The frequency range, being relevant for the MVB communication, is between 0.75MHz and 3MHz. This is far below the tested bandwidth from the TDR. Reflections are far lower at these frequencies.

Therefore, TDR equipment can be used very well for cable continuity checks, but not very well for defining an absolute level for good/bad- decision.

Note: This situation is completely different for e.g. 100MBit/s- Ethernet: There, the higher frequencies are really used for communication; reflections are therefore critical. And: they use twisted pair wires in a point-to-point- arrangement, i.e. there are no nodes in the middle of a transmission line! By this way, there are no capacitive reflections from transceiver chips or from T- stubs.

Interference by reflections

Reflections, when travelling in the opposite direction of their originating frame, may disturb communication.
There are basically two different causes for reflections: the terminators at the very end of the line and the nodes with their capacitive behaviour.

- The terminator and the cable impedance are specified with 10% tolerance. When two terminators and the cable impedance use their tolerances to the worst, reflections of up to 20% amplitude are generated (10% reflection by mismatch from one end, superposition from both ends). Without the cable attenuation, an exciting waveform of 5V amplitude will generate reflections of 1V, which is by far too high. Unfortunately, a mismatch in termination will yield reflections also in the interesting frequency range of up to 3MHz.
    - As a remedy, we highly recommend to specify the terminators to a closer tolerance than 10%. For EMD, it seems practical to use 1% resistors; for ESD the proposed combination of 390Ohm and 150Ohm resistors with 1% tolerance has also yielded good results from the field experience.
- The reflections from the nodes mainly consist of high frequency contributions. Due to the higher cable attenuation for these frequencies, an effect can only be seen in the close proximity of the transmitting node. As a result, the reflections disappear within a short time, usually within less than the interframe spacing.
    - For the EMD, the frame decoder will neglect the remaining interference: This feature is basically type tested by the different topic "noise rejection" of the MVB standard (see there paragraph 3.2.5.10.6).
    - For the ESD, there are no specifications with regard to the decoder´s robustness against noise, i.e. all noise with an amplitude beyond 0.75V may become critical.
      As a remedy, we recommend to use a frame decoder design, which filters noise in the same way as the EMD- counterpart. This way, high frequency noise is reliably filtered out.

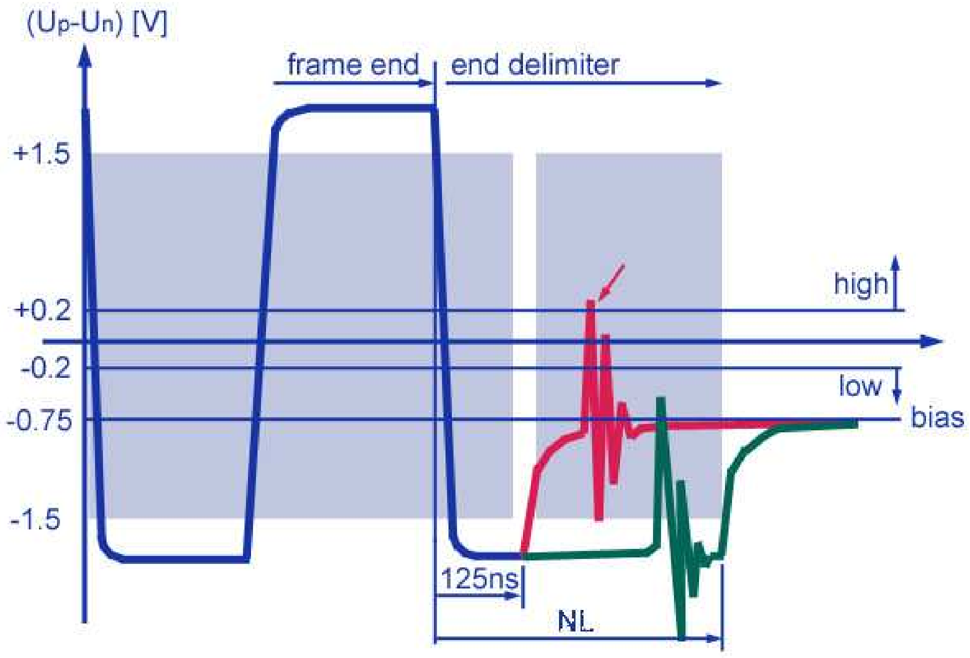
Note: Unfortunately, the popular MVBC01 controller chip does not support this kind of filtering. On top of that, due to a lack in specification, it is excessively sensitive for noise at the end delimiter. If it is not possible to avoid the use of MVBC01- based devices at all, we recommend to have a closer look on communication stability.

ESD end delimiter

About the details...

In the IEC61375, Part 3 "MVB", there are statements that look like a contradiction in itself:

- In clause 3.2.4.8.4, the statement for the "end of frame" is, that the "transmitter shall drive the line actively towards the LOW state for at least 0.125microseconds and at most 1.0BT".
  After this time, the transmitter will stop driving actively, and the line will go back to the idle status, i.e. the 0.75V bias voltage as defined by the terminators (see 3.2.4.2.4).
- In clause 3.3.1.6 "End Delimiter" it is written, that the transmitter shall "append an NL symbol and cease transmission as specified in 3.2.4.8.4". How to understand this? An "NL symbol" is the same as "1.0BT low level": A device, that sends "NL" complies to both clauses. A device, that sends only for 0.125microseconds complies to the first statement, to the second not.



In this picture, the red and green signals represent the two different points in time, where the transmitter switches off (becomes high impedance).

- The usual interpretation of this issue is, that the MVB line, when becoming inactive will have the status "low level" automatically, i.e. the "NL" is not necessarily to be driven.

This interpretation finds its application within the widely used MVBC01 chip: The transmitter stalls driving after the minimum time of 0.125microseconds.

On the other side, all MVB controllers check the existence of the NL symbol and will not recognise a frame as valid if there is something else as NL. Again, this is not a problem, as long as the idle line shows a valid low level (= idle state by bias voltage).

- A potential problem occurs now under non-ideal conditions. Since the MVB line is not driven any more during the end delimiter, the line is to a higher extent subject to interference: this could be any general EMI, but also reflections from other nodes, that propagate backwards on the line. In the picture from before, this is represented by the "red" and "green" disturbances: Obviously, for the same size of disturbance, it is easier to cross to the high level from the bias voltage (-0.75V) than from the transmitter voltage (-1.5V to -5.0V).

...and the consequences

The critical situation is, when interference becomes bigger than the bias voltage, i.e. 0.75V. This could be by means of reflections or other differential mode noise, but also by common mode noise (typically by high EMI levels within the node electronics). "Common mode" type of noise will be suppressed to a certain ratio, but will still contribute to the differential figure.

How to deal with missing frames?

The usual effect from this MVBC01- issue are statistically missing frames (in most cases found out by missing PD frames). When you encounter such a situation, try to make measurements with a scope on the involved nodes: Have a close look to the received end delimiters. When the signal level is "high" or at least close to "high", then you should have a tight look to the general noise situation of your system: reflections, node internal noise, e.g. by DC/DC- converter, etc.

A substitution of the MVBC01 by a better MVB- controller will enhance noise rejection approximately by a factor of 5.

Note 1: For example, the non-MVBC01- duagon products always transmit frames with the full end delimiter. This leads to the fact, that such a device transmitting data to all other nodes is highly reliable.
The problem rises only in case a MVBC01 chip sends a frame to any other node.

Note 2: the EMD bus is not influenced by this discussion. There are completely different end delimiters as well as another strategy against noise (no bias voltage).

Personal note from the author

Obviously, the specification within the IEC61375 for the ESD end delimiter is less than perfect. It unneccessarily reduces noise rejection, especially for reflections.
My recommendation would be: Fix the end delimiter to "transmission of NL" and do not allow "cease transmission after 125ns".
However, this would have the effect, that existing (and basically working!) solutions would not be compliant any more: does everybody agree?
My silent hope is, that any new controller designs will always "send NL".

Oscilloscope Checks

Using an oscilloscope helps a lot during debugging of physical layer issues.

However, the pictures you get from a simple attachment to the MVB lines are often very confusing.

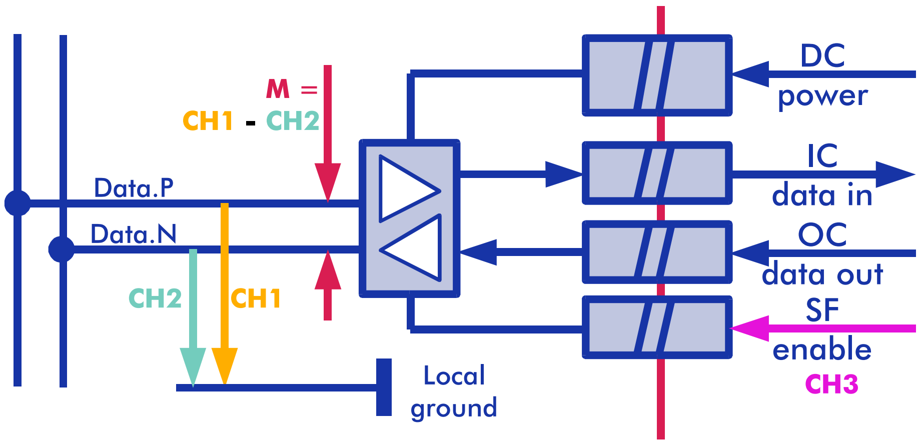
The following examples of oscilloscope printouts may help to understand your own measurements much better.

Scope attachment

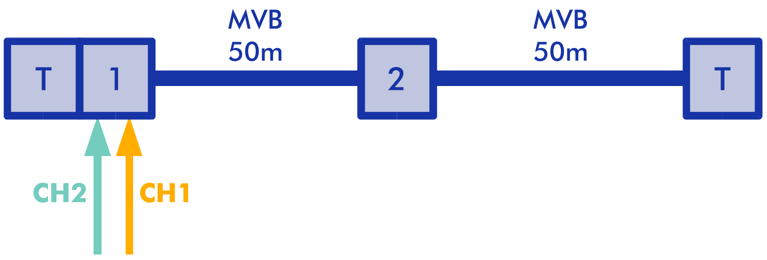
Channel 1 and **Channel 2** represent the MVB lines as they are attached to the node. In order to keep the symmetric properties of the MVB lines, do not use e.g. Data.P for the signal tap and Data.N for the ground strap! The respective ground line is connected to the local ground of the RS485 transceiver chip.

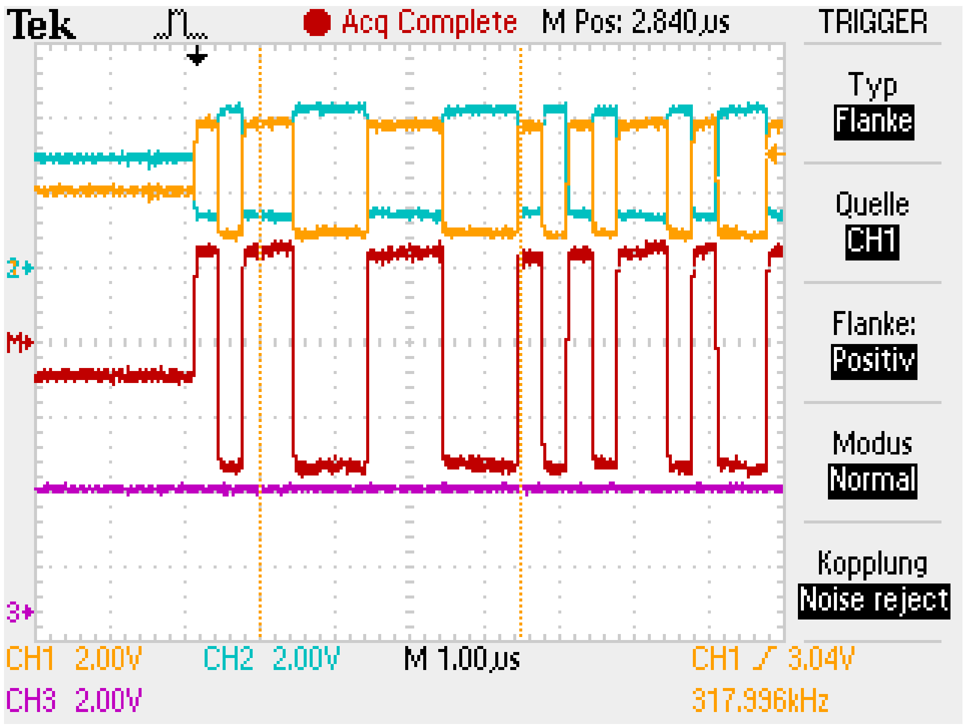
The mathematics trace **M** is the difference between Channel 1 and 2. The RS485-receiver detects the difference between the two inputs, i.e. the M trace is effectively the signal waveform being relevant for the receiver.

Channel 3 is sometimes used for trigger purposes. "SF" is the signal, which enables the RS485- transmitter; this is ideal to trigger on transmission from the own node.



Everything OK



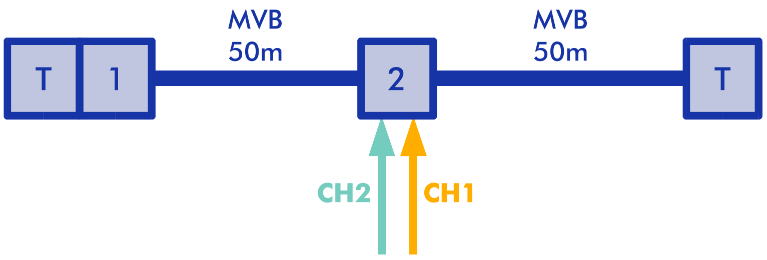


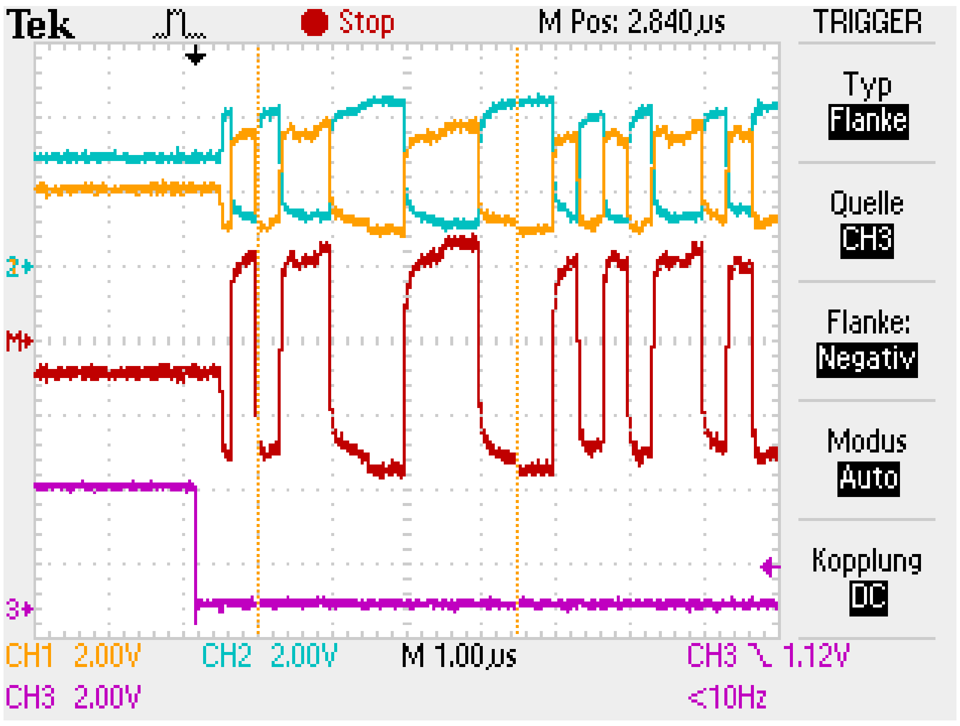
This is a perfect looking start of a frame, in this case sent out from node 1 and measured at the same place.

Ch1 and CH2 are working inverted to each other.

Before the frame starts, there is a “bias” voltage between the lines (ESD!). The nominal value is -0.75V.

Everything OK



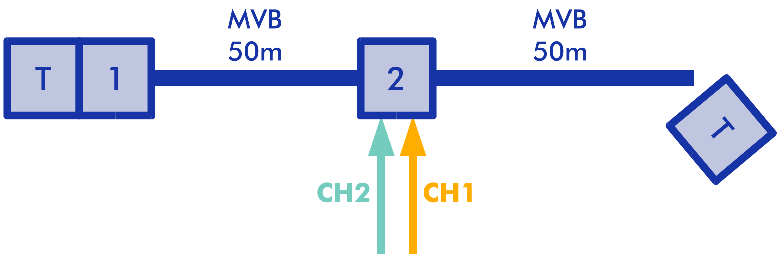


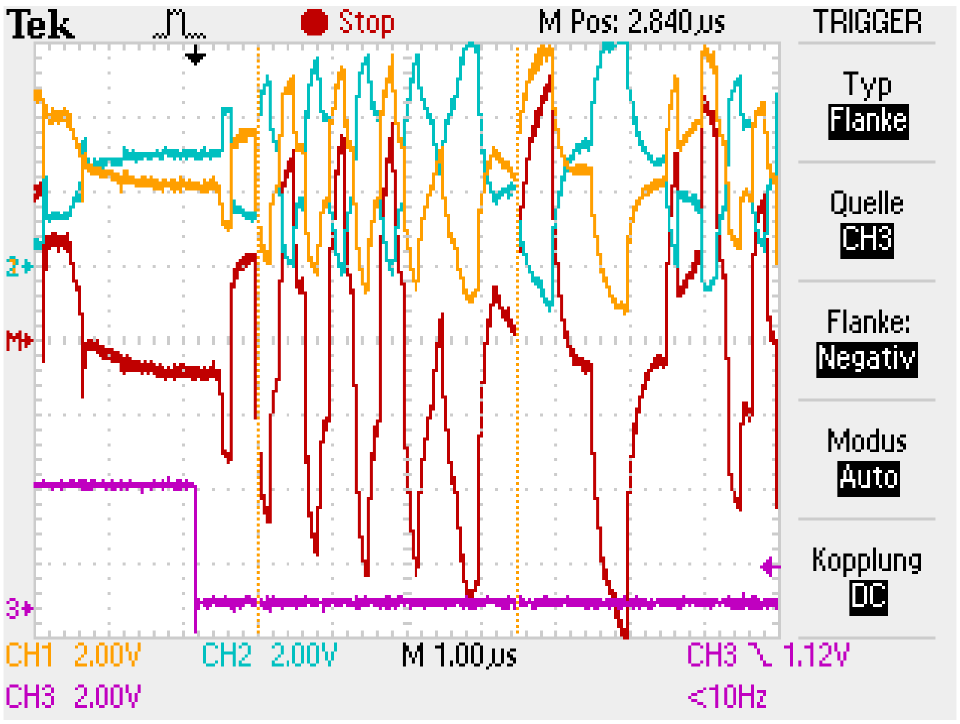
In this case the frame is sent out from node 1 and measured at node 2.

This is still a perfect looking frame, but now after 50m of cable. The “square wave” suffered a little bit from the low pass behaviour of the cable.

Note: there are no other nodes on the cable between node 1 and 2. If this would be the case, the low pass effect would be much higher.

Terminator missing

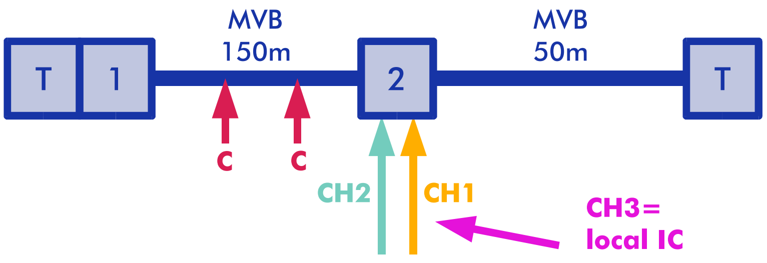


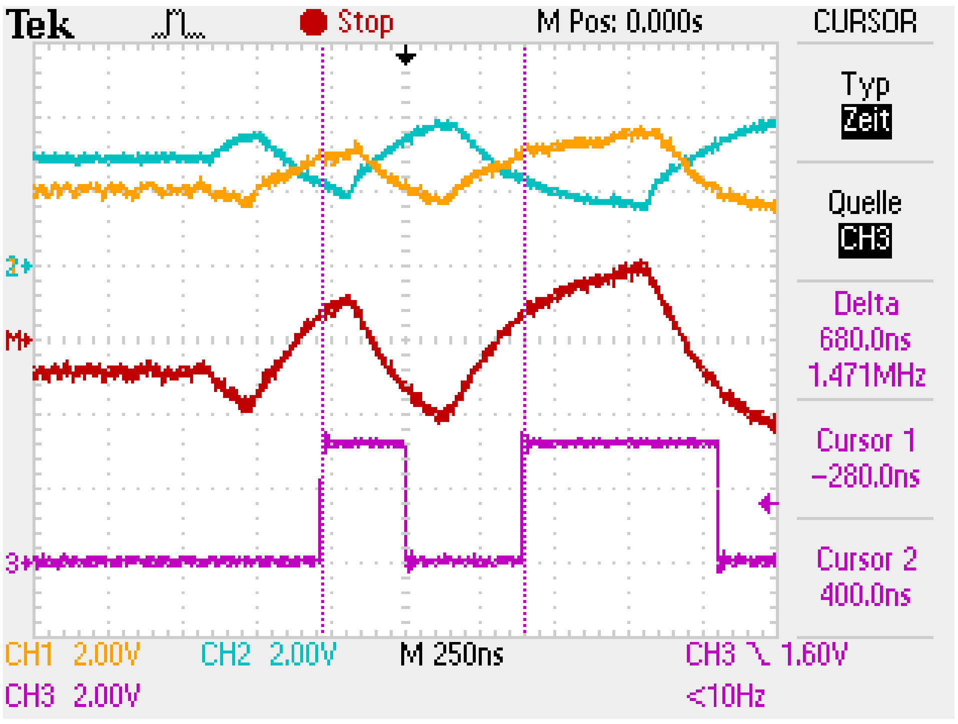


Oops! Communication is not possible any more. What happened?

This is the same measurement as before, but now the **bus terminator** at the far end is missing. Obviously, there is a reason for using terminators.

Full Bus Load ESD+





In this test setup, the total bus segment length was increased to the maximum of **200m**. In addition, a capacitive load was introduced in order to simulate the effect of alltogether **32 nodes**.

Remark: The RS485 transceivers introduce mainly a capacitive load to the bus lines. There is also a resistive load, but this does not play a role at the typical MVB frequencies.

The attachment of nodes increases the "low pass" filter effect of the cable drastically.

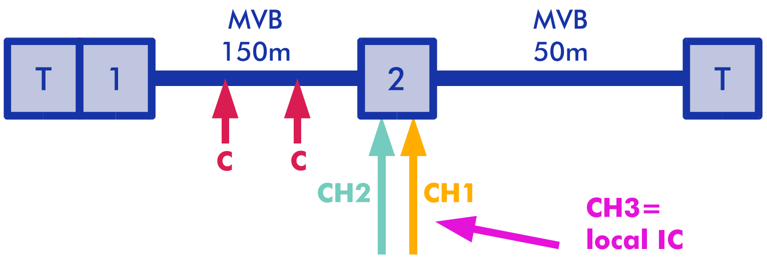
The signal shown in the picture is well understood; i.e. it is still normal. This is reflected by the digital input to the MVB controller (IC), **Ch3**: It is a clear wave form.

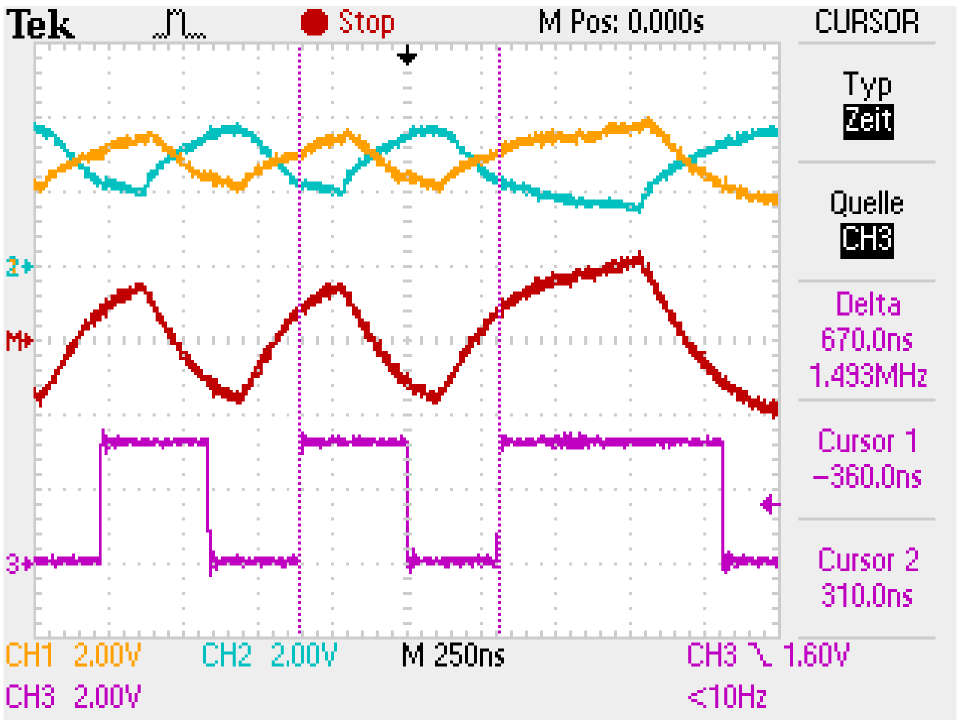
However, a certain trend can be seen:

- The signal is attenuated, showing a triangle wave form.
- The RS485 receiver detects a level change around the M- zero line.
- Since the bias voltage of the ESD+ shifts the signal wave form to the negative side, the MVB controller will see a shorter high level resp. a longer low level.

If the cable would become longer, then the signal wave form would become more attenuated: the communication will stop, because the signal is lost below the bias voltage.

Full Bus Load EMD





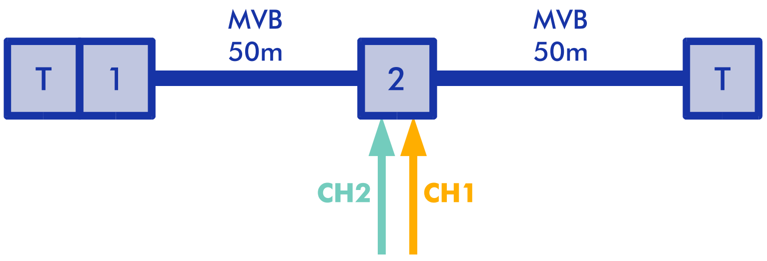
Basically, this is the same situation as in the previous paragraph.

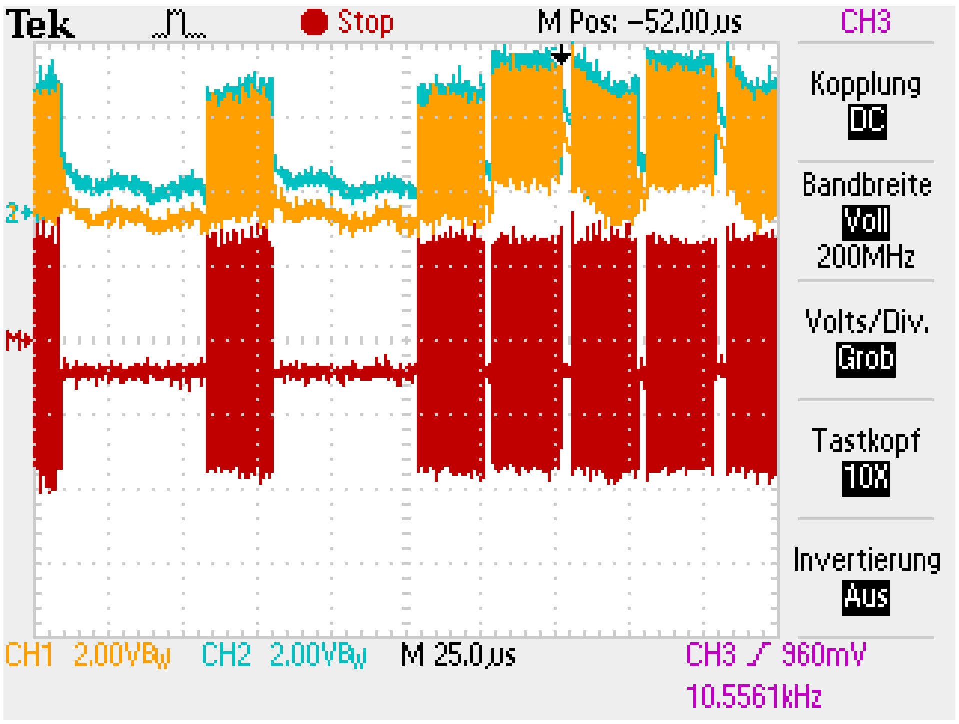
But now, there is **no bias voltage** any more: **EMD** does not use bias and the signal waveform is centered around zero, by definition.
There is no effect of “signals dropping below bias”, as it was the case for ESD+.

The digital receiver signal **Ch3** is much more symmetric than before, but still does not reflect the perfect high and low times. The main reason for this is the used RS485- receiver, which has a threshold being shifted to the negative side by manufacturer specification. This effect is introduced on purpose; it avoids noise ringing during bus idle lines.

Conclusion: For full load situations, the **EMD has obviously more safety margin for reliable operation**.

ESD+ Power equalisation line





This is the same test setup, as it was shown for the “everything OK” case on page 97.

But now, the **power equalisation lines have been taken away**:

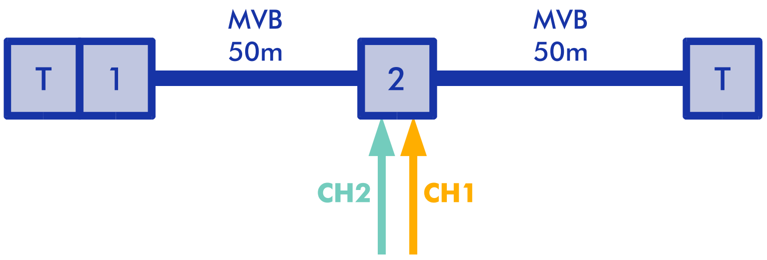
The time scale of the scope was changed to show a sequence of several frames. The frames come from different nodes. Since there is no connection between the local grounds of the involved nodes any more, the average voltage with respect to the local ground will vary:

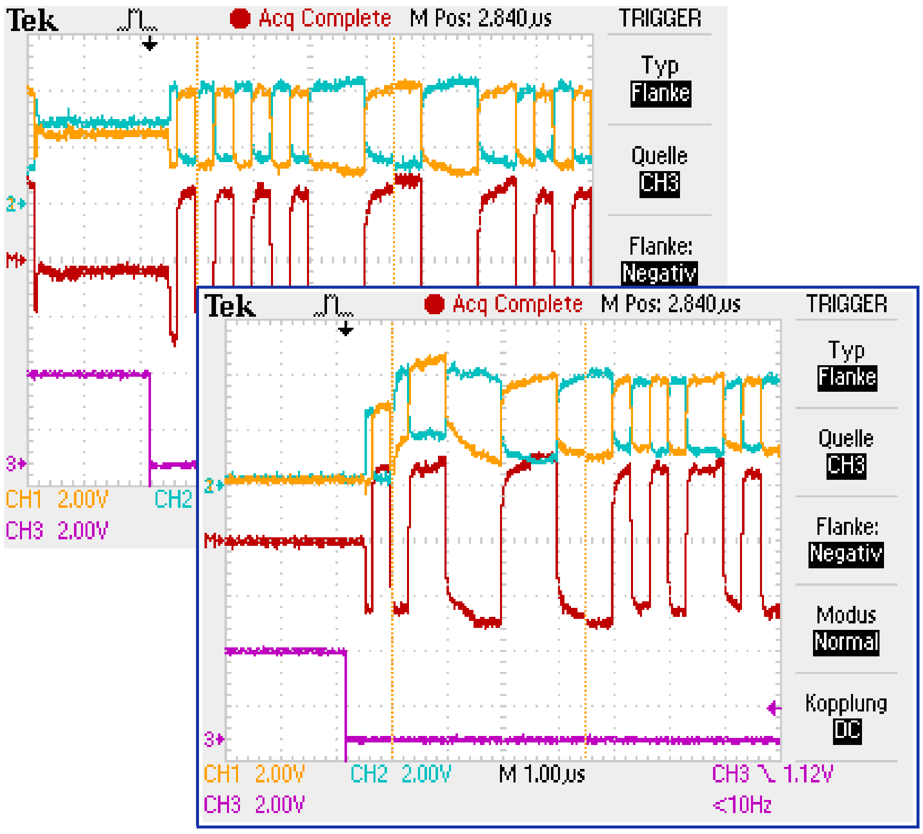
- As long as the node is not active, the weak resistors within the RS485 receiver pull the local ground in direction to the signal signal lines.
- When the node is active, the average of the signal lines is pulled to approximately half of the local driver supply, i.e. typically the local ground is pushed 2.5V to the negative direction.

Conclusion 1: With missing power equalisation lines, the local grounds of the involved nodes, when compared to each other, will bounce up and down.

Conclusion 2: Conclusion 1 does not matter. The relevant issue is the difference between the two signal lines, see trace M. It is still perfect; no bouncing at all. Communication is fully operational.

ESD+ Loss of Bias





This is the same setup as shown on page 97 for the "everything OK" case.

First **one of the two terminators looses power, then the second one**, too.

- The loss of bias can be seen when the lines are idle, i.e. between the frames. One terminator powered will yield approx. 0.75V/2 = 0.375V. When both terminators are down, no bias is left, i.e. 0V.
- In case, both terminators are without power, the signal lines do not have their average voltage of around 2.5V to local ground any more. During idle times, the two lines have around 0V against ground. However, when a node drives the lines, the average of +2.5V is forced, again. This leads to an interesting "common mode wave" (see the right picture above); the wave travels from the transmitting node to the terminators and via the power equalisation lines back to the originating node.

Conclusion (similar to the one on page 101): The difference signal M is not disturbed, communication is up and running – with the exception of the two unpowered nodes with the terminators., of course.

Mixed pickles

This chapter lists some very special issues; intended as “tips and tricks” for the advanced MVB expert.

Priorities during Event Arbitration

The MVB allows an event arbitration to take place with two levels of priority (low or high). By this way, it could be possible to select message data faster or slower, according to its importance.

Unfortunately, the “event arbitration priority” is a feature of the Link Layer, only. It is not supported by the RTP upper layers, i.e. there is no way to control this feature from the application.

Due to this, we recommend not to make use of this feature; use always „LOW priority“, only.

Base Period Timing Variants

The standard allows also other basic cycle times; like e.g. 1.667ms. The idea behind it was to synchronise the bus traffic with for example 60Hz power line frequency (10 base periods * 1.667ms = 16.67ms, which is one cycle of 60Hz).

The potential benefit for this could be, that certain measurements (current, voltage) can be distributed over the MVB without latency. Fast control loops in power controllers do not accept very much delay.

However, this is not this simple to implement: how can a bus administrator be synchronised to an external signal? In addition, the performance figure is not favourable for both sides, the MVB as well as the power drive: The MVB bandwidth for general tasks suffers from the high amount of bandwidth required for the control loops; the control loops suffer from the general traffic.

We recommend not to make use of an other base period as 1.000ms.

MD protocols: transport layer packets

In the IEC61375 document 2 “Real Time Protocols”, paragraph 2.3.6.5, the transport layer packets are defined. In various packet types, the “control field (cl)” is used. The definition is: “(cl) shall distinguish a Call (1) from a Reply (0) message”.

In real life, we have found implementations, that understand this bit differently:

- There is a well-proven implementation (now in the field for more than 10 years), which changes the bit during the course of a “Call” or “Reply”; obviously the understanding is somehow mixed with the “Consumer” versus “Producer” issue.
- Following the text in the standard, the mentioned bit should be constant during the course of a “Call” or “Reply”, respectively. This is the way, how some newer implementations are built.

As long as there are nodes of the same implementation on one MVB, they will work.

When connecting nodes with two different understandings, they will fail to communicate.

Device address allocation

In principle, the whole address space for device addresses should be open for all nodes, i.e. the range from 0 to 4095.

In practice, it is useful to allocate addresses first for the complex Class 2+ nodes:

- The involved bus administrators wait a certain time after power up before they start operation; the waiting time is derived from the device address: low device addresses mean less waiting time: allocate the BAs at the lowest addresses.
- There is a restriction for the RTP implementations, that the “station address” has a format of “INTEGER8”; which limits the addresses to below 256.

After this, all Class 1 nodes may be allocated at random addresses.

Wire redundancy and two Manchester Decoders

It was mentioned in the previous text, that an MVB-controller implemented to the IEC61375 standard includes just one Manschester decoder (...and newer implementations include two of them).

This seems to be a contradiction to the figure shown in Clause 3.1.2. “Structure of this clause”: There are two decoders shown.

Both statements are “a little bit wrong”: The IEC standard defines one decoder for reception of data from the “trusted line” and another decoder which monitors the “observed line” for valid Manchester frames. However, this second decoder is not able to receive data and can therefore not be used for a dynamic switch of the trusted and observed line; i.e. it has a reduced functionality. From this point, may be it would be ok to say “the IEC standard defines 1.5 decoders...”.

EMD insensitivity: Comments to IEC61375

- In paragraph 3.2.5.10.2, the IEC standard asks for the receiver chips to have a “hysteresis of at least 50mV, but no more than 0,25V”. This allows a square signal of e.g. 51mV to be correctly recognised.
- In paragraph 3.2.5.10.4, “the receiver shall not detect a valid frame when the test signal amplitude is less than 0.1V”. On top of this, it is possible to “increase this threshold in certain applications”.

How to understand this discrepancy? Obviously, the second phrase had the intention to blank out frame signals being too low; these may occur for example with a wire fault on the far side. By this way, the paragraph about insensitivity helps to keep the “simple failure model” working (see page 59); unfortunately, it does not work perfect.

Personal note from the author

From more practical considerations, the EMD receiver will be one of the RS485 receivers readily available on the semiconductor market. These devices fulfill the first paragraph, but not the second. To implement a variable threshold, a specific solution for the MVB would have to be created; this does not make sense from the economical point of view.

Assumed, the “simple failure model” is given up (as I propose also for other reasons), then there is no justification for the “variable insensitivity” paragraph. Insensitivity should be the implicit 50mV from the RS485- receivers, nothing more.

Whether a frame is recognised to be OK or not, this is decided by other means (MVB decoder: Manchester coding, CRC calulation etc.), but not by signal level.

Appendix A: Error counters for duagon products

One way to find cable faults is by using error counters. In contrast to the proposal using DSRs only, the error counters will get fine-grained information about communication status between the MVB nodes.

- global error counters will give an overview about the communication status
- detailed error counters could be made available for all process data and DSRs used by an application

Line reception status

Duagon's MVB controller provides for each process data sink port (and each device status port) status information about reception, i.e.:

Status Information	Meaning
SINK_A	Slave frame was received on MVB Line_A.
SINK_B	Slave frame was received on MVB Line_B.

The following information can be get by reading a process data sink port (resp. a device status port) from the MVB controller:

SINK_A	SINK_B	Meaning
0	0	No data received from MVB since last read.
0	1	Data received on MVB line B.
1	0	Data received on MVB line A.
1	1	Data received on both MVB lines.

Time reception status

Besides the above described line reception status, duagon's MVB controller provides for each port (process data as well as DSR, i.e. device status) an "age"-information (see "sink time supervision").

Note: The maximum allowed "age" for a port is derived from the BA repetition rate (a typical application uses 32ms for "fast" and 512ms for "slow" process data).

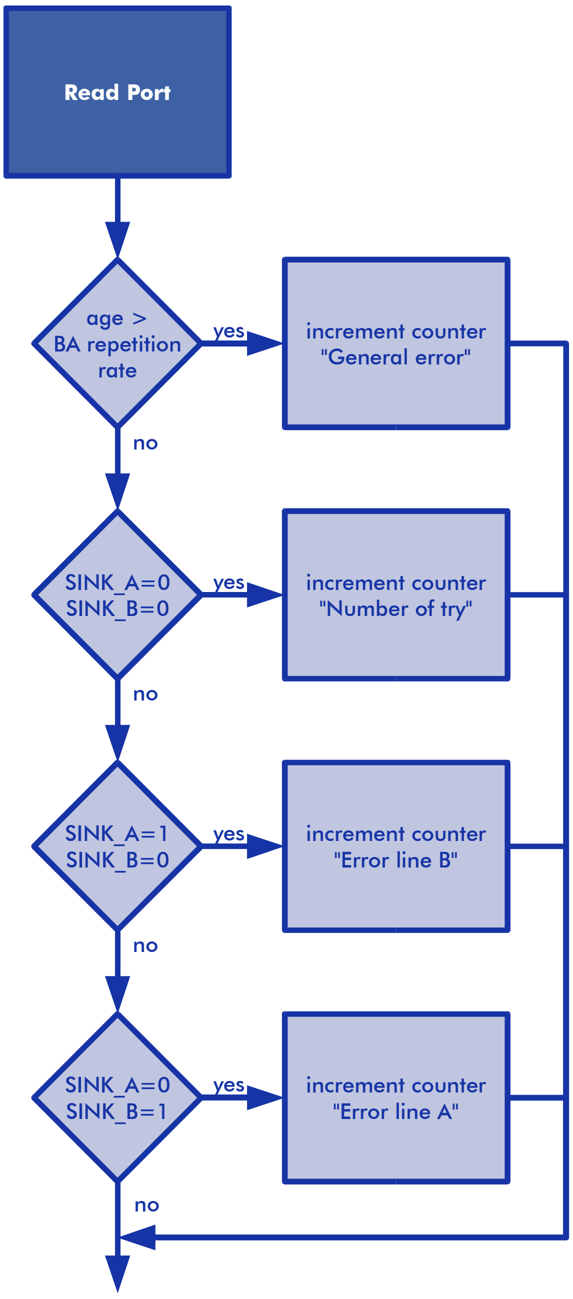
Definition of error counters

The figure on next page shows the algorithm for handling of several error counters on a node. This algorithm should only applied to process data (sink) ports and DSRs, which are used by the MVB application.

Error Counter	Meaning
General error	Nothing has been received at all. **Note**: If the check routine for this error counter was applied to ports, which are used by the MVB application, then this error counter indicates a real error on MVB, e.g. node missing, BA inactive, MVB dead. Otherwise this error counter will increment and indicates a read access to not used ports. Typically process data ports are known to an application software, the list of nodes on a MVB network rather not. From this point of view, this error counter is more suitable for process data ports. **Note**: The check routine for this error counter will indicate each missing reception. An application software should use a similar check routine for the max. accepted "age", e.g. age≤(2*BA_repetition_rate).
Number of try	Nothing has been received since last read access. **Note**: This error counter does NOT indicate a real error on MVB. It just indicates, that the application software is reading a port faster than BA repetition rate.
Error line B Error line A	Slave frame was received on MVB line A, but not on MVB line B. Slave frame was received on MVB line B, but not on MVB line A. **Note**: These error counters indicates a real error on MVB, e.g. cable fault, hardware fault within node. These error counters are applicable for process data as well as DSR.

Note: The link layer software should be responsible for handling these error counters.

Link layer algorithm for handling several error counters on a node:


Read Port
age > BA repetition rate
yes
increment counter "General error"
no
SINK_A=0 SINK_B=0
yes
increment counter "Number of try"
no
SINK_A=1 SINK_B=0
yes
increment counter "Error line B"
no
SINK_A=0 SINK_B=1
yes
increment counter "Error line A"
no

Index