

EKE-Trainnet®
MVB1820B
MVB EMD Interface Module
Technical Manual

EKE-Electronics Ltd.

Piispanportti 7, FIN-02240 Espoo, FINLAND

Tel. +358 9 6130 3308

Fax +358 9 6130 3300

e-mail: electronics@eke.com

Marketing and sales

e-mail: electronics@eke.com

TMS implementations, passenger train functions and

IEC 61131-3 type application programming

e-mail: electronics@eke.com

Technical support and maintenance requests

e-mail: support@eke.com

The most recent information on EKE Electronics' products and services is available at www.eke.com.

Under copyright law no part of this document may be copied, reproduced or transferred electrically or manually, not even partly, without prior written permission of EKE-Electronics Ltd. This document is subject to change without notice.

The EKE-Trainnet® is a registered trademark of EKE-Electronics Ltd.

Copyright © 2016 EKE-Electronics Ltd. All rights reserved.

EKE-Trainnet® MVB1820B Technical Manual V. 1.00

Contents

1. General information	5
1.1 Symbols used in this manual	6
1.2 Correct handling of the module.....	7
1.3 Safety considerations	7
2. MVB module features and operation	8
2.1 Module identification	9
2.2 MVB hardware functionality	11
2.2.1 Hardware overview	11
3. Installation.....	13
3.1 Warnings	13
3.2 Installation in to the rack	14
3.2.1 Diagnostics from the leds.....	15
3.3 Cable Recommendations	17
3.3.1 Bus cable types.....	17
3.3.2 Bus cable connector	18
3.3.3 Termination.....	18
3.4 MVB Firmware Installation.....	19
4. Configuration	21
4.1 Acronyms.....	21
4.2 References	21
4.3 CPU File System (tms-base)	22
4.3.1 MVB related CPU Files	23
4.3.2 Installing File System	27
4.4 ISaGRAF Project	28
4.4.1 Data types.....	28
4.4.2 C functions and function blocks	30
4.4.3 I/O Devices	34
4.4.4 Add PLC to Initialise MVB Interface	38
4.5 Serial console	40
4.5.1 Connection.....	40
4.5.2 Debug Commands	41
4.6 Console from the Master CPU	52
5. MVB Signalling	53
5.1 Connecting oscilloscope	53
5.2 Example waveforms	54
5.2.1 All normal	54
5.2.2 Attenuated signal	55
5.2.3 Missing half.....	56
5.2.4 Missing or wrong termination	57
6. Technical data.....	58
6.1 Dimensions.....	58
6.2 Non-metallic parts.....	59
6.3 Conformal coating	59
6.4 Front connectors.....	60

6.5	Reliability	62
6.6	Safety	62
6.7	Insulation and voltage withstand.....	62
6.8	Environmental and EMC test specifications.....	63
6.9	Disposal of the module	64

EKE-Electronics Ltd. pursues a policy of continual product development. Although every effort is made to produce up-to-date product documentation this publication should not be regarded as an infallible guide to current specifications. We reserve the right to make changes without prior notice.

COPYRIGHT EKE-ELECTRONICS LTD. 2016

1. General information

This is the MVB1820B Technical Manual for the EKE-Trainnet® Multifunction Vehicle Bus Electrical Middle Distance Module (MVB module).



This Technical manual is for system user and for designers.

This manual contains following chapters:

- Chapter 1: General information
- Chapter 2: MVB module features and operation
- Chapter 3: Installation
- Chapter 4: Configuration
- Chapter 5: Technical data



This manual does not include information on other EKE-Trainnet® products than MVB module. If you need information about other EKE-Trainnet® products (e.g. modules, racks or accessories), refer to the manual of the product in question.

This chapter includes general information about this manual. The following topics are covered in this chapter:

- Symbols used in this manual
- Safety considerations
- Correct handling of the module
- Disposal of the module

1.1 Symbols used in this manual

In this manual, situations that require caution are marked with special warning symbols. The warning symbols appear in the beginning of the appropriate chapter. The following symbols are used.



Figure 1.1 Electric shock warning symbol



Figure 1.2 ESD warning symbol



Figure 1.3 General caution symbol



Figure 1.4 Note symbol



Figure 1.5 Separate collection for Waste Electrical and Electronic Equipment according to Directive 2002/96/EC (WEEE)

1.2 Correct handling of the module

Electrostatic discharge (ESD) can damage electronic circuits. EKE products are protected against ESD. However, you run the risk of delivering electrostatic discharges to the module whenever you handle it or any of its components.



To avoid this risk, only handle the MVB module at a static-free workstation. If this is not possible, you must ground yourself using a wrist strap and a resistive connection cord.

Remember to handle the module according to these instructions even when you are removing a defective module and sending it to maintenance.

1.3 Safety considerations

The MVB module is a low voltage device. In a normal situation, it presents no safety risk to the user. However, in case of a severe train malfunction or wiring errors, there is a risk of an electric shock through the bus cables.



However, in case of a severe train malfunction or wiring errors, there is a risk of an electric shock through the bus cables.



Never disconnect the MVB module from a rack with bus cables connected to the module. This can damage the MVB module.



Never disconnect the MVB module from a rack with active power, as this can damage the MVB module or other modules in the system.

2. MVB module features and operation

This chapter includes information about MVB module features and operation. The following topics are covered in this chapter:

- Module identification
- MVB hardware functionality

2.1 Module identification

The following figure shows where to find the version and the modification information.

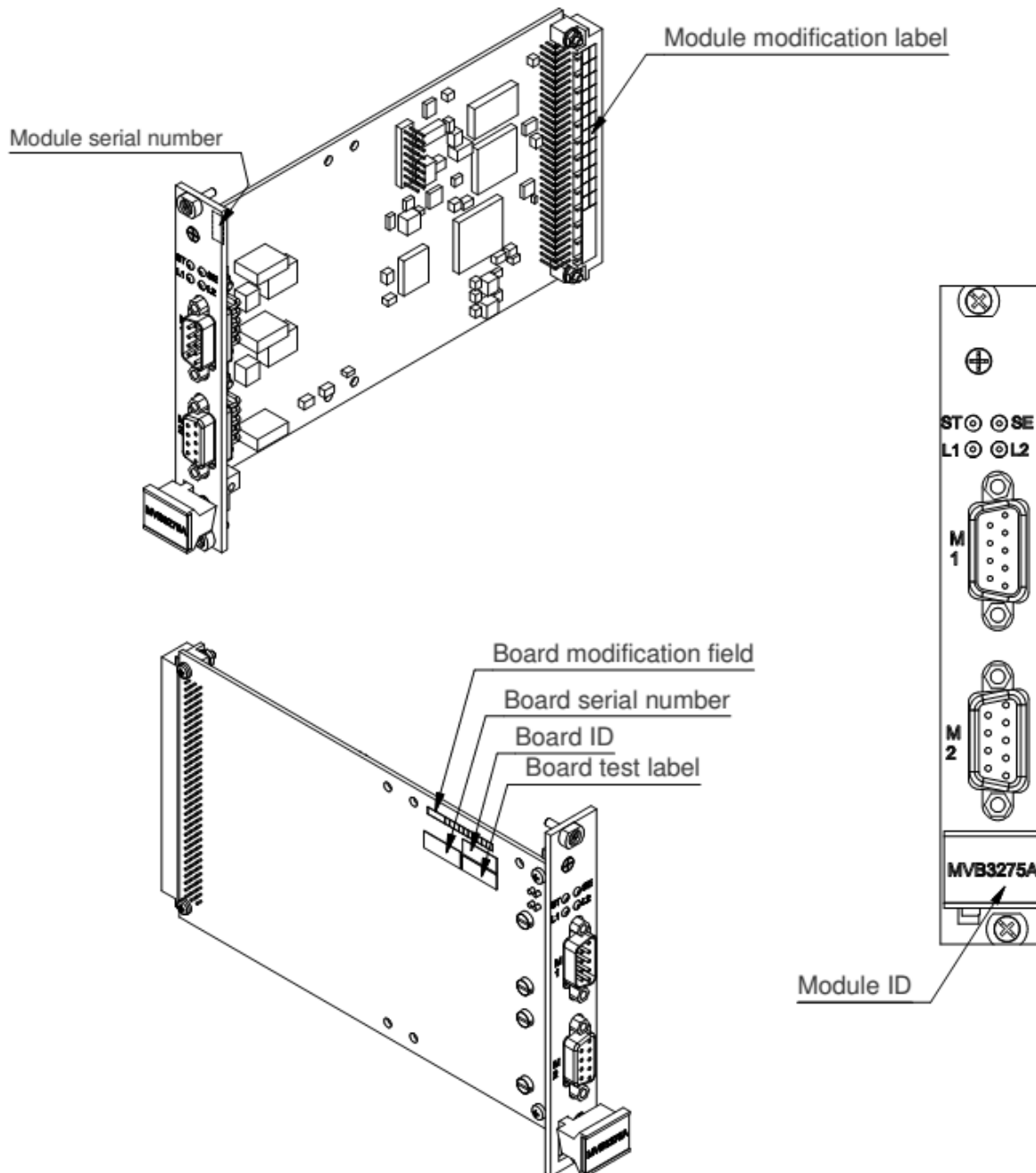


Figure 2.1 The location of version and modification information

A modification label shows the module modification. It is located on the back connector. The label includes letters, and the modification is indicated by crossing out one. If letter A is crossed out, the modification is A, and so on. This modification field is for the entire module.

The circuit board may have a different modification level, indicated by numbers, which are printed on the board's modification field. This is found on the back side of the circuit board on the lower left section.

The Board test label indicates that the module has gone through the required testing cycle. It is located above the board modification field.

The Board serial number identifies the module. It is located above the board test label.

The Board ID above the Board serial number indicates an ID for the circuit board.

The Module ID located just below the front connector indicates an ID for this entire module.

2.2 MVB hardware functionality

2.2.1 Hardware overview

The following figure presents a general functionality diagram of the MVB module.

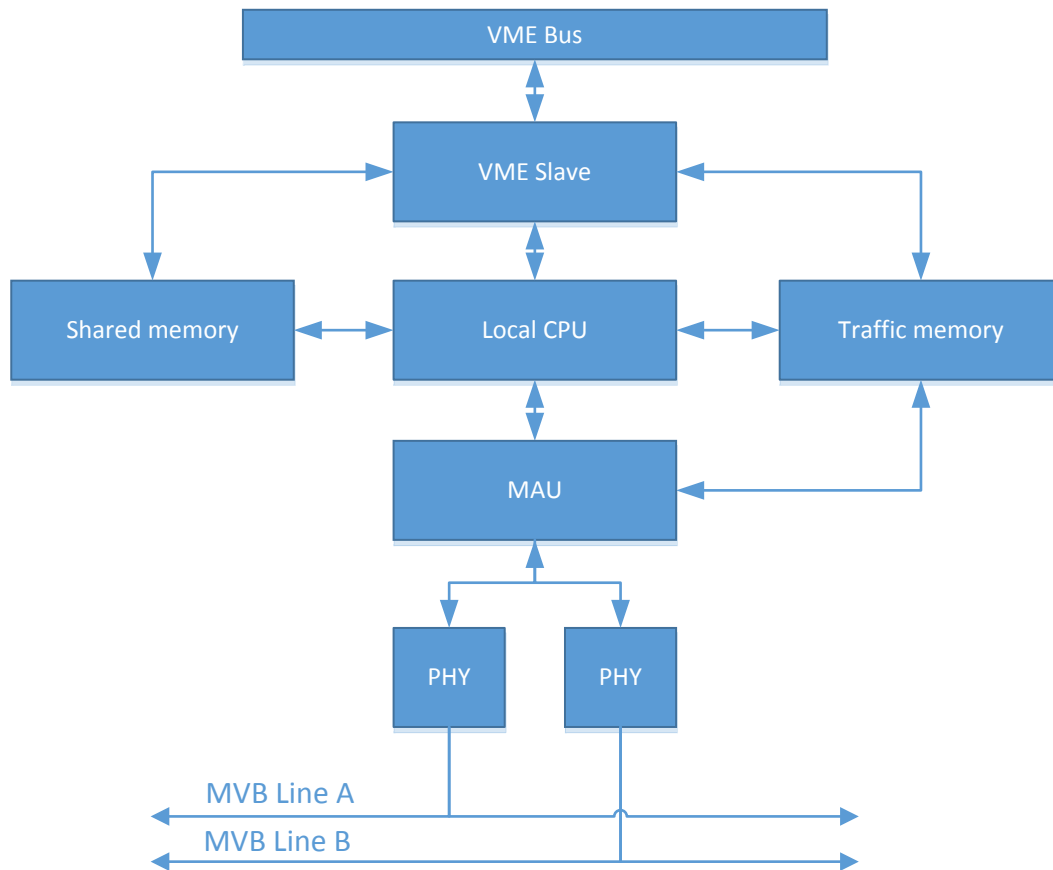


Figure 2.2 Functionality diagram of the MVB module

The module has a fully IEC 61375-1 compliant redundant Multifunction Vehicle Bus interface with Electrical Middle Distance Media (EMD), isolated by transformers. The maximum bus cable length for MVB EMD is 200 metres.

The MVB module is designed to ensure reliable data transmission on board a train. The following sections present some MVB features intended for this purpose.

CPU

The Host CPU MPC8306 combines a high-performance CPU core with a variety of system and resources useful in a broad range of applications.

DUPLICATED LINE

To ensure uninterrupted functionality of the Multifunction Vehicle Bus, the bus is duplicated: there are two lines (A and B) through which devices transmit data. In case one of the lines is temporarily out of order, the other line can take over, and full redundancy is ensured. This way, the flow of important data can continue without interruption even in case of potential problems.

SHARED MEMORY

The MVB module configuration is done through the Gateway CPU and the MVB reads it through its shared memory at the start-up of the system. The configuration is for the MVB module read-only, and it is managed by the CPU software. If any changes are made through the debug terminal, the original configuration is retained when the module is reset. The module has a Medium Attachment unit (MAU) with a MVB FPGA which supports internally the EMD interface.

3. Installation

This chapter includes instructions on how to install the EKE-Trainnet® MVB module into a rack.

The following topics are covered in this chapter:

- Warnings
- Installation in rack
- Diagnostics from the leds
- Cable Recommendations
- MVB cable connection

3.1 Warnings

This section contains warnings you need to consider before and during the MVB module installation.



Do not turn on the rack power before you have finished the installation of the rack.

Never remove the MVB module from a rack when power is applied. Always turn OFF the rack power first BEFORE removing the module.



Electrostatic discharge can damage electronic circuits. Always handle the module at a static free workstation. If this is not possible, ground yourself using a wrist strap and a resistive connection cord.



The module and also the rack can contain sharp edges. Use protective clothing (gloves etc.).

Do not drop the module.



Make sure that the module or any of the components on it do not get wet.

When handling the module, only touch the front mounting brackets. Do not touch the board or any of the components on it. Even when you are removing a defective module and sending it to maintenance, handle the module according to these instructions.

3.2 Installation in to the rack

MVB Module shall be installed in VME or VME/IO slot in the rack.



Figure 3.1 MVB Module installed in a rack

1. Hold the module by the front plate.
2. Place the module on the rails of the rack.
3. Slide the module towards the back plane of the rack until it clicks into its place. Do not use force.
4. Tighten the two screws (slotted head collar screws) on the front panel.
5. If there are other modules in the rack, make sure that the front plate of the MVB module is at the same level with the other modules.

3.2.1 Diagnostics from the leds

When you have successfully installed the module in the rack, the next step is to check that the module works. The front panel of the rack contains four LEDs: one green, one red and two bicolour. You can see the status of the MVB module by looking at the LEDs in the front panel.

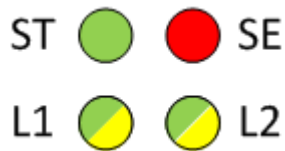


Figure 3.2 Status leds

Led	Colour	State	Description
ST	Green	Lit (normal)	Hardware status. MVB Module hardware is ok to run software.
		Off	Module has no power, firmware completely missing.
SE	Red	Off (normal)	No critical faults.
		On	System error, Self test failure
		Blink	Internal firmware version mismatch.
L1	Green/Yellow	Off	Self tests not completed yet.
		Green blink (normal)	60/60 ms: MVB Traffic OK, active master 100/100 ms: MVB Traffic OK, slave or standby master
		Yellow on	No traffic on MVB Bus
		Yellow blink	300/300ms: Waiting for configuration from Master CPU 60/60 ms: Errors or reply timeouts, active master 100/100 ms: Errors or reply timeouts, slave or standby master
L2	Green/Yellow	Off (normal)	Redundant line OK, or in single line mode.
		Yellow on	Redundant line disturbed (Not used in single line mode)
		Green	Not used

Table 3.1 Some problems that can be detected from the leds

Led behaviour	Possible cause	What to do
Green ST: Off Red SE: On	Backplane SYSRESET~ signal is stuck active, or supply voltage to the module is too low.	Check the other module leds in the rack, if they are the same, try to remove modules one by one from the rack to see if one of them is faulty.
	No firmware	Download firmware to the module, if problem persists, replace with new one.
	Module is defected.	Replace module with new one.
Green ST: Toggles slowly Red SE: On	Self test failure or firmware download has been interrupted	Check from serial console for the possible error message. Download firmware to the module, if problem persists, replace with new one.
L1: Keeps blinking at 300/300 ms.	No MVB Configuration or application on Master CPU	Check Master CPU configurations.
L2: On	Redundant line is disturbed.	Check connectors and cabling
	Single-line cabling, but module configured to use redundancy.	Check Master CPU configurations.

3.3 Cable Recommendations

This chapter includes cable recommendations for the MVB module.

It is very important to connect the cable shield directly to the metallic connector housing. Metallic connector housing provides low impedance path for interference to the rack casing which is connected to the Protective Earth (PE).

3.3.1 Bus cable types

EKE suggests using Huber+Suhner cables. These cables comply with EN 45545-2 fire safety standard.

Table 3.2 Bus cables

Manufacturer	Item number	Description
Huber+Suhner	85003600	2 x 2 x 0.5mm ² , two pairs
Huber+Suhner	85001338	4 x 0.5mm ² , star quad

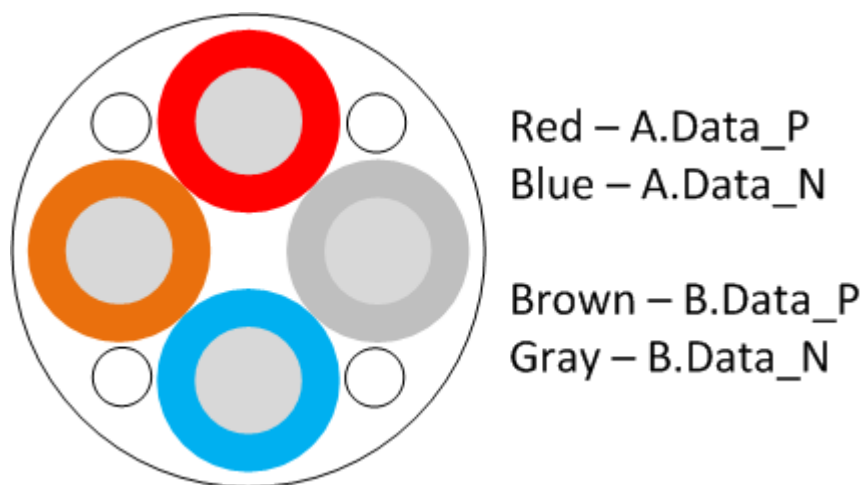


Figure 3.3 Connection of star-quad cable



Note: Older Huber+Suhner cable Item 12553902 has additional equipotential wires (4x 0.25mm²) If using this type of cable, leave these wires not connected.

3.3.2 Bus cable connector

The following tables includes suggested parts to assembly MVB bus cable connectors:

Table 3.3 Bus cable connector part list for M1 connector

Description	Manufacturer	Type
Connector housing, Metal, M3	Harting	09 67 009 0335
D-Sub crimp 9-pole female	Harting	09 67 009 4701
Crimp socket, AWG 22-18; 0.33-0.82 mm ² (4 pcs required)	Harting	09 67 000 3476

Table 3.4 Bus cable connector part list for M2 connector

Description	Manufacturer	Type
Connector housing, Metal, M3	Harting	09 67 009 0335
D-Sub crimp 9-pole male	Harting	09 67 009 5601
Crimp pin, AWG 22-18; 0.33-0.82 mm ² (4 pcs required)	Harting	09 67 000 3576

3.3.3 Termination

A cable segment must be terminated electrically at each end by a terminator. The connector shall be strapped by connecting pin 1 to pin 6, pin 2 to pin 7, pin 4 to pin 8 and pin 5 to pin 9.

Standard terminators are available from EKE-Electronics:

- MVT2752A Terminator for MVB EMD, Connector M1
- MVT2753A Terminator for MVB EMD, Connector M2

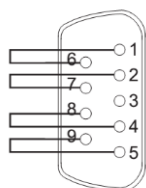


Figure 3.4 MVB EMD Terminator

3.4 MVB Firmware Installation

To update the firmware the following is needed:

- PC with USBFloder adapter box and program
- FAC1027 cable
- MVB Firmware file (UFO-file)



Disconnect power from the rack before connecting download cable

The 16-pin flat ribbon cable connector is connected to the MVB module's download connector, which is marked X4 (DLOAD) on the PCB. To the other end is connected the USBFloder adapter box.

Connect power to the rack.

Double-click the *.ufo file (e.g. MVB1820B_B021_P0000_000.ufo). The loading will start and a loading progress window will appear. During the download the download status indicator is yellow.

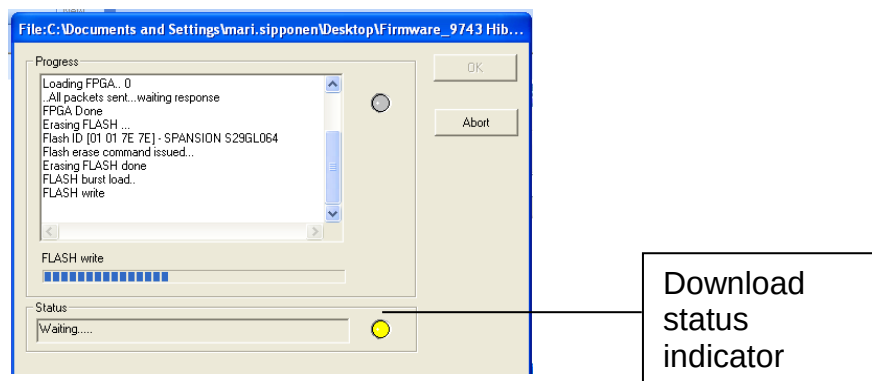


Figure 3.5 USBFloder status dialog during programming

The program informs when the download has been executed successfully by green indicator.

Disconnect power from the rack.

Disconnect download cable from the module.

In case the firmware downloading fails then download status indicator is red.

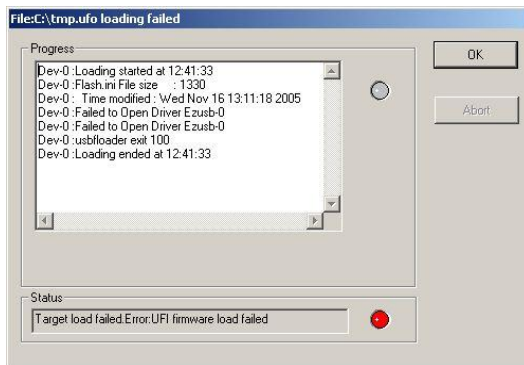


Figure 3.6 USBFloder status dialog showing failed programming

Check the following:

- USBFloder cables are connected properly
- The rack power supply is on
- FAC1027 Flat cable is in good condition.

4. Configuration

This document describes how to configure MVB module.

4.1 Acronyms

CPS	Central Processing unit with Serial links module
CPU	Central Processing Unit
HW	Hardware
MVB	Multifunction Vehicle Bus
TDB	Target Definition Builder
TMS	Train Monitoring System
VME	Versa Module Eurocard bus

4.2 References

/15449/	15449.doc ISaGRAF MVB I/O Card
/16819/	16819.doc MVB – C Application Interface
/22800/	22800.doc MVB – ISaGRAF Application Interface

4.3 CPU File System (tms-base)

All necessary Linux software and parameter files, as well as application software and its configuration, are stored in the CPU module file system. Most of stored information is used by the CPU module or Linux OS and shall not be modified.

Template file systems are delivered as zip files; tms-base.zip. Upload and extract file system into a working directory as these contain configuration files that may be modified later.

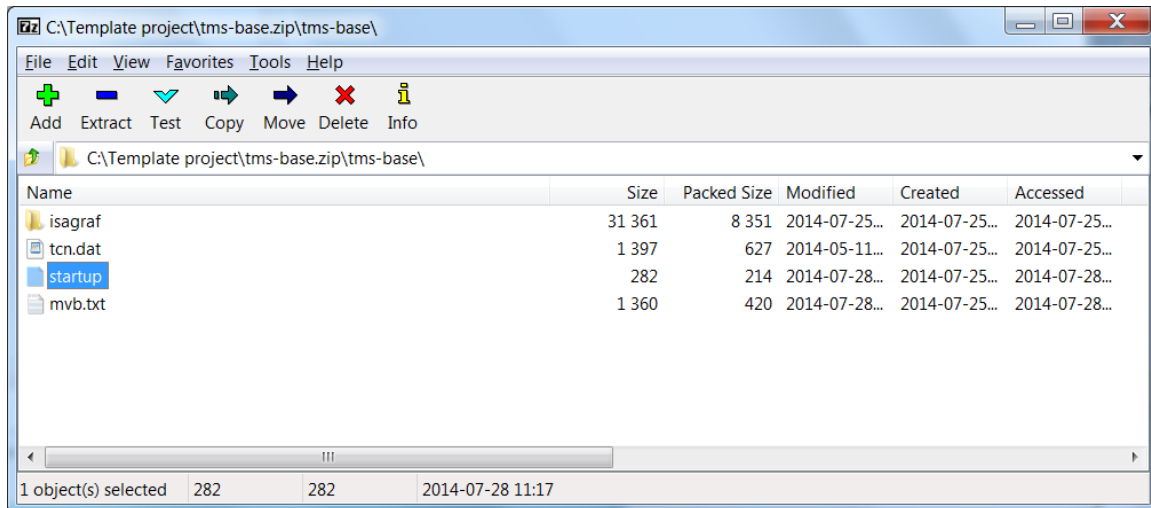


Figure 4.1 Example CPU file system content

The only location having user defined information is in folder

```
var/opt/tms-base
```

and

```
var/opt/tms-base/isagraf
```

For this demo implementation, some existing definitions are there for possible future use only. Mandatory parts are described in detail.

CPU module can be accessed with serial link or Ethernet (Telnet & FTP). For detailed instruction see the specific module manual. For connection details refer to specific CPU module manual.

4.3.1 MVB related CPU Files

The following files are used to configure the target.

Basically only the two files need to be modified **startup**, **tcn.dat** and **mbv.txt**. These are normal text files that can be modified with any ASCII editor, e.g. Notepad.



EKE CPU contains inbuilt editor called Nano. That can be used for convenient, direct editing of the files on the CPU module.

1. Make terminal connection
2. Use Linux command to change directory to `/var/opt/tms-base`
3. Give command `nano [filename]` to start the editor

File	Description
startup	Executed at startup, defining network addresses and starting tasks with parameters
mbv.txt	MVB Bus definition file. Defines devices, ports, polling cycles, etc. Information in this file is loaded from CPU to MVB Module at start-up
tcn.dat	TCN definition file. Defines stations, busses, functions and groups for the TCN communication in WTB and MVB networks

Table 5 Configuration Files Summary

Note: If not advised separately by EKE, do not modify any other files that the above. Further configuring of MVB bus process and message data definitions are done in ISaGRAF application using Target Definition Builder.

startup

Below is template for startup file.

File: startup
<pre># EKE-Electronics ltd 2016 # Created: # Modified: [NETWORK] DEVICE eth0 ADDRESS 10.0.0.11 [TCN] CONFIG /var/opt/tms-base/tcn.dat [ISAGRAF] SCHEDULER NONE CONFIG /var/opt/tms-base/isagraf/ SOCKET 1</pre>

Table 6 startup file content

mvb.txt

Below is template for mvb.txt file.

File: mvb.txt				
#				
# MVB master configuration example				
#				
THIS DEVICE 10h MASTER # MVB master, address 10h				
DEVICE 20h SLAVE # MVB slave, address 20h				
OWN EVENT TYPE HIGH # Own event type is high priority				
POLL EVENTS LOW_AND_HIGH # Poll both event priority types				
MACRO CYCLES PER TURN 5 # Mastership transfer after 5 macro periods				
REPORTING ACTIVE # Send MVB_Reports				
# Process data port configurations:				
#Type	Port	Address	Size (bytes)	Cycle (ms)
SINK	1h		4	1
SINK	4h		32	4
SINK	16h		4	16
SINK	64h		2	64
SINK	256h		16	256
SINK	24h		8	1024
#Type	Port	Address	Size (bytes)	Cycle (ms)
SOURCE	2h		8	2
SOURCE	8h		32	8
SOURCE	32h		16	32
SOURCE	128h		4	128
SOURCE	512h		4	512

Table 7 mvb.txt file content

tcn.dat

Below is template for tcn.dat.

File: tcn.dat
<pre># EKE-Electronics ltd 2016 # Created: # Modified: #This station identification. NAME,Station number THIS STATION GW_STATION 10h #Busses defined for the TCN kernel. Possible types are WTB,MVB and INTERNAL #Notice. If there is more than one module for same bus type. Definitions are #based on rack configuration. Leftmost module is defined first #BUS WTB1 BUS MVB1 #STATIONS DEFINED IN SYSTEM. Station name = Station Number,BUS,ADDRESS #If station is not directly reachable the next station is defined #Only stations (devices) under MVB contain device address # Functions defined in this system. # Functions between F0-FF are set by the system. They are EKE/TCN specific and should not be used by application # Function 1h is used internally in application between x and y FUNCTION 1h CPU_STATION FUNCTION 2h CPU_STATION FUNCTION 3h CPU_STATION FUNCTION 4h CPU_STATION FUNCTION 5h CPU_STATION FUNCTION 9h CPU_STATION FUNCTION Bh CPU_STATION FUNCTION 10h CPU_STATION FUNCTION 11h CPU_STATION FUNCTION FCh CPU_STATION FUNCTION 0fh CPU_STATION # UIC-556 mapping server fucntion FUNCTION FAh CPU_STATION # Event manage FUNCTION FBh CPU_STATION # ROUTER FUNCTION THIS HAS TO BE DEFINED FOR THE GATEWAY INITIALISING WTB FUNCTION FDh CPU_STATION # TCN Agent function FUNCTION FEh CPU_STATION # TCN manager function #Groups defined for this node. GROUP COACH 1 GROUP COACH 2 GROUP ALL_UNITS 10</pre>

Table 8 tcn.dat file content

4.3.2 Installing File System

To install tms-base file system to CPU module the following procedure is recommended:

1. Pack the tms-base folder to tms-base.zip using e.g 7-zip program.
2. Send tms-base.zip to CPU module /tmp/ folder using FTP (e.g. WinSCP).
3. Open Console connection to VCU and install file system:

```
tms-setup /tmp/tms-base.zip --erase
```

Files can also be transferred one by one, or edited directly on target using *vi* or *nano* editor.

4.4 ISaGRAF Project

4.4.1 Data types

MVB_device_status

This structure contains definition of the device status information got as a response to F-code 15 frame. Definition of frame structure is found from IEC 61375-1 paragraph 3.6.4.1.2.1.

MVB_STRUCT

This structure is an example of one MVB-port definition. This structure contains port data elements with copying rules. Each project will contain own MVB-structure based on MVB-ports found from the system. In other words, there shall be a project specific TDB created for each project.

MVB_TrainNode

This structure is needed in an MVB train. See also MVB_TrainNodeArray and mvb_set_train_structure.

Table 4.9 MVB Train Node Structure

Name	Type	Comment
StationId	USINT	Station ID of an MVB train node. This ID mapped to the MVB device address in <i>tcn.dat</i> file located in that node's CPU.
UICIdentificationNR	ULINT	Serial number of the coach in which the MVB node is located.

MVB_TrainNodeArray

Array of MVB Train Node structures. It imitates Node Address Dictionary (NADI) in a train without Wire Train Bus (WTB). See also MVB_TrainNode and mvb_set_train_structure.

Table 4.10 MVB Train Node Array

Name	Type	Comment
MVB_TrainNodeArray	ARRAY[32] of MVB_TrainNode	List of 32 MVB Train Node structures.

MVB_StatusStruct

This structure contains definition of the MVB status card.

Table 4.11 MVB Status Structure

Name	Type	Comment
HeartBeat	DINT	Value increments every macro period. Before initialisation of the MVB card the counter increments every 10ms.
CommunicationErrors	DINT	Number of erroneous frames (or RTIs) sent or received.
CommunicationCnt	DINT	Amount of successfully sent/received frames
ChStatusA	BOOL	Channel status FALSE – Communication failed TRUE – Communication OK
ChStatusB	BOOL	Channel status FALSE – Communication failed TRUE – Communication OK
LinesBroken	BOOL	Indicates disconnect cables or missing terminator FALSE – Terminator detected TRUE – Cable disconnected/no terminator detected
ModuleStatus	BOOL	FALSE – Module communicating TRUE – Module not configured

4.4.2 C functions and function blocks

DisableMVB

This function can be used to stop MVB module communication.

Table 4.12 Function DisableMVB

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
	Output	BOOL	FALSE – MVB disconnected TRUE - Failure

EnableMVB

After disabling MVB communication this function can be used to enable it back to the active communication state.

Table 4.13 Function EnableMVB

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
	Output	BOOL	FALSE – MVB enabled TRUE - Failure

InitialiseMVB

Configures the MVB module and enables it to the line. Normally this function is called once at a start up phase of the ISaGRAF application. If the configuration of the module is large the function call can take even one second. For this reason call this during first cycle before watchdog kick call.

See *Table 7* for example of configuration file.

Table 4.14 Function InitialiseMVB

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
ConfFile	Input	STRING[80]	Configuration file name with path.
	Output	BOOL	FALSE – MVB configured TRUE - Failure

mbv_get_device_status

Function call to read device status. Device status is given back as 32-bit field, which can be split to ISaGRAF structure by calling mbv_dev_stat_i_conv function.

Table 4.15 Function Mvb_get_device_status

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
Device	Input	DINT	Status read from given device address
Status	Output	DINT	Status got from line
ReturnValue	Output	SINT	0 – Success 1 – Given device address was not found from MVB configuration.

mbv_dev_stat_i_conv

Function converts given 32-bit device field to ISaGRAF structure based on copying rule given in structure definition.

Table 4.16 Function mbv_dev_stat_i_conv

Name	Direction	Type	Comment
DevStatus	Input	DINT	Device status field
DevStatusIsa	Output	MVB_device_status	Fields are copied to this structure based on copying rules found from the structure elements.

SetRedundancyMVB

Changes MVB line redundancy. Normally this function is not needed as the MVB-configuration file contains the redundancy definition.

Table 4.17 Function SetRedundancyMVB

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
Redundancy_Set	Input	SINT	New redundancy value taken in use 1 – Line A 2 – Line B 3 – Both lines
	Output	BOOL	FALSE – Redundancy set to new value TRUE – Failure

mvb_get_port_sink_time_supervision

Table 4.18 Function mvb_get_port_sink_time_supervision

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
Port	Input	DINT	Port which sink time supervision counter is read
Counter	Output	DINT	Counter value read for the port
ReturnValue	Output	SINT	0 – Success 1 – Port not in configuration

mvb_set_address

Table 4.19 Function mvb_set_address

Name	Direction	Type	Comment
Index	Input	SINT	MVB card index to which message is sent. 1 – Leftmost module 2 – Second module
Address	Input	INT	Own MVB device address
	Output	BOOL	FALSE – Success TRUE – Failure

mvb_set_train_structure

This function must be called in order to create a Node Address Directory (NADI) in an MVB train.

Table 4.20 Function mvb_set_train_structure

Name	Direction	Type	Comment
OwnNodeIndex	Input	USINT	Own node index. Node number in the local CPU where this command is given.
TrainNodes	Input	MVB_TrainNodeArray	See 0
ReturnValue	Output	USINT	0 – Success 1 – Failure

SetDynamicConfMVB

Defines dynamic groups. See document /15449/ for information on the dynamic configuration file syntax.

Table 4.21 Function SetDynamicConfMVB

Name	Direction	Type	Comment
DynamicConfFile	Input	STRING[80]	Configuration file containing logical addresses for MVB ports
	Output	BOOL	FALSE – File read successfully TRUE – File read fail

SetDynamicGroupMVB

Activation of dynamic group.

Table 4.22 Function SetDynamicGroupMVB

Name	Direction	Type	Comment
Group	Input	USINT	Group activated
	Output	SINT	

4.4.3 I/O Devices

MVB_Inp (static)

Example static input card. Using this template, define own IO-card for each MVB-port as needed. Also own structure with copying rules should be defined for each port. In case of heartbeat defined the default values are copied to the MVB-data structure. Default values are defined in copying rule entries (Offset;Length;Deffault value). OEM-Parameters.

Table 4.23 IO Card MVB input (static)

Name	Type	Comment
CARD	String	MVB card containing defined port MVB1 – Leftmost MVB2 – Second
Port_Id	Word Hexa	Process data port number to be connected
HB_Offset	Word	If port data contains heartbeat counter the offset to it is defined
HB_Length	Word	Size of the heartbeat counter
Timeout	Word	Amount of ISaGRAF cycles before heartbeat is considered not functioning. If heartbeat is not in use set this to 0.
ChannelAddressAddition	Word	If MVB IO-card contains more than one channel port address for each channel has to be calculated. Calculation is based on this parameter. Channel N port address is: $ChannelAddr = PortAddr + (n-1)*ChannelAddressaddition$

MVB_Inp (logical)

Example logical input card. Using this template, define own IO-card for each MVB-port as needed. Also own structure with copying rules should be defined for each port. In case of heartbeat defined the default values are copied to the MVB-data structure. Default values are defined in copying rule entries (Offset;Length;Deffault value). OEM-Parameters.

Table 4.24 IO Card MVB Input (logical)

Name	Type	Comment
CARD	String	MVB card identification: $MVBX_GROUPS=Y$, where X defines the card in rack. MVB1 is the leftmost MVB card in rack. Literal $_GROUPS=Y$ defines that the card is logical group card, where Y defines the groups to which card belongs
Port_Id	Word Hexa	Process data port number to be connected
HB_Offset	Word	If port data contains heartbeat counter the offset to it is defined
HB_Length	Word	Size of the heartbeat counter

Configuration

Timeout	Word	Amount of ISaGRAF cycles before heartbeat is considered not functioning. If heartbeat is not in use set this to 0.
ChannelAddressAddition	Word	If MVB IO-card contains more than one channel port address for each channel has to be calculated. Calculation is based on this parameter. Channel N port address is: $ChannelAddr = PortAddr + (n-1) * ChannelAddressaddition$

MVB_Inp (dynamic)

Example dynamic input card. Using this template, define own IO-card for each MVB-port as needed. Also own structure with copying rules should be defined for each port. In case of heartbeat defined the default values are copied to the MVB-data structure. Default values are defined in copying rule entries (Offset;Length;Deffault value). OEM-Parameters.

Table 4.25 IO Card MVB Input (dynamic)

Name	Type	Comment
CARD	String	Static definition MVB_DYNAMIC
Port_Id	Word Hexa	Dynamic card index definition
HB_Offset	Word	If port data contains heartbeat counter the offset to it is defined
HB_Length	Word	Size of the heartbeat counter
Timeout	Word	Amount of ISaGRAF cycles before heartbeat is considered not functioning. If heartbeat is not in use set this to 0.
ChannelAddressAddition	Word	If MVB IO-card contains more than one channel port, an address for each channel has to be calculated. Calculation is based on this parameter. Channel N port address is: $ChannelAddr = PortAddr + (n-1) * ChannelAddressaddition$

MVB_Out (static)

Example static output card. Using this template, define own IO-card for each MVB-port as needed. Also own structure with copying rules should be defined for each port. Output cards define the types of the own source ports. OEM-Parameters.

Table 4.26 IO Card MVB Output (static)

Name	Type	Comment
CARD	String	MVB card containing defined port MVB1 – Leftmost MVB2 – Second
Port_Id	Word Hexa	Process data port number to be connected
ChannelAddressAddition	Word	If MVB IO-card contains more than one channel port address for each channel has to be calculated. Calculation is based on this parameter. Channel N port address is: $ChannelAddr = PortAddr + (n-1)*ChannelAddressaddition$

MVB_Out (logical)

Example logical output card. Using this template, define own IO-card for each MVB-port as needed. Also own structure with copying rules should be defined for each port. Output cards define the types of the own source ports. OEM-Parameters.

Table 4.27 IO Card MVB Output (logical)

Name	Type	Comment
CARD	String	MVB card identification: $MVBX_GROUPS=Y$, where X defines the card in rack. MVB1 is the leftmost MVB card in rack. Literal $_GROUPS=Y$ defines that the card is logical group card, where Y defines the groups to which card belongs
Port_Id	Word Hexa	Process data port number to be connected
ChannelAddressAddition	Word	If MVB IO-card contains more than one channel port address for each channel has to be calculated. Calculation is based on this parameter. Channel N port address is: $ChannelAddr = PortAddr + (n-1)*ChannelAddressaddition$

MVB_Out (dynamic)

Example dynamic output card. Using this template, define own IO-card for each MVB-port as needed. Also own structure with copying rules should be defined for each port. Output cards define the types of the own source ports. OEM-Parameters.

Table 4.28 IO Card MVB Output (dynamic)

Name	Type	Comment
CARD	String	Static definition MVB_DYNAMIC
Port_Id	Word Hexa	Dynamic card index definition
ChannelAddressAddition	Word	If MVB IO-card contains more than one channel port, an address for each channel has to be calculated. Calculation is based on this parameter. Channel N port address is: $ChannelAddr = PortAddr + (n-1)*ChannelAddressaddition$

MVBSTATUSCARD

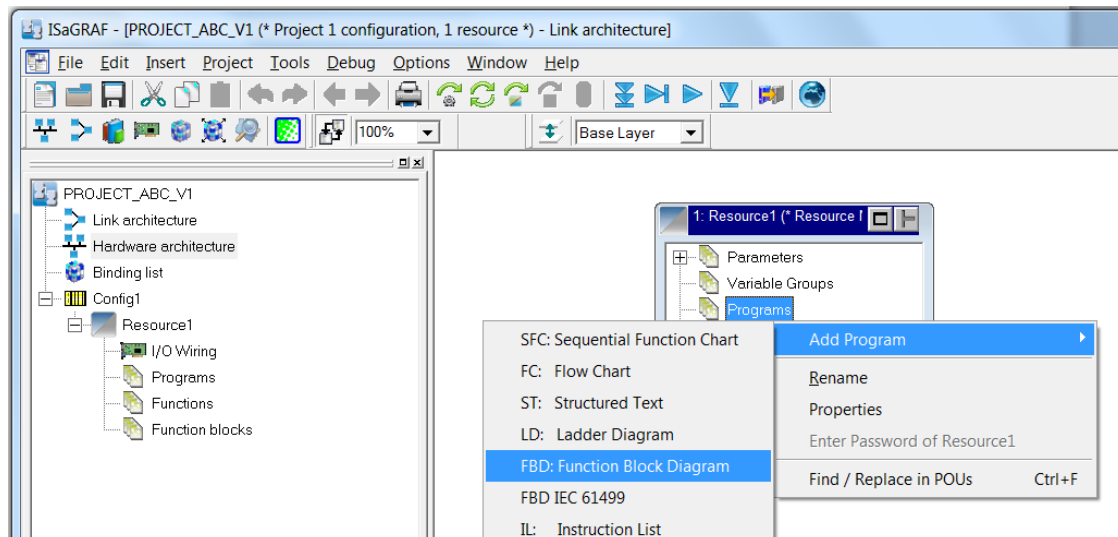
This card is direct interface to the MVB status information. The connection gives user debug information and current line status. OEM-Parameters.

Table 4.29 IO Card MVB Status

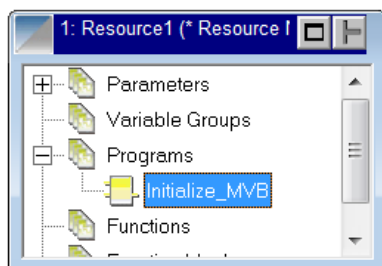
Name	Type	Comment
CARD	String	Module to which card is connected. Normally system contain one MVB and used value is MVB1

4.4.4 Add PLC to Initialise MVB Interface

Step 1: Open Link Architecture view. On the right panel, click Programs > Add Program > FBD: Function Block Diagram



Give a descriptive name to the program.



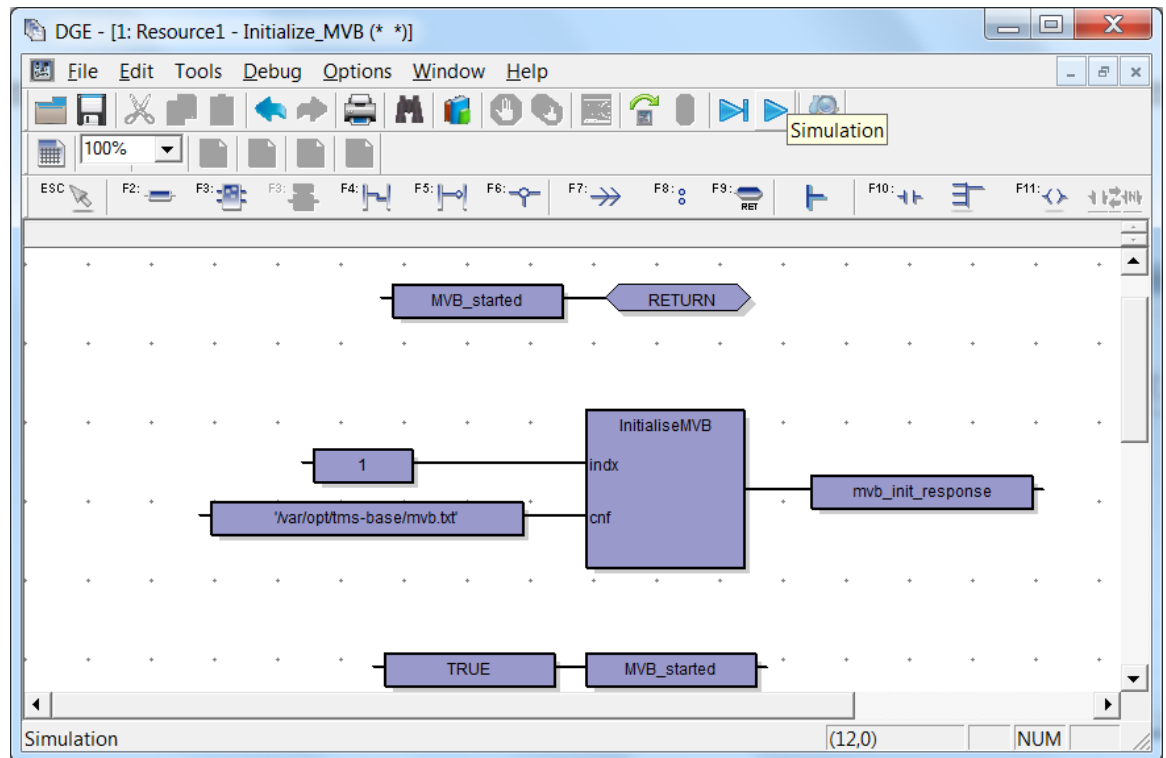
Configuration

Step 2: Add logic to POU (InitialiseMVB).

To add the function InitialiseMVB, press F3 and write the function. If function is not available, import it again from mvb.tdb (or mvb.tdb).

The variables MVB_started and mvb_init_response are of Bool type, with default value FALSE.

The logic below ensures that InitialiseMVB is only called once.



Step 3: Build and Save POU.

4.5 Serial console

4.5.1 Connection

The command line interface is accessible by attaching to a serial console.

In the PC, a terminal emulation program is needed, e.g. Tera Term or Hyper Terminal suits fine. The parameters of the terminal program are:

Baud rate	115200 bps
Data bits	8
Parity	none
Stop bits	1
Flow control	none

Adapter cables of type FAC1027 and CTM1619 are needed.

The cable FAC1027 is to be connected to the MVB module's download connector, which is marked X4 (DLOAD) on the PCB. The other end of FAC1027 is connected to CTM1619A. CTM1619A is connected to PC, either directly to Serial Port or via Serial Link – USB adapter to PC USB port.

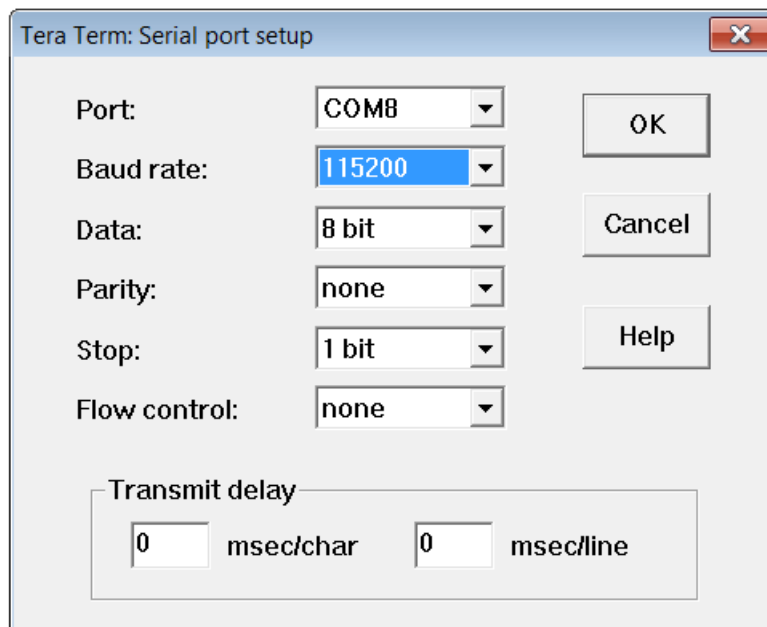


Figure 4.2 Tera Term Serial port setup

4.5.2 Debug Commands

In the following instructions, possible parameters are separated by a vertical line. If parameters are enclosed in square brackets, this means that you can give the command without a parameter. For example, `dm` -command can be used in three different modes. `dm`, `dm.h` or `dm.l` which is shown as follows in the usage:

```
dm[.h|.l]
```

If you do not enter any parameter after the command, the system uses the default value for the applicable commands.

Arguments shown as `<arg_name>` are mandatory, `[arg_name]` are optional.

With some commands you can change the module operation and, thus, the operation of the entire bus. These commands are mainly intended for troubleshooting in laboratory conditions, not on board the train with an active system running. The system will however go back to the default values if reset. In the following sections, the commands are divided into four groups:

The debug terminal commands are listed and described in *Table 4.30*.



The commands marked with  may alter the functionality of the MVB Module permanently.

Some of the command may alter the functionality of the MVB Module temporarily. Consider the operative situation carefully before using this command on board the train.

Table 4.30 System debug commands

Command	Description
help	Shows a brief list of available shell commands.
adm 	<i>This command is reserved for EKE internal use only.</i>
ps	Shows running threads.
dm	Display memory. <code>dm[.h .w] <start_address>[.end_address]</code> Command without size argument, shows output in bytes, with <code>dm.h</code> shows in half-words (16-bit) and with <code>dm.w</code> shows in words (32-bits). Example: <code>dm.h 1000..100f</code> 00001000 9421 FF40 9061 001C 7C7A 02A6 9061 00A8

pm	Patch memory. pm[.h .w] <address> <value> Writes the value in selected width to the memory at address
volts	Shows module internal voltages and currents. Example output: volts <pre> PHY_CURRENT 4 mA VCC_CURRENT 396 mA VTT09 911 mV V10 1000 mV V12 1199 mV V18 1824 mV V33 3338 mV VCC 5047 mV </pre>
temp	Shows module temperature sensor readings. Example output: temp <pre> AMB 32 C CPU 42 C ETH 30 C DDR 31 C </pre>
reboot	Reboots the module.
diic	Dump I²C device register. diic <address> <register> [count] Example: diic 4c 0 5 Reading I2C: Addr: 4Ch, Reg 00h: 20 2A 1E 1F FF
piic	Patch I²C device register. piic <address> <register> <data>
deep	Dump EEPROM contents. deep <start_address>[.end_address] Example: deep 0..f 0000 6D 6F 64 75 6C 65 3D 4D 56 42 33 32 37 35 41
peep	Patch EEROM contents. peep <address> <value1> [value2]
memtest	Fill and verify memory with a pattern. memtest[.h .w] <start_address> <size> [opt] Where opt is 0 – write and verify once (default) 1 – just verify once 2 – just verify, repeat forever 3 – write and verify, repeat forever Warning: Options 2 and 3 will repeat until module is rebooted.
uptime	Shows module uptime.
set	Show or set environment variable. set [variable_name] [value] Without arguments, a list of current variables are shown.
unset	Clear enviroment variable. unset <variable_name>
store	Stores environment variables to EEPROM. Use set and unset to manage variables.
fpgauid	Shows module FPGA version string.
ver	Shows module version details.
sernum	Shows module serial number.



The commands marked with  may alter the functionality of the MVB Bus. Consider the operative situation carefully before using this command on board the train.

Table 4.31 MVB debug commands

Command	Description
chport 	Change logical port parameters. <code>chport <handle> <i o> <addr> <size> <bp></code> Where : <code>handle</code> – Reference to port in Class 1 device (hex) <code>i</code> – Sink or <code>o</code> – Source <code>addr</code> – Port address (hex) <code>size</code> – Port size in words (hex; 1, 2, 4, 8 or 10) <code>bp</code> – Basic period in which the port is polled (hex) Use this command to modify a logical port specified by a handle. You can change all of the preconfigured parameters: type (source/sink), logical address, size and basic period. Source is identified with the letter 'o' and sink with the letter 'i'. Note: The Class 1 device must have been created first with command <code>mpl</code> or <code>mpl.r</code> Example: <code>chport 4 i 204 10 2</code> Modifies port with handle 4 to a sink port, size of 16 words (256 bits), logical address 0x204, and will be polled in 2nd basic period of the macro period.
check_line 	Check line A or line B. <code>check_line [line]</code> Where line is: <code>a</code> – Line A, default <code>b</code> – Line B This command sends 255 frames to either line A or line B, and reports if there were errors. Frame is composed according to IEC-61375-2 (2007), MVB Conformance Test, chapter 3.2.5.1.4., and contains bit sequence: 64 bits at "1", 64 bits at "0", 64 bits with sequence "10" repeated 32 times and 64 bits with sequence "01" repeated 32 times. Example: <code>check_line a</code> Testing line A: <pre>----- - frames sent: FF - errors: 00 - no SFC: 00 - no DTI6: 00</pre> Possible errors are split in two categories; <code>no SFC</code> – Slave Frame not Checked: Slave Frame not noticed on the line. <code>no DTI6</code> – No Data Transfer Interrupt #6: Data not transferred to the Message Sink port. This error indicates MVB internal malfunction.

ctrl 	Execute ls_v_Control. <pre>ctrl <dev_addr> <t_ignore> <command></pre> Where: dev_addr – Device address t_ignore – Not used. Always give 0 here. command – Command byte With this command, you may change device address, or execute a control command. t_ignore, time between a master and slave frame before RTI (reply timeout interrupt) occurs is fixed to 42.7µs and can't be changed. If an argument is zero, it is ignored. Supported commands: 01 – AON: Enable Bus Administrator 02 – AOF: Disable Bus Administrator 04 – SPL: Switch to next Periodic List 08 – TMS: Transfer mastership 10 – SLA: Trust always Line_A 20 – SLB: Trust always Line_B 40 – CLA: Clear Line_A error counter 80 – CLB: Clear Line_B error counter Example: ctrl 20 0 40 Sets the device address to 0x20, executes control command CLA to clear error counters on line A.
get_stat	Show MVB Status. <i>Use this command to display internal status information, line status information and functionality counters. This command also shows standard MVB_Status and MVB_Devices objects.</i> See Table 4.32 for details of shown information.
get_ints	Show interrupt counters <pre>get_ints [reset]</pre> If reset argument is given, counters are cleared. <i>The reply timeouts (RTI) shown in this list contain also reply timeouts to the event polling frames (F-codes 9 and 14).</i> Example output: get_ints <pre>MVBC FPGA active interrupts: DTI_4 (EFS) 0 DTI_5 (MSRC) 0 DTI_6 (MSNK) 0 BTI 1 RTI 12709 SFC 6354 AMFX 6357 DSF 0 DMF 0 ESF 0 EMF 0</pre>

lop	Show configuration information. Example output: lop Basic_Period: 1000 us Macro_Cycle: 1024 ms Macro_Cycles_Per_Turn: 5 Event_Poll_Strategy: HIGH and LOW Own_Event_Type: HIGH Process_Data Ports: <table><tr><th>Address</th><th>F-code</th><th>Type</th><th>Polls in Macro</th></tr><tr><td>0001h</td><td>01</td><td>source</td><td>1024</td></tr><tr><td>0002h</td><td>02</td><td>sink</td><td>512</td></tr><tr><td>0004h</td><td>04</td><td>source</td><td>256</td></tr><tr><td>0008h</td><td>04</td><td>sink</td><td>128</td></tr><tr><td>0016h</td><td>01</td><td>source</td><td>64</td></tr><tr><td>0024h</td><td>02</td><td>source</td><td>1</td></tr><tr><td>0032h</td><td>03</td><td>sink</td><td>32</td></tr><tr><td>0064h</td><td>00</td><td>source</td><td>16</td></tr><tr><td>0128h</td><td>01</td><td>sink</td><td>8</td></tr><tr><td>0256h</td><td>03</td><td>source</td><td>4</td></tr><tr><td>0512h</td><td>01</td><td>sink</td><td>2</td></tr></table>	Address	F-code	Type	Polls in Macro	0001h	01	source	1024	0002h	02	sink	512	0004h	04	source	256	0008h	04	sink	128	0016h	01	source	64	0024h	02	source	1	0032h	03	sink	32	0064h	00	source	16	0128h	01	sink	8	0256h	03	source	4	0512h	01	sink	2						
Address	F-code	Type	Polls in Macro																																																				
0001h	01	source	1024																																																				
0002h	02	sink	512																																																				
0004h	04	source	256																																																				
0008h	04	sink	128																																																				
0016h	01	source	64																																																				
0024h	02	source	1																																																				
0032h	03	sink	32																																																				
0064h	00	source	16																																																				
0128h	01	sink	8																																																				
0256h	03	source	4																																																				
0512h	01	sink	2																																																				
monitor	Configure MVB Module in monitoring mode. <i>The monitoring mode is meant to be used together with 'port_stats' command.</i>																																																						
mpl	Make port list. Creates a Class 1 type Configuration mpl <addr:hex> [port_amount] [ports_in_bp] Where: addr – Device address in slave mode (mandatory) port_amount – Number of sink and source ports (default 1, max 8) ports_in_bp – Number of ports per basic period (default 1) Example: mpl 100 4 2 Creates configuration with device address 0x100, 4 sink and 4 source ports, 2 ports per basic period. Device address (slave) 100 Logical ports: <table><tr><th>Handle</th><th>Desc</th><th>Addr</th><th>Size/words</th><th>F-code</th><th>BP</th></tr><tr><td>0</td><td>Sink</td><td>100h</td><td>1h</td><td>0</td><td>0h</td></tr><tr><td>1</td><td>Sink</td><td>101h</td><td>1h</td><td>0</td><td>0h</td></tr><tr><td>2</td><td>Sink</td><td>102h</td><td>1h</td><td>0</td><td>1h</td></tr><tr><td>3</td><td>Sink</td><td>103h</td><td>1h</td><td>0</td><td>1h</td></tr><tr><td>4</td><td>Source</td><td>104h</td><td>1h</td><td>0</td><td>2h</td></tr><tr><td>5</td><td>Source</td><td>105h</td><td>1h</td><td>0</td><td>2h</td></tr><tr><td>6</td><td>Source</td><td>106h</td><td>1h</td><td>0</td><td>3h</td></tr><tr><td>7</td><td>Source</td><td>107h</td><td>1h</td><td>0</td><td>4h</td></tr></table> To reconfigure MVB module as slave with above configuration, use rec command. To reconfigure MVB module as master which polls a slave with this config, use rec m command.	Handle	Desc	Addr	Size/words	F-code	BP	0	Sink	100h	1h	0	0h	1	Sink	101h	1h	0	0h	2	Sink	102h	1h	0	1h	3	Sink	103h	1h	0	1h	4	Source	104h	1h	0	2h	5	Source	105h	1h	0	2h	6	Source	106h	1h	0	3h	7	Source	107h	1h	0	4h
Handle	Desc	Addr	Size/words	F-code	BP																																																		
0	Sink	100h	1h	0	0h																																																		
1	Sink	101h	1h	0	0h																																																		
2	Sink	102h	1h	0	1h																																																		
3	Sink	103h	1h	0	1h																																																		
4	Source	104h	1h	0	2h																																																		
5	Source	105h	1h	0	2h																																																		
6	Source	106h	1h	0	3h																																																		
7	Source	107h	1h	0	4h																																																		

mpl.r 	Make port list reverse. Creates a Class 1 type Configuration with reverse port order. Same as <code>mpl</code> -command, but source ports will be low order addresses. Example: <code>mpl.r 100 4 2</code> Creates configuration with device address 0x100, 4 sink and 4 source ports, 2 ports per basic period. Device address (slave) 100 Logical ports: <table><tr><th>Handle</th><th>Desc</th><th>Addr</th><th>Size/words</th><th>F-code</th><th>BP</th></tr><tr><td colspan="6">-----</td></tr><tr><td>0</td><td>Source</td><td>100h</td><td>1h</td><td>0</td><td>0h</td></tr><tr><td>1</td><td>Source</td><td>101h</td><td>1h</td><td>0</td><td>0h</td></tr><tr><td>2</td><td>Source</td><td>102h</td><td>1h</td><td>0</td><td>1h</td></tr><tr><td>3</td><td>Source</td><td>103h</td><td>1h</td><td>0</td><td>1h</td></tr><tr><td>4</td><td>Sink</td><td>104h</td><td>1h</td><td>0</td><td>2h</td></tr><tr><td>5</td><td>Sink</td><td>105h</td><td>1h</td><td>0</td><td>2h</td></tr><tr><td>6</td><td>Sink</td><td>106h</td><td>1h</td><td>0</td><td>3h</td></tr><tr><td>7</td><td>Sink</td><td>107h</td><td>1h</td><td>0</td><td>4h</td></tr></table>	Handle	Desc	Addr	Size/words	F-code	BP	-----						0	Source	100h	1h	0	0h	1	Source	101h	1h	0	0h	2	Source	102h	1h	0	1h	3	Source	103h	1h	0	1h	4	Sink	104h	1h	0	2h	5	Sink	105h	1h	0	2h	6	Sink	106h	1h	0	3h	7	Sink	107h	1h	0	4h
Handle	Desc	Addr	Size/words	F-code	BP																																																								

0	Source	100h	1h	0	0h																																																								
1	Source	101h	1h	0	0h																																																								
2	Source	102h	1h	0	1h																																																								
3	Source	103h	1h	0	1h																																																								
4	Sink	104h	1h	0	2h																																																								
5	Sink	105h	1h	0	2h																																																								
6	Sink	106h	1h	0	3h																																																								
7	Sink	107h	1h	0	4h																																																								
msgr	Read unread messages for the receive queue. See <code>msgs</code> command.																																																												
msgs 	Send test messages point-to-point or broadcast. <code>msgs <count> <mode> <dest_addr></code> Where: count – Number of test messages to be sent mode – 0: Point-to-point, F: Broadcast dest_addr – Destination address in point-to-point mode. Example: <code>msgs 2 0 20</code> Send 2 test messages, point-to-point to device 0x20 Received messages can be read with <code>msgr</code> command. If device address is own address: <code>msgr</code> Received message: 1020 6854 7369 6920 2073 6574 7473 6D20 7365 6173 ... Received message: 1020 6854 7369 6920 2073 6574 7473 6D20 7365 6173 ... Receive queue empty.																																																												

rec 	Reconfigure MVB rec [m] if argument m is given: set to terminal mode and configure as master. If no parameters given the MVB will be configured according to the current mode. If <code>mpl</code> command has been used, the MVB is in “terminal” mode and will be configured as slave. Otherwise the configuration is done normally according to the current one in the shared memory. Master has fixed device address 2. Example: rec Slave conf Event: lMV_RECONFIGURE_MVB Event: lMV_START_MVB_TRAFFIC. This device is SLAVE
rp	Read logical port contents (process data). rp <port_address> Where: port_address – Address of port in hex (mandatory)
sp 	Set logical port content (process data) rp <port_address> <word0> [word1 word2] Where: port_address – Address of port in hex (mandatory) word_n – Data to be written to the port in hex

Table 4.32 *get_stat command output information in detail*

MVB Internal	
Internal MVB state	One of the following: • none (not configured) • idle (slave) • periodic_phase (master state) • sporadic_phase (master state)
Internal MVB sub-state	Sub-states for sporadic phase. One of the following: • none (not configured or slave or master in standby state) • dev_poll_state • mastership_transfer • event_poll_start • event_poll_cont
BA state	Bus administrator's state. One of the following: • none (not configured or slave) • standby • regular • find_next • interim
Event poll state	One of the following: • none (event polling not configured) • poll_type_0 • poll_type_1
MVB_Status	
device_address	Own device address
device_status	Own device status
hardware_name	MVB type and FPGA version
software_name	Firmware version name
t_ignore	The time between master frame and slave frame before Reply Timeout Interrupt (RTI) occurs.
line_A_errors	Number of transmission errors occurred when line A was active.
line_B_errors	Number of transmission errors occurred when line B was active.
Counters	
heartbeat (bp)	When the module is acting as active MVB master this counter is incremented every basic period. Otherwise this is incremented in the following intervals: • Waiting configuration: every 300 ms, • Slave or standby master: every 10 ms, • MVB bus traffic disabled (master and slave): every 100 ms.
macro cycles	Number of macro cycles. This counter is relevant only for master.
correct frames	Number of received correct slave frames.

erroneous frames	Number of erroneous frames. If the MVB is configured as slave, the reply timeouts are subtracted from the value as RTIs are not errors from the slave's point of view.
reply timeouts	Number of reply timeouts.
Bus	
status	Ok or BAD. BAD means most likely disconnected cables or bad termination.
lines	Contains the following information: - Line mode: "single" or "redundant", - Line A and B statuses: "1" = Ok, "0" = disturbed, - Current active line. Disturbed status "0" means: - Redundant line disturbed (RLD) bit is set for this line or - Errors occurred for this line or - Single line mode is selected and this line is not used
MVB_Devices List of known devices, Bus Administrators marked with "BA" in master mode.	
address	Device address.
status	Device status. Marked as "n/a" if the device has not responded to device status poll for a certain time. The time time depends on the amount of the polled devices.

Configuration

Example output of get_stat command:

```
# get_stat
MVB Internal
Internal MVB state:      periodic_phase
Internal MVB sub-state:  event_poll_cont
BA state:                regular
Event poll state:        poll_type_1

MVB_Status
device_address           0002
device_status            5380
hardware_name            EMD-A3355-P00-REV00
software_name            E021.P0000.003
t_ignore                 42 us
line_A_errors            0
line_B_errors            0

Counters
heartbeat (bp)           26195
macro cycles             1635
correct frames           104519
erroneous frames         26131
reply timeouts           26180

Bus
status                   Ok
lines                    redundant, A=1, B=1, active line A

MVB_Devices
address                  status
-----
0002                     5380 BA
0100                     n/a
```

4.6 Console from the Master CPU

A command `shell <module> [command]` is available on the Master CPU that can be used to start a console session over the backplane bus.

The module is a mandatory argument, e.g. MVB1 for the left-most MVB Module in the system.

To run a single command, you can give command directly as argument, e.g.

```
shell MVB1 get_stat
```

will print out the MVB status of the leftmost MVB Module.

5. MVB Signalling

This chapter shows few typical problems on the MVB Bus signals. For taking the measurements, you need a MVB Bus cable breakout connector and an two-channel oscilloscope with Math capability, or a differential probe.

Typically signalling errors are indicated on the L1 or L2 yellow leds on the MVB Module.

5.1 Connecting oscilloscope

If you don't have a differential probe for your oscilloscope, you'll need to configure a Math waveform showing result of subtraction: Channel1 – Channel2.

Good starting point for vertical scale is 2 or 5 V/div, and for horizontal scale 1 μ s of 500 ns/div.

For measurement without differential probe:

- Connect Channel 1 to positive half of MVB pair, pin 1 for Line A or pin 4 for Line B.
- Connect Channel 2 to negative half of MVB pair, pin 2 for Line A or pin 5 for Line B.
- Connect probe ground leads to PE, e.g. to the cable shield or rack chassis.

5.2 Example waveforms

5.2.1 All normal

In normal waveform the differential voltage (M2 channel) is around 4 Vpp, and waveform looks like a square wave with little roundness on the corners.

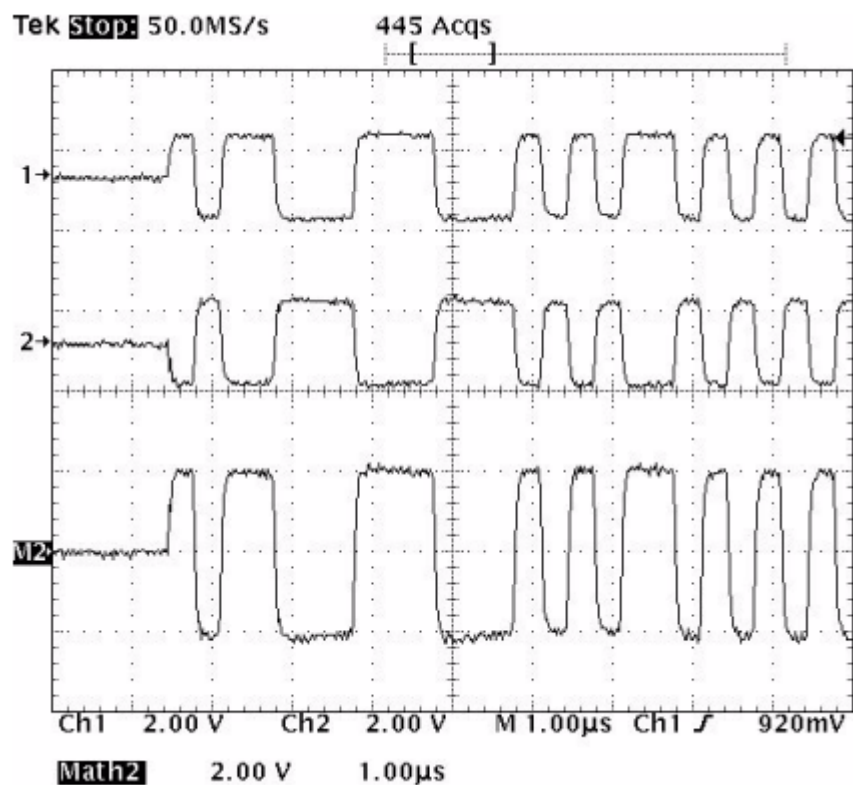


Figure 5.1 Waveform of normal MVB Bus signal

5.2.2 Attenuated signal

This waveform shows attenuated signal, with differential voltage around 300 mVpp. This signal is in the limit of reception.

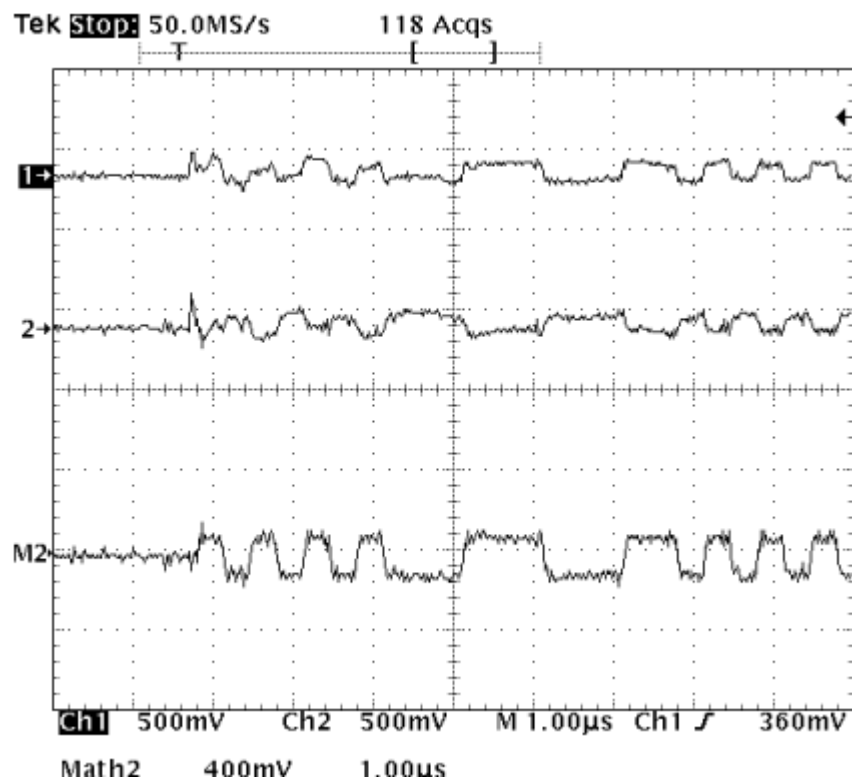


Figure 5.2 Waveform of attenuated MVB Bus signal

5.2.3 Missing half

This waveform shows a good signal, but the other half of the differential pair is missing.

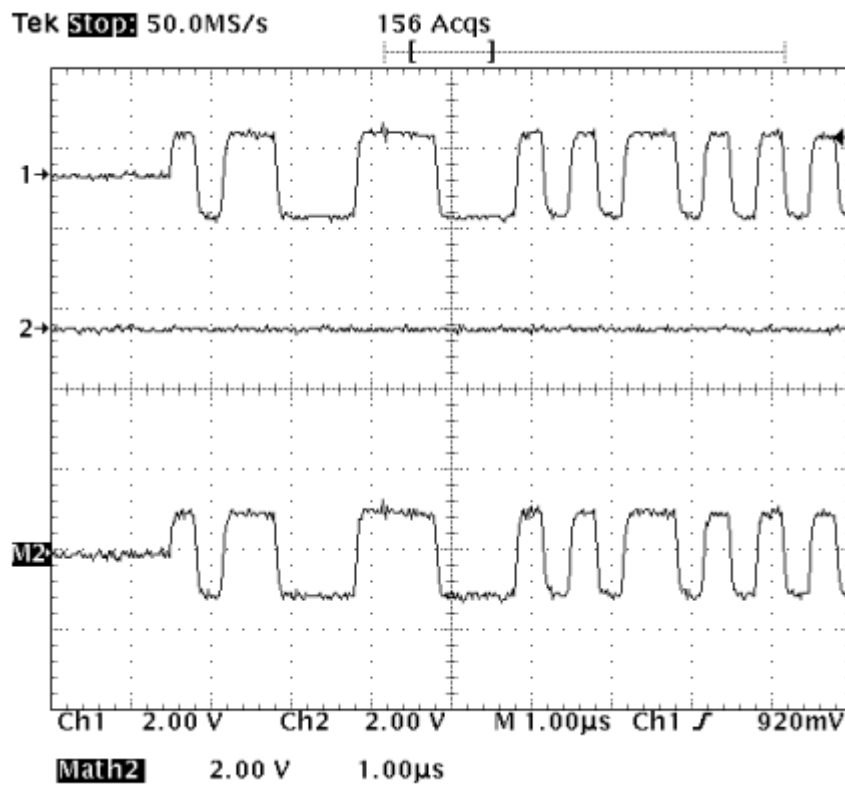


Figure 5.3 Missing half of differential pair

5.2.4 Missing or wrong termination

These waveforms show examples of signals when the MVB Bus termination is missing or the bus is terminated only on one end, or termination resistor is incorrect.

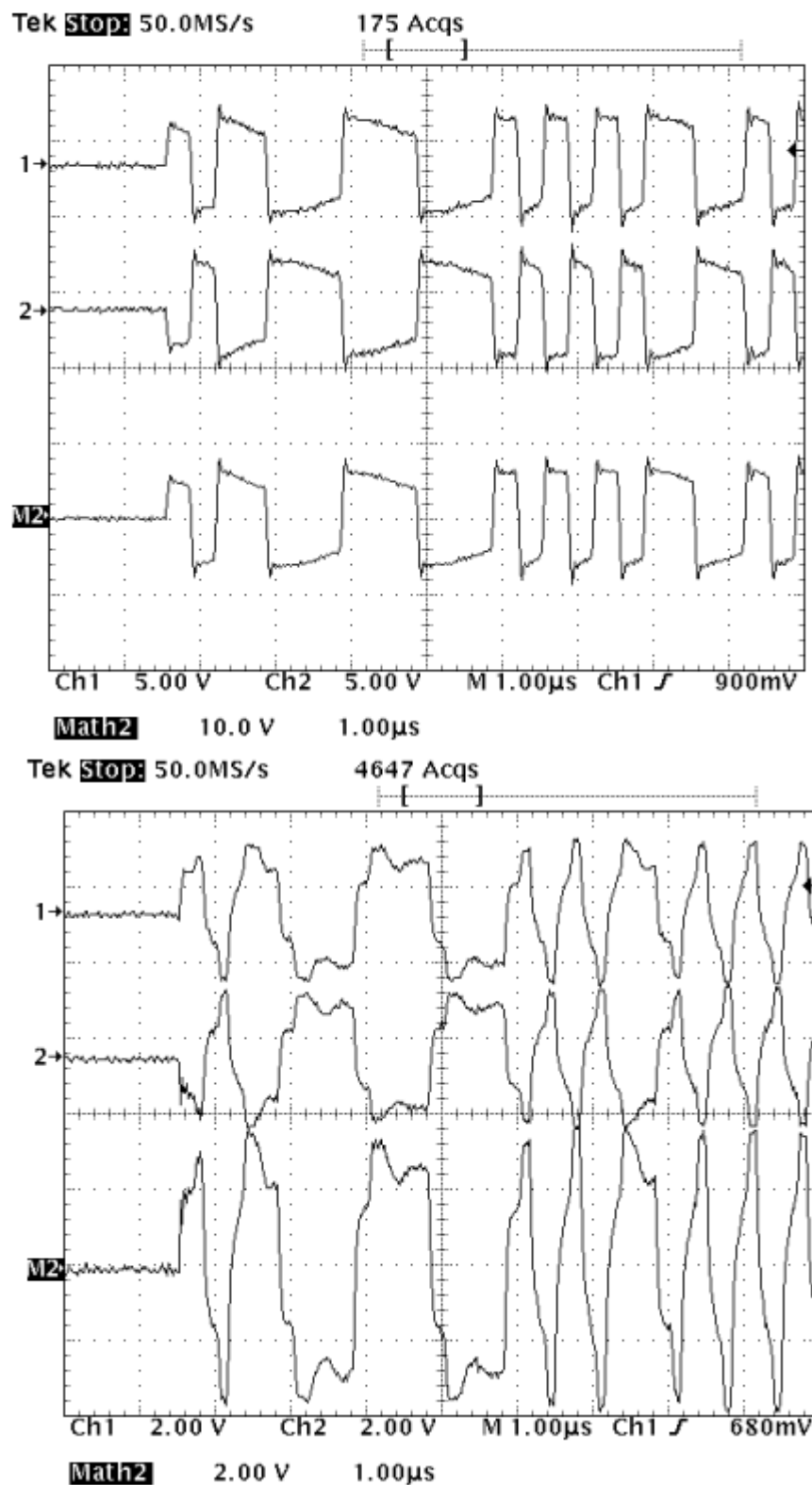


Figure 5.4 Problem with termination

6. Technical data

This chapter includes mechanical, electrical, reliability, safety, environmental and EMC information about the MVB module.

6.1 Dimensions

Table 6.1 Module dimensions

Dimension	Value
Height,	3U (128.5 mm)
Width	4TE (20.32 mm)
Depth	179 mm
Weight	155 g

Free space required in front of the module: 75 mm
(depends on type of connectors used)

6.2 Non-metallic parts

The following table specifies the non-metallic parts used in the MVB module.



The values displayed in this table are for helping to estimate the environmental impact potentially caused by the MVB module in case of fire.

Table 6.2 Non-metallic parts

Material	Amount
Polyester	33.5 g
Acrylics	3.8 g
FR 4, UL94-V0	58 g

6.3 Conformal coating

MVB module has protective surface coating against humidity.

Table 6.3 Conformal coating

Feature	Reference	Properties
Conformal coating	MIL-I-46058C	Acrylics, UL approved

6.4 Front connectors

The following tables specify MVB front connector signals.

Connector M1 is standard 9 pin sub D-type, male. MVB EMD Connector_1

Table 6.4 M1 connector signal pin-out

Pin	Signal	Description
1	A.Data_P	Positive wire, Line_A
2	A.Data_N	Negative wire, Line_A
3		
4	B.Data_P	Positive wire, Line_B
5	B.Data_N	Negative wire, Line_B
6	A.Term_P	Positive pole of Terminator, Line_A
7	A.Term_N	Negative pole of Terminator, Line_A
8	B.Term_P	Positive pole of Terminator, Line_B
9	B.Term_N	Negative pole of Terminator, Line_B
Shell	PE	Connect to cable shield

Connector M2 is standard 9 pin sub D-type, female. MVB EMD Connector_2

Table 6.5 M2 connector signal pin-out

Pin	Signal	Description
1	A.Data_P	Positive wire, Line_A
2	A.Data_N	Negative wire, Line_A
3		
4	B.Data_P	Positive wire, Line_B
5	B.Data_N	Negative wire, Line_B
6	A.Term_P	Positive pole of Terminator, Line_A
7	A.Term_N	Negative pole of Terminator, Line_A
8	B.Term_P	Positive pole of Terminator, Line_B
9	B.Term_N	Negative pole of Terminator, Line_B
Shell	PE	Connect to cable shield

Connector X4

The X4 interface is located on the circuit board and is used for testing and debugging the module and also to download the latest firmware. The connector type is a 2x8 pin male header, 2.54mm spacing.

Table 6.6 X4 connector signal pin-out

Pin	Signal	Description
1	D_CCLK	Download, configuration clock, input
2	GND	Signal ground for RS-232 ports A and B
3	D_FDIN	Download, data in, input
4	RSATX	Transmit data, RS-232 port A
5	D_BFPGADONE	Download, FPGA configuration done, output
6	RSARX	Receive data, RS-232 port A
7	D_CDIN	Download, configuration data, input
8	RSBRX	Transmit data, RS-232 port B
9	D_CDEN~	Download, configuration data enable, input
10	RSBRX	Receive data, RS-232 port B
11	D_FPGAMODE	Download, FPGA configuration mode select, input
12	VCC	Supply output, +5V max 50mA
13	D_DOUT	Download, data out, output
14	GND	Signal ground for download signals
15		Not Connected
16		<No pin> Polarizing key

Table 6.7 Module electrical specifications

Parameter	Specification
Operating voltage	5 VDC (4.75...5.25 VDC)
Operating current	0.45 A
Operating temperature	-40...+70 °C

6.5 Reliability

Table 6.8 Reliability specifications

Parameter	Specification
Reliability, MTBF, greater than	1 700 000 h calculated from component FIT data at +40 °C ambient temperature

6.6 Safety

Table 6.9 Safety

Feature	Reference	Properties
Safety Integrity Level	EN 50128	SIL 0
	EN 50129	SIL 0
	IEC 61508	SIL 0

6.7 Insulation and voltage withstand

Table 6.10 Insulation and voltage withstand

Feature	Reference	Properties
Insulation	EN 50155	500 VDC
Voltage withstand	EN 50155	500 VAC 1 min

6.8 Environmental and EMC test specifications

Table 6.11 Environmental specifications

Parameter	Standard	Test conditions
Operating temperature	EN 50155 class TX	-40... +70 °C
Cooling test	EN 50155:2007 12.2.3 EN 60068-2-1 Ad	2 h at -40 °C, power off
Dry heat test	EN 50155:2007 12.2.4 EN 60068-2-2 Bd	6 h at +70 °C 10 min at +85 °C, power on
Damp heat	EN 50155:2007 12.2.5 HD 323.2.30 Db	+55 °C/+25 °C, 2 cycles, 2x24 h
Vibration	EN 50155:2007 12.2.11 IEC 61373 Category 1 Class B	5...150 Hz, 7.9 m/s ²
Shock	EN 50155:2007 12.2.11 IEC 61373 Category 1 Class B	50 m/s ² for 30 ms

Table 6.12 EMC test specification

Parameter	Standard	Test conditions
Conducted emission	EN 55011	< 99 dBμV quasi-peak at 0.15–0.5 MHz, < 93 dBμV quasi-peak at 0.5–30 MHz
Radiated emission	EN 55011	<40 dBμV/m quasi-peak at 30–230 MHz, 10m < 47 dBμV/m quasi-peak at 230–1000 MHz, 10m
Fast transient burst immunity	EN 61000-4-4	± 2 kV, 5/50 ns, 5 kHz
Surge immunity	EN 61000-4-5	1.2/50 μs, 42 Ohm + 0.5 μF, ± 2 kV line-to-ground, ± 1 kV line-to-line
Radiated RF field immunity	EN 61000-4-3	20 V/m at 80–1000 MHz and 10 V/m at 1.4–2.5 GHz
Conducted RF common mode immunity	EN 61000-4-6	10 Vrms, AM, 1 kHz, unmodulated carrier, 80 % at 0.15–80 MHz
Electrostatic discharge	EN 61000-4-2	± 6 kV contact, ± 8 kV air, which is applicable

6.9 Disposal of the module

In order to reduce the environmental load over the product's entire lifecycle, EKE-Electronics' products are designed to be as safe as possible to manufacture, use, and dispose of



Parts which can be recycled should always be taken to the appropriate processing centres, after hazardous waste has been removed. All parts and components containing hazardous materials must be disposed of in accordance with waste legislation and instructions issued by the environmental authorities. The risks involved and the necessary precautions must be taken into account when handling waste products. Therefore, please execute proper processing of disposal in accordance with local law.

Table 6.13 Disposal of the MVB module, X = action, (X) = action in cases where processing is available

Part	Main materials of disposal	Recyclable materials	Waste disposal sites	Hazardous waste (separate collection)
Frame and covers	Aluminium	X		
Metal	Stainless steel	X		
Plastic	PUR, PU	X		
Rubber	Other plastics		X X	
Component boards		(X)		
	Lead			X
Packing	Cardboard	X		
	PE	X		