TE0725 HyperRAM

Table of contents

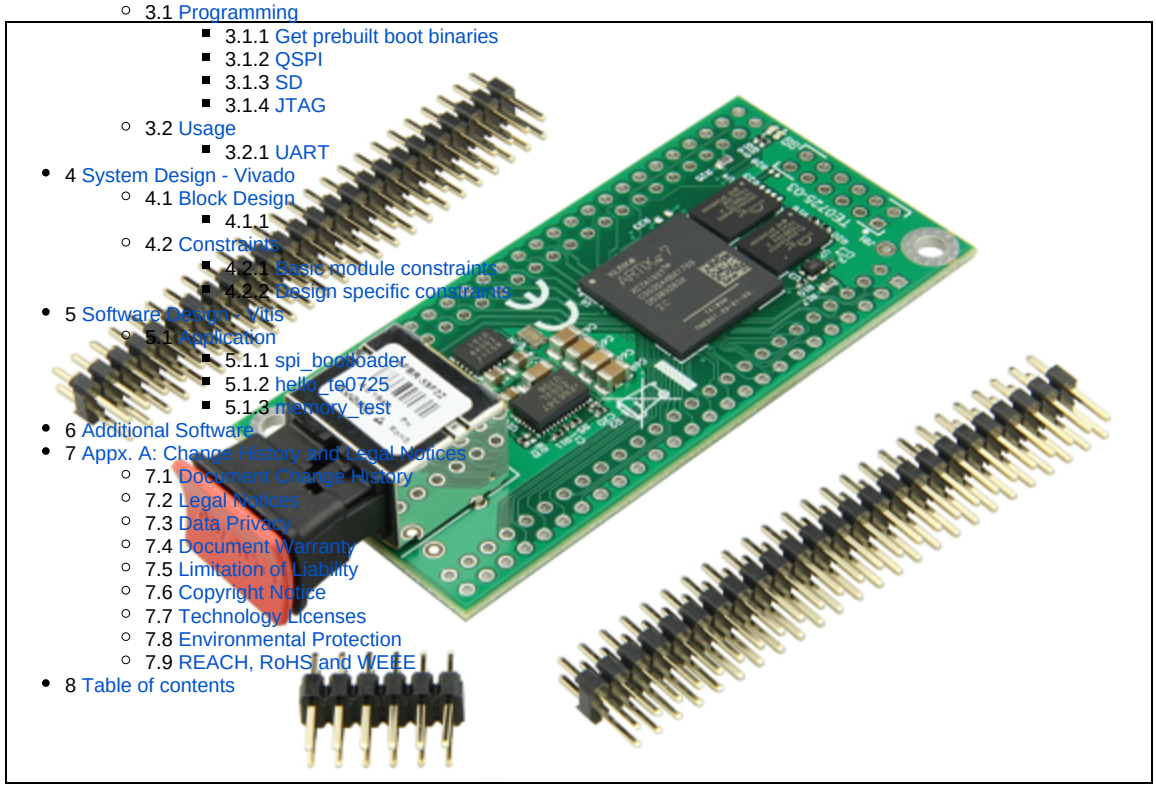
- 1 Overview
  - 1.1 Key Features
  - 1.2 Revision History
  - 1.3 Memory Controller IP Supplied by Synaptic Laboratories Ltd (SLL).
  - 1.4 Requirements
  - 1.4.1 Software
  - 1.4.2 Hardware
  - 1.5 Crowdfunding
    - 1.5.1 Design Solutions
    - 1.5.2 Additional Sources
    - 1.5.3 Prebuilt
    - 1.5.4 Download
- 2 Design Flow
- 3 Launch
  - 3.1 Programming
    - 3.1.1 Get prebuilt boot binaries
    - 3.1.2 QSPI
    - 3.1.3 SD
    - 3.1.4 JTAG
  - 3.2 Usage
    - 3.2.1 UART
- 4 System Design - Vivado
  - 4.1 Block Design
    - 4.1.1
  - 4.2 Constraints
    - 4.2.1 Basic module constraints
    - 4.2.2 Design specific constraints
- 5 Software Design - Vitis
  - 5.1 Application
    - 5.1.1 spi_bootloader
    - 5.1.2 hello_te0725
    - 5.1.3 memory_test
- 6 Additional Software
- 7 Appx. A: Change History and Legal Notices
  - 7.1 Document Change History
  - 7.2 Legal Notices
  - 7.3 Data Privacy
  - 7.4 Document Warranty
  - 7.5 Limitation of Liability
  - 7.6 Copyright Notice
  - 7.7 Technology Licenses
  - 7.8 Environmental Protection
  - 7.9 REACH, RoHS and WEEE
- 8 Table of contents

Overview

Micro Blaze Design with 10 minutes HyperRAM memory test example.

This reference design is bundled with a FREE evaluation edition of the low-cost, commercially proven, high performance memory controller IP supplied by Synaptic Laboratories Ltd (SLL). This free IP evaluation license never expires, and no customer registration or NIC ID is required. Click here to find the latest free trials of SLL's memory controller IP for HyperBus, OctaBus, Xccela Bus, JEDEC xSPI Profile 1.0 and JEDEC xSPI Profile 2.0 for Intel, Microchip, and Xilinx FPGA. SLL IP is also qualified for use with Trenz HS Crowdfunded boards. Please send all sales enquiry and technical support questions for SLL's IP to 1.5.1 Design Solutions.

Refer to http://trenz.org/te0725-info for the current online version of this manual and other available documentation.



Key Features

- Vivado/Vitis 2021.2
- MicroBlaze
- QSPI
- I2C
- UART
- HyperRAM
- S/Labs HBMC IP (Free Trail IP)

Revision History

Date	Vivado	Project Built	Authors	Description
2022-08-30	2021.2	TE0725-HyperRAM_noprebuilt-vivado_2021.2-build_15_20220830154134.zip TE0725-HyperRAM-vivado_2021.2-build_15_20220830154134.zip	Waldemar Hanemann	• 2021.2 update • new spi_bootloader to load elf file from qspi to hyperram • Documentation style update
2020-04-29	2019.2	TE0725-HyperRAM_noprebuilt-vivado_2019.2-build_10_20200429134457.zip TE0725-HyperRAM-vivado_2019.2-build_10_20200429134447.zip	John Hartfiel	• add srec application which loads hello_te0725 from qspi into hyperram
2020-04-17	2019.2	TE0725-HyperRAM-vivado_2019.2-build_10_20200427163950.zip TE0725-HyperRAM_noprebuilt-vivado_2019.2-build_10_20200427163959.zip	John Hartfiel	• 2019.2 update
2018-08-09	2018.2	TE0725-HyperRAM_noprebuilt-vivado_2018.2-build_02_20180809122634.zip TE0725-HyperRAM-vivado_2018.2-build_02_20180809122623.zip	John Hartfiel	• 2018.2 update • new HBMC IP version (v1_3_57)
2018-06-05	2017.4	TE0725-HyperRAM_noprebuilt-vivado_2017.4-build_10_20180605162539.zip TE0725-HyperRAM-vivado_2017.4-build_10_20180605162425.zip	John Hartfiel	• initial release

Design Revision History

Release Notes and Know Issues

Issues	Description	Workaround	To be fixed version
No known issues	---	---	---

Known Issues

Requirements

Software

Software	Version	Note
Vitis	2021.2	needed, Vivado is included into Vitis installation

Software

Hardware

Basic description of TE Board Part Files is available on TE Board Part Files.

Complete List is available on <design name>/board_files/*_board_files.csv

Design supports following modules:

Module Model	Board Part Short Name	PCB Revision Support	DDR	QSPI Flash	EMMC	Others	Notes
TE0725-03-15-1C*	15_1c	REV03|REV02|REV01	NA	32MB	NA	8MB HypeRAM	NA
TE0725-03-35-2C	35_2c	REV03|REV02|REV01	NA	32MB	NA	8MB HypeRAM	NA
TE0725-03-100-2C	100_2c	REV03|REV02|REV01	NA	32MB	NA	8MB HypeRAM	NA
TE0725-03-100-2CF	100_2c	REV03|REV02|REV01	NA	32MB	NA	8MB HypeRAM	POF assembled
TE0725-03-100-2I9	100_2i	REV03|REV02|REV01	NA	32MB	NA	8MB HypeRAM	NA
TE0725-03-35-2I	35_2i	REV03|REV02|REV01	NA	32MB	NA	8MB HypeRAM	NA

*used as reference

Hardware Modules

Design supports following carriers:

Carrier Model	Notes

Hardware Carrier

Additional HW Requirements:

Additional Hardware	Notes
TE0790 JTAG Programmer	It's not recommended to use TE0790 for power supply(TE0790 TRM#PowerandPower-OnSequence)
External power supply	

Additional Hardware

Content

For general structure and of the reference design, see Project Delivery - AMD devices

Design Sources

Type	Location	Notes
Vivado	<project folder>\block_design <project folder>\constraints <project folder>\ip_lib <project folder>\board_files	Vivado Project will be generated by TE Scripts
Vitis	<project folder>\sw_lib	Additional Software Template for Vitis and apps_list.csv with settings automatically for Vitis app generation

Design sources

Additional Sources

Type	Location	Notes
--	--	--

Additional design sources

Prebuilt

File	File-Extension	Description
BIT-File	*.bit	FPGA (PL Part) Configuration File
DebugProbes-File	*.ltx	Definition File for Vivado/Vivado Labtools Debugging Interface
Diverse Reports	---	Report files in different formats
Hardware-Platform-Specification-Files	*.xsa	Exported Vivado Hardware Specification for Vitis and PetaLinux
LabTools Project-File	*.lpr	Vivado Labtools Project File
MCS-File	*.mcs	Flash Configuration File with Boot-Image (MicroBlaze or FPGA part only)
MMI-File	*.mmi	File with BRAM-Location to generate MCS or BIT-File with *.elf content (MicroBlaze only)
Software-Application-File	*.elf	Software Application for Zynq or MicroBlaze Processor Systems
SREC-File	*.srec	Converted Software Application for MicroBlaze Processor Systems

Prebuilt files (only on ZIP with prebult content)

Download

Reference Design is only usable with the specified Vivado/Vitis/PetaLinux version. Do never use different Versions of Xilinx Software for the same Project.

Reference Design is available on:

- TE0725 "HyperRAM" Reference Design

Design Flow

Reference Design is available with and without prebuilt files. It's recommended to use TE prebuilt files for first lunch.

Trenz Electronic provides a tcl based built environment based on Xilinx Design Flow.

See also:

- AMD Development Tools#XilinxSoftware-BasicUserGuides
- Vivado Projects - TE Reference Design
- Project Delivery.

The Trenz Electronic FPGA Reference Designs are TCL-script based project. Command files for execution will be generated with "_create_win_setup.cmd" on Windows OS and "_create_linux_setup.sh" on Linux OS.

TE Scripts are only needed to generate the vivado project, all other additional steps are optional and can also executed by Xilinx Vivado/Vitis GUI. For currently Scripts limitations on Win and Linux OS see: Project Delivery Currently limitations of functionality

Caution! Win OS has a 260 character limit for path lengths which can affect the Vivado tools. To avoid this issue, use Virtual Drive or the shortest possible names and directory locations for the reference design (for example "x:\<project folder>")

1. Run _create_win_setup.cmd/_create_linux_setup.sh and follow instructions on shell:

_create_win_setup.cmd/_create_linux_setup.sh

```
-----------------------Set design paths--------------------------
-- Run Design with: _create_win_setup
-- Use Design Path: <absolute project path>
-----------------------------------------------------------------
-----------------------TE Reference
Design-------------------------
-----------------------------------------------------------------
-- (0)  Module selection guide, project creation...prebuilt export...
-- (1)  Create minimum setup of CMD-Files and exit Batch
-- (2)  Create maximum setup of CMD-Files and exit Batch
-- (3)  (internal only) Dev
-- (4)  (internal only) Prod
-- (c)  Go to CMD-File Generation (Manual setup)
-- (d)  Go to Documentation (Web Documentation)
-- (g)  Install Board Files from Xilinx Board Store (beta)
-- (a)  Start design with unsupported Vivado Version (beta)
-- (x)  Exit Batch (nothing is done!)
----
Select (ex.:'0' for module selection guide):
```

2. Press 0 and enter to start "Module Selection Guide"
3. Create project and follow instructions of the product selection guide, settings file will be configured automatically during this process.
    - optional for manual changes: Select correct device and Xilinx install path on "design_basic_settings.cmd" and create Vivado project with "vivado_create_project_guimode.cmd"

Note: Select correct one, see also Vivado Board Part Flow

a. Create hardware description file (.xsa file) for PetaLinux project and export to prebuilt folder

run on Vivado TCL (Script generates design and export files into "<project folder>\prebuilt\hardware\<short name>")

```
TE::hw_build_design -export_prebuilt
```

Using Vivado GUI is the same, except file export to prebuilt folder.

4. Generate Programming Files with Vitis

run on Vivado TCL (Script generates applications and bootable files, which are defined in "test_board\sw_lib\apps_list.csv")

```
TE::sw_run_vitis -all
TE::sw_run_vitis (optional; Start Vitis from Vivado GUI or start
with TE Scripts on Vivado TCL)
```

Note: Scripts generate applications and bootable files, which are defined in "sw_lib\apps_list.csv"

App from Firmware folder will be add into BlockRAM. If you add other app, you must select *.elf

manually on Vivado

TCL scripts generate also platform project, this must be done manually in case GUI is used. See Vitis

5. (optional) Copy Application (spi_bootloader.elf) from prebuilt-folder into \firmware\microblaze_0\ and regenerate design with

run on Vivado TCL (Script generates design and export files into "<project folder>\prebuilt\hardware\<short name>")
`TE::hw_build_design -export_prebuilt`

Launch

Programming

Check Module and Carrier TRMs for proper HW configuration before you try any design.

Reference Design is also available with prebuilt files. It's recommended to use TE prebuilt files for first launch.

Xilinx documentation for programming and debugging: Vivado/SDK/SDSoC-Xilinx Software Programming and Debugging

Get prebuilt boot binaries

1. Run _create_win_setup.cmd/_create_linux_setup.sh and follow instructions on shell
2. Press 0 and enter to start "Module Selection Guide"
    a. Select assembly version
    b. Validate selection
    c. Select create and open delivery binary folder

Note: Folder "<project folder>_binaries_<Article Name>" with subfolder "boot_<app name>" for different applications will be generated

QSPI

1. Connect JTAG and power on carrier with module
2. Open Vivado Project with "vivado_open_existing_project_guimode.cmd" or if not created, create with "vivado_create_project_guimode.cmd"

run on Vivado TCL (Script programs .mcs-File on QSPI flash)
`TE::pr_program_flash -swapp hello_te0725`

3. Press the reset button to start the application and see the output in the console

SD

Not used on this Example.

JTAG

1. Connect JTAG and power on PCB
2. Open Vivado HW Manager
3. Program FPGA with Bitfile from "prebuilt\hardware\<short dir>\"

Usage

HBMC IP is a 10 minute run-time limited evaluation version of the full-edition

1. Prepare HW like described on section Programming
2. Connect UART USB (most cases same as JTAG)
3. 1. FPGA Loads Bitfile from Flash

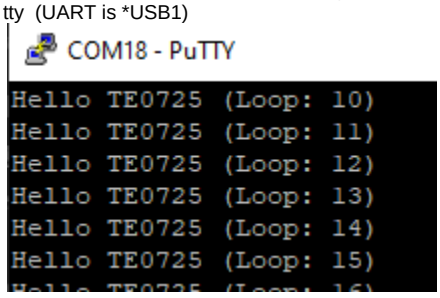
   3. Hello Trenz will be run on UART console.

   info: Do not reboot, if Bitfile programming over JTAG is used as programming method.

   a. ## UART

      Open Serial Console (e.g. putty) Hello TE0725 will run on endless loop.

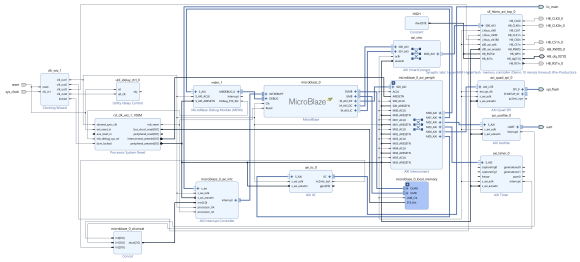
      i. Speed: 9600
      ii. COM Port: Win OS, see device manager, Linux OS see dmesg |grep tty (UART is *USB1)



      Power On PCB (Do not restart, if you use Bitfile programming)

System Design - Vivado

Block Design



Constraints

Basic module constraints

_i_bitgen_common.xdc

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 50 [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR YES [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.CONFIG.M1PIN PULLNONE [current_design]
set_property BITSTREAM.CONFIG.M2PIN PULLNONE [current_design]
set_property BITSTREAM.CONFIG.M0PIN PULLNONE [current_design]

set_property BITSTREAM.CONFIG.USR_ACCESS TIMESTAMP [current_design]
```

Design specific constraints

_i_hyperram.xdc

```
set_property PACKAGE_PIN A13 [get_ports HB_CLK0_0]
set_property PACKAGE_PIN A14 [get_ports HB_CLK0n_0]

set_property PACKAGE_PIN E17 [get_ports {HB_dq_0[0]}]
set_property PACKAGE_PIN B17 [get_ports {HB_dq_0[1]}]
set_property PACKAGE_PIN F18 [get_ports {HB_dq_0[2]}]
set_property PACKAGE_PIN F16 [get_ports {HB_dq_0[3]}]
set_property PACKAGE_PIN G17 [get_ports {HB_dq_0[4]}]
set_property PACKAGE_PIN D18 [get_ports {HB_dq_0[5]}]
set_property PACKAGE_PIN B18 [get_ports {HB_dq_0[6]}]
set_property PACKAGE_PIN A16 [get_ports {HB_dq_0[7]}]


set_property PACKAGE_PIN E18 [get_ports HB_RWDS_0]
```

```
set_property PACKAGE_PIN D17 [get_ports HB_CS1n_0]
set_property PACKAGE_PIN J17 [get_ports HB_RSTn_0]

#set_property PACKAGE_PIN A18 [get_ports HB_CS0n_0 ]
#set_property PACKAGE_PIN J18 [get_ports HB_INTn_0 ]
#set_property PACKAGE_PIN C17 [get_ports HB_RSTOn_0]


#
# FPGA Pin Voltage assignment
#
set_property IOSTANDARD LVCMOS18 [get_ports HB_CLK0_0]
set_property IOSTANDARD LVCMOS18 [get_ports HB_CLK0n_0]
set_property IOSTANDARD LVCMOS18 [get_ports {HB_dq_0[*]}]
set_property IOSTANDARD LVCMOS18 [get_ports HB_CS1n_0]
set_property IOSTANDARD LVCMOS18 [get_ports HB_RSTn_0]
set_property IOSTANDARD LVCMOS18 [get_ports HB_RWDS_0]

#set_property IOSTANDARD LVCMOS18 [get_ports HB_CS0n_0]
#set_property IOSTANDARD LVCMOS18 [get_ports HB_INTn_0]
#set_property IOSTANDARD LVCMOS18 [get_ports HB_RSTOn_0]

#set_property PULLUP true [get_ports HB_RSTOn_0]
#set_property PULLUP true [get_ports HB_INTn_0]

#
#Hyperbus Clock - change according to clk pin on PLL
#
create_generated_clock -name clk_0   -source [get_pins msys_i/clk_wiz_0
/inst/mmcm_adv_inst/CLKIN1] -master_clock sys_clock [get_pins msys_i
/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT0]
create_generated_clock -name clk_90  -source [get_pins msys_i/clk_wiz_0
/inst/mmcm_adv_inst/CLKIN1] -master_clock sys_clock [get_pins msys_i
/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT1]
create_generated_clock -name clk_180 -source [get_pins msys_i/clk_wiz_0
/inst/mmcm_adv_inst/CLKIN1] -master_clock sys_clock [get_pins msys_i
/clk_wiz_0/inst/mmcm_adv_inst/CLKOUT2]

#
#100Mhz clock freqeuncy - change accordingly
#
set hbus_freq_ns   10

set dqs_in_min_dly -0.5
set dqs_in_max_dly  0.5

set HB_dq_ports    [get_ports HB_dq_0[*]]

#
#Create RDS clock and RDS virtual clock
#
create_clock -period $hbus_freq_ns -name rwds_clk      [get_ports
HB_RWDS_0]
create_clock -period $hbus_freq_ns -name virt_rwds_clk

#
#Input Delay Constraint -  HB_RWDS-HB_DQ
#
set_input_delay -clock [get_clocks virt_rwds_clk]             -max
${dqs_in_max_dly} ${HB_dq_ports}
set_input_delay -clock [get_clocks virt_rwds_clk] -clock_fall -max
```

```
${dqs_in_max_dly} ${HB_dq_ports} -add_delay

set_input_delay -clock [get_clocks virt_rwds_clk]             -min
${dqs_in_min_dly} ${HB_dq_ports} -add_delay
set_input_delay -clock [get_clocks virt_rwds_clk] -clock_fall -min
${dqs_in_min_dly} ${HB_dq_ports} -add_delay

set_multicycle_path -setup -end -rise_from [get_clocks virt_rwds_clk] -
rise_to [get_clocks rwds_clk] 0
set_multicycle_path -setup -end -fall_from [get_clocks virt_rwds_clk] -
fall_to [get_clocks rwds_clk] 0

set_false_path  -fall_from [get_clocks virt_rwds_clk] -rise_to [get_clocks
rwds_clk] -setup
set_false_path  -rise_from [get_clocks virt_rwds_clk] -fall_to [get_clocks
rwds_clk] -setup
set_false_path  -fall_from [get_clocks virt_rwds_clk] -fall_to [get_clocks
rwds_clk] -hold
set_false_path  -rise_from [get_clocks virt_rwds_clk] -rise_to [get_clocks
rwds_clk] -hold

set_false_path -from [get_clocks clk_0] -to [get_clocks rwds_clk]
set_false_path -from [get_clocks rwds_clk] -to [get_clocks clk_0]

#
#Output Delay Constraint -  HB_CLK0-HB_DQ
#

create_generated_clock -name HB_CLK0_0 -source [get_pins */*/*/U_IO/U_CLK0
/dq_idx_[0].ODDR_inst/C] -multiply_by 1 -invert [get_ports HB_CLK0_0]

set_output_delay -clock [get_clocks HB_CLK0_0] -min -1.000 ${HB_dq_ports}
set_output_delay -clock [get_clocks HB_CLK0_0] -max  1.000 ${HB_dq_ports}
set_output_delay -clock [get_clocks HB_CLK0_0] -min -1.000 ${HB_dq_ports} -
clock_fall -add_delay
set_output_delay -clock [get_clocks HB_CLK0_0] -max  1.000 ${HB_dq_ports} -
clock_fall -add_delay


set_false_path -from [get_pins */*/*/U_HBC/*/dq_io_tri_reg/C] -to
${HB_dq_ports}

set_false_path -from * -to [get_pins */*/inst/*/i_iavs0_270_rstn_1_reg/CLR]
set_false_path -from * -to [get_pins */*/inst/*/i_iavs0_270_rstn_2_reg/CLR]
set_false_path -from * -to [get_pins */*/inst/*/i_iavs0_270_rstn_3_reg/CLR]


set_false_path -from [get_clocks rwds_clk] -to [get_clocks -of_objects
[get_pins msys_i/clk_wiz_1/inst/mmcm_adv_inst/CLKOUT0]]
set_false_path -from [get_clocks virt_rwds_clk] -to [get_clocks rwds_clk]
```

Software Design - Vitis

For SDK project creation, follow instructions from:

Vitis

Application

Template location: ./sw_lib/sw_apps/

spi_bootloader

TE modified SPI Bootloader from Henrik Brix Andersen.

Bootloader to load app or second bootloader from flash into DDR.

Here it loads the hello_te0725.elf from QSPI-Flash to RAM. Hence *.srec becomes redundant.

Descriptions:

- Modified Files: bootloader.c
- Changes:
  - Change the SPI defines in the header
  - Add some reiteration in the frist spi read call

hello_te0725

Hello TE0725 is a Xilinx Hello World example as endless loop instead of one single console output.

memory_test

Xilinx default memory test.

Additional Software

No additional software is needed.

Appx. A: Change History and Legal Notices

Document Change History

To get content of older revision got to "Change History" of this page and select older document revision number.

Date	Document Revision	Authors	Description
			• 2021.2 update • new spi_bootloader to load elf file from qspi to hyperram • Documentation style update

Error rendering macro 'page-info' Ambiguous method overloading for method jdk.proxy24.$Proxy3589#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page]	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy24.$Proxy3589#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page]	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy24.$Proxy3589#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page]

due to overlapping prototypes between: [interface com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.

due to overlapping prototypes between: [interface com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.

due to overlapping prototypes between: [interface com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.

atlassian . confluence.core. ContentEntityObject]	atlassian . confluence.core. ContentEntityObject]	atlassian . confluence.core. ContentEntityObject]	
2021-07-17	v.9	John Hartfiel	• update block design image
2021-07-06	v.8	John Hartfiel	• new overview description
2020-04-29	v.7	John Hartfiel	• Design SW update with SREC Bootloader
2020-04-27	v.5	John Hartfiel	• 2019.2 update • Documentation style update
2018-08-09	v.4	John Hartfiel	• 2018.2 update
2018-06-06	v.3	John Hartfiel	• Documentation update
2018-06-05	v.2	John Hartfiel	• 2017.4 release
	All	**Error rendering macro 'page-info'** Ambiguous method overload	

ing for method jdk.proxy244.$Proxy3589#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interface com.atlassian

		. confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	

Document change history.

Legal Notices

Data Privacy

Please also note our data protection declaration at https://www.trenz-electronic.de/en/Data-protection-Privacy

Document Warranty

The material contained in this document is provided “as is” and is subject to being changed at any time without notice. Trenz Electronic does not warrant the accuracy and completeness of the materials in this document. Further, to the maximum extent permitted by applicable law, Trenz Electronic disclaims all warranties, either express or implied, with regard to this document and any information contained herein, including but not limited to the implied warranties of merchantability, fitness for a particular purpose or non infringement of intellectual property. Trenz Electronic shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein.

Limitation of Liability

In no event will Trenz Electronic, its suppliers, or other third parties mentioned in this document be liable for any damages whatsoever (including, without limitation, those resulting from lost profits, lost data or business interruption) arising out of the use, inability to use, or the results of use of this document, any documents linked to this document, or the materials or information contained at any or all such documents. If your use of the materials or information from this document results in the need for servicing, repair or correction of equipment or data, you assume all costs thereof.

Copyright Notice

No part of this manual may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Trenz Electronic.

Technology Licenses

The hardware / firmware / software described in this document are furnished under a license and may be used /modified / copied only in accordance with the terms of such license.

Environmental Protection

To confront directly with the responsibility toward the environment, the global community and eventually also oneself. Such a resolution should be integral part not only of everybody's life. Also enterprises shall be conscious of their social responsibility and contribute to the preservation of our common living space. That is why Trenz Electronic invests in the protection of our Environment.

REACH, RoHS and WEEE

REACH

Trenz Electronic is a manufacturer and a distributor of electronic products. It is therefore a so called downstream user in the sense of REACH. The products we supply to you are solely non-chemical products (goods). Moreover and under normal and reasonably foreseeable circumstances of application, the goods supplied to you shall not release any substance. For that, Trenz Electronic is obliged to neither register nor to provide safety data sheet. According to present knowledge and to best of our knowledge, no SVHC (Substances of Very High Concern) on the Candidate List are contained in our products. Furthermore, we will immediately and unsolicited inform our customers in compliance with REACH - Article 33 if any substance present in our goods (above a concentration of 0,1 % weight by weight) will be classified as SVHC by the European Chemicals Agency (ECHA).

RoHS

Trenz Electronic GmbH herewith declares that all its products are developed, manufactured and distributed RoHS compliant.

WEEE

Information for users within the European Union in accordance with Directive 2002/96/EC of the European Parliament and of the Council of 27 January 2003 on waste electrical and electronic equipment (WEEE).

Users of electrical and electronic equipment in private households are required not to dispose of waste electrical and electronic equipment as unsorted municipal waste and to collect such waste electrical and electronic equipment separately. By the 13 August 2005, Member States shall have ensured that systems are set up allowing final holders and distributors to return waste electrical and electronic equipment at least free of charge. Member States shall ensure the availability and accessibility of the necessary collection facilities. Separate collection is the precondition to ensure specific treatment and recycling of waste electrical and electronic equipment and is necessary to achieve the chosen level of protection of human health and the environment in the European Union. Consumers have to actively contribute to the success of such collection and the return of waste electrical and electronic equipment. Presence of hazardous substances in electrical and electronic equipment results in potential effects on the environment and human health. The symbol consisting of the crossed-out wheeled bin indicates separate collection for waste electrical and electronic equipment.

Trenz Electronic is registered under WEEE-Reg.-Nr. DE97922676.

Error rendering macro 'page-info'

Ambiguous method overloading for method jdk.proxy244.$Proxy3589#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interface com.atlassian.confluence.user.ConfluenceUser, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject] [interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]