TE0728 Test Board

Table of contents

Table of contents

- 1 Table of contents
- 2 Overview
  - 2.1 Key Features
  - 2.2 Revision History
  - 2.3 Release Notes and Know Issues
  - 2.4 Requirements
    - 2.4.1 Software
    - 2.4.2 Hardware
  - 2.5 Content
    - 2.5.1 Design Sources
    - 2.5.2 Additional Sources
    - 2.5.3 Prebuilt
    - 2.5.4 Download
- 3 Design Flow
- 4 Launch
  - 4.1 Programming
    - 4.1.1 Get prebuilt boot binaries
    - 4.1.2 QSPI-Boot mode
    - 4.1.3 SD-Boot mode
    - 4.1.4 JTAG
  - 4.2 Usage
    - 4.2.1 Linux
- 5 System Design - Vivado
  - 5.1 Block Design
    - 5.1.1 PS Interfaces
  - 5.2 Constrains
    - 5.2.1 Basic module constrains
    - 5.2.2 Design specific constrain
- 6 Software Design - Vitis
  - 6.1 Application
    - 6.1.1 zynq_fsbl
    - 6.1.2 zynq_fsbl_flash
    - 6.1.3 hello_te0728
    - 6.1.4 u-boot
- 7 Software Design - PetaLinux
  - 7.1 Config
  - 7.2 U-Boot
  - 7.3 Device Tree
  - 7.4 FSBL patch
  - 7.5 Kernel
  - 7.6 Rootfs
  - 7.7 Applications
    - 7.7.1 startup
    - 7.7.2 webfwu
- 8 Additional Software
- 9 Appx. A: Change History and Legal Notices
  - 9.1 Document Change History
  - 9.2 Legal Notices
  - 9.3 Data Privacy
  - 9.4 Document Warranty
  - 9.5 Limitation of Liability
  - 9.6 Copyright Notice
  - 9.7 Technology Licenses
  - 9.8 Environmental Protection
  - 9.9 REACH, RoHS and WEEE
- 10 Table of contents

- 1 Table of contents
- 2 Overview
  - 2.1 Key Features
  - 2.2 Revision History
  - 2.3 Release Notes and Know Issues
  - 2.4 Requirements
    - 2.4.1 Software
    - 2.4.2 Hardware
  - 2.5 Content
    - 2.5.1 Design Sources
    - 2.5.2 Additional Sources
    - 2.5.3 Prebuilt
    - 2.5.4 Download
- 3 Design Flow
- 4 Launch
  - 4.1 Programming
    - 4.1.1 Get prebuilt boot binaries
    - 4.1.2 QSPI-Boot mode
    - 4.1.3 SD-Boot mode
    - 4.1.4 JTAG
  - 4.2 Usage
    - 4.2.1 Linux
- 5 System Design - Vivado
  - 5.1 Block Design
    - 5.1.1 PS Interfaces
  - 5.2 Constrains
    - 5.2.1 Basic module constrains
    - 5.2.2 Design specific constrain
- 6 Software Design - Vitis
  - 6.1 Application
    - 6.1.1 zynq_fsbl
    - 6.1.2 zynq_fsbl_flash
    - 6.1.3 hello_te0728
    - 6.1.4 u-boot
- 7 Software Design - PetaLinux
  - 7.1 Config
  - 7.2 U-Boot
  - 7.3 Device Tree
  - 7.4 FSBL patch
  - 7.5 Kernel
  - 7.6 Rootfs
  - 7.7 Applications
    - 7.7.1 startup
    - 7.7.2 webfwu
- 8 Additional Software
- 9 Appx. A: Change History and Legal Notices
  - 9.1 Document Change History
  - 9.2 Legal Notices
  - 9.3 Data Privacy
  - 9.4 Document Warranty
  - 9.5 Limitation of Liability
  - 9.6 Copyright Notice
  - 9.7 Technology Licenses
  - 9.8 Environmental Protection
  - 9.9 REACH, RoHS and WEEE
- 10 Table of contents

Overview

Zynq Design PS with Linux and two Ethernet PHYs connected over EMIO and PL.

Refer to http://trenz.org/te0728-info for the current online version of this manual and other available documentation.

Key Features

- Vitis/Vivado 2020.2
- PetaLinux
- SD
- 2x ETH (Independent MDIO Interface and DP83848 PHY)
- I2C
- RTC
- Special FSBL for QSPI programming

Revision History

Date	Vivado	Project Built	Authors	Description
2021-11-03	2020.2	TE0728-test_board-vivado_2020.2-build_8_20211103093707.zip TE0728-test_board_noprebuilt-vivado_2020.2-build_8_20211103093732.zip	Mohsen Chamanbaz	• 2020.2 release
2018-12-12	2018.2	TE0728-test_board-vivado_2018.2-build_03_20181212131950.zip TE0728-test_board_noprebuilt-vivado_2018.2-build_03_20181212134902.zip	John Hartfiel	• rework board part files • rework petalinux device tree, driver • small changes on xdc
2017-10-06	2017.2	TE0728-test_board_noprebuilt-vivado_2017.2-build_03_20171006103655.zip TE0728-test_board-vivado_2017.2-build_03_20171006103634.zip	John Hartfiel	• initial release

Design Revision History

Release Notes and Know Issues

Issues	Description	Workaround	To be fixed version
Wrong UBoot ETH PHY Address	PHY Address is not set correctly for UBoot	---	solved with 2018-12-12 update

Linux Message: "macb ... . ethernet eth...: unable to generate target frequency: 25000000 Hz"	This can be ignored, ETH works.	---	---

Known Issues

Requirements

Software

Software	Version	Note
Vitis	2020.2	needed, Vivado is included into Vitis installation
Petalinux	2020.2	needed

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	Others	
TE0728-03-1Q	03_1q	REV01, REV02, REV03	512MB	16MB		
TE0728-04-1Q*	04_1q	REV04	512MB	16MB		

*used as reference

Hardware Modules

Design supports following carriers:

Carrier Model	Notes
TEB0728	

Hardware Carrier

Additional HW Requirements:

Additional Hardware	Notes
USB Cable for JTAG/UART	
TE0790 XMOD Programmer	

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 folde>\sw_lib	Additional Software Template for Vitis and apps_list.csv with settings automatically for Vitis app generation
PetaLinux	<project folder>\os\petalinux	PetaLinux template with current configuration

Design sources

Additional Sources

Type	Location	Notes
init.sh	<project folder>\misc\sd\	Additional Initialization Script for Linux

Additional design sources

Prebuilt

File	File-Extension	Description
BIF-File	*.bif	File with description to generate Bin-File
BIN-File	*.bin	Flash Configuration File with Boot-Image (Zynq-FPGAs)
BIT-File	*.bit	FPGA (PL Part) Configuration File
Diverse Reports	---	Report files in different formats
Hardware-Platform-Description-File	*.xsa	Exported Vivado hardware description file for Vitis and PetaLinux
LabTools Project-File	*.lpr	Vivado Labtools Project File
OS-Image	*.ub	Image with Linux Kernel (On Petalinux optional with Devicetree and RAM-Disk)
Software-Application-File	*.elf	Software Application for Zynq or 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:

- TE0728 "Test Board" Reference Design

Design Flow

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

MIO Bank 501 Power is Carrier depends and set to 3.3V. Please check Settings, if you use a own carrier.

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

4. Create Project
    a. 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 TE Board Part Files
5. 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.

6. Create and configure your PetaLinux project with exported .xsa-file, see PetaLinux KICKstart
    - use TE Template from "<project folder>\os\petalinux"
    - use exported .xsa file from "<project folder>\prebuilt\hardware\<short name>" . **Note:** HW Export from Vivado GUI creates another path as default workspace.
    - The build images are located in the "<plnx-proj-root>/images/linux" directory
7. Configure the **boot.scr** file as needed, see Distro Boot with Boot.scr
8. Copy PetaLinux build image files to prebuilt folder
    - copy **u-boot.elf**, **image.ub** and **boot.scr** from "<plnx-proj-root>/images/linux" to prebuilt folder

"<project folder>\prebuilt\os\petalinux\<ddr size>" or "<project folder>\prebuilt\os\petalinux\<short name>"

9. 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)
```

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

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-Boot mode

Option for **Boot.bin** on QSPI Flash and **image.ub** and **boot.scr** on **SD** or **USB**.

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 BOOT.bin on QSPI flash)

```
TE::pr_program_flash -swapp u-boot
TE::pr_program_flash -swapp hello_te0820 (optional)
```


To program with Vitis/Vivado GUI, use special FSBL (fsbl_flash) on setup

3. Copy **image.ub** and **boot.scr** on **SD** or **USB**
    - use files from "<project folder>_binaries_<Article Name>\boot_linux" from generated binary folder,see: Get prebuilt boot binaries
    - or use prebuilt file location, see "<project folder>\prebuilt\file_location.txt"
4. Set Boot Mode to **QSPI-Boot** and insert **SD** or **USB**.
    - Depends on Carrier, see carrier TRM.

SD-Boot mode

1. Copy **image.ub**, **boot.src** and **Boot.bin** on **SD**
    - use files from "<project folder>_binaries_<Article Name>\boot_linux" from generated binary folder, see: Get prebuilt boot binaries
    - or use prebuilt file location, see "<project folder>\prebuilt\file_location.txt"
2. Set Boot Mode to SD-Boot.
    - Depends on Carrier, see carrier TRM.
3. Insert SD-Card in SD-Slot.

JTAG

Not used on this Example.

Usage

1. Prepare HW like described on section Programming
2. Connect UART USB (most cases same as JTAG)
3. Select SD Card as Boot Mode (or QSPI - depending on step 1)

Note: See TRM of the Carrier, which is used.

Starting with Petalinux version 2020.1, the industry standard "Distro-Boot" boot flow for U-Boot was introduced, which significantly expands the possibilities of the boot process and has the primary goal of making booting much more standardised and predictable.
The boot options described above describe the common boot processes for this hardware; other boot options are possible.
For more information see Distro Boot with Boot.scr

4. Power On PCB
    1. Zynq Boot ROM loads FSBL from SD/QSPI into OCM,
    2. FSBL init PS, programs PL using the bitstream and loads U-boot from SD into DDR,
    3. U-boot loads Linux (**image.ub**) from SD/QSPI/... into DDR

Linux

1. Open Serial Console (e.g. putty)
    - Speed: 115200
    - select COM Port

    > Win OS, see device manager, Linux OS see dmesg |grep tty (UART is *USB1)

2. Linux Console:

```
petalinux login: root
Password: root
```

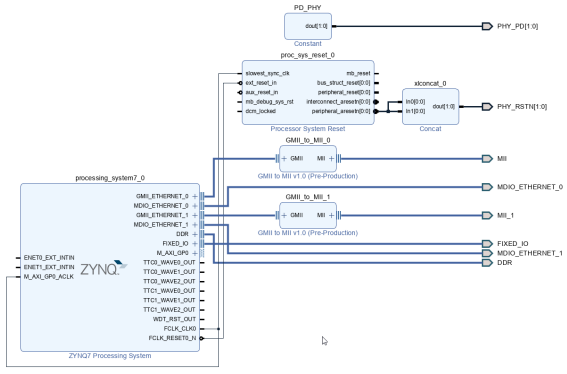
> Note: Wait until Linux boot finished

3. You can use Linux shell now.

```
i2cdetect -y -r 0                    (check I2C 0 Bus)
dmesg | grep rtc                     (RTC check)
udhcpc                                                (ETH0/ETH1 check)
cd /etc/init.d/networking restart (Network setting can be reset if
it is necessary)
ifconfig            (It is visible that both ethernet devices eth0
and eth1 have their own IP address.)
```

System Design - Vivado

Block Design



Block Design

PS Interfaces

Type	Note
DDR	---
QSPI	MIO
CAN1	MIO
ETH0	EMIO
ETH1	EMIO
SD0	MIO
UART1	MIO
I2C0	MIO
SPI1	MIO
CAN1	MIO
GPIO	MIO
WDT	EMIO
TTC0..1	EMIO

PS Interfaces

Constrains

Basic module constrains

_i_bitgen_common.xdc

```
#
# Common bitgen related settings
#

set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]

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

Design specific constrain

_i_eth.xdc

```
#############
#ETH0/ETH1
#####
#pwr_down
set_property PACKAGE_PIN L21 [get_ports {PHY_PD[0]}]
```

```
set_property PACKAGE_PIN R20 [get_ports {PHY_PD[1]}]
#rst_n
set_property PACKAGE_PIN M15 [get_ports {PHY_RSTN[0]}]
set_property PACKAGE_PIN R16 [get_ports {PHY_RSTN[1]}]
#io standard
set_property IOSTANDARD LVCMOS33 [get_ports {PHY*}]
set_property IOSTANDARD LVCMOS33 [get_ports MDIO_*]
set_property IOSTANDARD LVCMOS33 [get_ports {MII_*}]
#pullup/down for PHY address 1
set_property PULLUP   true [get_ports MII_col]
set_property PULLDOWN true [get_ports {MII_rxd[0]}]
set_property PULLDOWN true [get_ports {MII_rxd[1]}]
set_property PULLDOWN true [get_ports {MII_rxd[2]}]
set_property PULLDOWN true [get_ports {MII_rxd[3]}]
#pullup/down for PHY address 3
set_property PULLUP true [get_ports MII_1_col]
set_property PULLUP true [get_ports {MII_1_rxd[0]}]
set_property PULLDOWN true [get_ports {MII_1_rxd[1]}]
set_property PULLDOWN true [get_ports {MII_1_rxd[2]}]
set_property PULLDOWN true [get_ports {MII_1_rxd[3]}]

#############
#ETH0
#####
set_property PACKAGE_PIN M16 [get_ports MDIO_ETHERNET_0_mdio_io]
set_property PACKAGE_PIN P16 [get_ports MDIO_ETHERNET_0_mdc]
set_property PACKAGE_PIN M22 [get_ports {MII_txd[3]}]
set_property PACKAGE_PIN K21 [get_ports {MII_txd[2]}]
set_property PACKAGE_PIN M17 [get_ports {MII_txd[1]}]
set_property PACKAGE_PIN J22 [get_ports {MII_txd[0]}]
set_property PACKAGE_PIN J20 [get_ports {MII_rxd[3]}]
set_property PACKAGE_PIN J18 [get_ports {MII_rxd[2]}]
set_property PACKAGE_PIN K18 [get_ports {MII_rxd[1]}]
set_property PACKAGE_PIN L17 [get_ports {MII_rxd[0]}]
set_property PACKAGE_PIN L16 [get_ports MII_col]
set_property PACKAGE_PIN N15 [get_ports MII_crs]
set_property PACKAGE_PIN L18 [get_ports MII_rx_clk]
set_property PACKAGE_PIN P15 [get_ports MII_rx_dv]
set_property PACKAGE_PIN P17 [get_ports MII_rx_er]
set_property PACKAGE_PIN K19 [get_ports MII_tx_clk]
set_property PACKAGE_PIN J21 [get_ports MII_tx_en]

#############
#ETH1
#####
set_property PACKAGE_PIN T16 [get_ports MDIO_ETHERNET_1_mdio_io]
set_property PACKAGE_PIN T17 [get_ports MDIO_ETHERNET_1_mdc]
set_property PACKAGE_PIN R21 [get_ports {MII_1_txd[3]}]
set_property PACKAGE_PIN P22 [get_ports {MII_1_txd[2]}]
set_property PACKAGE_PIN P21 [get_ports {MII_1_txd[1]}]
set_property PACKAGE_PIN N22 [get_ports {MII_1_txd[0]}]
set_property PACKAGE_PIN T19 [get_ports {MII_1_rxd[3]}]
set_property PACKAGE_PIN T18 [get_ports {MII_1_rxd[2]}]
set_property PACKAGE_PIN R19 [get_ports {MII_1_rxd[1]}]
set_property PACKAGE_PIN R18 [get_ports {MII_1_rxd[0]}]
set_property PACKAGE_PIN P20 [get_ports MII_1_col]
set_property PACKAGE_PIN N18 [get_ports MII_1_crs]
set_property PACKAGE_PIN M19 [get_ports MII_1_rx_clk]
set_property PACKAGE_PIN N17 [get_ports MII_1_rx_dv]
set_property PACKAGE_PIN P18 [get_ports MII_1_rx_er]
set_property PACKAGE_PIN N19 [get_ports MII_1_tx_clk]
```

```
set_property PACKAGE_PIN M21 [get_ports MII_1_tx_en]
```

Software Design - Vitis

For Vitis project creation, follow instructions from:

Vitis

Application

Template location: "<project folder>\sw_lib\sw_apps\"

zynq_fsbl

TE modified 2020.2 FSBL

General:

- Modified Files: main.c, fsbl_hooks.h/.c (search for 'TE Mod' on source code)
- Add Files: te_fsbl_hooks.h/.c (for hooks and board)
- General Changes:
  - Display FSBL Banner and Device ID

Module Specific:

- only active FSBL banner independence form debug flags

zynq_fsbl_flash

TE modified 2020.2 FSBL

FSBL(for Vivado/Vitis GUI only) to initialise Zynq for QSPI programming

General:

- Modified Files: main.c
- General Changes:
  - Display FSBL Banner
  - Set FSBL Boot Mode to JTAG
  - Disable Memory initialisation

hello_te0728

Hello TE0728 is a Xilinx Hello World example as endless loop instead of one console output.

u-boot

U-Boot.elf is generated with PetaLinux. Vitis is used to generate Boot.bin.

Software Design - PetaLinux

For PetaLinux installation and project creation, follow instructions from:

- PetaLinux KICKstart

Config

Start with **petalinux-config** or **petalinux-config --get-hw-description**

Changes:

- No changes.

U-Boot

Start with **petalinux-config -c u-boot**

Changes:

- No changes.

Device Tree

```
/include/ "system-conf.dtsi"
/ {
};

/* QSPI PHY */
&qspi {
    #address-cells = <1>;
    #size-cells = <0>;
    status = "okay";
    flash0: flash@0 {
        compatible = "jedec,spi-nor";
        reg = <0x0>;
        #address-cells = <1>;
        #size-cells = <1>;
    };
};


/* SDIO */

&sdhci0 {
   disable-wp;
};


/* ETH PHY */

&gem0{
    status = "okay";
    phy-mode = "mii";
    phy-handle = <&phy1>;
    xlnx,has-mdio = <0x1>;
    mdio {
        #address-cells = <1>;
        #size-cells = <0>;
```

```
        phy1: phy@1 {
            device_type = "ethernet-phy";
            compatible = "ethernet-phy-id2000.5C90";
        max-speed = <0x64>;
            reg = <1>;
        };
    };
};


&gem1{
    status = "okay";
    phy-mode = "mii";
    phy-handle = <&phy3>;
    xlnx,has-mdio = <0x1>;
    mdio {
        #address-cells = <1>;
        #size-cells = <0>;
        phy3: phy@3 {
            device_type = "ethernet-phy";
        compatible = "ethernet-phy-id2000.5C90";
        max-speed = <0x64>;
            reg = <3>;
        };
    };
};


/* RTC */
&i2c0 {
    rtc@56 {        // Real Time Clock
       compatible = "rv3029c2";
       reg = <0x56>;
   };

};
```

FSBL patch

Must be add manually, see template

Kernel

Start with **petalinux-config -c kernel**

Changes:

- RTC_DRV_RV3029C2=y
- DP83848_PHY=y

Rootfs

Start with **petalinux-config -c rootfs**

Changes:

- I2C-tools=y

- CONFIG_util-linux-mount=y
- CONFIG_util-linux-umount=y
- busybox-httpd = y
- CONFIG_util-linux-umount=y
- CONFIG_util-linux-mount=y

Applications

See "<project folder>\os\petalinux\project-spec\meta-user\recipes-apps\"

startup

Script App to load init.sh from SD Card if available.

See: \os\petalinux\project-spec\meta-user\recipes-apps\startup\files

webfwu

Webserver application suitable for Zynq access. Need busybox-httpd

See: \os\petalinux\project-spec\meta-user\recipes-apps\webfwu\files

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
Error renderi ng macro 'page-info' Ambiguous	**Error renderi ng macro 'page-info'** Ambiguous	**Error renderi ng macro 'page-info'** Ambiguous	• style changes Revision table • remove CLBPro from source list

method overloading for method jdk.proxy241.$Proxy3496#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interfac

method overloading for method jdk.proxy241.$Proxy3496#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interfac

method overloading for method jdk.proxy241.$Proxy3496#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class java.lang.String, class com.atlassian.confluence.pages.Page] due to overlapping prototypes between: [interfac

e com.
atlassian
.
confluen
ce.user.
Conflue
nceUser
, class
java.
lang.
String,
class
com.
atlassian
.
confluen
ce.core.
Content
EntityOb
ject]
[interfac
e com.
atlassian
.user.
User,
class
java.
lang.
String,
class
com.
atlassian
.
confluen
ce.core.
Content
EntityOb
ject]

e com.
atlassian
.
confluen
ce.user.
Conflue
nceUser
, class
java.
lang.
String,
class
com.
atlassian
.
confluen
ce.core.
Content
EntityOb
ject]
[interfac
e com.
atlassian
.user.
User,
class
java.
lang.
String,
class
com.
atlassian
.
confluen
ce.core.
Content
EntityOb
ject]

e com.
atlassian
.
confluen
ce.user.
Conflue
nceUser
, class
java.
lang.
String,
class
com.
atlassian
.
confluen
ce.core.
Content
EntityOb
ject]
[interfac
e com.
atlassian
.user.
User,
class
java.
lang.
String,
class
com.
atlassian
.
confluen
ce.core.
Content
EntityOb
ject]

2021-11-03	v.14	Mohsen Chamanbaz	• Release 2020.2
2018-12-12	v.13	John Hartfiel	• Release 2018.2 • Design and Documentation is changed
2018-02-08	v.10	John Hartfiel	• Release 2017.2
2017-09-11	v.1	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy241.$Proxy3496#hasContentLevelPermission. Cannot resolve which method to invoke for [null, class	• Initial release

java. lang. String, class com. atlassian . confluen ce. pages. Page] due to overlapp ing prototyp es between : [interfac e com. atlassian . confluen ce.user. Conflue nceUser , class java. lang. String, class com. atlassian . confluen ce.core. Content EntityOb ject]

		[interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	
	All	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy241.$Proxy3496#ha	

sConten tLevelPe rmission . Cannot resolve which method to invoke for [null, class java. lang. String, class com. atlassian . confluen ce. pages. Page] due to overlapp ing prototyp es between : [interfac e com. atlassian . confluen ce.user. Conflue nceUser , class

		java. lang. String, class com. atlassian . confluen ce.core. Content EntityOb ject] [interfac e com. atlassian .user. User, class java. lang. String, class com. atlassian . confluen ce.core. Content EntityOb ject]	

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

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.proxy241.$Proxy3496#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]