TE0727 Test Board

Table of contents

- 1 Overview
  - 1.1 Key Features
  - 1.2 Revision History
  - 1.3 Release Notes and Know Issues
  - 1.4 Requirements
    - 1.4.1 Software
    - 1.4.2 Hardware
  - 1.5 Content
    - 1.5.1 Design Sources
    - 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-Boot mode
    - 3.1.3 SD-Boot mode
    - 3.1.4 JTAG
  - 3.2 Usage
    - 3.2.1 Linux
    - 3.2.2
- 4 System Design - Vivado
  - 4.1 Block Design
    - 4.1.1 PS Interfaces
  - 4.2 Constrains
    - 4.2.1 Basic module constrains
    - 4.2.2 Design specific constrain
- 5 Software Design - Vitis
  - 5.1 Application
    - 5.1.1 fsbl
    - 5.1.2 fsbl_flash
    - 5.1.3 hello_te0727
    - 5.1.4 u-boot
- 6 Software Design - PetaLinux
  - 6.1 Config
  - 6.2 U-Boot
  - 6.3 Device Tree
  - 6.4 FSBL patch
  - 6.5 Kernel
  - 6.6 Rootfs
  - 6.7 Applications
    - 6.7.1 startup
- 7 Additional Software
- 8 App. A: Change History and Legal Notices
  - 8.1 Document Change History
  - 8.2 Legal Notices
  - 8.3 Data Privacy
  - 8.4 Document Warranty
  - 8.5 Limitation of Liability
  - 8.6 Copyright Notice
  - 8.7 Technology Licenses
  - 8.8 Environmental Protection
  - 8.9 REACH, RoHS and WEEE
- 9 Table of contents

Overview

Zynq PS Design with Linux Example

Refer to http://trm.trenz.org/te0727-trm for the current online version of this manual and other available documentation.

Key Features

- Vitis/Vivado 2020.2
- PetaLinux
- SD
- USB
- I2C
- Special FSBL for QSPI programming

Revision History

Date	Vivado	Project Built	Authors	Description
2021-09-21	2020.2	TE0727-test_board-vivado_2020.2-build_8_20210921113440.zip TE0727-test_board_noprebuilt-vivado_2020.2-build_8_20210921113407.zip	Manuela Strücker	• 2020.2 release

Design Revision History

Release Notes and Know Issues

Issues	Description	Workaround	To be fixed version
FSBL/ Kernel	Petalinux does not restart after first booting	use 0001-QSPI-s25fl127_8-2020_2.patch from test_board\os\petalinux\project-spec\meta-user\recipes-kernel\linux\linux-xlnx\	---
QSPI Flash	Programming QSPI flash fails sometimes	1. Use fsbl_flash.elf for ZYNQ FSBL and try it twice 2. use Vivado 2019.2 for programming	---

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 "<project folder>\board_files*_board_files.csv"

Design supports following modules:

Module Model	Board Part Short Name	PCB Revision Support	DDR	QSPI Flash	EMMC	Others	Notes
TE0727-01-010-1C	10_512MB	REV01	512MB	16MB	NA	NA	SW Design changes for I2C are necessary
TE0727-02-41C34*	10_512MB	REV02	512MB	16MB	NA	NA	NA

*used as reference

Hardware Modules

Design supports following carriers:

Carrier Model	Notes
---	

*used as reference

Hardware Carrier

Additional HW Requirements:

Additional Hardware	Notes
USB cable	Connect to USB2 or better USB3 Hub for proper power over USB

*used as reference

Additional Hardware

Content

For general structure and usage 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
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
Boot Source	*.scr	Distro Boot file
DebugProbes-File	*.ltx	Definition File for Vivado/Vivado Labtools Debugging Interface
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
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)
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
SREC-File	*.srec	Converted Software Application for MicroBlaze Processor Systems

Prebuilt files (only on ZIP with prebuilt 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:

- TE0727 "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 launch.

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 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.

5. 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
6. Configure the **boot.scr** file as needed, see Distro Boot with Boot.scr
7. 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>"

8. 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/Vitis/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

Boot.bin on **QSPI Flash** and **image.ub** and **boot.scr** on **SD**.

1. Connect **USB Power In** to get power on module
2. Open Vivado Project with "vivado_open_existing_project_guimode.cmd" or if not created, create with "vivado_create_project_guimode.cmd"

3. **run on Vivado TCL (Script programs BOOT.bin on QSPI flash)**

```
TE::pr_program_flash -swapp u-boot
TE::pr_program_flash -swapp hello_te0727 (optional)
```

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

4. Remove cable from **USB Power In**
5. Copy **image.ub** and **boot.scr** 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"
   - **Important:** Do not copy Boot.bin on SD (it is not used; see SD note), only other files.
6. Copy **init.sh** on **SD**
   - location: <project folder>/misc/sd/
7. Insert SD-Card in SD-Slot.
8. Connect **USB Power In** to get power on module

SD-Boot mode

Xilinx Zynq devices in CLG225 package do not support SD Card boot directly from ROM bootloader. Use QSPI for primary boot (fsbl, u-boot) and SD for secondary boot (image.ub, boot.src)

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. Insert SD Card with image.ub and boot.src

> ✓ 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 QSPI into OCM,

   2. FSBL init PS, programs PL using the bitstream and loads U-boot from QSPI 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

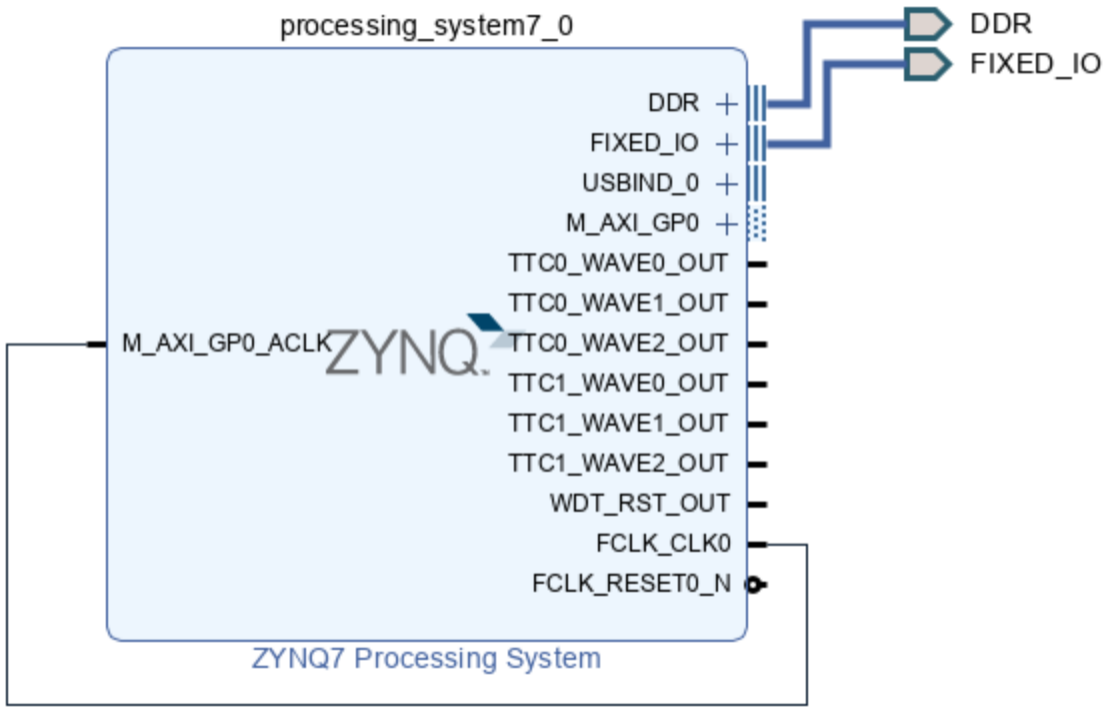
3. You can use Linux shell now.

```
i2cdetect -y -r 0        (check I2C 1 Bus)
lsusb                             (USB check)
```

4. Option Features
    - init.sh scripts
        - add init.sh script on SD, content will be load automatically on startup (template included in "<project folder>\misc\SD")

System Design - Vivado

Block Design



Block Design

PS Interfaces

Activated interfaces:

Type	Note
DDR	---
QSPI	MIO
SD0	---
SD1	MIO
I2C0	---
I2C1	MIO
UART1	MIO
GPIO MIO	MIO
SWDT	EMIO
TTC0..1	EMIO
USB0	MIO

Constrains

Basic module constrains

_i_bitgen_common.xdc

```
#
# Common BITGEN related settings for TE0727 SoM
#
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

Design specific constrain

_i_common.xdc

```
#
#
#
set_property BITSTREAM.CONFIG.UNUSEDPIN PULLUP [current_design]
```

_i_TE0727.xdc

```
#set_property PACKAGE_PIN G11 [get_ports {CEC_A[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {CEC_A[0]}]
#set_property PACKAGE_PIN H13 [get_ports {HPD_A}]
#set_property IOSTANDARD LVCMOS33 [get_ports {HPD_A}]
#set_property PACKAGE_PIN G14 [get_ports {GLED[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {GLED[0]}]
#set_property PACKAGE_PIN G12 [get_ports {IIC_A_scl_io}]
#set_property PACKAGE_PIN H12 [get_ports {IIC_A_sda_io}]
#set_property IOSTANDARD LVCMOS33 [get_ports {IIC_A_*}]
#set_property PACKAGE_PIN K12 [get_ports {CT_HPD[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {CT_HPD[0]}]
#
#set_property PACKAGE_PIN F12 [get_ports {HDMI_TXC_P}]
#set_property PACKAGE_PIN E13 [get_ports {HDMI_TXC_N}]
#set_property PACKAGE_PIN E11 [get_ports {HDMI_TX_P[0]}]
#set_property PACKAGE_PIN E12 [get_ports {HDMI_TX_N[0]}]
#set_property PACKAGE_PIN G15 [get_ports {HDMI_TX_P[1]}]
#set_property PACKAGE_PIN F15 [get_ports {HDMI_TX_N[1]}]
#set_property PACKAGE_PIN F14 [get_ports {HDMI_TX_N[2]}]
#set_property PACKAGE_PIN F13 [get_ports {HDMI_TX_P[2]}]
#set_property IOSTANDARD TMDS_33 [get_ports {HDMI_*}]
#
#set_property PACKAGE_PIN J11 [get_ports {GPIO_tri_io[0]}]
#set_property PACKAGE_PIN H11 [get_ports {GPIO_tri_io[1]}]
#set_property PACKAGE_PIN J15 [get_ports {GPIO_tri_io[2]}]
#set_property PACKAGE_PIN L15 [get_ports {GPIO_tri_io[3]}]
#set_property PACKAGE_PIN N13 [get_ports {GPIO_tri_io[4]}]
```

```
#set_property PACKAGE_PIN P8  [get_ports {GPIO_tri_io[5]}]
#set_property PACKAGE_PIN M10 [get_ports {GPIO_tri_io[6]}]
#set_property PACKAGE_PIN L12 [get_ports {GPIO_tri_io[7]}]
#set_property PACKAGE_PIN M11 [get_ports {GPIO_tri_io[8]}]
#set_property PACKAGE_PIN P10 [get_ports {GPIO_tri_io[9]}]
#set_property PACKAGE_PIN P9  [get_ports {GPIO_tri_io[10]}]
#set_property PACKAGE_PIN K15 [get_ports {GPIO_tri_io[11]}]
#set_property PACKAGE_PIN M9  [get_ports {GPIO_tri_io[12]}]
#set_property PACKAGE_PIN L13 [get_ports {GPIO_tri_io[13]}]
#set_property PACKAGE_PIN L14 [get_ports {GPIO_tri_io[14]}]
#set_property PACKAGE_PIN M15 [get_ports {GPIO_tri_io[15]}]
#set_property PACKAGE_PIN J14 [get_ports {GPIO_tri_io[16]}]
#set_property PACKAGE_PIN N14 [get_ports {GPIO_tri_io[17]}]
#set_property PACKAGE_PIN K11 [get_ports {GPIO_tri_io[18]}]
#set_property PACKAGE_PIN N9  [get_ports {GPIO_tri_io[19]}]
#set_property PACKAGE_PIN J13 [get_ports {GPIO_tri_io[20]}]
#set_property PACKAGE_PIN H14 [get_ports {GPIO_tri_io[21]}]
#set_property PACKAGE_PIN R10 [get_ports {GPIO_tri_io[22]}]
#set_property PACKAGE_PIN M14 [get_ports {GPIO_tri_io[23]}]
#set_property PACKAGE_PIN P15 [get_ports {GPIO_tri_io[24]}]
#set_property PACKAGE_PIN M12 [get_ports {GPIO_tri_io[25]}]
#set_property PACKAGE_PIN K13 [get_ports {GPIO_tri_io[26]}]
#set_property PACKAGE_PIN R15 [get_ports {GPIO_tri_io[27]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {GPIO_tri_io*}]
#
```

Software Design - Vitis

For Vitis project creation, follow instructions from:

Vitis

Application

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

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:

- Add Files: all TE Files start with te_*
    - READ MAC from EEPROM and make Address accessible by UBOOT (need copy defines on uboot platform-top.h)
    - CPLD access
    - Read CPLD Firmware and SoC Type
    - Configure Marvell PHY

fsbl_flash

TE modified 2020.2 FSBL

General:

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

hello_te0727

Hello TE0727 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:

- CONFIG_ENV_IS_NOWHERE=y
- # CONFIG_ENV_IS_IN_SPI_FLASH is not set

Change platform-top.h:

Device Tree

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

&qspi {
    #address-cells = <1>;
    #size-cells = <0>;
    status = "okay";
    flash0: flash@0 {
```

```
        compatible = "jedec,spi-nor";
        reg = <0x0>;
        #address-cells = <1>;
        #size-cells = <1>;
        spi-max-frequency = <50000000>;
        partition@0x00000000 {
            label = "boot";
            reg = <0x00000000 0x00500000>;
        };
        partition@0x00500000 {
            label = "bootenv";
            reg = <0x00500000 0x00020000>;
        };
        partition@0x00520000 {
            label = "kernel";
            reg = <0x00520000 0x00a80000>;
        };
        partition@0x00fa0000 {
            label = "spare";
            reg = <0x00fa0000 0x00000000>;
        };
    };
};

&gpio0 {
    interrupt-controller;
    #interrupt-cells = <2>;
};


/* I2C1 */
&i2c1 {
    #address-cells = <1>;
    #size-cells = <0>;

    i2cmux: i2cmux@70  {
        compatible = "nxp,pca9540";
        #address-cells = <1>;
        #size-cells = <0>;
        reg = <0x70>;

        ID_I2C@0 {
            #address-cells = <1>;
            #size-cells = <0>;
            reg = <0>;
        };
        CSI_I2C@1 {
            #address-cells = <1>;
            #size-cells = <0>;
            reg = <1>;
        };
    };

};
/* USB */



/{
    usb_phy0: usb_phy@0 {
        compatible = "ulpi-phy";
```

```
        #phy-cells = <0>;
        reg = <0xe0002000 0x1000>;
        view-port = <0x0170>;
        drv-vbus;
    };
};

&usb0 {
    usb-phy = <&usb_phy0>;
} ;
```

FSBL patch

Must be add manually, see template

Kernel

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

Changes:

- No changes

Change linux-xlnx_%.bbappend:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}:"

SRC_URI += "file://devtool-fragment.cfg"
SRC_URI += "file://0001-QSPI-s25fl127_8-2020_2.patch"
```

- Add 0001-QSPI-s25fl127_8-2020_2.patch to "<project folder>\project-spec\meta-user\recipes-kernel\linux\linux-xlnx\"

Rootfs

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

Changes:

- CONFIG_i2c-tools=y
- CONFIG_packagegroup-petalinux-utils=y
- CONFIG_util-linux-mount=y
- CONFIG_util-linux-umount=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.

Additional Software

No additional software is needed.

App. A: Change History and Legal Notices

Document Change History

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

Date	Document Revision	Authors	Description
Error rendering macro 'page-info' Ambiguous method overloading for method jdk.proxy24.$Proxy3572#hasContentLevelPermission. Cannot resolve which method	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy24.$Proxy3572#hasContentLevelPermission. Cannot resolve which method	**Error rendering macro 'page-info'** Ambiguous method overloading for method jdk.proxy24.$Proxy3572#hasContentLevelPermission. Cannot resolve which method	• initial release 2020.2

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.confluen

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.confluen

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.confluen

ce.core.Content EntityObject][interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	ce.core.Content EntityObject][interface com.atlassian.user.User, class java.lang.String, class com.atlassian.confluence.core.ContentEntityObject]	ce.core.Content EntityObject][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.	--

proxy24
4.$Proxy
3572#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

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.$Proxy3572#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]