TE0802 Test Board

Table of contents

Overview

Refer to http://trenz.org/te0xyz-info for the current online version of this manual and other available documentation.

Key Features

- Vivado 2018.3
- PetaLinux
- SD
- ETH
- USB
- I2C
- DP
- VGA
- DIPS, LEDs, Buttons
- Audio
- MAC from EEPROM
- Modified FSBL for Resets
- Special FSBL for QSPI programming

Revision History

Date	Vivado	Project Built	Authors	Description
2019-08-30	2018.3	TE0802-test_board-vivado_2018.3-build_07_20190830103019.zip TE0802-test_board_noprebuilt-vivado_2018.3-build_07_20190830103313.zip	Oleksandr Kiyenko, 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
Vivado	2018.3	needed
SDK	2018.3	needed
PetaLinux	2018.3	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	EMMC	Others	Notes
TE0802-02-2AEV2-A	2cg_s1gb	REV02	1GB	32MB	NA	NA	Samsung DDR4L
TE0802-02-2AEU2-A	2cg_i1gb	REV02	1GB	32MB	NA	NA	ISSI DDR4L

Hardware Modules

Design supports following carriers:

Carrier Model	Notes

Hardware Carrier

Additional HW Requirements:

Additional Hardware	Notes
M2 SSD	tested withSamsung 050 Pro 256GB
headphones	
Monitor with DP support	Note: not all monitors will be supported by Xilinx. Adapter to other connector standard is not supported

Additional Hardware

Content

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

Design Sources

Type	Location	Notes
Vivado	<design name>/block_design <design name>/constraints <design name>/ip_lib	Vivado Project will be generated by TE Scripts
SDK/HSI	<design name>/sw_lib	Additional Software Template for SDK/HSI and apps_list.csv with settings for HSI
PetaLinux	<design name>/os/petalinux	PetaLinux template with current configuration

Design sources

Additional Sources

Type	Location	Notes
init.sh	<design name>/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
DebugProbes-File	*.ltx	Definition File for Vivado/Vivado Labtools Debugging Interface
Diverse Reports	---	Report files in different formats
Hardware-Platform-Specification-Files	*.hdf	Exported Vivado Hardware Specification for SDK/HSI 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/SDK/PetaLinux/SDx version. Do never use different Versions of Xilinx Software for the same Project.

Reference Design is available on:

- TE0802 "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.

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

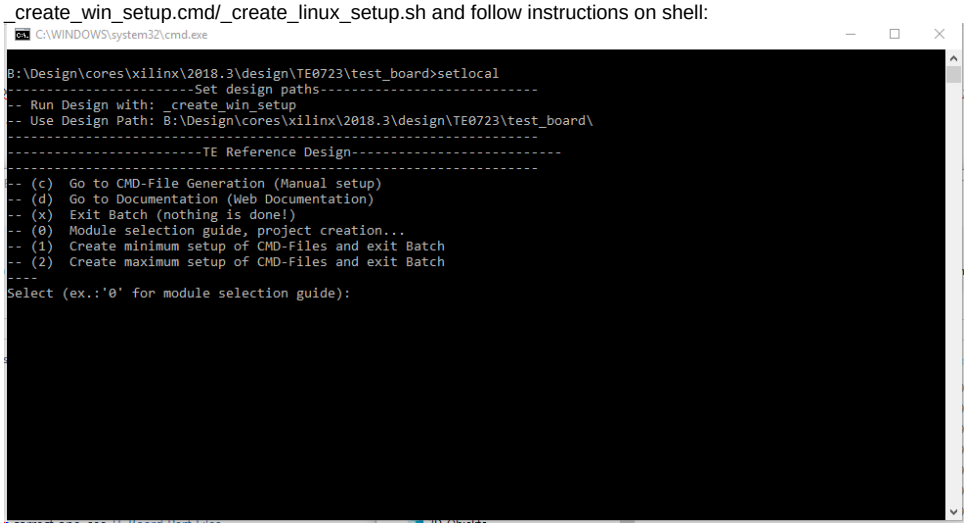
See also:

- Vivado/SDK/SDSoC#XilinxSoftware-BasicUserGuides
- Vivado Projects
- 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/SDK GUI. For currently Scripts limitations on Win and Linux OS see: Project Delivery Currently limitations of functionality

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



2. Press 0 and enter to start "Module Selection Guide"
3. (optional Win OS) Generate Virtual Drive or use short directory for the reference design (for example x:\<design name>)
4. Create Project (follow instruction of the product selection guide), settings file will be configured automatically during this process
    a. (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 TE Board Part Files
5. Create HDF and export to prebuilt folder
    a. Run on Vivado TCL: TE::hw_build_design -export_prebuilt
    Note: Script generate design and export files into \prebuilt\hardware\<short dir>. Use GUI is the same, except file export to prebuilt folder
6. Create Linux (uboot.elf and image.ub) with exported HDF
    a. HDF is exported to "prebuilt\hardware\<short name>"
    Note: HW Export from Vivado GUI create another path as default workspace.
    Create Linux images on VM, see PetaLinux KICKstart
        i. Use TE Template from /os/petalinux
7. Add Linux files (uboot.elf and image.ub) to prebuilt folder
    a. "prebuilt\os\petalinux\<ddr size>" or "prebuilt\os\petalinux\<short name>"
8. Generate Programming Files with HSI/SDK
    a. Run on Vivado TCL: TE::sw_run_hsi
    Note: Scripts generate applications and bootable files, which are defined in "sw_lib\apps_list.csv"
    b. (alternative) Start SDK with Vivado GUI or start with TE Scripts on Vivado TCL: TE::sw_run_sdk
    Note: See SDK Projects

Launch

Programming

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

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

QSPI

Optional for Boot.bin on QSPI Flash and image.ub on SD.

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"

3. Type on Vivado TCL Console: TE::pr_program_flash_binfile -swapp u-boot
   Note: To program with SDK/Vivado GUI, use special FSBL (zynqmp_fsbl_flash) on setup
   optional "TE::pr_program_flash_binfile -swapp hello_te0802" possible
4. Copy image.ub on SD-Card
   - For correct prebuilt file location, see <design_name>/prebuilt/readme_file_location.txt
5. Set Boot Mode to QSPI-Boot and insered SD.
   - Depends on Carrier, see carrier TRM.

SD

1. Copy image.ub, Boot.bin and init.sh(optional on /misc/sd) on SD-Card.
   - For correct prebuilt file location, see <design_name>/prebuilt/readme_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 43680037
2. Connect UART USB (most cases same as JTAG)
3. Connect Monitors, ETH, M2...
4. Select SD Card as Boot Mode (or QSPI - depending on step 1)
   Note: See TRM of the Carrier, which is used.
5. Power On PCB
   Note: 1. Zynq Boot ROM loads FSBL from SD into OCM, 2. FSBL loads U-boot from SD into DDR, 3. U-boot load Linux from SD into DDR

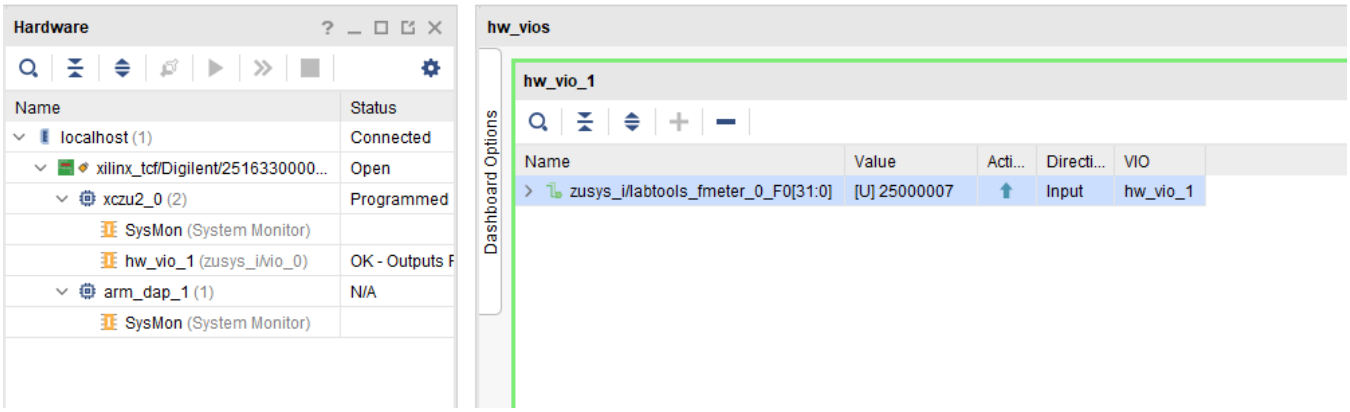
Linux

1. Open Serial Console (e.g. putty)
   a. Speed: 115200
   b. COM Port: Win OS, see device manager, Linux OS see dmesg |grep tty (UART is *USB1)
2. Linux Console:
   Note: Wait until Linux boot finished For Linux Login use:
   a. User Name: root
   b. Password: root
3. You can use Linux shell now.
   a. I2C 0 Bus type: i2cdetect -y -r 0
   b. RTC check: dmesg | grep rtc
   c. ETH0 works with udhcpc
   d. USB type "lsusb" or connect USB device
   e. PCIe (M2 SSD) type "lspci"
   f. VGA connect Monitor (it show test screen)
   g. DP: second console will be shown on the monitor, when boot process is finished. (conneced keyboard to USB, to interact with the second console)
   h. Audio type: aplay /run/media/mmcblk0p1/<filename>.wav Note: DP must be connected to activate audio drivers. Use .wav or other aplay supported formate
4. Option Features
   a. Webserver to get access to Zynq
      i. insert IP on web browser to start web interface
   b. init.sh scripts
      i. add init.sh script on SD, content will be load automatically on startup (template included in ./misc/SD)
5. All button cross will be reset LEDs with values from DIP
6. LCD is connected to counter

Vivado HW Manager

Open Vivado HW-Manager and add VIO signal to dashboard (*.ltx located on prebuilt folder)

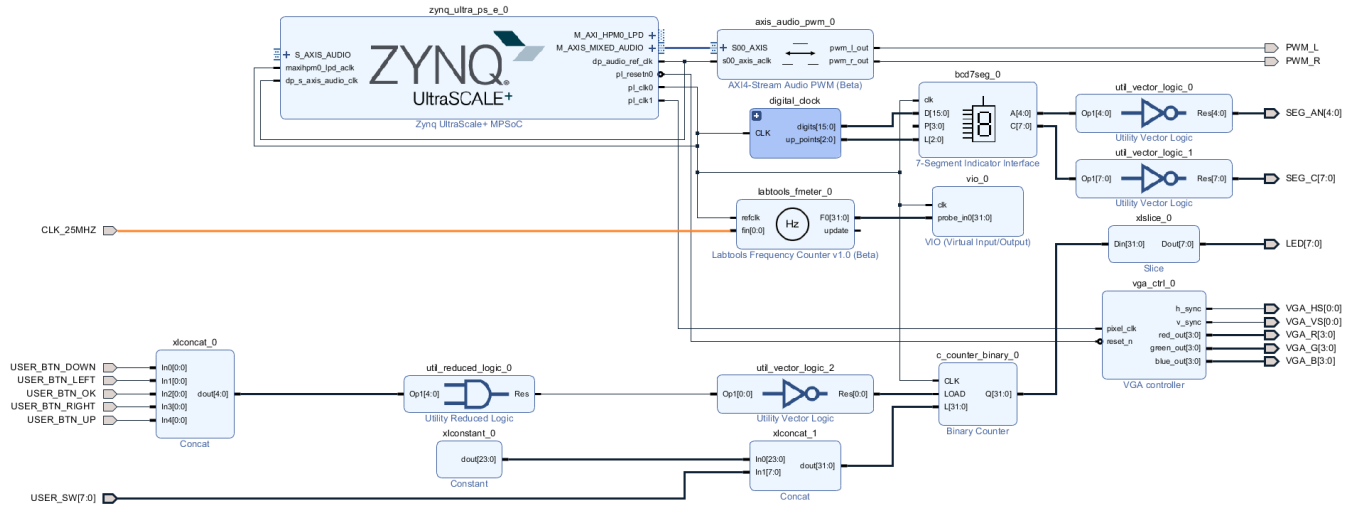
- Monitoring:
  - 25MHz CLK Set radix from VIO signals to unsigned integer. Note: Frequency Counter is inaccurate and displayed unit is Hz)



Vivado Hardware Manager

System Design - Vivado

Block Design



Block Design

PS Interfaces

Activated interfaces:

Type	Note
DDR	

QSPI	MIO
SD0	MIO
I2C0	MIO
I2C1	MIO
UART0	MIO
GPIO0	MIO
GPIO1	MIO
GPIO2	MIO
SWDT0..1	
TTC0..3	
GEM3	MIO
USB0	MIO+ GT Lane1
PCIe	MIO + GT Lane0 (as rootcomplex)
DP	MIO+ GT Lane2,3

PS Interfaces

Constrains

Basic module constrains

_i_bitgen_common.xdc

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.UNUSEDPIN PULLNONE [current_design]
```

Design specific constrain

_i_io.xdc

```
set_property PACKAGE_PIN E3 [get_ports PWM_L]
set_property PACKAGE_PIN F4 [get_ports PWM_R]
set_property IOSTANDARD LVCMOS18 [get_ports PWM_*]

#set_property PACKAGE_PIN T2 [ get_ports USER_BTN_DOWN ]
#set_property PACKAGE_PIN U2 [ get_ports USER_BTN_UP ]
#set_property PACKAGE_PIN U1 [ get_ports USER_BTN_RIGHT ]
#set_property PACKAGE_PIN R1 [ get_ports USER_BTN_LEFT ]
#set_property PACKAGE_PIN T1 [ get_ports USER_BTN_OK ]
#set_property IOSTANDARD LVCMOS18 [ get_ports USER_BTN_* ]

set_property PACKAGE_PIN P3 [get_ports {USER_SW[0]}]
set_property PACKAGE_PIN P2 [get_ports {USER_SW[1]}]
set_property PACKAGE_PIN M1 [get_ports {USER_SW[2]}]
set_property PACKAGE_PIN L1 [get_ports {USER_SW[3]}]
set_property PACKAGE_PIN K1 [get_ports {USER_SW[4]}]
```

```
set_property PACKAGE_PIN J2 [get_ports {USER_SW[5]}]
set_property PACKAGE_PIN M4 [get_ports {USER_SW[6]}]
set_property PACKAGE_PIN M5 [get_ports {USER_SW[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports USER_SW*]

set_property PACKAGE_PIN U2 [get_ports {USER_BTN_UP}]
set_property PACKAGE_PIN U1 [get_ports {USER_BTN_RIGHT}]
set_property PACKAGE_PIN T2 [get_ports {USER_BTN_DOWN}]
set_property PACKAGE_PIN R1 [get_ports {USER_BTN_LEFT}]
set_property PACKAGE_PIN T1 [get_ports {USER_BTN_OK}]
set_property IOSTANDARD LVCMOS18 [get_ports USER_BTN*]

set_property PACKAGE_PIN P1 [get_ports {LED[0]}]
set_property PACKAGE_PIN N2 [get_ports {LED[1]}]
set_property PACKAGE_PIN M2 [get_ports {LED[2]}]
set_property PACKAGE_PIN L2 [get_ports {LED[3]}]
set_property PACKAGE_PIN J1 [get_ports {LED[4]}]
set_property PACKAGE_PIN H2 [get_ports {LED[5]}]
set_property PACKAGE_PIN L4 [get_ports {LED[6]}]
set_property PACKAGE_PIN L3 [get_ports {LED[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports LED*]

set_property PACKAGE_PIN F2 [get_ports {VGA_R[0]}]
set_property PACKAGE_PIN F1 [get_ports {VGA_R[1]}]
set_property PACKAGE_PIN G2 [get_ports {VGA_R[2]}]
set_property PACKAGE_PIN G1 [get_ports {VGA_R[3]}]
set_property PACKAGE_PIN C2 [get_ports {VGA_G[0]}]
set_property PACKAGE_PIN D2 [get_ports {VGA_G[1]}]
set_property PACKAGE_PIN D1 [get_ports {VGA_G[2]}]
set_property PACKAGE_PIN E1 [get_ports {VGA_G[3]}]
set_property PACKAGE_PIN A3 [get_ports {VGA_B[0]}]
set_property PACKAGE_PIN A2 [get_ports {VGA_B[1]}]
set_property PACKAGE_PIN B2 [get_ports {VGA_B[2]}]
set_property PACKAGE_PIN B1 [get_ports {VGA_B[3]}]
set_property PACKAGE_PIN B7 [get_ports {VGA_VS[0]}]
set_property PACKAGE_PIN A6 [get_ports {VGA_HS[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_B[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_B[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_B[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_B[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_G[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_G[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_G[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_G[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {VGA_HS[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_R[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_R[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_R[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {VGA_R[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {VGA_VS[0]}]

set_property PACKAGE_PIN J3 [get_ports CLK_25MHZ]
set_property IOSTANDARD LVCMOS18 [get_ports CLK_25MHZ]
# SEG_C[0] = SEG_CA
set_property PACKAGE_PIN E4 [get_ports {SEG_C[0]}]
set_property PACKAGE_PIN D3 [get_ports {SEG_C[1]}]
set_property PACKAGE_PIN N5 [get_ports {SEG_C[2]}]
set_property PACKAGE_PIN P5 [get_ports {SEG_C[3]}]
set_property PACKAGE_PIN N4 [get_ports {SEG_C[4]}]
set_property PACKAGE_PIN C3 [get_ports {SEG_C[5]}]
set_property PACKAGE_PIN N3 [get_ports {SEG_C[7]}]
```

```
set_property PACKAGE_PIN R5 [get_ports {SEG_C[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports SEG_C*]

set_property PACKAGE_PIN A8 [get_ports {SEG_AN[0]}]
set_property PACKAGE_PIN A9 [get_ports {SEG_AN[1]}]
set_property PACKAGE_PIN B9 [get_ports {SEG_AN[2]}]
set_property PACKAGE_PIN A7 [get_ports {SEG_AN[3]}]
set_property PACKAGE_PIN B6 [get_ports {SEG_AN[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports SEG_AN*]
```

Software Design - SDK/HSI

For SDK project creation, follow instructions from:

SDK Projects

Application

Template location: ./sw_lib/sw_apps/

zynqmp_fsbl

TE modified 2018.3 FSBL

General:

- Modified Files: xfsbl_main.c, xfsbl_hooks.h/.c, xfsbl_board.h/.c(search for 'TE Mod' on source code)
- Add Files: te_xfsbl_hooks.h/.c (for hooks and board)\n\
- General Changes:
    - Display FSBL Banner and Device Name

Module Specific:

- Add Files: all TE Files start with te_*
    - ETH+OTG+SSD Reset over MIO

zynqmp_fsbl_flash

TE modified 2018.3 FSBL

General:

- Modified Files: xfsbl_initialisation.c, xfsbl_hw.h, xfsbl_handoff.c, xfsbl_main.c
- General Changes:
    - Display FSBL Banner
    - Set FSBL Boot Mode to JTAG
    - Disable Memory initialisation

zynqmp_pmufw

Xilinx default PMU firmware.

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:

- CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_3_MAC=""

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:

```
#include <configs/platform-auto.h>
#define CONFIG_SYS_BOOTM_LEN 0xF000000

#define DFU_ALT_INFO_RAM \
"dfu_ram_info=" \
"setenv dfu_alt_info " \
"image.ub ram $netstart 0x1e00000\0" \
"dfu_ram=run dfu_ram_info && dfu 0 ram 0\0" \
"thor_ram=run dfu_ram_info && thordown 0 ram 0\0"

#define DFU_ALT_INFO_MMC \
"dfu_mmc_info=" \
"set dfu_alt_info " \
"${kernel_image} fat 0 1\\\\;" \
"dfu_mmc=run dfu_mmc_info && dfu 0 mmc 0\0" \
"thor_mmc=run dfu_mmc_info && thordown 0 mmc 0\0"

/*Required for uartless designs */
#ifndef CONFIG_BAUDRATE
#define CONFIG_BAUDRATE 115200
#ifdef CONFIG_DEBUG_UART
#undef CONFIG_DEBUG_UART
#endif
#endif

/*Define CONFIG_ZYNQMP_EEPROM here and its necessaries in u-boot menuconfig if you had EEPROM memory. */
#define CONFIG_ZYNQMP_EEPROM
#ifdef CONFIG_ZYNQMP_EEPROM
#define CONFIG_SYS_I2C_EEPROM_ADDR_LEN 1
#define CONFIG_CMD_EEPROM
#define CONFIG_ZYNQ_EEPROM_BUS 1
#define CONFIG_ZYNQ_GEM_EEPROM_ADDR 0x50
#define CONFIG_ZYNQ_GEM_I2C_MAC_OFFSET 0xFA
```

```
#endif
```

Device Tree

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

#include <dt-bindings/gpio/gpio.h>

/* SD */

&sdhci0 {
    disable-wp;
    no-1-8-v;
};

/* USB  */

&dwc3_0 {
    status = "okay";
    dr_mode = "host";
    //snps,usb3_lpm_capable;
    //snps,dis_u3_susphy_quirk;
    //snps,dis_u2_susphy_quirk;
    //phy-names = "usb2-phy","usb3-phy";
        //phys = <&lane1 4 0 2 26000000>;
    //maximum-speed = "super-speed";
};

/ {
    leds {
        compatible = "gpio-leds";
        ndp_en {
            label = "ndp_en";
            gpios = <&gpio 26 GPIO_ACTIVE_HIGH>;
            default-state = "on";
        };
        ssd_sleep {
            label = "ssd_sleep";
            gpios = <&gpio 32 GPIO_ACTIVE_HIGH>;
            default-state = "on";
        };
        usb_reset {
            label = "usb_reset";
            gpios = <&gpio 38 GPIO_ACTIVE_HIGH>;
            default-state = "on";
        };
    };
};

/* ETH PHY */

&gem3 {
    phy-handle = <&phy0>;
    phy0: phy0@1 {
        device_type = "ethernet-phy";
```

```
        reg = <1>;
    };
};

/* QSPI */

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

Kernel

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

Changes:

- CONFIG_CPU_IDLE is not set (only needed to fix JTAG Debug issue)
- CONFIG_CPU_FREQ is not set (only needed to fix JTAG Debug issue)
- CONFIG_CPU_FREQ_DEFAULT_GOV_USERSPACE is not set (only needed to fix JTAG Debug issue)
- CONFIG_EDAC_CORTEX_ARM64=y

Rootfs

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

Changes:

- CONFIG_i2c-tools=y
- CONFIG_busybox-httpd=y (for web server app)
- CONFIG_packagegroup-petalinux-utils(util-linux,cpufrequtils,bridge-utils,mtd-utils,usbutils,pciutils,canutils,i2c-tools,smartmontools,e2fsprogs)
- CONFIG_alsa-utils=y
- CONFIG_alsa-utils-aplay=y

Applications

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 accemble for Zynq access. Need busybox-httpd

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
2019-08-30	v.1	John Hartfiel	• 2018.3
--	all	John Hartfiel	--

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.

2019-06-07