TE0802 Getting Started Guide

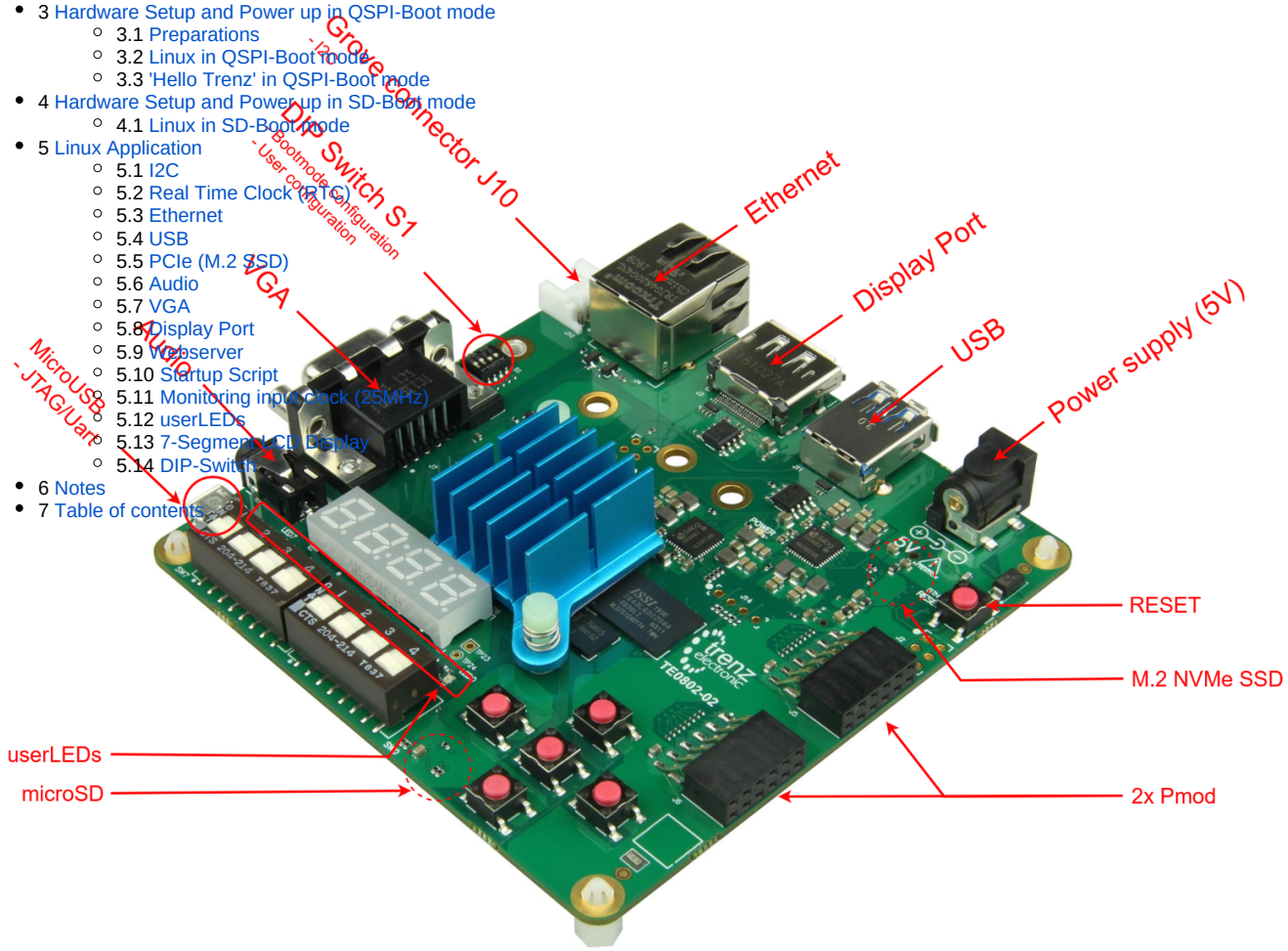
Overview

Table of contents

This guide shows the main components of the TE0802 module and introduces the first steps to get the provided reference design up and running.

- 1 Overview
  - 1.1 Prerequisites
  - 1.2 Documentation
  - 1.3 Hardware Features and Overview
- 2 Reference Design - Introduction
- 3 Hardware Setup and Power up in QSPI-Boot mode
  - 3.1 Preparations
  - 3.2 Linux in QSPI-Boot mode
  - 3.3 'Hello Trenz' in QSPI-Boot mode
- 4 Hardware Setup and Power up in SD-Boot mode
  - 4.1 Linux in SD-Boot mode
- 5 Linux Application
  - 5.1 I2C
  - 5.2 Real Time Clock (RTC)
  - 5.3 Ethernet
  - 5.4 USB
  - 5.5 PCIe (M.2 SSD)
  - 5.6 Audio
  - 5.7 VGA
  - 5.8 Display Port
  - 5.9 Webserver
  - 5.10 Startup Script
  - 5.11 Monitoring input clock (25MHz)
  - 5.12 userLEDs
  - 5.13 7-Segment LCD Display
  - 5.14 DIP-Switch
- 6 Notes
- 7 Table of contents

This module TE0802 has a Xilinx Zynq Ultrascale+ and several hardware features onboard that allows you to create digital hardware and software designs. For communication and configuration the module board offers a JTAG/UART Interface.



Module TE0802

Prerequisites

Hardware	Software

• TE0802 module • power supply (5V) • MicroUSB cable • 3.5mm earphone jack (optional) • VGA cable (optional) • ethernet cable • display port cable (optional) • USB keyboard (optional) • SD card • M.2 NVMe SSD (optional)	• Vitis (Vivado included) • PuTTY (or any other serial communicator) • TE0802 Reference Design

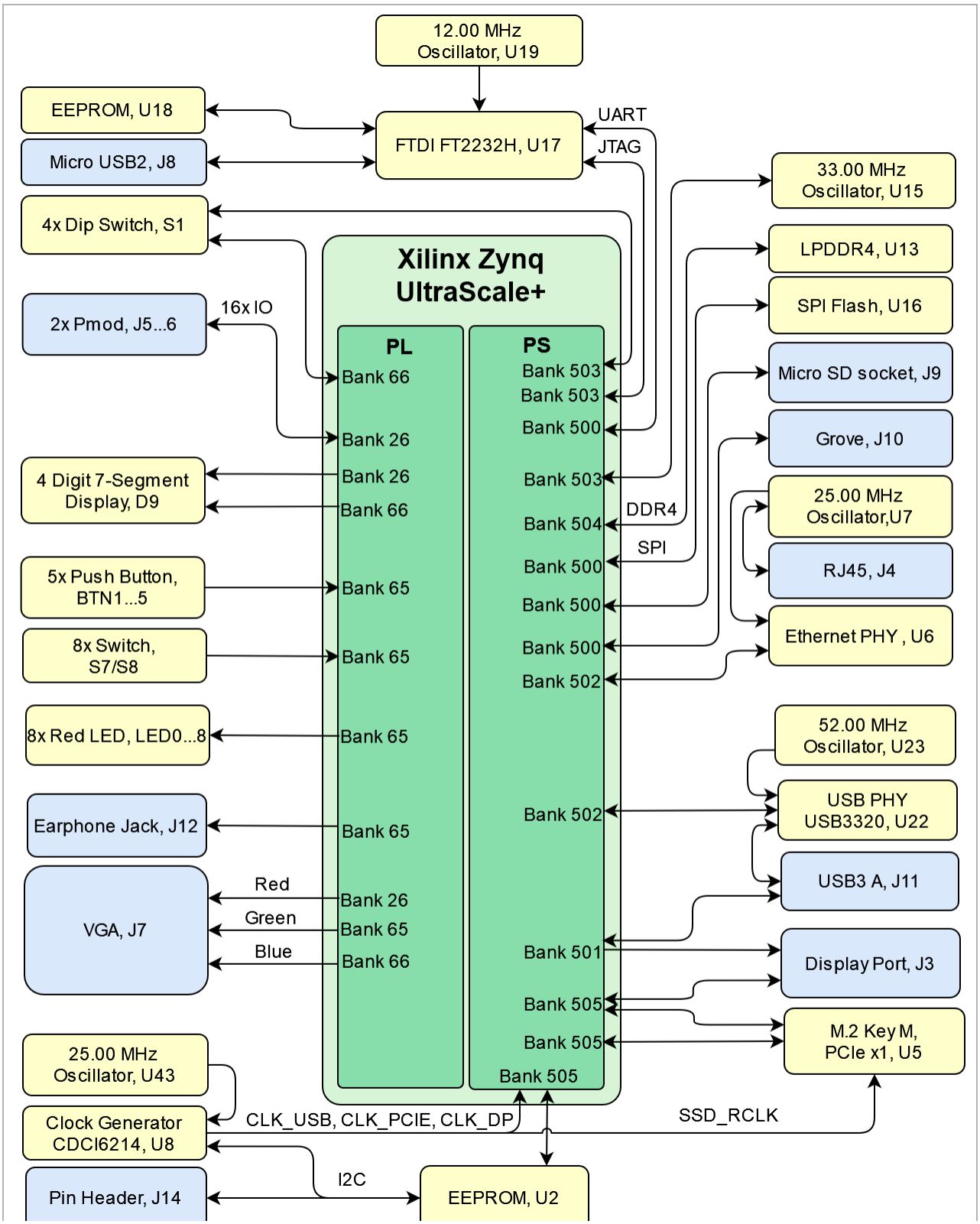
Documentation

- Official links to the shop:
  - TE0802-02-1AEV2-A-MPSoC-Development-Board-mit-Xilinx-Zynq-UltraScale-ZU1-und-1-GB-LPDDR4
  - TE0802-02-2AEV2-A-MPSoC-Development-Board-mit-Xilinx-Zynq-UltraScale-ZU2-und-LPDDR4
- Technical Reference Manual:
  - TE0802 Technical Reference Manual
- Resources & Reference Designs:
  - TE0802 Resources
  - TE0802 Reference Design

Hardware Features and Overview

	TE0802-02-1AEV2-A	**TE0802-02-2AEV2-A**
MPSoC	Xilinx Zynq UltraScale+ • XC**ZU1**CG-1SBVA484E • Speed Grade: -1 • Temperature Grade: Extended (0 to +100 °C)	Xilinx Zynq UltraScale+ • XC**ZU2**CG-1SBVA484E • Speed Grade: -1 • Temperature Grade: Extended (0 to +100 °C)
Storage	• 1 GByte LPDDR4 • 32 MByte SPI Flash • MicroSD-Karte • M2 PCIe SSD support • EEPROM	
Display	• DisplayPort • VGA • 4-digit 7-segment LED • 8 LEDs	
Audio	• 3.5 mm earphone jack (PWM output)	
Connectors	• 2x Pmod	

Communication & Debug	• USB 3.0 Host (type A connector) • USB JTAG/UART Micro-USB • 1GB Ethernet RJ45
Input	• 5 push buttons • 8 bit slide switches • Reset button


12.00 MHz
Oscillator, U19
EEPROM, U18
Micro USB2, J8
FTDI FT2232H, U17
UART
JTAG
4x Dip Switch, S1
2x Pmod, J5...6
16x IO
Xilinx Zynq
UltraScale+
PL
PS
Bank 66
Bank 26
Bank 26
Bank 66
Bank 65
Bank 65
Bank 65
Bank 65
Bank 26
Bank 65
Bank 66
Bank 503
Bank 503
Bank 500
Bank 503
Bank 504
Bank 500
Bank 500
Bank 500
Bank 502
Bank 502
Bank 501
Bank 505
Bank 505
Bank 505
4 Digit 7-Segment
Display, D9
5x Push Button,
BTN1...5
8x Switch,
S7/S8
8x Red LED, LED0...8
Earphone Jack, J12
VGA, J7
Red
Green
Blue
25.00 MHz
Oscillator, U43
Clock Generator
CDCI6214, U8
Pin Header, J14
I2C
EEPROM, U2
CLK_USB, CLK_PCIE, CLK_DP
SSD_RCLK
DDR4
SPI
33.00 MHz
Oscillator, U15
LPDDR4, U13
SPI Flash, U16
Micro SD socket, J9
Grove, J10
25.00 MHz
Oscillator,U7
RJ45, J4
Ethernet PHY , U6
52.00 MHz
Oscillator, U23
USB PHY
USB3320, U22
USB3 A, J11
Display Port, J3
M.2 Key M,
PCIe x1, U5

TE0802 Hardware overview

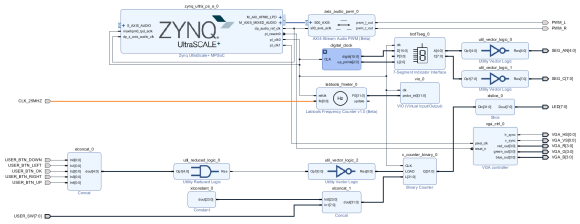
Information on IO routing and FPGA pin connections can be found in the

- schematics
- TE0802 Technical Reference Manual

Reference Design - Introduction

The provided reference design *"TE0802 test board"* we are introducing in this Getting started guide interacts with most of the peripheral on the module. It shows as an example how to connect the different parts of the module to simplify the development of your own application. You can use it for your own design but keep in mind the overall FPGA resources and power consumption before deployment. The most important steps to get it up and running from the scratch are explained on TE0802 Test Board. The Download is available here.

An overview of the components used in this reference design is illustrated in the following figure:



Block Design

Hardware Setup and Power up in QSPI-Boot mode

Preparations

1. Download the source code and configuration files for "TE0802 test_board" reference design. Ensure that your download files match your Vivado version.
2. Check the settings from DIP-Switch S1 (JTAG):

S1.1	S1.2	S1.3	S1.4
OFF	OFF	OFF	OFF

3. 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):
```

4. Press '0' and enter to start "Module Selection Guide"
    a. Select your assembly version
    b. validate selection
    c. press '2' and enter to "create Vivado project" and "create and open delivery binary folder"
5. Depending on the preferred application, continue with chapter "Linux in QSPI-Boot mode" or "'Hello Trenz' in QSPI-Boot mode"

Linux in QSPI-Boot mode

1. Connect the MicroUSB cable from your module board with your PC
2. Connect peripherals to devices
    - VGA, display port monitor
    - USB keyboard
    - ...
3. Connect the module board with the power supply (5V)
4. Power on module board
5. Program 'u-boot' application on QSPI flash

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

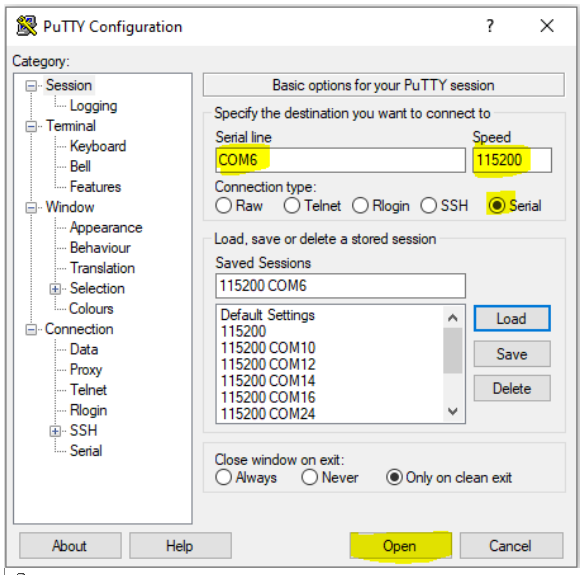
```
TE::pr_program_flash -swapp u-boot
```

6. Power off module board
7. Copy **image.ub, init.sh** and **boot.scr** on SD card (e.g. <project folder>\test_board_binaries_TE0802-02-2AEV2-A\boot_linux)
8. Switch the DIP-Switch S1 to QSPI-Boot mode

S1.1	S1.2	S1.3	S1.4
ON	OFF	OFF	OFF

9. Insert the SD card into the module board

10. Power on the module board
11. In case the QSPI Flash is loaded with the reference design, you can connect to the board with a program like *PuTTY*. Just open up a serial session with baudrate of **115200** and the right COM port (visible in Device Manager).



Terminal example

12. Boot process
    a. Zynq Boot ROM loads FSBL from QSPI into OCM,
    b. FSBL init PS, programs PL using the bitstream and loads U-boot from QSPI into DDR,
    c. U-boot loads Linux (**image.ub**) from SD into DDR
13. For usage instructions please refer to chapter Linux application

'Hello Trenz' in QSPI-Boot mode

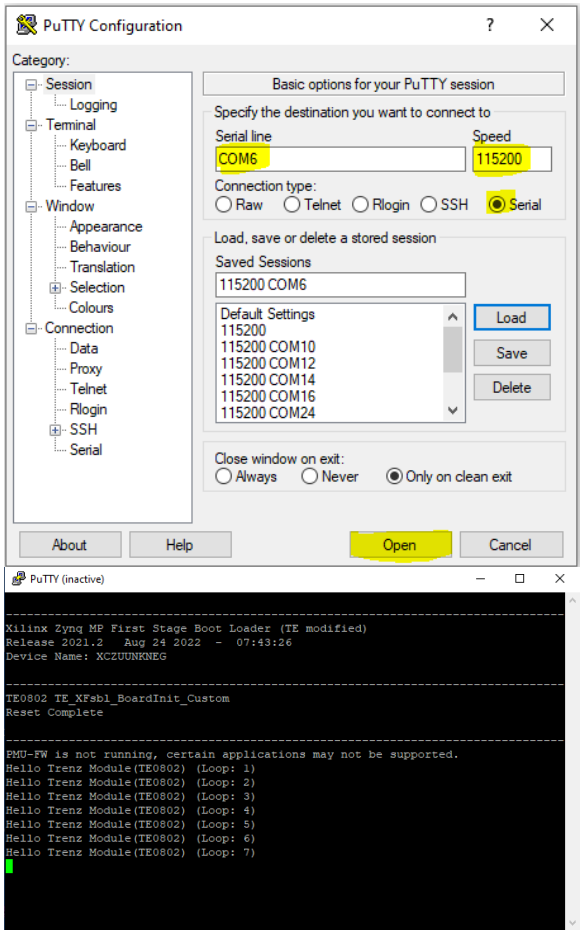
1. Connect the MicroUSB cable from your module board with your PC
2. Connect the module board with the power supply (5V)
3. Power on module board
4. Program 'hello_te0802' application on QSPI flash

run on Vivado TCL (Script programs BOOT.bin on QSPI flash)
`TE::pr_program_flash -swapp hello_te0802`

5. Switch the DIP-Switch S1 to QSPI-Boot mode:

S1.1	S1.2	S1.3	S1.4
ON	OFF	OFF	OFF

6. Restart the module board
7. In case the QSPI Flash is loaded with the reference design, you can connect to the board with a program like *PuTTY*. Just open up a serial session with baudrate of **115200** and the right COM-port (visible in Device Manager).



Terminal example

Hardware Setup and Power up in SD-Boot mode

Linux in SD-Boot mode

1. Download the source code and configuration files for "TE0802 test_board" reference design. Ensure that your download files match your Vivado version.
2. Run _create_win_setup.cmd/_create_linux_setup.sh and follow instructions on shell:

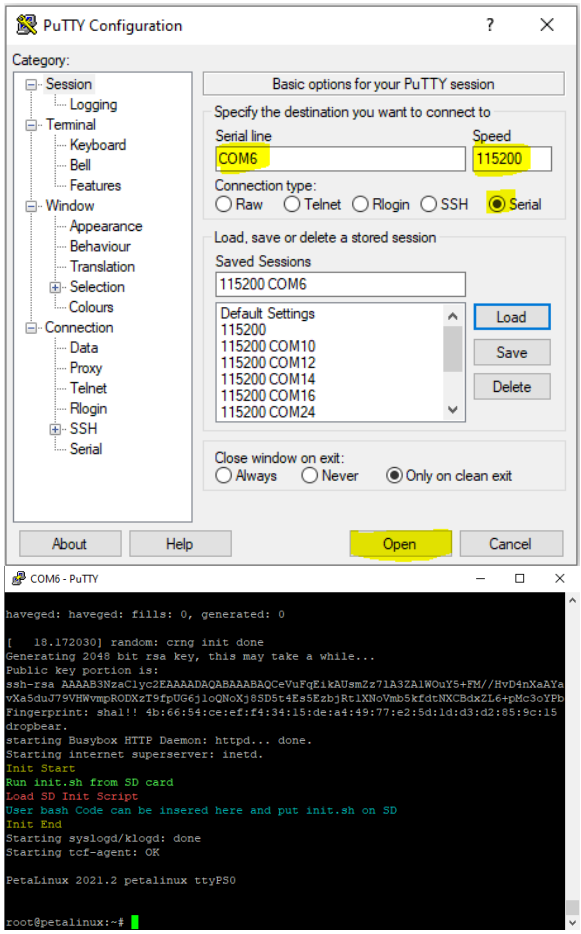
_create_win_setup.cmd/_create_linux_setup.sh

```
------------------------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):
```

3. Press '0' and enter to start "Module Selection Guide"
    a. Select your assembly version
    b. validate selection
    c. press '1' and enter to "create and open delivery binary folder"
4. Connect the MicroUSB cable from your module board with your PC
5. Connect peripherals to devices
    - VGA, display port monitor
    - USB keyboard
    - ...
6. Connect the module board with the power supply (5V)
7. Copy **BOOT.bin, image.ub, init.sh** and **boot.scr** on SD card (e.g. <project folder>\test_board_binaries_TE0802-02-2AEV2-A\boot_linux)
8. Switch the DIP-Switch S1 to SD-Boot mode

S1.1	S1.2	S1.3	S1.4
ON	ON	OFF	OFF

9. Insert the SD card into the module board
10. Power on the module board
11. You can connect to the board with a program like *PuTTY*. Just open up a serial session with baudrate of **115200** and the right COM port (visible in Device Manager).



Terminal example

12. Boot process
    a. Zynq Boot ROM loads FSBL from SD into OCM,
    b. FSBL init PS, programs PL using the bitstream and loads U-boot from SD into DDR,
    c. U-boot loads Linux (**image.ub**) from SD into DDR
13. For usage instructions please refer to chapter Linux application

Linux Application

After the Linux boot is complete, you can use the Linux shell and the connected peripherals

I2C

use linux shell
`i2cdetect -l  (Shows a list of the available I2C buses)` `i2cdetect -y -r 1  (check I2C 1 Bus)`

Real Time Clock (RTC)

use linux shell
dmesg | grep rtc (RTC check) hwclock --test

Ethernet

use linux shell
udhcpc (ETH0 check) ifconfig (shows the configuration of the network interface)

USB

use linux shell
lsusb (USB check)

PCIe (M.2 SSD)

use linux shell
lspci (PCIe check)

Audio

use linux shell
aplay /<link to mounted sd card>/<filename>.wav (e.g. aplay /run/mount/sd /<filename>.wav)

> Display Port must be connected to activate audio drivers. Use .wav or other aplay supported formate

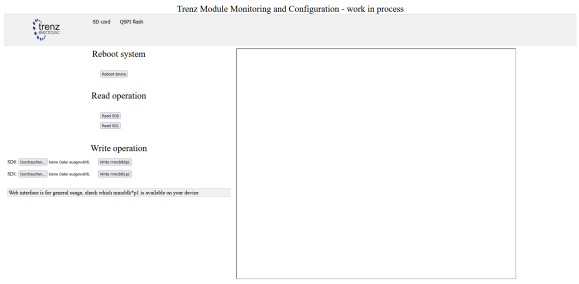
VGA

- connect VGA to monitor and adjust source (it shows test pattern)

Display Port

- second linux console output will be shown on the monitor, when boot process is finished.
- connect keyboard to TE0802 USB, to interact with the second console
    - petalinux login: root
    - password: root

Webserver

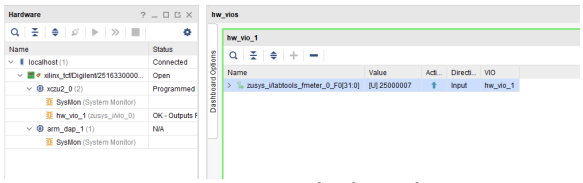


Startup Script

- If there is a start script named 'init.sh' on the SD card, it is loaded and executed shortly before the Linux boot process is completed.
- User bash code can be inserted on 'init.sh'

Monitoring input clock (25MHz)

- Open Vivado HW-Manager and add VIO signal to dashboard (*.ltx located on prebuilt folder)
- changed Value from 25MHz CLK to unsigned. Note: Frequency Counter is inaccurate and displayed unit is Hz



Vivado Hardware Manager

userLEDs

- The user LEDs indicate a binary counter, which is reset by pressing one of the cross buttons

7-Segment LCD Display

- LCD is connected to counter

DIP-Switch

- Determines the reset start value from the binary counter of userLEDs

Notes

Document Revision History

Date	Version	Description	Authors
Error rendering macro 'page-info'	Error rendering macro 'page-info'	• initial release	Error rendering macro 'page-info' Ambiguous match

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]

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]

hodoverloadingformethodjdk.proxy2444.$Proxy3572#ha

sContentLevelPermission.CannotresolvewhichmethodtoinV

o
k
e
fo
r
[n
ul
l,
cl
a
s
s
ja
v
a.
la
n
g.
St
ri
n
g,
cl
a
s
s
c
o
m
.
at
la
s
si
a
n.
c
o
nf

luence.pages.Page] due to overlapping prototypes between:

[interface com.atlassian.confluence.user.ConfluenceUser

,
cl
a
s
s
ja
v
a.
la
n
g.
St
ri
n
g,
cl
a
s
s
c
o
m
.
at
la
s
si
a
n.
c
o
nf
lu
e
n
c
e.
c
or

e.
C
o
nt
e
nt
E
nt
it
y
O
bj
e
ct
]
[i
nt
er
fa
c
e
c
o
m
.
at
la
s
si
a
n.
u
s
er
.
U
s
er
,

class java.lang.String, class com.atlassian.confluence.core.

			Content Entity Object]
--	all		**Error rendering macro 'page-info'** Am

biguousmethodoverloadingformethodjdk.proxy244.$Prox

y
3
5
7
2
#
h
a
s
C
o
nt
e
nt
L
e
v
el
P
er
m
is
si
o
n.
C
a
n
n
ot
re
s
ol
v
e
w
hi
c
h

method to invoke for [null, class java.lang.String, class com.at

lassian.confluence.pages.Page] due to overlapping prototypep

esbetween:[interface com.atlassian.confluence.user.Conf

lu
e
n
c
e
U
s
er
,
cl
a
s
s
ja
v
a.
la
n
g.
St
ri
n
g,
cl
a
s
s
c
o
m
.
at
la
s
si
a
n.
c
o

nfluence.core.ContentEntityObject][interface com.atlassian.

user.User, class java.lang.String, class com.atlassian.conf

			lu e n c e. c or e. C o nt e nt E nt it y O bj e ct]

Revision history.