1 Overview

MicroBlaze Design with Linux example.

Refer to http://trenz.org/te0713-info for the current online version of this manual and other available documentation.

1.1 Key Features

- Vitis/Vivado 2020.2
- PetaLinux
- MIG
- FLASH

1.2 Revision History

Date	Vivado	Project Built	Authors	Description
2020-12-08	2020.2	TE0713-test_board_noprebuilt-vivado_2020.2-build_9_20211210090602.zip TE0713-test_board-vivado_2020.2-build_9_20211210090545.zip	Waldemar Hanemann	• 2020.2 update • template style
2020-07-09	2019.2	TE0713-test_board_noprebuilt-vivado_2019.2-build_13_20200709071700.zip TE0713-test_board-vivado_2019.2-build_13_20200709071613.zip	John Hartfiel	• initial release

Table 1: Design Revision History

1.3 Release Notes and Know Issues

Issues	Description	Workaround	To be fixed version
petalinux-build failed on 2020.2	---	activate "Networking support" in petalinux-config -c u-boot	• implemented in vivado 2020.2

Table 2: Known Issues

1.4 Requirements

1.4.1 Software

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

Table 3: Software

1.4.2 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
TE0713-02-100-2c*	100_2c	REV02|REV01	1GB	32MB	NA	NA	NA
TE0713-02-200-2c	200_2c	REV02|REV01	1GB	32MB	NA	NA	NA

Table 4: Hardware Modules

*used as reference

Design supports following carriers:

Carrier Model	Notes
TE0701	
TE0703*	
TE0705	
TE0706	
TEBA0841	

Table 5: Hardware Carrier

*used as reference

Additional HW Requirements:

Additional Hardware	Notes
USB Cable for JTAG/UART	Check Carrier Board and Programmer for correct typ
XMOD Programmer	Carrier Board dependent, only if carrier has no own FTDI

Table 6: Additional Hardware

1.5 Content

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

1.5.1 Design Sources

Type	Location	Notes
Vivado	<project folder>/block_design <project folder>/constraints <project folder>/ip_lib	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

Table 7: Design sources

1.5.2 Additional Sources

Type	Location	Notes
--	--	--

Table 8: Additional design sources

1.5.3 Prebuilt

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

Table 9: Prebuilt files (only on ZIP with prebult content)

1.5.4 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:

- TE0713 "Test Board" Reference Design

2 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:

- Xilinx Development Tools
- 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. (optional Win OS) Generate Virtual Drive or use short directory for the reference design (for example x:\<design name>)
4. 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
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 "\prebuilt\hardware\")

```
\prebuilt\hardware\")">
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. Add Linux files (uboot.elf and image.ub) to prebuilt folder

   - copy **u-boot.elf** and **image.ub** 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

3 Launch

3.1 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

3.1.1 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

3.1.2 QSPI-Boot mode

Option for **u-boot.mcs** on QSPI Flash.

1. Connect **JTAG** and **power supply** 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". Enter the following TCL-Command inside Vivado to program the QSPI Flash.

run on Vivado TCL (Script programs u-boot.mcs on QSPI flash)

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

3.1.3 SD-Boot mode

Not used on this Example.

3.1.4 JTAG

Not used on this Example.

3.2 Usage

1. Prepare HW like described on section #Programming
2. Connect UART USB (most cases same as JTAG)
3. Select QSPI as Boot Mode

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

4. Power On PCB
   boot process

   1. FPGA Loads Bitfile from Flash,

   2. SREC Bootloader from Bitfile Firmware loads U-Boot into DDR (This takes a while),

3. U-boot loads Linux from QSPI Flash into DDR

```
SREC SPI Bootloader (TE modified): Start initialization

SREC SPI Bootloader (TE modified): SPI driver Init passed

SREC SPI Bootloader (TE modified): Serial Flash Library Init passed

SREC SPI Bootloader (TE modified): Load Image
Loading SREC image from flash @ address: 005e0000
Please wait...
```

Figure 1: Boot process takes a while, please wait...

3.2.1 Linux

1. Open Serial Console (e.g. PuTTY)
   - Speed: 9600
   - 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

3.2.2 Vivado HW Manager

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

- Monitoring:
  - MIG Calibration Done
  - Main Reset
  - MicroBlaze Reset

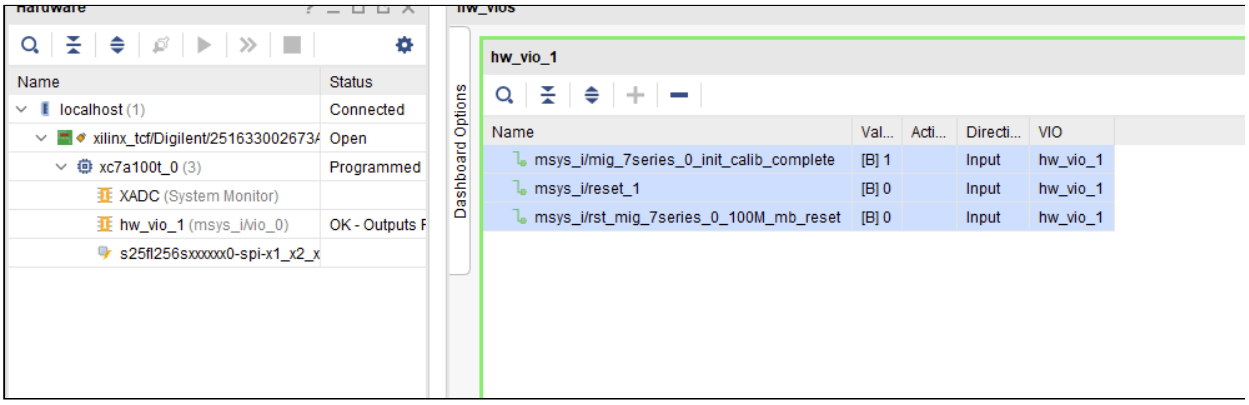

Figure 2: Vivado Hardware-Manager

4 System Design - Vivado

4.1 Block Design

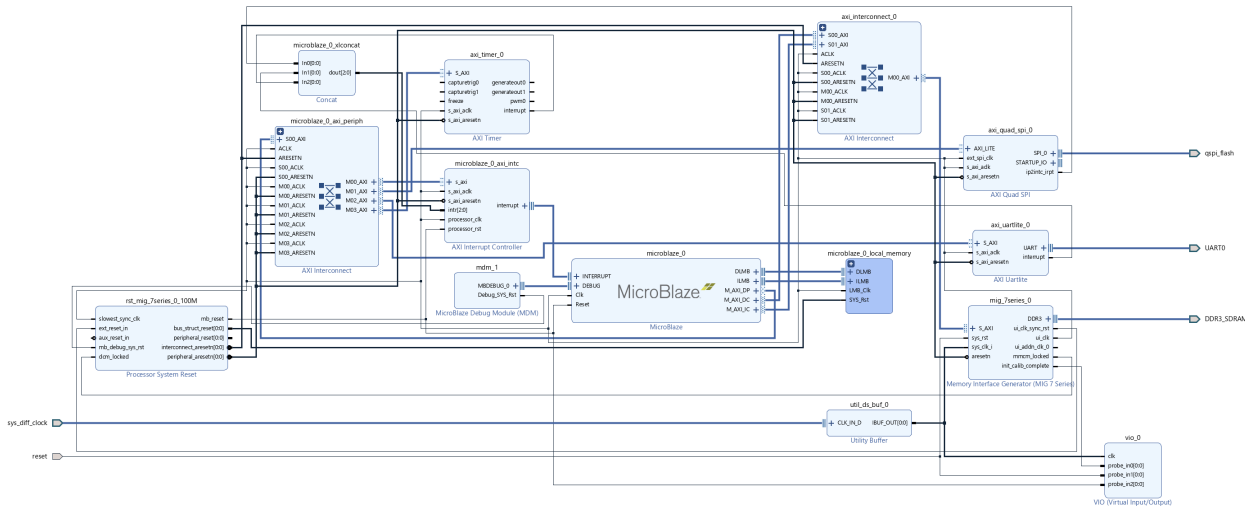

Figure 3: Block Design

4.2 Constraints

4.2.1 Basic module constraints

_i_bitgen_common.xdc

```
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 66 [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
set_property CONFIG_MODE SPIx4 [current_design]
set_property BITSTREAM.CONFIG.SPI_32BIT_ADDR YES [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property BITSTREAM.CONFIG.M1PIN PULLNONE [current_design]
set_property BITSTREAM.CONFIG.M2PIN PULLNONE [current_design]
set_property BITSTREAM.CONFIG.M0PIN PULLNONE [current_design]

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

4.2.2 Design specific constraints

_i_bitgen.xdc

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

5 Software Design - Vitis

For Vitis project creation, follow instructions from:

Vitis

5.1 Application

Template location: ./sw_lib/sw_apps/

5.1.1 srec_spi_bootloader

TE modified 2020.2 SREC

Bootloader to load app or second bootloader from flash into DDR

Descriptions:

- Modified Files: blconfig.h, bootloader.c
- Changes:
  - Add some console outputs and changed bootloader read address.
  - Add bugfix for 2018.2 qspi flash (some reinitialisation)

5.1.2 hello_te0713

Hello TE0713 is a Xilinx Hello World example as endless loop instead of one console output.

5.1.3 u-boot

U-Boot.elf is generated with PetaLinux. Vitis is used to generate the file u-boot.srec. Vivado is used to generate the file *.mcs

6 Software Design - PetaLinux

For PetaLinux installation and project creation, follow instructions from:

- PetaLinux KICKstart

6.1 Config

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

(Tipp: Search for Settings with shortcut "Shift"+"/")

Changes:

- Set flash content offset(Set kernel flash Address to 0x900000 and Kernel size to 0xA00000)
  - SUBSYSTEM_FLASH_AXI_QUAD_SPI_0_BANKLESS_PART0_SIZE = 0x5E0000
  - SUBSYSTEM_FLASH_AXI_QUAD_SPI_0_BANKLESS_PART1_SIZE = 0x300000
  - SUBSYSTEM_FLASH_AXI_QUAD_SPI_0_BANKLESS_PART2_SIZE = 0x20000
  - SUBSYSTEM_FLASH_AXI_QUAD_SPI_0_BANKLESS_PART3_SIZE = 0xA00000

6.2 U-Boot

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

Changes: (e.g. activate CONFIG via petalinux gui like [*] Networking support - - ->)

- CONFIG_ENV_IS_NOWHERE=y
- # CONFIG_ENV_IS_IN_SPI_FLASH is not set
- CONFIG_NET=y

Change platform-top.h under <project folder>/os/petalinux/project-spec/meta-user/recipes-bsp/u-boot/files:

```
#include <configs/platform-auto.h>
```

```
#define CONFIG_SYS_BOOTM_LEN 0xF000000


/* Extra U-Boot Env settings */
#define CONFIG_EXTRA_ENV_SETTINGS \
    SERIAL_MULTI \
    CONSOLE_ARG \
    TTYUL0 \
    "nc=setenv stdout nc;setenv stdin nc;\0" \
    "ethaddr=00:0a:35:00:22:01\0" \
    "autoload=no\0" \
    "sdbootdev=0\0" \
    "clobstart=0x80000000\0" \
    "netstart=0x80000000\0" \
    "dtbnetstart=0x81e00000\0" \
    "netstartaddr=0x81000000\0" \
  "bootcmd=sf probe 0 0 0 && sf read ${imageub_addr} ${imageub_flash_addr} $
{imageub_flash_size} && echo QSPI: Trying to boot image.ub at ${imageub_addr} &&
bootm ${imageub_addr}; \0" \
  "imageub_addr=0x81000000\0" \
  "imageub_flash_addr=0x0900000\0" \
  "imageub_flash_size=0x00b00000\0" \
  "loadaddr=0x81000000\0" \
    "initrd_high=0x0\0" \
    "bootsize=0x300000\0" \
    "bootstart=0x5e0000\0" \
    "boot_img=u-boot-s.bin\0" \
    "load_boot=tftpboot ${clobstart} ${boot_img}\0" \
    "update_boot=setenv img boot; setenv psize ${bootsize}; setenv installcmd
\"install_boot\"; run load_boot test_img; setenv img; setenv psize; setenv
installcmd\0" \
    "install_boot=sf probe 0 && sf erase ${bootstart} ${bootsize} && " \
        "sf write ${clobstart} ${bootstart} ${filesize}\0" \
    "bootenvsize=0x20000\0" \
    "bootenvstart=0x8e0000\0" \
    "eraseenv=sf probe 0 && sf erase ${bootenvstart} ${bootenvsize}\0" \
    "kernelsize=0xa00000\0" \
    "kernelstart=0x900000\0" \
    "kernel_img=image.ub\0" \
    "load_kernel=tftpboot ${clobstart} ${kernel_img}\0" \
    "update_kernel=setenv img kernel; setenv psize ${kernelsize}; setenv
installcmd \"install_kernel\"; run load_kernel test_crc; setenv img; setenv psize;
setenv installcmd\0" \
    "install_kernel=sf probe 0 && sf erase ${kernelstart} ${kernelsize} && " \
        "sf write ${clobstart} ${kernelstart} ${filesize}\0" \
    "cp_kernel2ram=sf probe 0 && sf read ${netstart} ${kernelstart} ${kernelsize}
\0" \
    "fpgasize=0x5e0000\0" \
    "fpgastart=0x0\0" \
    "fpga_img=system.bit.bin\0" \
    "load_fpga=tftpboot ${clobstart} ${fpga_img}\0" \
    "update_fpga=setenv img fpga; setenv psize ${fpgasize}; setenv installcmd
\"install_fpga\"; run load_fpga test_img; setenv img; setenv psize; setenv
installcmd\0" \
    "install_fpga=sf probe 0 && sf erase ${fpgastart} ${fpgasize} && " \
        "sf write ${clobstart} ${fpgastart} ${filesize}\0" \
```

```
    "fault=echo ${img} image size is greater than allocated place - partition $
{img} is NOT UPDATED\0" \
    "test_crc=if imi ${clobstart}; then run test_img; else echo ${img} Bad CRC - $
{img} is NOT UPDATED; fi\0" \
    "test_img=setenv var \"if test ${filesize} -gt ${psize}\\; then run fault\\;
else run ${installcmd}\\; fi\"; run var; setenv var\0" \
    "netboot=tftpboot ${netstartaddr} ${kernel_img} && bootm\0" \
    "default_bootcmd=bootcmd\0" \
""
```

6.3 Device Tree

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

6.4 Kernel

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

Changes:

- No changes.

6.5 Rootfs

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

Changes:

- No changes.

6.6 Applications

No additional application.

7 Additional Software

No additional software is needed.

8 Appx. A: Change History and Legal Notices

8.1 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
2022-01-03	v.3	Waldemar Hanemann	• 2020.2 release • petalinux workarounds
2020-07-09	v.1	John Hartfiel	• 2019.2 initial release
--	all	John Hartfiel, John Hartfiel , Waldemar Hanemann	--

Table 10: Document change history.

8.2 Legal Notices

8.3 Data Privacy

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

8.4 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.

8.5 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.

8.6 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.

8.7 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.

8.8 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.

8.9 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