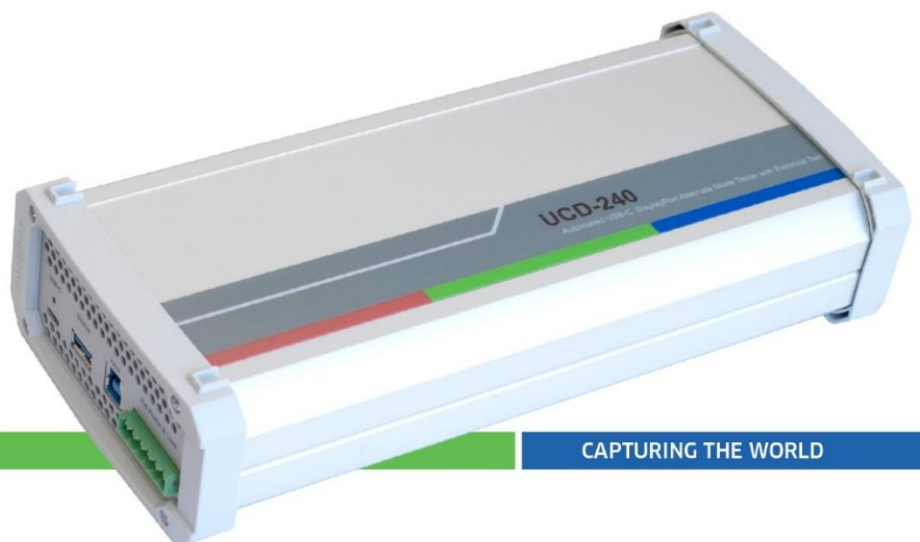




UCD-240

User Manual



CAPTURING THE WORLD

Copyright

This manual, Copyright © 2019 Unigraf Oy. All rights reserved

Reproduction of this manual in whole or in part without a written permission of Unigraf Oy is prohibited.

Notice

The information given in this manual is verified in the correctness on the date of issue. The authors reserve the rights to make any changes to this product and to revise the information about the products contained in this manual without an obligation to notify any persons about such revisions or changes.

Edition

UCD-240 User Manual, Version 5

Date: 11.2.2020

Company Information

Unigraf Oy

Piispantilankuja 4
FI-02240 ESPOO
Finland

Tel. +358 9 859 550

<mailto:info@unigraf.fi>

<https://www.unigraf.fi>

<http://www.unigraf-china.cn>

Trademarks

Unigraf, UCD and TSI are trademarks of Unigraf Oy.

DisplayPort™ and the DisplayPort™ logo are trademarks owned by the Video Electronics Standards Association (VESA®) in the United States and other countries.

USB™, USB Type-C™ and USB-C™ are trademarks of USB Implementers Forum Inc.

HDCP is a trademark of Digital Content Protection LLC.

Windows® 10, Windows® 8, Windows® 7 and Windows® XP are trademarks of Microsoft Corporation.

All other trademarks are properties of their respective owners.

Limited Warranty

Unigraf warrants its hardware products to be free from defects in workmanship and materials, under normal use and service, for twelve (12) months from the date of purchase from Unigraf or its authorized dealer.

If the product proves defective within the warranty period, Unigraf will provide repair or replacement of the product. Unigraf shall have the whole discretion whether to repair or replace, and replacement product may be new or reconditioned. Replacement product shall be of equivalent or better specifications, relative to the defective product, but need not to be identical. Any product or part repaired by Unigraf pursuant to this warranty shall have a warranty period of not less than 90 days, from the date of such repair, irrespective of any earlier expiration of original warranty period. When Unigraf provides replacement, then the defective product becomes the property of Unigraf.

Warranty service may be obtained by contacting Unigraf within the warranty period. Unigraf will provide instructions for returning the defective product.

CE Mark

UCD-200 products meet the essential health and safety requirements, is in conformity with and the CE marking has been applied according to the relevant EU Directives using the relevant section of the corresponding standards and other normative documents.

Table of Contents

	Copyright.....	2
	Notice.....	2
	Edition	2
	Company Information.....	2
	Trademarks	3
	Limited Warranty.....	3
	CE Mark	3
1.	About This Manual.....	6
	Purpose.....	6
	Product and Driver Version	6
	Notes.....	6
2.	Introduction	7
	Product Description.....	7
	UCD-240 Product Features.....	7
	UCD-240	8
	External USB-C Power Test Unit	9
3.	Unpacking.....	10
	Unigraf USB Flash Drive	10
4.	Hardware installation	11
	Unigraf Electrical Test Cable.....	11
	Connecting DUT.....	12
	Setting the DUT to DP Alt Mode	12
	Checking the Status of the DUT	12
5.	Software installation.....	13
	Installation Package.....	13
	Software Installation.....	14
6.	Firmware Update	15
	Updated Modules.....	16
7.	License Manager	17
	Licensing.....	17
	License Manager GUI	17
	Managing Licenses	18
8.	UCD Console GUI	19
	Device Selection	19
	Select Role.....	19
9.	Source DUT Testing Tab	20
	DP RX CRC test set.....	20
	Test Configuration.....	20
	Test sets available in the CRC based video test set are.....	22
	CRC Based Single Reference Frame Video Test	22
	CRC Based Single Frame Video Stability Test	23
	CRC Based Sequence of Reference Frames Test	23
	CRC Based Continuous Sequence of Reference Frames Test	23
	Link Test Set.....	25
	Link Training at All Supported Lane Counts and Link Rates	25
	USB-C Electrical Test Set.....	26
	Up Face Port CC and Vconn Test.....	27

	AUX (SBU) Lines Test	27
	DUT as Power Sink.....	28
	DUT as Power Source	30
10.	Sink dut testing tab	31
	USB-C Electrical Test Set.....	31
	Test sets available in the USBC Electrical Test Set are	31
	Up Face Port CC and Vconn Test.....	31
	AUX (SBU) Lines Test	32
	DUT as Power Sink.....	33
	DUT as Power Source	35
11.	Automated test examples	36
	Electrical Test • Up Face Port CC and VCONN	36
	Test Parameters	37
	Run Tests	39
	Test Steps.....	40
	Test Results	43
	CRC Based Video Test • Single Frame Video Stability Test.....	44
	Test Parameters and Reference Frame.....	44
	Run Tests	45
	Test Steps.....	45
	Test Results	46
12.	TSI Programming.....	47
	Operator feedback during test execution	47
	Operator feedback implementation in TSI.....	47
	Selecting which application TSI will run	47
	External application requirements.....	48
	Request parameters	49
	Request parameter details	53
	Operator Feedback configuration items	55
	Extended scripting engine.....	58
	Getting started	58
	Defining test equipment devices	58
	Commands.....	58
	Defining command macros	60
	Running TSI.EXE	61
	Output.....	68
	Running Tests.....	69
	Devices	70
13.	TestStand integration.....	71
	Integration functions.....	71
	TSI_TST_Init.....	71
	TSI_TST_RunTest	72
	TSI_TST_RunScript.....	73
	TSI_TST_Clean	74
	TSI Scripts in TestStand	74
	TestStand script syntax.....	74
	Writing a script for TSI_TST_Init.....	75
	Writing a script for TSI_TST_RunTest	76
	Writing a script for TSI_TST_RunScript	76
Appendix A.	Product Specification.....	78
	UCD-240	78
	Version History.....	79

1. ABOUT THIS MANUAL

Purpose

This guide is the User Manual for UCD-240, USB-connected interface test units for testing the features of USB Type-C interface.

UCD Console can be used in a PC with Windows® 10 Windows® 8 or Windows® 7 operating system.

The purpose of this guide is to

- Provide an overview of the product and its features.
- Provide instruction for the user on how to install the software and the drivers.
- Introduce the HW features of the UCD-240 units.
- Provide instructions for the user how to use UCD Console software.
- Provide examples on how to perform automated tests with UCD-240 test tool.

Product and Driver Version

This manual explains features found in UCD Console Software Package **1.8.6** Please consult Unigraf for differences or upgrades of previous versions.

Please consult the Release Notes document in the installation package for details of the SW and FW versions and changes to previous releases.

Notes

On certain sections of the manual, when important information or notification is given, text is formatted as follows. Please read these notes carefully.

Note	This text is an important note
------	--------------------------------

2. INTRODUCTION

Product Description

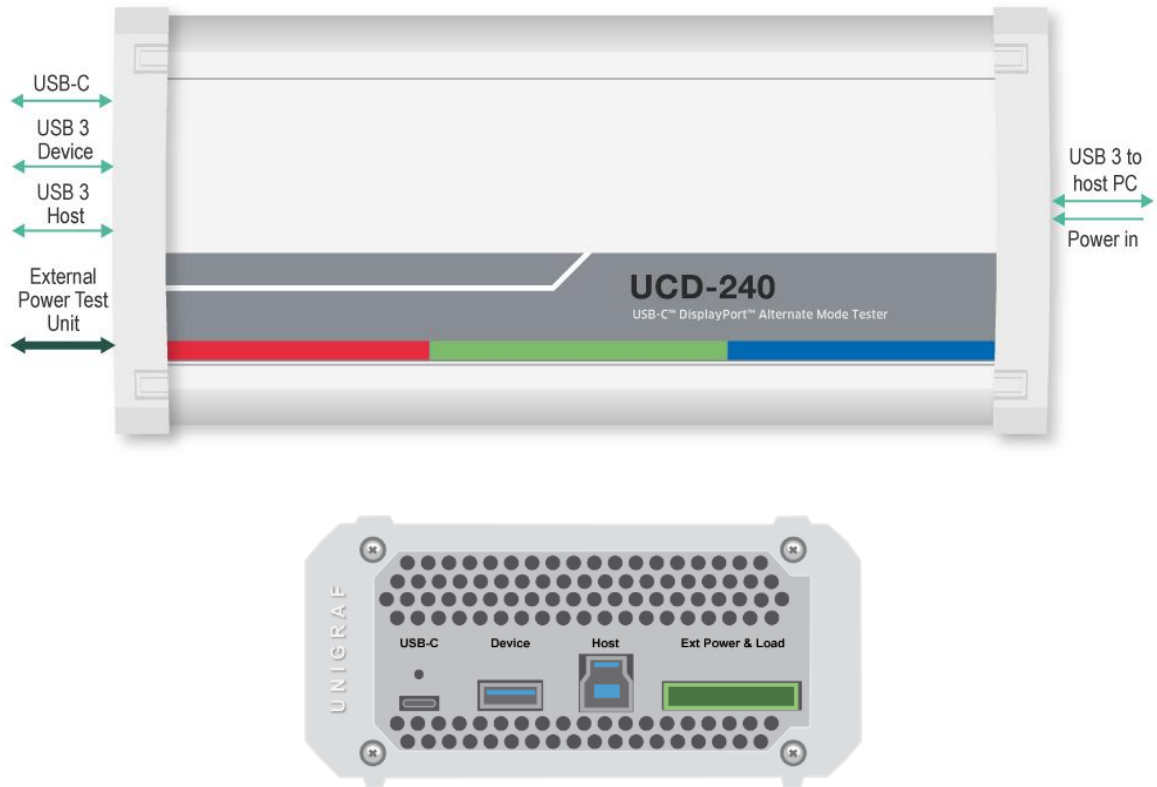
UCD-240 is a Test Automation tool for testing USB-C DisplayPort Alt Mode products.

UCD-240 Product Features

- Test unit for testing power delivery and DP Alt Mode functions of USB-C interface
- Automated tests for USB-C™ DisplayPort™ Alt Mode
- Electrical Test to verify whole USB-C connector with single cable insert
- Monitoring and control of data and power delivery roles of the USB-C interface
- High resolution DP alt mode video and audio capture up to 4K / UHD 60 Hz
- Compatible with HDCP versions 1.3 and 2.3
- Optional Power Test Unit for testing high voltage and power options
- High speed USB 3.0 host PC interface

Please refer to [Product Specifications](#) in the appendix of this document for details.

UCD-240



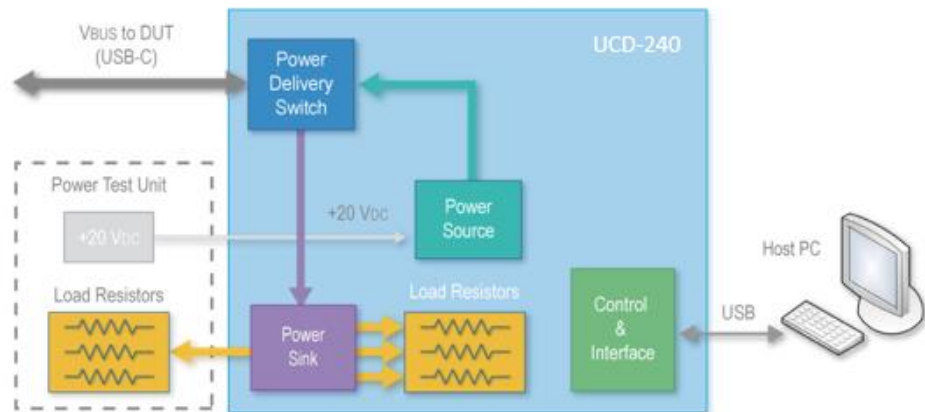
The image below describes the connections in **UCD-240** unit.

Name	Description
USB-C	USB Type-C Connection to the evaluated device (DUT)
USB 3 Device	USB Type-A connector for an auxiliary USB 3.1 "Device" for data pass-through
USB 3 Host	USB Type-B connector for an auxiliary USB 3.1 "Host" for data pass-through
External Power Test Unit	Connector for Unigraf's "USB-C External Power Test Unit"
Power in	+12 Vdc Power Supply Input
USB 3 to host PC	USB 3.0 connection to the controlling PC

Note Capturing and sourcing high resolution video modes, especially 4K video modes set stringent requirements on the video cables and connectors. Please contact Unigraf for assistance and details about evaluated cables.

Warning In order to avoid possible damage to UCD-240 unit and the PC, please **always attach the power cord (Power In) to UCD-240 unit first**, and after that connect the USB cable to your PC.

External USB-C Power Test Unit



Unigraf *External USB-C Power Test Unit* enables testing extended USB power options. Please find below a matrix describing Power Source and Power Sink options of UCD-240 internally and with attached *Unigraf External Power Test Unit*. Please find description of controlling the power options with UCD Console later in this manual.

	5 V				9 V	12 V	15 V	20 V
	0.5 A	0.9 A	1.5 A	3 A	3 A	3 A	3 A	5 A
Vbus Source capability								
UCD-240	✓	✓	✓	✓				
UCD-240 with External USB-C Power Test Unit	✓	✓	✓	✓	✓	✓	✓	✓
Vbus Sink capability								
UCD-240	✓	✓	✓	✓				
UCD-240 with External USB-C Power Test Unit	✓	✓	✓	✓	✓	✓	✓	✓

3. UNPACKING

The UCD-240 product shipment contains:

- The UCD-240 unit
- AC/DC Power supply (100 to 240 Vac 50/60 Hz input, +12 Vdc output)
- USB 3.0 compliant cable
- USB-C to USB-C test cable
- Unigraf Electrical test cable with units including Electrical Test
- USB 3.0 Memory Stick



Unigraf USB Flash Drive

The USB Flash drive delivered with UCD-240 device includes the following items:

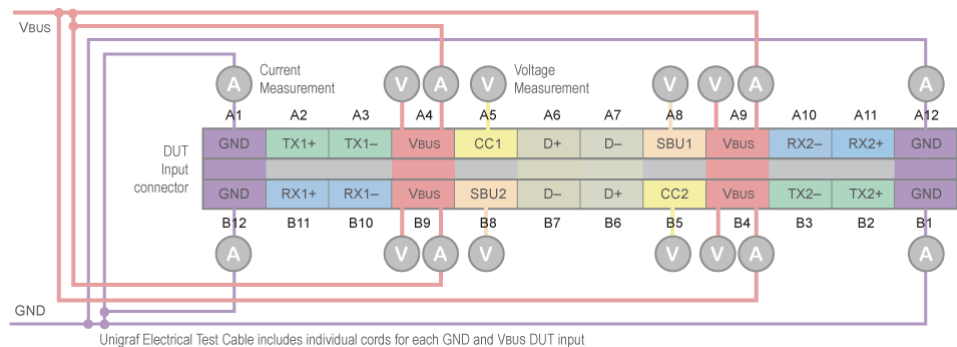
- Test Sequences and macros
- UCD-240 Software installation package
- Tutorial Videos
- UCD-240 Quick Guide
- Data Sheet
- Test Content (mp4 video)
- High/Low resolution reference images
- User Manual download link
- TSI manual download link

4. HARDWARE INSTALLATION

UCD-240 is delivered with scripts that enable installing the device without using the UCD Console GUI. Please, follow the steps introduced in the following chapters for installing Unigraf Electrical Test Cable and DUTs.

Unigraf Electrical Test Cable

Some features of the Electrical Test function are available only when using the special Electrical Test Cable provided by Unigraf. Electrical Test Cable includes extra wires enabling UCD-240 to individually measure the electrical continuity of DUT's four VBUS and four GND signals that are normally tied together at DUT's end of the cable. Please refer to the image below for clarification.



Connecting the Unigraf Electrical Test Cable

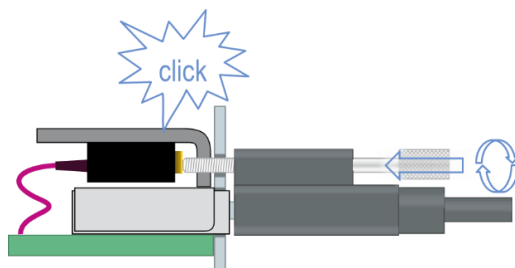
You can ensure that the Electrical Test Cable is connected correctly by running the *WaitForUnigrafCableConnected* Windows Batch File. The file can be found from the USB Flash Drive delivered with UCD-240 from the *Test Automation* folder. If the cable is connected properly, "Unigraf Test Cable CONNECTED!" text will show. You can exit the command-line by pressing any key.

```

TSI-X V1.10 [R16]. (C) 2018, Unigraf Oy. All Rights Reserved.
C:\Users\tester\UCD-240 USB Stick>tsi -r WaitForUnigrafCableConnected.txt
Connect Unigraf Test Cable to UCD....
Unigraf Test Cable CONNECTED!
C:\Users\tester\UCD-240 USB Stick>pause
Press any key to continue . . .
  
```

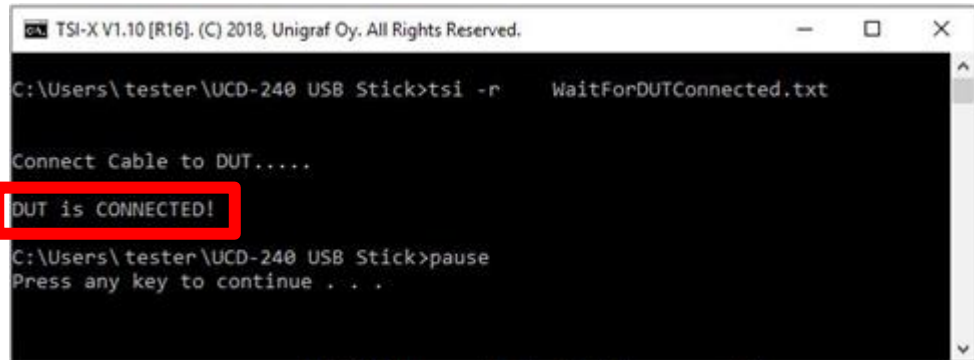
Note

When attaching Unigraf Electric Test Cable, please make sure that the finger-screw is tightened all-the-way into its hole. When tightened, the finger-screw triggers a switch indicating the presence of Unigraf Electric Test Cable.



Connecting DUT

You can ensure that the DUT is connected correctly by running the *WaitForDUTConnected* Windows Batch File. The file can be found from the USB Flash Drive delivered with UCD-240 from the *Test Automation* folder. If the cable is connected properly, a “DUT is CONNECTED!” text will show. You can exit the command-line by pressing any key.

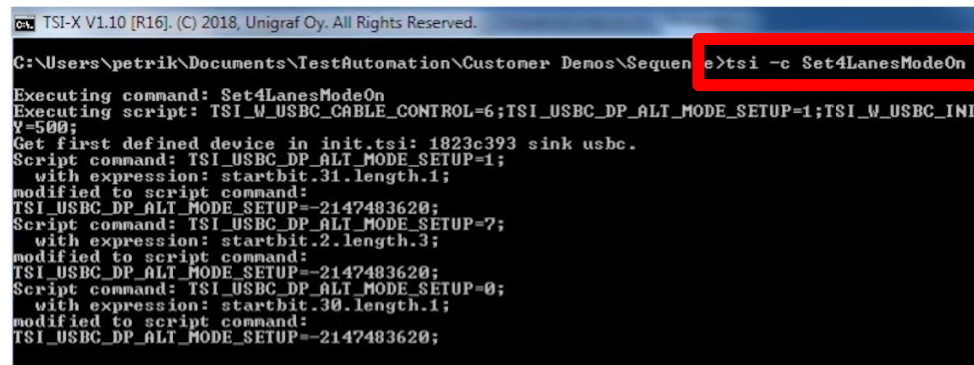


```

TSI-X V1.10 [R16]. (C) 2018, Unigraf Oy. All Rights Reserved.
C:\Users\tester\UCD-240 USB Stick>tsi -r  WaitForDUTConnected.txt
Connect Cable to DUT.....
DUT is CONNECTED!
C:\Users\tester\UCD-240 USB Stick>pause
Press any key to continue . . .
  
```

Setting the DUT to DP Alt Mode

You can set the DUT to 4 lanes DP Alt Mode by running the *Set4LanesModeOn* macro. Correspondingly you can set the DUT to 2 lanes DP Alt Mode by running *Set2LanesModeOn* macro. Detailed information about the macros can be found from the USB Flash Drive delivered with UCD-240 from the macros text file.

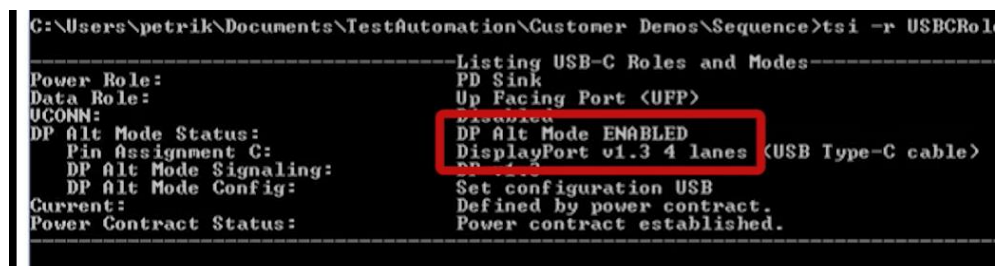


```

TSI-X V1.10 [R16]. (C) 2018, Unigraf Oy. All Rights Reserved.
C:\Users\petrik\Documents\TestAutomation\Customer Demos\Sequence>tsi -c Set4LanesModeOn
Executing command: Set4LanesModeOn
Executing script: TSI_W_USBC_CABLE_CONTROL=6;TSI_USBC_DP_ALT_MODE_SETUP=1;TSI_W_USBC_INIT
V=500;
Get first defined device in init.tsi: 1823c393 sink usbc.
Script command: TSI_USBC_DP_ALT_MODE_SETUP=1;
with expression: startbit.31.length.1;
modified to script command:
TSI_USBC_DP_ALT_MODE_SETUP=-2147483620;
Script command: TSI_USBC_DP_ALT_MODE_SETUP=7;
with expression: startbit.2.length.3;
modified to script command:
TSI_USBC_DP_ALT_MODE_SETUP=-2147483620;
Script command: TSI_USBC_DP_ALT_MODE_SETUP=0;
with expression: startbit.30.length.1;
modified to script command:
TSI_USBC_DP_ALT_MODE_SETUP=-2147483620;
  
```

Checking the Status of the DUT

You can check the status of the DUT by running the *USBCRoleStatuses* text file. The file can be found from the USB Flash Drive delivered with UCD-240. The script shows power role, data role, VCONN, DP Alt Mode and power contract status of the DUT. By running this script you can ensure that the DUT has been successfully set to the correct DP Alt Mode.



```

C:\Users\petrik\Documents\TestAutomation\Customer Demos\Sequence>tsi -r USBCRole
-----Listing USB-C Roles and Modes-----
Power Role: PD Sink
Data Role: Up Facing Port (UFP)
VCONN:
DP Alt Mode Status: DP Alt Mode ENABLED
Pin Assignment C: DisplayPort v1.3 4 lanes (USB Type-C cable)
DP Alt Mode Signaling:
DP Alt Mode Config: Set configuration USB
Current: Defined by power contract.
Power Contract Status: Power contract established.
  
```

5. SOFTWARE INSTALLATION

Installation Package

The UCD-240 software installation package is stored in the USB flash drive delivered with the device. The UCD-240 software installation package can also be obtained from Unigraf download page at www.unigraf.fi/support/download-links. For updating the software for UCD-240, please refer to the aforementioned page for downloading the latest software version.

Please log in using the following credentials before an attempt to access download page:

Username: *unigraf*

Password: *ruukintie*

The installation package is a bundle between the components needed for UCD Console and for TSI SDK. The bundle contains the following items:

- Windows drivers (installed during set up)
- UCD Console software GUI (installed during set up)
- License Manager (installed during set up)
- UCD Configuration Utility (installed during set up)
- TSI SDK
- User Manuals including this document.

In some cases, you will need to update also the firmware of the unit. If in doubt, please contact Unigraf.

Note: The software should be installed before connecting the UCD-240 unit in your PC.

Note: System administrator's privileges are required for performing the installation.

Software Installation

- ▶ Start the installation by running **Unigraf Software Bundle Setup.exe**
Once the installer has started, a welcome page is displayed. The welcome page shows the software package release version.
- ▶ Click **Next** to continue. In the next dialogs, you will be able to define the software components installed.
The next two dialogs will allow you to define the installation folder in your PC and the Start Menu folder used.
- ▶ When you are ready with the selections, click **Install** to start the installation.
- ▶ Click **Finish** to exit the installation dialog.

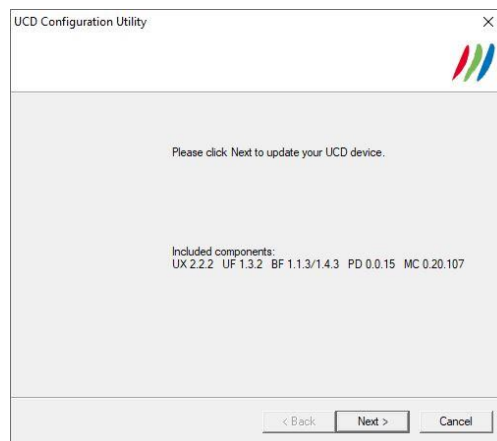
6. FIRMWARE UPDATE

UCD Configuration Utility is used to load an updated firmware to the device. As an option, UCD Configuration Utility enables the user to select the operation roles present in the UCD-240 unit. The utility configures a firmware set for the selected operation roles and programs the firmware set to the device. Please contact Unigraf for details.

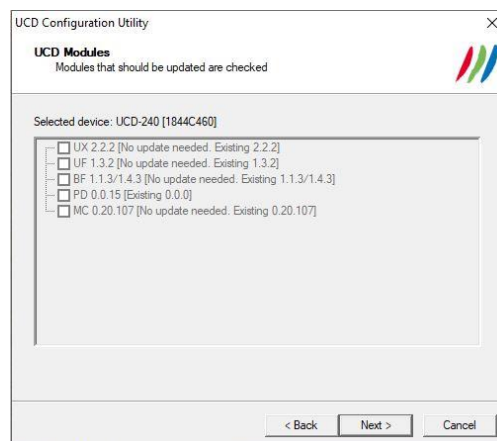
To update the firmware or create a new configuration on a UCD-300 device, please perform the following steps:

- ▶ Connect the UCD-240 unit to a power supply and connect the USB cable.
- ▶ Open **UCD Console**. Select **Tools > Firmware update**.

You can launch the utility alternatively by running the **UCD Firmware Configuration** in Start Menu > Unigraf / UCD-240 /.



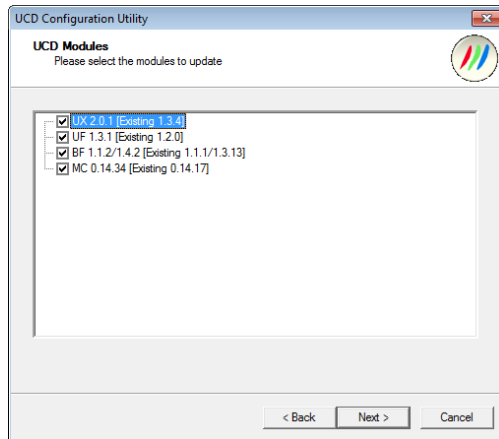
The first page of the utility indicates the firmware component versions present in the package. Please click **Next**.



- ▶ From the list of connected UCD-200 devices please select the one that you want to update. Click **Next**.

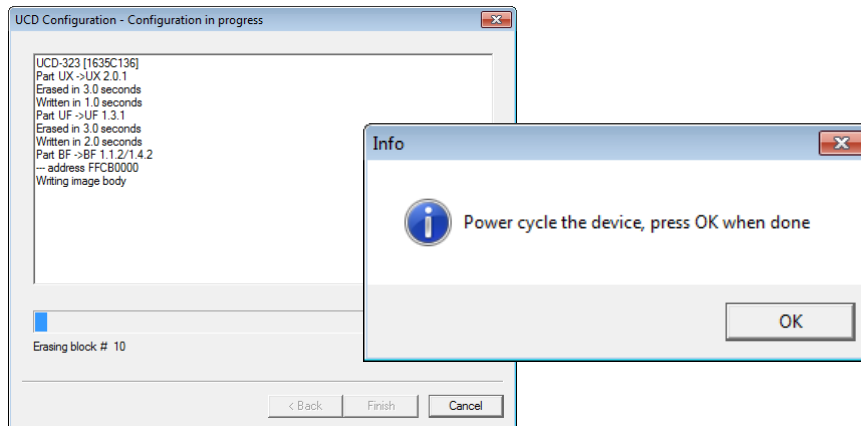
Updated Modules

The tool prompts you for selection of the firmware modules to be updated. It compares the modules in the selected device and omits the ones that are the same.



- ▶ When you are done, click **Next**.

While the configuration is in progress you will be asked to power cycle the device between the parts.



Note: The procedure may take several minutes depending on the speed of the USB connection of the host PC.

- ▶ When completed, power cycle the device.

Note: The new firmware will be taken in use only when the device is powered up next time.

7. LICENSE MANAGER

Licensing

The features of UCD Console GUI are divided into groups based on the target use of the device. Some basic features can be used without licenses. Advanced feature groups have their dedicated licenses that open the related part of the GUI or enable the related control.

Unigraf licenses are provided as strings of characters, **License Keys**. Each License Key enables a dedicated function in one device. Each device has its dedicated **Seed Number**. Each **License Key** is tied to one **Seed Number**. License Keys can be freely used in any number of PCs

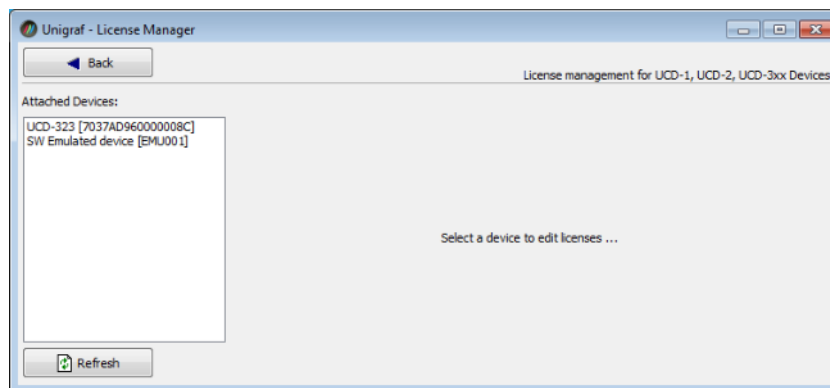
License keys are managed with Unigraf **License Manager**. By default, shortcut to Unigraf License Manager can be found in Start Menu under: **All programs/Unigraf/ TSI**.

Please click **Yes** in the first dialog. License Manager can be run only with Administrator rights.

Note: System administrator's privileges are required for accessing the licenses.

License Manager GUI

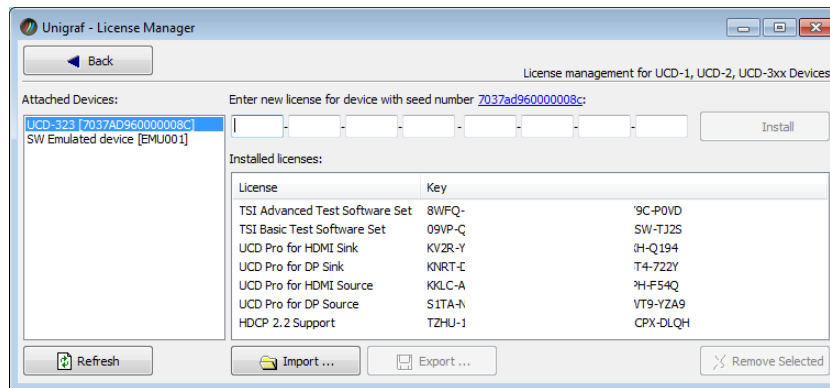
Once the application starts up, it will look for any licensing enabled Unigraf device. If no suitable device families are detected, License Manager will exit. Please first select one of the available device families by clicking one of the device family selector buttons.



In the list of Attached Devices please select the device in question. You can find the *serial number* and the *seed number* of your device in a sticker attached to the bottom of your device.

The **Back** button will bring you back to the device family selection screen. The **Refresh** button will re-scan the system for installed hardware.

Managing Licenses



Seed Number

Each license is tied to a hardware unit with the help of the **Seed Number**. Each unit has a unique Seed Number. You can find the Seed Number of the selected unit in the top of the dialog.

The Seed Number link in the dialog will allow you to copy & paste the Seed Number of the selected device for e.g. ordering Licenses.

Adding New License Keys

To add a new license key for a device, please enter the characters from the license sticker to the boxes provided. The License Manager will automatically move the caret across the edit boxes as you type. If you have the key in text format, copy it and paste to the leftmost box.

Once the license key is fully entered, click the **Install**. The license is authenticated and if it is valid, the license will appear in the list of installed licenses. If the key fails to authenticate, an error message is displayed. If this happens, please make sure that you have typed the key correctly and that the seed number on the license key sticker matches the seed number displayed seed number for the device.

Please note that to avoid confusion, some letters will never appear in a license key because they resemble numbers: For example, capital 'G' and number '6' are very similar when printed with small font. When in doubt, use numbers.

Also, please notice, that characters that can't be part of valid license key are not accepted as input. When appropriate an automatic conversion is applied while typing: For example, lower case letters are converted to upper case automatically.

Managing Installed Licenses

The Installed licenses list shows all currently installed licenses for the currently selected device. The list shows the actual license key, and what that key unlocks.

Remove Selected will uninstall selected licenses. To uninstall a license, click on the license and then click the Remove Selected button.

Export will allow you to save all installed licenses for the currently selected device into an INI file for backup and distribution to other PCs. Please notice that licenses from multiple devices can be exported into the same INI file.

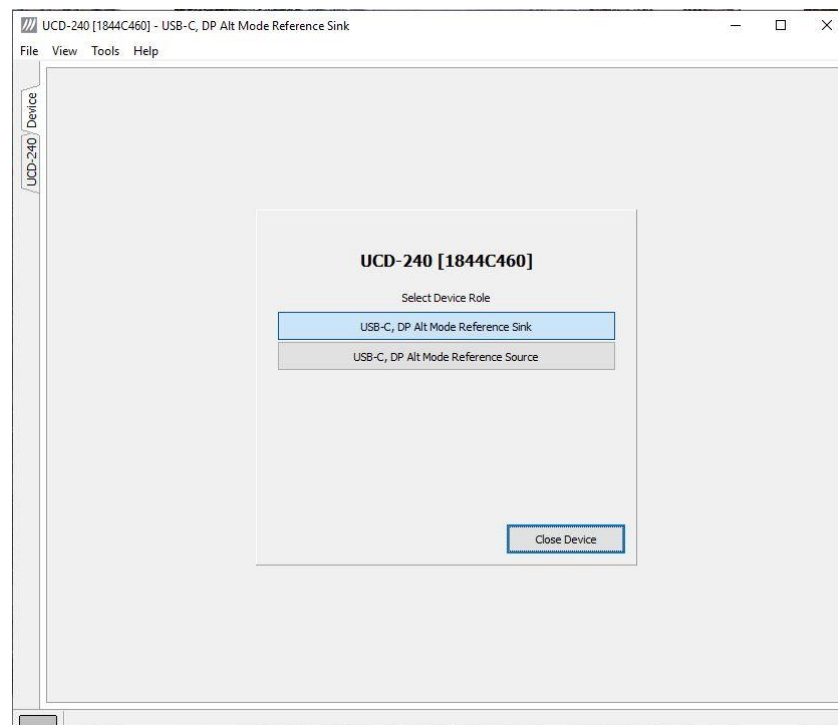
Import will install licenses from an INI file for the currently selected device.

8. UCD CONSOLE GUI

UCD Console is the graphical user interface (GUI) of UCD-240. It provides the user access to set and save all necessary test parameters used in automated tests.

Device Selection

You can find the shortcut of UCD Console by default under Start Menu path: **All programs/Unigraf/UCD Console**. Once UCD Console GUI is launched the dialog provides a list of UCD devices connected in the PC. Please select your target device by clicking on the appropriate button. If you cannot find your device in the list, please confirm the power and USB connection to the device and click the **Rescan ...** button.



Select Role

The use of UCD-240 devices with UCD Console is divided in display interface specific roles. The structure of UCD Console varies between roles by having a varying set of tabs dedicated to functionalities available in the enabled role. Please find a detailed description of each role in the later chapters of this manual.

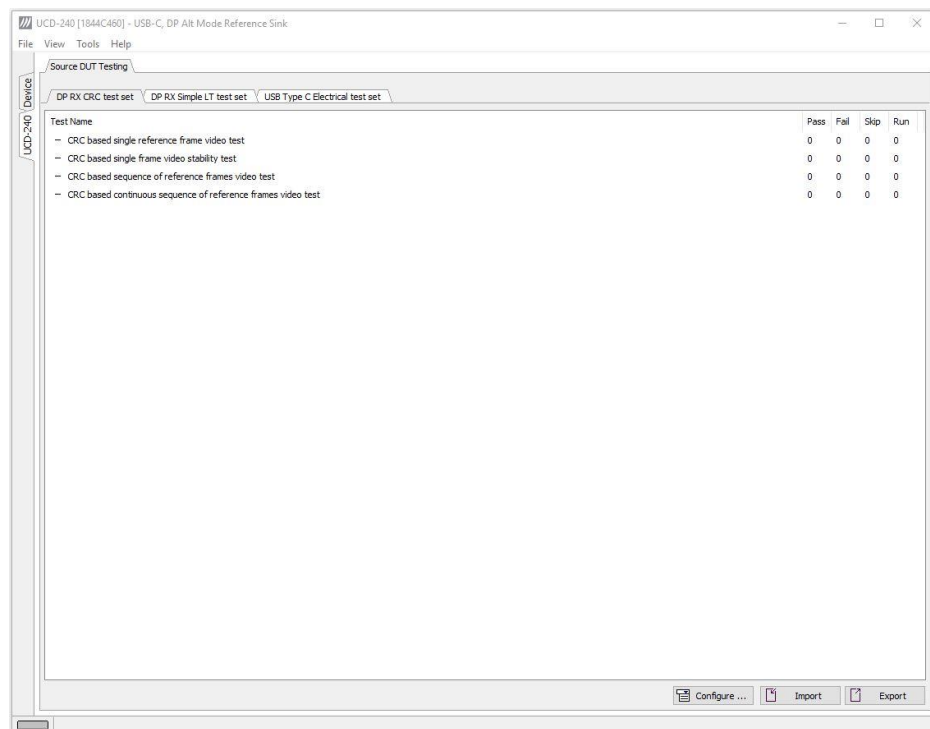
9. SOURCE DUT TESTING TAB

In UCD Console GUI *Source DUT Testing* tab you can configure and save tests for running automated tests for USB-C Source DUTs. Please refer the TSI Reference in the chapter 12 of this document to get a full definition of the TSI Test Cases, the parameters and the reporting.

Available ready-made test sets in the Source DUT testing tab are

- DP RX CRC test set
- DP RX Simple LT test set
- USB Type C Electrical test set

DP RX CRC test set

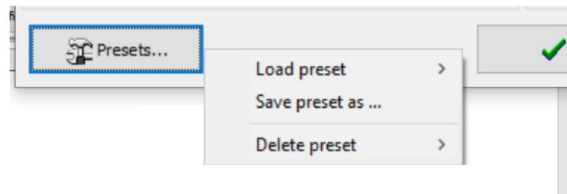
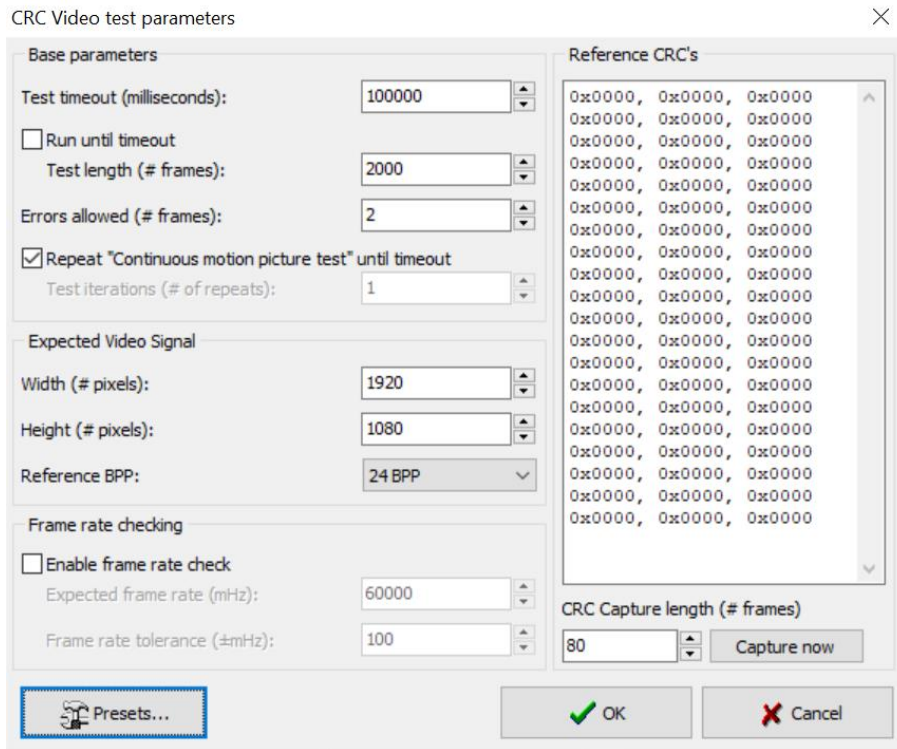


Test Configuration

By clicking the **Configure...** button you can modify the test parameters. Click **OK**, to apply the changes. By clicking the **Presets...** button you can create preset test configurations and save them by clicking **Save preset as....** You can load the saved presets by clicking **Load preset** and selecting the preset you want to load.

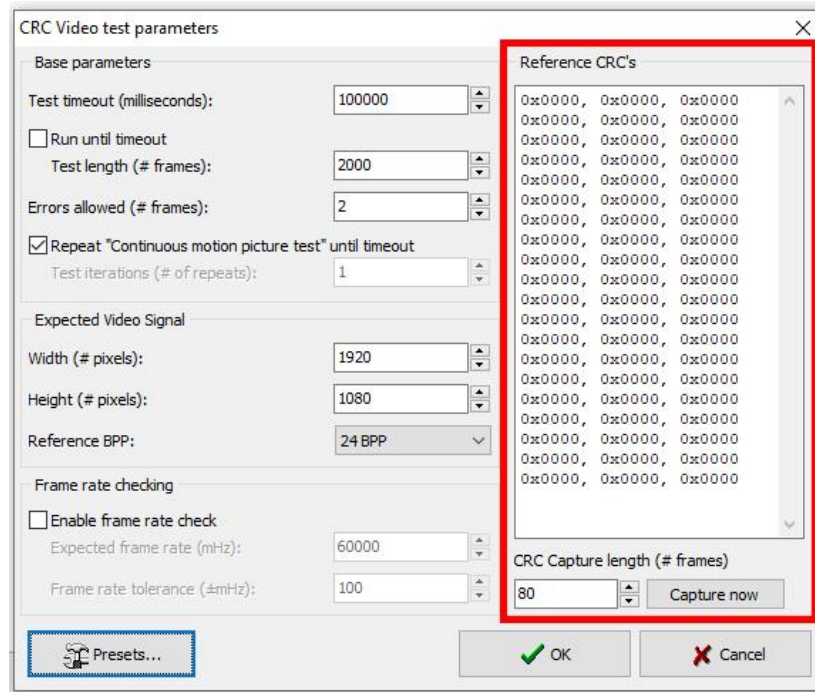
To save your test configuration click the **Export** button. You can load previously saved test configuration by clicking the **Import** button.

Please, note that you have to select a destination for the saved presets in **Tools > Options**.



Capturing a Reference frame for the CRC tests

Before running the CRC based video test set a Reference CRC must be captured. Set the number of frames to capture and press ‘Capture now’ button. Please note that the DUT needs to be set to DP Alt Mode before the capturing can be done. For instructions to set the DUT to DP Alt Mode, please refer to chapter 4 in this manual. Also, notice that the captured reference frame should not include any items that might change during the testing procedure such as time or batter level indicators. After the capturing has been started, automated tests can be run from the command-line.



Note: Please note that any change in the captured frame (e.g. clock) will result as a failed test.

Test sets available in the CRC based video test set are

- CRC Based Single Reference Frame Video Test
- CRC Based Single Frame Video Stability Test
- CRC Based Sequence of Reference Frames video Test
- CRC Based Continuous Sequence of Reference Frames Test

CRC Based Single Reference Frame Video Test

The test compares captured frames to a provided reference.

TE compares the video mode (Frame Width, Height, BPP and optionally Frame rate) to provided parameters and after that captures frames and compares the CRC (check sum) of their three color components to the provided reference until the number of bad frame limit provided is detected or the provided total number of frames is reached.

The test is judged FAIL if video mode does not match or the number of bad frames is exceeded.

The test optionally captures the failed frames as bitmap images and stores them into the hard disc.

Parameters in use

- Test Timeout (default 100 000 ms)
- Total number of frames (default 2 000 ms)
- Number of bad frames allowed (default 2)
- Reference width (default 1920)
- Reference height (default 1080)
- Reference BPP (default 24)
- Expected frame rate (mHz)

- Frame rate tolerance (mHz)
- Reference CRCs (R, G, B)

CRC Based Single Frame Video Stability Test

The test verifies that the captured video is stable.

TE captures a frame and sets the CRC of its color components as reference. After that TE captures frames and compares their CRC (check sum) to the reference until the number of bad frame limit provided is detected or the provided total number of frames is reached.

The test is judged FAIL if the number of bad frames is exceeded.

The test optionally captures the failed frames as bitmap images and stores them into the hard disc.

Parameters in use

- Test Timeout (default 100 000 ms)
- Total number of frames (default 2 000 ms)
- Number of bad frames allowed (default 2)

CRC Based Sequence of Reference Frames Test

The verifies that a sequence of frames is captured in the right order.

TE compares the video mode (frame Width, Height, BPP and optionally Frame rate) to provided parameters. After that captures frames to find a frame with matching CRC (check sum) of their three color components to the first provided reference. After the first matching CRC is found it compares the CRC of the following frames until the Number of frames tested parameter is reached.

The test is judged FAIL if video mode does not match, the first frame in the list is not found or the CRC of the following frames do not match the provided list.

The test optionally captures the failed frames as bitmap images and stores them into the hard disc.

Parameters in use

- Test Timeout (default 100 000 ms)
- Number of frames to be tested (default 20)
- Reference width (default 1920)
- Reference height (default 1080)
- Reference BPP (default 24)
- Expected frame rate (mHz)
- Frame rate tolerance (mHz)
- Reference CRCs (R, G, B)

Note: Please note that in order for the TE to maintain the sequence, all CRCs in the reference frame list should be different.

CRC Based Continuous Sequence of Reference Frames Test

The test verifies that a sequence of frames is captured in the right order many times repeatedly.

TE compares the video mode (frame Width, Height, BPP and optionally Frame rate and Color format) to provided parameters. After that captures frames to find a frame with matching CRC (check sum) of their three color components to the first provided reference. After the first matching CRC is found it compares the CRC of the following frames until the Number of frames tested parameter is reached. After that it resets the list and starts from the first CRC. The list is repeated until timeout or until the provided number of repetitions is reached.

The test is judged FAIL if video mode does not match, the first frame in the list is not found or the CRC of the following frames do not match the provided list.

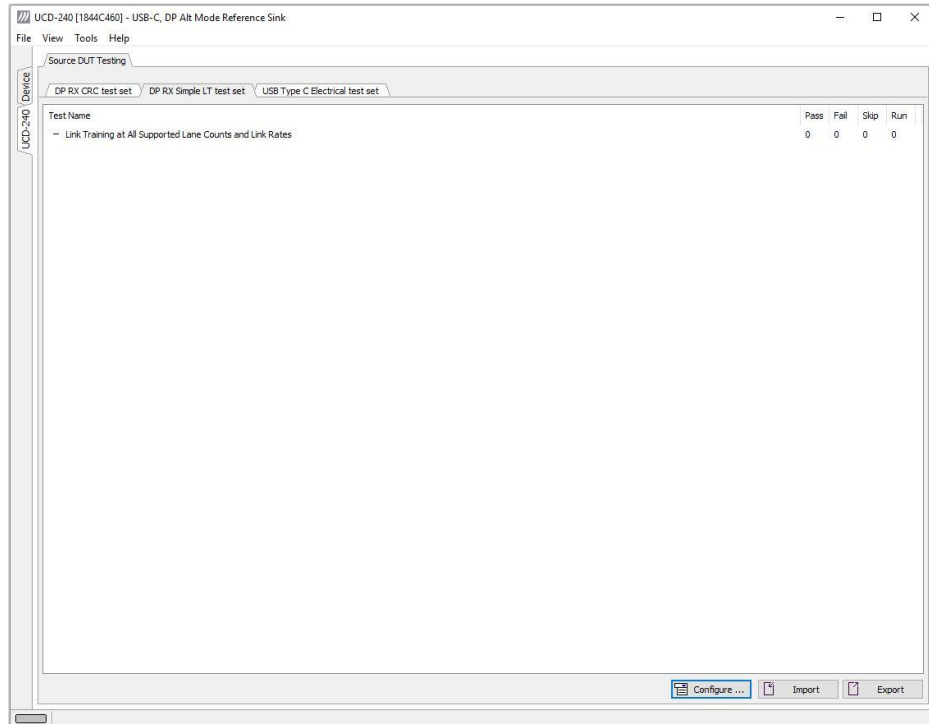
The test optionally captures the failed frames as bitmap images and stores them into the hard disc.

Parameters in use

- Test Timeout (default 100 000 ms)
- Number of frames to be tested (default 20)
- Number of iterations
- Reference width (default 1920)
- Reference height (default 1080)
- Reference BPP (default 24)
- Expected frame rate (mHz)
- Frame rate tolerance (mHz)
- Expected color format
- Reference CRCs (R, G, B)

Note: Please note that in order for the TE to maintain the sequence, all CRCs in the reference frame list should be different.

Link Test Set



To save your test configuration click the **Export** button. You can load previously saved test configuration by clicking the **Import** button.

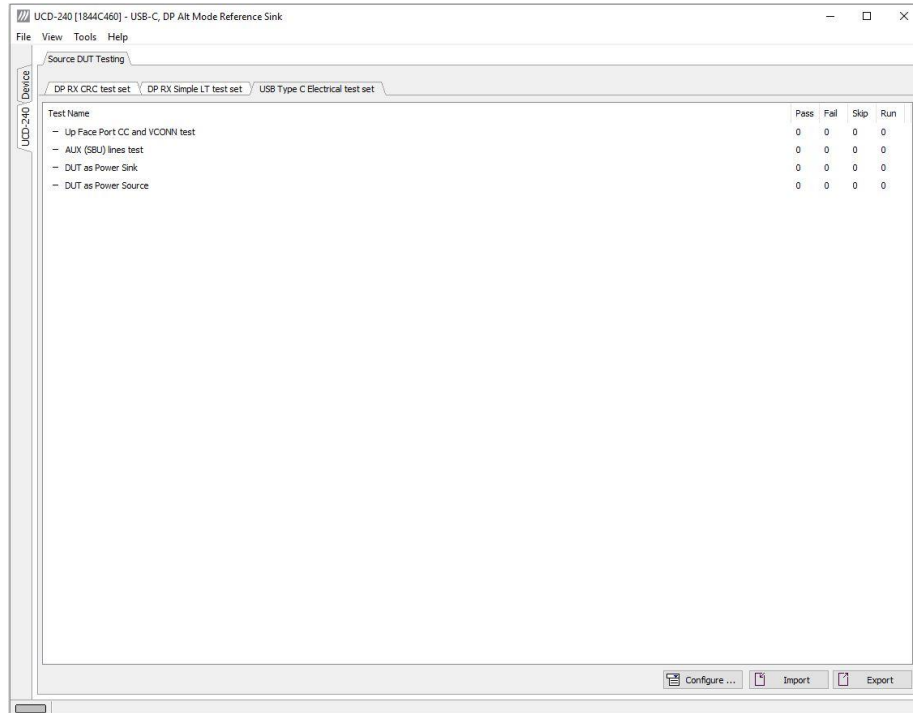
Link Training at All Supported Lane Counts and Link Rates

Test requests link training on all supported lane counts and link rates. Each link training must be successfully completed in order to pass the test.

Parameters in use

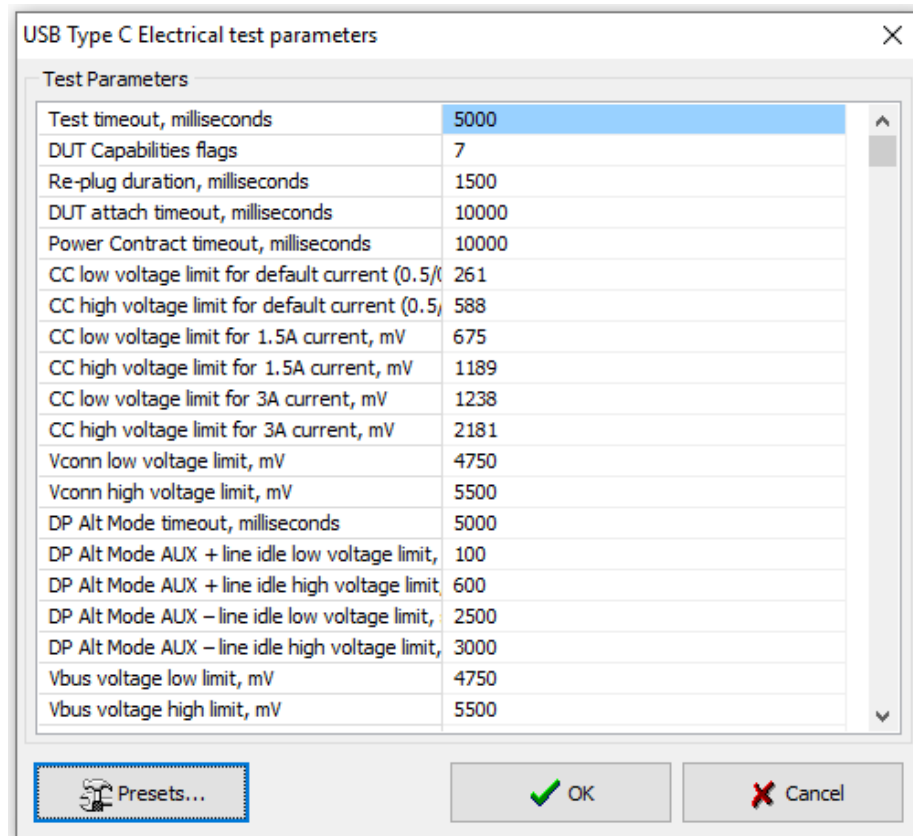
- Test Timeout (default 5 000 ms)
- Max lane count supported by DUT (default 4)
- Max lane rate supported by DUT as multiple of 0.27 Gbps.
(valid settings 6, 10 and 20; default 20)
- Long HPD pulse duration (default 1 000 ms)
- Link training start timeout (default 5 000 ms)
- Delay between test cycles (default 3 000 ms)

USB-C Electrical Test Set



To save your test configuration click the **Export** button. You can load previously saved test configuration by clicking the **Import** button.

You can modify the test parameters by clicking the **Configure...** button. Click the value you would like to change and type in the desired value. You can save the test parameters by clicking the **Presets...** button. Click **OK** to save the changes and exit the window. Click **Cancel** to exit the window without saving the parameters.



Up Face Port CC and Vconn Test

This test verifies operation of CC lines for short-circuit and open-circuit failures, and that hardware directly related to CC lines is working properly. During the test, TE will operate as Type-C UFP device.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event.

After the re-plug event, Ra is connected to CC2 and Rd is connected to CC1. DUT is expected to have Rp, or a current source applied to both CC1 and CC2 lines. The impedance of DUT's Rp resistor, or current source must be adjusted so that the voltage drop on Rd resistor in TE is within one of the voltage ranges defined by the provided parameters.

TE will measure the voltage drop on Rp.

Once DUT has started to provide Vconn on CC2, TE will measure the voltage present on CC2. After that TE will do a cable-flip and repeat the steps as above.

For a PASS result, measured values from CC1, CC2 and Vconn must be within ranges defined by the provided parameters.

Note	Configuration items for this test should be programmed with averaged values from several "golden sample" DUT's.
-------------	---

Important	In order to run this test with UCD-240, Unigraf Electrical Test Cable must be used and Electrical Testing feature enabled with a corresponding license.
------------------	--

Parameters in use

- Test timeout (default 5 000 ms)
- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- CC low voltage limit for default current (0.5/0.9A) (default 261 mV)
- CC high voltage limit for default current (0.5/0.9A) (default 588 mV)
- CC low voltage limit for 1.5A current (default 675 mV)
- CC high voltage limit for 1.5A current (default 1 189 mV)
- CC low voltage limit for 3A current (default 1 238 mV)
- CC high voltage limit for 3A current (default 2 181 mV)
- Vconn low voltage limit (4 750 mV)
- Vconn high voltage limit (5 500 mV)

AUX (SBU) Lines Test

This test verifies operation of SBU lines for short-circuit and open-circuit failures and that hardware directly related to SBU lines is working properly. During the test the TE will operate as Type-C UFP device. For this test, DUT must support DisplayPort Alternate Mode.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event and waits for DUT to enter DP Alternate mode.

Once DUT has entered the DP alternate mode, TE will measure voltage levels on SBU1 (AUX+) and SBU2 (AUX-) lines. Please notice that if TE is acting as DP Sink, it will de-assert HPD signal to keep AUX bus at IDLE state during the voltage measurements.

Once the voltages are measured, TE will do a cable-flip and repeat the steps as above.

For a PASS result, the measured voltages must be within the ranges defined by the provided parameters.

Note Parameters for this test should be programmed with averaged values from several “golden sample” DUT’s.

Important In order to run this test with UCD-240, **Unigraf Electrical Test Cable** must be used and Electrical Testing feature enabled with a corresponding license.

Parameters in use

- Test timeout (default 5 000 ms)
- DUT Capabilities flags (default 7)

Bits	Description
0	DUT Support for DisplayPort Alt Mode (1 Yes, 0 No)
1	DUT can act as power source (1 Yes, 0 No)
2	DUT can receive power from TE (1 Yes, 0 No)
31:0	Reserved

- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- DP Alt Mode timeout (default 5 000 ms)
- DP Alt Mode AUX+ line idle low voltage (default 100 mV)
- DP Alt Mode AUX+ line idle high voltage (default 600 mV)
- DP Alt Mode AUX– line idle low voltage (default 2 500 mV)
- DP Alt Mode AUX– line idle high voltage (default 3 000 mV)

DUT as Power Sink

This test verifies operation of Vbus and GND lines for short-circuit and open-circuit failures. The test is performed using mandatory PDO for power contract. During the test, the TE will operate as power source, and advertise only **vSafe5V** for power contract. In order to run this test, the DUT must support Power Sink role.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event and waits for Power Contract to be established.

After Power Contact has been established TE will wait for the delay stated in *Measurement delay* parameter and measures the current in Vbus and GND lines and the voltage of Vbus. The purpose of the delay is to allow the DUT time to stabilize its power consumption.

Note The test assumes that the current flows through the four separate Vbus and GND lines evenly. Since current measurements are done sequentially, any variance in DUT power consumption during the measurement can cause this test to fail.

Total currents are calculated for Vbus, and for GND. The difference in the measured values of the four connections may not exceed the programmed deviation limits.

For PASS result, the measured Vbus voltage, Vbus current and GND line current must be within the ranges defined by provided parameters.

Important In order to run this test with UCD-240, **Unigraf Electrical Test Cable** must be used and Electrical Testing feature enabled with a corresponding license.

Parameters in use

- Test timeout (default 5 000 ms)
- DUT Capabilities flags (default 7)

Bits	Description
0	DUT Support for DisplayPort Alt Mode (1 Yes, 0 No)
1	DUT can act as power source (1 Yes, 0 No)
2	DUT can receive power from TE (1 Yes, 0 No)
31:0	Reserved

- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- Power Contract timeout (default 10 000 ms)
- Vbus voltage low limit (default 4 750 mV)
- Vbus voltage high limit (default 5 500 mV)
- Vbus current deviation (between wires) (default 100 mA)
- Return (GND) current deviation (between wires) (default 100 mA)
- Measurement delay (default 2 000 ms)
- Minimal current (default 20 mA)

DUT as Power Source

This test verifies operation Vbus and GND lines for short-circuit and open-circuit failures. During the test, TE will operate as power sink and selects only *vSafe5V* PDO for power contract. In order to run this test, the DUT must support Power Source role.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event and waits for Power Contract to be established

After Power Contact has been established TE will wait for the delay stated in *Power measurement delay* parameter and measures the current in Vbus and GND lines and the voltage of Vbus. The purpose of the delay is to allow the DUT time to stabilize its power consumption.

Note	The test assumes that the current flows through the four separate Vbus and GND lines evenly. Since current measurements are done sequentially, any variance in DUT power consumption during the measurement can cause this test to fail.
-------------	--

Total currents are calculated for Vbus, and for GND. The difference in the measured values of the four connections may not exceed the programmed deviation limits.

For PASS result, the measured Vbus voltage, Vbus current and GND line current must be within the ranges defined by provided parameters.

Important	In order to run this test with UCD-240, Unigraf Electrical Test Cable must be used and Electrical Testing feature enabled with a corresponding license.
------------------	--

Parameters in use

Parameters in use

- Test timeout (default 5 000 ms)
- DUT Capabilities flags (default 7)

Bits	Description
0	DUT Support for DisplayPort Alt Mode (1 Yes, 0 No)
1	DUT can act as power source (1 Yes, 0 No)
2	DUT can receive power from TE (1 Yes, 0 No)
31:0	Reserved

- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- Power Contract timeout (default 10 000 ms)
- Vbus voltage low limit (default 4 750 mV)
- Vbus voltage high limit (default 5 500 mV)
- Vbus current deviation (between wires) (default 100 mA)
- Return (GND) current deviation (between wires) (default 100 mA)
- Measurement delay (default 2 000 ms)

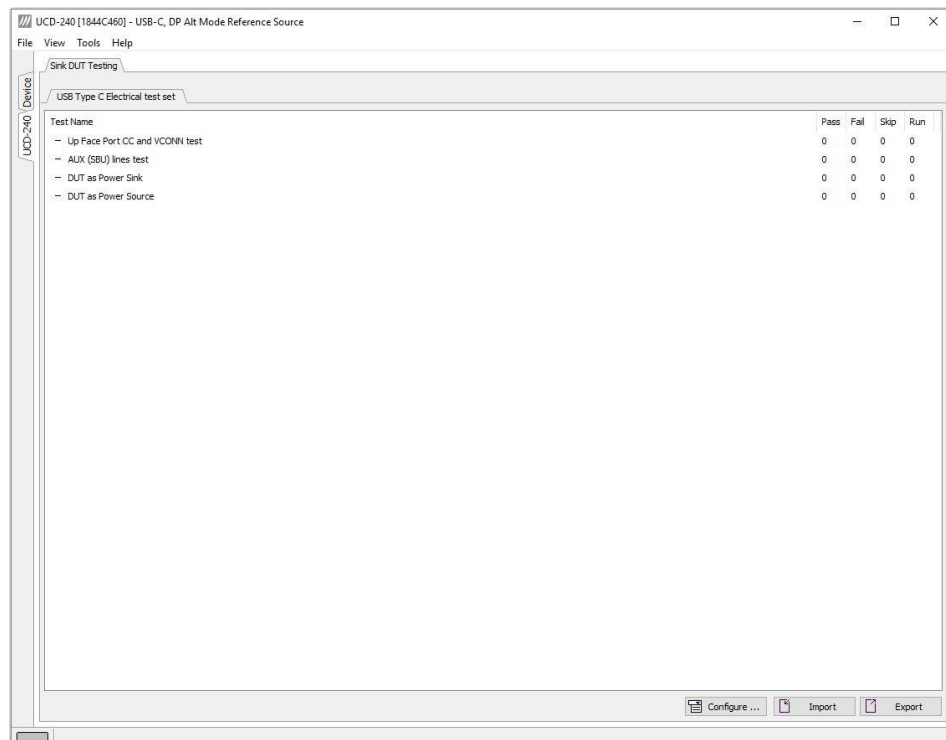
10. SINK DUT TESTING TAB

In UCD Console GUI *Sink DUT Testing* tab you can configure and save tests for running automated tests for USB-C Sink devices. Please refer the TSI Reference in the chapter 7 of this document to get a full definition of the TSI Test Cases, the parameters and the reporting.

Available ready-made test sets in the Sink DUT testing tab are

- USB-C Electrical Test Set

USB-C Electrical Test Set



To save your test configuration click the **Export** button. You can load previously saved test configuration by clicking the **Import** button.

For detailed information about setting the test configuration, please refer to page 26 of this manual.

Test sets available in the USBC Electrical Test Set are

- Up Face Port CC and Vconn Test
- AUX (SBU) Lines Test
- DUT as Power Sink
- DUT as Power Source

Up Face Port CC and Vconn Test

This test verifies operation of CC lines for short-circuit and open-circuit failures, and that hardware directly related to CC lines is working properly. During the test, TE will operate as Type-C UFP device.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event.

After the re-plug event, Ra is connected to CC2 and Rd is connected to CC1. DUT is expected to have Rp, or a current source applied to both CC1 and CC2 lines. The impedance of DUT's Rp resistor, or current source must be adjusted so that the voltage drop on Rd resistor in TE is within one of the voltage ranges defined by the provided parameters.

TE will measure the voltage drop on Rp.

Once DUT has started to provide Vconn on CC2, TE will measure the voltage present on CC2. After that TE will do a cable-flip and repeat the steps as above.

For a PASS result, measured values from CC1, CC2 and Vconn must be within ranges defined by the provided parameters.

Note	Configuration items for this test should be programmed with averaged values from several "golden sample" DUT's.
-------------	---

Important	In order to run this test with UCD-240, Unigraf Electrical Test Cable must be used and Electrical Testing feature enabled with a corresponding license.
------------------	--

Parameters in use

- Test timeout (default 5 000 ms)
- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- CC low voltage limit for default current (0.5/0.9A) (default 261 mV)
- CC high voltage limit for default current (0.5/0.9A) (default 588 mV)
- CC low voltage limit for 1.5A current (default 675 mV)
- CC high voltage limit for 1.5A current (default 1 189 mV)
- CC low voltage limit for 3A current (default 1 238 mV)
- CC high voltage limit for 3A current (default 2 181 mV)
- Vconn low voltage limit (4 750 mV)
- Vconn high voltage limit (5 500 mV)

AUX (SBU) Lines Test

This test verifies operation of SBU lines for short-circuit and open-circuit failures and that hardware directly related to SBU lines is working properly. During the test the TE will operate as Type-C UFP device. For this test, DUT must support DisplayPort Alternate Mode.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event and waits for DUT to enter DP Alternate mode.

Once DUT has entered the DP alternate mode, TE will measure voltage levels on SBU1 (AUX+) and SBU2 (AUX-) lines. Please notice that if TE is acting as DP Sink, it will de-assert HPD signal to keep AUX bus at IDLE state during the voltage measurements.

Once the voltages are measured, TE will do a cable-flip and repeat the steps as above.

For a PASS result, the measured voltages must be within the ranges defined by the provided parameters.

Note	Parameters for this test should be programmed with averaged values from several "golden sample" DUT's.
-------------	--

Important In order to run this test with UCD-240, **Unigraf Electrical Test Cable** must be used and Electrical Testing feature enabled with a corresponding license.

Parameters in use

- Test timeout (default 5 000 ms)
- DUT Capabilities flags (default 7)

Bits	Description
0	DUT Support for DisplayPort Alt Mode (1 Yes, 0 No)
1	DUT can act as power source (1 Yes, 0 No)
2	DUT can receive power from TE (1 Yes, 0 No)
31:0	Reserved

- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- DP Alt Mode timeout (default 5 000 ms)
- DP Alt Mode AUX+ line idle low voltage (default 100 mV)
- DP Alt Mode AUX+ line idle high voltage (default 600 mV)
- DP Alt Mode AUX– line idle low voltage (default 2 500 mV)
- DP Alt Mode AUX– line idle high voltage (default 3 000 mV)

DUT as Power Sink

This test verifies operation of Vbus and GND lines for short-circuit and open-circuit failures. The test is performed using mandatory PDO for power contract. During the test, the TE will operate as power source, and advertise only *vSafe5V* for power contract. In order to run this test, the DUT must support Power Sink role.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event and waits for Power Contract to be established.

After Power Contact has been established TE will wait for the delay stated in *Measurement delay* parameter and measures the current in Vbus and GND lines and the voltage of Vbus. The purpose of the delay is to allow the DUT time to stabilize its power consumption.

Note The test assumes that the current flows through the four separate Vbus and GND lines evenly. Since current measurements are done sequentially, any variance in DUT power consumption during the measurement can cause this test to fail.

Total currents are calculated for Vbus, and for GND. The difference in the measured values of the four connections may not exceed the programmed deviation limits.

For PASS result, the measured Vbus voltage, Vbus current and GND line current must be within the ranges defined by provided parameters.

Important In order to run this test with UCD-240, **Unigraf Electrical Test Cable** must be used and Electrical Testing feature enabled with a corresponding license.

Parameters in use

- Test timeout (default 5 000 ms)
- DUT Capabilities flags (default 7)

Bits	Description
0	DUT Support for DisplayPort Alt Mode (1 Yes, 0 No)
1	DUT can act as power source (1 Yes, 0 No)

2	DUT can receive power from TE (1 Yes, 0 No)
31:0	Reserved

- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- Power Contract timeout (default 10 000 ms)
- Vbus voltage low limit (default 4 750 mV)
- Vbus voltage high limit (default 5 500 mV)
- Vbus current deviation (between wires) (default 100 mA)
- Return (GND) current deviation (between wires) (default 100 mA)
- Measurement delay (default 2 000 ms)
- Minimal current (default 20 mA)

DUT as Power Source

This test verifies operation Vbus and GND lines for short-circuit and open-circuit failures. During the test, TE will operate as power sink and selects only *vSafe5V* PDO for power contract. In order to run this test, the DUT must support Power Source role.

In the start of the test TE temporarily disconnects the CC lines to simulate a re-plug event and waits for Power Contract to be established

After Power Contact has been established TE will wait for the delay stated in *Power measurement delay* parameter and measures the current in Vbus and GND lines and the voltage of Vbus. The purpose of the delay is to allow the DUT time to stabilize its power consumption.

Note	The test assumes that the current flows through the four separate Vbus and GND lines evenly. Since current measurements are done sequentially, any variance in DUT power consumption during the measurement can cause this test to fail.
-------------	--

Total currents are calculated for Vbus, and for GND. The difference in the measured values of the four connections may not exceed the programmed deviation limits.

For PASS result, the measured Vbus voltage, Vbus current and GND line current must be within the ranges defined by provided parameters.

Important	In order to run this test with UCD-240, Unigraf Electrical Test Cable must be used and Electrical Testing feature enabled with a corresponding license.
------------------	--

Parameters in use

Parameters in use

- Test timeout (default 5 000 ms)
- DUT Capabilities flags (default 7)

Bits	Description
0	DUT Support for DisplayPort Alt Mode (1 Yes, 0 No)
1	DUT can act as power source (1 Yes, 0 No)
2	DUT can receive power from TE (1 Yes, 0 No)
31:0	Reserved

- R-plug duration (default 1 500 ms)
- DUT attach timeout (default 10 000 ms)
- Power Contract timeout (default 10 000 ms)
- Vbus voltage low limit (default 4 750 mV)
- Vbus voltage high limit (default 5 500 mV)
- Vbus current deviation (between wires) (default 100 mA)
- Return (GND) current deviation (between wires) (default 100 mA)

Measurement delay (default 2 000 ms)

11. AUTOMATED TEST EXAMPLES

In this chapter we will demonstrate step by step how you can run automated tests straight from the command-line. (For more detailed information of TSI scripting extensions, please refer to TSI-X Reference Manual)

UCD-240 is also fully compatible with third-party test management systems (e.g. NI TestStand). You can also integrate the C++ code into existing test systems. Running automated tests from the command-line is the simplest option for test automation as it does not require any coding or external Software. In these examples automated tests were run using USB Type-C smartphone as a DUT.

First, we will showcase Unigraf's Electrical Test feature by running Electrical Tests on Up Face Port CC and VCONN pins. By measuring the actual voltages through USB-C connector pins, we can ensure that all pins are functioning, and the soldering has been done properly. *Second*, we will run automated DP Alt Mode tests for both two and four lines to make sure that the DUT is functional in both DP Alternate Modes. *Third*, we will perform one CRC based video test (single frame video stability test). All different Electrical Tests and CRC based video tests are available for configuration in UCD Console GUI under the *Source DUT testing* tab.

Electrical Test • Up Face Port CC and VCONN

Before running the automated tests with UCD-240 it needs to be ensured that the Unigraf Test Cable and DUT are connected. Detailed instructions for ensuring the connection can be found from chapter 4.

```

TSI-X V1.10 [R16], (C) 2018, Unigraf Oy. All Rights Reserved.
PD 0000000F
Connect Unigraf Test Cable to UCD.....
Unigraf Test Cable CONNECTED!
Connect Cable to DUT.....
DUT is CONNECTED!

-----USB-C Electrical Test Set / Up Face Port CC and UCONN test-----
[2019-03-12, 20:44:42.058]: Starting USB-C Electrical Test Set / Up Face Port CC and UCONN test (Test ID 786432)
[2019-03-12, 20:44:42.058]: Test Params:
[2019-03-12, 20:44:42.058]: Test timeout (ms): 5000
[2019-03-12, 20:44:42.058]: DUT Supports alt mode: yes
[2019-03-12, 20:44:42.058]: DUT can feed power: yes
[2019-03-12, 20:44:42.058]: DUT can sink power: yes
[2019-03-12, 20:44:42.058]: Replug time period (ms): 3000
[2019-03-12, 20:44:42.058]: DUT connection timeout (ms): 20000
[2019-03-12, 20:44:42.058]: Power contract timeout (ms): 20000
[2019-03-12, 20:44:42.058]: CC Line voltage low limit @0.5/0.9A (mV): 261
[2019-03-12, 20:44:42.058]: CC Line voltage hi limit @0.5/0.9A (mV): 580
[2019-03-12, 20:44:42.058]: CC Line voltage low limit @1.5A (mV): 675
[2019-03-12, 20:44:42.058]: CC Line voltage hi limit @1.5A (mV): 1189
[2019-03-12, 20:44:42.058]: CC Line voltage low limit @3.0A (mV): 1238
[2019-03-12, 20:44:42.058]: CC Line voltage hi limit @3.0A (mV): 2181
[2019-03-12, 20:44:42.058]: Uconn low voltage limit (mV): 4750
[2019-03-12, 20:44:42.059]: Uconn hi voltage limit (mV): 5500
[2019-03-12, 20:44:42.059]: DP Alternate entry timeout (ms): 5000
[2019-03-12, 20:44:42.059]: DP AUX p-line low voltage limit (mV): 100
[2019-03-12, 20:44:42.059]: DP AUX p-line hi voltage limit (mV): 600
[2019-03-12, 20:44:42.059]: DP AUX n-line low voltage limit (mV): 2500
[2019-03-12, 20:44:42.059]: DP AUX n-line hi voltage limit (mV): 3000
[2019-03-12, 20:44:42.059]: Ubus low voltage limit (mV): 4750
[2019-03-12, 20:44:42.059]: Ubus hi voltage limit (mV): 5500
[2019-03-12, 20:44:42.059]: Ubus current deviation (per-mille): 100
[2019-03-12, 20:44:42.059]: GND current deviation (per-mille): 100
[2019-03-12, 20:44:42.059]: Power contract to power measure delay (ms): 2000
[2019-03-12, 20:44:42.059]: DUT Minimum power use (mA): 20
[2019-03-12, 20:44:42.143]: Test max runtime 300000 ms
[2019-03-12, 20:44:42.153]: 0000.000.001: Start test "Up Face Port CC and UCONN test"
[2019-03-12, 20:44:42.153]: 0000.000.073: USB Et cable contain 1 USB2.0 lanes
[2019-03-12, 20:44:42.156]: 0000.002.457: Setup for testing CC1 line
[2019-03-12, 20:44:42.157]: 0000.003.312: Disconnect CC lines for 3000 ms ...
[2019-03-12, 20:44:45.151]: 0003.005.817: Set UFP role
[2019-03-12, 20:44:45.152]: 0003.006.979: Connect CC lines
[2019-03-12, 20:44:45.152]: 0003.007.026: Wait for DUT to be attached 20000 ms ...
[2019-03-12, 20:44:46.795]: 0004.638.114: DUT attached
[2019-03-12, 20:44:46.795]: 0004.638.216: Measure voltage level on CC1 line ... 0.9520
[2019-03-12, 20:44:46.796]: 0004.638.327: Measure voltage level on CC2 line ... 4.0940
[2019-03-12, 20:44:47.100]: 0004.939.908: Measure voltage level on Uconn line ... 5.2530
[2019-03-12, 20:44:47.100]: 0004.940.098: Setup for testing CC2 line
[2019-03-12, 20:44:47.101]: 0004.940.961: Disconnect CC lines for 3000 ms ...
[2019-03-12, 20:44:50.110]: 0007.943.848: Set UFP role
[2019-03-12, 20:44:50.110]: 0007.945.007: Connect CC lines
[2019-03-12, 20:44:50.110]: 0007.945.052: Wait for DUT to be attached 20000 ms ...
[2019-03-12, 20:44:54.048]: 0011.895.506: DUT attached
[2019-03-12, 20:44:54.048]: 0011.895.609: Measure voltage level on CC2 line ... 0.9680
[2019-03-12, 20:44:54.048]: 0011.895.720: Measure voltage level on CC1 line ... 4.0940
[2019-03-12, 20:44:54.049]: 0011.895.792: Wait for Uconn powered on 300 ms ...
[2019-03-12, 20:44:54.348]: 0012.196.971: Measure voltage level on Uconn line ... 5.2530
[2019-03-12, 20:44:54.348]: 0012.197.161: Test results:
[2019-03-12, 20:44:54.348]: 0012.197.208: Straight cable:
[2019-03-12, 20:44:54.348]: 0012.197.311: CC1: 0.9520 is in ranges [0.6750, 1.1890]
[2019-03-12, 20:44:54.348]: 0012.197.643: Uconn1: 5.2530 is in range [4.7500, 5.5000]
[2019-03-12, 20:44:54.348]: 0012.197.934: Flipped cable:
[2019-03-12, 20:44:54.348]: 0012.198.036: CC2: 0.9680 is in ranges [0.6750, 1.1890]
[2019-03-12, 20:44:54.348]: 0012.198.372: Uconn2: 5.2530 is in range [4.7500, 5.5000]
[2019-03-12, 20:44:54.367]: 0012.218.813: Test PASSED: "Up Face Port CC and UCONN test"
[2019-03-12, 20:44:54.387]: Test Complete

```

Test Parameters

You can use UCD Console GUI to set the test parameters for the automated tests. Please note that test parameters should always be set according to the characteristics of the DUT and test parameters used in this test set-up might not be suitable for other DUTs. Also, some of the test parameters, such as the voltage limits in electrical tests, are based on the USB-C specifications.

Test parameters are saved as a text file. When an automated test script is run the parameters are automatically searched from the test parameters text file. For detailed instructions on how to set the test parameters, please refer to the *USB-C Electrical Test Set* chapter in Chapter 9.

UCD-240 [1844C460] - USB-C, DP Alt Mode Reference Sink

File View Tools Help

Source DUT Testing

Device

UCD-240

DP RX CRC test set DP RX Simple LT test set USB Type C Electrical test set

Test Name

■ Up Face Port CC and VCONN test

USB Type C Electrical test parameters

Test Parameters	
Power Contract timeout, milliseconds	10000
CC low voltage limit for default current (0.5A), mV	261
CC high voltage limit for default current (0.5A), mV	588
CC low voltage limit for 1.5A current, mV	675
CC high voltage limit for 1.5A current, mV	1189
CC low voltage limit for 3A current, mV	1238
CC high voltage limit for 3A current, mV	2181
Vconn low voltage limit, mV	4750
Vconn high voltage limit, mV	5500
DP Alt Mode timeout, milliseconds	5000
DP Alt Mode AUX + line idle low voltage limit, mV	100
DP Alt Mode AUX + line idle high voltage limit, mV	600
DP Alt Mode AUX - line idle low voltage limit, mV	2500
DP Alt Mode AUX - line idle high voltage limit, mV	3000
Vbus voltage low limit, mV	4750
Vbus voltage high limit, mV	5500
Vbus current deviation (between wires), mA	100
Return (GND) current deviation (between wires), mA	100
Measurement delay, msec	2000
Minimal current, mA	20

Presets...

OK Cancel

Run Tests

Automated tests are run from the command-line by calling the test scripts. Alternatively, you can run automated tests by opening the batch files that will automatically perform the automated tests. When test is run from the command-line the set test parameters set earlier in the UCD Console GUI are shown in the test log.

```

---USB-C Electrical Test Set / Up Face Port CC and UCONN test-----
4:42.0581: Starting USB-C Electrical Test Set / Up Face Port CC and UCONN test
4:42.0581: Test Params:
4:42.0581: Test timeout (ms): 5000
4:42.0581: DUT Supports alt mode: yes
4:42.0581: DUT can feed power: yes
4:42.0581: DUT can sink power: yes
4:42.0581: Replug time period (ms): 3000
4:42.0581: DUT connection timeout (ms): 20000
4:42.0581: Power contract timeout (ms): 20000
4:42.0581: CC Line voltage low limit @0.5/0.9A (mV): 261
4:42.0581: CC line voltage hi limit @0.5/0.9A (mV): 588
4:42.0581: CC Line voltage low limit @1.5A (mV): 675
4:42.0581: CC line voltage hi limit @1.5A (mV): 1189
4:42.0581: CC Line voltage low limit @3.0A (mV): 1238
4:42.0581: CC line voltage hi limit @3.0A (mV): 2181
4:42.0581: Uconn low voltage limit (mV): 4750
4:42.0591: Uconn hi voltage limit (mV): 5500
4:42.0591: DP Alternate entry timeout (ms): 5000
4:42.0591: DP AUX p-line low voltage limit (mV): 100
4:42.0591: DP AUX p-line hi voltage limit (mV): 600
4:42.0591: DP AUX n-line low voltage limit (mV): 2500
4:42.0591: DP AUX n-line hi voltage limit (mV): 3000
4:42.0591: Ubus low voltage limit (mV): 4750
4:42.0591: Ubus hi voltage limit (mV): 5500
4:42.0591: Ubus current deviation (per-mille): 100
4:42.0591: GND current deviation (per-mille): 100
4:42.0591: Power contract to power measure delay (ms): 2000
4:42.0591: DUT Minimum power use (mA): 20
4:42.1431: Test max runtime 300000 ms
4:42.1531: 0000.000.001: Starting USB-C Electrical Test Set / Up Face Port CC and UCONN test"
4:42.1531: 0000.000.073: USB ET cable contain 1 USB2.0 lanes
4:42.1561: 0000.000.458: Starting USB-C Electrical Test Set / Up Face Port CC and UCONN test

```

Test Steps

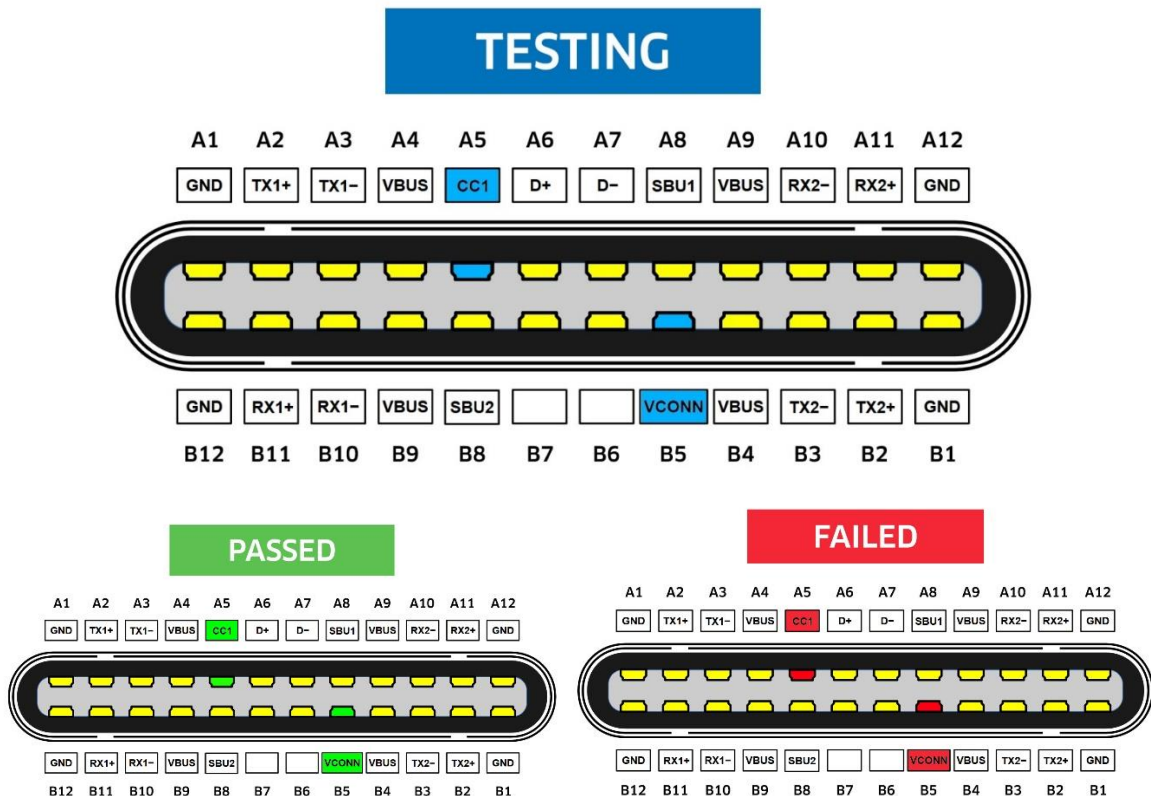
Measuring voltages on CC and Vconn lines on straight cable orientation

The actual measured voltages in straight cable orientation for each pin are shown in the test log.

```

1521: 0003.006.979: Connect CC lines
1521: 0003.007.026: Wait for DUT to be attached 20000 ms ...
7951: 0004.638.114: DUT attached
7951: 0004.638.216: Measure voltage level on CC1 line ... 0.9520
7961: 0004.638.327: Measure voltage level on CC2 line ... 4.0940
7961: 0004.638.397: Wait for Uconn powered on 300 ms ...
1001: 0004.939.908: Measure voltage level on Uconn line ... 5.2530
1001: 0004.940.098: Setup for testing CC2 line
1011: 0004.940.961: Disconnect CC lines for 3000 ms ...
1101: 0007.943.840: Set UFP role
1101: 0007.945.007: Connect CC lines
1101: 0007.945.052: Wait for DUT to be attached 20000 ms ...
0481: 0011.895.506: DUT attached
0481: 0011.895.609: Measure voltage level on CC2 line ... 0.9680
0481: 0011.895.720: Measure voltage level on CC1 line ... 4.0940
0491: 0011.895.792: Wait for Uconn powered on 300 ms ...
3401: 0012.196.971: Measure voltage level on Uconn line ... 5.2530
3401: 0012.197.161: Test results:
3401: 0012.197.208:   Straight cable:
3401: 0012.197.311:     CC1: 0.9520 is in ranges [0.6750, 1.1890]
3401: 0012.197.643:     Uconn1: 5.2530 is in range [4.7500, 5.5000]
3401: 0012.197.934:   Flipped cable:
3401: 0012.198.036:     CC2: 0.9680 is in ranges [0.6750, 1.1890]
3411: 0012.198.372:     Uconn2: 5.2530 is in range [4.7500, 5.5000]
3671: 0012.218.813: Test PASSED: "Up Face Port CC and UCONN test"
3701:
    
```

The *testing* icon below will appear on the command-line as an external window to demonstrate which pins are being measured at each step of the test. After the test a gif will indicate if the test passed or failed.



Performing a SW based cable flip

After finishing the first set of measurements, Unigraf's unique *SW based cable flip* feature will change the cable orientation from straight to flipped. The SW based cable flip enables testing both cable orientations without unplugging the cable.

```
.1521: 0003.006.979: Connect CC lines
.1521: 0003.007.026: Wait for DUT to be attached 20000 ms ...
.7951: 0004.638.114: DUT attached
.7951: 0004.638.216: Measure voltage level on CC1 line ... 0.952U
.7961: 0004.638.327: Measure voltage level on CC2 line ... 4.094U
.7961: 0004.638.397: Wait for Uconn powered on 300 ms ...
.1001: 0004.939.908: Measure voltage level on Uconn line ... 5.253U
.1001: 0004.940.098: Setup for testing CC2 line
.1011: 0004.940.961: Disconnect CC lines for 3000 ms ...
.1101: 0007.943.840: Set UFP role
.1101: 0007.945.007: Connect CC lines
.1101: 0007.945.052: Wait for DUT to be attached 20000 ms ...
.0481: 0011.895.506: DUT attached
.0481: 0011.895.609: Measure voltage level on CC2 line ... 0.968U
.0481: 0011.895.720: Measure voltage level on CC1 line ... 4.094U
.0491: 0011.895.792: Wait for Uconn powered on 300 ms ...
.3401: 0012.196.971: Measure voltage level on Uconn line ... 5.253U
.3401: 0012.197.161: Test results:
.3401: 0012.197.208:   Straight cable:
.3401: 0012.197.311:     CC1: 0.952U is in ranges [0.675U, 1.189U]
.3401: 0012.197.643:     Uconn1: 5.253U is in range [4.750U, 5.500U]
.3401: 0012.197.934:   Flipped cable:
.3401: 0012.198.036:     CC2: 0.968U is in ranges [0.675U, 1.189U]
.3411: 0012.198.372:     Uconn2: 5.253U is in range [4.750U, 5.500U]
.3671: 0012.218.813: Test PASSED: "Up Face Port CC and UCONN test"
.3701:
.3871: Test Complete
```

Measuring voltages on CC and Vconn lines on flipped cable

Same measurements as in the first step of the test are now performed in the flipped cable orientation.

```
1521: 0003.006.979: Connect CC lines
1521: 0003.007.026: Wait for DUT to be attached 20000 ms ...
7951: 0004.638.114: DUT attached
7951: 0004.638.216: Measure voltage level on CC1 line ... 0.952U
7961: 0004.638.327: Measure voltage level on CC2 line ... 4.094U
7961: 0004.638.397: Wait for Uconn powered on 300 ms ...
1001: 0004.939.908: Measure voltage level on Uconn line ... 5.253U
1001: 0004.940.098: Setup for testing CC2 line
1011: 0004.940.961: Disconnect CC lines for 3000 ms ...
1101: 0007.943.840: Set UFP role
1101: 0007.945.007: Connect CC lines
1101: 0007.945.052: Wait for DUT to be attached 20000 ms ...
0481: 0011.895.506: DUT attached
0481: 0011.895.609: Measure voltage level on CC2 line ... 0.968U
0481: 0011.895.720: Measure voltage level on CC1 line ... 4.094U
0491: 0011.895.792: Wait for Uconn powered on 300 ms ...
3401: 0012.196.971: Measure voltage level on Uconn line ... 5.253U
3401: 0012.197.161: Test results:
3401: 0012.197.208:   Straight cable:
3401: 0012.197.311:     CC1: 0.952U is in ranges [0.675U, 1.189U]
3401: 0012.197.643:     Uconn1: 5.253U is in range [4.750U, 5.500U]
3401: 0012.197.934:   Flipped cable:
3401: 0012.198.036:     CC2: 0.968U is in ranges [0.675U, 1.189U]
3411: 0012.198.372:     Uconn2: 5.253U is in range [4.750U, 5.500U]
3671: 0012.218.813: Test PASSED: "Up Face Port CC and UCONN test"
3701:
3871: Test Complete
```

Test Results

Finally, the test results are shown for both cable orientations. Measured voltages and ranges defined in the USB-C specification are shown for each measurement. In the final row of the test-run it's shown if the test passed or failed. In this test case all measurements were within the accepted range and the *Up Face Port and VCONN test* passed.

```
.1521: 0003.006.979: Connect CC lines
.1521: 0003.007.026: Wait for DUT to be attached 20000 ms ...
.7951: 0004.638.114: DUT attached
.7951: 0004.638.216: Measure voltage level on CC1 line ... 0.952U
.7961: 0004.638.327: Measure voltage level on CC2 line ... 4.094U
.7961: 0004.638.397: Wait for Uconn powered on 300 ms ...
.1001: 0004.939.908: Measure voltage level on Uconn line ... 5.253U
.1001: 0004.940.098: Setup for testing CC2 line
.1011: 0004.940.961: Disconnect CC lines for 3000 ms ...
.1101: 0007.943.840: Set UFP role
.1101: 0007.945.007: Connect CC lines
.1101: 0007.945.052: Wait for DUT to be attached 20000 ms ...
.0481: 0011.895.506: DUT attached
.0481: 0011.895.609: Measure voltage level on CC2 line ... 0.968U
.0481: 0011.895.720: Measure voltage level on CC1 line ... 4.094U
.0491: 0011.895.792: Wait for Uconn powered on 300 ms ...
.3401: 0012.196.971: Measure voltage level on Uconn line ... 5.253U
.3401: 0012.197.161: Test complete
.3401: 0012.197.208:
.3401: 0012.197.311:   Straight cable:
.3401: 0012.197.643:     CC1: 0.952U is in ranges [0.675U, 1.189U]
.3401: 0012.197.934:     Uconn1: 5.253U is in range [4.750U, 5.500U]
.3401: 0012.198.036:   Flipped cable:
.3411: 0012.198.372:     CC2: 0.968U is in ranges [0.675U, 1.189U]
.3411: 0012.198.372:     Uconn2: 5.253U is in range [4.750U, 5.500U]
.3671: 0012.218.813: Test PASSED: "Up Face Port CC and VCONN test"
.3701:
.3871: Test Complete
```

CRC Based Video Test • Single Frame Video Stability Test

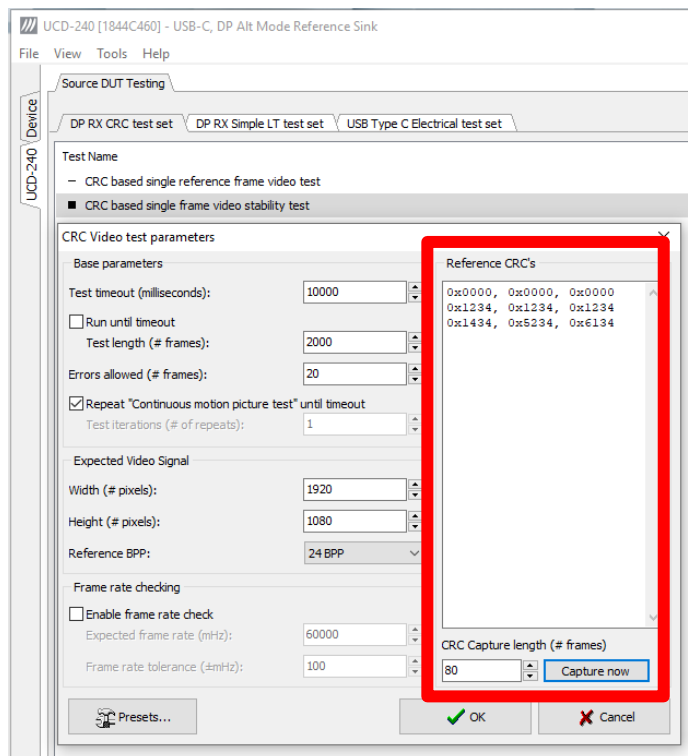
Test Parameters and Reference Frame

Similar to Electrical Test, test parameters for CRC based video tests are set in UCD Console GUI. You can use UCD Console GUI to set the test parameters for the automated tests. Please note that test parameters should be set according to the characteristics of the DUT and test parameters use in this test set-up might not be suitable for other DUTs. For detailed instructions on how to set the test parameters, please refer to the CRC based Video Test Set chapter in Chapter 9. Please, note that UCD Console GUI cannot be operated if UCD-240 is running scripts on the command-line.

Before running automated CRC based video test, a reference frame must be captured and the DUT must be set to DP Alt Mode.

You can set the DUT to DP Alt Mode by running the *Set4LanesModeOn* or *Set2LanesModeOn* macro. For detailed instructions refer to chapter 4 of this manual. Detailed information about the macros can be found from the USB Flash Drive delivered with UCD-240 from the macros text file.

For detailed information on capturing the reference frame, please refer to chapter 9 in this manual. Please note that the Reference frame should not include any changing items such as clock or battery level indicator as a change in the reference CRC will result as a failed test.



Run Tests

Automated tests are run from the command-line by calling the test scripts. Alternatively, you can run automated tests by opening the batch files that will perform the automated tests. When test is run from the command-line the set test parameters set earlier in the UCD Console GUI are shown in the test log.

```

04.7841: Starting CRC video test (Test ID 393216)
04.7851: Stage 1: Test initialization
04.7851: Test params:
04.7851: - Test timeout (ms) = 10000
04.7851: - Test duration (# frames) = 100
04.7851: - Allowed mismatches (# frames) = 2
04.7851: - Reference width (# pixels) = 3840
04.7851: - Reference height (# Lines) = 2160
04.7851: - Reference colordepth (bpp) = 24
04.7851: - CRC Value set count = 10
04.7851: - Require frame rate (mHz) = 0
04.7851: - Frame rate tolerance (mHz) = 0
04.7851: - Motion test iterations (# loops): 0
04.7861: - Color format (ID): 0
04.8331: Test max runtime 300000 ms

```

Test Steps

Measuring timings and checking parameters

Test sequence checks that the width, height and BPP of the incoming video is correct.

```

04.8331: Test max runtime 300000 ms
04.8431: 0000.000.001: Start test "CRC based single reference frame video test"
04.8441: 0000.000.082: Test params:
04.8441: 0000.000.223: Reference Width = 3840:
04.8441: 0000.000.314: Reference Height = 2160:
04.8441: 0000.000.402: Reference BPP = 24:
04.8451: 0000.000.465: Frames to test = 100
04.8451: 0000.000.537: Stage 1: - connecting to input interface...
04.8721: 0000.014.049: done.
04.8731: 0000.014.091: Stage 2: - measure timings and check params...
04.8731: 0000.014.394: Measured Width is correct
04.8731: 0000.014.441: Measured Height is correct
04.8731: 0000.014.488: Measured BPP is correct
04.8741: 0000.014.534: Frame rate is ignored
04.8741: 0000.014.594: done.
04.8741: 0000.014.633: Stage 3: - test data collection...
04.8741: 0000.030.710: Expected frame with crc : 0xa03a, 0x7e94, 0xccbe
04.8741: 0000.030.819: done.
04.8741: 0000.030.860: Stage 4: - gathering information...
06.5471: 0001.698.274: 100 frames were tested.
06.5491: 0001.698.355: 0 mismatches were found.
06.5491: 0001.698.419: done.
06.5491: 0001.698.462: Stage 5: - test data collection completed
06.5501: 0001.698.671: Test PASSED: "CRC based single reference frame video test"
06.5521:
06.5601: Test Complete

```

Measuring that resolution and BPP match the reference frame

Testing the CRC of the incoming video against the reference frame's CRC values captured and defined earlier. The number of mismatches found during the are shown.

```

04.8331: Test max runtime 300000 ms
04.8431: 0000.000.001: Start test "CRC based single reference frame video test"
04.8441: 0000.000.002: Test params:
04.8441: 0000.000.223: Reference Width = 3840:
04.8441: 0000.000.314: Reference Height = 2160:
04.8441: 0000.000.402: Reference BPP = 24:
04.8451: 0000.000.465: Frames to test = 100
04.8451: 0000.000.537: Stage 1: - connecting to input interface...
04.8721: 0000.014.049: done.
04.8731: 0000.014.091: Stage 2: - measure timings and check params...
04.8731: 0000.014.394: Measured Width is correct
04.8731: 0000.014.441: Measured Height is correct
04.8731: 0000.014.488: Measured BPP is correct
04.8741: 0000.014.534: Frame rate is ignored
04.8741: 0000.014.594: done.
04.8741: 0000.014.633: Stage 3: - synchronization...
04.8741: 0000.030.710: Expected frame with crc : 0xa03a, 0x7e94, 0xcche
04.8741: 0000.030.819: done.
04.8741: 0000.030.860: Stage 4: - gathering information...
06.5471: 0001.698.274: 100 frames were tested.
06.5491: 0001.698.355: 0 mismatches were found.
06.5491: 0001.698.419: done.
06.5491: 0001.698.462: Stage 5: - test data collection completed
06.5501: 0001.698.671: Test PASSED: "CRC based single reference frame video test"
06.5521:
06.5601: Test Complete

```

Test Results

Finally, it shown if the test passed. In the UCD Console GUI it is set how many mismatched on the test are allowed. In this case 0 mismatched were found and the test passed. The automated CRC based single frame video stability test took 2.14 seconds to run in this test case.

```

04.8331: Test max runtime 300000 ms
04.8431: 0000.000.001: Start test "CRC based single reference frame video test"
04.8441: 0000.000.002: Test params:
04.8441: 0000.000.223: Reference Width = 3840:
04.8441: 0000.000.314: Reference Height = 2160:
04.8441: 0000.000.402: Reference BPP = 24:
04.8451: 0000.000.465: Frames to test = 100
04.8451: 0000.000.537: Stage 1: - connecting to input interface...
04.8721: 0000.014.049: done.
04.8731: 0000.014.091: Stage 2: - measure timings and check params...
04.8731: 0000.014.394: Measured Width is correct
04.8731: 0000.014.441: Measured Height is correct
04.8731: 0000.014.488: Measured BPP is correct
04.8741: 0000.014.534: Frame rate is ignored
04.8741: 0000.014.594: done.
04.8741: 0000.014.633: Stage 3: - synchronization...
04.8741: 0000.030.710: Expected frame with crc : 0xa03a, 0x7e94, 0xcche
04.8741: 0000.030.819: done.
04.8741: 0000.030.860: Stage 4: - gathering information...
06.5471: 0001.698.274: 100 frames were tested.
06.5491: 0001.698.355: 0 mismatches were found.
06.5491: 0001.698.419: done.
06.5491: 0001.698.462: Stage 5: - test data collection completed
06.5501: 0001.698.671: Test PASSED: "CRC based single reference frame video test"
06.5521:
06.5601: Test Complete

```

12. TSI PROGRAMMING

This section defines mechanisms used in TSI that might not be otherwise clear from the other parts of the reference manual. The intent is to clarify the operation of the more complex parts by providing additional descriptions and details on intended uses. For full details of the TSI, please refer to the TSI Manual which can be found from the USB Flash Drive delivered with the device.

Operator feedback during test execution

Some of the new TSI tests may require operator feedback and/or operator actions during test execution. This section describes the implementation details in TSI that are used to carry out the operator communications. The provided solution only prompts operator to carry out operations on the DUT side, but the intention of this mechanism is to enable test system manufacturers to fully automate test execution even when the test requires an operation to be performed on the DUT side.

Operator feedback implementation in TSI

TSI is planned to support more than one way of operator intervention / feedback mechanisms. The CI named `TSI_TS_OFMODE` is reserved to select which method is used. Currently, the only method supported is `TSI_OFMODE_RUN_EXTERNAL`. In this mode, TSI will run an external application. A default command line application is distributed with TSI, along with its source code. TSI itself can be the external application (see Error: Reference source not found Running Tests).

Selecting which application TSI will run

By default, TSI will use its own operator feedback application for all operator feedback requests. However, it is possible to define separate applications for each operator feedback request type. The definition is controlled by two separate CIs. The first of these is `“TSI_TS_OF_REQ_ID”`, which defines the request ID related to the application setting. The default setting is `-1`, which means that the application setting applies to all request IDs. The second CI is `“TSI_TS_OF_EXT_APP”`, which contains the path and name of the executable file to be started as a NULL terminated string. These definitions are in the device scope, so each test device will have its own set of application definitions.

In short, to use a single application for all operator feedback requests:

1. Write `-1` to `“TSI_TS_OF_REQ_ID”` CI.
2. Write a null terminated string containing full path and name of your application to `“TSI_TS_OF_EXT_APP”` CI.

To configure an application for any operator feedback request (this step can be repeated to create multiple settings) :

1. Write the request ID to `“TSI_TS_OF_REQ_ID”`.
2. Write a null terminated string containing full path and name of your application to `“TSI_TS_OF_EXT_APP”` CI.

External application requirements

When TSI calls an external application, it passes information about the request as a command-line parameter containing a number of key-value pairs, or as a reference to a file containing a number of key-value pairs.

TSI will pass a file reference only if the request information string would grow too large for passing it through command line parameters. One way to determine if the given string is a file reference or not, is to check the string's contents. If the string contains no equal characters ('='), it is very likely a file reference as it can't be a valid key-value script. In this case, the application should load this file and process it the same way it would do for the command-line passed information string.

(Continued...)

(...Continued)

The external application is expected to parse the key-value pairs and result in the DUT carrying out the requested operation. It is up to the designer of this application to decide how to implement it.

The external application is expected to return one of the response values listed in the requested information based on the outcome of the operation.

TSI Default application operation

The default application will show a command-prompt window with a message to the test operator, and wait for the test operator to enter a response. Possible responses are:

- *'Enter'* only → “Proceed” response.
- *'P'* + *'Enter'* → “Pass” response. (Note: Case-insensitive)
- *'F'* + *'Enter'* → “Fail” response. (Note: Case-insensitive)

Entering an invalid response, or a response that is not listed in the request information will not be accepted and the application will repeat the prompt for a valid response.

Request parameters

When a test requires operator intervention to configure the DUT and/or other operator feedback, TSI will run the defined application with command-line parameters that define the contents of the request. The command-line parameters are passed as a key-value pair script. The key-value pair script assumes the following rules in its syntax:

- “KEY” and “VALUE” -fields are separated with the equal character (=).
- “VALUE” -field is followed by a semi-colon (;).
- “KEY” and “VALUE” -field pair must be on a single line of text.
- If either the “KEY” or “VALUE” -field contains spaces, the field must be enclosed with single quote-marks.
- Strings: The closing single-quote-mark also marks end of the field being processed.
- White-spaces characters are allowed anywhere outside “KEY” and “VALUE” -fields.
- New-lines (and white-spaces) may always appear before beginning of the next “KEY” -field. For example, at start of file and after semi-colon (;) ending the previous key-value pair.

The “KEY” -fields are always character strings, and “VALUE” -fields are decimal numeric values. Please see the table on the next page for a list of the “KEY”-fields generated by TSI, along with the meaning of the associated “VALUE”-fields.

(Continued...)

(...Continued)

Key	Value / Description	Reference	
"op"	The value is a request ID that identifies what is being requested for the operator to do and/or evaluate. Currently valid ID's are listed below.	-	
	1	Request for DUT to start link training with given lane count and link speed	1.1.5.1
	2	Request for DUT to transmit test pattern #1 in given video mode	1.1.5.2
	3	Request for DUT to read EDID from TE	1.1.5.3
	4	Request for DUT to reduce number of lanes being used	1.1.5.4
	5	Request for DUT to increase number of lanes being used	1.1.5.5
	6	Request for DUT to transmit test pattern #1 in given video and color mode	1.1.5.6
	7	Request for DUT to enter power save mode	1.1.5.7
	8	Request for DUT to exit power save mode	1.1.5.8
	9	Request for DUT to transmit test pattern #1 in given video mode and color depth	1.1.5.9
	10	Request for DUT to transmit video and audio with defined parameters.	1.1.5.10
	11	Request for operator to check audio test pattern playback and respond with "pass" if audio playback is correct, and with "fail" if audio playback is not correct.	1.1.5.11
	12	Request for operator to check video pattern and respond with "pass" if video playback is correct, and with "fail" if video playback is not correct.	1.1.5.12
	13	Request for operator to play the "ping test pattern" audio stream three times and verify it played correctly. If the playback is correct, respond with "pass" or if the playback was not correct, respond with "fail"	1.1.5.13
14	Request for DUT to write FEC_READY bit in DPCD register 0x120	1.1.5.14	

	15	Request for DUT to send ENABLE_FEC sequence	1.1.5.15
	16	Request for DUT to send DISABLE_FEC sequence	1.1.5.16
“res_x”	The value indicates the width of the active area of the requested video mode in terms of number of pixels		-
“res_y”	The value indicates the height of the active area of the requested video mode in terms of number of pixels		-
“res_bpp”	The value indicates the number of bits per pixel for the requested video mode		-
“res_frame_rate”	The value indicates the frame-rate of the requested video mode as milli-Hertz		-
“col_format”	Indicates requested color format as an ID value. Currently valid ID’s are listed below:		-
	0	RGB	-
	1	YCbCr 4:2:2	-
	2	YCbCr 4:4:4	-
“col_range”	Indicates the dynamic range standard for the requested color space as an ID value. Currently valid ID’s are listed below:		-
	0	VESA	-
	1	CEA	-
“col_gamma”	Indicates the colorimetry standard for the requested color space as an ID value. Currently valid ID’s are listed below:		-
	0	ITU-601	-
	1	ITU-709	-
“col_bits”	Indicates color-depth as bits per pixel for the requested color space		-

(Continued...)

(...Continued)

Key	Value / Description	Reference
“dp_lanes”	Indicates number of DP lanes to be used in the next link training performed by the DUT	-
“dp_linkrate”	Indicates link-rate of DP lanes to be used in the next link training performed by the DUT. The link rate is provided as a multiplier of 0.27Gbps.	-
“aud_pat”	Indicates the requested audio pattern type	
	0	Operator specific
	1	Sawtooth pattern
“aud_chn”	Indicates number of audio channels to send. Typical range from 2 to 8.	-
“aud_freq”	Indicates the sampling rate of the audio stream to send, as Hz.	-
“aud_bits”	Indicates the sample size as number of bits.	-
“aud_alloc”	Channel allocation bits to be used. The bits are passed as a decimal number. Important: A value of zero mean no channel allocations bits to be used – This is valid for stereo audio.	
	Bit	Purpose
	0	FL + FR (Front-left and Front-right channels present)
	1	LFE (Low-frequency-effects channel present)
	2	FC (Front-center channel present)
	3	RL + RR (Rear-left and Rear-right channels present)
	4	RC (Rear-center channel present)
	5	FLC + FRC (Front-left-of-center and Front-right-of-center channels present)
	6	RLC + RRC (Rear-left-of-center and Rear-right-of-center channels present)
	7	FLW + FRW (Front-left-wide and Front-right-wide channels present)
	8	FLH + FRH (Front-left-high and Front-right-high channels present)
	9	TC (Top-center channel present)
	10	FHC (Front-high-center channel present)
“aud_ppr_1” ... “aud_ppr_8”	Indicates the pattern period as number of samples. Important: The values are only present for number of channels requested. Do not expect there to always be same number of period keys. Important: the values will not be passed if the requested audio pattern is not “sawtooth”.	-

“exit_proceed”	Defines expected exit code for “proceed” response	-
“exit_pass”	Defines expected exit code for “pass” response.	-
“exit_fail”	Defines expected exit code for “fail” response.	-

Request parameter details

Request ID 1

Request for DUT to start link training with given lane count and link speed. Passed parameters include at least the following:

- “op” – ID for this request (1)
- “dp_lanes” – Number of lanes to use. Typically 1, 2 or 4.
- “dp_linkrate” – Link rate multiplier for 0.27 Gbps
- “exit_proceed” – Value for the application to return as its exit code

Request ID 2

Request for DUT to transmit test pattern #1 in given video mode. Passed parameters include at least the following:

- “op” – ID for this request (2)
- “res_x” – Number of horizontal pixels on the active area
- “res_y” – Number of vertical pixels on the active area
- “res_frate” – Number of frames per second, as milli-hertz (60 fps = 60000)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 3

Request for DUT to read EDID from TE. Passed parameters include at least the following:

- “op” – ID for this request (3)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 4

Request for DUT to reduce number of lanes being used. Passed parameters include at least the following:

- “op” – ID for this request (4)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 5

Request for DUT to increase number of lanes being used. Passed parameters include at least the following:

- “op” – ID for this request (5)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 6

Request for DUT to transmit test pattern #1 in given video and color mode. Passed parameters include at least the following:

- “op” – ID for this request (6)
- “res_x” – Number of horizontal pixels on the active area
- “res_y” – Number of vertical pixels on the active area

- “res_frate” – Number of frames per second, as milli-hertz (60 fps = 60000)
- “res_bpp” – Number of bits per pixel, as number of bits
- “col_format” – Indicates the wanted pixel color formatting
- “col_range” – Indicates dynamic range (VESA/CEA)
- “col_yc” – Indicates YCbCr coefficients (ITU-601/ITU-709)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 7

Request for DUT to enter power save mode. Passed parameters include at least the following:

- “op” – ID for this request (7)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 8

Request for DUT to exit power save mode. Passed parameters include at least the following:

- “op” – ID for this request (8)
- “exit_proceed” – Value for the application to return as its exit code

Request ID 9

Request for DUT to transmit test pattern #1 in given video mode and color depth. Passed parameters include at least the following:

- “op” – ID for this request (9).
- “res_x” – Number of horizontal pixels on the active area.
- “res_y” – Number of vertical pixels on the active area.
- “res_frate” – Number of frames per second, as milli-hertz (60 fps = 60000).
- “res_bpp” – Number of bits per pixel, as number of bits.
- “exit_proceed” – Value for the application to return as its exit code.

Request ID 10

Request for DUT to transmit video and audio with defined parameters. The following parameters are included:

- “op” – ID for this request (10).
- “res_x” – Number of horizontal pixels on the active area.
- “res_y” – Number of vertical pixels on the active area.
- “res_frate” – Number of frames per second, as milli-hertz (60 fps = 60000).
- “res_bpp” – Number of bits per pixel, as number of bits.
- “aud_pat” – Indicates the audio pattern to be used.
- “aud_chn” – Indicates the number of audio channels to be used.
- “aud_freq” – Indicates the sampling rate to be used.
- “aud_bits” – Indicates the sample size, in bits, to be used.
- “aud_alloc” – Indicates the channel allocation bits to be used.
- “aud_ppr_1” ... “aud_ppr_8” – Indicates the pattern periods as count of samples for each channel to be used. NOTE: These values are present only when the requested audio pattern is “sawtooth”.
- “exit_proceed” – Value for the application to return as its exit code.

Request ID 11

Request for operator to check audio test pattern playback and respond with “pass” if audio playback is correct, and with “fail” if audio playback is not correct. The following parameters are included:

- “op” – ID for this request (11)
- “exit_pass” – Value for the application to return for “pass” response.
- “exit_fail” – Value for the application to return for “fail” response.

Request ID 12

Request for operator to check video pattern and respond with “pass” if video playback is correct, and with “fail” if video playback is not correct. The following parameters are included:

- “op” – ID for this request (12)
- “exit_pass” – Value for the application to return for “pass” response.
- “exit_fail” – Value for the application to return for “fail” response.

Request ID 13

Request for operator to play the “ping test pattern” audio stream three times and verify it played correctly. If the playback is correct, respond with “pass” or if the playback was not correct, respond with “fail”. The following parameters are included:

- “op” – ID for this request (13)
- “exit_pass” – Value for the application to return for “pass” response.
- “exit_fail” – Value for the application to return for “fail” response.

Request ID 14

Request for DUT to write FEC_READY bit in DPCD register 0x120 FEC_CONFIGURATION. The following parameters are included:

- “op” – ID for this request (14)
- “exit_proceed” – Value for the application to return as its exit code.

Request ID 15

Request for DUT to send the ENABLE_FEC sequence as described in the DP specifications. The following parameters are included:

- “op” – ID for this request (15)
- “exit_proceed” – Value for the application to return as its exit code.

Request ID 16

Request for DUT to send the DISABLE_FEC sequence as described in the DP specifications. The following parameters are included:

- “op” – ID for this request (16)
- “exit_proceed” – Value for the application to return as its exit code.

Operator Feedback configuration items

The following configuration items are used to configure the operator feedback mechanism.

TSI_TS_OF_MODE

TSI_TS_OF_MODE	0x215
unsigned int of_mode	U32
4 bytes	RW

Synopsis

Currently not configurable as only a single mode is available. In this mode TSI will run an external application when a test needs operator feedback or intervention with the test.

TSI_TS_OF_REQ_ID

TSI_TS_OF_REQ_ID	0x216
int of_request_id	S32
4 bytes	RW

Synopsis

Selects which request's configuration is accessed through TSI_TS_OF_EXT_APP. Please see table below for request ID values currently defined:

Request ID	Description	Reference
-1	Select All requests. This ID has special meaning for configuring the external application. Setting the external application with this setting clears the internal list and sets the default application. Reading the external application with this setting will only read the default application and leave internal list unchanged.	-
1	Select request for DUT to start link training with given lane count and link speed	1.1.5.1
2	Select request for DUT to transmit test pattern #1 in given video mode	1.1.5.2
3	Select request for DUT to read EDID from TE	1.1.5.3
4	Select request for DUT to reduce number of lanes being used	1.1.5.4
5	Select request for DUT to increase number of lanes being used	1.1.5.5
6	Select request for DUT to transmit test pattern #1 in given video and color mode	1.1.5.6
7	Select request for DUT to enter power save mode	1.1.5.7
8	Select request for DUT to exit power save mode	1.1.5.8
9	Select request for DUT to transmit test pattern #1 in given video mode and color depth	1.1.5.9
10	Select request for DUT to transmit video and audio with defined parameters.	1.1.5.10
11	Select request for operator to check audio test pattern playback and respond with "pass" if audio playback is correct, and with "fail" if audio playback is not correct.	1.1.5.11

12	Select request for operator to check video pattern and respond with “pass” if video playback is correct, and with “fail” if video playback is not correct.	1.1.5.12
13	Select request for operator to play the “ping test pattern” audio stream three times and verify it played correctly. If the playback is correct, respond with “pass” or if the playback was not correct, respond with “fail”	1.1.5.13
14	Request for DUT to write FEC_READY bit in DPCD register 0x120	1.1.5.14
15	Request for DUT to send ENABLE_FEC sequence	1.1.5.15
16	Request for DUT to send DISABLE_FEC sequence	1.1.5.16

TSI_TS_OF_EXT_APP

TSI_TS_OF_EXT_APP	0x217
char external_app[]	ARRAY_U8
260 bytes	RW

Synopsis

Contains a NULL terminated string defining the path and name of the application to be executed for the request ID defined in TSI_TS_OF_REQ_ID configuration item.

Important: When the selected request ID is -1 (TSI_TS_OF_REQ_ID), accessing this CI will have special processing applied to it as defined below

- On write, the internal list of external applications is cleared, and then the default external application is added.
- On read, this CI will return the default application path and name.

Extended scripting engine

The Extended TSI Scripting Engine is an executable scripting language program, `tsi.exe`, built on top of the Extended TSI API.

The program can be run from the Windows command line or from scripting languages such as python.

All input characters from the command line and files need to be ASCII characters.

The "#" is used for comments. All input on a line after a "#" will be ignored. This means you must not try to print this symbol. One should also not use the ";" withing text to be printed as it is used as a delimiter for expressions.

Getting started

To run *TSI.exe*, two (text format) files are generally used: *init.tsi* (required) and *macros.tsi* (optional). These need to be in the same directory (the working directory) from which *TSI.exe* is run.

You need to be aware that changing the state of a device via Configuration Items, CI's, can take time and scripting commands calls are asynchronous: So make ample use of the **-waiton** command (see 1.2.4 Running TSI.EXE).

Defining test equipment devices

init.tsi contains the device names (or aliases) and are of the form:

```
DeviceName = 1813C261 source_sink dp;
```

Here DeviceName is the name you use when opening it, *1813C261* is the device serial number, *source_sink* specifies the sink and source capability and *dp* states that the device uses display port technology (see section 1.2.7 Devices).

inti.tsi may be created from the command line as follows:

```
tsi.exe -l > init.tsi
```

init.tsi will contain device definitions of the form (one definition per line):

```
Dev0 = 1823c395 sink usbc;
```

```
Dev1 = 1823c395 source usbc;
```

The names can be edited for convenience:

```
MyUDC340sink = 1823c395 sink usbc;
```

```
MyUDC340source = 1823c395 source usbc;
```

Commands

Commands are of the form KVE (KeyValueExpression):

```
Key = Value.Expression;
```

Key is usually a configuration item write command from this manual (such as *TSI_W_USBC_CABLE_CONTROL*) or ".read" or ".loadtd" and *Value* is the value you want to set or get (also from this manual).

For example, to disconnect CC lines, you would use the following command:

```
-c TSI_W_USBC_CABLE_CONTROL=5;
```

.Expression (optional) selects the bits of *Value* that you want to test or modify for the configuration item in question.

Expressions for commands that write to configuration items are dot-separated keywords of the form

```
.startbit.number1.length.number2
```

where the first *number1* is the first bit of the sequence and *number2* is the length in bits of the sequence.

For example, to change the color space, one of the KVE's in a chain the might be

```
TSI_PG_CUSTOM_TIMING_FLAGS=1.startbit.11.length.4;
```

The above KVE sets the *TSI_PG_CUSTOM_TIMING_FLAGS* configuration item flags 11-12-13-14 to 1 (0001), which is color space YCbCr 4:4:4. Note that the configuration item value will be read and bits 11-12-13-14 will be modified and the resulting value will be written back to the configuration item.

Commands of the form:

```
-c .read = Value;
```

are used to read the value of a configuration item. For example

```
-c .read = TSI_R_USBC_DP_ALT_MODE_STATUS;
```

will output the value of the configuration item in decimal format.

Commands of the form:

```
-c .read = Value.Expression;
```

are used to check values and bits. The configuration item *Value* modified by *Expression* up to a relational operator (*eq*, *gt* or *lt*) is used to determine if *Value.Expression* is true or false.

Expressions for .read commands are dot-separated modifier keywords (at, bitson, bitsoff, startbit, length and gt, lt, eq) and integer values:

- *.bytes* selects the maximum number of bytes to be read with ARRAY_XX types. It must come right after the Value.
- *.at* selects the CI via an index for .read KV's when reading ARRAY_32 types. It must come right after .bytes if present and right after Value if bytes is not present.
- *.bitson.1.22* is true when bits 1 and 22 are set.
- *.bitsoff.1.22* is true when bits 1 and 22 are off.
- *.and* is used to chain expressions.
- *.eq*, *.lt* and *.gt* are relational operators for "equal", "less than", or "greater than".

Example:

```
-c .read=TSI_R_USBC_DP_ALT_MODE_STATUS.bitson.0.and.startbit.8.length.4.eq.4;
```

The above command reads the usbc alt-mode status configuration item and will be true if bit 0 is on and bits 8-11 have a value of 4 ("D": DisplayPort v1.3 2 lanes + USB 3.1 (USB Type C cable)).

Example:

```
-c .read=TSI_R_DPRX_ERROR_COUNTS.at.1;
```

The above command reads the error count for DP lane 1.

Example:

```
-c .read=TSI_VERSION_TEXT.bytes.1024;
```

The above command reads the version information (a maximum of 1024 bytes) and prints it out.

Defining command macros

The *macros.tsi* text file is automatically read if found.

It should contain definitions of user-defined commands of the form (a mix of KVE's and macros delimited by semi-colons all on the SAME line):

```
MyMacro = Key0=Value0.Expression0; MyMacroA; Key1=Value1; MyMacroB;
```

where *MyMacro* is the new macro being defined, *MyMacroA* and *MyMacroB* are other macros, *Key0* and *Key1* are configuration items, which are defined elsewhere in this manual. *Value0* and *Value1* are the values that are to be read or set for the configuration items. Macros and KVE's must be delimited with semi-colons.

An example of macros file is:

```
# this line is a comment and will be skipped by the macro definition parser.
# 2lanes macro: Disconnect CC lines, set roles for 2 lanes, make delay of
# 200ms and reconnect.
# "→" means no newline character in actual script file.
```

```
2lanes = TSI_W_USBC_CABLE_CONTROL=6;
TSI_USBC_DP_ALT_MODE_SETUP=0x80000008;→
    TSI_W_SCRI_DELAY=200;TSI_W_USBC_CABLE_CONTROL=7;
```

```
# 4lanes macro: Disconnect CC lines, set roles for 4 lanes, make delay of
# 200ms and reconnect.
```

```
4lanes = TSI_W_USBC_CABLE_CONTROL=6;
TSI_USBC_DP_ALT_MODE_SETUP=0x80000012;→
    TSI_W_SCRI_DELAY=200; TSI_W_USBC_CABLE_CONTROL=7;
```

```
#set source timing colorspace and pattern.
```

```
setcolor_to_YCbCr444 = TSI_PG_CUSTOM_TIMING_FLAGS=1.startbit.11.length.4;
apply = TSI_W_PG_COMMAND=4;
set_timing_color_pattern = TSI_W_PG_PREDEF_TIMING_SELECT=3;→
setcolor_to_YCbCr444; TSI_W_PG_PREDEF_PATTERN_SELECT=10; apply;
```

Macros example to set source device to two lane DP Alt Mode and check that it succeeded:

```
-d SourceDevice #source device from init.tsi
-o 1 #output level 1
-timeout 20000 #time to wait for -waiton below
# Set DP Alt Mode for UCD-240. This is a comment
-pr " " # a print command
-pr "-----"
-pr "DP Alt Mode 2 lane Test"
-pr "-----"
-pr "Setting Source into 2 lane DP Alt-mode..."
-c Source_2lanesOn #run macro to set source into 2 lane DP Alt-mode

-d SinkDevice #sink device from init.tsi
-pr "Setting Sink into 2 lane DP Alt-mode..."
-c 2lanes #run macro to set sink into 2 lane DP Alt-mode
-waiton ?2lanesModeOn #wait until ?2lanesModeOn is true (a macro)
-pr "Verifying DUT is in 2 lane DP Alt-mode..."
-c ?2lanesModeOn -ontrue -pr "Test PASSED" -pr "Test FAILED" #print results
-pr " "
```

Running TSI.EXE

The program can be run from the Windows command line or from scripting languages such as python. The program is executed as follows:

```
> tsi.exe [option1] [target1] [option2] [target2]
```

When executed without any parameters, help will be printed (-h).

See the Table below for options and targets.

Option	Target	Description
-o		Output level (0 – 3).
-err	"stop" or "continue"	Action on test fail (stop or continue).
-l		List connected devices.
-ll		List licenses (a device must be open).
-devinfo		Device information.

-pr	"text"	Print text on new line.
-prapp	"text"	Print text. Append to previous line.
-prlastread	"text"	Print text and CI from last .read=Value expression.
-prvar	Name Name+=Number Name-=Number Name*=Number Name/=Number	Print value of variable (loop or global variable) appended to the previous line. Number may be an integer or variable name. Operators may produce decimal output.
-d	device name	Select device.
-t		List source timings.
-p		List source patterns.
-m	file name	Load macro definitions from a file (default is macros.tsi).
-c	key=val.exp; or macro	Execute commands.
-r	file name	Execute commands from a file.
-x	program name	Launch program (*.exe or *.bat) and continue on close.
-xfeedback	Full path and name of application	Supplies the full path and name of application that is to be used to handle operator feedback when running tests that require them.
-if	.read=val.exp n1.RO.n2.LO. n3...	Conditionally run commands between paired -if -else (optional) or -else -endif. n1, n2 etc. are variables or numbers. RO is a relational operator (.eq. .ne. .lt. .gt.). LO is a logical operator (.and. .or.).
-else		Marks the end of the true block of an -if command.
-endif		Marks the end of an -if command.
-yesno	"text"	Prints "text" and asks the user to enter "Y" for yes and "N" for no (works as a truth value for -ontrue).
-var	Name=Number	Define or update a global variable. Number/N0,N1... may be integers or variable names. Variables created using the initializer list Name=N0,N1,... will be named Name[0],

	Name=N0,N1,.. .. Name+=Number Name- =Number Name*=Number Name/=Number	Name[1] with values N0, N1,... and the variable Name is the list SIZE. Values within braces can be variables and can be nested: <i>-var Test=Skip[bool[1]],Skip[bool[0]]</i> .
-loops	number of loops or Name=Number Name=VarName	Starts looping. Name can be used as a nested variable in -c commands. Name starts at zero when loop starts. (VarName is a variable name.)
-continue		Causes subsequent commands up to the -loops pairing -endloops to be skipped.
-endloops		Ends looping (with respect to preceding -loops command).
-skip		Skip, no operation (can be used with -ontrue).
-exit	"text"	Exit scripting and print text.
-ontrue		If preceding .read with expression was true, run the next command, else run that following it.
-timeout	time milliseconds	Timeout for options that require them.
-waiton	.read=val.exp;	Wait on a .read=Value.Expression to become true or time out (a device must be open).
-pic	file name	Capture and save a picture to file (a device must be open).
-beginhtms		Begin new html test report (a device must be open).
-endhtml		End html test report (a html file must be open).
-wpdosink	file name	Write device sink PDO data to file (a device must be open).
-rpdosink	file name	Read sink PDO data from file to device (must be open).

-wpdosource	file name	Write device source PDO data to file (a device must be open).
-rpdosource	file name	Read source PDO data from file to device(must be open).
-h		List help for each command.

- **-o** specifies the amount output desired: **-o 0** selects minimum output.
- **-err** specifies the whether to continue or stop when a test or query fails.
- **-l** lists the connected devices.
- **-ll** lists the licenses for an open device.
- **-devinfo** lists device information.
- **-pr** prints the target text on a new line.
- **-prapp** prints the target text appended to previous line (no carriage return).
- **-prlastread** prints the target text followed by the value of the last CI value read.
- **-prvar** prints the variable value (loop or global variable) appended to the previous line. The + - * and /

operators can be used to modify the output. These operators can be used to produce decimal output.

- **-d** selects a device by name (which must be listed in *init.tsi*).
 - **-t** lists source device timings.
 - **-p** lists source device patterns.
 - **-m** loads macro definitions from a file (the default *macros.tsi* is also loaded if found first).
- Important:** This is read in as it is parsed so commands that rely on these macros must come after. Macros with the same name overwrite those defined earlier.
- **-c** executes an expression of the form *Key=Value.Expression*; or a Macro.
 - **-r** executes commands from a file. These are equivalent to console program command parameters: Each space not within parentheses marks the start of a new option or target. Options and targets must be on the SAME line. **-c .read = TSI_R_XXX** would be interpreted as four command line parameters while **-c ".read=TSI_R_XXX"** would be interpreted as two command line parameters.
 - **-x** is used to launch an external program or run a batch file (see 1.2.4.1 EXAMPLES below). Use **** if you need to put parentheses within parentheses: "Like this **** Hello World****."
 - **-xfeedback** is used to handle operator feedback when running tests that require them. The target is the full path and name of application used to handle operator feedback (see "Operator feedback during test execution" in this manual). This needs to be run before the test: **-xfeedback "C:\TSIX\tsi.exe" -c .loadtd=test443.td**
 - **-if** conditionally runs commands between paired **-if-else** (optional) or **-else-endif** commands. It's target must be a **.read=Key.Expression** or a **n1.RO.n2.LO.n3...** type of expression. **n1, n2** etc. are variables or numbers. **RO** is a relational operator (**.eq. .ne. .lt. .gt.**). **LO** is a logical operator (**.and. .or.**). The latter expression type is evaluated from left to right. Ordering of the form **-if n0.eq.1.and.n1.ne.10.or.n2.ne.1.and.n3.lt.5.or.n4.eq.0** makes things easy to understand. See examples below.
 - **-else** marks the true block of an **-if** command.
 - **-endif** marks the end of an **-if** command.
 - **-yesno** Prints "text" and asks the user to enter "Y" for yes and "N" for no (works as a truth value for **-ontrue**).
 - **-var** defines or updates a global variable. Variable names must start with an alphabetic character and may also contain numerical digits as well as the underscores ("**_**") and brackets ("**[**" and "**]**"). Subsequent statements with the same variable name update the existing global variable. Global variables may be incremented or decremented via the **+=** and **-=** operators. Also the ***=** and **/=** operators are available for scaling variables. Built-in variables are also available (only *lastread*, which is the value of the last **.read=TSIX_XXX** operation, is

currently available). Variables defined are global except *-loops* variables which are nested (they vanish when *-endloops* is reached). If there is a nested and global variable of the same name, the nested variable is evaluated in expressions. Variables created using an initializer list such as *-var Name=N0,N1* will be named *Name[0]* and *Name[1]* with values *N0* and *N1*. *Name* itself is a variable and is the list *SIZE* (two in this example). Values within braces can be variables and can be nested: *-var Test=Skip[bool[1]],Skip[bool[0]]*.

- *-loops* starts a command loop with the target number of loops. The target can also be a *Name=Number* or *Name=VarName* (a variable name) expression. The name can be used in *-c* commands within the *-loops* block. Looping is to the next (same nesting level) *-endloops* command option. Loops can be nested and there must be the same number of *-loops* and *-endloops* command options.
- *-continue* causes subsequent commands up to the *-loops* pairing *-endloops* to be skipped.
- *-endloops* marks the end of a loop of the same nesting level started by the previous *-loops* command option.
- *-skip* is a no-operation and can be used with *-ontrue*.
- *-exit* exits scripting and prints a message (can be used with *-ontrue*, *-if-else-endif*).
- *-ontrue* is a conditional operator: If the preceding *.read* with expression (required) was true, the next command (*-option target*) is run, else that following it is run. It cannot be nested.
- *-timeout* will set the timeout time milliseconds (default is 10000) for commands that require them.
- *-waiton* will wait for a *.read=Value.Expression* to become true or a time out (a device must be open). Can be used to wait for a DUT to be plugged in, for example. Its main purpose though is making sure CI's to be tested are in a proper state for reading as setting CI's are asynchronous (see 1.2.4.1 EXAMPLES below).
- *-pic* is used to capture and save a picture to a file. A device sink must be open and a DUT source must be connected for this to succeed. The format output is a lossless bitmap (*.bmp).
- *-beginhtml* begins new html test report that receives output ONLY for tests (*-c .loadtd=test.td*). A device must be open. If the device is changed via a *-d Dev* command, the file will be closed (equivalent to *-endhtml*). The html output file will be of the form *<devsink>_20181008105224_log.html*, where *<devsink>* is the device name.
- *-endhtml* ends the current open html test report (a html file must be open).
- *-wpdosink* writes device sink PDO data (see *TSI_USBC_PWR_LOCAL_SINK_PDO*) to a file.
- *-rpdosink* reads sink PDO data from a file and sets it on the open device.
- *-wpdosource* writes device source PDO data (see *TSI_USBC_PWR_SOURCE_SINK_PDO*) to a file.
- *-rpdosource* reads source PDO data from a file and sets it on the open device.
- *-h* lists options and targets.

Examples

Looping colors spaces, patterns and timings (lastread below is a built-in global variable):

```
-timeout 20000 #Wait on video signal
-d SourceDevice #defined in init.tsi
-c .read=TSI_R_PG_PREDEF_TIMING_COUNT;
-var SzT=lastread -pr "Timeing count is " -prvar SzT
-c .read=TSI_R_PG_PREDEF_PATTERN_COUNT;
-var SzP=lastread -pr "Pattern count is " -prvar SzP
-var dpix=10; #delta value for pixel width and height tests
```

```

-var dhz=2; #delta value for frequency
-var bool=0,1 #defines bool[0]=0 and bool[1]=1
-var CsSkip=bool[0],bool[1],bool[1] # CsSkip[0]=0 CsSkip[1]=1 CsSkip[2]=1
-loops cs=3 #three color spaces
-loops p=SzP #p starts at 0 for patterns
  -loops t=SzT #t starts at 0 for timings
    -if CsSkip[cs].eq.1.and.p.ne.10
      -continue #only want to run non-RGB tests for pattern 10
    -endif
  -d SourceDevice
  -c TSI_W_PG_PREDEF_TIMING_SELECT=t;
  -c TSI_W_PG_PREDEF_PATTERN_SELECT=p;
  -if p.ne.10.and.cs.ne.0.or.cs.eq.0
    -c setcolorSpace_RGB #a macro
  -else
    -if cs.eq.1
      -c setcolorSpace_YcbCr420 #a macro
    -else
      -c setcolorSpace_YcbCr422 #a macro
    -endif
  -endif
  -c TSI_W_PG_COMMAND=1; #apply command
  -c .read=TSI_PG_CUSTOM_TIMING_HACTIVE;
  -var wdpix=lastread; # lastread is a global variable
  -c .read=TSI_PG_CUSTOM_TIMING_VACTIVE;
  -var htpix=lastread;
  -c .read=TSI_PG_CUSTOM_TIMING_FIELD_RATE;
  -var hz=lastread;
  -var hz/=100; # scaling to work with TSI_R_INPUT_FREQ below
  -pr "" -pr "Timing " -prvar t -prapp " " -prvar wdpix
  -prapp "x" -prvar htpix -prapp " @ " -prvar hz/10 -prapp "Hz"
  -prapp "\" -prapp "Pattern " -prvar p -prapp " "
  -c .read=TSI_R_PG_PREDEF_PATTERN_NAME.bytes.256
  -d SinkDevice
  -var wdpixmin=wdpix; -var wdpixmin-=dpix;
  -var wdpixmax=wdpix; -var wdpixmax+=dpix;

```

```

-waiton .read=TSI_R_INPUT_WIDTH.gt.wdpixmin.and.lt.wdpixmax;
-var htpixmin=htpix; -var htpixmin-=dpix;
-var htpixmax=htpix; -var htpixmax+=dpix;
-waiton .read=TSI_R_INPUT_HEIGHT.gt.htpixmin.and.lt.htpixmax;
-var hzmin=hz; -var hzmin-=dhz;
-var hzmax=hz; -var hzmax+=dhz;
-waiton .read=TSI_R_INPUT_FREQ.gt.hzmin.and.lt.hzmax;
-if .read=TSI_R_INPUT_WIDTH.eq.wdpix
  -if .read=TSI_R_INPUT_HEIGHT.eq.htpix
    -if .read=TSI_R_INPUT_FREQ.gt.hzmin.and.lt.hzmax
      -pr " ----> PASS"
    -else
      -prlastread "Frequency ----> FAILED: Value Read = "
    -endif
  -else
    -prlastread "Height ----> FAILED: Value Read = "
  -endif
-else
  -prlastread "Width ----> FAILED: Value Read = "
-endif
-endl loops #t=SzT
-endl loops #p=SzP
-endl loops #cs=3

```

Executing a *.exe and a *.bat could be as follows:

```

-x "C:\Windows\system32\notepad.exe"
-x "cmd.exe /c hello.bat"

```

Print message based on a .read=Value.Expression:

```

-c .read=TSI_R_USBC_DP_ALT_MODE_STATUS.bitson.0
-ontrue -pr "Alt Mode On." -pr "Alt Mode Off."

```

Print if type-C cable is in straight or flipped orientation:

```

-c .read = TSI_R_USBC_CABLE_STATUS.bitsoff.1;
-ontrue -pr "Strait orientation" -pr "Flipped orientation"

```

Looping:

```

-loops 2 #outer loop
-pr "outer loop cmd 0"

```

```

-pr "outer loop cmd 1"
-loops 3 #inner loop
  -pr "  inner loop cmd 0"
  -pr "  inner loop cmd 1"
-endloops #inner loop
-pr "outer loop cmd 2"
-endloops

```

Looping with variables:

```

-var Times=3,5,6,8,10
-var Pats=0,5,10
-loops i=Pats #outer loop 3 times
  -c TSI_W_PG_PREDEF_PATTERN_SELECT=Pats[i];
  -loops j=Times #inner loop 5 times
    -c TSI_W_PG_PREDEF_TIMING_SELECT=Times[j];
  -endloops
-endloops

```

Using if-else-endif:

```

-if .read = TSI_R_USBC_CABLE_STATUS.bitsoff.1
  -pr "Straight orientation"
-else
  -pr "Flipped orientation"
-endif

```

Output

Output is directly to the console window and possibly to an output file.

Output to a file is automatic when scripts are run from a file:

```
-r file.txt
```

The output file will be of the form: *file_20181008105217_log.txt*

Extra html output for tests (see 1.2.6 Running Tests below) can be started via *-beginhtml* and ended via *-endhtml*.

If the device is changed via a *-d Dev* command or if another *-beginhtml* command is executed, the file will be closed (equivalent to *-endhtml*).

The html output file will be of the form *<devsink>_20181008105224_log.html* where *<devsink>* is the device alias name.

Running Tests

Tests are of the form:

```
-c .loadtd = filename.td;
```

filename.td is a test that has been saved (from the UCD Console program for example).

Some tests require operator feedback: the DUT (device under testing) needs to be set into a defined state. Tsi.exe can be used for this. A typical test requiring feedback would be run as follows:

```
-xfeedback "C:\TSIX\tsi.exe" -c .loadtd=test443.td
```

When tsi.exe calls for operator feedback it will launch another instance of tsi.exe with various parameters (see "*Operator feedback during test execution*" in this manual). The first parameter is op=number which defines the operator feedback id. The parameters may look like: op=9;res_x=1280;res_y=800;res_frate=60000;res_bpp=12;exit_proceed=9; Tsi.exe takes these parameters (except for op and exit_proceed) and creates global variables with these names. exit_proceed is used for the instance return value. Tsi.exe then creates a single command of the form: -r OperatorFeedBackRequest_ID_9.txt (for op=9). Tsi.exe expects to find this file and proceeds to run it. It will exit with the value "exit_proceed" and the instance running the test will proceed. An example of a feedback file for op=2 could be:

```
#Operator Request op=2;res_x=640;res_y=480;res_frate=60000;exit_proceed=2
```

```
-o 0
```

```
-var testrun=0 #Set to 1 for testing this script
```

```
#below for testing
```

```
-if testrun.eq.1
```

```
  -var op=2
```

```
  -var res_x=640
```

```
  -var res_y=480
```

```
  -var res_frate=60000
```

```
  -var exit_proceed=2
```

```
-endif
```

```
#above for testing
```

```
#actual script to use as operator feedback below
```

```
-d SourceDevice #select device
```

```
-var timing=0 #timing variable set below
```

```
-r FindPredefinedTiming.txt #calculate timing id from res_x, res_y, res_frate
```

```
-c TSI_W_PG_PREDEF_TIMING_SELECT=timing
```

```
-pr "Predefined timing = " -prvar timing
```

```
-prapp " " -prvar res_x
```

```
-prapp " x " -prvar res_y
```

```
-prapp " @ " -prvar res_frate*0.001 -prapp "Hz"
```

```
-c TSI_W_PG_PREDEF_PATTERN_SELECT=9;
-c TSI_W_PG_COMMAND=4; #Set_PG_Output_On
-c WAIT_3s
```

When `-r OperatorFeedBackRequest_ID_2.txt` is run it will output a regular log file of the form: `OperatorFeedBackRequest_ID_2_20181102131212_log.txt` which can be used for checking and debugging.

Devices

Devices need to be listed in the file `init.tsi`. A method for creating this file from the command line is:

```
tsi.exe -l > init.tsi
```

The user can edit `init.tsi` and name the devices as desired (default names are `dev0`, `dev1`, etc.).

A physical device may have one or more “devices” in different roles. Devices are of the form:
Name = SerialNumber Role Technology

Examples:

Devices such as the UDC-400 are listed as:

```
Dev0 = 1813c261 source_sink dp;
```

Devices such as the UDC-340 are listed as:

```
Dev0 = 1823c395 sink usbc;
```

```
Dev1 = 1823c395 source usbc;
```

Devices such as the UDC-323 are listed as:

```
Dev0 = 1642c158 sink dp;
```

```
Dev1 = 1642c158 source dp;
```

```
Dev2 = 1642c158 sink hdmi;
```

```
Dev2 = 1642c158 source hdmi;
```

Devices such as the UDC-301 are listed as:

```
Dev0 = 1636c147 sink dp;
```

```
Dev1 = 1636c147 sink hdmi;
```

```
Dev2 = 1636c147 sink hdmi_spdif;
```

13. TESTSTAND INTEGRATION

This version of TSI introduces integration functions for National Instrument's TestStand. This section describes how to use these functions to configure the TE device, set TSI test parameters and run TSI tests.

Integration functions

TestStand integration is done by creating 4 new functions. Please notice that these functions are exported with C++ name mangling enabled in order to have the function parameter types show up in TestStand automatically.

TSI_TST_Init

ClientVersion 12, and higher No license requirements

```
void __cdecl TSI_TST_Init
(
    char *iSetupScript,
    bool &oErrors,
    long &oErrorCode,
    char oErrorMsg[1024]
);
```

Description

This function is expected to be used to initialize TSI library when using it through TestStand. The function will run the given script through the iSetupScript parameter. If the script contains errors, or the device(s) being used are not available oErrors is set to true, oErrorCode will have a machine readable error code, and the oErrorMsg will be set to a human-readable error message.

Parameters

iSetupScript

Pointer to a NULL terminated string. The string content can be either the setup-script directly, or a reference to a file that contains the setup script.

Important: For details on the scripting, please refer to 1.2 TSI Scripts in TestStand.

oErrors

Reference to a variable that receives error flag. The value is set to "true", if the function failed, or "false" if the function succeeded.

(Continued...)

(...Continued)

oErrorCode

Reference to a variable that receives an error code. Error codes are non-zero negative values. Zero (and positive values) indicate no error result.

oErrorMsg

Pointer to the first byte of a 1024 byte character array which receives a human readable error message if the function failed. If there were no errors, the resulting error message is empty.

Result

If the function succeeds, the error flag is set to “false”, Error code and Error message are cleared. If the function fails, the error flag is set to “true”, Error code and Error message variables are set to indicate the error.

TSI_TST_RunTest

ClientVersion 12, and higher No license requirements

```

void __cdecl TSI_TST_RunTest
(
    char *Device,
    char *ConfigScript,
    int TestID,
    bool &Passed,
    bool &Error,
    char oErrorMsg[1024],
    char oReportText[64000]
);

```

Description

The function runs the test indicated by TestID value. The test is executed on a hardware unit indicated by the Device value. Before the test is started, any script given in ConfigScript is executed. If the test passes, the Passed is set to true, and otherwise false. If the function encounters any errors, the Error flag is set to true. If Error flag is set to true, the oErrorMsg will contain a human readable error message. oReportText will contain test log as human readable text data.

Parameters

Device

Pointer to a string containing the name of the device which is to run the test. The device names are user defined in the initialization script at the time TSI_TST_Init was called.

ConfigScript

Pointer to a test configuration script. If the device is already configured this script can be empty. The script can be directly passed as string, or it can be a file-name from which the script is loaded.

(Continued...)

(...Continued)

TestID

Indicates which test to run. Please refer to chapter 6 for information about tests and their ID values.

Passed

Reference (pointer) to a boolean variable that will be set to “true” if the test was completed with pass status. If the test failed the variable will be set to “false”.

Error

Reference (pointer) to a boolean variable that will be set to “false” if the test completed successfully. If there were errors during the test execution, this variable set set to “true”.

oReportMsg

Pointer to a character buffer of 1024 characters. The buffer will receive a human readable error message in the event that “Error” was set to “true”.

oReportText

Pointer to a character buffer of 64000 characters. The buffer will receive test log printout as human readable ASCII text data.

Result

The function will process the given configuration script. If the script contains errors, the error flag will be set to “true”, and report message will contain message about the error in the script. If the script was processed without errors, it will continue to run the test indicated by the TestID and Passed, Error, oReportMsg and oReportText variables will be set according to the outcome of the test.

TSI_TST_RunScript

ClientVersion 12, and higher No license requirements

```
void __cdecl TSI_TST_RunScript
(
    char *Device,
    char *iScript,
    bool &Error,
    char oErrorMsg[1024],
    int *oIntTable,
    int iMaxCount
)
```

Description

This function is used to run scripts on a device without running any tests. In addition to test configuration scripts, the scripts passed to this function are also capable of returning integers from the script.

Parameters

Device

Pointer to a string containing the name of the device which is to run the test. The device names are user defined in the initialization script at the time TSI_TST_Init was called.

iScript

Pointer to a string containing the script to be processed. The script can be directly passed as string, or it can be a file-name from which the script is loaded.

Error

Reference (pointer) to a boolean variable which will be set to “true” if the script contained errors. If the script was completed without errors, this variable will be set to “false”.

oErrorMsg

Pointer to a character buffer of 1024 bytes which will receive a human readable error message if the script contained errors.

oIntTable

Pointer to a table of integers that will receive any values returned by the script. The returned values are placed into the output table in the order they were read.

iMaxCount

Size of the integer table passed in oIntTable, as count of integers.

Result

The given script is executed. Any values read are placed into the table given in oIntTable. In case of error in the scripts, the Error and oErrorMsg variables are set accordingly.

TSI_TST_Clean

ClientVersion 12, and higher No license requirements

```
void __cdecl TSI_TST_Clean()
```

Description

Release all allocated resources and close all devices. This function should be called when the TSI functions are not going to be used by the test program.

Result

All resources allocated by TSI are released and devices are closed.

TSI Scripts in TestStand

This section defines how to write scripts to be used with the TSI_TST_Init, TSI_TST_RunTest and TSI_TST_RunScript functions. As each of these functions has a slightly different use for the scripts, the scripts also are slightly different.

Please notice, that TSI has a two-levels of scripts. The scripts used with TestStand integration are considered simplistic, and it offers read/write operations for the configuration items. It has no flow control mechanisms or other advanced features – these types of features are expected to be provided on a higher level language (like within a test sequence in TestStand).

TestStand script syntax

The script language syntax follows these rules.

- A statement consists of a key and an associated value.
- The key and value must be on the same line of text.
- There must be an equal sign (=) between key and value.
- There must be a semi-colon character (;) after the value.
- If either key or value contain spaces, it must be enclosed with single-quote marks (').
- White-space characters that are not part of key nor value are ignored, and therefore can be used to format the script to be more readable.

Below are some examples of valid script lines. The first line assigns a filename into a configuration item, and the second one assigns a numeric value. You can find the used configuration item definitions elsewhere in this document.

```
·····TSI_LOG_FILE = 'C:\My\TSI\Logs\Log.txt'; ···
·····TSI_W_SCRI_DELAY = 1000;
```

Important: The (') characters are white-space characters, such as spaces or tabs.

Writing a script for TSI_TST_Init

The intention of an initialization script is to create a known environment for the rest of the test sequence and make test station configuration easier. If there is a need to distribute a TestStand test sequence to multiple stations, it can be very useful to put the initialization script into a file, and refer to that file from the TestStand test sequence. This way, each test station can be configured to use the appropriate hardware by modifying the script file with notepad, or some other simple text editor utility, while the TestStand test sequence itself remains unchanged.

The script provided for TSI_TST_Init through the iSetupScript pointer is used to define aliases for the devices used in the test sequence.

```
MyDevice = 'UCD-323 [1234C5678]: DisplayPort Reference Source';
```

The above script would define an alias “**MyDevice**”. The alias is later used to identify the device by providing it through the *Device* string pointer parameters available in the TSI_TST_RunTest and TSI_TST_RunScript functions. The “**MyDevice**” alias will refer to the device given in the value field – (In this case a UCD-323 device with S/N 1234C5678 being used as a DisplayPort source). Please notice that defining a device alias will also open the device. As a result, it is not possible to define multiple aliases that would require use the same physical hardware.

In addition to selecting a device, it is necessary to also select which interfaces are going to be used. TSI offers selection of an input, and an output. An input and output can be used simultaneously, provided this is supported by the hardware. Currently, this is supported by UCD-400 alone at the moment of writing this document. Below is a script that selects both, an input and an output:

```
MyDevice = 'UCD-400 [1234C5678]: DisplayPort Reference Sink and Source';
.output='DP-1';
.input='DP-1';
```

An output is selected with a special key “**.output**”, and input is selected with another special key “**.input**”. The values are expected to contain the name of input or output being selected for use. The “**.input**” and “**.output**” selectors apply to the previous device alias definition. Please do remember that available inputs and/or outputs depend on the type of hardware being used. These special keys (“**.output**” and “**.input**”) are available for use in a script that is provided for TSI_TST_Init, and can't be used in other scripts.

It is also possible to define multiple device aliases: Consider a case where two UCD-323 devices are being used, for example, to test a repeater device. One of the UCD-323 devices would be used as a source device, with the other one acting a sink device. A following script might be used in this case:

```
Source = 'UCD-323 [1234C567]: HDMI Reference Source';
.output = 'HDMI-1';
Sink = 'UCD-323 [4321C876]: HDMI Reference Sink';
.input = 'HDMI-1';
```

This will create two device aliases “**Source**” and “**Sink**”, with “**Source**” using the HDMI-1 output port, and “**Sink**” using the HDMI-1 input port.

Writing a script for TSI_TST_RunTest

The intention of script given to this function is to configure a test before running it. The script can also change device settings if necessary.

The script provided for TSI_TST_RunTest is used to configure test parameters. This can be done by assigning each configuration item one by one, or by loading a **TD-file** (Test Description file). The TD-Files can be saved from UCD-Console.

To load a TD-File from a script, use ‘.loadtd’ key, and assign the TD-File name to it. Example:

```
.loadtd = 'C:\my_td_files\Test 1.td';
```

Important: The TSI_TST_RunTest function will not run the test defined in the TD file. It will load the test settings from it, but it will run the test identified by the TestID parameter.

Alternatively, the test settings can be done one-by-one by assigning (one or more) configuration items. The following example sets the audio test parameters:

```
TSI_EXPECTED_SAMPLE_RATE = 96000;
TSI_EXPECTED_AUDIO_FREQUENCY = 1000;
TSI_AUDIO_FREQUENCY_TOLERANCE = 1;
TSI_AUDIO_GLITCH_DETECT_TRESHOLD = 250000;
TSI_AUDIO_GLITCHES_ALLOWED = 0;
```

Important: You can also refer to configuration items by their ID code number, not just their names. This can be useful, in case there is some kind of problem with identifying a configuration item by name.

Writing a script for TSI_TST_RunScript

The intention of script given to this function is to extend the functionality of the scripts.

The script provided for TSI_TST_RunScript is intended to be used for device control and status checking. The TSI_TST_RunScript has additional parameters that enable 32 bit data words to be returned from the scripts. These scripts can also load TD-Files, same way as scripts provided to TSI_TST_RunTest.

To read 32-bit data and return it out of the script, use ‘.read’ key, and assign the configuration item ID to read. See below for an example:

```
.read = TSI_R_INPUT_WIDTH;
.read = TSI_R_INPUT_HEIGHT;
```

The above script would return two 32-bit integer values through the oIntTable pointer. The 32-bit word at index 0 would contain the current x-resolution, and the 32-bit word at index 1 would contain the current y-resolution.

Any actions based on the returned data must be decided on the TestStand side. This scripting system is very simple, and has no flow control or other advanced features.

APPENDIX A. PRODUCT SPECIFICATION

UCD-240

Test Connections	USB Type-C (Dual Role Port) USB Type-A (test signal Device) pass-thru USB Type-B (Host) pass-thru External Power Source / Sink connector
DP Alt Mode	Resolution up to 4096×2160p60 Up to HBR2 rate in up to 4 lanes Color depth up to 48 bits Support HDCP 1.3 and 2.2
USB Over USB-C	USB 3.1 Gen1 (5 Gbps) and USB 2.0 pass-thru
USB Power Delivery	Sink and source 5 V up to 3.0 A, up to 20 V / 5 A with external power test unit (Optional)
Electrical Test	Verify current flow in USB Type-C interface signals (VBUS, GROUND, CC1/2, SBU1/2). (Optional)
Computer interface	USB 3.0 and USB 2.0
Software	Windows 10, 8 and 7 compatible software driver UCD Console application for Windows TSI API with ready Test Cases
Power supply	AC/DC Power supply (100 to 240 Vac 50/60 Hz input, +12 Vdc output)
Mechanical Size	281 × 128 × 62 mm
Weight	0.9 kg w/o power supply

All specifications are subject to change without notice.

Version History

Rev.	Date	Author	Description
1	12.12.2018	JEs	- Created first version
2	29.4.2019	JEs	- First published version
3	17.12.2019	PKO	- Minor corrections
4	24.1.2020	PKO	- Minor correction, chapter 4. Connecting DUT
5	11.2.2020	PKO	- Minor corrections