TSI

Full Reference Manual

16. March 2018

UNIGRAF

Copyright

Notice

The information in this manual has been verified on the date of issue. The authors reserve rights to make any changes to this product and revise the information without obligation to notify any person about such revisions or changes.

Edition

Title TSI Reference Manual

Document ID

Issue date 16. March 2018

Company information

Unigraf Oy

Piispantilankuja 4
FI-02240 ESPOO
Finland

Phone. +358 9 589 550

e-mail: info@unigraf.fi
web: http://www.unigraf.fi

Trademarks

Table of Contents

1. General....9
1.1. About this document....9
1.1.1. History....9
1.2. Acronyms and abbreviations....13

2. Components and features....14
2.1. API DLL....14
2.1.1. Features....14
2.2. Using the TSI API....15

3. Functions....16
3.1. External functions....16
3.1.1. TSI_LoadAPI....16
3.1.2. TSI_UnloadAPI....17
3.2. API Base level functions....18
3.2.1. TSI_Init....18
3.2.2. TSI_Clean....19
3.3. Device management functions....20
3.3.1. TSI_DEV_GetParameterCount....20
3.3.2. TSI_DEV_GetParameterID....21
3.3.3. TSI_DEV_SetSearchMask....22
3.3.4. TSI_DEV_GetDeviceInfo....23
3.3.5. TSI_DEV_GetDeviceCount....24
3.3.6. TSI_DEV_GetDeviceName....25
3.3.7. TSI_DEV_Select....26
3.3.8. TSI_DEV_Location....27
3.4. Input management functions....28
3.4.1. TSI_VIN_GetParameterCount....28
3.4.2. TSI_VIN_GetParameterID....29
3.4.3. TSI_VIN_GetInputCount....30
3.4.4. TSI_VIN_GetInputName....31
3.4.5. TSI_VIN_Select....32
3.4.6. TSI_VIN_Enable....33
3.4.7. TSI_VIN_Disable....34
3.5. Output management functions....35
3.5.1. TSI_VOUT_GetParameterCount....35
3.5.2. TSI_VOUT_GetParameterID....36
3.5.3. TSI_VOUT_GetOutputCount....37
3.5.4. TSI_VOUT_GetOutputName....38
3.5.5. TSI_VOUT_Select....39
3.5.6. TSI_VOUT_Enable....40
3.5.7. TSI_VOUT_Disable....41
3.6. Video Preview functions....42
3.6.1. TSI_VPREV_SetWindowHandle....42
3.7. Audio Preview Functions....43
3.7.1. TSI_APREV_SetWindowHandle....43
3.7.2. TSI_APREV_GetDeviceCount....44
3.7.3. TSI_APREV_GetDeviceName....45
3.7.4. TSI_APREV_SelectDevice....46
3.8. Test system related functions....47
3.8.1. TSI_TS_GetTestCount....47
3.8.2. TSI_TS_GetTestInfo....48
3.8.3. TSI_TS_GetTestParameterCount....49
3.8.4. TSI_TS_GetReqParameterID....50
3.8.5. TSI_TS_Clear....51
3.8.6. TSI_TS_SetConfigItem....52
3.8.7. TSI_TS_GetConfigItem....53
3.8.8. TSI_TS_SaveConfig....54
3.8.9. TSI_TS_LoadConfig....55
3.8.10. TSI_TS_RunTest....56
3.8.11. TSI_TS_CaptureReference....57

3.8.12. TSI_TS_WaitInputSignal....58
3.9. Misc functions....59
3.9.1. TSI_MISC_SaveReference....59
3.9.2. TSI_MISC_LoadReference....60
3.9.3. TSI_MISC_SetOption....61
3.9.4. TSI_MISC_GetErrorDescription....62
3.10. Status log functions....63
3.10.1. TSI_STLOG_GetMessageCount....63
3.10.2. TSI_STLOG_Clear....63
3.10.3. TSI_STLOG_GetMessageData....64
3.10.4. TSI_STLOG_WaitMessage....65
3.11. Report generator functions....66
3.11.1. TSI_REP_BeginLogRecord....66
3.11.2. TSI_REP_EndLogRecord....67

4. Types and test definitions....68
4.1. Types....68
4.1.1. TSI_VERSION_ID....68
4.1.2. TSI_RESULT....68
4.1.3. TSI_DEVICE_ID....68
4.1.4. TSI_INPUT_ID....68
4.1.5. TSI_FLAGS....68
4.1.6. TSI_AUDIO_DEVICE_ID....68
4.1.7. TSI_CONFIG_ID....68
4.1.8. TSI_TEST_ID....69
4.1.9. TSI_OPTION_ID....69
4.2. Tests....70
4.2.1. Compare video frame sequence with a single reference....70
4.2.2. Validate audio signal frequency and glitch-free audio reproduction....74
4.2.3. Electrical Test Set / Power test....76
4.2.4. Electrical Test Set / HPD test....77
4.2.5. Electrical Test Set / DDC and CEC test....78
4.2.6. Electrical Test Set / TMDS test....79
4.2.7. CEC functional Test set / CEC functional test....80
4.2.8. Electical Test Set / Main Link test....81
4.2.9. Electrical Test Set / AUX test....82
4.2.10. Electrical Test Set / HPD test....83
4.2.11. CRC based Video Test set / CRC based single frame reference video test....84
4.2.12. CRC based Video Test set / CRC based single frame video stability test....85
4.2.13. CRC based Video Test set / CRC based sequence of frames reference video test....86
4.2.14. CRC Based Video Test Set / CRC based continuous sequence of reference frames....87
4.2.15. Link Test set / Link Training at All Supported Lane Counts and Link Rates....89
4.2.16. USBC Electrical Test Set / Up Face port CC and Vconn test....90
4.2.17. USBC Electrical Test Set / AUX (SBU) lines test....92
4.2.18. USBC Electrical Test Set / DUT as Power Sink....94
4.2.19. USBC Electrical Test Set / DUT as Power Source....96

5. Configuration Item definitions....98
5.1. Generic realtime measurements....98
5.1.1. ADC Data access CI range....98
5.1.2. ADC Data available on UCD-340....99
5.1.3. TSI_W_USBC_ADC_CTRL....99
5.2. Generic low-level test results....100
5.2.1. RAW test results access CI range....100
5.2.2. TSI_R_TDATA_BLOCK_SIZE....100
5.2.3. TSI_R_TDATA_GENERIC_STRUCT_VERSION....101
5.2.4. TSI_R_TDATA_USBC_EL_VCC*....101
5.2.5. TSI_R_TDATA_USBC_EL_VCONN*....101
5.2.6. TSI_R_TDATA_USBC_VAUX1_*....102
5.2.7. TSI_R_TDATA_USBC_VAUX2_*....102
5.2.8. TSI_R_TDATA_USBC_EL_VBUS_V....102
5.2.9. TSI_R_TDATA_USBC_EL_VBUS_I*....103
5.2.10. TSI_R_TDATA_USBC_EL_GND_I*....103
5.3. Reference frames....104
5.3.1. TSI_REF1_WIDTH....104
5.3.2. TSI_REF1_HEIGHT....104
5.3.3. TSI_REF1_ELEMENT_SIZE....104
5.3.4. TSI_REF1_ELEMENT_WIDTH....105
5.3.5. TSI_REF1_ELEMENT_HEIGHT....105

5.3.6. TSI_REF1_COLOR_DEPTH....105
5.3.7. TSI_REF1_ELEMENT_FORMAT....106
5.3.8. TSI_REF1_FRAME_DATA....106
5.4. Input video format....107
5.4.1. TSI_R_INPUT_WIDTH....107
5.4.2. TSI_R_INPUT_HEIGHT....107
5.4.3. TSI_R_INPUT_FREQ....107
5.4.4. TSI_R_INPUT_ELEMENT_SIZE....108
5.4.5. TSI_R_INPUT_ELEMENT_WIDTH....108
5.4.6. TSI_R_INPUT_ELEMENT_HEIGHT....108
5.4.7. TSI_R_INPUT_COLOR_DEPTH....109
5.4.8. TSI_R_INPUT_ELEMENT_FORMAT....109
5.4.9. TSI_R_INPUT_INTERLACE....109
5.5. Input audio format....110
5.5.1. TSI_R_AUDIO_CHANNELS....110
5.5.2. TSI_R_AUDIO_SAMPLE_RATE....110
5.5.3. TSI_R_AUDIO_SAMPLE_SIZE....110
5.5.4. TSI_CAPTURE_AUDIO_MASK....111
5.6. V-by-One inputs....111
5.6.1. TSI_VX1_SIGNAL_COLOR_DEPTH....111
5.6.2. TSI_VX1_SIGNAL_CHANNELS_PER_UNIT....111
5.6.3. TSI_VX1_SIGNAL_SYNC_MODE....112
5.6.4. TSI_VX1_HTPDN_CONTROL....112
5.6.5. TSI_VX1_LOCKN_CONTROL....112
5.6.6. TSI_VX1_LOCKN_DELAY....112
5.6.7. TSI_VX1_VIDEO_VALID_DELAY....113
5.6.8. TSI_VX1_FRAME_COMBINE_METHOD....113
5.6.9. TSI_VX1_SECTION_COUNT....113
5.7. LVDS Inputs....114
5.7.1. TSI_LVDS_CHANNELS....114
5.7.2. TSI_LVDS_SIGNAL_COLOR_DEPTH....114
5.7.3. TSI_LVDS_MAPPING_MODE....114
5.8. Video test....115
5.8.1. TSI_TEST_LENGTH....115
5.8.2. TSI_LIM_FRAME_MISMATCHES....115
5.8.3. TSI_LIM_PIXEL_MISMATCHES....115
5.8.4. TSI_PIXEL_TOLERANCE....116
5.8.5. TSI_MAX_AUTO_SAVE_FAILED....116
5.8.6. TSI_FAILED_FRAME_TARGET_FOLDER....116
5.8.7. TSI_MAX_EXPORT_FAILED....116
5.8.8. TSI_R_VIDEO_TEST_RAW_RESULTS_DATA....117
5.8.9. TSI_EXPORTED....117
5.8.10. TSI_EXPORT_ACCESS_INDEX....117
5.8.11. TSI_EXPORT_WIDTH....118
5.8.12. TSI_EXPORT_HEIGHT....118
5.8.13. TSI_EXPORT_ELEMENT_SIZE....118
5.8.14. TSI_EXPORT_ELEMENT_WIDTH....118
5.8.15. TSI_EXPORT_ELEMENT_HEIGHT....119
5.8.16. TSI_EXPORT_COLOR_DEPTH....119
5.8.17. TSI_EXPORT_ELEMENT_FORMAT....119
5.8.18. TSI_EXPORT_FRAME_DATA....120
5.9. Audio test....120
5.9.1. TSI_EXPECTED_SAMPLE_RATE....120
5.9.2. TSI_EXPECTED_AUDIO_FREQUENCY....120
5.9.3. TSI_AUDIO_FREQUENCY_TOLERANCE....121
5.9.4. TSI_AUDIO_GLITCH_DETECT_TRESHOLD....121
5.9.5. TSI_AUDIO_GLITCHES_ALLOWED....121
5.10. HDMI Receiver Electrical tests....122
5.10.1. TSI_HDMI_RX_TIMEOUT....122
5.10.2. TSI_HDMI_RX_POWER_LOW_LIMIT....123
5.10.3. TSI_HDMI_RX_POWER_HIGH_LIMIT....123
5.10.4. TSI_HDMI_RX_LINK_LOW_LIMIT....123
5.10.5. TSI_HDMI_RX_LINK_HIGH_LIMIT....124
5.10.6. TSI_HDMI_RX_HPD_ZERO_LOW_LIMIT....124
5.10.7. TSI_HDMI_RX_HPD_ZERO_HIGH_LIMIT....124
5.10.8. TSI_HDMI_RX_HPD_ONE_LOW_LIMIT....125
5.10.9. TSI_HDMI_RX_HPD_ONE_HIGH_LIMIT....125
5.10.10. TSI_HDMI_RX_DDC_LOW_LIMIT....125
5.10.11. TSI_HDMI_RX_DDC_HIGH_LIMIT....125
5.10.12. TSI_HDMI_RX_CEC_ZERO_LOW_LIMIT....126
5.10.13. TSI_HDMI_RX_CEC_ZERO_HIGH_LIMIT....126

5.10.14. TSI_HDMI_RX_CEC_ONE_LOW_LIMIT....126
5.10.15. TSI_HDMI_RX_CEC_ONE_HIGH_LIMIT....126
5.11. DP Receiver electrical tests....127
5.11.1. TSI_DP_RX_TEST_TIMEOUT....127
5.11.2. TSI_DP_RX_LINKS_*_VOLTAGE....128
5.11.3. TSI_DP_RX_HPD_ZERO_*_VOLTAGE....128
5.11.4. TSI_DP_RX_HDP_ONE_*_VOLTAGE....128
5.11.5. TSI_DP_RX_AUX_P_IDLE_*_VOLTAGE....129
5.11.6. TSI_DP_RX_AUX_N_IDLE_*_VOLTAGE....129
5.11.7. TSI_DP_RX_AUX_*_TRIG_VOLTAGE....129
5.11.8. TSI_DP_RX_AUX_SIGNAL_CAPT_TIMEOUT....130
5.11.9. TSI_DP_RX_AUX_SIGNAL_CAPT_TRIES....130
5.11.10. TSI_DP_RX_MAX_DUT_LANE_COUNT....130
5.11.11. TSI_DP_RX_MAX_DUT_LINK_RATE....130
5.12. Accessing Info frames....131
5.12.1. TSI_R_HDMI_INFOFRAME_RANGE_*....131
5.12.2. TSI_R_HDMI_INFOFRAME_UPDATE_FLAGS....131
5.12.3. Additional Info-frame CI definitions, and update bits....132
5.13. Miscellaneous....133
5.13.1. TSI_R_GENERIC_STATUS....133
5.13.2. TSI_R_UNITS_PRESENT....133
5.13.3. TSI_W_FORCE_HOT_PLUG_STATE....134
5.13.4. TSI_EDID_TE_INPUT....134
5.13.5. TSI_EDID_TE_OUTPUT....134
5.13.6. TSI_VERSION_TEXT....135
5.13.7. TSI_LOG_FILE....135
5.13.8. TSI_HPD_LENGTH....135
5.13.9. TSI_W_ARC_CONTROL....136
5.14. CRC based video tests....137
5.14.1. TSI_CRC_TIMEOUT....137
5.14.2. TSI_CRC_FRAMES_TO_TEST....137
5.14.3. TSI_CRC_LIM_FRAME_MISMATCHES....138
5.14.4. TSI_CRC_REF_WIDTH....138
5.14.5. TSI_CRC_REF_HEIGHT....138
5.14.6. TSI_CRC_REF_COLORDEPTH....138
5.14.7. TSI_CRC_REFERENCE_CRC_VALUES....139
5.14.8. TSI_CRC_REQUIRED_FRAME_RATE....139
5.14.9. TSI_CRC_FRAME_RATE_TOLERANCE....139
5.14.10. TSI_CRC_MOTION_TEST_ITERATIONS....140
5.14.11. TSI_CRC_COLOR_FORMAT....140
5.15. DP RefSource simple link test....141
5.15.1. TSI_DP_LTT_TIMEOUT....141
5.15.2. TSI_DP_LTT_MAX_LANE_COUNT....141
5.15.3. TSI_DP_LTT_MAX_RATE....142
5.15.4. TSI_DP_LTT_HPD_PULSE_DURATION....142
5.15.5. TSI_DP_LTT_LT_START_TIMEOUT....142
5.15.6. TSI_DP_LTT_TEST_LOOP_DELAY....142
5.16. HDCP Debugging configuration items....143
5.16.1. TSI_R_HDCP_1X_STATUS....144
5.16.2. TSI_W_HDCP_1X_COMMAND....146
5.16.3. TSI_R_HDCP_2X_STATUS....147
5.16.4. TSI_W_HDCP_2X_COMMAND....149
5.16.5. TSI_W_FORCE_HOT_PLUG_STATE....150
5.17. DP Sink – Link status....151
5.17.1. TSI_R_DPRX_LINK_STATUS_FLAGS....152
5.17.2. TSI_R_DPRX_LT_STATUS_FLAGS....153
5.17.3. TSI_R_DPRX_LINK_VOLTAGE_SWING....154
5.17.4. TSI_R_DPRX_LINK_PRE_EMPHASIS....154
5.17.5. TSI_R_DPRX_LINK_LANE_COUNT....155
5.17.6. TSI_R_DPRX_LINK_RATE....155
5.17.7. TSI_R_DPRX_ERROR_COUNTS....155
5.17.8. TSI_W_DPRX_DPCD_BASE....156
5.17.9. TSI_DPRX_DPCD_DATA....156
5.18. DP Sink - Capabilities....157
5.18.1. TSI_DPRX_MAX_LANES....157
5.18.2. TSI_DPRX_MAX_LINK_RATE....157
5.18.3. TSI_DPRX_LINK_FLAGS....158
5.19. Configuration items for USB Type-C....159
5.19.1. TSI_W_USBC_CABLE_CONTROL....161
5.19.2. TSI_W_USBC_INITIAL_ROLE....162

5.19.3. TSI_USBC_DP_ALT_MODE_SETUP....162
5.19.4. TSI_W_USBC_ROLE_CONTROL....163
5.19.5. TSI_W_USBC_DP_ALT_MODE_COMMAND....164
5.19.6. TSI_R_USBC_TE_HW_CONFIGURATION....164
5.19.7. TSI_R_USBC_CABLE_STATUS....165
5.19.8. TSI_R_USBC_IDO_TABLE....166
5.19.9. TSI_R_USBC_ROLE_STATUS....166
5.19.10. TSI_R_USBC_DP_ALT_MODE_STATUS....167
5.19.11. TSI_R_USBC_POWER_STATUS....168
5.19.12. TSI_R_USBC_POWER_SOURCE_PDO....169
5.19.13. TSI_R_USBC_POWER_SINK_RDO....169
5.19.14. TSI_R_USBC_IDO_TABLE....169
5.19.15. TSI_USBC_EPU_LOAD_CONTROL....170
5.19.16. TSI_USBC_PWR_CONTRACT_CONTROL....171
5.19.17. TSI_USBC_PWR_CONTRACT_SELECT....172
5.19.18. TSI_USBC_PWR_LOCAL_SINK_PDO....172
5.19.19. TSI_USBC_PWR_LOCAL_SOURCE_PDO....172
5.19.20. TSI_R_USBC_PWR_REMOTE_SINK_PDO....173
5.19.21. TSI_R_USBC_PWR_REMOTE_SOURCE_PDO....173
5.20. USB Type-C Electrical tests....174
5.20.1. TSI_USBC_EL_TIMEOUT....174
5.20.2. TSI_USBC_EL_DUT_CAPS....174
5.20.3. TSI_USBC_EL_REPLUG_TIME....175
5.20.4. TSI_USBC_EL_DUT_ATTACH_TIMEOUT....175
5.20.5. TSI_USBC_EL_PWR_CONTRACT_TIMEOUT....175
5.20.6. TSI_USBC_EL_CC_LOW_VOLTAGE_1....176
5.20.7. TSI_USBC_EL_CC_HI_VOLTAGE_1....176
5.20.8. TSI_USBC_EL_CC_LOW_VOLTAGE_2....176
5.20.9. TSI_USBC_EL_CC_HI_VOLTAGE_2....177
5.20.10. TSI_USBC_EL_CC_LOW_VOLTAGE_3....177
5.20.11. TSI_USBC_EL_CC_HI_VOLTAGE_3....177
5.20.12. TSI_USBC_EL_VCON_LOW_VOLTAGE....178
5.20.13. TSI_USBC_EL_VCON_HI_VOLTAGE....178
5.20.14. TSI_USBC_EL_DP_ALT_TIMEOUT....178
5.20.15. TSI_USBC_EL_AUX_P_IDLE_LOW_VOLTAGE....179
5.20.16. TSI_USBC_EL_AUX_P_IDLE_HI_VOLTAGE....179
5.20.17. TSI_USBC_EL_AUX_N_IDLE_LOW_VOLTAGE....179
5.20.18. TSI_USBC_EL_AUX_N_IDLE_HI_VOLTAGE....180
5.20.19. TSI_USBC_EL_VBUS_LOW_VOLTAGE....180
5.20.20. TSI_USBC_EL_VBUS_HI_VOLTAGE....180
5.20.21. TSI_USBC_EL_VBUS_CURRENT_MAX_DEV....181
5.20.22. TSI_USBC_EL_GND_CURRENT_MAX_DEV....181
5.20.23. TSI_USBC_EL_PWR_MEASURE_DELAY....181
5.20.24. TSI_USBC_EL_MIN_DUT_CURRENT....182
5.21. Pattern generator CI definitions....183
5.21.1. TSI_PG_ENABLED_STREAM_COUNT....183
5.21.2. TSI_R_PG_MAX_STREAM_COUNT....183
5.21.3. TSI_PG_STREAM_SELECT....184
5.21.4. TSI_W_PG_COMMAND....184
5.21.5. TSI_PG_CUSTOM_TIMING_HTOTAL....184
5.21.6. TSI_PG_CUSTOM_TIMING_HSTART....185
5.21.7. TSI_PG_CUSTOM_TIMING_HACTIVE....185
5.21.8. TSI_PG_CUSTOM_TIMING_HSYNCW....185
5.21.9. TSI_PG_CUSTOM_TIMING_VTOTAL....186
5.21.10. TSI_PG_CUSTOM_TIMING_VSTART....186
5.21.11. TSI_PG_CUSTOM_TIMING_VACTIVE....186
5.21.12. TSI_PG_CUSTOM_TIMING_VSYNCW....187
5.21.13. TSI_PG_CUSTOM_TIMING_FLAGS....188
5.21.14. TSI_PG_CUSTOM_TIMING_FIELD_RATE....189
5.21.15. TSI_R_PG_PREDEF_TIMING_COUNT....189
5.21.16. TSI_W_PG_PREDEF_TIMING_SELECT....189
5.21.17. TSI_PG_PREDEF_PATTERN_COUNT....190
5.21.18. TSI_W_PG_PREDEF_PATTERN_SELECT....190
5.21.19. TSI_R_PG_PREDEF_PATTERN_NAME....190
5.21.20. TSI_R_PG_PREDEF_PATTERN_ID....191
5.21.21. TSI_PG_PREDEF_PATTERN_PARAMS....191
5.21.22. TSI_PG_CUSTOM_PATTERN_WIDTH....192
5.21.23. TSI_PG_CUSTOM_PATTERN_HEIGHT....192
5.21.24. TSI_PG_CUSTOM_PATTERN_PIXEL_FORMAT....193
5.21.25. TSI_PG_CUSTOM_PATTERN_DATA....194
5.22. Displayport interface specific CI definitions....195
5.22.1. TSI_SRC_DP_LINK_CFG_LANES....195
5.22.2. TSI_SRC_DP_LINK_CFG_BIT_RATE....195
5.22.3. TSI_SRC_DP_LINK_CFG_FLAGS....196

5.22.4. TSI_SRC_DP_OVERRIDE_VOLTAGE_SWING....197
5.22.5. TSI_SRC_DP_OVERRIDE_PRE_EMPHASIS....198
5.22.6. TSI_SRC_DP_LINK_PATTERN....199
5.22.7. TSI_W_SRC_DP_COMMAND....199
5.22.8. TSI_R_SRC_DP_HPD_STATUS....200
5.22.9. TSI_R_SRC_DP_LT_RESULT....200
5.22.10. TSI_R_SRC_DP_LINK_STATUS_BITS....201
5.22.11. TSI_R_SRC_DP_LINK_STATUS_VOLT_SWING....202
5.22.12. TSI_R_SRC_DP_LINK_STATUS_LANE_COUNT....202
5.22.13. TSI_R_SRC_DP_LINK_STATUS_BIT_RATE....203
5.22.14. TSI_R_SRC_DP_LINK_STATUS_PRE_EMP....203
5.23. CEC Functional test....204
5.23.1. TSI_HDMI_RX_CEC_TIMEOUT....204
5.23.2. TSI_HDMI_RX_CEC_LOCAL_PHY_ADDR....204
5.24. HDMI Source specific....205
5.24.1. TSI_W_SRC_HDMI_CONTROL....205
5.24.2. TSI_R_SRC_HDMI_STATUS....206
5.24.3. TSI_R_SRC_HDMI_DUT_CAPS....206
5.25. HDMI Sink specific....207
5.25.1. TSI_R_HDRX_LINK_STATUS....207
5.25.2. TSI_W_HDRX_LINK_CONTROL....207
5.26. Error codes....208

1. GENERAL

1.1.About this document

This document applies to TSI software release 1.9 [R11] (TSI.DLL [1.9.11])

1.1.1.History

- 8.7.2014
  Initial version for evaluation.
- 4.8.2014
  Revised for first release. Added error codes. Added history entry for first release.
- 2.9.2014
  Revised for second release. Added error codes, Added status log function descriptions, Added reference frame capture function description, Added description of test 2, Added descriptions of new configuration items.
- 27.11.2014
  Added Input parameter related functions, Added device parameter related functions, Added load/save reference functions, Added audio graphical preview function.
- 15.1.2015
  Finalized and revised for release 1.2
- 22.1.2015
  Added message log descriptions to tests
- 13.2.2015
  Added V-by-One related configuration item descriptions.
- 30.4.2015
  Added missing configuration item definitions, Revised 3.4.6 TSI_VIN_Enable, 3.4.5 TSI_VIN_Select, Test run example logs updated. Revised and finalized for release 1.2 [R2], Replaced Vx1 short with “V-by-One”, except for defines and references to defines.
- 26.6.2015
  Revised the history section. Updated TSI_TS_RunTest description. Updated 2.2 Using the TSI API with more detailed information.
- 18.8.2015
  Added descriptions for functions 3.11.1 TSI_REP_BeginLogRecord, 3.11.2 TSI_REP_EndLogRecord and 3.10.4 TSI_STLOG_WaitMessage
- 14.10.2015
  Revised error descriptions; Added TSI_R_INPUT_INTERLACE configuration item; Added TSI_TS_WaitInputSignal, Revised for 1.3 [R6] release.
- 29.6.2016
  Revised error description, Added configuration items TSI_R_INFOFRAME_* and TSI_HDMI_RX_*, Added ARC configuration item.

(Continued...)

(...Continued)

- 24.8.2016
  Revised for TSI 1.6 [R1]. Added PPM file type ID to save reference function, Added DP RX electrical test parameter definitions table, Added description of HDMI and DP electrical tests, Added CEC functionality test, Changed the info frame access interface to be more extensible.

- 24.11.2016
  Revised for TSI 1.6 [R3]. Added CRC Test definitions, Added CRC configuration item definitions.

- 9.12.2016
  Revised for TSI 1.6 [R4]. Removed un-used CI definitions.

- 20.12.2016
  Revised for TSI 1.6 [R6]. Added UCD-1 related CI definitions.

- 12.1.2017
  Revised for software bundle release 1.0.3. Added HDCP Debugging CI definitions, revised UCD-1 and UCD-2 related configuration item definitions.

- 1.3.2017
  Revised for software bundle release 1.0.5. Added DP Sink Link status and DP Sink Link Configuration CI's. Added DisplayPort Reference Source Simple Link test, and related CI's. Added missing CI definition of TSI_HPD_LENGTH.

- 28.4.2017
  Revised for software bundle release 1.1.0. Added USB-Type C related configuration item definitions from the UCD-340 support proposal version 1.1b, section 5.19. Added USB Type-C electrical test definitions, sections 4.2.16, 4.2.17, 4.2.18 and 4.2.19. Added CI definitions relating to USB Type-C electrical tests, section 5.20

- 10.7.2017
  Revised for TSI 1.8 [R4] and software bundle 1.2.3. Updated electrical test parameter definitions to match with implementation. Added missing USB Type-C electrical test parameter (Chapter 5.20.23). Revised USB Type-C electrical test descriptions (chapters 4.2.16, 4.2.17, 4.2.18 and 4.2.19) to match with implementation.

- 14.7.2017
  Fixed mistakes/typos in chapters 4.2.18 and 4.2.19, Added description for function TSI_DEV_Location, chapter 3.3.8. This manual version is still for TSI 1.8 [R4].

- 8.9.2017
  Revised for TSI 1.9 [R2] and software bundle release 1.3.2. Split section for moving the configuration items into section 5. Added 5.2, Re-formatted sections 5.3, 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9; Respective old sections were removed. Added new functions for Source functionality, section 3.5. Added CI definitions used with Sources, sections 5.21 and 5.22. Removed definitions of unused CI's. Added cross references to RAW results sections from tests that deliver this type of information, Sections 4.2.16, 4.2.17, 4.2.18 and 4.2.19.
  Important: *This version of the manual is preliminary.*

- 15.9.2017
  Revised for TSI 1.9 [R3] and software bundle release 1.3.3. Added sections 5.19.14 and 5.19.15.
  Important: *This version of the manual is preliminary.*

(Continued...)

(...Continued)

- 19.9.2017
  Revised for TSI 1.9 [R4] and software bundle release 1.3.4. Added sections 5.19.16, 5.19.17, 5.19.18, 5.19.19, 5.19.20 and 5.19.21
  Important: *This version of the manual is preliminary.*

- 30.10.2017
  Revised for TSI 1.9 [R6] and software bundle release 1.3.8. Renamed sections 4.2.3 through to 4.2.19 to match with actual test names displayed by software applications. Fixed broken cross-references in sections 3.4.5, 3.8.6, 3.9.3 and 4.2.1. Reformatted sections from 5.10 to 5.15; Respective old sections were removed. Added section 5.1. Revised sections 5.13, 5.21.4, 5.21.8, 5.21.12, 5.21.13, 5.21.15, 5.21.16 and 5.26

- 24.11.2017
  Revised for TSI 1.9 [R7] and software bundle release 1.3.9. Added custom pattern use details to sections 5.21.18 and 5.21.25. Added two missing CRC test configuration items 5.14.10 and 5.14.11. Revised 5.14.2 to contain correct information. Additional information describing sections 3.1 and 3.2. Some minor cosmetic updates.

- 12.12.2017
  Revised for TSI 1.9 [R8] and software bundle release 1.3.10. Added new USB-Type C parameter 5.20.24. Revised 4.2.18.

- 5.1.2018
  Added description of the fourth CRC based video test that was missing from previous version (4.2.14). Only the manual was updated, no new TSI release was done.

- 16.3.2018
  Revised description of TSI_DEV_Select (3.3.7) to describe the function more accurately. Revised HDCP debugging (5.16) to also cover Source side.

1.2.Acronyms and abbreviations

API
: Application Programming Interface.

UAPI
: Unified Application Programming Interface.

DLL
: Dynamic Link Library.

CI
: Configuration Item.

GUI
: Graphical User Interface.

CTS
: Compliancy Testing System.

MSA
: Main Stream Attributes.

CRC
: Cyclic Redundancy Check.

IDE
: Integrated Development Environment

OS
: Operating System

EDID
: Extended Display Identification Data.

TSI
: Test System Interface

PDO
: Power Delivery Object.

2. COMPONENTS AND FEATURES

This section describes the features and components of the TSI API.

2.1.API DLL

The TSI API implementation is available as a 32-bit DLL and as 64-bit DLL. Naturally, the 64-bit version is only available for 64-bit operating systems.

On 32-bit operating systems (Windows XP, Windows Vista/7/8 x86 versions), the 32-bit TSI.DLL is typically located in the "C:\program files\common files\Unigraf\shared" folder.

On 64-bit operating systems (Windows XP x64, Windows Vista/7/8 x64 versions), the 32-bit TSI.DLL is typically in the "C:\program files (x86)\common files\Unigraf\shared" folder, while the 64-bit TSI.DLL is typically in "C:\program files\common files\Unigraf\shared" folder.

The provided loader will primarily load the TSI.DLL from one of the default locations, and if the DLL file is not found, it tries default OS search paths.

2.1.1.Features

- **Backwards compatibility guaranteed**: New versions of TSI are guaranteed to support all functionality of all previous TSI API versions. This means that application built using older TSI SDK can use TSI DLL from a later version SDK.

  Important: *Engineering builds may introduce features that will not be available and/or might be modified in a following actual release. Engineering builds are always clearly marked as such.*

  Important: *The backwards compatibility starts from the first official production release of TSI, which is version 1.2.*

- **Simplified usage**: The API offers a high level of intelligence built in so that client applications can use less API function calls and avoid using multiple threads for one device.

2.2.Using the TSI API

This API is intended to work as a high level API offering specific features to customers using the existing API interfaces of supported hardware.

Simplified usage

This API does not use handles to refer to devices. Instead, the API commits to using only the selected device. Using multiple devices with this API requires that each device is handled in it's own process.

Thread safety

TSI API is protected against harmful concurrent access.

Firmware Versions

TSI does not communicate directly with hardware, instead it uses lower level APIs to do so. As a result, TSI has no specific requirements of firmware versions.

Low-level API versions

TSI attempts to allow using several versions of low-level APIs per supported hardware, but may require specific versions. The required minimum versions are listed in the "release notes.txt" file that comes in the TSI release package.

Function return values

The primary rules for all functions that use a return value of type TSI_RESULT are the following:

- Negative value is always an error.
- Zero value indicates a generic success.
- Non-zero positive values indicate a generic success and additionally convey information specific to the function that returned the value.

The outcome of the rules is that

- Test for success can, and must be done as follows (or equivalent in your programming language):
```
if(Result >= TSI_SUCCESS) { /* Success */ }
```
- Test for failure can, and must be done as follows (or equivalent in your programming language):
```
if(Result < TSI_SUCCESS) { /* Failure */ }
```
- Please note that the following:
```
if(Result == TSI_SUCCESS) { /* Specific return check */ }
```
  does not test for success in general, instead it tests if a specific success condition has happened.

Important: *The return values defined per function do not override these rules, unless specifically indicated that the rules are violated.*

3. FUNCTIONS

This section describes the TSI client callable functions.

3.1.External functions

The functions in this chapter are delivered as source code in files “TSI.C” and “TSI.H”. Please include these files to your project to access TSI_LoadAPI and TSI_UnloadAPI functions.

3.1.1.TSI_LoadAPI

```
TSI_RESULT __stdcall TSI_LoadAPI
(
     _TCHAR *LibName
);
```

Synopsis

Load the API DLL, and resolve all the service functions. After this call the API client functions are available for use. If the DLL load is successful the function continues to call the API Init function on behalf of the client application. When calling the Init function, the LoadAPI function will use the client version constant in TSI_Types.h.

Important: *If the LibName parameter is NULL, the loader will look for TSI.DLL from the default install locations first, and from OS search paths second. If you wish to have TSI.DLL loaded from OS search paths only, you should give pointer to string containing “TSI.DLL” as parameter to this function.*

Parameters

LibName

A NULL terminated string containing the name (and optionally path) of TSI.DLL. This parameter can be NULL: If the parameter is NULL, then the default install location is attempted first, followed by system default search paths.

Result

If the function succeeds, the return value is a non-zero positive number indicating the number of times the Init function has been called.

Important: *If the DLL was loaded succesfully, but the Init function failed, the return value is zero. In this case it will be necessary to call the Init function again.*

If the function fails, the return value is a negative error code.

See Also

3.1.2 TSI_UnloadAPI, 3.2.1 TSI_Init

3.1.2.TSI_UnloadAPI

```
TSI_RESULT __stdcall TSI_UnloadAPI();
```

Synopsis

This function will first call the TSI_Clean() function of the API. If the TSI_Clean() call was last the Unload continues to free the TSI.DLL and release API loader resources.

Important: *If the TSI_Clean() call was not called for last time, the function will call TSI_Init() and return with an error status.*

Important: *Receiving an error result from this function indicates a resource management issue in the application.*

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.2.2 TSI_Clean, 3.2.1 TSI_Init, 3.1.1 TSI_LoadAPI

3.2.API Base level functions

The functions described in this section are part of TSI.DLL. Please notice that the exported names vary between x86 and x64 versions. Please use the provided source code to dynamically load the TSI.DLL. This document refers to the functions as made available by the provided loader.

3.2.1.TSI_Init

ClientVersion 1, and higher	*No license requirements*

```
TSI_RESULT __stdcall TSI_Init
(
     TSI_VERSION_ID ClientVersion
);
```

Synopsis

Initializes the API for use. Calls to Init are reference counted: Clean() must be called equal number of times for correct operation. If Init is not called, all service functions will fail with TSI_ERROR_NOT_INITIALIZED.

Parameters

ClientVersion

Indicates the TSI_Types.h file's version used to call the API functions. Always use the TSI_CURRENT_VERSION define as parameter when calling Init to ensure compatibility with later versions of the DLL.

Important: *The first call to Init will set the compatibility layer for the entire process. Following calls are required to use same ClientVersion value. If the ClientVersion is different between two calls, the later function-call will fail with TSI_ERROR_COMPATIBILITY_MISMATCH.*

Important: *If the wanted ClientVersion is NOT supported by the loaded DLL, then this function will fail with TSI_ERROR_NOT_COMPATIBLE.*

Result

If the function succeeds, the return value is a non-zero positive value indicating the API reference count after the function call.

If the function fails, the return value is a negative error code.

See Also

3.2.2 TSI_Clean

3.2.2.TSI_Clean

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_Clean();
```

Synopsis

Closes device and releases API resources if the reference count after the function call equals zero. Calls to Init are reference counted: Clean() must be called equal number of times for correct operation.

Result

If the function succeeds, the return value is a positive value (or zero) indicating the API reference count after the function call. If the return value is zero, the API functions are not available after this function call.

See Also

3.2.1 TSI_Init

3.3.Device management functions

3.3.1.TSI_DEV_GetParameterCount

ClientVersion 4, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_GetParameterCount
(
     TSI_DEVICE_ID DeviceID
);
```

Synopsis

Retrieves the number of parameters that can change a device's behavior when the device is selected. To read the list of parameters, please iterate through it by calling the TSI_DEV_GetParameterID function in a loop.

Important: *Use of this function is not needed for applications that only use known device types, or applications that always expect default behavior on device selection.*

Parameters

DeviceID

Identifies the device from which to read the parameter count. Valid DeviceID values range from zero to the number of devices returned by TSI_DEV_GetDeviceCount minus one.

Result

If the function succeeds, the return value is a positive value indicating the number of configuration items that can change the device's behavior during device selection.

If the return value is zero, there are no configuration items that could change the device's behavior when it is selected.

If the function fails, the return value is a negative error code.

See Also

3.3.2 TSI_DEV_GetParameterID, 3.3.7 TSI_DEV_Select, 3.3.5 TSI_DEV_GetDeviceCount

3.3.2.TSI_DEV_GetParameterID

ClientVersion 4, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_GetParameterID
(
     TSI_DEVICE_ID DeviceID,
     int ParameterIndex,
     TSI_CONFIG_ID *ParamID,
     unsigned int *ParamFlags
);
```

Synopsis

Retrieves information about a configuration item that may effect the behavior of a device while it's being selected. Some devices may have features that must be enabled or configured during device opening. This function exists to dynamically resolve any such configuration items per device.

Important: *Use of this function is not needed for applications that only use known device types, or applications that always expect default behavior on device selection.*

Parameters

DeviceID

Identifies the device from which to get the configuration ID value. Valid DeviceID values range from zero (0) to the number of devices returned by TSI_DEV_GetDeviceCount minus one.

ParameterIndex

Identifies the index of the parameter being queried. The first valid index is zero (0). Last valid index is value returned by successful call to TSI_DEV_GetParameterCount minus one.

ParamID

Pointer to TSI_CONFIG_ID type variable, which will receive a configuration item ID value.

ParamFlags

Pointer to an unsigned int type variable, which will receive configuration item related flag information.

Result

If the function succeeds, the return value is zero and information about a configuration item is placed to variables pointed by ParamID and ParamFlags. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code. The variable contents pointed by ParamID and ParamFlags remain unchanged.

See Also

3.3.1 TSI_DEV_GetParameterCount, 3.3.5 TSI_DEV_GetDeviceCount

3.3.3.TSI_DEV_SetSearchMask

ClientVersion 3, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_SetSearchMask
(
     TSI_DEVICE_CAPS RequiredCaps,
     TSI_DEVICE_CAPS UnallowedCaps
);
```

Synopsis

Limit number of devices found. Not all TSI applications support all features of TSI, and as such it may be necessary to limit the number of devices listed in the device list. For example, an application such as AV Test is not interested in seeing devices that don't have support for video capture.

Important: *Versions 1 and 2 clients will have RequiredCaps and UnallowedCaps pre-defined so that the operation remains identical.*

Important: *Version 3 and later clients will default to listing all devices present.*

Parameters

RequiredCaps

Flag bits that define which features are required for listed devices.

Important: *Do not issue capability bits that are undefined. If an undefined capability bit is set, the function will fail.*

UnallowedCaps

Flag bits that define which features must not be present on listed devices.

Important: *Do not issue capability bits that are undefined. If an undefined capability bit is set, the function will fail.*

Result

If the function succeeds, the return value is a positive value indicating the number of supported capture device attached to the local system. If there are no supported device present, the return value is zero.

If the function fails, the return value is a negative error code.

See Also

-

3.3.4.TSI_DEV_GetDeviceInfo

ClientVersion 3, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_GetDeviceInfo
(
     TSI_DEVICE_ID DeviceID,
     TSI_DEVICE_CAPS *Caps
);
```

Synopsis

Retrieves device capabilities bit-field of the indicated device.

Parameters

DeviceID

Identifies device from which to retrieve the capabilities flags. Valid DeviceID values range from zero to the number of devices returned by TSI_DEV_GetDeviceCount minus one.

Caps

Pointer to TSI_DEVICE_CAPS bit-field, which will receive the capabilities flags.

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.3.5 TSI_DEV_GetDeviceCount

3.3.5.TSI_DEV_GetDeviceCount

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_GetDeviceCount();
```

Synopsis

Enumerates supported Unigraf capture devices attached to the local system and returns the number of devices found. Please note, that in some cases a hardware device may not directly map into a single TSI device: Depending on the device itself, and it's low-level features, one hardware device may appear many times in TSI device enumeration. One example of such devices are the UCD family of devices. To form device ID used with other functions which require reference to a device use a number starting from zero (0) to the number returned by this function minus one.

Result

If the function succeeds, the return value is a positive value indicating the number of supported capture device attached to the local system. If there are no supported devices present, the return value is zero.

If the function fails, the return value is a negative error code.

See Also

3.3.6 TSI_DEV_GetDeviceName, 3.3.7 TSI_DEV_Select

3.3.6.TSI_DEV_GetDeviceName

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_GetDeviceName
(
     TSI_DEVICE_ID_ID DeviceID,
     char *DevNameString,
     unsigned int NameStringMaxLength
);
```

Synopsis

Retrieves a human readable name to identify the device associated to a DeviceID.

Parameters

DeviceID

ID Value identifying the wanted device. Valid DeviceID values range from zero to the number of devices returned by TSI_DEV_GetDeviceCount minus one.

DevNameString

Pointer to an array of characters that will receive the device's name. The string is guaranteed to be NULL terminated. If the buffer is not large enough to store the full name, the string is truncated.

NameStringMaxLength

Length of the DevNameString character array in chars. The recommended buffer size is 64 chars or more.

Result

If the function succeeds, the return value is the number of chars required by the full name of the device regardless of the NameStringMaxLength parameter. If the returned value is EQUAL or HIGHER than NameStringMaxLength, it means that the name was truncated.

If the function fails, the return value is a negative error code.

See Also

3.3.5 TSI_DEV_GetDeviceCount, 3.3.5 TSI_DEV_GetDeviceCount

3.3.7.TSI_DEV_Select

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_DEV_SELECT
(
     TSI_DEVICE_ID DeviceID
);
```

Synopsis

Selects a device to be controlled and apply the default state of the device. The default state of a device depends on the firmware on the device.

Important: *Any previously selected device is deselected. The deselected device state is not saved by TSI -- Re-selecting a device later will not restore the previous state of the device.*

Parameters

DeviceID

Identifies the device to open. Valid DeviceID values range from zero to the number of devices returned by TSI_DEV_GetDeviceCount minus one.

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.3.5 TSI_DEV_GetDeviceCount

3.3.8.TSI_DEV_Location

ClientVersion 11, and higher *No License requirements*

```
TSI_RESULT __stdcall TSI_DEV_Location
(
     const char * Location
);
```

Synopsis

Provides hints about the realm of device discovery used by TSI_DEV_GetDeviceCount.

Important: *This function is not needed if all devices are connected via USB.*

Parameters

Location

A string which is a combination of location hints. Each hint is represented as key-value pair. Hints are separated with ';' sign. Types of the hints are following:

- network=x.x.x.x

  This specifies sub-network mask for discovery broadcast, e.g. 192.168.1.255

Result

If the function succeeds, the return value is TSI_SUCCESS.

If the function fails, the return value is a negative error code.

3.4.Input management functions

3.4.1.TSI_VIN_GetParameterCount

ClientVersion 4, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_GetParameterCount
(
     TSI_INPUT_ID InputID
);
```

Synopsis

Retrieves the number of parameters that changes an input's behavior when the input is being selected. To read the list of parameters, iterate through it by calling TSI_VIN_GetParameterID in a loop.

Important: *Use of this function is not needed for applications that only use known device types, or applications that always expect default behavior on input selection.*

Parameters

InputID

Identifies the input from which to read the parameter count. Valid InputID values range from zero (0) to the number of inputs returned by TSI_VIN_GetInputCount function minus one.

Result

If the function succeeds, the return value is a positive value indicating the number of configuration items that changes the input's behavior during input selection.

If the return value is zero, there are no configuration items that could change the input's behavior when it is selected.

If the function fails, the return value is a negative error code.

See Also

3.4.2 TSI_VIN_GetParameterID, 3.4.3 TSI_VIN_GetInputCount

3.4.2.TSI_VIN_GetParameterID

ClientVersion 4, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_GetParameterID
(
     TSI_INPUT_ID InputID,
     int ParameterIndex,
     TSI_CONFIG_ID *ParamID,
     unsigned int *ParamFlags
);
```

Synopsis

Retrieves information about a configuration item that changes the behavior of an input while it is being selected. Some inputs may have features that must be enabled or configured during input activation. The function exists to dynamically resolve any such configuration items per input.

Important: Use of this function is not needed for applications that only use known device types, or applications that always expect default behavior on input selection.

Parameters

InputID

Identifies the input from which to get the configuration ID value. Valid InputID values range from zero (0) to the number of inputs returned by TSI_VIN_GetInputCount function minus one.

ParameterIndex

Identifies the index of the parameter being queried. The first valid index is zero (0). Last valid index is value returned by successful call to TSI_VIN_GetParameterCount minus one.

ParamID

Pointer to TSI_CONFIG_ID type variable, which will receive a configuration item ID value.

ParamFlags

Pointer to an unsigned int type variable, which will receive configuration item related flag information.

Result

If the function succeeds, the return value is zero and information about a configuration item is placed to variables pointed by ParamID and ParamFlags. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code. The variable contents pointed by ParamID and ParamFlags remain unchanged.

See Also

3.4.1 TSI_VIN_GetParameterCount, 3.4.3 TSI_VIN_GetInputCount, 3.4.5 TSI_VIN_Select

3.4.3.TSI_VIN_GetInputCount

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_GetInputCount();
```

Synopsis

Returns the number of inputs on the active capture device. Input ID Values range from zero (0) to the value returned by this function. If no capture device is active, this function will activate capture device with device ID of zero.

Result

If the function succeeds, the return value is a non-zero positive value indicating the number of audio/video interfaces present on the active device.

If the function fails, the return value is a negative error code.

See Also

3.4.4 TSI_VIN_GetInputName, 3.4.5 TSI_VIN_Select

3.4.4.TSI_VIN_GetInputName

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_GetInputName
(
     TSI_INPUT_ID InputID,
     char *InputNameString,
     unsigned int NameStringMaxLen
);
```

Synopsis

Retrieve a human readable name for the input associated with the given InputID.

Parameters

InputID

ID value of the input to be identified. Valid InputID values range from zero (0) to the number of inputs returned by TSI_VIN_GetInputCount function minus one.

InputNameString

Pointer to an array of characters that will receive a human readable name of the input. The resulting string is guaranteed to be NULL terminated. If the available string space is not long enough to contain the full name, the string is truncated.

NameStringMaxLen

Number of characters available in the InputNameString buffer. The recommended length for Input name is 64 characters, or more.

Result

If the function succeeds, the return value is the number of characters required by the full input name regardless of NameStringMaxLen parameter. If the returned value is EQUAL or HIGHER than NameStringMaxLen, it means that the name string was truncated.

If the function fails, the return value is zero.

See Also

3.4.3 TSI_VIN_GetInputCount, 3.4.3 TSI_VIN_GetInputCount

3.4.5.TSI_VIN_Select

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_Select
(
     TSI_INPUT_ID InputID
);
```

Synopsis

Selects an audio/video input to be activated. The intention of this function is to provide client application means to select one input out of many. By default, input number 0 is selected on the active device. If there is no need to change the active input this function call can be omitted completely.

Parameters

InputID

Identifies the input to be activated. Valid InputID values range from zero (0) to the number of inputs returned by TSI_VIN_GetInputCount function minus one.

Result

If the function succeeds, the return value is zero, or non-zero positive value. A non-zero value indicates data types which can be captured from the input. Please see table below for flag definitions:

Bit	Define	Description
0	TSI_FLAG_VIDEO	If set, video signal can be received from the selected input
1	TSI_FLAG_AUDIO	If set, audio signal can be received from the selected input
2	TSI_FLAG_CEC	If set, CEC data can be received from the selected input.

If the function fails, the return value is a negative error code.

See Also

3.4.3 TSI_VIN_GetInputCount, 3.4.6 TSI_VIN_Enable

3.4.6.TSI_VIN_Enable

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_Enable
(
     TSI_FLAGS Flags
);
```

Synopsis

Enable Audio/Video input. This function can succeed only if the input status at time of call is disabled.

The Input's enable state works as a gate-keeper for all data capturing: If the input is disabled, no data is being captured nor processed. When the input is enabled, all available data formats are captured.

If no device or audio/video input is selected before calling this function, this function will select the device with device ID of zero, and uses the default input. Please refer to 3.3.7 TSI_DEV_Select and 3.4.5 TSI_VIN_Select for further details.

Parameters

Flags

Obsolete. Any value passed here is ignored, and data formats to be captured are automatically detected and captured as they become available.

Result

If the function succeeds, the return values is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.3.7 TSI_DEV_Select, 3.4.5 TSI_VIN_Select, 3.4.7 TSI_VIN_Disable

3.4.7.TSI_VIN_Disable

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VIN_Disable();
```

Synopsis

Disables the audio/video input. This will stop all audio/video processing in the API.

Result

If the function succeeds, the audio/video input state is set to disabled and the functions return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.4.6 TSI_VIN_Enable

3.5.Output management functions

3.5.1.TSI_VOUT_GetParameterCount

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_GetParameterCount
(
     TSI_OUTPUT_ID OutputID
);
```

Synopsis

Retrieves the number of parameters that changes an output's behavior when output is being selected. To read the list of parameters, iterate through it by calling TSI_VOUT_GetParameterID in a loop.

Important: *Use of this function is not needed for applications that use known device types, or applications that always expect default behavior on output selections.*

Parameters

OutputID

Identifies the output from which to read the parameter count. Valid OutputID values ranges from zero (0) to the number of outputs returned by TSI_VOUT_GetOutputCount function minus one.

Result

If the function succeeds, the return value is a positive value indicating the number of configuration items that changes the output's behavior during output selection.

If the return value is zero, there are no configuration items that could change the output's behavior when it is selected.

If the function fails, the return value is a negative error code.

See Also

-

3.5.2.TSI_VOUT_GetParameterID

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_GetParameterID
(
     TSI_OUTPUT_ID OutputID,
     int ParameterIndex,
     TSI_CONFIG_ID *ParamID
     unsigned int *ParamFlags
);
```

Synopsis

Retrieves information about a configuration item that chnages the behavior of an output while it is being selected. Some outputs may have features that must be enabled or configured during output activation. The function exists to dynamically resolve any such configuration items per output.

Important: *Use of this function is not needed for applications that use known device types, or applications that always expect default behavior on output selections.*

Parameters

OutputID

Identifies the output from which to get the configuration ID value. Valid OutputID values range from zero (0) to the number of outputs returned TSI_VOUT_GetOutputCount function minus one.

ParameterIndex

Identifies the index of the parameter being queried. The first valid index is zero (0). Last valid index is value returned by successful call to TSI_VOUT_GetParameterCount minus one.

ParamID

Pointer to TSI_CONFIG_ID type variable, which will receive a configuration item ID value.

ParamFlags

Pointer to an unsigned int type variable, which will receive configuration item related flag information.

Result

If the function succeeds, the return value is zero and information about a configuration item is placed to variables pointed by ParamID and ParamFlags. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code. The variable contents pointed by ParamID and ParamFlags remain unchanged.

See Also

-

3.5.3.TSI_VOUT_GetOutputCount

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_GetOutputCount();
```

Synopsis

Returns the number of outputs on active device. Output ID values range from zero (0) to the value returned by this function (minus one). If no device is active, this function will activate the device that has device ID of zero.

Result

If the function succeeds, the return value is a non-zero positive value indicating the number of output interfaces present on the active device.

If the return value is zero, it means there are no output interfaces present on the active device.

If the function fails, the return value is a negative error code.

See Also

-

3.5.4.TSI_VOUT_GetOutputName

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_GetOutputName
(
     TSI_OUTPUT_ID OutputID
     char *OutputNameString,
     unsigned int NameStringMaxLength
);
```

Synopsis

Retrieves a human readable name for the output associated with the given OutputID.

Parameters

OutputID

ID value of the output to be identified. Valid OutputID values range from zero (0) to the number of outputs returned by TSI_VOUT_GetOutputCount function minus one.

OutputNameString

Pointer to an array of characters that will receive a human readable name of the output. The resulting string is guaranteed to be NULL terminated. If the available string space is not long enough to contain the full name, the string is truncated.

NameStringMaxLength

Number of characters available in the OutputNameString buffer. The recommended length of the Output name is 64 characters, or more.

Result

If the function succeeds, the return value is the number of characters required by the full output name regardless of NameStringMaxLen parameter. If the return value is EQUAL or HIGHER than NameStringMaxLen, it means that the name string was truncated.

If the function fails, the return value is zero.

See Also

-

3.5.5.TSI_VOUT_Select

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_Select
(
     TSI_OUTPUT_ID OutputID
);
```

Synopsis

Selects an audio/video output for configuration. The intention of this function is to provide client application means to select one output out of many. By default, TSI uses input interfaces. To use an output interface this function must be used to select the output interface to use.

Important: *If no device is selected before calling this function, the default device is selected automatically.*

Parameters

OutputID

Identifies the output to be configurable. Valid outputID values range from zero (0) to the number of inputs returned by TSI_VOUT_GetOutputCount functions minus one.

Result

If the function succeeds, the return value is zero.

If the function fails, the return value is a negative error code.

See Also

-

3.5.6.TSI_VOUT_Enable

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_Enable
(
     TSI_FLAGS Flags
);
```

Synopsis

Enable Audio/Video output. This function can succeed only if the output status at the time of call is disabled. Please note that the state of the output can be “enabled” immediately after call to TSI_VOUT_Select(…) function depending on the used hardware. A typical situation with such devices is that the output is always on and can't be disabled. With this type of hardware, TSI_VOUT_Enable(…) will always succeed, and TSI_VOUT_Disable(…) will always fail.

If no device and/or Audio/video output is selected before calling this function, the default device and output will be selected automatically.

Parameters

Flags

The flags field is currently ignored, and should be set to NULL (0) when calling.

Result

If the function succeeds, the return value is zero and video output will be enabled showing an output pattern using current settings. Please note that future versions of TSI may return non-zero positive values to indicate success.

If the function fails, the return value is a negative value indicating an error code.

See Also

-

3.5.7.TSI_VOUT_Disable

ClientVersion 11, and higher *<license:TBD>*

```
TSI_RESULT __stdcall TSI_VOUT_Disable();
```

Synopsis

Disables the currently selected Video/audio output. Please notice that some hardware devices may not support disabling the video output. In these cases, this function call will always fail.

Result

If the function succeeds, the return value is zero and the audio/video interface will not output data anymore. Please note that future versions of TSI may return non-zero positive values to indicate success.

If the function fails, the return value is negative indicating an error code.

See Also

-

3.6.Video Preview functions

3.6.1.TSI_VPREV_SetWindowHandle

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_VPREV_SetWindowHandle
(
     HWND Container
);
```

Synopsis

Creates video preview inside the given window. The preview will cover the entire client area of the window. The underlying video technology is selected automatically. The API will automatically show video preview in this window if video input is enabled. To disable video preview set preview window handle to NULL.

Parameters

Container

Handle to the window to contain the video preview. If a video is already being shown in another window that preview will stop and then start again in the new window. If this parameter is NULL, then any existing video preview is stopped.

Result

If the function succeeds, the video preview is enabled and the function returns zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

3.7.Audio Preview Functions

3.7.1.TSI_APREV_SetWindowHandle

ClientVersion 3, and later *No license requirements*

```
TSI_RESULT __stdcall TSI_APREV_SetWindowHandle
(
     HWND Container
);
```

Synopsis

Creates a graphical audio preview component. The graphics implementation in version 1.2 is a very basic spectral analysis display which will likely be improved with later versions. The preview will cover the entire client area of the window. The underlying video technology is selected automatically. The API will automatically show spectral analysis graph of the incoming audio if audio capture is enabled and audio is received. To disable the preview set preview window handle to NULL.

Parameters

Container

Handle to the window to contain the video preview. If a video is already being shown in another window that preview will stop and then start again in the new window. If this parameter is NULL, then any existing video preview is stopped.

Result

If the function succeeds, the audio preview is enabled and the function returns zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

-

3.7.2.TSI_APREV_GetDeviceCount

ClientVersion 1, and higher	*No license requirements*

```
TSI_RESULT __stdcall TSI_APREV_GetDeviceCount();
```

Synopsis

Returns the number of audio playback devices found from the local system.

Result

If the function succeeds, the return value is a positive value (or zero) indicating the number of available audio playback devices. If the return value is zero, there are no suitable audio playback devices present.

If the function fails, the return value is a negative error code.

See Also

3.7.3 TSI_APREV_GetDeviceName, 3.7.4 TSI_APREV_SelectDevice

3.7.3.TSI_APREV_GetDeviceName

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_APREV_GetDeviceName
(
     TSI_AUDIO_DEVICE_ID PlaybackDeviceID,
     char *PlaybackDeviceNameString,
     unsigned int NameStringMaxLen
);
```

Synopsis

Retrieves a human readable name associated with the given playback device ID.

Parameters

PlaybackDeviceID
Indicates the audio device to be identified.

PlaybackDeviceNameString
Pointer to an array of characters that will receive the name of the audio device. The string is guaranteed to be NULL terminated. If the string buffer is not large enough to contain the full name it will be truncated.

NameStringMaxLen
Number of chars available in the PlaybackDeviceNameString character array. Recommended size is 128 characters or more.

Result

If the function succeeds, the return value is a positive number indicating the number of characters required to hold the device's full name not counting the terminating NULL. If the return value is EQUAL or HIGHER than NameStringMaxLen parameter, it means that the string was truncated.

If the function fails, the return value is a negative error code.

See Also

3.7.2 TSI_APREV_GetDeviceCount

3.7.4.TSI_APREV_SelectDevice

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_APREV_SelectDevice
(
     TSI_AUDIO_DEVICE_ID DeviceID
);
```

Synopsis

Select an audio device for audio preview. The system default audio device will always have the DeviceID of zero (0). To disable audio preview, issue device ID negative one (-1).

Parameters

DeviceID

Identifies the device to use for audio preview.

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.7.2 TSI_APREV_GetDeviceCount

3.8.Test system related functions

3.8.1.TSI_TS_GetTestCount

ClientVersion 3, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_GetTestCount();
```

Synopsis

Retrieves the number of test available on the currently selected device. To get a list of tests, please iterate through the list by calling TSI_TS_GetTestInfo function in a loop.

Result

If the function succeeds, the return value is a positive value (or zero) indicating the number of tests available on the device. If the return value is zero, then there are no tests available on the device at the moment.

If the function fails, the return value is a negative error code.

See Also

3.8.2 TSI_TS_GetTestInfo

3.8.2.TSI_TS_GetTestInfo

ClientVersion 3, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_GetTestInfo
(
     int TestIndex,
     TSI_TEST_ID *ID,
     char *TestName,
     unsigned int TestNameMaxLength
);
```

Synopsis

Retrieves the test ID values and test names of the test available on the currently selected device.

Parameters

TestIndex

Test index value ranging from zero (0) to value returned by a call to TSI_TS_GetTestCount function minus one.

ID

Pointer to a TESTI_TEST_ID variable, which will receive the test ID value of test being identified. This ID value is used to start this test.

TestName

Pointer to a char string which will receive the name of the test being identified. If a string is returned, it is guaranteed to be NULL terminated.

This parameter can be NULL. If this parameter is NULL, then TestNameMaxLength parameter must be zero and no test name is returned.

TestNameMaxLength

Number of bytes available in the TestName array. The recommended minimum size is 128 bytes.

Result

If the function succeeds, the return value is the number of chars required by the full name of test regardless of the TestNameMaxLength parameter. If the returned value is EQUAL or HIGHER than TestNameMaxLength, it means that the name was truncated.

If the function fails, the return value is a negative error code.

See Also

3.8.1 TSI_TS_GetTestCount, 3.8.3 TSI_TS_GetTestParameterCount

3.8.3.TSI_TS_GetTestParameterCount

ClientVersion 3, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_GetTestParameterCount
(
     TSI_TEST_ID ID
);
```

Synopsis

Retrieves the number of parameters required for a particular test. To read the list of parameters, please iterate through the list by calling the TSI_TS_GetReqParameterID function in a loop.

Important: *Use of this function is not needed for application that only uses fully known device types.*

Parameters

ID

Identifies the test of which to get the parameter count. This test ID value is retrieved either by using the TSI_TS_GetTestInfo function, or by using a constant value defined in a devices specific documentation.

Result

If the function succeeds, the return value is a positive value (or zero) identifying the number of parameters required by the indicated test.

If the function fails, the return value is a negative error code.

See Also

3.8.4 TSI_TS_GetReqParameterID

3.8.4.TSI_TS_GetReqParameterID

ClientVersion 3, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_GetReqParameterID
(
     TSI_TEST_ID ID,
     int ParamIndex,
     TSI_CONFIG_ID *ParamID
     unsigned int *ParamFlags
);
```

Synopsis

Retrieves base information about a parameter that has a bearing on a test. This function can be used to create a generic GUI, which adapts and extends to tests available on a particular device.

Parameters

ID

Identifies the test to query.

ParamIndex

Index value ranging from zero (0) to value returnde by a call to TSI_TS_GetTestParameterCount function minus one.

ParamID

Pointer to a TSI_CONFIG_ID variable, which will receive the parameter ID value.

ParamFlags

Pointer to an unsigned int value, which will receive flags that provides additional information about the parameter. See table below for flag bits:

Bit	Define	Description
0	TSI_PID_MUST_SET	Indicates that the parameter must be set before attempting to run the test. If this bit is not set, then setting the parameter is optional (= it has a default value).
1	TSP_PID_VIRTUAL_GROUP	The parameter is a group identifier. Group identifiers can't be directly set. Instead each group refers to a set of other ID values.

Result

If the function succeeds, the return value is a positive value (or zero) indicating the size of the configuration item in bytes. If the return value is zero, it means that the configuration item's size is not constant, or depends on values contained in other configuration items.

If the function fails, the return value is a negative error code.

See Also

3.8.3 TSI_TS_GetTestParameterCount

3.8.5.TSI_TS_Clear

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_Clear();
```

Synopsis

Resets the test system to it's default settings. Please refer to section 4.2 Tests for details on the test specific defaults. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device.

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

4.2 Tests

3.8.6.TSI_TS_SetConfigItem

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_SetConfigItem
(
     TSI_CONFIG_ID ConfigItemID,
     void *ItemData,
     unsigned int ItemSize
);
```

Synopsis

Set a test-system configuration item. If the given configuration ID is valid, the function will copy the client provided data into API internal data storage for later use by the test system. Please refer to 5 Configuration Item definitions for details on the configuration items. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device.

Parameters

ConfigItemID
Identifies which configuration item to set.

ItemData
Pointer to the new data-set for the configuration item.

ItemSize
Size of the new data to be set for the configuration item.

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.8.7 TSI_TS_GetConfigItem, 3.8.9 TSI_TS_LoadConfig

3.8.7.TSI_TS_GetConfigItem

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_GetConfigItem
(
     TSI_CONFIG_ID ConfigItemID,
     void *ConfigItemData,
     unsigned int ItemMaxSize
);
```

Synopsis

Retrieve the current setting of a configuration item. If the ConfigItemID is valid and the provided data buffer is large enough, the function will copy the data to the provided buffer. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device.

Parameters

ConfigItemID
Identifies the configuration item to read.

ConfigItemData
Pointer to a buffer which will receive the configuration item data.

ItemMaxSize
Size of the ConfigItemData buffer in bytes.

Result

If the function succeeds, the return value is the number of bytes required to hold the configuration item data regardless of the ItemMaxSize parameters.

Important: *If the return value is HIGHER than ItemMaxSize parameter it means that no data was actually copied to the ConfigItemData buffer. In this case the contents of the ConfigItemData buffer are unchanged.*

If the function fails, the return value is a negative error code.

See Also

3.8.6 TSI_TS_SetConfigItem, 3.8.8 TSI_TS_SaveConfig

3.8.8.TSI_TS_SaveConfig

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_SaveConfig
(
     char *Filename
);
```

Synopsis

Saves the current test system status to a file for later use. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device.

Parameters

FileName

Pointer to a NULL terminated string containing the fully qualified filename of the target file. The API will overwrite any existing file.

Result

If the function succeeds, the return value is a positive, non-zero value indicating the number of bytes written to the target file.

If the function fails, the return value is a negative error code.

See Also

3.8.9 TSI_TS_LoadConfig

3.8.9.TSI_TS_LoadConfig

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_LoadConfig
(
     char *FileName
);
```

Synopsis

The API will first open the file. If the file was succesfully opened, the API continues to clear the test system configuration and loads new configuration from the given file. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device.

Important: *If the file is corrupted and/or there is problem reading the file the test system state after the function call will be the API default configuration.*

Parameters

FileName

Pointer to a NULL terminated string containing the fully qualified path of the configuration file.

Result

If the function succeeds, the return value is zero and the test system configuration was loaded from the given file. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code and the test system configuration status in undefined.

See Also

3.8.8 TSI_TS_SaveConfig

3.8.10.TSI_TS_RunTest

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_RunTest
(
     TSI_TEST_ID TestID
);
```

Synopsis

Run the given test. The function will block the calling application until the test is completed. Please refer to chapter 4.2 for details on available tests. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device. Also, an input must be selected before calling this function. If no input is selected before calling this function, this function will select the default input.

Parameters

TestID

Identifies the test to execute.

Result

If the function was completed without resource allocation issues, hardware problems or other OS errors, the return value is a positive value (or zero) indicating the test result. Please see the table below for test result values.

Value	Define	Description
0	TSI_TEST_PASS	The test is completed with “PASS” status.
1	TSI_TEST_FAIL	The test is completed with “FAIL” status.
2	TSI_TEST_NOT_STARTED	The test procedure failed before the test start conditions were met.

Important: *A previous version of this manual stated different values for the test results. This was due to mistake in the document. To avoid mistakes like these, please use the named constants whenever provided.*

If the function failed due to resource allocation issues; hardware problems or other OS errors, the return value is a negative error code.

See Also

4.2 Tests

3.8.11.TSI_TS_CaptureReference

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_CaptureReference
(
     int RequiredMatches,
     int ReferenceIndex
);
```

Synopsis

Captures a reference frame from the currently selected device and input. The function will block the calling thread until a reference frames is captured, or an error is encountered. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device. Also, an input must be selected before calling this function. If no input is selected before calling this function, this function will select the default input.

Important: *The RequiredMatches parameter can be used perform a sanity check in order to select a known good reference frame. If the RequiredMatches parameter is non-zero, the function will require a sequence of identical frames to be captured before accepting a frame as reference. Recommended setting for RequiredMatches is 2 for digital sources. Analog sources should always use 0 (=disable), since analog captures are practically never identical.*

Important: *If RequiredMatches is non-zero, the function will attempt to capture up to 60 frames in order to get a good reference frame. If no acceptable reference frame is captured within the period of 60 frames, the function fails.*

Important: *This function should be used only when the source device is supposed to be sending a static image.*

Parameters

RequiredMatches

Number of identical frames to be received before accepting the frame as reference. Allowed range is 0 – 10. Zero setting will not do any checking and will accept the first frame captured.

ReferenceIndex

Identifies which reference frame is to be set. This parameter must be zero.

Result

If the function succeeds, the return value is zero and the reference frame and related configuration items are set automatically. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, no reference frame was captured and any previous reference frame configuration remains unchanged.

See Also

3.8.6 TSI_TS_SetConfigItem, 3.8.9 TSI_TS_LoadConfig

3.8.12.TSI_TS_WaitInputSignal

ClientVersion 6, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_TS_WaitInputSignal
(
     unsigned int MaxWait
);
```

Synopsis

Blocks the calling thread until video or audio signal is detected on the selected device and input, or the timeout period has elapsed.

Parameters

MaxWait

Indicates maximum amount of time to wait for input signal to be detected, in milliseconds.

Results

If the function succeeds, and input signal is detected within the given timeout period, the return value is zero.

If the timeout expires before input signal is detected, the return value is TSI_ERROR_TIMEOUT.

If the function fails, the return value will be a negative error code.

3.9.Misc functions

3.9.1.TSI_MISC_SaveReference

ClientVersion 4, and later *No license requirements*

```
TSI_RESULT __stdcall TSI_MISC_SaveReference
(
     char *FileName,
     unsigned int RefIndex,
     TSI_FRAME_FORMAT_ID FormatID
);
```

Synopsis

Save the current reference frame into a file.

Parameters

FileName

Pointer to a NULL terminated char string identifying the target file. If the file already exists, it will be overwritten without prompt.

RefIndex

Reference Frame index. This parameter must be zero.

FormatID

Identifies the image file's format. The reference frame data will be converted to be suitable for the given format.

ID	Define	Description
1	TSI_FRAME_FORMAT_BMP	Identifies 24 effective bits per pixel BMP file type.
2	TSI_FRAME_FORMAT_PPM	Identiefies 24 or 48 effective bits per pixel PPM file type.

Result

If the function succeeds, the return value is the size of the resulting image file in bytes.

If the function fails, the return value is a negative error code.

See Also

3.9.2 TSI_MISC_LoadReference, 3.8.11 TSI_TS_CaptureReference, 3.8.8 TSI_TS_SaveConfig

3.9.2.TSI_MISC_LoadReference

ClientVersion 4, and later *No license requirements*

```
TSI_RESULT __stdcall TSI_MISC_LoadReference
(
     char *FileName,
     unsigned int RefIndex
);
```

Synopsis

Load a reference image from a file.

Parameters

FileName

Identifies the source file from which to read data.

RefIndex

Reference frame index. This parameter must be zero.

Result

If the function succeeds, the return value is zero and a reference frame is loaded. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

3.9.1 TSI_MISC_SaveReference, 3.8.9 TSI_TS_LoadConfig

3.9.3.TSI_MISC_SetOption

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_MISC_SetOption
(
     TSI_OPTION_ID OptionID,
     int OptionValue
);
```

Synopsis

Important: *Currently, there are no options defined, and the function is not used.*

Get and set option value. The function will set the new option value, and return the previous setting to the client application. A device must be selected before calling this function. If no device is selected before calling this function, this function will select the default device.

Important: *If the new option value is negative or out of range for the option in question, the option will remain unchanged and the function returns the current value of the option: Therefore, negative OptionValue parameter transforms the function to only read the current option value without changing it.*

Parameters

OptionID

Identifies the option to get and set.

OptionValue

Contains the new value for the option. Valid option setting values are positive numbers (or zero).

Result

If the function succeeds, the return value is a positive value (or zero) indicating the previous setting of the option.

If the function fails, the return value is a negative error code.

See Also

-

3.9.4.TSI_MISC_GetErrorDescription

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_MISC_GetErrorDescription
(
     TSI_RESULT ErrorCode,
     char *ErrorString,
     unsigned int StringMaxLen
);
```

Synopsis

Retrieves a human readable error message matching the given ErrorCode.

Parameters

ErrorCode

Indicates the error code to identify.

ErrorString

Pointer to an array of characters that will receive a human readable description of the error code. The resulting string may contain newlines. The string is guaranteed to be NULL terminated. If the string buffer is not large enough to store the complete description string, the string is truncated.

StringMaxLen

Maximum length of the string in characters. Recommended size for error message strings is 128 chars or more.

Result

If the function succeeds, the return value is the number of characters required for the complete error description string. If the return value is EQUAL or HIGHER than StringMaxLen parameter's value, it means that the string was truncated.

If the function fails, the return value is a negative error code.

See Also

5.26 Error codes

3.10.Status log functions

3.10.1.TSI_STLOG_GetMessageCount

ClientVersion 1, and higher	*No license requirements*

```
TSI_RESULT __stdcall TSI_STLOG_GetMessageCount();
```

Synopsis

Retrieves the number of queued status log message.

Result

If the function succeeds, the return value is a positive value indicating the number of queued status messages lines. If the return value is zero, then there are no messages queued.

If the function fails, the return value is a negative error code.

See Also

3.10.2 TSI_STLOG_Clear, 3.10.3 TSI_STLOG_GetMessageData

3.10.2.TSI_STLOG_Clear

ClientVersion 1, and higher	*No license requirements*

```
TSI_RESULT __stdcall TSI_STLOG_Clear();
```

Synopsis

Clear the status log buffer.

Result

If the function succeeds, the return value is zero. Please note that future versions may return non-zero positive value to indicate success.

If the function fails, the return value is a negative error code.

See Also

-

3.10.3.TSI_STLOG_GetMessageData

ClientVersion 1, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_STLOG_GetMessageData
(
     char *MsgBuffer,
     unsigned int MaxReadSize,
     unsigned int *OutputSize
);
```

Synopsis

Reads status log message buffer data. The function will output only a single, complete line of text without newline character(s) – insert newline character(s) as necessary if writing to a file and/or displaying on screen.

Parameters

MsgBuffer

Pointer to an array of characters which will receive a single line of status log text. The resulting string is guaranteed to be NULL terminated. The string will not contain newline character(s).

This parameter can be NULL. If this parameter is NULL, the MaxReadSize parameter must be set to zero. If this parameter is NULL, the function will return the MsgBuffer size required to get the next line of status log text.

MaxReadSize

Number of characters allocated for the MsgBuffer. Recommended minimum size for status log message line is 256 characters.

OutputSize

Pointer to an unsigned int variable that will receive the number of characters copied to the MsgBuffer string not counting the terminating NULL character.

If this parameter is NULL, then the number of copied characters is not returned.

Result

If the function succeeds, the return value is zero and the NULL terminated status log string is placed to MsgBuffer.

If the MsgBuffer was not large enough to contain the line of text plus the terminating NULL, the return value is a positive value indicating the required MsgBuffer size. The given MsgBuffer is erased.

If the function fails, the return value is a negative error code and the contents of the MsgBuffer are undefined.

See Also

3.10.1 TSI_STLOG_GetMessageCount

3.10.4.TSI_STLOG_WaitMessage

ClientVersion 5, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_STLOG_WaitMessage
(
     int MaxWait
);
```

Synopsis

Wait for at least one status log messages to become available for reading. If no messages arrive within given period, the function will return zero.

Parameters

MaxWait

Maximum time to wait for message(s) to arrive, in milliseconds.

Result

If the function succeeds, the return value is zero, or a positive number indicating the number of readable status log messages lines available for reading.

If the function fails, the return value is a negative error code.

See Also

3.10.1 TSI_STLOG_GetMessageCount and 3.10.3 TSI_STLOG_GetMessageData

3.11.Report generator functions

3.11.1.TSI_REP_BeginLogRecord

ClientVersion 5, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_REP_BeginLogRecord
(
     char *TargetFile,
     char *DUT_Information
);
```

Synopsis

Starts HTML report generator. The report generator will gather information about the current TE, software versions being used during the testing, and test configurations. The report will also record all test activity and test results. The end result is a HTML formatted report. The intention is that for each tested DUT, a separate report file is generated.

To correctly report a test sequence into a report follow these steps:

- Call this function, and make sure it succeeded.
- Run each of the tests planned for a specific DUT device.
- Call the TSI_REP_EndLogRecord function that will generate the final report file.

Parameters

TargetFile
A NULL terminated string containing the HTML report file name. The file name preferably includes full path to the file.

DUT_Information
A NULL terminated string containing information about the DUT. The information is embedded into the resulting report file.

Results

If the function succeeds, the return value is zero and any relevant information is being gathered into the report.

If the function fails, the return value is a negative error code and the report file is not being generated.

See Also

3.11.2 TSI_REP_EndLogRecord

3.11.2.TSI_REP_EndLogRecord

ClientVersion 5, and higher *No license requirements*

```
TSI_RESULT __stdcall TSI_REP_EndLogRecord();
```

Synopsis

Generate the HTML report file contents and release report generator resources. Call this function to complete a DUT test cycle and get the finalized report file about the tests.

Results

If the function succeeds, the return value is zero and the target HTML report file is finalized.

If the function fails, the return value is a negative error code and the target file's contents are undefined.

See Also

3.11.1 TSI_REP_BeginLogRecord

4. TYPES AND TEST DEFINITIONS

This chapter describes type definitions and test definitions.

4.1.Types

4.1.1.TSI_VERSION_ID

```
Typedef unsigned int TSI_VERSION_ID;
```

4.1.2.TSI_RESULT

```
Typedef int TSI_RESULT;
```

4.1.3.TSI_DEVICE_ID

```
Typedef unsigned int TSI_DEVICE_ID;
```

4.1.4.TSI_INPUT_ID

```
Typedef unsigned int TSI_INPUT_ID;
```

4.1.5.TSI_FLAGS

```
Typedef int TSI_FLAGS;
```

4.1.6.TSI_AUDIO_DEVICE_ID

```
Typedef unsigned int TSI_AUDIO_DEVICE_ID;
```

4.1.7.TSI_CONFIG_ID

```
Typedef unsigned int TSI_CONFIG_ID;
```

4.1.8.TSI_TEST_ID

```
Typedef unsigned int TSI_TEST_ID;
```

4.1.9.TSI_OPTION_ID

```
Typedef unsigned int TSI_OPTION_ID;
```

4.2.Tests

All tests, and their associated configurations and requirements are listed here.

4.2.1.Compare video frame sequence with a single reference

ClientVersion 1, and later *Basic license required*

```
#define TSI_TEST_VIDEO_PXL_TOLERANCE     2     // TEST ID
```

Synopsis

Compare a defined number of captured frames to a single reference frame. The test will capture the required number of consecutive frames into system RAM and then perform analysis between each frame and reference frame. Test is considered passed, if the number of failed frames does not exceed the programmed value.

Configuration items

TSI_PARAGRP_REFERENCE_1

The test always uses reference frame 1.

No default reference frame exists: the reference frame configuration must be set somehow before the test can be executed – This can be done by capturing a reference frame or by loading it from disk, or through other means: Please refer to 5.3 Reference frames for exact details in which configuration items to set.

TSI_TEST_LENGTH

Number of frames to capture and compare to reference. Default setting is 60.

Important: *Capturing a high-resolution frames into system RAM will consume a considerable amount of memory. If a memory error is encountered when trying to start this test, try reducing this value.*

TSI_LIM_FRAME_MISMATCHES

If the number of "bad" frames exceeds this number during the comparison stage, the test outcome will be "failed". Default setting is 0.

TSI_LIM_PIXEL_MISMATCHES

If the number of failed pixels (per frame) exceeds this number when comparing a single frame to reference, the compared frame is considered "bad". Default setting is 0.

TSI_PIXEL_TOLERANCE

This test allows deviation between captured and reference frame pixels. The deviation is calculated for each color-channel of each pixel. If the deviation on any channel exceeds this value, the pixel is considered "bad". Default setting is 0.

(Continued...)

(...Continued)

Additional/optional configuration items

In addition to just giving pass/fail result, the test can also be programmed to save failed frames on disk automatically. To enable auto-saving the following configuration items must be programmed:

TSI_MAX_AUTO_SAVE_FAILED
: Maximum number of failed frames saved into target folder per test. To enable auto-saving, this configuration item must be set to non-zero value. Default setting is 0.

  Important: *Auto-saving will never overwrite files, therefore it is necessary for the client application to watch the size of the directory so that the application will not fill out the entire hard-disk with failed frame data. Also, folders that have thousands of files will slow down saving new files.*

TSI_FAILED_FRAME_TARGET_FOLDER
: A character string that identifies the target folder into which the failed frames are saved into. The file-names are “Failed_#.bmp”, with the '#'-char replaced with a serial number for the frame. No default.

  Important: *If this feature is enabled, this setting should be assigned to a known location. If nothing is assigned, there may be unexpected behavior.*

If an application requires access to the failed frames, but does not need/want them to be automatically saved on disk, the test can also be programmed to provide access to the failed frames. The following configuration items are used to enable and access the failed frames:

TSI_MAX_EXPORT_FAILED
: Maximum number of failed frames exported from a single test run. To enable exports, this setting must be set to a non-zero value. Default setting is 0.

  Important: *The frames are available until any other test is started. Starting a test will release the memory allocations.*

TSI_EXPORTED
: **READ ONLY**. Number of frames currently exported.

TSI_EXPORT_ACCESS_INDEX
: **WRITE ONLY**. Which frame of the currently exported frames to access. Valid range is from zero (0) to value from *TSI_EXPORTED* minus one.

TSI_EXPORT_WIDTH
: **READ ONLY**. Width of the frame, in elements. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

TSI_EXPORT_HEIGHT
: **READ ONLY**. Height of the frame, in elements. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

TSI_EXPORT_ELEMENT_SIZE
: **READ ONLY**. The size of a single element, in bytes. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

(Continued...)

(...Continued)

TSI_EXPORT_PIXELS_PER_ELEMENT

READ ONLY. The width of a single element, in real pixels. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

TSI_EXPORT_LINES_PER_ELEMENT

READ ONLY. The height of a single element, in real pixels. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

TSI_EXPORT_COLOR_DEPTH

READ ONLY. Color depth as bits per color channel. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

TSI_EXPORT_PIXEL_FORMAT

READ ONLY. Identifies the color encoding used within the element. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

TSI_EXPORT_FRAME_DATA

READ ONLY. Contains the RAW frame data as described by the previous configuration items. The frame being accessed is indicated by the *TSI_EXPORT_ACCESS_INDEX* configuration item.

The application can also read exact failed pixel counts per frame. This data is gathered automatically, and is available until the next video test run is started.

TSI_R_VIDEO_TEST_RAW_RESULTS_DATA

READ ONLY. Provides access to RAW results generated by the video test. This configuration item has variable size, so please determine it's size before attempting to read the results into a buffer. The RAW results block is a list of unsigned integers, see the table below for description of the resulting list:

Index	Description
0	Total number of failed sub-pixels found on red color channel for frame 1
1	Total number of failed sub-pixels found on green color channel for frame 1
2	Total number of failed sub-pixels found on blue color channel for frame 1
3	Total failed pixels for frame 1
4	Total number of failed sub-pixels found on red color channel for frame 2
5	Total number of failed sub-pixels found on green color channel for frame 2
6	Total number of failed sub-pixels found on blue color channel for frame 2
7	Total failed pixels for frame 2
8 … n	… Additional data for following frames ...

(Continued...)

(...Continued)

Log messages

The test run is divided into three (3) stages.

Stage one is initialization, resource allocation and basic check of test parameters. The most important test parameters are logged. If no problems are detected, the test proceeds to stage two. Example log:

```
Starting video test (Test ID 2)
Stage 1: Test initialization. Test params:
  - Test length 60
  - Reference Width = 640
  - Reference Height = 480
  - Reference Element Width = 1
  - Reference Element Height = 1
  - Reference Format = 0
  - Reference Element byte size = 3
  - Reference Bit per channel = 8
```

Stage two is data gathering. During this stage the frames to be tested are simply captured into system RAM for the next stage. Example log:

```
Stage 1 Completed -- Entering stage 2: Data gathering
Stage 2 Completed -- Entering Stage 3: Compare and analysis
```

Stage three is analysis. Each frame is compared to the reference frame, and the frame analysis results are logged:

```
Stage 3, Frame 1 analysis results:
  - Failed pixels per sub channel: Red = 201914, Green = 201914, Blue = 201808
  - Total pixel errors = 269710, Highest deviation = 255
  - Mean deviation of pixels = 10.106
  - Total failed pixel errors exceed allowed pixel errors (0): BAD frame
  - The number of total frames exceed allowed bad frames (0).
Stage 3 completed -- Test failed
```

What the above actually means is this: There are 201914 pixels with errors on the red color channel, 201914 pixels with errors on green color channel and 201808 pixels with errors on the blue color channel.

269710 Pixels had error within the pixel (on at least one of three color channels). This value is at least equal to highest color-channel specific error count, and it can be as high as all color channel specific error counts combined. The Highest deviation tells the highest difference on any color channel between reference frame and compared frame.

Mean deviation is calculated by adding all deviations to together and dividing by the number of pixels in the frame.

The number of failed pixels exceed the number of allowed failures (which was zero), so the frame is considered “bad”.

The number of bad frames exceeds the number of allowed bad frames (which was also zero).

Thus the test outcome is “failed”.

See Also

5.4 Input video format

4.2.2.Validate audio signal frequency and glitch-free audio reproduction

ClientVersion 4, and later *Basic license required*

```
#define TSI_TEST_AUDIO_KILOHERTZ         3     // TEST ID
```

Synopsis

Perform frequency check on the digital audio content and verify the content to be glitch-free. This test assumes that a pure sine-wave audio signal content is being transmitted to the test equipment.

The test will first capture minimum of one second of audio content. The audio is then analyzed in two stages. First, the power spectrum is calculated and the highest peak must be within the defined window. The peak frequency check resolution is better than ±1 Hz. In second stage, the audio is checked to contain no random glitches, such as dropped or duplicated samples. This is achieved by examining how the RDV ("Relative Distortion Value") changes over time within the sampled audio.

The test is considered passed if the audio content spectrum has the highest power within the defined window, and the number of detected audio glitches does not exceed programmed value.

Configuration items

TSI_EXPECTED_SAMPLE_RATE
: The expected samples per second of the digital audio stream. If the actual sample rate does not match this value, the test can't be executed. Default setting is 44100.

TSI_EXPECTED_AUDIO_FREQUENCY
: The frequency that expected to have the highest power in the spectrum, in Hz. Default setting 1000.

TSI_AUDIO_FREQUENCY_TOLERANCE
: The allowed deviation between the measured highest-power and the expected highest power, in Hz. Default setting is 1.

TSI_AUDIO_GLITCH_DETECT_TRESHOLD
: This value defines the accepted RDV range by adding/subtracting it from the calculated base RDV when performing glitch detection. Lower values mean more sensitive to glitches – please note that setting this value too low will cause even perfectly good signal to fail the test. Valid range for this setting is 0 to 32767.0; The default setting is 5.0.

  Important: *FIXED POINT ENCODING. When setting this value parameter, the value being set must be multiplied by 65536 and set as a 32-bit integer. When reading the value, the received value must be divided by 65536 and shown as a floating point quantity.*

(Continued...)

(...Continued)

TSI_AUDIO_GLITCHES_ALLOWED
Number of detected glitches allowed per test.

Important: *Due to implementation specific characteristics, a single (but very audible) glitch is probably detected multiple times. The number of times a glitch is detected depends greatly on the severity of the glitch, and it's location respective to the sine waveform. Because of this, setting a non-zero but very low value may not make sense.*

Log messages

The audio test run is divided into four (4) stages. Stage one is test initialization, basic parameter validation and resource allocation.

```
Starting audio test (Test ID 3)
Stage 1: Test initialization. Test params:
  - Test length 65536 samples (1.49 seconds of audio)
  - Channel count = 2
  - Expected sample Rate = 44100
  - Reference Frequency = 1000 Hz
```

Important: *Test length (samples) is automatically selected to hold at least one second of audio for all channels.*

Important: *While it is possible to change the reference frequency, it is recommended to use the default frequency of 1kHz.*

Stage two is data gathering. During this stage, audio signal is captured to system memory.

Stage three is audio content frequency verification. The audio content must have the highest power peak within <Reference Frequency> ± <frequency tolerance> range:

```
Stage 2 Completed -- Entering Stage 3: Frequency check
  - Channel 0, Max power found at 999.95 Hz
  - Channel 1, Max power found at 999.95 Hz
```

Important: *The measurement accuracy is always better than ±0.5 Hz for pure sine signal.*

Stage four is audio glitch detection. The intent is to find frequently and randomly dropped, duplicated or otherwise damaged samples:

```
Stage 3 completed -- Entering Stage 4: Glitch detect
  - RDV value = 15.80
  - Glitch detected: Channel 0, Within sample range 3258 - 3386
    (RDV Value = 42.76)
  - Glitch detected: Channel 0, Within sample range 3302 - 3430
    (RDV Value = 42.19)
  - Glitch detected: Channel 0, Within sample range 7665 - 7793
    (RDV Value = 42.76)
```

The RDV (“Relative Distortion Value”) is calculated over the entire audio signal to provide a base-line RDV. The RDV value can vary greatly depending on how clean the audio signal is and can also be effected by the audio signal's amplitude, which is why the base line is calculated rather than programmed with strict limits. The ideal value for RDV is 1.00, but it is unreachable due to the limitations of digital audio and mathematical analysis.

Important: *The RDV is a unit-less value that comes out of a computational algorithm, and must be compared with other values that come out of the same algorithm with same expected input signal in order to draw conclusions – a single sample of RDV is useless.*

(Continued...)

(...Continued)

To detect a glitch, the calculated RDV must change more than allowed (<Base-line RDV ± <Audio Glitch detect threshold>). If a large enough change is detected, each detection is reported with information on which channel had it, range of samples within which it is located and the calculated RDV value for that range. A single glitch can be detected multiple times depending on the magnitude of the glitch.

See Also

5.4 Input video format

4.2.3.Electrical Test Set / Power test

```
ClientVersion 7, and higher
```

```
#define TSI_TEST_HDMI_EL_POWER_LINE            0x00020000
```

Synopsis

This test checks voltage level on the +5V power line of the DUT source. HDMI defines 4.7V … 5.3V as acceptable voltage range on the sink side connector. (Called “TP2” in the HDMI specification).

The test will measure the power line voltage with 0 mA load, and with 55 mA load as required in the CTS spceification (Test ID 7-11: +5V Power). The test will fail if voltage level on the power line is below or above the defined voltage range.

Configuration items

TSI_HDMI_RX_TIMEOUT
: Timeout for the electrical test in milliseconds. Default setting is 5000 ms.

TSI_HDMI_RX_POWER_LOW_LIMIT
: Lower voltage limit for the power line test, in millivolts. Default setting is 4700 mV.

TSI_HDMI_RX_POWER_HIGH_LIMIT
: Higher voltage limit for the power line test, in millivolts. Default setting is 5300 mV.

See Also

5.4 Input video format

4.2.4.Electrical Test Set / HPD test

ClientVersion 7, and higher

```
#define TSI_TEST_HDMI_EL_HPD_LINE              0x00020002
```

Synopsis

HPD line test checks cable/DUT source HPD line for short circuits to power or ground.

The test runs in two stages:

1. The HPD line is released to logical high state and voltage is measured from the HPD line. If the voltage level on the HPD line is outside the defined HPD ONE voltage window, the test considers that the HPD line is shorted to ground or power depending if the measured value is below the allowed window, or above it.

2. The HPD line is driven to logical low state and voltage is measured from the HPD line. If the voltage level on the HPD line is outside the defined HPD ZERO voltage window, the test considers that the HPD line is shorted to ground or power depending if the measured value is below the allowed window, or above it.

Configuration items

TSI_HDMI_RX_TIMEOUT
: Timeout for the electrical test in milliseconds. Default setting 5000 ms.

TSI_HDMI_RX_HPD_ZERO_LOW_LIMIT
: HPD Logical zero voltage window – lower allowed voltage boundary, in millivolts. Default setting is 0 mV.

TSI_HDMI_RX_HPD_ZERO_HIGH_LIMIT
: HPD Logical zero voltage window – higher allowed voltage boundary, in millivolts. Default setting is 400 mV.

TSI_HDMI_RX_HPD_ONE_LOW_LIMIT
: HPD Logical one voltage window – Lower allowed voltage boundary, in millivolts. Default setting is 2400 mV.

TSI_HDMI_RX_HPD_ONE_HIGH_LIMIT
: HPD Logical one voltage window – higher allowed voltage boundary, in millivolts. Default setting is 5300 mV.

See Also

5.4 Input video format

4.2.5.Electrical Test Set / DDC and CEC test

```
ClientVersion 7, and higher
```

```
#define TSI_TEST_HDMI_EL_DDC_CEC_LINES          0x00020003
```

Synopsis

DDC/CEC lines test measured voltage from the SCL, SDA and CEC lines when not being driven low.

If the DDC or CEC line voltage levels are outside the defined ranges, the test fails.

Configuration items

TSI_HDMI_RX_TIMEOUT
: Timeout for the electrical test in milliseconds. Default setting 5000 ms.

TSI_HDMI_RX_DDC_LOW_LIMIT
: DDC line voltage window – Lower allowed voltage boundary, in millivolts. Default is 4500 mV.

TSI_HDMI_RX_DDC_HIGH_LIMIT
: DDC line voltage window – Higher allowed voltage boundary, in millivolts. Default is 5500 mV.

TSI_HDMI_RX_CEC_ZERO_LOW_LIMIT
: CEC logical zero voltage window – Lower allowed voltage boundary, in millivolts. Default is 0 mV.

TSI_HDMI_RX_CEC_ZERO_HIGH_LIMIT
: CEC logical zero voltage window – Higher allowed voltage boundary, in millivolts. Default is 600 mV.

TSI_HDMI_RX_CEC_ONE_LOW_LIMIT
: CEC logical one voltage window – Lower allowed voltage boundary, in millivolts. Default is 2500 mV.

TSI_HDMI_RX_CEC_ONE_HIGH_LIMIT
: CEC logical one voltage window – Higher allowed voltage boundary, in millicolts. Default is 3600 mV.

See Also

5.4 Input video format

4.2.6.Electrical Test Set / TMDS test

```
ClientVersion 7, and higher
```

```
#define TSI_TEST_HDMI_EL_TMDS_LINES            0x00020001
```

Synopsis

This test measures average voltage levels on TMDS signal lines.

TMDS will guarantee DC balanced signalling. Sink will pull up a line to 3.3V AVcc voltage and source will pull down the line. On active HDMI line average voltage level is expected to fall below AVcc for the value of the voltage swing divide by two and defaults to range 2.6V… 3.1V. Values out of the set range mean a problem with TMDS lines, such as short circuit or broken output driver. An open circuit measures 3.3V AVcc. DVI TMDS test has the same functionality as HDMI, but voltage range defaults to 3.0V…3.1V. TMDS differential pair positive and negative lines are measured separately.

Important: *Acceptable range should be set by the user depending on the source DUT and cable setup.*

Configuration items

TSI_HDMI_RX_TIMEOUT
: Timeout for the electrical test in milliseconds. Default setting 5000 ms.

TSI_HDMI_RX_LINK_LOW_LIMIT
: HDMI TMDS link line voltage window – Lower allowed voltage boundary, in millivolts. Default is 2900 mV.

TSI_HDMI_RX_LINK_HIGH_LIMIT
: HDMI TMDS link line voltage window – Higher allowed voltage boundary, in millivolts. Default is 3100 mV.

See Also

5.4 Input video format

4.2.7.CEC functional Test set / CEC functional test

ClientVersion 7, and higher

```
#define TSI_TEST_HDMI_CEC                          0x00050000
```

Synopsis

The test verifies that source DUT correctly handles HPD event, reads EDID and broadcasts the CEC *"Report physcal address"* message.

First, the TE allocates the given physical address and issues a HPD pulse simulating cable detach/attach. The it waits for DUT to broadcast the CEC "Report physical address" message. The test is considered passed if the TE finds that the DUT broadcasts with the physical address allocated by the TE for the test.

Important: *As a side effect, the CEC will also verify functionality of HPD and EDID reading if the test passes.*

Configuration items

TSI_HDMI_RX_CEC_TIMEOUT
: HDMI CEC test timeout in milliseconds. Default is 5000 ms.

TSI_HDMI_RX_CEC_LOCAL_PHY_ADDR
: The local CEC physical address used for generating downstream physical addresses. Please, see HDMI CEC specification for details. Default is 4.0.0.0 (0x4000)

See Also

5.4 Input video format

4.2.8.Electical Test Set / Main Link test

ClientVersion 7, and higher

```
#define TSI_TEST_DP_EL_MAIN_LINK                0x00010001
```

Synopsis

The test measures power of DP input signal and checks that the result lies within an allowed voltage window.

The measured value follows the input signal's amplitude and is large for large input swing. Measured power value depends on signal waveform and it varies because of e.g. used cable. Due to this, the measurement only provides a relative value which does not represent any absolute value, e.g. input signal voltage levels.

“No signal” -level is initially set to 2.3V. Note that even a disconnected line will give a relatively high value. Good signal levels are expected to be within range 2.6V…4.0V. The allowed voltage window should be set separately for each device model after testing of several units.

Measured values are expected to be close to each other within a differential pair. Also, all main link differential pair measurements should produce a value close to each other if link training result is the same for all pairs.

Measurement results are given in volts but this is only the voltage level of power measurement circuitry output and does not relate to input signal. Main link differential pair positive and negative lines are measured separately.

Configuration items

TSI_DP_RX_TEST_TIMEOUT
: Timeout for the electrical test in milliseconds. Default value is 5000 ms.

TSI_DP_RX_LINKS_LOW_VOLTAGE
: DP link power window, lower voltage boundary in millivolts. Default is 2600 mV.

TSI_DP_RX_LINKS_HI_VOLTAGE
: DP link power window, higher voltage boundary in millivolts. Default is 4000 mV.

TSI_DP_RX_MAX_DUT_LANE_COUNT
: Max. number of lanes supported by the DUT. Default is 4.

TSI_DP_RX_MAX_DUT_LINK_RATE
: Max. lane frequency as multiple of 0.27Gbps. Default is 20 (5.4Gbps).

See Also

5.4 Input video format

4.2.9.Electrical Test Set / AUX test

```
ClientVersion 7, and higher
```

```
#define TSI_TEST_DP_EL_AUX_LINE                0x00010002
```

Synopsis

Verifies voltage levels on AUX lines, and AUX connectivity to DUT.

The test runs in two stages:

1. The idle AUX voltage level is measured. It is expected that voltages match to values defined by resistor dividers set by connected DisplayPort sink and source devices (see AUX CH Differential Pair in the DP specification).
2. The TE creates a short HPD pulse to have the DUT to generate an AUX request. The DUT is expected to read 0x200 – 0x205 DPCD registers. Test captures sync sequence of AUX transaction and checks the unit interval timings.

Configuration items

TSI_DP_RX_TEST_TIMEOUT
: Timeout for the electrical test in milliseconds. Default value is 5000 ms.

TSI_DP_RX_AUX_P_IDLE_LOW_VOLTAGE
: DP AUX P-Line idle state voltage window, lower boundary voltage in millivolts. Default 20 mV.

TSI_DP_RX_AUX_P_IDLE_HI_VOLTAGE
: DP AUX P-Line idle state voltage window, higher boundary voltage in millivolts. Default 500 mV.

TSI_DP_RX_AUX_N_IDLE_LOW_VOLTAGE
: DP AUX N-Line idle state voltage window, lower boundary voltage in millivolts. Default 2600 mV.

TSI_DP_RX_AUX_N_IDLE_HI_VOLTAGE
: DP AUX N-Line idle state voltage window, higher boundary voltage in millivolts. Default 3600 mV.

TSI_DP_RX_AUX_P_TRIG_VOLTAGE
: DP AUX P-Line trigger voltage level in millivolts. Default is 150 mV.

TSI_DP_RX_AUX_N_TRIG_VOLTAGE
: DP AUX N-Line trigger voltage level in millivolts. Default is 200 mV.

TSI_DP_RX_AUX_SIGNAL_CAPT_TIMEOUT
: Period of time TE waits for AUX transactions starting from end of HPD pulse. Default is 200 ms.

TSI_DP_RX_AUX_SIGNAL_CAPT_TRIES
: If AUX transactions are not detected within the timeout period TW will try again. Default is 5.

See Also

5.4 Input video format

4.2.10.Electrical Test Set / HPD test

```
ClientVersion 7, and later
```

```
#define TSI_TEST_DP_EL_HPD_LINE                0x00010000
```

Synopsis

HPD line test checks cable/DUT source HPD line for short circuits to power or ground.

The test runs in two stages:

1. The HPD line is released to logical high state and voltage is measured from the HPD line. If the voltage level on the HPD line is outside the defined HPD ONE voltage window, the test considers that the HPD line is shorted to ground or power depending if the measured value is below the allowed window, or above it.

2. The HPD line is driven to logical low state and voltage is measured from the HPD line. If the voltage level on the HPD line is outside the defined HPD ZERO voltage window, the test considers that the HPD line is shorted to ground or power depending if the measured value is below the allowed window, or above it.

Configuration items

TSI_DP_RX_TEST_TIMEOUT
: Timeout for the electrical test in milliseconds. Default value is 5000 ms.

TSI_DP_RX_HPD_ZERO_LOW_VOLTAGE
: HPD line logical zero voltage window, lower boundary in millivolts. Default is -100 mV.

TSI_DP_RX_HPD_ZERO_HI_VOLTAGE
: HPD line logical zero voltage window, higher boundary in millivolts. Default is 800 mV.

TSI_DP_RX_HPD_ONE_LOW_VOLTAGE
: HPD line logical one voltage window, lower boundary in millivolts. Default is 800 mV.

TSI_DP_RX_HPD_ONE_HI_VOLTAGE
: HPD line logical one voltage window, higher boundary in millivolts. Default is 5500 mV.

See Also

5.4 Input video format

4.2.11.CRC based Video Test set / CRC based single frame reference video test

ClientVersion 8, and higher

```
#define TSI_TEST_DP_VIDEO_CRC_SINGLE_REF       0x00060000
#define TSI_TEST_HD_VIDEO_CRC_SINGLE_REF       0x000b0000
(#define TSI_TEST_VIDEO_CRC_SINGLE_REF          0x00060000)
```

Synopsis

The test checks input frames to match with provided resolution and color depth, and the contents of the frames are checked to be identical with the reference through comparing CRC values of the reference frame and the input frame. This test uses only the first reference CRC value set.

Important: *This test has separate implementation for DP and HDMI inputs, while the test parameters and operation are identical.*

Configuration items

TSI_CRC_TIMEOUT
: Indicates test timeout in milliseconds. Default value is 1000ms

TSI_CRC_FRAMES_TO_TEST
: Indicates number of frames to be tested. Default value is 20.

  Important: *Make sure the TSI_CRC_TIMEOUT is long enough to allow this many frames to be received.*

TSI_CRC_LIM_FRAME_MISMATCHES
: Number of frames that are allowed to have mismatching CRC with the reference frame(s). Default value is 0.

TSI_CRC_REF_WIDTH
: Width of the reference frame used, in pixels. Default value is 1920.

TSI_CRC_REF_HEIGHT
: Height of the reference frame used, in pixels. Default value is 1080.

TSI_CRC_REF_COLORDEPTH
: Bit depth of the reference frame used, in bits per pixel. Default value is 24.

TSI_CRC_REFERENCE_CRC_VALUES
: A block of CRC reference value sets. Each CRC reference value set is a block of 3 16-bit values, with Red/Cr CRC at the lowest address, and Blue/Cb CRC at the highest address.

See Also

5.4 Input video format

4.2.12.CRC based Video Test set / CRC based single frame video stability test

ClientVersion 8, and higher

```
#define TSI_TEST_DP_CRC_VIDEO_STABILITY        0x00060001
#define TSI_TEST_HD_CRC_VIDEO_STABILITY        0x000b0001
(#define TSI_TEST_CRC_VIDEO_STABILITY           0x00060001)
```

Synopsis

A simple test that is used to verify if a video stream is stable without providing a CRC value set as reference. If the CRC values remain identical for the duration of the test, the test is passed.

Important: *This test has separate implementation for DP and HDMI inputs, while the test parameters and operation are identical.*

Configuration items

TSI_CRC_TIMEOUT
: Indicates test timeout in milliseconds. Default value is 1000ms

TSI_CRC_FRAMES_TO_TEST
: Indicates number of frames to be tested. Default value is 20.

  Important: *If the value is zero, the test will run until time out, but still passes if no mismatches were detected.*

  Important: *Make sure the TSI_CRC_TIMEOUT is long enough to allow this many frames to be received.*

See Also

5.4 Input video format

4.2.13.CRC based Video Test set / CRC based sequence of frames reference video test

ClientVersion 8, and higher

```
#define TSI_TEST_DP_CRC_VIDEO_SEQUENCE          0x00060002
#define TSI_TEST_HD_CRC_VIDEO_SEQUENCE          0x000b0002
(#define TSI_TEST_CRC_VIDEO_SEQUENCE             0x00060002)
```

Synopsis

The Source DUT should be sending a repeating video sequence to the TE, without the sequence containing no identical frames within the loop.

The test will first synchronize with the provided CRC sequence by finding the by finding video frame with the CRC matching CRC of the first frame in the reference sequence. Once the match is detected, the test proceeds comparing CRC values of every frame to the CRC values in the reference sequence. If the test fails to synchronize to the input video stream the test will fail with timeout. Test will fail immediately if a CRC mismatch is detected. DUT will PASS the test if TE finds input video resolution and color format matching to reference parameters, found reference frame sequence and no mismatches CRC in frame sequence.

Important: *This test has separate implementation for DP and HDMI inputs, while the test parameters and operation are identical.*

Configuration items

TSI_CRC_TIMEOUT
: Indicates test timeout in milliseconds. Default value is 1000ms

TSI_CRC_FRAMES_TO_TEST
: Indicates number of frames to be tested. Default value is 20.

  Important: *Make sure the TSI_CRC_TIMEOUT is long enough to allow this many frames to be received.*

TSI_CRC_REF_WIDTH
: Width of the reference frame used, in pixels. Default value is 1920.

TSI_CRC_REF_HEIGHT
: Height of the reference frame used, in pixels. Default value is 1080.

TSI_CRC_REF_COLORDEPTH
: Bit depth of the reference frame used, in bits per pixel. Default value is 24.

TSI_CRC_REFERENCE_CRC_VALUES
: A block of CRC reference value sets. Each CRC reference value set is a block of 3 16-bit values, with Red/Cr CRC at the lowest address, and Blue/Cb CRC at the highest address.

See Also

5.4 Input video format

4.2.14.CRC Based Video Test Set / CRC based continuous sequence of reference frames

ClientVersion 8, and higher

```
#define TSI_TEST_DP_CRC_CONT_VIDEO_SEQUENCE    0x00060003
#define TSI_TEST_HD_CRC_CONT_VIDEO_SEQUENCE    0x000b0003
(#define TSI_TEST_CRC_CONT_VIDEO_SEQUENCE      0x00060003)
```

Synopsis

The Source DUT should be sending a repeating video sequence to the TE, without the sequence containing no identical frames within the loop.

The test will first synchronize with the provided CRC sequence by finding the by finding video frame with the CRC matching CRC of the first frame in the reference sequence. Once the match is detected, the test proceeds comparing CRC values of every frame to the CRC values in the reference sequence. When the entire reference sequence is compared, the test expects to find the same sequence repeated without any intermediate frames between the last defined frame and the first one. The reference sequence is tested multiple times, as defined by the TSI_CRC_MOTION_TEST_ITERATIONS configuration item. If the test fails to synchronize to the input video stream the test will fail with timeout. Test will fail immediately if a CRC mismatch is detected. DUT will PASS the test if TE finds input video resolution and color format matching to reference parameters, found reference frame sequence and no mismatches CRC in frame sequence.

Important: *This test has separate implementation for DP and HDMI inputs, while the test parameters and operation are identical.*

(Continued…)

(...Continued)

Configuration items

TSI_CRC_TIMEOUT
Indicates test timeout in milliseconds. Default value is 1000ms

TSI_CRC_FRAMES_TO_TEST
Indicates number of frames to be tested. Default value is 20.

Important: *Make sure the TSI_CRC_TIMEOUT is long enough to allow this many frames to be received.*

TSI_CRC_REF_WIDTH
Width of the reference frame used, in pixels. Default value is 1920.

TSI_CRC_REF_HEIGHT
Height of the reference frame used, in pixels. Default value is 1080.

TSI_CRC_REF_COLORDEPTH
Bit depth of the reference frame used, in bits per pixel. Default value is 24.

TSI_CRC_REFERENCE_CRC_VALUES
A block of CRC reference value sets. Each CRC reference value set is a block of 3 16-bit values, with Red/Cr CRC at the lowest address, and Blue/Cb CRC at the highest address.

TSI_CRC_MOTION_TEST_ITERATIONS
Defines the number of times the defined reference sequence must be found in order to pass the test.

TSI_CRC_COLOR_FORMAT
Defines which color format the input stream is expected to be.

4.2.15.Link Test set / Link Training at All Supported Lane Counts and Link Rates

ClientVersion 8, and higher

```
#define TSI_TEST_DP_SIMPLE_LINK                 0x00070000
```

Synopsis

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.

TSI_DP_LTT_TIMEOUT
: Defines time-out for each link training, in milliseconds. Default is 5000 ms.

TSI_DP_LTT_MAX_LANE_COUNT
: Defines maximum lane count to be tested. Valid settings are 1, 2 and 4. Default setting is 4.

TSI_DP_LTT_MAX_RATE
: Defines maximum link rate to be tested as multiple of 0.27 Gbps. Valid settings are 6, 10 and 20. Default setting is 20.

TSI_DP_LTT_HPD_PULSE_DURATION
: Defines the HPD pulse length used to start each link training, in milliseconds. Default is 1000 ms.

TSI_DP_LTT_LT_START_TIMEOUT
: Defines the maximum timeout allowed for the link training to start after HPD pulse, in milliseconds. Default is 5000 ms.

TSI_DP_LTT_TEST_LOOP_DELAY
: Defines idle-period at end of each link training before starting the next link training, in milliseconds. Default is 3000 ms.

4.2.16.USBC Electrical Test Set / Up Face port CC and Vconn test

ClientVersion 10, and higher *Electrical test license required*

```
#define TSI_TEST_USBC_CC_VCON                    0x000c0000
```

Synopsis

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

At test start, the TE will temporarily disconnect CC lines to simulate a re-plug event. After the re-plug event, Power source capability resistor R_a is connected CC2 line, and R_d resistor is connected to CC1 line. DUT is expected to have resistor R_p, or a current source applied to both CC1 and CC2 lines. The impedance of the DUT's R_p resistor, or current source must be adjusted so that the voltage drop on R_d resistor on the TE is within one of the voltage ranges defined by *TSI_USBC_EL_CC_LOW_VOLTAGE_** and *TSI_USBC_EL_CC_HI_VOLTAGE_**. (By default the ranges are 261mV → 588mV, 675mV → 1189mV and 1238mV → 2181mV). TE will measure the voltage drop on R_p.

The TE will measure the voltage present on CC2 after DUT has started to provide Vconn voltage on CC2 for an active cable.

Once the Vconn is measured, the TE will enable the cable-flip feature and repeat the steps as above.

In order to pass the test, the measured values from CC1, CC2 and Vconn must be within a respective range. The passable ranges are defined by the following configuration items:

- CC lines voltage range 1: *TSI_USBC_EL_CC_LOW_VOLTAGE_1* and *TSI_USBC_EL_CC_HI_VOLTAGE_1*.
- CC lines voltage range 2: *TSI_USBC_EL_CC_LOW_VOLTAGE_2* and *TSI_USBC_EL_CC_HI_VOLTAGE_2*.
- CC lines voltage range 3: *TSI_USBC_EL_CC_LOW_VOLTAGE_3* and *TSI_USBC_EL_CC_HI_VOLTAGE_3*.
- Vconn voltag range: *TSI_USBC_EL_VCON_LOW_VOLTAGE* and *TSI_USBC_EL_VCON_HI_VOLTAGE.*

These configuration items should be programmed with values averaged from several fully operational DUT's.

Important: *In order to run this test with UCD-340, a special cable provided by Unigraf must be used.*

TSI_USBC_EL_TIMEOUT
: Test timeout, in milliseconds. Default setting is 5000ms.

TSI_USBC_EL_REPLUG_TIME
: Simulated cable re-plug disconnected state time, in milliseconds. Default setting is 1500ms.

TSI_USBC_EL_DUT_ATTACH_TIMEOUT
: Maximum time allowed for the DUT to respond to cable plug, in milliseconds. Default setting is 10000ms.

(Continued…)

(...Continued)

TSI_USBC_EL_CC_LOW_VOLTAGE_1
CC Line voltage range 1, lower limit, in millivolts. This range applies when sink current is 0.5A or 0.9A. Default setting is 261mV

TSI_USBC_EL_CC_HI_VOLTAGE_1
CC Line voltage range 1, higher limit, in millivolts. This range applies when sink current is 0.5A or 0.9A. Default setting is 588mV.

TSI_USBC_EL_CC_LOW_VOLTAGE_2
CC line voltage range 2, lower limit, in millivolts. This range applies when sink current is 1.5A. Default setting is 675mV.

TSI_USBC_EL_CC_HI_VOLTAGE_2
CC line voltage range 2, higher limit, in millivolts. This range applies when sink current is 1.5A. Default setting is 1189mV.

TSI_USBC_EL_CC_LOW_VOLTAGE_3
CC line voltage range 3, lower limit, in millivolts. This range applies when sink current is 3.0A. Default setting is 1238mV.

TSI_USBC_EL_CC_HI_VOLTAGE_3
CC line voltage range 3, higher limit, in millivolts. This range applies when sink current is 3.0A. Default setting is 2181mV.

TSI_USBC_EL_VCON_LOW_VOLTAGE
Vconn line voltage range, lower limit, in millivolts. Default setting is 4750mV.

TSI_USBC_EL_VCON_HI_VOLTAGE
Vconn line voltage range, higher limit, in millivolts. Default setting is 5500mV.

RAW Test results data

*TSI_R_TDATA_USBC_EL_VCC**
CC1 and CC2 line voltages in millivolts.

*TSI_R_TDATA_USBC_EL_VCONN**
VCONN1 and VCONN2 line voltages in millivolts.

4.2.17.USBC Electrical Test Set / AUX (SBU) lines test

ClientVersion 10, and higher *Electrical test license required*

```
#define TSI_TEST_USBC_SBU_DP_AUX                0x000c0001
```

Synopsis

This test verifies operation of SBU lines against short-circuit and open-circuit failures, and that directly related hardware is working properly. During the test the TE will operate as Type-C UFP device. In order to run this test, the DUT must support DisplayPort alternate mode.

At test start, the TE will temporarily disconnect CC lines to simulate a re-plug event. After the re-plug event, the TE waits for the DUT to enter DP Alternate mode. Once the DUT has entered the DP alternate mode, TE will measure voltage levels on AUX+ (SBU1) line, and AUX- (SBU2) line. Please notice that if the TE is acting as DP Sink, it will de-assert the HPD signal to keep AUX bus at IDLE state during the voltage measurements.

Once the voltages are measured, the TE will enable the cable-flip feature and repeat the above steps.

In order to pass the test, all the measured AUX- and AUX+ voltages must be within the respective ranges (By default AUX- range is 100mV → 600mV, and AUX+ range is 2500mV → 3000mV). The ranges can be configured with the following configuration items:

- AUX+ voltage range: *TSI_USBC_AUX_P_IDLE_LOW_VOLTAGE* and *TSI_USBC_AUX_P_IDLE_HI_VOLTAGE*.
- AUX- voltage range: *TSI_USBC_AUX_N_IDLE_LOW_VOLTAGE* and *TSI_USBC_AUX_N_IDLE_HI_VOLTAGE*.

These configuration items should be programmed with values averaged from several fully operational DUT's.

Important: *In order to run this test with UCD-340, a special cable provided by Unigraf must be used.*

TSI_USBC_EL_TIMEOUT
: Test timeout, in milliseconds. Default setting is 5000ms.

TSI_USBC_EL_DUT_CAPS
: Defines DUT capabilities. See 5.20.2 TSI_USBC_EL_DUT_CAPS for details. Default setting is 0.

TSI_USBC_EL_REPLUG_TIME
: Simulated cable re-plug disconnected state time, in milliseconds. Default setting is 1500ms.

TSI_USBC_EL_DUT_ATTACH_TIMEOUT
: Maximum time allowed for the DUT to respond to cable plug, in milliseconds. Default setting is 10000ms.

(Continued…)

(...Continued)

TSI_USBC_EL_AUX_P_IDLE_LOW_VOLTAGE
DP AUX Positive line low voltage limit when AUX is idle, in millivolts. Default setting is 100mV.

TSI_USBC_EL_AUX_P_IDLE_HI_VOLTAGE
DP AUX Positive line high voltage limit when AUX is idle, in millivolts. Default setting is 600mV.

TSI_USBC_EL_AUX_N_IDLE_LOW_VOLTAGE
DP AUX Negative line low voltage limit when AUX is idle, in millivolts. Default setting is 2500mV.

TSI_USBC_EL_AUX_N_IDLE_HI_VOLTAGE
DP AUX Negative line high voltage limit when AUX is idle, in millivolts. Default setting is 3000mV

RAW Test results data

*TSI_R_TDATA_USBC_VAUX1_**
AUX1 positive and negative line voltage levels in millivolts.

*TSI_R_TDATA_USBC_VAUX2_**
AUX2 positive and negative line voltage levels in millivolts.

4.2.18.USBC Electrical Test Set / DUT as Power Sink

ClientVersion 10, and higher *Electrical test license required*

```
#define TSI_TEST_USBC_DUT_PWR_SINK              0x000c0002
```

Synopsis

This test verifies operation 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 the mandatory PDO for power contract. In order to run this test, the DUT must support Power Sink role.

The test starts by negotiating the power contract. Once the power contract is established, the TE will wait for power measurement delay before measuring current over Vbus and GND lines. Voltage over Vbus is also measured. The power measurement delay can be modified with the *TSI_USBC_EL_PWR_MEASURE_DELAY* Configuration item. The purpose of the delay is to allow the DUT some time to stabilize it's power consumption. As the currents are measured one at a time, any variance in power consumption in the DUT during the measurement can cause this test to fail. The test assumes that the current flows through the four separate Vbus and GND lines evenly, and the contacts are verified with this characteristic in mind. Total currents are calculated for Vbus, and for GND. The highest difference between each of the four connections may not exceed the programmed deviation limits. The deviation is defined as per-mill of the total currents for Vbus and GND respectively.

In order to pass the test, the measured Vbus voltage must be between the *TSI_USBC_EL_VBUS_LOW_VOLTAGE* and *TSI_USBC_EL_VBUS_HI_VOLTAGE*. In addition, the currents measured from Vbus may not deviate more than indicated by *TSI_USBC_EL_VBUS_CURRENT_MAX_DEV*. Also, the currents measured from GND lines may not deviate more than indicated by *TSI_USBC_EL_GND_CURRENT_MAX_DEV*.

Important: *In order to run this test with UCD-340, a special cable provided by Unigraf must be used.*

Important: *In order to run this test with UCD-340, the Electrical Testing add-on board must be installed on the device.*

TSI_USBC_EL_TIMEOUT
: Test timeout, in milliseconds. Default setting is 5000ms.

TSI_USBC_EL_DUT_CAPS
: Defines DUT capbabilities. See 5.20.2 TSI_USBC_EL_DUT_CAPS for details. Default setting is 0.

TSI_USBC_EL_REPLUG_TIME
: Simulated cable re-plug disconnected state time, in milliseconds. Default setting is 1500ms.

TSI_USBC_EL_DUT_ATTACH_TIMEOUT
: Maximum time allowed for the DUT to respond to cable plug, in milliseconds. Default setting is 10000ms.

TSI_USBC_EL_PWR_CONTRACT_TIMEOUT
: Maximum time allowed for the DUT to establish power contract, in milliseconds. Default setting is 5000ms

(Continued…)

(...Continued)

TSI_USBC_EL_VBUS_LOW_VOLTAGE
Vbus voltage range, lower limit, in millivolts. Default setting is 4750mV.

TSI_USBC_EL_VBUS_HI_VOLTAGE
Vbus voltage range, higher limit, in millivolts. Default setting is 5500mV.

TSI_USBC_EL_VBUS_CURRENT_MAX_DEV
Maximum deviation allowed between the four individual Vbus pins defined as per-mill from measured total current. Default setting 100‰.

TSI_USBC_EL_GND_CURRENT_MAX_DEV
Maximum deviation allowed between the four individual GND pins defined as per-mill from measured total current. Default setting 100‰.

TSI_USBC_EL_PWR_MEASURE_DELAY
Delay between Power Contract completion and power measurement, in milliseconds. Default setting is 2000ms.

TSI_USBC_EL_MIN_DUT_CURRENT
Minimum current that must be used by the Power Sink DUT in order to pass the test. Default setting is 0mA.

RAW Test results data

TSI_R_TDATA_USBC_EL_VBUS_V
Vbus voltage, in millivolts.

*TSI_R_TDATA_USBC_EL_VBUS_I**
Vbus currents for power lines 1, 2, 3 and 4 in milliamperes.

*TSI_R_TDATA_USBC_EL_GND_I**
Vbus currents for ground lines 1, 2, 3 and 4 in milliamperes.

4.2.19.USBC Electrical Test Set / DUT as Power Source

ClientVersion 10, and higher *Electrical test license required*

```
#define TSI_TEST_USBC_DUT_PWR_SOURCE          0x000c0003
```

Synopsis

This test verifies operation 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 sink, and selects the mandatory PDO for power contract. In order to run this test, the DUT must support Power Source role.

The test starts by negotiating the power contract. Once the power contract is established, the TE will wait for power measurement delay before measuring current over Vbus and GND lines. Voltage over Vbus is also measured. The power measurement delay can be modified with the *TSI_USBC_EL_PWR_MEASURE_DELAY* Configuration item. As the currents are measured one at a time, any variance in power delivery form the DUT during the measurement can cause this test to fail. The test assumes that the current flows through the four separate Vbus and GND lines evenly, and the contacts are verified with this characteristic in mind. Total currents are calculated for Vbus, and for GND. The highest difference between each of the four connections may not exceed the programmed deviation limits. The deviation is defined as per-mill of the total currents for Vbus and GND respectively.

In order to pass the test, the measured Vbus voltage must be between the *TSI_USBC_EL_VBUS_LOW_VOLTAGE* and *TSI_USBC_EL_VBUS_HI_VOLTAGE*. In addition, the currents measured from Vbus may not deviate more than indicated by *TSI_USBC_EL_VBUS_CURRENT_MAX_DEV*. Also, the currents measured from GND lines may not deviate more than indicated by *TSI_USBC_EL_GND_CURRENT_MAX_DEV*.

Important: *In order to run this test with UCD-340, a special cable provided by Unigraf must be used.*

Important: *In order to run this test with UCD-340, the Electrical Testing add-on board must be installed on the device.*

TSI_USBC_EL_TIMEOUT
: Test timeout, in milliseconds. Default setting is 5000ms.

TSI_USBC_EL_DUT_CAPS
: Defines DUT capabilities. See 5.20.2 TSI_USBC_EL_DUT_CAPS for details. Default setting is 0.

TSI_USBC_EL_REPLUG_TIME
: Simulated cable re-plug disconnected state time, in milliseconds. Default setting is 1500ms.

TSI_USBC_EL_DUT_ATTACH_TIMEOUT
: Maximum time allowed for the DUT to respond to cable plug, in milliseconds. Default setting is 10000ms.

TSI_USBC_EL_PWR_CONTRACT_TIMEOUT
: Maximum time allowed for the DUT to establish power contract, in milliseconds. Default setting is 5000ms

(Continued…)

(…Continued)

TSI_USBC_EL_VBUS_LOW_VOLTAGE
Vbus voltage range, lower limit, in millivolts. Default setting is 4750mV.

TSI_USBC_EL_VBUS_HI_VOLTAGE
Vbus voltage range, higher limit, in millivolts. Default setting is 5500mV.

TSI_USBC_EL_VBUS_CURRENT_MAX_DEV
Maximum deviation allowed between the four individual Vbus pins defined as per-mill from measured total current. Default setting 100‰.

TSI_USBC_EL_GND_CURRENT_MAX_DEV
Maximum deviation allowed between the four individual GND pins defined as per-mill from measured total current. Default setting 100‰.

TSI_USBC_EL_PWR_MEASURE_DELAY
Delay between Power Contract completion and power measurement, in milliseconds. Default setting is 2000ms.

RAW Test results data

TSI_R_TDATA_USBC_EL_VBUS_V
Vbus voltage, in millivolts.

*TSI_R_TDATA_USBC_EL_VBUS_I**
Vbus currents for power lines 1, 2, 3 and 4 in milliamperes.

*TSI_R_TDATA_USBC_EL_GND_I**
Vbus currents for ground lines 1, 2, 3 and 4 in milliamperes.

5. CONFIGURATION ITEM DEFINITIONS

This section defines configuration items for tests and control features available in TSI.

5.1.Generic realtime measurements

This section defines Configuration Items used to access generic measurement data for devices that support it.

5.1.1.ADC Data access CI range

ClientVersion 11, and higher

```
TSI_R_ADC_FIRST                                 0x12000
TSI_R_ADC_LAST                                  0x120ff
```

Type information

```
unsigned int ADC_Data[0x100]                    ARRAY_U32
0 to 1024 bytes                                 RO
```

Description

The CI range from 0x12000 to 0x120ff maps into a block of up to 256 4-byte data containers. These data available through this data block depends on the used device.

Each CI ID in this range supports variable size read, so it is possible to get complete results structure in one read, starting at a user selectable DWORD offset into the structure.

5.1.2.ADC Data available on UCD-340

ClientVersion 11, and higher

See table below for convenience CI ID definitions that make it easy to access certain data from the ADC Data range.

Define	Config ID	Description
TSI_R_ADC_VBUS_VOLTAGE	0x12000	Voltage on Vbus, in millivolts (mV).
TSI_R_ADC_VBUS_CURRENT	0x12001	Current on Vbus, in milliamperes (mA).
TSI_R_ADC_VCC1_VOLTAGE	0x12002	Voltage on CC1, in millivolts (mV).
TSI_R_ADC_VCC2_VOLTAGE	0x12003	Voltage on CC2, in millivolts (mV).
TSI_R_ADC_VCONN_VOLTAGE	0x12004	Voltage on Vconn, in millivolts (mV).
TSI_R_ADC_VCONN_CURRENT	0x12005	Current on Vconn, in milliamperes (mA).

5.1.3.TSI_W_USBC_ADC_CTRL

ClientVersion 11, and higher

```
TSI_W_USBC_ADC_CTRL                              0x12100
unsigned int usbc_adc_ctrl                       U32
4 bytes                                          WO
```

Synopsis

Controls the ADC real time measurement system. Available commands are: 1 = Enable ADC data scanner. 2 = Disable ADC data scanner. The ADC scanner is disabled by default.

5.2.Generic low-level test results

This section defines Configuration Items used to read low-level test results from various tests.

5.2.1.RAW test results access CI range

ClientVersion 11, and higher

```
TSI_R_TDATA_BLOCK_FIRST                          0x0F800
TSI_R_TDATA_BLOCK_LAST                           0x0FFD0
```

Type information

```
unsigned int TData[0x7d0]                        ARRAY_U8
1 to 8000 bytes                                  RO
```

Description

The CI range starting from CI ID 0x0F800 up to 0x0FFD0 maps to a 8000 byte memory region with 4 byte access granularity. Any TSI test can return results through this data block in a test specific structure. Please refer to test descriptions / CI information for details per test.

Each CI ID in this range supports variable size read, so it is possible to get complete results structure in one read, starting at a user selectable DWORD offset into the structure. Unused bytes will have the value 0xAF.

5.2.2.TSI_R_TDATA_BLOCK_SIZE

ClientVersion 11, and higher

```
TSI_R_TDATA_BLOCK_SIZE                           0x0FFFF
unsigned int TdataRawSize                        U32
4 bytes                                          RO
```

Description

This CI contains number of bytes of RAW test results data available for reading from CI TSI_R_TDATA_GENERIC_STRUCT_VERSION.

5.2.3.TSI_R_TDATA_GENERIC_STRUCT_VERSION

ClientVersion 11, and higher

```
TSI_R_TDATA_GENERIC_STRUCT_VERSION            0x0f800
unsigned int TData                            U32
4 bytes                                       RO
```

Description

Each test results table begins with version information so that applications can detect if the wanted field is available in the results data.

Available after running any test that supports RAW results gathering.

5.2.4.TSI_R_TDATA_USBC_EL_VCC*

ClientVersion 11, and higher

```
TSI_R_TDATA_USBC_EL_VCC1                      0x0f801
TSI_R_TDATA_USBC_EL_VCC2                      0x0f803
unsigned int Tdata                            U32
4 bytes                                       RO
```

Description

Contains CC1 and CC2 voltage measurements as millivolts.

Available after running test with ID 0xC0000 (4.2.16 USBC Electrical Test Set / Up Face port CC and Vconn test).

5.2.5.TSI_R_TDATA_USBC_EL_VCONN*

ClientVersion 11, and higher

```
TSI_R_TDATA_USBC_EL_VCONN1                    0x0f802
TSI_R_TDATA_USBC_EL_VCONN2                    0x0f804
unsigned int Tdata                            U32
4 bytes                                       RO
```

Description

Contains VConn1 and VConn2 line voltages when in Vconn role as millivolts.

Available after running test with ID 0xC0000 (4.2.16 USBC Electrical Test Set / Up Face port CC and Vconn test).

5.2.6.TSI_R_TDATA_USBC_VAUX1_*

ClientVersion 11, and higher

TSI_R_TDATA_USBC_VAUX1_P	0x0f801
TSI_R_TDATA_USBC_VAUX1_N	0x0f802
unsigned int Tdata	U32
4 bytes	RO

Description

Positive (_P) and Negative (_N) voltage levels on AUX1 line when in direct cable mode as millivolts.

Available after running test with ID 0xC0001 (4.2.17 USBC Electrical Test Set / AUX (SBU) lines test).

5.2.7.TSI_R_TDATA_USBC_VAUX2_*

ClientVersion 11, and higher

TSI_R_TDATA_USBC_VAUX2_P	0x0f803
TSI_R_TDATA_USBC_VAUX2_N	0x0f804
unsigned int Tdata	U32
4 bytes	RO

Description

Positive (_P) and Negative (_N) voltage levels on AUX2 line when in crossed cable mode as millivolts.

Available after running test with ID 0xC0001 (4.2.17 USBC Electrical Test Set / AUX (SBU) lines test).

5.2.8.TSI_R_TDATA_USBC_EL_VBUS_V

ClientVersion 11, and higher

TSI_R_TDATA_USBC_EL_VBUS_V	0x0f801
unsigned int TData	U32
4 bytes	RO

Description

Contains Vbus voltage, in millivolts.

Available after running tests with ID 0xC0002 (4.2.18 USBC Electrical Test Set / DUT as Power Sink) or 0xC0003 (4.2.19 USBC Electrical Test Set / DUT as Power Source).

5.2.9.TSI_R_TDATA_USBC_EL_VBUS_I*

ClientVersion 11, and higher

TSI_R_TDATA_USBC_EL_VBUS_I1	0x0f802
TSI_R_TDATA_USBC_EL_VBUS_I2	0x0f803
TSI_R_TDATA_USBC_EL_VBUS_I3	0x0f804
TSI_R_TDATA_USBC_EL_VBUS_I4	0x0f805
unsigned int Tdata	U32
4 bytes	RO

Description

Contains Vbus current, in milliamps for each of the four Vbus wires separately. To get total current, the values must be added together.

Available after running tests with ID 0xC0002 (4.2.18 USBC Electrical Test Set / DUT as Power Sink) or 0xC0003 (4.2.19 USBC Electrical Test Set / DUT as Power Source).

5.2.10.TSI_R_TDATA_USBC_EL_GND_I*

ClientVersion 11, and higher

TSI_R_TDATA_USBC_EL_GND_I1	0x0f806
TSI_R_TDATA_USBC_EL_GND_I1	0x0f807
TSI_R_TDATA_USBC_EL_GND_I1	0x0f808
TSI_R_TDATA_USBC_EL_GND_I1	0x0f809
unsigned int Tdata	U32
4 bytes	RO

Description

Contains GND current, in milliamps, for each of the four Vbus ground wires separately. To get total current, the values must be added together.

Available after running tests with ID 0xC0002 (4.2.18 USBC Electrical Test Set / DUT as Power Sink) or 0xC0003 (4.2.19 USBC Electrical Test Set / DUT as Power Source).

5.3.Reference frames

This section defines configuration items used to define reference frame for pixel-by-pixel video test. Logically, the method used for describing the frame attempts to cover most cases where the actual bitmap is not consisting out of singular pixels, like for example YCbCr 4:2:2, where one repeating structure defines two pixels. In TSI, these repeating structures are referred to as elements. Each element has width, height, size and format information. For typical RGB images, the elements are 1x1 in size, but YCbCr 4:2:2 elements are 2x1, and YCbCr elements are 2x2. The image must be a multiple of elements.

5.3.1.TSI_REF1_WIDTH

```
TSI_REF1_WIDTH                                  0x10
unsigned int Ref1W                              U32
4 bytes                                         RW
```

Description

Defines frame width as number of elements. Actual width in pixels is therefore this value multiplied by the element width.

5.3.2.TSI_REF1_HEIGHT

```
TSI_REF1_HEIGHT                                 0x11
unsigned int Ref1H                              U32
4 bytes                                         RW
```

Description

Defines frame height as number of elements. Actual height in pixels is therefore this value multiplied by the element height.

5.3.3.TSI_REF1_ELEMENT_SIZE

```
TSI_REF1_ELEMENT_SIZE                           0x12
unsigned int Ref1ElementSize                    U32
4 bytes                                         RW
```

Description

Defines the size of the element body, in as bytes of storage required. Certain formats allow this container to be of different size: For example RGB 8:8:8 can have size of 3 and/or 4. Often, the 4 byte version is referred to as ARGB, but TSI does not process the Alpha (“A”) channel, so the presence of that is ignored.

5.3.4.TSI_REF1_ELEMENT_WIDTH

```
TSI_REF1_ELEMENT_WIDTH                          0x13
(TSI_REF1_PIXELS_PER_ELEMENT                    0x13)
unsigned int Ref1ElementW                       U32
4 bytes                                         RW
```

Description

Defines the width of a single element as number of pixels.

Important: *TSI_REF1_PIXELS_PER_ELEMENT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.3.5.TSI_REF1_ELEMENT_HEIGHT

```
TSI_REF1_ELEMENT_HEIGHT                         0x14
(TSI_REF1_LINES_PER_ELEMENT                     0x14)
unsigned int Ref1ElementH                       U32
4 bytes                                         RW
```

Description

Defines the height of a single element as number of pixels.

Important: *TSI_REF1_LINES_PER_ELEMENT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.3.6.TSI_REF1_COLOR_DEPTH

```
TSI_REF1_COLOR_DEPTH                            0x15
unsigned itn Ref1ColorDepth                     U32
4 bytes                                         RW
```

Description

Defines the color depth of the image as number of bits per color channel regardless of the color format. Please notice that this information is not in fact used as operational value in TSI, but rather as a generic meta information.

5.3.7.TSI_REF1_ELEMENT_FORMAT

```
TSI_REF1_ELEMENT_FORMAT                          0x16
(TSI_REF1_PIXEL_FORMAT                           0x16)
unsigned int Ref1ElementFormat                   U32
4 bytes                                          RW
```

Description

Defines the element format used to encode the pixel data of the bitmap. Please see table below for currently defined format ID values:

Define	ID	Description
TSI_ELF_RGB_080808	0	RGB color, max color depth 8 bits per channel. Encoded as 3 unsigned bytes or 4 unsigned bytes per element.
TSI_ELF_RGB_161616	1	RGB color, max color depth 16 bits per channel. Encoded as 3 unsigned shorts or 4 unsigned shorts per element.
TSI_ELF_YCbCr_080808	0x100	YCbCr color, max color depth 8 bits per channel. Encoded as 3 unsigned bytes or 4 unsigned bytes per element.
TSI_ELF_YCbCr_161616	0x101	YCbCr color, max color depth 16 bits per channel. Encoded as 3 unsigned shorts or 4 unsigned shorts per element.

Important: *TSI_REF1_PIXEL_FORMAT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.3.8.TSI_REF1_FRAME_DATA

```
TSI_REF1_FRAME_DATA                              0x17
unsigned char Ref1Data[]                         ARRAY_U8
Variable size                                    RW
```

Description

Contains bitmap data encoded as defined in other TSI_REF1_* CI's.

5.4.Input video format

The following configuration items define the video format being received by the selected video input and the memory layout selected for it by TSI.

5.4.1.TSI_R_INPUT_WIDTH

```
TSI_R_INPUT_WIDTH                                   0x200
unsigned int InputW                                 U32
4 bytes                                             RO
```

Description

Defines the video frame width as number of elements. To get width as number of pixels, multiply this value with value of TSI_R_INPUT_ELEMENT_WIDTH CI.

5.4.2.TSI_R_INPUT_HEIGHT

```
TSI_R_INPUT_HEIGHT                                  0x201
unsigned int InputH                                 U32
4 bytes                                             RO
```

Description

Defines the video frame height as number of elements. To get height as number of pixels, multiply this value with value of TSI_R_INPUT_ELEMENT_HEIGHT CI.

5.4.3.TSI_R_INPUT_FREQ

```
TSI_R_INPUT_FREQ                                    0x202
unsigned int InputFrequency                         U32
4 bytes                                             RO
```

Description

Defines the frame rate of the input video signal as a fixed point value with scale factor of 10. To get Hz, divide the value with the scale factor.

5.4.4.TSI_R_INPUT_ELEMENT_SIZE

```
TSI_R_INPUT_ELEMENT_SIZE                    0x203
unsigned int InputElemSize                  U32
4 bytes                                     RO
```

Description

Defines the RAM storage size of a single element as number of bytes.

5.4.5.TSI_R_INPUT_ELEMENT_WIDTH

```
TSI_R_INPUT_ELEMENT_WIDTH                   0x204
(TSI_R_INPUT_PIXELS_PER_ELEMENT             0x205)
unsigned int InputElemW                     U32
4 bytes                                     RO
```

Description

Defines the width of a single element as number of pixels.

Important: *The define TSI_R_INPUT_PIXELS_PER_ELEMENT is now considered obsolete, but it remains to be defined for backwards compatibility. The define was changed to better describe the data.*

5.4.6.TSI_R_INPUT_ELEMENT_HEIGHT

```
TSI_R_INPUT_ELEMENT_HEIGHT                  0x205
(TSI_R_INPUT_LINES_PER_ELEMENT              0x205)
unsigned int InputElemH                     U32
4 bytes                                     RO
```

Description

Defines the height of a single element as number of pixels.

Important: *The define TSI_R_INPUT_LINES_PER_ELEMEMENT is now considered obsolete, but it remains to be defined for backwards compatibility. The define was changed to better describe the data.*

5.4.7.TSI_R_INPUT_COLOR_DEPTH

```
TSI_R_INPUT_COLOR_DEPTH                          0x206
unsigned int InputColorDepth                     U32
4 bytes                                          RO
```

Description

Defines the color depth of the input signal. While this value has no effect on the element memory layout, it does indicate how many color bits are expected to be received from the video input.

5.4.8.TSI_R_INPUT_ELEMENT_FORMAT

```
TSI_R_INPUT_ELEMENT_FORMAT                       0x207
(TSI_R_INPUT_PIXEL_FORMAT                        0x207)
unsigned int InputElemFormat                     U32
4 bytes                                          RO
```

Description

Defines the element format used to encode the pixel data of the bitmap. Please see table below for currently defined format ID values:

Define	ID	Description
TSI_ELF_RGB_080808	0	RGB color, max color depth 8 bits per channel. Encoded as 3 unsigned bytes or 4 unsigned bytes per element.
TSI_ELF_RGB_161616	1	RGB color, max color depth 16 bits per channel. Encoded as 3 unsigned shorts or 4 unsigned shorts per element.
TSI_ELF_YCbCr_080808	0x100	YCbCr color, max color depth 8 bits per channel. Encoded as 3 unsigned bytes or 4 unsigned bytes per element.
TSI_ELF_YCbCr_161616	0x101	YCbCr color, max color depth 16 bits per channel. Encoded as 3 unsigned shorts or 4 unsigned shorts per element.

Important: *TSI_R_INPIT_PIXEL_FORMAT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.4.9.TSI_R_INPUT_INTERLACE

```
TSI_R_INPUT_INTERLACE                            0x208
unsigned int InputInterlaced                     U32
4 bytes                                          RO
```

Description

Indicates if the input signal is interlaced. 0 = Progressive, 1 = Interlaced.

5.5.Input audio format

This section defines configuration items define input audio stream properties.

5.5.1.TSI_R_AUDIO_CHANNELS

```
TSI_R_AUDIO_CHANNELS                          0x220
unsigned int AudioChannels                    U32
4 bytes                                       RO
```

Description

Indicates number of active audio channels. Number of channels depends on source device, and typically range from 1 to 8 channels.

5.5.2.TSI_R_AUDIO_SAMPLE_RATE

```
TSI_R_AUDIO_SAMPLE_RATE                       0x221
unsigned int AudioRate                        U32
4 bytes                                       RO
```

Description

Indicates audio sample rate in Hz.

5.5.3.TSI_R_AUDIO_SAMPLE_SIZE

```
TSI_R_AUDIO_SAMPLE_SIZE                       0x222
unsigned int AudioSampleSize                  U32
4 bytes                                       RO
```

Description

Indicates audio sample size in bits.

5.5.4.TSI_CAPTURE_AUDIO_MASK

```
TSI_CAPTURE_AUDIO_MASK                          0x238
unsigned int AudioCapMask                       U32
4 bytes                                         RW
```

Description

This configuration item is supported on devices that cannot detect active audio channels and must rely on client application defining the active channels instead. The value is a bit-mask that defines which audio channels from 0 to 7 are being used. Normally, a 5.1 audio would result to value of 63, while the default value “3” translates to stereo (Channels 0 and 1).

5.6.V-by-One inputs

This section contains definitions for configuration items related to V-by-One inputs.

5.6.1.TSI_VX1_SIGNAL_COLOR_DEPTH

```
TSI_VX1_SIGNAL_COLOR_DEPTH                      0x240
unsigned int VX1_ColorDepth                     U32
4 bytes                                         RW
```

Description

Defines color depth expected on the V-by-One input. 0 = 6 bits per color channel, 1 = 8 bits per color channel, 2 = 10 bits per color channel, 3 = 12 bits per color channel. Default setting is “1” (8 bits per color channel).

5.6.2.TSI_VX1_SIGNAL_CHANNELS_PER_UNIT

```
TSI_VX1_SIGNAL_CHANNELS_PER_UNIT                0x241
unsigned int VX1_ChannelsPerU                   U32
4 bytes                                         RW
```

Description

Defines the number of V-by-One channels to capture per unit. Valid settings are 1, 2, 4 or 8. Actual number of channels is this setting multiplied by the value from TSI_R_UNITS_PRESENT CI. <REF_TODO>. Default value is 8.

Important: *Setting this value also resets <REF_TODO> TSI_VX1_SECTION_COUNT CI to it’s default value of one (1).*

5.6.3.TSI_VX1_SIGNAL_SYNC_MODE

```
TSI_VX1_SIGNAL_SYNC_MODE                    0x242
unsigned int VX1_SyncMode                   U32
4 bytes                                     RW
```

Description

Defines the signal sync mode used. 0 = Data enable, 1 = H-Sync + V-Sync. Default setting is “0” (Data enable).

5.6.4.TSI_VX1_HTPDN_CONTROL

```
TSI_VX1_HTPDN_CONTROL                       0x243
unsigned int VX1_HTPDN                      U32
4 bytes                                     RW
```

Description

Defines behavior of the HTPDN signal. 0 = Normal operation, 1 = Force Low, 2 = Force High. Default setting is “0” (Normal operation).

5.6.5.TSI_VX1_LOCKN_CONTROL

```
TSI_VX1_LOCKN_CONTROL                       0x244
unsigned int VX1_LOCKN                      U32
4 bytes                                     RW
```

Description

Defines the behavior of the LOCKN signal. 0 = Normal operation, 1 = Force Low, 2 = Force High, 3 = Force low after TSI_VX1_LOCKN_DELAY. Default setting is “0” (Normal operation).

5.6.6.TSI_VX1_LOCKN_DELAY

```
TSI_VX1_LOCKN_DELAY                         0x245
unsigned int VX1_LOCKN_Delay                U32
4 bytes                                     RW
```

Description

Delay time in µs. This delay is used only if TSI_VX1_LOCKN_CONTROL is set to “3”. Default setting is 41900 µs.

5.6.7.TSI_VX1_VIDEO_VALID_DELAY

```
TSI_VX1_VIDEO_VALID_DELAY                    0x246
unsigned int VX1_VideoValidDelay             U32
4 bytes                                      RW
```

Description

Delay time in µs. Default setting is 41900 µs.

5.6.8.TSI_VX1_FRAME_COMBINE_METHOD

```
TSI_VX1_FRAME_COMBINE_METHOD                 0x247
unsigned int VX1_CombineMode                 U32
4 bytes                                      RW
```

Description

Defines how the frame is built from the individual V-by-One channels. 0 = Default mode (One from each lane is sequence).

5.6.9.TSI_VX1_SECTION_COUNT

```
TSI_VX1_SECTION_COUNT                        0x248
unsigned int VX1_Sections                    U32
4 bytes                                      RW
```

Description

Number of V-by-One Sections the source is sending. Valid setting is any positive value greater than 1 that produces a zero modulus when dividing V-by-One total channels by the new setting. Total V-by-One channels used can be calculated by multiplying values of configuration items TSI_R_UNITS_PRESENT and TSI_VX1_SIGNAL_CHANNELS_PER_UNIT. Default is 1 section.

Important: *Set this configuration item after setting the configuration item TSI_VX1_SIGNAL_CHANNELS_PER_UNIT, as setting it also sets this CI to it's default value of 1.*

5.7.LVDS Inputs

This section contains definitions related to LVDS inputs.

5.7.1.TSI_LVDS_CHANNELS

TSI_LVDS_CHANNELS	0x2a0
unsigned int LVDS_Channels	U32
4 bytes	RW

Description

Defines number of LVDS channels to use per LVDS input. Valid settings are 1, 2 or 4. Default setting is 4 (Quad) LVDS.

5.7.2.TSI_LVDS_SIGNAL_COLOR_DEPTH

TSI_LVDS_SIGNAL_COLOR_DEPTH	0x2a1
unsigned int LVDS_ColorDepth	U32
4 bytes	RW

Description

Defines input signal's color depth. 0 = 6 bits per color channel, 1 = 8 bits per color channel, 2 = 10 bits per color channel, 3 = 12 bits per color channel. Default setting is "1" (8 bits per color channel).

5.7.3.TSI_LVDS_MAPPING_MODE

TSI_LVDS_MAPPING_MODE	0x2a2
unsigned int LVDS_MapMode	U32
4 bytes	RW

Description

Defines pin mapping mode for an LVDS input. 0 = VESA, 1 = JEIDA. Default setting is "1" (JEIDA).

5.8.Video test

The configuration items below are used with test “Compare video frame sequence with a single reference frame”.

5.8.1.TSI_TEST_LENGTH

```
TSI_TEST_LENGTH                                 0x1000
unsigned int TSI_VidTestLen                     U32
4 bytes                                         RW
```

Description

Defines the length of the video test as number of frames. Default setting is 60 frames.

Important: *32-bit version of TSI with very high resolution inputs may require the test length to be reduced.*

5.8.2.TSI_LIM_FRAME_MISMATCHES

```
TSI_LIM_FRAME_MISMATCHES                        0x1001
unsigned int TSI_FrameMismatches                U32
4 bytes                                         RW
```

Description

Defines number of frame that are allowed to be considered as “failed” before the entire test is considered as “failed”. Default setting is 0.

5.8.3.TSI_LIM_PIXEL_MISMATCHES

```
TSI_LIM_PIXEL_MISMATCHES                        0x1002
unsigned int TSI_PxlMismatches                  U32
4 bytes                                         RW
```

Description

Defines the number of pixels that allowed to be considered as “failed” before the frame is considered as “failed”. Default setting is 0.

5.8.4.TSI_PIXEL_TOLERANCE

```
TSI_PIXEL_TOLERANCE                          0x1003
unsigned int TSI_ColorTolerance              U32
4 bytes                                      RW
```

Description

Defines maximum difference allowed between reference image and captured image. If the difference is larger than the value of this CI on any color channel, the pixel is considered “failed”. Default setting is 0.

5.8.5.TSI_MAX_AUTO_SAVE_FAILED

```
TSI_MAX_AUTO_SAVE_FAILED                     0x1080
unsigned int TSI_MaxAutosave                 U32
4 bytes                                      RW
```

Description

Maximum number of frames failed frames saved per test run. Default setting is 0. If the setting is “0”, no frames are saved.

5.8.6.TSI_FAILED_FRAME_TARGET_FOLDER

```
TSI_FAILED_FRAME_TARGET_FOLDER               0x1081
char FailedFramesFolder[]                    ARRAY_U8
Variable size from 1 to 260 bytes            RW
```

Description

Contains the full path to the folder where failed frames are to be saved **without** trailing backslash (‘\’). No default. Failed frame file-name will be “Failed_<#>.ppm”, where <#> is replaced with an auto-incremented number.

5.8.7.TSI_MAX_EXPORT_FAILED

```
TSI_MAX_EXPORT_FAILED                        0x1082
unsigned int MaxExportFailed                 U32
4 bytes                                      RW
```

Description

Defines the number of failed frames to be exported from the video test. Default setting is 0. If the setting is 0, no frames are exported.

5.8.8.TSI_R_VIDEO_TEST_RAW_RESULTS_DATA

```
TSI_R_VIDEO_TEST_RAW_RESULTS_DATA                0x1008
unsigned int RawResults[]                        ARRAY_U32
Variable size                                    RO
```

Description

Provides access to an array of integers that contain error counts per each compared frame. The maximum size for this block is test length multiplied with size of 4 unsigned integers (unsigned 32 bit values). See below for description of each block of 4 integers:

Byte Offset	Description
0	Number of errors on red color channel (Or Cr channel for YCbCr).
4	Number of errors on green color channel (or Y channel for YCbCr).
8	Number of errors on blue color channel (or Cb channel for YCbCr).
12	Number of failed pixels.

5.8.9.TSI_EXPORTED

```
TSI_EXPORTED                                     0x108e
unsigned int Exported                            U32
4 bytes                                          RO
```

Description

Defines number of frames failed frames exported from the video test. No default. Please notice that exported frames are not accessible after the next test is started.

5.8.10.TSI_EXPORT_ACCESS_INDEX

```
TSI_EXPORT_ACCESS_INDEX                          0x108f
unsigned int ExportIndex                         U32
4 bytes                                          WO
```

Description

Defines which export frame information to access with CI ID's 0x1090 to 0x1097. Allowed value range is from 0 to number of exported frames.

5.8.11.TSI_EXPORT_WIDTH

```
TSI_EXPORT_WIDTH                                0x1090
unsigned int ExportW                            U32
4 bytes                                         RO
```

Description

Defines the width of an exported frame as number of elements.

5.8.12.TSI_EXPORT_HEIGHT

```
TSI_EXPORT_HEIGHT                               0x1091
unsinged int ExportH                            U32
4 bytes                                         RO
```

Description

Defines height of an exported frame as number of elements.

5.8.13.TSI_EXPORT_ELEMENT_SIZE

```
TSI_EXPORT_ELEMENT_SIZE                         0x1092
unsigned int ExportElemSize                     U32
4 bytes                                         RO
```

Description

Defines the size of element as number of bytes.

5.8.14.TSI_EXPORT_ELEMENT_WIDTH

```
TSI_EXPORT_ELEMENT_WIDHT                        0x1093
(TSI_EXPORT_PIXELS_PER_ELEMENT                  0x1093)
unsinged int ExportElemW                        U32
4 bytes                                         RO
```

Description

Defines the width of a single element in pixels.

Important: *TSI_EXPORT_PIXELS_PER_ELEMENT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.8.15.TSI_EXPORT_ELEMENT_HEIGHT

```
TSI_EXPORT_ELEMENT_HEIGHT                    0x1094
(TSI_EXPORT_LINES_PER_ELEMENT                0x1094)
unsigned int ExportElemH                     U32
4 bytes                                      RO
```

Description

Defines the height of a single element in pixels.

Important: *TSI_EXPORT_LINES_PER_ELEMENT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.8.16.TSI_EXPORT_COLOR_DEPTH

```
TSI_EXPORT_COLOR_DEPTH                       0x1095
unsinged int ExportColorDepth                U32
4 bytes                                      RO
```

Description

Defines the color depth per color channel as number of bits.

5.8.17.TSI_EXPORT_ELEMENT_FORMAT

```
TSI_EXPORT_ELEMENT_FORMAT                    0x1096
(TSI_EXPORT_PIXEL_FORMAT                     0x1096)
unsigned int ExportElemFormat                U32
4 bytes                                      RO
```

Description

Defines the element format used to encode the pixel data of the bitmap. Please see table below for currently defined format ID values:

Define	ID	Description
TSI_ELF_RGB_080808	0	RGB color, max color depth 8 bits per channel. Encoded as 3 unsigned bytes or 4 unsigned bytes per element.
TSI_ELF_RGB_161616	1	RGB color, max color depth 16 bits per channel. Encoded as 3 unsigned shorts or 4 unsigned shorts per element.
TSI_ELF_YCbCr_080808	0x100	YCbCr color, max color depth 8 bits per channel. Encoded as 3 unsigned bytes or 4 unsigned bytes per element.
TSI_ELF_YCbCr_161616	0x101	YCbCr color, max color depth 16 bits per channel. Encoded as 3 unsigned shorts or 4 unsigned shorts per element.

Important: *TSI_EXPORT_PIXEL_FORMAT name define is considered obsolete, however it continues to be defined for backwards compatibility. The new name was incorporated as it is more descriptive.*

5.8.18.TSI_EXPORT_FRAME_DATA

TSI_EXPORT_FRAME_DATA	0x1097
void *ExportFrameData	ARRAY_U8
Variable size	RO

Description

Contains the frame data. The data size (in bytes) can be calculated by multiplying TSI_EXPORT_WIDTH by TSI_EXPORT_HEIGHT by TSI_EXPORT_ELEMENT_SIZE.

5.9.Audio test

This section defines the configuration items used to set-up audio test.

5.9.1.TSI_EXPECTED_SAMPLE_RATE

TSI_EXPECTED_SAMPLE_RATE	0x2020
unsigned int AudioSampleRate	U32
4 bytes	RW

Description

Sample rate that should be present when running the test, in Hz. Default setting is 44100 Hz.

5.9.2.TSI_EXPECTED_AUDIO_FREQUENCY

TSI_EXPECTED_AUDIO_FREQUENCY	0x2021
unsinged int AudioFreq	U32
4 bytes	RW

Description

Expected audible signal frequency that should be present when running the test, in Hz. Default setting is 1000 Hz.

5.9.3.TSI_AUDIO_FREQUENCY_TOLERANCE

```
TSI_AUDIO_FREQUENCY_TOLERANCE                    0x2022
unsigned int AudioFreqTolerance                  U32
4 bytes                                          RW
```

Description

Maximum allowed frequency deviation for the audible signal from the reference frequency, in Hz. Default setting is 1 Hz.

5.9.4.TSI_AUDIO_GLITCH_DETECT_TRESHOLD

```
TSI_AUDIO_GLITCH_DETECT_TRESHOLD                 0x2023
unsigned int AudioGlitchTreshold                 U32
4 bytes                                          RW
```

Description

Threshold value used to detect audible clicks caused by dropped or duplicated samples. Value is encoded as fixed point with scale factor of 65536 (16 bits). Integer portion is stored in bits 31:16, and fraction is stored in bits 15:0. Default setting is 5.0 (327680 scaled).

5.9.5.TSI_AUDIO_GLITCHES_ALLOWED

```
TSI_AUDIO_GLITCHES_ALLOWED                       0x2024
unsigned int AudioMaxGlitches                    U32
4 bytes                                          RW
```

Description

Defines how many glitches are allowed before the audio test is considered failed.

5.10.HDMI Receiver Electrical tests

This section defines configuration items used with HDMI receiver electrical tests. Please refer to sections 4.2.3, 4.2.4, 4.2.5 and 4.2.6 for test details.

Quick reference table:

Define	Config ID	Default	Description	Reference
TSI_HDMI_RX_TIMEOUT	0x10200	5000	Electrical test timeout, (ms).	5.10.1
TSI_HDMI_RX_POWER_LOW_LIMIT	0x10201	4700	HDMI power-supply low limit, (mV).	5.10.2
TSI_HDMI_RX_POWER_HIGH_LIMIT	0x10202	5300	HDMI power-supply high limit, (mV).	5.10.3
TSI_HDMI_RX_LINK_LOW_LIMIT	0x10203	2900	HDMI Link voltage low limit, (mV).	5.10.4
TSI_HDMI_RX_LINK_HIGH_LIMIT	0x10204	3100	HDMI Link voltage high limit, (mV).	5.10.5
TSI_HDMI_RX_HPD_ZERO_LOW_LIMIT	0x10205	0	HDMI HPD logical zero voltage level low limit, (mV).	5.10.6
TSI_HDMI_RX_HPD_ZERO_HIGH_LIMIT	0x10206	400	HDMI HPD logical zero voltage level high limit, (mV).	5.10.7
TSI_HDMI_RX_HPD_ONE_LOW_LIMIT	0x10207	2400	HDMI HPD logical one voltage level low limit, (mV).	5.10.8
TSI_HDMI_RX_HPD_ONE_HIGHT_LIMIT	0x10208	5300	HDMI HPD logical one voltage level high limit, (mV).	5.10.9
TSI_HDMI_RX_DDC_LOW_LIMIT	0x10209	4500	HDMI DDC voltage level low limit, (mV).	5.10.10
TSI_HDMI_RX_DDC_HIGH_LIMIT	0x1020a	5500	HDMI DDC voltage level high limit, (mV).	5.10.11
TSI_HDMI_RX_CEC_ZERO_LOW_LIMIT	0x1020b	0	HDMI CEC Logical zero low voltage limit, (mV).	5.10.12
TSI_HDMI_RX_CEC_ZERO_HIGH_LIMIT	0x1020c	600	HDMI CEC logical zero hight voltage limit, (mV).	5.10.13
TSI_HDMI_RX_CEC_ONE_LOW_LIMIT	0x1020d	2500	HDMI CEC logical one low voltage limit, (mV).	5.10.14
TSI_HDMI_RX_CEC_ONE_HIGH_LIMIT	0x1020e	3600	HDMI CEC logical one high voltage limit, (mV).	5.10.15

5.10.1.TSI_HDMI_RX_TIMEOUT

```
TSI_HDMI_RX_TIMEOUT                              0x10200
unsigned int hdmi_rx_timeout                     U32
4 bytes                                          RW
```

Synopsis

Timeout period used for all HDMI RX electrical tests, in milliseconds. Default timeout is 5000ms.

5.10.2.TSI_HDMI_RX_POWER_LOW_LIMIT

```
TSI_HDMI_RX_POWER_LOW_LIMIT                    0x10201
unsigned int hdmi_rx_pwr_ll                    U32
4 bytes                                        RW
```

Synopsis

HDMI power line voltage low limit, in millivolts. The voltage detected from HDMI power line must be higher than this value in order to pass tests. Default setting is 4700mV.

5.10.3.TSI_HDMI_RX_POWER_HIGH_LIMIT

```
TSI_HDMI_RX_POWER_HIGH_LIMIT                   0x10202
unsigned int hdmi_rx_pwr_hl                    U32
4 bytes                                        RW
```

Synopsis

HDMI power line voltage high limit, in millivolts. The voltage detected from HDMI power line must be less than this value in order to pass tests. Default setting is 5300mV.

5.10.4.TSI_HDMI_RX_LINK_LOW_LIMIT

```
TSI_HDMI_RX_LINK_LOW_LIMIT                     0x10203
unsigned int hdmi_rx_lnk_ll                    U32
4 bytes                                        RW
```

Synopsis

HDMI link line voltage low limit, in millivolts. The voltage detected from HDMI link line(s) during test must be higher than this value in order to pass test. Default setting is 2900mV.

Important: *The acceptable setting for this value can be different for different types of DUT's. Proper calibration of this value will require testing multiple DUT's of same type in order to find typical value for the DUT in question.*

5.10.5.TSI_HDMI_RX_LINK_HIGH_LIMIT

```
TSI_HDMI_RX_LINK_HIGH_LIMIT                    0x10204
unsigned int hdmi_rx_lnk_hl                    U32
4 bytes                                        RW
```

Synopsis

HDMI link line voltage high limit, in millivolts. The voltage detected from HDMI link line(s) during test must be less than this value in order to pass test. Default setting is 3100mV.

Important: *The acceptable setting for this value can be different for different types of DUT's. Proper calibration of this value will require testing multiple DUT's of same type in order to find typical value for the DUT in question.*

5.10.6.TSI_HDMI_RX_HPD_ZERO_LOW_LIMIT

```
TSI_HDMI_RX_HPD_ZERO_LOW_LIMIT                 0x10205
unsigned int hdmi_rx_hpd_0_ll                  U32
4 bytes                                        RW
```

Synopsis

HDMI HPD logical zero voltage level, lower limit, in millivolts. When HPD line is expected to be in logical zero state, the measured voltage must be higher than this value in order to pass test. Default setting is 0mV.

5.10.7.TSI_HDMI_RX_HPD_ZERO_HIGH_LIMIT

```
TSI_HDMI_RX_HPD_ZERO_HIGH_LIMIT                0x10206
unsigned int hdmi_rx_hpd_0_hl                  U32
4 bytes                                        RW
```

Synopsis

HDMI HPD logical zero voltage level, higher limit, in millivolts. When HDP line is expected to be in logical zero state, the measured voltage must be lower than this value in order to pass test. Default setting is 400mV.

5.10.8.TSI_HDMI_RX_HPD_ONE_LOW_LIMIT

```
TSI_HDMI_RX_HPD_ONE_LOW_LIMIT                    0x10207
unsigned int hdmi_rx_hpd_1_ll                    U32
4 bytes                                          RW
```

Synopsis

HDMI HPD logical one voltage level, lower limit, in millivolts. When HPD line is expected to be in logical one state, the measured voltage must be less than this value in order to pass test. Default setting is 2400mV.

5.10.9.TSI_HDMI_RX_HPD_ONE_HIGH_LIMIT

```
TSI_HDMI_RX_HPD_ONE_HIGH_LIMIT                   0x10208
unsigned int hdmi_rx_hpd_1_hl                    U32
4 bytes                                          RW
```

Synopsis

HDMI HPD logical one voltage level, higher limit, in millivolts. When HPD line is expected to be in logical one state, the measured voltage must be less than this value in order to pass test. Default setting is 5300mV.

5.10.10.TSI_HDMI_RX_DDC_LOW_LIMIT

```
TSI_HDMI_RX_DDC_LOW_LIMIT                        0x10209
unsigned int hdmi_rx_ddc_ll                      U32
4 bytes                                          RW
```

Synopsis

DDC Line voltage low limit, in millivolts. Test will measure DDC line voltage when the line is not being driven low. The measured value must be higher than this value in order to pass test. Default setting is 4500mV.

5.10.11.TSI_HDMI_RX_DDC_HIGH_LIMIT

```
TSI_HDMI_RX_DDC_HIGH_LIMIT                       0x1020a
unsigned int hdmi_rx_ddc_hl                      U32
4 bytes                                          RW
```

Synopsis

DDC Line voltage high limit, in millivolts. Test will measure DDC line voltage when the line is not being driven low. The measured value must be lower than this value in order to pass test. Default setting is 5500mV.

5.10.12.TSI_HDMI_RX_CEC_ZERO_LOW_LIMIT

```
TSI_HDMI_RX_CEC_ZERO_LOW_LIMIT                0x1020b
unsigned int hdmi_rx_cec_0_ll                 U32
4 bytes                                       RW
```

Synopsis

CCE Line logical zero voltage level, lower limit, in millivolts. The CCE line voltage is measured when CCE line state is logical zero. The measured value must be higher than this value in order to pass test. Default setting is 0mV.

5.10.13.TSI_HDMI_RX_CEC_ZERO_HIGH_LIMIT

```
TSI_HDMI_RX_CEC_ZERO_HIGH_LIMIT               0x1020c
unsigned int hdmi_rx_cec_0_hl                 U32
4 bytes                                       RW
```

Synopsis

CCE Line logical zero voltage level, higher limit, in millivolts. The CCE line voltage is measured when CCE line state is logical zero. The measured value must be lower than this value in order to pass test. Default setting is 600mV.

5.10.14.TSI_HDMI_RX_CEC_ONE_LOW_LIMIT

```
TSI_HDMI_RX_CEC_ONE_LOW_LIMIT                 0x1020d
unsigned int hdmi_rx_cec_1_ll                 U32
4 bytes                                       RW
```

Synopsis

CCE Line logical one voltage level, lower limit, in millivolts. The CCE line voltage is measured when CCE line state is logical one. The measured value must be higher than this setting in order to pass test. Default setting is 2500mV.

5.10.15.TSI_HDMI_RX_CEC_ONE_HIGH_LIMIT

```
TSI_HDMI_RX_CEC_ONE_HIGH_LIMIT                0x1020e
unsigned int hdmi_rx_cec_1_hl                 U32
4 bytes                                       RW
```

Synopsis

CCE Line logical one voltage level, higher limit, in millivolts. The CCE line voltage is measured when CCE line state is logical one. The measured value must be lower that this setting in order to pass test. Default setting is 3600mV.

5.11.DP Receiver electrical tests

This section defines configuration items used with DP receiver electrical tests. Please refer to sections 4.2.8, 4.2.9 and 4.2.10 for test details.

Quick reference table:

Define	Config ID	Default	Description	Reference
TSI_DP_RX_TEST_TIMEOUT	0x10100	5000	Electrical test timeout (ms).	5.11.1
TSI_DP_RX_LINKS_LOW_VOLTAGE	0x10101	2600	Data links low voltage (mV).	5.11.2
TSI_DP_RX_LINKS_HI_VOLTAGE	0x10102	4000	Data links hi voltage (mV).	5.11.2
TSI_DP_RX_HPD_ZERO_LOW_VOLTAGE	0x10103	-100	HPD logical zero voltage level low limit (mV).	5.11.3
TSI_DP_RX_HPD_ZERO_HI_VOLTAGE	0x10104	800	HPD logical zero voltage level hi limit (mV).	5.11.3
TSI_DP_RX_HPD_ONE_LOW_VOLTAGE	0x10105	800	HPD logical one voltage level low limit (mV).	5.11.4
TSI_DP_RX_HPD_ONE_HI_VOLTAGE	0x10106	5500	HDP logical one voltage level hi limit (mV).	5.11.4
TSI_DP_RX_AUX_P_IDLE_LOW_VOLTAGE	0x10107	20	AUX P-Line idle state low voltage limit (mV).	5.11.5
TSI_DP_RX_AUX_P_IDLE_HI_VOLTAGE	0x10108	500	AUX P-Line idle state hi voltage limit (mV).	5.11.5
TSI_DP_RX_AUX_N_IDLE_LOW_VOLTAGE	0x10109	2600	AUX N-Line idle state low voltage limit (mV).	5.11.6
TSI_DP_RX_AUX_N_IDLE_HI_VOLTAGE	0x1010a	3600	AUX N-Line idel state hi voltage limit (mV).	5.11.6
TSI_DP_RX_AUX_P_TRIG_VOLTAGE	0x1010b	150	AUX P-Line trigger voltage level (mV).	5.11.7
TSI_DP_RX_AUX_N_TRIG_VOLTAGE	0x1010c	200	AUX N-Line trigger voltage level (mV).	5.11.7
TSI_DP_RX_AUX_SIGNAL_CAPT_TIMEOUT	0x1010d	200	AUX Signal capture timeout (ms)	5.11.8
TSI_DP_RX_AUX_SIGNAL_CAPT_TRIES	0x1010e	5	AUX Signal capture retries.	5.11.9
TSI_DP_RX_MAX_DUT_LANE_COUNT	0x1010f	4	DUT Max. lanes.	5.11.10
TSI_DP_RX_MAX_DUT_LINK_RATE	0x10110	20	DUT Max. lane frequency as multiplier of 0.27Gbps	5.11.11

5.11.1.TSI_DP_RX_TEST_TIMEOUT

```
TSI_DP_RX_TEST_TIMEOUT                          0x10100
unsigned int dp_rx_test_timeout                 U32
4 bytes                                         RW
```

Synopsis

Timeout period used for all DP RX electrical tests, in milliseconds. Default timeout is 5000ms.

5.11.2.TSI_DP_RX_LINKS_*_VOLTAGE

```
TSI_DP_RX_LINKS_LOW_VOLTAGE                    0x10101
TSI_DP_RX_LINKS_HI_VOLTAGE                     0x10102
unsigned int dp_rx_link_ll, dp_rx_link_hl      U32
4 bytes                                        RW
```

Synopsis

These two CI's define the acceptable voltage range DP link lines. The measured voltage must be higher than TSI_DP_RX_LINKS_LOW_VOLTAGE setting, and lower than TSI_DP_RX_LINKS_HI_VOLTAGE setting in order to pass test. Default setting for low voltage limit is 2600mV, and for high voltage limit 4000mV.

5.11.3.TSI_DP_RX_HPD_ZERO_*_VOLTAGE

```
TSI_DP_RX_HPD_ZERO_LOW_VOLTAGE                 0x10103
TSI_DP_RX_HDP_ZERO_HI_VOLTAGE                  0x10104
unsigned int dp_rx_hpd_0_ll, dp_rx_hpd_0_hl    U32
4 bytes                                        RW
```

Synopsis

These to CI's define the acceptable voltage range for HDP line when it is in logical zero state. The measured voltage must be higher than TSI_DP_RX_HDP_ZERO_LOW_VOLTAGE setting, and lower than TSI_DP_RX_HPD_ZERO_HI_VOLTAGE setting in order to pass test. Default setting for low voltage limit is -100mV, and for high voltage limit 800mV.

5.11.4.TSI_DP_RX_HDP_ONE_*_VOLTAGE

```
TSI_DP_RX_HDP_ONE_LOW_VOLTAGE                  0x10105
TSI_DP_RX_HDP_ONE_HI_VOLTAGE                   0x10106
unsigned int dp_rx_hpd_1_ll, dp_rx_hpd_1_hl    U32
4 bytes                                        RW
```

Synopsis

These two CI's define the acceptable voltage range for HPD line when it is in logical one state. The measured voltage must be higher than TSI_DP_RX_HDP_ONE_LOW_VOLTAGE setting, and lower than TSI_DP_RX_HDP_ONE_HI_VOLTAGE setting in order to pass test. Default setting for low voltage limit is 800mV, and for high voltage limit 5500mV.

5.11.5.TSI_DP_RX_AUX_P_IDLE_*_VOLTAGE

```
TSI_DP_RX_AUX_P_IDLE_LOW_VOLTAGE              0x10107
TSI_DP_RX_AUX_P_IDLE_HI_VOLTAGE               0x10108
unsigned int dp_rx_aux_p_ll, dp_rx_aux_p_hl   U32
4 bytes                                       RW
```

Synopsis

These two CI's define the acceptable AUX+ line idle voltage range when the AUX is idle. The measured voltage must be higher than TSI_DP_RX_AUX_P_IDLE_LOW_VOLTAGE setting, and lower than TSI_DP_RX_AUX_P_IDLE_HI_VOLTAGE setting in order to pass test. Default setting for low voltage limit is 500mV, and for high voltage limit 2600mV.

5.11.6.TSI_DP_RX_AUX_N_IDLE_*_VOLTAGE

```
TSI_DP_RX_AUX_N_IDLE_LOW_VOLTAGE              0x10109
TSI_DP_RX_AUX_N_IDLE_HI_VOLTAGE               0x1010a
unsigned int dp_rx_aux_n_ll, dp_rx_aux_n_hl   U32
4 bytes                                       RW
```

Synopsis

These two CI's defined the acceptable AUX- line idle voltage range when the AUX is idle. The measured voltage must be higher than TSI_DP_RX_AUX_N_IDLE_LOW_VOLTAGE setting, and lower than TSI_DP_RX_AUX_N_IDLE_HI_VOLTAGE setting in order to pass test. Default setting for low voltage limit is 2600mV, and for high voltage limit 3600mV.

5.11.7.TSI_DP_RX_AUX_*_TRIG_VOLTAGE

```
TSI_DP_RX_AUX_P_TRIG_VOLTAGE                  0x1010b
TSI_DP_RX_AUX_N_TRIG_VOLTAGE                  0x1010c
unsigned int dp_rx_aux_ptrig, dp_rx_aux_ntrig U32
4 bytes                                       RW
```

Synopsis

These two CI's define the AUX+ (TSI_DP_RX_AUX_P_TRIG_VOLTAGE) and AUX- (TSI_DP_RX_AUX_N_TRIG_VOLTAGE) line state change trigger levels. Default settings are for AUX+ 150mV and for AUX- 200mV.

5.11.8.TSI_DP_RX_AUX_SIGNAL_CAPT_TIMEOUT

```
TSI_DP_RX_AUX_SIGNAL_CAPT_TIMEOUT                0x1010d
unsigned int dp_rx_aux_cap_timeout               U32
4 bytes                                          RW
```

Synopsis

Timeout for AUX signal capture, in milliseconds. When the TE generates a HPD pulse during test, it waits for this amount of time (max.) for DUT to read DPCD locations 0x200 to 0x205. If this transaction is not seen, the test will fail. Default setting is 200ms.

5.11.9.TSI_DP_RX_AUX_SIGNAL_CAPT_TRIES

```
TSI_DP_RX_AUX_SIGNAL_CAPT_TRIES                  0x1010e
unsigned int dp_rx_aux_cap_retries               U32
4 bytes                                          RW
```

Synopsis

Retry count for AUX signal capture. If the AUX signal capture after TE generated a HPD pulse fails, the TE will re-try this many times. Default setting is 5.

5.11.10.TSI_DP_RX_MAX_DUT_LANE_COUNT

```
TSI_DP_RX_MAX_DUT_LANE_COUNT                     0x1010f
unsigned int dp_rx_max_dut_lanes                 U32
4 bytes                                          RW
```

Synopsis

Maximum number of lanes supported by the connected DUT. Typical values are 1, 2 or 4. Default setting is 4.

5.11.11.TSI_DP_RX_MAX_DUT_LINK_RATE

```
TSI_DP_RX_MAX_DUT_LINK_RATE                      0x10110
unsigned int dp_rx_max_dut_link_rate             U32
4 bytes                                          RW
```

Synopsis

Maximum link rate supported by the connected DUT, as multiplier of 0.27Gbps. Typical values are 6 (RBR), 10 (HBR), 20 (HBR-2) or 30 (HBR-3). Please note that HBR3 speed is not supported on all TE devices.

5.12.Accessing Info frames

This section defines configuration items for accessing Info Frames.

5.12.1.TSI_R_HDMI_INFOFRAME_RANGE_*

```
TSI_R_HDMI_INFOFRAME_RANGE_START                0x11000
TSI_R_HMDI_INFOFRAME_RANGE_END                  0x110ff
unsigned char data[]                            ARRAY_U8
Variable size                                   RO
```

Synopsis

The HDMI standard allows for maximum of 256 different info-frames. The CI Space starting at 0x11000 has one CI for each possible info-frame. All CI's within the range 0x11000 to 0x110ff are read-only. To read a specific info-frame, add it's ID value to 0x11000 and read that CI. Each CI in this area has dynamic size, and no validity checks are performed on the received info frames. When attempting to read a specific info-frame, please note that if that info frame is not received (ever), the read may fail with error code -32 (No data available).

Some known info-frames have specific CI definitions available for convenience. Please refer to 5.12.3 Additional Info-frame CI definitions, and update bits for info frame CI names and definitions.

5.12.2.TSI_R_HDMI_INFOFRAME_UPDATE_FLAGS

```
TSI_R_HDMI_INFOFRAME_UPDATE_FLAGS               0x11100
unsigned char Flags[32]                         ARRAY_U8
32 bytes                                        CRO
```

Synopsis

Info-frame updated flags. When an info-frame is received, the bit corresponding to it's raw ID is set – For example, for AVI info-frame (ID = 0x82), the bit 0x82 is set.

The data accessible from the CI is a 256-bit flags field stored as a little-endian 256-bit word. The flags are cleared on read. Please refer to 5.12.3 Additional Info-frame CI definitions, and update bits for info frame CI names and definitions.

5.12.3.Additional Info-frame CI definitions, and update bits

Define	Config ID	Update Bit #	Description
TSI_R_HDMI_INFOFRAME_NULL	0x11000	0	***READ ONLY.*** NULL infoframe.
TSI_R_HDMI_INFOFRAME_ACR	0x11001	1	***READ ONLY.*** Audio Clock Regeneration.
TSI_R_HDMI_INFOFRAME_ASP	0x11002	2	***READ ONLY.*** Audio Sample Packet
TSI_R_HDMI_INFOFRAME_GCP	0x11003	3	***READ ONLY.*** General Control Packet
TSI_R_HDMI_INFOFRAME_ACP	0x11004	4	***READ ONLY.*** Audio Content Protection packet
TSI_R_HDMI_INFOFRAME_ISRC1	0x11005	5	***READ ONLY.*** International Standard Recording Code
TSI_R_HDMI_INFOFRAME_ISRC2	0x11006	6	***READ ONLY.*** International Standard Recording Code
TSI_R_HDMI_INFOFRAME_OBA	0x11007	7	***READ ONLY.*** One Bit Audio sample packet
TSI_R_HDMI_INFOFRAME_DTS	0x11008	8	***READ ONLY.*** DTS audio packet
TSI_R_HDMI_INFOFRAME_HBR	0x11009	9	***READ ONLY.*** High BitRate audio stream packet
TSI_R_HDMI_INFOFRAME_GMP	0x1100a	10	***READ ONLY.*** Gamut Metadata Packet
TSI_R_HDMI_INFOFRAME_VSI	0x11081	129	***READ ONLY.*** Vendor Specific Infoframe
TSI_R_HDMI_INFOFRAME_AVI	0x11082	130	***READ ONLY.*** Auxiliary Video Information infoframe
TSI_R_HDMI_INFOFRAME_SPD	0x11083	131	***READ ONLY.*** Source Product Descriptor infoframe
TSI_R_HDMI_INFOFRAME_AIF	0x11084	132	***READ ONLY.*** Audio Infoframe
TSI_R_HDMI_INFOFRAME_MPEG	0x11085	133	***READ ONLY.*** MPEG Source infoframe
TSI_R_HDMI_INFOFRAME_3D_ASP	0x1100b	11	***READ ONLY.*** 3D Audio Sample Packet
TSI_R_HDMI_INFOFRAME_3D_OBA	0x1100c	12	***READ ONLY.*** 3D One Bit Audio sample packet
TSI_R_HDMI_INFOFRAME_AMP	0x1100d	13	***READ ONLY.*** Audio Metadata Packet
TSI_R_HDMI_INFOFRAME_MST_ASP	0x1100e	14	***READ ONLY.*** Multi-stream Audio Sample Packet
TSI_R_HDMI_INFOFRAME_MST_OBA	0x1100f	15	***READ ONLY.*** Multi-stream One Bit Audio sample packet
TSI_R_HDMI_INFOFRAME_DRM	0x11087	135	***READ ONLY.*** Dynamic Range and Mastering infoframe

5.13.Miscellaneous

This section defines miscellaneous configuration items that do not fit any of the other "categories".

Define	Config ID	Default	Description	Reference
TSI_R_GENERIC_STATUS	0x210	N/A	Report generic device status as status bits.	5.13.1
TSI_R_UNITS_PRESENT	0x211	N/A	Number of units chained.	5.13.2
TSI_W_FORCE_HOT_PLUG_STATE	0x212	N/A	Controls HPD state: 0 = low, 1 = High.	5.13.3
TSI_EDID_TE_INPUT (TSI_CURRENT_SINK_EDID)	0x1100	N/A	Access TE side EDID.	5.13.4
TSI_EDID_TE_OUTPUT	0x1101	N/A	Access DUT side EDID over signal cable.	5.13.5
TSI_VERSION_TEXT	0x80000001	N/A	TSI Version information as text.	5.13.6
TSI_LOG_FILE	0x80000002	N/A	Log file-name for recording logs easily.	5.13.7
TSI_INVALID_CONFIG_ITEM	0xffffffff	N/A	Reserved for invalid CI indication. Do not use.	N/A
TSI_HPD_LENGTH	0x1201	1000	HPD Pulse length for TSI generated pulses.	5.13.8
TSI_W_ARC_CONTROL	0x1210	0	Audio Return Channel control.	5.13.9

5.13.1.TSI_R_GENERIC_STATUS

```
TSI_R_GENERIC_STATUS                          0x210
unsigned int generic_status_bits              U32
4 bytes                                       RO
```

Synopsis

Reserved for generic device status bits. Currently this CI is not implemented.

5.13.2.TSI_R_UNITS_PRESENT

```
TSI_R_UNITS_PRESENT                           0x211
unsigned int chain_count                      U32
4 bytes                                       RO
```

Synopsis

Indicates number of chained capture devices including the master device. Please notice that capture device chaining only works with signal types that are scalable by adding more lanes of same type, such as LVDS and V-by-One.

Important: *This CI is implemented only for devices that support chaining.*

5.13.3.TSI_W_FORCE_HOT_PLUG_STATE

```
TSI_W_FORCE_HOT_PLUG_STATE                    0x212
unsigned int force_hpd                        U32
4 bytes                                       WO
```

Synopsis

Forces the HPD to either asserted or deasserted state. 0 = force deasserted state, 1 = force asserted state.

Important: For interfaces that do not support HPD this CI is either not implemented, or has no effect.

5.13.4.TSI_EDID_TE_INPUT

```
TSI_EDID_TE_INPUT                             0x1100
(TSI_CURRENT_SINK_EDID                        0x1100)
unsigned char edid_data[]                     ARRAY_U8
Variable size                                 RW
```

Synopsis

Allows access to TE side EDID block(s). For reads, use at least 512 bytes buffer. Write sizes are checked by the TE firmware. Typical requirement for size is multiple of 128 bytes.

5.13.5.TSI_EDID_TE_OUTPUT

```
TSI_EDID_TE_OUTPUT                            0x1101
unsigned char edid_data[]                     ARRAY_U8
Variable size                                 RW
```

Synopsis

Allows access to DUT side EDID block(s) over connecting signal cable. Read capability is always available, while write capability depends on connected DUT device. For reading, use at least 512 bytes buffer. Write sizes depend on DUT and are not checked by TSI. Typical requirement for size is multiple of 128 bytes.

5.13.6.TSI_VERSION_TEXT

```
TSI_VERSION_TEXT                                    0x80000001
char version_string[]                               ARRAY_S8
Variable size                                       RO
```

Synopsis

Reading returns a NULL terminated ASCII text string containing TSI version information, loaded lower-level API's and their versions, and devices available to TSI through them.

Important: *Firmware versions are typically not available as reading this information requires opening the device.*

5.13.7.TSI_LOG_FILE

```
TSI_LOG_FILE                                        0x800000002
char log_filename[]                                 ARRAY_S8
Variable size                                       WO
```

Synopsis

Log filename for easy recording of status log messages. Default setting is "Empty" (No log recorded). To start recording, set a valid filename as NULL terminated ASCII text string into this CI. TSI will automatically record all status log messages into this file. To stop recording, use NULL as the source string pointer.

```
//Start recording:
char *MyLogFile = "C:\\Temp\\MyLog.txt";
TSI_RESULT Result;
Result = TSI_TS_SetConfigItem(TSI_LOG_FILE, MyLogFile, strlen(MyLogFile));
if(Result < TSI_SUCCESS) { /* Handle error */

/* Your actions that need to be logged */

// Stop recording log:
Result = TSI_TS_SetConfigItem(TSI_LOG_FILE, NULL, 0);
if(Result < TSI_SUCCESS) { /* Handle error */
```

5.13.8.TSI_HPD_LENGTH

```
TSI_HPD_LENGTH                                      0x1201
unsigned int hpd_length                             U32
4 bytes                                             WO
```

Synopsis

Length of TSI generated HPD pulses, in milliseconds. Default setting is 1000ms. Set this to zero to disable HPD pulse generation in TSI.

Important: *Even if the HPD pulse generation is disabled, the TE device's firmware may still generate HPD pulses on certain situations.*

5.13.9.TSI_W_ARC_CONTROL

```
TSI_W_ARC_CONTROL                                   0x1210
unsigned int arc_control                            U32
4 bytes                                             WO
```

Synopsis

Controls Audio Return Channel on TE side HDMI inputs that support it. Valid settings are: 0 (=Disabled), 1 (=Generate audio) and 2 (=Loop-back audio from DUT). Default setting is 0 (Disabled).

Important: *Some devices might not support loop-back mode. In these cases writing 2 will fail with invalid configuration item value error (error code -39).*

Important: *Audio streams generated by the device vary by the device model.*

5.14.CRC based video tests

This section defines configuration items used with CRC based video tests.

Define	Config ID	Default	Description	Reference
TSI_CRC_TIMEOUT	0x10300	1000	CRC Test max. run time, in milliseconds.	5.14.1
TSI_CRC_FRAMES_TO_TEST	0x10301	20	Number of input frames to be tested.	5.14.2
TSI_CRC_LIM_FRAME_MISMATCHES	0x10302	0	Number of frames that are allowed to have mismatching CRC.	5.14.3
TSI_CRC_REF_WIDTH	0x10303	1920	Required Input video width in pixels.	5.14.4
TSI_CRC_REF_HEIGHT	0x10304	1080	Required Input video height in pixels.	5.14.5
TSI_CRC_REF_COLORDEPTH	0x10305	24	Required bits per pixel on input video.	5.14.6
TSI_CRC_REFERENCE_CRC_VALUES	0x10306	-	Block of memory containing max. 65535 CRC value sets.	5.14.7
TSI_CRC_REQUIRED_FRAME_RATE	0x10307	0	Required input video frame rate, in millihertz (mHz).	5.14.8
TSI_CRC_FRAME_RATE_TOLERANCE	0x10308	0	Allowed deviation from require frame rate, in millihertz (mHz).	5.14.9
TSI_CRC_MOTION_TEST_ITERATIONS	0x10309	1	Number of iterations the defined CRC sequencey must be found.	5.14.10
TSI_CRC_COLOR_FORMAT	0x1030a	0	Color format. 0 = RGB. Other values invalid	5.14.11

5.14.1.TSI_CRC_TIMEOUT

```
TSI_CRC_TIMEOUT                                 0x10300
unsigned int crc_test_timeout                   U32
4 bytes                                         RW
```

Synopsis

Defines timeout for all CRC based video tests, in milliseconds. Default setting is 1000ms.

5.14.2.TSI_CRC_FRAMES_TO_TEST

```
TSI_CRC_FRAMES_TO_TEST                          0x10301
unsigned int crc_frames_to_test                 U32
4 bytes                                         RW
```

Synopsis

Defines timeout for all CRC based video tests, as number of frames. This setting, and the TSI_CRC_TIMEOUT together define the length of the test: The limit that is reached first applies. Set this value to zero to disable frame count limint.

5.14.3.TSI_CRC_LIM_FRAME_MISMATCHES

```
TSI_CRC_LIM_FRAME_MISMATCHES                    0x10302
unsigned int crc_mismatches_allowed             U32
4 bytes                                         RW
```

Synopsis

Defines number of frames that are allowed to fail without causing the test result to be “failed”. Default setting is 0.

5.14.4.TSI_CRC_REF_WIDTH

```
TSI_CRC_REF_WIDTH                               0x10303
unsigned int crc_ref_w                          U32
4 bytes                                         RW
```

Synopsis

Defined the expected video width, in pixels. If the video being received does not match this setting, the test will fail. Default setting is 1920.

5.14.5.TSI_CRC_REF_HEIGHT

```
TSI_CRC_REF_HEIGHT                              0x10304
unsigned int crc_ref_h                          U32
4 bytes                                         RW
```

Synopsis

Defines the expected video height, in pixels. If the video being received does not match this setting, the test will fail. Default setting is 1080.

5.14.6.TSI_CRC_REF_COLORDEPTH

```
TSI_CRC_REF_COLORDEPTH                          0x10305
unsigned int crc_ref_colordepth                 U32
4 bytes                                         RW
```

Synopsis

Defines the color depth as bits per pixel. If the input video color depth does not match this setting, the test will fail. Default setting is 24.

5.14.7.TSI_CRC_REFERENCE_CRC_VALUES

```
TSI_CRC_REFERENCE_CRC_VALUES                    0x10306
unsigned short CRC_Values[]                     ARRAY_U16
Variable size                                   RW
```

Synopsis

Contains CRC reference values. Each CRC set consists of 3 16-bit words; One word for each color channel. Red / Cr color channel CRC is at the lowest address (first word), followed by Green / Y channel (second word) and then Blue / Cb channel (third word). Maximum number of CRC value sets is 65535. Default CRC set is empty (=no default value).

5.14.8.TSI_CRC_REQUIRED_FRAME_RATE

```
TSI_CRC_REQUIRED_FRAME_RATE                     0x10307
unsigned int crc_req_frate                      U32
4 bytes                                         RW
```

Synopsis

Defines the required frame rate for CRC based tests, in millihertz. Setting of zero (0) disables the frame-rate requirement. Default setting is 0.

5.14.9.TSI_CRC_FRAME_RATE_TOLERANCE

```
TSI_CRC_FRAME_RATE_TOLERANCE                    0x10308
unsigned int crc_frate_tolerance                U32
4 bytes                                         RW
```

Synopsis

Defines the maximum allowed deviation of input frame-rate from the required frame rate (TSI_CRC_REQUIRED_FRAME_RATE), in millihertz. When this setting is non-zero, it defines the range of allowed frame rate as requirements ± tolerance. If the frame-rate requirement is set to zero, this setting has no effect. Default setting is 50 mHz.

5.14.10.TSI_CRC_MOTION_TEST_ITERATIONS

```
TSI_CRC_MOTION_TEST_ITERATION                    0x10309
unsigned int crc_motion_test_iters               U32
4 bytes                                          RW
```

Synopsis

Defines the number of iterations the defined CRC sequnce must be found in order to pass the test. Default is 1.

5.14.11.TSI_CRC_COLOR_FORMAT

```
TSI_CRC_COLOR_FORMAT                             0x1030a
unsinged int crc_color_format                    U32
4 bytes                                          RW
```

Synopsis

Defines the color format of the expected video input. Default value is 0 (=RGB).

Important: *This configuration item is reserved for future support of additional color spaces. Currently, it must be set to zero.*

5.15.DP RefSource simple link test

This section defines configuration items used with DP Reference source simple link test.

Define	Config ID	Default	Description	Reference
TSI_DP_LTT_TIMEOUT	0x10700	5000	Timeout for each test loop iteration, in milliseconds.	5.15.1
TSI_DP_LTT_MAX_LANE_COUNT	0x10701	4	Max. number of lanes to be tested. Valid settings are 1, 2 and 4	5.15.2
TSI_DP_LTT_MAX_RATE	0x10702	20	Max. link rate to be tested as multiple of 0.27 Gbps. Valid settings are 6, 10 and 20.	5.15.3
TSI_DP_LTT_HPD_PULSE_DURATION	0x10705	1000	HPD pulse duration used to trigger link-training, in milliseconds.	5.15.4
TSI_DP_LTT_LT_START_TIMEOUT	0x10706	5000	Timeout for link training start after HPD pulse, in milliseconds.	5.15.5
TSI_DP_LTT_TEST_LOOP_DELAY	0x10707	3000	Delay time after link training before the next next test loop iteration, in milliseconds.	5.15.6

5.15.1.TSI_DP_LTT_TIMEOUT

```
TSI_DP_LTT_TIMEOUT                                  0x10700
unsigned int ltt_timeout                            U32
4 bytes                                             RW
```

Synopsis

Defines timeout for each test iteration, in milliseconds. The test iterates through a number of iterations depending on other tests. Each iteration must complete within this timeout in order for the test succeed. Default setting is 5000ms.

5.15.2.TSI_DP_LTT_MAX_LANE_COUNT

```
TSI_DP_LTT_MAX_LANE_COUNT                           0x10701
unsigned int ltt_max_lanes                          U32
4 bytes                                             RW
```

Synopsis

Defines the maximum number of lanes to be tested. Valid settings are 1, 2 and 4. Default setting is 4.

5.15.3.TSI_DP_LTT_MAX_RATE

```
TSI_DP_LTT_MAX_RATE                          0x10702
unsigned int ltt_max_rate                    U32
4 bytes                                      RW
```

Synopsis

Defines the maximum link rate to be tested. The setting is in multiplier of 0.27Gbps. Valid settings are 6, 10 and 20. Default setting is 20.

5.15.4.TSI_DP_LTT_HPD_PULSE_DURATION

```
TSI_DP_LTT_HPD_PULSE_DURATION                0x10705
unsigned int ltt_hpd_duration                U32
4 bytes                                      RW
```

Synopsis

Defines the length of the HPD pulse used to start each test iteration, in milliseconds. Default setting is 1000ms.

5.15.5.TSI_DP_LTT_LT_START_TIMEOUT

```
TSI_DP_LTT_LT_START_TIMEOUT                  0x10706
unsigned int ltt_lt_start_timeout            U32
4 bytes                                      RW
```

Synopsis

Defines how long the test waits for LT start after issuing HPD pulse, in milliseconds. Default setting is 5000ms.

5.15.6.TSI_DP_LTT_TEST_LOOP_DELAY

```
TSI_DP_LTT_TEST_LOOP_DELAY                   0x10707
unsigned int ltt_loop_delay                  U32
4 bytes                                      RW
```

Synopsis

Defines the additional delay inserted in between test iterations, in milliseconds. Default setting is 3000ms.

5.16.HDCP Debugging configuration items

The HDCP debugging is divided into two groups of configuration items, and the 0x280 – 0x2ff configuration item ID range is reserved for the HDCP debug system.

In order to access HDCP Debugging features, a *TSI Advanced license* is required. In addition, accessing HDCP 2.X debugging features, an *TSI Advanced + HDCP 2.2 license* is required.

First group is for HDCP 1.x, ranging from CI ID 0x280 to 0x28f. Unlisted CI's are reserved for future additions. The new CI's for this group are listed below:

Name	Description	Reference
TSI_R_HDCP_1X_STATUS	HDCP 1.x status indicator bits.	5.16.1
TSI_W_HDCP_1X_COMMAND	HDCP 1.x related commands.	5.16.2

The second CI group is for HDCP 2.x, ranging from 0x290 to 0x29f. Unlisted CI's are reserved for future additions. The new CI's are listed below:

TSI_R_HDCP_2X_STATUS	HDCP 2.x status indicator bits.	5.16.3
TSI_W_HDCP_2X_COMMAND	HDCP 2.x related commands.	5.16.4

In addition, a single other general purpose CI is added (0x212) to allow client applications to generate HPD pulses.

Name	Description	Reference
TSI_W_FORCE_HOT_PLUG_STATE	HPD control of current input	5.16.5

5.16.1.TSI_R_HDCP_1X_STATUS

```
TSI_R_HDCP_1X_STATUS                          0x280
unsigned int hdcp_1x_status;                  U32
4 bytes                                       RO
```

Description

Current status indication of the HDCP 1.x. See table below for defined status bits:

Bits	Source / Sink	Value	Description
0	Sink	0	HDCP 1.x Encryption not active.
		1	HDCP 1.x Encryption is active.
2:1	Sink	0	HDCP 1.x Keys not loaded.
		1	HDCP 1.x Test keys loaded.
		2	HDCP 1.x production keys loaded.
		3	RESERVED
3	Sink	0	Current HDCP 1.x indication “not supported” by device.
		1	Current HDCP 1.x indication “supported” by device.
4	Sink	0	HDCP 1.x Link not authenticated.
		1	HDCP 1.x Link authenticated.
7:5		0	RESERVED
8	Source	0	HDCP 1.x Encryption not active.
		1	HDCP 1.x Encryption is active.
10:9	Source	0	HDCP 1.x Keys not loaded.
		1	HDCP 1.x Test keys loaded.
		2	HDCP 1.x production keys loaded.
		3	RESERVED
11	Source	0	Authentication not in progress
		1	Authentication in progress
12	Source	0	Not Authenticated
		1	Authenticated
15:13		0	RESERVED

(Continued...)

(...Continued)

Bits	Source / Sink	Value	Description
16	Sink	0	HDCP 1.x support is not available on the hardware.
		1	HDCP 1.x support is available on the hardware
17	Sink	0	HDCP 1.x production use keys are not available
		1	HDCP 1.x production use keys are available with command ID 2
18	Sink	0	HDCP 1.x test keys are not available
		1	HDCP 1.x test keys are available
23:19		0	RESERVED
24	Source	0	HDCP 1.x support is not available on the hardware.
		1	HDCP 1.x support is available on the hardware
25	Source	0	HDCP 1.x production use keys are not available
		1	HDCP 1.x production use keys are available with command ID 2
26	Source	0	HDCP 1.x test keys are not available
		1	HDCP 1.x test keys are available
31:27		0	RESERVED

5.16.2.TSI_W_HDCP_1X_COMMAND

```
TSI_W_HDCP_1X_COMMAND                          0x281
unsigned int hdcp_1x_command;                  U32
4 bytes                                        WO
```

Description

Write command ID's to issue commands to the HDCP 1.x debug system. Available commands are listed below. TSI Version 1.9 [R11] adds support for Source side HDCP controls. As a result of this addition, the following defines are to be considered obsolete:

- H1_LOAD_TEST_KEYS (→ H1_SINK_LOAD_TEST_KEYS)
- H1_LOAD_PROD_KEYS (→ H1_SINK_LOAD_PROD_KEYS)
- H1_UNLOAD_KEYS (→ H1_SINK_UNLOAD_KEYS)
- H1_SET_CAPABLE (→ H1_SINK_SET_CAPABLE)
- H1_CLEAR_CAPABLE (→ H1_SINK_CLEAR_CAPABLE)

Important: *The obsolete definitions are still available for backward compatibility. Existing application can continue to use them, but new applications should use the SINK specific command definitions.*

Command	ID	Sink / Source	Description
H1_SINK_LOAD_TEST_KEYS	0x001	SINK	Load “facsimile” test keys.
H1_SINK_LOAD_PROD_KEYS	0x002	SINK	Load production keys.
H1_SINK_UNLOAD_KEYS	0x003	SINK	Unload HDCP keys.
H1_SINK_SET_CAPABLE	0x004	SINK	Set device to indicate HDCP 1.x supported.
H1_SINK_CLEAR_CAPABLE	0x005	SINK	Set device to indicate HDCP 1.x not supported.
H1_SOURCE_LOAD_TEST_KEYS	0x101	SOURCE	Load “facsimile” test keys.
H1_SOURCE_LOAD_PROD_KEYS	0x102	SOURCE	Load production keys.
H1_SOURCE_UNLOAD_KEYS	0x103	SOURCE	Unload HDCP keys.
H1_SOURCE_AUTHENTICATE	0x104	SOURCE	Authenticate link to DUT
H1_SOURCE_DE_AUTHENTICATE	0x105	SOURCE	Mark link as de-authenticated.
H1_SOURCE_ENABLE_ENCRYPT	0x106	SOURCE	Enable encryption.
H1_SOURCE_DISABLE_ENCRYPT	0x107	SOURCE	Disable encryption.

Important: *Unlisted command ID values are reserved and should not be used.*

5.16.3.TSI_R_HDCP_2X_STATUS

```
TSI_R_HDCP_2X_STATUS                                0x290
unsigned int hdcp_2x_status;                        U32
4 bytes                                             RO
```

Description

Current status indication of the HDCP 2.x. See table below for defined status bits:

Bits	Source / Sink	Value	Description
0	Sink	0	HDCP 2.x Encryption not active.
		1	HDCP 2.x Encryption is active.
2:1	Sink	0	HDCP 2.x Keys not loaded.
		1	RESERVED
		2	HDCP 2.x production keys loaded.
		3	RESERVED
3	Sink	0	Current HDCP 2.x indication “not supported” by device.
		1	Current HDCP 2.x indication “supported” by device.
4	Sink	0	HDCP 2.x Link not authenticated.
		1	HDCP 2.x Link authenticated.
7:5		0	RESERVED
8	Source	0	HDCP 2.x Encryption not active.
		1	HDCP 2.x Encryption is active.
10:9	Source	0	HDCP 2.x Keys not loaded.
		1	RESERVED
		2	HDCP 2.x production keys loaded.
		3	RESERVED
11	Source	0	Authentication not in progress
		1	Authentication in progress
12	Source	0	Not authenticated
		1	Authenticated
13	Source	0	Km storage not supported
		1	Km storage supported
15:4		0	RESERVED
16	Sink	0	HDCP 2.x support is not available on the hardware.
		1	HDCP 2.x support is available on the hardware
17	Sink	0	HDCP 2.x production use keys are not available
		1	HDCP 2.x production use keys are available with command ID 2
31:19		0	RESERVED.

(Continued...)

(...Continued)

Bits	Source / Sink	Value	Description
16	Sink	0	HDCP 2.x support is not available on the hardware.
		1	HDCP 2.x support is available on the hardware
17	Sink	0	HDCP 2.x production use keys are not available
		1	HDCP 2.x production use keys are available with command ID 2
23:19		0	RESERVED.
24	Source	0	HDCP 2.x support not available.
		1	HDCP 2.x support available.
25	Source	0	HDCP 2.x keys not present.
		1	HDCP 2.x keys present.
26	Source	0	HDCP 2.x Test keys not present.
		1	HDCP 2.x Test keys present.
27	Source	0	Store Km support not available.
		1	Store Km support available
31:28		0	RESERVED

5.16.4.TSI_W_HDCP_2X_COMMAND

```
TSI_W_HDCP_2X_COMMAND                          0x291
unsigned int hdcp_2x_command;                  U32
4 bytes                                        WO
```

Description

Write command ID's to issue commands to the HDCP 2.x debug system. TSI Version 1.9 [R11] adds support for Source side HDCP controls. As a result of this addition, the following defines are to be considered obsolete:

- H2_LOAD_PROD_KEYS (→ H2_SINK_LOAD_PROD_KEYS)
- H2_UNLOAD_KEYS (→ H2_SINK_UNLOAD_KEYS)
- H2_SET_CAPABLE (→ H2_SINK_SET_CAPABLE)
- H2_CLEAR_CAPABLE (→ H2_SINK_CLEAR_CAPABLE)

Important: *The obsolete definitions are still available for backward compatibility. Existing application can continue to use them, but new applications should use the SINK specific command definitions.*

Command	ID	Sink / Source	Description
H2_SINK_LOAD_PROD_KEYS	0x002	SINK	Load production keys.
H2_SINK_UNLOAD_KEYS	0x003	SINK	Unload HDCP keys.
H2_SINK_SET_CAPABLE	0x004	SINK	Set device to indicate HDCP 2.x supported.
H2_SINK_CLEAR_CAPABLE	0x005	SINK	Set device to indicate HDCP 2.x not supported.
H2_SOURCE_LOAD_PROD_KEYS	0x102	SOURCE	Load production keys
H2_SOURCE_UNLOAD_KEYS	0x103	SOURCE	Unload keys.
H2_SOURCE_AUTHENTICATE	0x104	SOURCE	Authenticate link to DUT
H2_SOURCE_DE_AUTHENTICATE	0x105	SOURCE	Mark link as de-authenticated.
H2_SOURCE_ENABLE_ENCRYPT	0x106	SOURCE	Enable encryption.
H2_SOURCE_DISABLE_ENCRYPT	0x107	SOURCE	Disable encryption.

Important: *Unlisted command ID values are reserved and should not be used.*

5.16.5.TSI_W_FORCE_HOT_PLUG_STATE

```
TSI_W_FORCE_HOT_PLUG_STATE                    0x212
unsigned int hpd_state;                       U32
4 bytes                                       WO
```

Description

Writing this register will force the HPD status to the indicated state until another state is forced, or TSI generates a HPD pulse for another reason.

To force HPD to low state, write zero. To force HPD to high state, write one.

Important: *Writing this CI does not stop TSI from issuing HPD state changes at a later time. To stop TSI from generating HPD pulses by itself, set the TSI_HDP_LENGTH (0x1201) CI to zero.*

5.17.DP Sink – Link status

The DP Link status access is divided into two groups: Sink current link status and Sink link capabilities. The configuration item ID range 0x2B0 – 0x2EF is reserved for these features. Undefined CI's are reserved for future expansions.

The first group is DP Sink link status data: 0x2B0 - 0x2BF

Name	Description	Reference
TSI_R_DPRX_LINK_STATUS_FLAGS	DP Link status flags	5.17.1
TSI_R_DPRX_LT_STATUS_FLAGS	Previous DP Link training results	5.17.2
TSI_R_DPRX_LINK_VOLTAGE_SWING	DP Link voltage swing data	5.17.3
TSI_R_DPRX_LINK_PRE_EMPHASIS	DP Link pre-emphasis data	5.17.4
TSI_R_DPRX_LINK_LANE_COUNT	DP Link lane count	5.17.5
TSI_R_DPRX_LINK_RATE	DP Link rate	5.17.6
TSI_R_DPRX_ERROR_COUNTS	DP lane error counters	5.17.7
TSI_W_DPRX_DPCD_BASE	DP DPCD base pointer for DPCD register access	5.17.8
TSI_DPRX_DPCD_DATA	For reading and writing DPCD data.	5.17.9

The second group is DP Sink link capabilities: 0x2C0 - 0x2CF

Name	Description	Reference
TSI_DPRX_MAX_LANES	DP Link max. lanes supported	5.18.1
TSI_DPRX_MAX_LINK_RATE	DP Link max. rate	5.18.2
TSI_DPRX_LINK_FLAGS	DP Link feature flags	5.18.3

5.17.1.TSI_R_DPRX_LINK_STATUS_FLAGS

```
TSI_R_DPRX_LINK_STATUS_FLAGS                    0x2B0
unsigned int dprx_link_status                   U32
4 bytes                                         RO
```

Description

Indicates current link flags as defined below

Bits	Field name	Value	Description
0	L0_CR		Clock recovery done for lane 0
1	L0_EQ		Channel EQ done for lane 0
2	L0_SL		Symbol lock for lane 0
3		*	RESERVED
4	L1_CR		Clock recovery done for lane 1
5	L1_EQ		Channel EQ done for lane 1
6	L1_SL		Symbol lock for lane 1
7		*	RESERVED
8	L2_CR		Clock recovery done for lane 2
9	L2_EQ		Channel EQ done for lane 2
10	L2_SL		Symbol lock for lane 2
11		*	RESERVED
12	L2_CR		Clock recovery done for lane 2
13	L2_EQ		Channel EQ done for lane 2
14	L2_SL		Symbol lock for lane 2
15		*	RESERVED
16	ILA		Inter-lane align status
17	FRAMING	0	Normal framing
		1	Enhanced framing
18	SCRAMBLING	0	Scrambling disabled
		1	Scrambling enabled
19	FORMAT	0	Single stream mode
		1	Multi stream mode
31:20		*	RESERVED

5.17.2.TSI_R_DPRX_LT_STATUS_FLAGS

```
TSI_R_DPRX_LT_STATUS_FLAGS                    0x2B1
unsigned int dprx_lt_status                   U32
4 bytes                                       RO
```

Description

Link status as achieved during previous link-training.

Bits	Field name	Value	Description
0	L0_CR		Clock recovery done for lane 0
1	L0_EQ		Channel EQ done for lane 0
2	L0_SL		Symbol lock for lane 0
3		*	RESERVED
4	L1_CR		Clock recovery done for lane 1
5	L1_EQ		Channel EQ done for lane 1
6	L1_SL		Symbol lock for lane 1
7		*	RESERVED
8	L2_CR		Clock recovery done for lane 2
9	L2_EQ		Channel EQ done for lane 2
10	L2_SL		Symbol lock for lane 2
11		*	RESERVED
12	L2_CR		Clock recovery done for lane 2
13	L2_EQ		Channel EQ done for lane 2
14	L2_SL		Symbol lock for lane 2
31:15		*	RESERVED

5.17.3.TSI_R_DPRX_LINK_VOLTAGE_SWING

```
TSI_R_DPRX_LINK_VOLTAGE_SWING                    0x2B2
unsigned int dprx_link_voltage_swing             U32
4 bytes                                          RO
```

Description

Indicates DP link voltage swing for all active lanes.

Bits	Field name	Value	Description
7:0	L0_VS	0	Lane 0 voltage swing 400 mVpp
		1	Lane 0 voltage swing 600 mVpp
		2	Lane 0 Voltage swing 800 mVpp
		3	Lane 0 Voltage swing 1200 mVpp
		255-4	RESERVED
15:8	L1_VS	*	Voltage swing for Lane 1 (See lane 0 for value definitions)
23:16	L2_VS	*	Voltage swing for Lane 2 (See lane 0 for value definitions)
31:24	L3_VS	*	Voltage swing for Lane 3 (See lane 0 for value definitions)

5.17.4.TSI_R_DPRX_LINK_PRE_EMPHASIS

```
TSI_R_DPRX_LINK_PRE_EMPHASIS                     0x2B3
unsigned int dprx_link_pre_emphasis              U32
4 bytes                                          RO
```

Description

Indicates DP link pre-emphasis for all active lanes.

Bits	Field name	Value	Description
7:0	L0_PE	0	Lane 0 pre-emphasis 0 dB
		1	Lane 0 pre-emphasis 3.5 dB
		2	Lane 0 pre-emphasis 6 dB
		3	Lane 0 pre-emphasis 9.5 dB
		255-4	RESERVED
15:8	L1_PE	*	Pre-emphasis for Lane 1 (See lane 0 for value definitions)
23:16	L2_PE	*	Pre-emphasis for Lane 2 (See lane 0 for value definitions)
31:24	L3_PE	*	Pre-emphasis for Lane 3 (See lane 0 for value definitions)

5.17.5.TSI_R_DPRX_LINK_LANE_COUNT

```
TSI_R_DPRX_LINK_LANE_COUNT                    0x2B4
unsigned int dprx_link_lanes                  U32
4 bytes                                       RO
```

Description

Indicates number of currently active lanes. Valid values are 1, 2 or 4.

5.17.6.TSI_R_DPRX_LINK_RATE

```
TSI_R_DPRX_LINK_RATE                          0x2B5
unsigned int dprx_link_rate                   U32
4 bytes                                       RO
```

Description

Indicates the current link rate as multiple of 0.27Gbps. Typical values are 6, 10 or 20. Please note that some DP standards allow additional link rates.

5.17.7.TSI_R_DPRX_ERROR_COUNTS

```
TSI_R_DPRX_ERROR_COUNTS                       0x2B8
unsigned int dprx_error_data[]                ARRAY_U32
16 bytes                                      CRO
```

Description

Contains error counts for each used DP Lane. Reading the counters also clears them.

Index	Description
0	Error counter for lane 0
1	Error counter for lane 1
2	Error counter for lane 2
3	Error counter for lane 3

5.17.8.TSI_W_DPRX_DPCD_BASE

```
TSI_W_DPRX_DPCD_BASE                              0x2B9
unsigned int dprx_dpcd_base_ptr                   U32
4 bytes                                           WO
```

Description

DPCD read and write start pointer into the DPCD register space. The DPCD address value may not exceed 0x00FFFFFF. Default value is 0.

5.17.9.TSI_DPRX_DPCD_DATA

```
TSI_DPRX_DPCD_DATA                                0x2BA
unsigned char dprx_dpcd_data[]                    ARRAY_U8
Variable size                                     RW
```

Description

Read and/or write DPCD registers. Each DPCD register is one byte (8 bits). Read/Write access size is not limited, but access size + DPCD base address may not exceed 0x01000000. Please refer to DP Specifications in order to decode the DPCD register data.

5.18.DP Sink - Capabilities

5.18.1.TSI_DPRX_MAX_LANES

```
TSI_DPRX_MAX_LANES                              0x2C0
unsigned int dprx_max_lanes                     U32
4 bytes                                         RW
```

Description

Defines maximum number of lanes for link training. Valid settings are 1, 2 and 4.

Important: *Trigger link training by using the TSI_W_FORCE_HOT_PLUG_STATE CI to generate a HPD signal.*

5.18.2.TSI_DPRX_MAX_LINK_RATE

```
TSI_DPRX_MAX_LINK_RATE                          0x2C1
unsigned int dprx_max_link_rate                 U32
4 bytes                                         RW
```

Description

Defines maximum link rate for link training. Setting is multipler for 0.27 Gbps. Typical settings are 6 (RBR), 10 (HBR) and 20 (HBR2).

Important: *Trigger link training by using the TSI_W_FORCE_HOT_PLUG_STATE CI to generate a HPD signal.*

5.18.3.TSI_DPRX_LINK_FLAGS

```
TSI_DPRX_LINK_FLAGS                                0x2C2
unsigned int dprx_link_flags                       U32
4 bytes                                            RW
```

Description

Defines which features are indicated as supported for link training. See table below for flag defitions:

Bits	Field name	Value	Description
0	MST	0	Indicate SST support only.
		1	Indicate MST supported.
1	TPS3	0	Indicate TPS3 feature is not supported.
		1	Indicate TPS3 feature is supported
		Important: *For HBR2 (Link rate = 20) capable devices, the TPS3 feature must be indicated as supported (1).*	
31:2		*	RESERVED

Important: *Trigger link training by using the TSI_W_FORCE_HOT_PLUG_STATE CI to generate a HPD signal.*

5.19.Configuration items for USB Type-C

This section defines the new configuration items that are specific to UCD-340 and USB Type-C. The CI Space from 0x600 to 0x6ff is reserved for USB-Type C specific controls.

Name	Description	Reference
TSI_W_USBC_CABLE_CONTROL	Cable related controls	5.19.1
TSI_W_USBC_INITIAL_ROLE	USB Type-C role control	5.19.2
TSI_USBC_DP_ALT_MODE_SETUP	DP Alternate mode settings	5.19.3
TSI_W_USBC_ROLE_CONTROL	USB Type-C Role swap requests	5.19.4
TSI_W_USBC_DP_ALT_MODE_COMMAND	DP Alternate mode commands	5.19.5
TSI_R_USBC_TE_HW_CONFIGURATION	HW Configuration info	5.19.6
TSI_R_USBC_CABLE_STATUS	Cable status information	5.19.7
TSI_R_USBC_ROLE_STATUS	USB Type-C Role status information	5.19.9
TSI_R_USBC_DP_ALT_MODE_STATUS	DP Alternate mode status information	5.19.10
TSI_R_USBC_POWER_STATUS	USB Type-C Power status information	5.19.11
TSI_R_USBC_POWER_SOURCE_PDO	Power contract information	5.19.12
TSI_R_USBC_POWER_SINK_RDO	Power contract information	5.19.13
TSI_R_USBC_IDO_TABLE	Cable E-marker information	5.19.8

(Continued...)

(...Continued)

Additionally, CI range from 0x10500 to 0x105ff is reserved for USB-C electrical testing.

Name	Description	Reference
TSI_USBC_EL_TIMEOUT	Test timeout in ms	5.20.1
TSI_USBC_EL_DUT_CAPS	DUT Capability bits	5.20.2
TSI_USBC_EL_REPLUG_TIME	Re-plug disconnected state time in ms	5.20.3
TSI_USBC_EL_DUT_ATTACH_TIMEOUT	DUT Attach max time in ms	5.20.4
TSI_USBC_EL_PWR_CONTRACT_TIMEOUT	DUT Power contract timeout in ms.	5.20.5
TSI_USBC_EL_CC_LOW_VOLTAGE_1	CC lines voltage range 1 low limit, mV	5.20.6
TSI_USBC_EL_CC_HI_VOLTAGE_1	CC lines voltage range 1 hi limit, mV	5.20.7
TSI_USBC_EL_CC_LOW_VOLTAGE_2	CC lines voltage range 2 low limit, mV	5.20.8
TSI_USBC_EL_CC_HI_VOLTAGE_2	CC lines voltage range 2 hi limit, mV	5.20.9
TSI_USBC_EL_CC_LOW_VOLTAGE_3	CC lines voltage range 3 low limit, mV	5.20.10
TSI_USBC_EL_CC_HI_VOLTAGE_3	CC lines voltage range 3 hi limit, mV	5.20.11
TSI_USBC_EL_VCON_LOW_VOLTAGE	Vcon voltage range, low limit, mV	5.20.12
TSI_USBC_EL_VCON_HI_VOLTAGE	Vcon voltage range, hi limit, mV	5.20.13
TSI_USBC_EL_DP_ALT_TIMEOUT	DP Alternate mode entry timeout, ms	5.20.14
TSI_USBC_EL_AUX_P_IDLE_LOW_VOLTAGE	DP AUX+ voltage range, low limit, mV	5.20.15
TSI_USBC_EL_AUX_P_IDLE_HI_VOLTAGE	DP AUX+ voltage range, hi limit, mV	5.20.16
TSI_USBC_EL_AUX_N_IDLE_LOW_VOLTAGE	DP AUX- voltage range, low limit, mV	5.20.17
TSI_USBC_EL_AUX_N_IDLE_HI_VOLTAGE	DP AUX- voltage range, hi limit, mV	5.20.18
TSI_USBC_EL_VBUS_LOW_VOLTAGE	Vbus voltage range, low limit, mV	5.20.19
TSI_USBC_EL_VBUS_HI_VOLTAGE	Vbus voltage range, hi limit, mV	5.20.20
TSI_USBC_EL_VBUS_CURRENT_MAX_DEV	Vbus pins, current max. deviation, µA	5.20.21
TSI_USBC_EL_GND_CURRENT_MAX_DEV	GND pins, current max. deviation, µA	5.20.22

5.19.1.TSI_W_USBC_CABLE_CONTROL

ClientVersion 9, and higher *Advanced License required*

```
TSI_W_USBC_CABLE_CONTROL                    0x600
unsigned int USBC_cable_control             U32
4 bytes                                     WO
```

Synopsis

Command CI Used to control cable related features, like connection, orientation and power sourcing capability.

Command ID	Description
0	No operation. Writing zero has no effect.
1	Set cable to “straight” orientation. **This is default orientation.** ***Important:*** *using this command requires the Unigraf Electrical Testing cable to be connected to the USB-Type C port.*
2	Set cable to “Flipped” orientation. ***Important:*** *using this command requires the Unigraf Electrical Testing cable to be connected to the USB-Type C port.*
3	Indicate 0.9A power source capability.
4	Indicate 1.5A power source capability.
5	Indicate 3.0A power source capability. **This is default pull-up selection**
6	Disconnect CC lines. The DUT will see this as cable unplugged. **Default after device open.**
7	Connect CC lines. The DUT will see this as cable plugged.

5.19.2.TSI_W_USBC_INITIAL_ROLE

ClientVersion 9, and higher *Advanced License required*

```
TSI_W_USBC_INITIAL_ROLE                          0x601
unsigned int USBC_initial_role                   U32
4 bytes                                          WO
```

Synopsis

Command CI Used to control initial role settings and related configuration. Issue any commands here before issuing the "Connect CC lines" command on the 5.19.1 TSI_W_USBC_CABLE_CONTROL.

Command ID	Description
0	No operation. Writing zero has no effect.
1	Set initial port role to UFP.
2	Set initial port role to DFP.
3	Set initial port role to DRP. ***Important:*** *Actual mode after cable plug will not be DRP – Instead it will be either UFP or DFP depending on connected DUT. If connected DUT is also DRP, it is not possible to predict which role will be selected by TE due to the way that DRP is implemented in the USB-Type C specifications.*

5.19.3.TSI_USBC_DP_ALT_MODE_SETUP

ClientVersion 9, and higher *Advanced License required*

```
TSI_USBC_DP_ALT_MODE_SETUP                       0x602
unsigned int USBC_DP_alt_mode_setup              U32
4 bytes                                          RW
```

Synopsis

DisplayPort alternate mode capabilities and feature definitions.

Bit	Description	
1:0	RESERVED	
3:2	Pin mapping capability flags for UFP port role.	
	Flag	
	1	Indicate support for pin mapping "C": DP v1.3, 4 lanes
	2	Indicate support for pin mapping "D": DP v1.3, 2 lanes + USB 3.1
30:4	RESERVED	
31	DP Alternate mode auto enter on cable plug	
	0	Auto-enter functionality is disabled
	1	Auto-enter functionality is enabled

5.19.4.TSI_W_USBC_ROLE_CONTROL

ClientVersion 9, and higher *Advanced License required*

```
TSI_W_USBC_ROLE_CONTROL                          0x603
unsigned int USBC_role_control                   U32
4 bytes                                          WO
```

Synopsis

Command CI to control port roles after cable is plugged into TE and roles are established. Once the command is issued, please read the 5.19.9 TSI_R_USBC_ROLE_STATUS CI to check if the role was actually changed.

Important: *A role swap is an expensive operation that can take up to 5 seconds to complete depending on used DUT.*

Important: *The TE can only request swapping of the roles, DUT can always reject the request leading to no change in the respective role.*

Command ID	Description
0	No operation. Writing zero has no effect.
1	Request swapping data role.
2	Request swapping power role.
3	Request swapping Vconn.

5.19.5.TSI_W_USBC_DP_ALT_MODE_COMMAND

ClientVersion 9, and higher *Advanced License required*

```
TSI_W_USBC_DP_ALT_MODE_COMMAND                  0x604
unsigned int USBC_DP_alt_mode_command           U32
4 bytes                                         WO
```

Synopsis

Used to manually control USB Alternate mode.

Important: *Issuing any command (except for "No operation") will clear the "DP alternate mode auto enter on cable plug" bit in 5.19.3 TSI_USBC_DP_ALT_MODE_SETUP CI.*

Command ID	Description
0	No operation. Writing zero has no effect.
1	Exit any DP Alternate mode currently in effect. ***Important:*** *This command is only available in the DFP role.*
2	Ignore any future DP Alternate mode entry requests originating from the DUT. ***Important:*** *This command is only available in the UFP role.*
3	Enter DP Alternate mode with pin assignment "D": DP v1.3, 2 lanes + USB 3.1. ***Important:*** *This command is only available in the DFP role.*
4	Enter DP Alternate mode with pin assignment "C": DP v1.3, 4 lanes. ***Important:*** *This command is only available in the DFP role.*

5.19.6.TSI_R_USBC_TE_HW_CONFIGURATION

ClientVersion 9, and higher *Basic License required*

```
TSI_R_USBC_TE_HW_CONFIGURATION                  0x605
unsigned int USBC_TE_HW_config                  U32
4 bytes                                         RO
```

Synopsis

Contains information about the optional hardware modules and external devices.

Bit	Description
0	Unigraf External power-supply / Power-sink device is connected to the TE.
1	Electrical Testing board is installed on the TE.
31:3	RESERVED

5.19.7.TSI_R_USBC_CABLE_STATUS

ClientVersion 9, and higher *Basic License required*

```
TSI_R_USBC_CABLE_STATUS                          0x606
unsigned int USBC_cable_status                   U32
4 bytes                                          RO
```

Synopsis

Contains cable-related status data.

Bit	Description	
0	Cable plugged state	
	0	Cable is unplugged
	1	Cable is plugged
1	Cable orientation	
	0	Straight orientation
	1	Flipped orientation
2	Indicates the connection status	
	0	Disconnected
	1	Connected
3	Cable E-Marking ***Important:*** *This information is valid only in DFP data role.*	
	0	Unmarked cable is attached.
	1	E-Marked cable is attached.
4	Unigraf Electrical testing cable detect	
	0	Normal USB-C cable
	1	Unigraf Electrical Testing cable is attached.
5	Indication for E-Marker IDO table in 5.19.8 TSI_R_USBC_IDO_TABLE validity. ***Important:*** *It may take a while to read the IDO table from an E-marked cable. The E-Marker should become available if the data role is DFP and the cable is E-Marked.*	
	0	E-Marker IDO table does not contain valid information.
	1	E-Marker IDO table contains valid information.
31:5	RESERVED	

5.19.8.TSI_R_USBC_IDO_TABLE

ClientVersion 9, and higher *Advanced License required*

```
TSI_R_USBC_IDO_TABLE                          0x60c
unsigned int USBC_IDO_Table[]                 ARRAY_U32
Variable size, Max. 24 bytes                  RO
```

Synopsis

Provides access to Identity Data Objects (IDO) which are received from near cable plug as the reply to Discover Identity request. The content of this table is cleared on reset or cable unplug event.

Index	IDO	Description
0	ID Header	See 6.4.4.3.1.1 of PD specification
1	Cert stat	32-bit integer assigned by USB-IF
2	Product	See 6.4.4.3.1.9 of PD specification
3..5	Produc type specific	See 6.4.4.3.1.10.1 for passive cable VDO, 6.4.4.3.1.10.2 for active cable VDO.

5.19.9.TSI_R_USBC_ROLE_STATUS

ClientVersion 9, and higher *Basic License required*

```
TSI_R_USBC_ROLE_STATUS                        0x607
unsigned int USBC_role_status                 U32
4 bytes                                       RO
```

Synopsis

Indicates the current device roles.

Bit		
0	Data Role	
	0	Up Facing Port (UFP)
	1	Down Facing Port (DFP)
1	Power role	
	0	Source
	1	Sink
2	Vconn	
	0	Off
	1	On
31:3	RESERVED	

5.19.10.TSI_R_USBC_DP_ALT_MODE_STATUS

ClientVersion 9, and higher *Basic License required*

```
TSI_R_USBC_DP_ALT_MODE_STATUS                    0x608
unsigned int USBC_alt_mode                       U32
4 bytes                                          RO
```

Synopsis

Indicates the current alternate mode and it's setup

Bit	Description	
0	DisplayPort Alternate mode mode indicator	
	0	DisplayPort alternate mode is not in use ***Important:*** *In this mode, the 2:1, 6:3 and 11:8 bit-fields do not contain valid data!*
	1	DisplayPort alternate mode is active.
2:1	DisplayPort Alternate mode configuration	
	0	Set configuration USB
	1	Set UFP_U as DFP_D
	2	Set UFGP_U as UFP_D
	3	RESERVED
6:3	Signaling	
	0	Unspecified
	1	DP v1.3
	2	GEN2
	*	RESERVED
7	RESERVED	
11:8	Pin Assignment	
	0	No pin assignment
	1	“A”: DisplayPort GEN2 4/2 lanes
	2	“B”: DisplayPort GEN2 2/1 lanes
	3	“C”: DisplayPort v1.3 4 lanes (USB Type-C cable)
	4	“D”: DisplayPort v1.3 2 lanes + USB 3.1 (USB Type-C cable)
	5	“E”: DisplayPort v1.3 4 lanes (USB Type-C to DP adapter)
	6	“F”: DisplayPort v1.3 2 lanes + USB GEN1 (USB Type-C to DP adapter
	*	RESERVED
31:12	RESERVED	

5.19.11.TSI_R_USBC_POWER_STATUS

ClientVersion 9, and higher *Basic License required*

```
TSI_R_USBC_POWER_STATUS                          0x609
unsigned int USBC_power_status                   U32
4 bytes                                          RO
```

Synopsis

Indicates current power status.

Important: *The indicated internal loading resistor values (bits 8:5) apply to UCD-340 front-end REV-C only.*

Bit	Description	
1:0	Power sink current	
	0	Legacy current selected: 0.9A current.
	1	Legacy current selected: 1.5A current
	2	Legacy current selected: 3.0A current
	3	Defined by power contract
2	Power contract established	
	0	Power contract not established.
	1	Power contract established.
4:3	RESERVED	
8:5	Internal loading resistors for power sinking.	
	0	Internal load resistor connections are open: no internal loading is applied.
	1	10Ω (0.5A) resistive load applied. (This setting is used for 0.5A USB power loading)
	2	5.6Ω (0.89A) resistive load applied. (This setting is used for 0.9A USB power loading)
	3	3.58Ω (1.39A) resistive load applied.
	4	3.4Ω (1.47A) resistive load applied. (This setting is used for 1.5A USB power loading)
	5	2.54Ω (1.97A) resistive load applied.
	6	2.12Ω (2.36A) resistive load applied.
	7	1.74Ω (2.86A) resistive load applied. (This setting is used for 3.0A USB power loading)
	*	RESERVED
31:9	RESERVED	

5.19.12.TSI_R_USBC_POWER_SOURCE_PDO

ClientVersion 9, and higher	*Advanced License required*

TSI_R_USBC_POWER_SOURCE_PDO	0x60a
unsigned int USBC_power_pdo	U32
4 bytes	RO

Synopsis

This CI provides access to the PDO that was requested by a power sink device from the TE. Data in this CI is valid if the power role is Source, and power contract established bit (2) is set in 5.19.11 TSI_R_USBC_POWER_STATUS CI.

The PDO Data is in raw form as defined in USB DP Standard.

5.19.13.TSI_R_USBC_POWER_SINK_RDO

ClientVersion 9, and higher	*Advanced License required*

TSI_R_USBC_POWER_SINK_RDO	0x60b
unsigned int USBC_power_rdo	U32
4 bytes	RO

Synopsis

This CI provides access to the RDO selected by TE. Data in this CI is valid if the power role is Sink, and power contract established bit (2) is set in 5.19.11 TSI_R_USBC_POWER_STATUS CI.

The RDO Data in RAW form as defined in USB DP Standard.

5.19.14.TSI_R_USBC_IDO_TABLE

ClientVersion 11, and higher	*Advanced License required*

TSI_R_USBC_IDO_TABLE	0x60c
unsigned int USBC_IDO_Table[]	ARRAY_U32
Variable size	RO

Synopsis

This CI provides access to USB-C IDO table. Read table with zero size to get required buffer size, or use sufficient buffer size on first attempt.

5.19.15.TSI_USBC_EPU_LOAD_CONTROL

ClientVersion 11, and higher *No license requirements*

```
TSI_USBC_EPU_LOAD_CONTROL                    0x60d
unsigned int EPU_Load_Bits                   U32
4 bytes                                      RW
```

Synopsis

Provides access to control loading resistors on external power unit. See below for valid selections:

Value	Description
0x001	Select EPU loading resistor configuration for 7.0 Ω load
0x002	Select EPU loading resistor configuration for 6.0 Ω load
0x004	Select EPU loading resistor configuration for 5.0 Ω load
0x008	Select EPU loading resistor configuration for 4.0 Ω load
0x010	Select EPU loading resistor configuration for 3.3 Ω load
0x020	Select EPU loading resistor configuration for 13.3 Ω load
0x080	Select EPU loading resistor configuration for 1.6 Ω load
0x100	Select EPU loading resistor configuration for 8.5 Ω load
0x200	Select EPU loading resistor configuration for 10.0 Ω load
*	RESERVED – Do not use!

5.19.16.TSI_USBC_PWR_CONTRACT_CONTROL

ClientVersion 11, and higher *Advanced License required*

```
TSI_USBC_PWR_CONTRAC_CONTROL                    0x60e
unsigned int PWR_Contract_Control               U32
4 bytes                                         RW
```

Synopsis

Defines how the UCD-340 selects which of the offered power contract options is to be selected. See table below for configuration options:

Bit	Description	
0	Auto negotiate flag	
	0	Don't auto-negotiate
	1	Automatically negotiate power contract, use fixed voltage PDO
1	Use battery PDO	
	0	Don't use battery PDO for power contract
	1	Use batter PDO for power contract negotiation.
2	Use variable PDO	
	0	Don't use variable power PDO for power contract
	1	Use variable power PDO in power contract negotiation.
3	USB Communication capable flag (copied from 5.19.13 TSI_R_USBC_POWER_SINK_RDO, bit #25).	
5:4	Contract preference	
	0	Prefer higher current power contract
	1	Prefer higher voltage power contract
	2	Prefer higher power power contract
	3	RESERVED
6	No USB suspend flag (copied from 5.19.13 TSI_R_USBC_POWER_SINK_RDO, bit #24).	
7	"Give back flag" (copied from 5.19.13 TSI_R_USBC_POWER_SINK_RDO, bit #27).	
8	Automatic minimum power	
	0	Don't calculate minimum power.
	1	Automatically calculate minimum required power.
25:16	Minimum required power.	
31	Manual power contract selection.	
	0	Auto-negotiate power contract.
	1	Power contract selected by index CI. (5.19.17 TSI_USBC_PWR_CONTRACT_SELECT)

5.19.17.TSI_USBC_PWR_CONTRACT_SELECT

ClientVersion 11, and higher	*Advanced License required*

TSI_USBC_PWR_CONTRACT_SELECT	0x60f
unsigned int PWR_Contract_Sel	U32
4 bytes	RW

Synopsis

If bit 31 is set in 5.19.16 TSI_USBC_PWR_CONTRACT_CONTROL, this CI is used to select which PDO is used to establish the power contract with link partner.

5.19.18.TSI_USBC_PWR_LOCAL_SINK_PDO

ClientVersion 11, and higher	*Advanced License required*

TSI_USBC_PWR_LOCAL_SINK_PDO	0x610
unsigned int LocalSinkPDO[]	ARRAY_U32
Variable size	RW

Synopsis

Contains binary images of PDO's that advertise the TE power sinking capabilities. Maximum number of PDO's is 7. For details on the format of these PDO's, please refer to "USB DP standard" (Tables 6-6, 6-8, 6-9).

5.19.19.TSI_USBC_PWR_LOCAL_SOURCE_PDO

ClientVersion 11, and higher	*Advanced License required*

TSI_USBC_PWR_LOCAL_SOURCE_PDO	0x611
unsigned int LocalSourcePDO[]	ARRAY_U32
Variable size	RW

Synopsis

Contains binary images of PDO's that advertise the TE power sourcing capabilities. Maximum number of PDO's is 7. For details on the format of these PDO's, please refer to "USB DP standard" (Tables 6-6, 6-8, 6-9).

5.19.20.TSI_R_USBC_PWR_REMOTE_SINK_PDO

ClientVersion 11, and higher *Advanced License required*

```
TSI_R_USBC_PWR_REMOTE_SINK_PDO                0x612
unsigned int RemoteSinkPDO[]                  ARRAY_U32
Variable size                                 RO
```

Synopsis

Contains binary images of PDO's received from DUT advertising the DUT's power sink requirements. Maximum number of PDO's is 7. For details on the format of these PDO's, please refer to "USB DP standard" (Tables 6-6, 6-8, 6-9).

5.19.21.TSI_R_USBC_PWR_REMOTE_SOURCE_PDO

ClientVersion 11, and higher *Advanced License required*

```
TSI_R_USBC_PWR_REMOTE_SOURCE_PDO              0x613
unsigned int RemoteSourcePDO[]                ARRAY_U32
Variable size                                 RO
```

Synopsis

Contains binary images of PDO's received from DUT advertising the DUT's power sourcing capabilities. Maximum number of PDO's is 7. For details on the format of these PDO's, please refer to "USB DP standard" (Tables 6-6, 6-8, 6-9).

5.20.USB Type-C Electrical tests

This section defines the USB Type-C Electrical test related CI's.

5.20.1.TSI_USBC_EL_TIMEOUT

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_TIMEOUT                              0x00010500
unsigned int usvc_el_timeout                     U32
4 bytes                                          RW
```

Synopsis

Defines the test activity maximum run-time, in milliseconds. Default setting is 5000ms.

Important: *When the test is waiting for DUT with a max. delay this timeout is not advancing during the wait.*

5.20.2.TSI_USBC_EL_DUT_CAPS

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_DUT_CAPS                             0x00010501
unsigned int usbc_el_dut_caps                    U32
4 bytes                                          RW
```

Synopsis

Defines DUT capabilities. Please see flag definitions below. Default setting is 0.

Bits	Description	
0	DUT Support DisplayPort Alternate mode.	
	0	Not supported
	1	Supported
1	DUT can act as a power source	
	0	No power source functionality
	1	Power source functionality present
2	DUT can receive power from TE	
	0	DUT Can not powered from USB Type-C
	1	DUT Can receive power from USB Type-C
31:0	RESERVED	

5.20.3.TSI_USBC_EL_REPLUG_TIME

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_REPLUG_TIME                          0x00010502
unsigned int usbc_replug_time                    U32
4 bytes                                          RW
```

Synopsis

Defines the time period for USB Type-C re-plug simulation “disconnected” state. The period is defined in milliseconds. Default value is 1500ms.

5.20.4.TSI_USBC_EL_DUT_ATTACH_TIMEOUT

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_DUT_ATTACH_TIMEOUT                   0x00010503
unsigned int usbc_dut_attach_timeout             U32
4 bytes                                          RW
```

Synopsis

Defines the time period that the TE will wait for DUT to complete connection after cable plug. Time is defined in milliseconds. Default value is 10000ms.

5.20.5.TSI_USBC_EL_PWR_CONTRACT_TIMEOUT

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_PWR_CONTRACT_TIMEOUT                 0x00010504
unsigned int usbc_pwr_contract_timeout           U32
4 bytes                                          RW
```

Synopsis

Defines the time period that the TE will wait for DUT to complete power contract negotiation. Time is defined in milliseconds. Default value is 5000ms.

5.20.6.TSI_USBC_EL_CC_LOW_VOLTAGE_1

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_CC_LOW_VOLTAGE_1                    0x00010505
unsigned int usbc_cc_low_voltage1               U32
4 bytes                                         RW
```

Synopsis

Defines the low limit for the voltage window when power sink current is 0.5A or 0.9A. The limit is defined in millivolts (mV). Default setting is 261mV.

5.20.7.TSI_USBC_EL_CC_HI_VOLTAGE_1

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_CC_HI_VOLTAGE_!                     0x00010506
unsigned int usbc_cc_hi_voltage1                U32
4 bytes                                         RW
```

Synopsis

Defines the high limit for the voltage window when power sink current is 0.5A or 0.9A. The limit is defined in millivolts (mV). Default settingf is 588mV.

5.20.8.TSI_USBC_EL_CC_LOW_VOLTAGE_2

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_CC_LOW_VOLTAGE_2                    0x00010507
unsigned int usbc_cc_low_voltage2               U32
4 bytes                                         RW
```

Synopsis

Defines the low limit for the voltage window when power sink current is 1.5A. The limit is defined in millivolts (mV). Default setting is 675mV.

5.20.9.TSI_USBC_EL_CC_HI_VOLTAGE_2

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_CC_HI_VOLTAGE_2                    0x00010508
unsigned int usbc_cc_hi_voltage2               U32
4 bytes                                        RW
```

Synopsis

Defines the high limit for the voltage window when power sink current is 1.5A. The limit is defined in millivolts (mV). Default setting is 1189mV.

5.20.10.TSI_USBC_EL_CC_LOW_VOLTAGE_3

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_CC_LOW_VOLTAGE_3                   0x00010509
unsigned int usbc_cc_low_voltage3              U32
4 bytes                                        RW
```

Synopsis

Defines the low limit for the voltage window when power sink current is 3.0A. The limit is defined in millivolts (mV). Default setting is 1238mV.

5.20.11.TSI_USBC_EL_CC_HI_VOLTAGE_3

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_HI_VOLTAGE_3                       0x0001050a
unsigned int usbc_cc_hi_voltage3               U32
4 bytes                                        RW
```

Synopsis

Defines the high limit for the voltage window when power sink current is 3.0A. The limit is defined in millivolts (mV). Default setting is 2181mV.

5.20.12.TSI_USBC_EL_VCON_LOW_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_VCON_LOW_VOLTAGE                    0x0001050b
unsigned int usbc_vcon_low_voltage              U32
4 bytes                                         RW
```

Synopsis

Defines the low limit for the Vcon voltage window. The limit is defined in millivolts (mV). Default setting is 4750mV.

5.20.13.TSI_USBC_EL_VCON_HI_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_VCON_HI_VOLTAGE                     0x0001050c
unsigned int usbc_vcon_hi_voltage               U32
4 bytes                                         RW
```

Synopsis

Defines the high limit for the Vcon voltage window. The limit is defined in millivolts (mV). Default setting is 5500mV.

5.20.14.TSI_USBC_EL_DP_ALT_TIMEOUT

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_DP_ALT_TIMEOUT                      0x0001050d
unsigned int usbc_dp_alt_timeout                U32
4 bytes                                         RW
```

Synopsis

Defines the timeout the TE will wait for the DUT to enter into DisplayPort alternate mode. The timeout is defined in milliseconds. Default setting is 5000ms.

5.20.15.TSI_USBC_EL_AUX_P_IDLE_LOW_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_AUX_P_IDLE_LOW_VOLTAGE              0x0001050e
unsigned int usbc_aux_p_idle_lo_voltage         U32
4 bytes                                         RW
```

Synopsis

Defines the low voltage limit for the positive DP AUX line when idle. The limit is defined in millivolts (mV). Default setting 100mV.

5.20.16.TSI_USBC_EL_AUX_P_IDLE_HI_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_AUX_P_IDLE_HI_VOLTAGE               0x0001050f
unsigned int usbc_aux_p_idle_hi_voltage         U32
4 bytes                                         RW
```

Synopsis

Defines the high voltage limit for the positive DP AUX line when idle. The limit is defined in millivolts (mV). Default setting is 600mV.

5.20.17.TSI_USBC_EL_AUX_N_IDLE_LOW_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_AUX_N_IDLE_LOW_VOLTAGE              0x00010510
unsigned int usbc_aux_n_idle_lo_voltage         U32
4 bytes                                         RW
```

Synopsis

Defines the low voltage limit for the negative DP AUX line when idle. The limit is defined in millivolts (mV). Default setting is 2500mV.

5.20.18.TSI_USBC_EL_AUX_N_IDLE_HI_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_AUX_N_IDLE_HI_VOLTAGE              0x00010511
unsigned int usbc_aux_n_idle_hi_voltage        U32
4 bytes                                        RW
```

Synopsis

Defines the high voltage limit for the negative DP AUX line when idle. The limit is defined in millivolts (mV). Default setting is 3000mV.

5.20.19.TSI_USBC_EL_VBUS_LOW_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_VBUS_LOW_VOLTAGE                   0x00010512
unsigned int usbc_vbus_lo_voltage              U32
4 bytes                                        RW
```

Synopsis

Defines the low limit for Vbus voltage window. The limit is defined in millivolts (mV). Default setting is 4750mV.

5.20.20.TSI_USBC_EL_VBUS_HI_VOLTAGE

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_VBUS_HI_VOLTAGE                    0x00010513
unsigned int usbc_vbus_hi_voltage              U32
4 bytes                                        RW
```

Synopsis

Defines the high limit for Vbus voltage window. The limit is defined in millivolts (mV). Default setting is 5500mV.

5.20.21.TSI_USBC_EL_VBUS_CURRENT_MAX_DEV

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_VBUS_CURRENT_MAX_DEV                0x00010514
unsigned int vbus_max_deviation                 U32
4 bytes                                         RW
```

Synopsis

Defines the highest allowed deviation between maximum and minimum currents measured from the individual Vbus pins as per-mill (‰) of total measured current. This means that if the total measured current is 3000mA, and the setting 100, the maximum difference that is allowed between maximum and minimum currents is 300mA. Default setting is 100‰.

5.20.22.TSI_USBC_EL_GND_CURRENT_MAX_DEV

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_GND_CURRENT_MAX_DEV                 0x00010515
unsigned int gnd_max_deviation                  U32
4 bytes                                         RW
```

Synopsis

Defines the highest allowed deviation between maximum and minimum currents measured from the individual GND pins as per-mill (‰) of total measured current. This means that if the total measured current is 3000mA, and the setting 100, the maximum difference that is allowed between maximum and minimum currents is 300mA. Default setting is 100‰.

5.20.23.TSI_USBC_EL_PWR_MEASURE_DELAY

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_PWR_MEASURE_DELAY                   0x000105016
unsigned int pwr_measure_delay                  U32
4 bytes                                         RW
```

Synopsis

Defines delay from end of power contract negotiation to voltage / current measurements. The delay is defined in milliseconds, and the default setting is 2000ms

5.20.24.TSI_USBC_EL_MIN_DUT_CURRENT

ClientVersion 10, and higher *Electrical test license required*

```
TSI_USBC_EL_MIN_DUT_CURRENT                    0x000105017
unsigned int dut_min_power_use                 U32
4 bytes                                        RW
```

Synopsis

Defines the minimum current, in mA, that a Power Sink DUT must use in order to pass the test. Set this value to zero (0) to disable minimum current check. Default value is 0 (=Disabled).

5.21.Pattern generator CI definitions

This section defines the new configuration items that are specific to pattern generator functionality in TSI. The CI Space from 0x700 to 0x77f is reserved for pattern generator specific controls.

5.21.1.TSI_PG_ENABLED_STREAM_COUNT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_ENABLED_STREAM_COUNT                    0x700
unsigned int enabled_stream_count              U32
4 bytes                                        RW
```

Synopsis

Used to get or set number of pattern generators running. Maximum number of supported streams depends on used video interface, and is available as run-time information by reading the 5.21.2 TSI_R_PG_MAX_STREAM_COUNT CI. Zero means no pattern generators are running.

5.21.2.TSI_R_PG_MAX_STREAM_COUNT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_PG_MAX_STREAM_COUNT                      0x701
unsigned int max_stream_count                  U32
4 bytes                                        RO
```

Synopsis

Indicates the maximum number of streams the currently selected interface can support. Zero means no pattern generators can be used on the interface.

5.21.3.TSI_PG_STREAM_SELECT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_STREAM_SELECT                          0x702
unsigned int stream_select                    U32
4 bytes                                       RW
```

Synopsis

Get or set the stream to be configured. Streams are numbered from zero to max number of streams (5.21.2 TSI_R_PG_MAX_STREAM_COUNT) minus one. For SST only interfaces reading or writing this selection has no effect.

Important: *You can select streams that are not enabled (yet) in order to configure the parameters before enabling the stream.*

5.21.4.TSI_W_PG_COMMAND

ClientVersion 11, and higher *<license:TBD>*

```
TSI_W_PG_COMMAND                              0x703
unsigned int pattern_gen_command              U32
4 bytes                                       WO
```

Synopsis

Write non-zero value to apply current timing and pattern on the active output.

5.21.5.TSI_PG_CUSTOM_TIMING_HTOTAL

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_HTOTAL                   0x704
unsigned int custom_timing_htotal             U32
4 bytes                                       RW
```

Synopsis

Indicates the total width of a scanline in pixel clocks.

5.21.6.TSI_PG_CUSTOM_TIMING_HSTART

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_HSTART                    0x705
unsigned int custom_timing_hstart              U32
4 bytes                                        RW
```

Synopsis

Defines number of pixels from end of H-Sync to start of active area.

5.21.7.TSI_PG_CUSTOM_TIMING_HACTIVE

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_HACTIVE                   0x706
unsigned int custom_timing_hacktive            U32
4 bytes                                        RW
```

Synopsis

Indicates the width of video frame that is visible on screen as number of pixel clocks.

5.21.8.TSI_PG_CUSTOM_TIMING_HSYNCW

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_HSYNCW                    0x707
int custom_timing_hsyncw                       I32
4 bytes                                        RW
```

Synopsis

Indicates the width of horizontal sync as number of pixel clocks.

Important: *Writing negative value means negative polarity of this sync signal.*

5.21.9.TSI_PG_CUSTOM_TIMING_VTOTAL

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_VTOTAL                    0x708
unsigned int custom_timing_vtotal              U32
4 bytes                                        RW
```

Synopsis

Indicates the total height of the frame as number of scanlines.

5.21.10.TSI_PG_CUSTOM_TIMING_VSTART

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_VSTART                    0x709
unsigned int custom_timing_vstart              U32
4 bytes                                        RW
```

Synopsis

Defines number of scan lines from end of V-Sync to start of active area.

5.21.11.TSI_PG_CUSTOM_TIMING_VACTIVE

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_VACTIVE                   0x70a
unsigned int custom_timing_vactive             U32
4 bytes                                        RW
```

Synopsis

Indicates the height of the frame that is visible on screen as number of scanlines.

5.21.12.TSI_PG_CUSTOM_TIMING_VSYNCW

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_VSYNCW                    0x70b
int custom_timing_VSYNCW                       I32
4 bytes                                        RW
```

Synopsis

Indicates the width of the vertical sync as number of scanlines.

Important: *Writing negative value means negative polarity of this sync signal.*

5.21.13.TSI_PG_CUSTOM_TIMING_FLAGS

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_FLAGS                    0x70c
unsigned int custom_timing_flags              U32
4 bytes                                       RW
```

Synopsis

Signal output timing flags and color information. These settings are used to set-up the physical output signal. Please see table below:

Bits	Description	
7:0	Color depth	
	0	6 bits per color channel
	1	8 bits per color channel
	2	10 bits per color channel
	3	12 bits per color channel
	4	16 bits per color channel
	*	RESERVED
8	Interlace	
	0	Progressive
	1	Interlaced
9	H-Sync polarity ***Important:*** *TSI LITE edition does not support this flag. Please use negative H-Sync value instead.*	
	0	Positive
	1	Negative
10	V-Sync polarity ***Important:*** *TSI LITE edition does not support this flag. Please use negative V-sync value instead.*	
	0	Positive
	1	Negative
14:11	Output color space	
	0	RGB
	1	YCbCr 4:4:4.
	2	YCbCr 4:2:2.
	3	YCbCr 4:2:0.
	*	RESERVED
15	RESERVED	
19:16	Output colorimetry	
	0	ITU-701
	1	ITU-609
31:20	RESERVED	

5.21.14.TSI_PG_CUSTOM_TIMING_FIELD_RATE

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_TIMING_FIELD_RATE                0x70d
unsigned int custom_timing_field_rate          U32
4 bytes                                        RW
```

Synopsis

Indicates the field rate as mHz (1000 = 1Hz). For non-interlaced timings, the field rate is same as frame rate.

5.21.15.TSI_R_PG_PREDEF_TIMING_COUNT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_PG_PREDEF_TIMING_COUNT                   0x70e
unsigned int predef_timing_count               U32
4 bytes                                        RO
```

Synopsis

Indicates the number of predefined timings available for the current interface. Please notice that the number of timings depends on the type of device being used, and on the interface being used.

5.21.16.TSI_W_PG_PREDEF_TIMING_SELECT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_PREDEF_TIMING_SELECT                    0x70f
unsigned int predef_timing_select              U32
4 bytes                                        WO
```

Synopsis

Selects a predefined timing. Writing into this CI will update the custom timing values as defined by the timing. The custom timing information is therefore made readable to the client application.

Important: *Selecting a custom timing does not apply it to the patter generator. To apply a predefined timing, first select the timing using this CI, and then issue apply new timing command with the 5.21.4 TSI_W_PG_COMMAND CI.*

5.21.17.TSI_PG_PREDEF_PATTERN_COUNT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_PREDEF_PATTERN_COUNT                    0x710
unsigned int predef_pattern_count              U32
4 bytes                                        RO
```

Synopsis

Indicates number of predefined patterns available for the current interface. Please notice that the number of patterns depends on the type of device being used, and of the interface being used.

5.21.18.TSI_W_PG_PREDEF_PATTERN_SELECT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_PREDEF_PATTERN_SELECT                   0x711
unsigned int predef_pattern_select             U32
4 bytes                                        WO
```

Synopsis

Select a predefined pattern for use and/or enumeration.

Important: *Selecting a pattern does not apply it on the pattern generator. To apply a predefined pattern, first select the pattern using this CI, and then issue apply selected predefined pattern command with the 5.21.4 TSI_W_PG_COMMAND CI.*

Important: *Selecting a pattern clears the currently assigned custom pattern and data. If custom pattern data is present, it is always used instead of any predefine pattern.*

5.21.19.TSI_R_PG_PREDEF_PATTERN_NAME

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_PG_PREDEF_PATTERN_NAME                   0x712
char predef_pattern_name[]                     ARRAY_U8
Variable size, max size 256                    RO
```

Synopsis

Contains an ASCII formatted string indicating the name of the currently selected predefined pattern.

5.21.20.TSI_R_PG_PREDEF_PATTERN_ID

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_PG_PREDEF_PATTERN_ID                    0x713
unsigned int predef_pattern_id                U32
4 bytes                                       RO
```

Synopsis

Contains pattern identifier as a computer friendly way. The pattern ID's are used to identify which predefined pattern parameters can be used with which pattern(s).

Important: *The pattern ID's are unique per parameters, meaning that not all patterns have a unique pattern ID.*

5.21.21.TSI_PG_PREDEF_PATTERN_PARAMS

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_PREDEF_PATTERN_PARAMS                  0x714
unsigned int predef_pattern_params            ARRAY_U32
Variable size                                 RW
```

Synopsis

Used to access pattern specific parameters that can be used to alter the appearance of procedurally generated patterns. Please read the 5.21.20 TSI_R_PG_PREDEF_PATTERN_ID CI to determine what type of parameters the pattern accepts. See table below for currently defined parameter sets:

ID	Size (Bytes)	Word Index	Description
0	0		No parameters
1	8	0	Color row width in pixels. Default value is 1
		1	Black row width in pixels. Default value is 1
2	4	0	Color step. Default value is 100
3	4	0	Number of frames. Default value is 1

Important: *Changing the pattern parameters does not change the visible pattern. To change parameters, select the predefined pattern first, then update the parameters and after that issue "apply selected predefined pattern" command on the 5.21.4 TSI_W_PG_COMMAND CI.*

5.21.22.TSI_PG_CUSTOM_PATTERN_WIDTH

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_PATTERN_WIDTH                    0x715
unsigned int custom_pat_width                  U32
4 bytes                                        RW
```

Synopsis

Indicates width of custom pattern bitmap as number of pixels.

5.21.23.TSI_PG_CUSTOM_PATTERN_HEIGHT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_PATTERN_HEIGHT                   0x716
unsigned int custom_pat_height                 U32
4 bytes                                        RW
```

Synopsis

Indicates height of custom pattern bitmap as number of pixels.

5.21.24.TSI_PG_CUSTOM_PATTERN_PIXEL_FORMAT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_PATTERN_PIXEL_FORMAT          0x717
unsigned int custom_pat_pxl_format          U32
4 bytes                                     RW
```

Synopsis

Defines which data format is delivered to TSI through TSI_PG_CUSTOM_PATTERN_DATA.

Important: *This definition is independent from TSI_PG_CUSTOM_TIMING_FLAGS, which defines the actual output video stream format. This setting defines the bitmap data format as it is arranged in computer memory.*

Important: *The correct way of filling up the pixel data is to "left-align" the effective bits. i.e. the most significant bit in pixel data will be most significant bit also in the output video stream regardless of how many bits there are in the video stream output compared to the pixel data.*

Important: *TSI can perform conversions between bit-depths and formats of same color space; However, the provided custom bitmap data must match the color space of the physical output. If these settings do not match, the output is blanked.*

Important: *The YCbCr "4:4:4", "4:2:2" and "4:2:0" data formats have different definitions, but are considered same color-space, and therefore can be converted into each others by TSI.*

Value	Description
0x000	RGB 8:8:8, 24-bit RGB image. Stored as 3 bytes. Lowest memory location stores "R" channel, and highest "B" channel
0x001	RGB 16:16:16, 48-bit RGB image. Stored as 3 little-endian 16-bit words. Lowest memory location stores "R" channel, and highest "B" channel.
0x100	YCbCr 8:8:8, 24-bit YCbCr "4:4:4" image. Stored as 3 bytes. Lowest memory location stores "Y" channel, and highest "Cr" channel
0x101	YCbCr 16:16:16, 48-bit YCbCr "4:4:4" image. Stored as 3 little-endian 16-bit words. Lowest memory location stores "Y" channel, and highest "Cr" channel.
0x200	YCbYCr 8:8:8:8, "16-bit" YCbCr "4:2:2" image. Stored as 4 bytes. Lowest memory location stores first "Y" value, and highest "Cr" channel. Each set of four bytes has data for two pixels.
0x201	YCbYCr 16:16:16:16, "32-bit" YCbCr "4:2:2" image. Stored as 4 little-endian 16-bit words. Lowest memory location stores first "Y" value, and highest "Cr" channel. Each set of four words has data for two pixels.
0x300	YYCbYYCr 8:8:8:8:8:8, "12-bit" YCbCr "4:2:0" image. Stored as 6 bytes. Lowest memory location stores the first "Y" value, and highest memory location stores "Cr" channel. Each set of six bytes has data for four pixels arranged as 2x2 block in output image. Y-values are filled in-order from left to right, and from top to bottom.
0x301	YYCbYYCr 16:16:16:16:16:16, "24-bit" YCbCr "4:2:0" image. Stored as 6 little-endian 16-bit words. Lowest memory location stores the first "Y" value, and highest memory location stores "Cr" channel. Each set of six words has data for four pixels arranged as 2x2 block in output image. Y-values are filled in-order from left to right, and from top to bottom.
0x8000 to 0xffff	ID values reserved for device specific formats. Device specific formats are NEVER converted into any other format, and require a matching setting in TSI_PG_CUSTOM_TIMING_FLAGS to be visible on the output.
*	RESERVED

5.21.25.TSI_PG_CUSTOM_PATTERN_DATA

ClientVersion 11, and higher *<license:TBD>*

```
TSI_PG_CUSTOM_PATTERN_DATA                    0x718
unsigned char pattern_data                    ARRAY_U8
Variable size                                 RW
```

Synopsis

Contains RAW data that makes up the custom pattern as defined in CI's TSI_PG_CUSTOM_PATTERN_WIDTH, TSI_PG_CUSTOM_PATTERN_HEIGHT and TSI_PG_CUSTOM_PATTERN_PIXEL_FORMAT. If any valid custom pattern data is assigned, it will be used when the timing/pattern is applied next time by issuing update command by writing into TSI_W_PG_COMMAND.

Important: *Writing into TSI_W_PG_PREDEF_TIMING_SELECT will clear custom pattern definitions and data. The custom pattern can also be cleared by writing zero bytes of data into TSI_PG_CUSTOM_PATTERN_DATA.*

5.22.Displayport interface specific CI definitions

This section lists all CI's that are available for reading status or setting parameters for DisplayPort source capable interfaces. Display TX related CI's are allocated in renage from 0x780 to 0x7ff.

5.22.1.TSI_SRC_DP_LINK_CFG_LANES

ClientVersion 11, and higher *<license:TBD>*

```
TSI_SRC_DP_LINK_CFG_LANES                    0x780
unsigned int dp_src_lanes                    U32
4 bytes                                      RW
```

Synopsis

Set the number lanes to be used on the DisplayPort link. Please note that writing this CI has no immediate effect. The configured value is used the next time there is a link training, or when the command to apply setting without LT is issued. Valid settings are 1, 2 and 4 lanes.

5.22.2.TSI_SRC_DP_LINK_CFG_BIT_RATE

ClientVersion 11, and higher *<license:TBD>*

```
TSI_SRC_DP_LINK_CFG_BIT_RATE                 0x781
unsigned int dp_src_bit_rate                 U32
4 bytes                                      RW
```

Synopsis

Defines the bit rate as multiplier of 0.27Gbps. Please note that writing this CI has no immediate effect. The configured value is used the next time there is a link training, or when the command to apply setting without LT is issued. Valid settings are 6, 10 and 20.

5.22.3.TSI_SRC_DP_LINK_CFG_FLAGS

ClientVersion 11, and higher *<license:TBD>*

```
TSI_SRC_DP_CFG_FLAGS                          0x782
unsigned int dp_src_framing                   U32
4 bytes                                       RW
```

Synopsis

Select misc. features for the DP link. Please note that writing this CI has no immediate effect. The configured value is used the next time there is a link training. See table below for defined flags:

Bits	Description	
0	RESERVED	
1	TPS3	
	0	Disable TPS3
	1	Enable TPS3
2	Framing selection	
	0	Standard framing
	1	Enhanced framing
31:3	RESERVED	

5.22.4.TSI_SRC_DP_OVERRIDE_VOLTAGE_SWING

ClientVersion 11, and higher *<license:TBD>*

```
TSI_SRC_DP_OVERRIDE_VOLTAGE_SWING              0x783
unsigned int dp_src_ovr_voltage_swing          U32
4 bytes                                        RW
```

Synopsis

Writing this CI will override the voltage swing values. If no override values have been written reading the CI will fail. See table below for definitions:

Bits	Descirption	
7:0	Voltage swing for Lane 0	
	0	400 mVpp
	1	600 mVpp
	2	800 mVpp
	3	1200 mVpp
	*	RESERVED
15:8	Voltage swing for Lane 1. The bit-field values are the same as with Lane 0 (See above).	
23:16	Voltage swing for Lane 2. The bit-field values are the same as with Lane 0 (See above).	
31:24	Voltage swing for Lane 3. The bit-field values are the same as with Lane 0 (See above).	

5.22.5.TSI_SRC_DP_OVERRIDE_PRE_EMPHASIS

ClientVersion 11, and higher *<license:TBD>*

```
TSI_SRC_DP_OVERRIDE_PRE_EMPHASIS                0x784
unsigned int dp_src_ovr_pre_emphasis            U32
4 bytes                                         RW
```

Synopsis

Writing this CI will override the pre emphasis values. Reading the CI will return the previous values written. See table below for definitions:

Bits	Description	
7:0	Pre-emphasis setting for Lane 0.	
	0	0 dB pre-emphasis
	1	3.5 dB pre-emphasis
	2	6 dB pre-emphasis
	3	9.5 dB pre-emphasis
	*	RESERVED
15:8	Pre-emphasis setting for Lane 1. The bit-field values are the same as with Lane 0 (See above).	
23:16	Pre-emphasis setting for Lane 2. The bit-field values are the same as with Lane 0 (See above).	
31:24	Pre-emphasis setting for Lane 3. The bit-field values are the same as with Lane 0 (See above).	

5.22.6.TSI_SRC_DP_LINK_PATTERN

ClientVersion 11, and higher *<license:TBD>*

```
TSI_SRC_DP_LINK_PATTERN                          0x785
unsigned int dp_src_lt_pattern                   U32
4 bytes                                          RW
```

Synopsis

Force device to output a pattern that is typically used only with Link Training. See table below for options:

ID	Description
0	Active video
1	Idle pattern
2	Training pattern 1
3	Training pattern 2
4	Training pattern 3
5	Training pattern 4
6	PRBS7
7	HBR2
8	SER

5.22.7.TSI_W_SRC_DP_COMMAND

ClientVersion 11, and higher *<license:TBD>*

```
TSI_W_SRC_DP_COMMAND                             0x786
unsigned int dp_src_cmd                          U32
4 bytes                                          WO
```

Synopsis

Carry out commands on the source. See table below for available commands:

ID	Description
0	No operation. Writing this has no effect.
1	Begin link-training.
2	Apply bit-rate and lane count settings without performing link-training.

5.22.8.TSI_R_SRC_DP_HPD_STATUS

ClientVersion 11, and higher *<License:TBD>*

```
TSI_R_SRC_DP_HPD_STATUS                          0x787
unsigned int dp_src_HPD_status                   U32
4 bytes                                          RO
```

Synopsis

Used to read HPD signal logical status. (0 = HPD de-asserted, 1 = HPD asserted).

5.22.9.TSI_R_SRC_DP_LT_RESULT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_SRC_DP_LT_RESULT                           0x788
unsigned int dp_src_lt_res                       U32
4 bytes                                          RO
```

Synopsis

Contains result of previous link training. See table below for values:

Bits	Description	
7:0	Link training procedure result	
	0	Link training not started
	1	Link training in progress
	2	Link training failed.
	3	Link training succeeded.
	*	RESERVED
31:8	RESERVED	

5.22.10.TSI_R_SRC_DP_LINK_STATUS_BITS

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_SRC_DP_LINK_STATUS_BITS                    0x789
unsigned int dp_src_lt_status                    U32
4 bytes                                          RO
```

Synopsis

Indicates Clock Recovery, Channel equalization and symbol locks states for each lane. The status is the result of the previous link training. See table below for bit definitions:

Bits	Description
0	Clock Recovery done for Lane 0.
1	Channel EQ done for Lane 0.
2	Symbol lock for Lane 0.
3	RESERVED
4	Clock Recovery done for Lane 1.
5	Channel EQ done for Lane 1.
6	Symbol lock for Lane 1.
7	RESERVED
8	Clock Recovery done for Lane 2.
9	Channel EQ done for Lane 2.
10	Symbol lock for Lane 2.
11	RESERVED
12	Clock Recovery done for Lane 3.
13	Channel EQ done for Lane 3.
14	Symbol lock for Lane 3.
30:15	RESERVED
31	ILA

5.22.11.TSI_R_SRC_DP_LINK_STATUS_VOLT_SWING

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_SRC_DP_LINK_STATUS_VOLT_SWING          0x78a
unsigned int dp_src_lt_voltage_swing         U32
4 bytes                                      RO
```

Synopsis

Indicates voltage swing values for each lane. The state is the result of the previous link training. See table below for definitions:

Bits	Descirption	
7:0	Voltage swing for Lane 0	
	0	400 mVpp
	1	600 mVpp
	2	800 mVpp
	3	1200 mVpp
	*	RESERVED
15:8	Voltage swing for Lane 1. The bit-field values are the same as with Lane 0 (See above).	
23:16	Voltage swing for Lane 2. The bit-field values are the same as with Lane 0 (See above).	
31:24	Voltage swing for Lane 3. The bit-field values are the same as with Lane 0 (See above).	

5.22.12.TSI_R_SRC_DP_LINK_STATUS_LANE_COUNT

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_SRC_DP_LINK_STATUS_LANE_COUNT          0x78b
unsigned int dp_src_lane_count               U32
4 bytes                                      RO
```

Synopsis

Indicates number of lanes achieved in the previous link training.

5.22.13.TSI_R_SRC_DP_LINK_STATUS_BIT_RATE

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_SRC_DP_LINK_STATUS_BIT_RATE              0x78c
unsigned int dp_src_bit_rate                   U32
4 bytes                                        RO
```

Synopsis

Indicates link bit-rate achieved during the previous link training as multiple of 0.27Gbps.

5.22.14.TSI_R_SRC_DP_LINK_STATUS_PRE_EMP

ClientVersion 11, and higher *<license:TBD>*

```
TSI_R_SRC_DP_LINK_STATUS_PRE_EMP               0x78d
unsigned int dp_src_pre_emp                    U32
4 bytes                                        RO
```

Synopsis

Indicates the pre-emphasis setting achieved during the previous link training. See table below for definitions:

Bits	Description	
7:0	Pre-emphasis setting for Lane 0.	
	0	0 dB pre-emphasis
	1	3.5 dB pre-emphasis
	2	6 dB pre-emphasis
	3	9.5 dB pre-emphasis
	*	RESERVED
15:8	Pre-emphasis setting for Lane 1. The bit-field values are the same as with Lane 0 (See above).	
23:16	Pre-emphasis setting for Lane 2. The bit-field values are the same as with Lane 0 (See above).	
31:24	Pre-emphasis setting for Lane 3. The bit-field values are the same as with Lane 0 (See above).	

5.23.CEC Functional test

This section defines CEC functional test related CI's.

Define	Config ID	Default	Description	Reference
TSI_HDMI_RX_CEC_TIMEOUT	0x10400	5000	HDMI CEC functional test timeout, in milliseconds	5.23.1
TSI_HDMI_RX_CEC_LOCAL_PHY_ADDR	0x10401	0x4000	Local PHY Address.	5.23.2

5.23.1.TSI_HDMI_RX_CEC_TIMEOUT

```
TSI_HDMI_RX_CEC_TIMEOUT                          0x10400
unsigned int cec_timeout                         U32
4 bytes                                          RW
```

Synopsis

Defines the CEC functional test timeout, in milliseconds. The test must complete within this time-period in order to succeed. Default setting is 5000ms.

5.23.2.TSI_HDMI_RX_CEC_LOCAL_PHY_ADDR

```
TSI_HDMI_RX_CEC_LOCA_PHY_ADDR                    0x10401
unsigned int cec_phy_addr                        U32
4 bytes                                          RW
```

Synopsis

Defines the CEC local PHY address. The address is stored in lowest 16-bits. Default setting is 0x4000 ("4.0.0.0").

```
Typically these addresses are given as "A.B.C.D", similar to IP addresses.
Each number in the address can be a value between 0 and 15.
Therefore, address "8.9.10.11" would become HEX value 0x000089AB.
```

5.24.HDMI Source specific

This section defines the CI's used to control HDMI Source side link and to see status of the link.

Define	Config ID	Default	Description	Reference

5.24.1.TSI_W_SRC_HDMI_CONTROL

```
TSI_W_SRC_HDMI_CONTROL                          0x800
unsigned int src_hdmi_ctrl                      U32
4 bytes                                         WO
```

Synopsis

Control bits for HDMI source side interfaces. See bit definitions below:

Bits	Description
0	Scrambler enable: 1 = Enabled, 0 = Disabled.
1	Link Mode. 0 = 3G, 1 = 6G
2	Port Mode. 0 = HDMI, 1 = DVI
3	RESERVED (Set to zero).
4	HDMI Behavior. 0 = HDMI 2.0 source behavior, 1 = HDMI 1.4 source behavior
31:5	RESERVED (Set to zero).

5.24.2.TSI_R_SRC_HDMI_STATUS

```
TSI_R_SRC_HDMI_STATUS                          0x801
unsigned int src_hdmi_status                   U32
4 bytes                                        RO
```

Synopsis

Status bits for the source side HDMI link

Bits	Description
0	Scrambler state: 0 = Disabled, 1 = Enabled.
1	Link mode: 0 = 3G mode, 1 = 6G mode.
2	Port mode: 0 = HDMI mode, 1 = DVI Mode.
3	Video: 0 = Video is disabled, 1 = Video is enabled.
4	Behavior: 0 = HDMI 2.0 Source behavior, 1 = HDMI 1.4 Source behavior.
5	HPD state: 0 = HPD De-asserted, 1 = HPD Asserted.
31:6	RESERVED (Set to zero).

5.24.3.TSI_R_SRC_HDMI_DUT_CAPS

```
TSI_R_SRC_HDMI_DUT_CAPS                        0x802
unsigned int src_hdmi_dut_caps                 U32
4 bytes                                        RO
```

Synopsis

Capabilities of the connected DUT. This CI is updated whenever the HDP line becomes asserted, and cleared if HPD line is de-asserted. The CI will also be cleared when behavior is set to HDMI 1.4.

Bits	Description
0	SCDC Support: 0 = SCDC not supported, 1 = SCDC supported.
1	Scrambler support: 0 = Scrambler not supported, 1 = Scrambler supported.
31:2	RESERVED (Set to zero).

5.25.HDMI Sink specific

This section defines CI's specific to HDMI Sink side link status and control.

5.25.1.TSI_R_HDRX_LINK_STATUS

```
TSI_R_HDRX_LINK_STATUS                          0x810
unsigned int sink_hdmi_link_status              U32
4 bytes                                         RO
```

Synopsis

Link status bits.

Bits	Description
0	TMDS Clock. 0 = No clock detected, 1 = Clock detected.
1	Clock rate: 0 = 3G, 1 = 6G
2	Input stream lock status. 0 = Not locked, 1 = Locked
3	Port mode: 0 = HDMI, 1 = DVI
6:4	Line X lock bits. Bit 4 for line 0, bit 5 for line 1 and bit 6 for line 2. 0 = No lock, 1 = Locked.
7	HPD State: 0 = HPD De-asserted, 1 = HPD Asserted.
31:8	RESERVED (Set to zero).

5.25.2.TSI_W_HDRX_LINK_CONTROL

```
TSI_W_HDRX_LINK_CONTROL                         0x811
unsigned int sink_hdmi_link_ctrl                U32
4 bytes                                         WO
```

Synopsis

Link control bits.

Bits	Description
0	HPD Control. 0 = Set HDP as De-asserted, 1 = Set HPD as Assrted.
31:1	RESERVED (Set to zero).

5.26.Error codes

-0: **TSI_SUCCESS**: A generic success indication.

-1: **TSI_ERROR_NOT_INITIALIZED**: The TSI API is not properly initialized for operations.

-2: **TSI_ERROR_COMPATIBILITY_MISMATCH**: The given Client version ID is different from the one provided with first call to TSI_Init(). The client application must always use the same Client version ID. Please make sure that the versions of TSI.C, TSI.H and TSI_Types.h match exactly. The version of these files is listed on the second line of the source code.

-3: **TSI_ERROR_NOT_COMPATIBLE**: The given Client version ID is not supported by the loaded API version. Since API is backward compatible with older applications, it means that the application is built for a later version of the API.

-4: **TSI_ERROR_DLL_NOT_FOUND**: Either the TSI.DLL is not found, or one of the lower level API DLLs was not found. Please try to re-install the TSI software package.

-5: **TSI_ERROR_DLL_VERSION_READ**: A failure has occurred while reading version data from a PE Executable file. The file might be corrupted on unreadable or otherwise not useable.

-6: **TSI_ERROR_OUT_OF_MEMORY**: A generic error message indicating a problem when memory was being allocated. Make sure your application is not leaking memory resources. Also remember that 32-bit process can only allocate up to about 2 GB of RAM – the system reserves part of the max. 4GB address space for itself per process.

-7: **TSI_ERROR_FUNCTION_NOT_FOUND**: A required function was not found in a DLL.

-8: **TSI_ERROR_ACCESS_DENIED**: An operation was attempted that requires a license key to be installed, but the license key is not installed for the device being used.

-9: **TSI_ERROR_NO_REFERENCES**: A reference counted item is already at zero references, or the item is already destroyed and can't have any references.

-10: **TSI_ERROR_DEVICE_INDEX_OUT_OF_RANGE**: No device present with the given device index.

-11: **TSI_ERROR_INVALID_PARAMETER**: One or more of the parameters passed to the function are invalid.

-12: **TSI_ERROR_INPUT_ENABLED**: The requested operation is not available while input is enabled.

-13: **TSI_ERROR_INPUT_DISABLED**: The requested operation is not available while input is disabled.

-14: **TSI_ERROR_INPUT_ENABLE_FAILED**: Failed to enable input due to resource allocation problems.

-15: **TSI_ERROR_OPEN_DEVICE**: Failed to open requested device. Usually happens when a previous application fails to exit properly.

-16: **TSI_ERROR_INPUT_SELECT**: Failed to select input.

-17: **TSI_ERROR_NOT_IMPLEMENTED**: The requested function is not implemented in current version.

-18: **TSI_ERROR_UNEXPECTED_ITEM_SIZE**: Get or Set configuration item function: The configuration item's size was unexpected. (Please refer to configuration item details for correct size information).

-19: **TSI_ERROR_UNSUPPORTED_CONFIG_ID**: Get or Set configuration item function: The given configuration item is not supported with the current hardware, or the ID is unknown.

-20: **TSI_ERROR_CONFIGURATION_ITEM_NOT_SET**: Get configuration item function: The given configuration function has no value assigned to it at the moment.

-21: **TSI_ERROR_FILE_CREATE**: Failed to create save file.

-22: **TSI_ERROR_FILE_WRITE**: Failed to write into a file.

-23: **TSI_ERROR_FILE_OPEN**: Failed to open an existing file.

-24: **TSI_ERROR_FILE_READ**: Failed to read from a file.

-25: **TSI_ERROR_INVALID_FILE**: Unsupported file format.

-26: **TSI_ERROR_DATA_CORRUPTED**: File contents are corrupted or the file is partial.

-27: **TSI_ERROR_FORMAT_MISMATCH**: Reference frame does not match incoming video signal.

-28: **TSI_ERROR_INVALID_TEST_MODE**: Requested testing mode is not supported.

-29: **TSI_ERROR_COMPARE_FAILED**: Test procedure failed – Test outcome is not determined.

-30: **TSI_ERROR_NO_REFERENCE_FRAME**: Can't start test because no reference frames are set.

-31: **TSI_ERROR_TIMEOUT**: An asynchronous operation is taking too long, and was aborted after a timeout event occurred.

-32: **TSI_ERROR_NO_DATA_AVAILABLE**: The requested data is not available. Please note that data can become available without intervention from client software. (For example video timing configuration items).

-33: **TSI_ERROR_CONFIG_ITEM_ACCESS**: Configuration item read or write failed.

-34: **TSI_ERROR_PRESENT_GRAPHICS**: Failed to show graphics preview frame/modify preview area properties.

-35: **TSI_ERROR_UNSUPPORTED_FORMAT**: The captured format is not supported.

-36: **TSI_ERROR_DEVICE_SPECIFIC**: An unexpected problem occurred in the device specific software component.

-37: **TSI_ERROR_DISK_FILE_IO**: A problem occurred while reading or writing to a file.

-38: **TSI_ERROR_INTERNAL**: Internal state is invalid, or some other internal issue.

-39: **TSI_ERROR_CONFIGURATION_ITEM_VALUE**: Attempted to set a configuration item to a value that is not allowed.

-40: **TSI_ERROR_CAPTURE_BROKEN**: Capture (Video, audio or other signal) failed and was re-started during testing.

-41: **TSI_ERROR_OS_ERROR**: An OS function call has failed. Note that some OS function call failures have specific error codes, like TSI_ERROR_FILE_CREATE.

-42: **TSI_ERROR_DATA_PROTECTION_ENABLED**: Video or Audio data is HDCP protected and can't be used for testing or saving.

-43: **TSI_ERROR_TEST_REQUIREMENTS_NOT_MET**: Test requirements were not met by the capture device.

-44: **TSI_ERROR_UNSUPPORTED_COLORSPACE**: The frame color space is not supported by the function

-45: **TSI_ERROR_NO_DEVICE_SELECTED:** No device is currently selected, and the attempted operation requires a device to be selected.

-46: **TSI_ERROR_INVALID_ACCESS_MODE:** Attempted to read a write-only CI, or write a read-only CI.

-47: **TSI_ERROR_OUTPUT_ENABLED:** The attempted operation is not available if source function is enabled.

-48: **TSI_ERROR_OPERATION_DATA_LOSS:** The attempted operation would have caused unwanted data loss. (For example, saving into a file that cannot support the necessary color depth of the reference data).

-49: **TSI_ERROR_BAD_FW_VERSION:** The firmware version on the device being selected is not compatible with the current TSI and/or device software.