Agilent U4305A Protocol Exerciser PCIe Port API

The PCIe port API allows you to program the Agilent U4305A Protocol Exerciser for PCI Express from the system under test (SUT). It comes as a separate installation program.

The PCIe port API adds a third option for programming the exerciser to the existing COM / TCL API and the Graphical user interface. The PCIe port API is intended for programming the exerciser directly from the SUT, where CPU interaction between test program and exerciser is needed. Communication with the exerciser is handled by the PCI Express port between exerciser and system, therefore eliminating the need for a separate LAN connection on the SUT.

Note:

The PCIe port API has generally slower performance than the COM/TCL port, so it should only be used when CPU interaction with the DUT is critical. Also, if testing on the DLL/Phy layer (error insertion, link width negotiation etc.) is intended, the COM/TCL API should be used, as manipulating the link will cause the in-system port to lose the connection to the exerciser.

Differences between COM/TCL and PCIe port API:

The PCIe port API is not using COM. Therefore, the COM initialization calls (CoInitialize, etc.) that are required with the system API are not applicable. Also, the Session and PortSelector interfaces are not available in the PCIe port API. The user must create a session, add the required port(s) and bring up the link using the system API before any of the PCIe port API calls can be used.

The first argument in most system API functions is the 'portHandle' which is used as a reference to a particular port within a session is used by PCIe port API functions as well. Unlike the system API where the portHandle is typically derived from a call to PortSelector->AddPort, the in-system API provides a new function: ConnectPort (physicalAddress) which returns a portHandle that can be used in subsequent calls to PCIe port API functions.

PCIe port API calls are using the physical PCI Express link between the exerciser and the system under test, therefore all functions that bring down the physical link will cut the connection between the PCIe port API library and the exerciser.

Changes in APIs:

- New API functions for reading and writing scratch pad registers have been added:
  - void ScratchPadRegisterGet(
    /* [in] */ AgtPortHandleT portHandle,
    /* [in] */ EPCIEScratchPadRegister registerNum,
    /* [retval][out] */ AgtValueT *value)

Reads the contents of a scratch pad register. ‘registerNum’ is the register number from which the values have to be read. The possible values are:

- PCIE_SCRATCH_PAD_REGISTER0: Scratch Pad Register 0
- PCIE_SCRATCH_PAD_REGISTER1: Scratch Pad Register 1
- PCIE_SCRATCH_PAD_REGISTER2: Scratch Pad Register 2
- PCIE_SCRATCH_PAD_REGISTER3: Scratch Pad Register 3

- void ScratchPadRegisterSet(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEScratchPadRegister registerNum,
  /* [in] */ AgtValueT value)

  Writes the values specified in ‘value’ to the scratch pad register specified in ‘registerNum’.

- void void RunFunctions(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ AgtValueT functions)

  Start transferring data from the Individual requester block memory. Please note that changes have been made to the interpretation of the parameter ‘functions’ in order to support the newly added functions. Please refer to PCIE_Exerciser_API_Online_Help.chm for further details.

- void StopFunctions(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ AgtValueT functions)

  Stop transferring data from the Individual requester block memory. Please note that changes have been made to the interpretation of the parameter ‘functions’ in order to support the newly added functions. Please refer to PCIE_Exerciser_API_Online_Help.chm for further details.

- void StartDebugLogging(
  /* [in] */ char* strFileName_p)

  Starts writing debug information to the specified log file.

- void FTRegRead(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ AgtSizeT offset,
  /* [retval][out] */ AgtValueT *val)

  Reads a DWORD from the MRIOV function table. This function now supports reading of function table values for VH2, VH3, VH4 entries. Please refer to PCIE_Exerciser_API_Online_Help.chm for further details.

- void FTRegWrite(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ AgtSizeT offset,
  /* [in] */ AgtValueT val)

  Writes a DWROD to the MRIOV function table. This function now supports writing of function table values for VH2, VH3, VH4 entries. Please refer to PCIE_Exerciser_API_Online_Help.chm for further details.

- Support for 2 new PFs and 3 new virtual hierarchies have been added to the API methods and enumerations. The following enumerations have been added to support the new PFs and VHs:
  - EPCIECompQueue
    - PCIE_COMPQUEUE_2
    - PCIE_COMPQUEUE_3
    - PCIE_COMPQUEUE_4
  - EPCIEHwChannelFunction
    - PCIE_HWCHANNEL_FUNCTION_D
    - PCIE_HWCHANNEL_FUNCTION_E
    - PCIE_HWCHANNEL_FUNCTION_DVF1
    - PCIE_HWCHANNEL_FUNCTION_DVF2
    - PCIE_HWCHANNEL_FUNCTION_EVF1
    - PCIE_HWCHANNEL_FUNCTION_EVF2
  - EPCIEVCResourceId
    - PCIE_VC_RESOURCE_2
    - PCIE_VC_RESOURCE_3
    - PCIE_VC_RESOURCE_4
  - EPCIEExerciser
    - PCIE_EXERCISER_MRIOV_VH2_ENABLE
    - PCIE_EXERCISER_MRIOV_VH3_ENABLE
    - PCIE_EXERCISER_MRIOV_VH4_ENABLE
  - EPCIEExerciserStatus
    - PCIE_EXERCISERSTATUS_HWCHANNEL_D_STATE
    - PCIE_EXERCISERSTATUS_HWCHANNEL_E_STATE
    - PCIE_EXERCISERSTATUS_HWCHANNEL_DVF1_STATE
    - PCIE_EXERCISERSTATUS_HWCHANNEL_DVF2_STATE
    - PCIE_EXERCISERSTATUS_HWCHANNEL_EVF1_STATE
    - PCIE_EXERCISERSTATUS_HWCHANNEL_EVF2_STATE
    - PCIE_EXERCISERSTATUS_MRIOV_VH2_INITIALIZED
    - PCIE_EXERCISERSTATUS_MRIOV_VH3_INITIALIZED
    - PCIE_EXERCISERSTATUS_MRIOV_VH4_INITIALIZED
    - PCIE_EXERCISERSTATUS_HWCHANNEL_D_OUTSTANDING_REQUESTS
    - PCIE_EXERCISERSTATUS_HWCHANNEL_E_OUTSTANDING_REQUESTS
    - PCIE_EXERCISERSTATUS_HWCHANNEL_DVF1_OUTSTANDING_REQUESTS
    - PCIE_EXERCISERSTATUS_HWCHANNEL_DVF2_OUTSTANDING_REQUESTS
    - PCIE_EXERCISERSTATUS_HWCHANNEL_EVF1_OUTSTANDING_REQUESTS
    - PCIE_EXERCISERSTATUS_HWCHANNEL_EVF2_OUTSTANDING_REQUESTS

- void CAgtPCIEExerciser::PerformanceCounterStatusReadAll(
  /* [in] */ AgtPortHandleT portHandle,
  /* [retval][out] */ AgtValueT *val)

  This function takes a snapshot and reads all the performance counter values in one single call. The function internally locks ISP access, takes a snapshot, reads all the values and then unlocks ISP access. The values are returned in an array that the user needs to allocate before calling this function. The array should be of type unsigned integer (32 bit wide) and of length 130, that is, 130 x 4 bytes. Please note that this function now supports the newly added functions and virtual hierarchies and the order of performance counter values is different from previous version. Please refer to the information mentioned in this document for the current order. The values are stored in the following order:

	Index	
	Low value	**High value**
PCIE_PERFORMANCECOUNTERSTATUS_INTERVAL_LEN	0	1
PCIE_PERFORMANCECOUNTERSTATUS_DLLPHY_CODING_ERR	2	3
PCIE_PERFORMANCECOUNTERSTATUS_DLLPHY_DISPARITY_ERR	4	5
PCIE_PERFORMANCECOUNTERSTATUS_DLLPHY_128B_130B_ERR	6	7
PCIE_PERFORMANCECOUNTERSTATUS_SUCCESSFUL_LINK_TRAININGS	8	9
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_ACK	10	11
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_NAK	12	13
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_TX_NAK	14	15
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 0 / VC 0)	16	17
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 1 / VC 1)	18	19
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 2)	20	21
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 3)	22	23
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 4)	24	25
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VL 0)	26	27
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function A)	28	29
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function B)	30	31
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function C)	32	33
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function D)	34	35
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function E)	36	37
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function BVF1)	38	39
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function BVF2)	40	41
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function CVF1)	42	43
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function CVF2)	44	45
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function DVF1)	46	47
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function DVF2)	48	49
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function EVF1)	50	51
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Function EVF2)	52	53
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function A)	54	55
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function B)	56	57
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function C)	58	59
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function D)	60	61

PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function E)	62	63
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function BVF1)	64	65
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function BVF2)	66	67
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function CVF1)	68	69
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function CVF2)	70	71
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function DVF1)	72	73
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function DVF2)	74	75
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function EVF1)	76	77
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Function EVF2)	78	79
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function A)	80	81
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function B)	82	83
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function C)	84	85
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function D)	86	87
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function E)	88	89
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function BVF1)	90	91
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function BVF2)	92	93
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function CVF1)	94	95
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function CVF2)	96	97
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function DVF1)	98	99
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function DVF2)	100	101
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function EVF1)	102	103
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Function EVF2)	104	105
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function A)	106	107
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function B)	108	109
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function C)	110	111
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function D)	112	113
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function E)	114	115
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function BVF1)	116	117
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function BVF2)	118	119
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function CVF1)	120	121
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function CVF2)	122	123
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function DVF1)	124	125
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function DVF2)	126	127
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function EVF1)	128	129
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Function EVF2)	130	131

- void CAgtPCIEExerciser::PerformanceCounterStatusReadFunctionAll(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEHwChannelFunction functionNum,
  /* [retval][out] */ AgtValueT *val)

  This function takes a snapshot and reads the performance counter values in one single call. This function reads the all the global values, all virtual channel / virtual hierarchy specific values and also for the specified function. The function internally locks ISP access, takes a snapshot, reads the values and then unlocks ISP access. The values are returned in an array that the user needs to allocate before calling this function. The array should be of type unsigned integer (32 bit wide) and of length

34, that is, 34 x 4 bytes. Please note that this function now supports the newly added functions and virtual hierarchies and the order of performance counter values is different from previous version. Please refer to the information mentioned in this document for the current order. The values are stored in the following order:

	Index	
	Low value	**High value**
PCIE_PERFORMANCECOUNTERSTATUS_INTERVAL_LEN	0	1
PCIE_PERFORMANCECOUNTERSTATUS_DLLPHY_CODING_ERR	2	3
PCIE_PERFORMANCECOUNTERSTATUS_DLLPHY_DISPARITY_ERR	4	5
PCIE_PERFORMANCECOUNTERSTATUS_DLLPHY_128B_130B_ERR	6	7
PCIE_PERFORMANCECOUNTERSTATUS_SUCCESSFUL_LINK_TRAININGS	8	9
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_ACK	10	11
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_NAK	12	13
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_TX_NAK	14	15
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 0 / VC 0)	16	17
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 1 / VC 1)	18	19
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 2)	20	21
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 3)	22	23
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VH 4)	24	25
PCIE_PERFORMANCECOUNTERSTATUS_DLLP_FC (VL 0)	26	27
PCIE_PERFORMANCECOUNTERSTATUS_TLP_NUM (Specified Function)	28	29
PCIE_PERFORMANCECOUNTERSTATUS_DW_NUM (Specified Function)	30	31
PCIE_PERFORMANCECOUNTERSTATUS_TX_TLP_NUM (Specified Function)	32	33
PCIE_PERFORMANCECOUNTERSTATUS_TX_DW_NUM (Specified Function)	34	35

- Support for separate data memory compare for each function has been added. Enumerations have been changed / added to support this. Also 2 new APIs have been added.

  The following enumerations have changed:
  - EPCIEExerciser
    - PCIE_EXERCISER_DATACMP_COMPLETIONS_ENABLE has been renamed to PCIE_EXERCISER_DATACMP_HWCHANNEL_A_COMPLETIONS_ENABLE.
  - EPCIEExerciserStatus
    - PCIE_EXERCISERSTATUS_DATAMEMCMP_INTADDR has been renamed to PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_A_INTADDR
    - PCIE_EXERCISERSTATUS_DATAMEMCMP_DATA_EXPECTED has been renamed to PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_A_DATA_EXPECTED
    - PCIE_EXERCISERSTATUS_DATAMEMCMP_DATA_REAL has been renamed to PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_A_DATA_REAL
    - PCIE_EXERCISERSTATUS_DATAMEMCMP_ERROR_OCCURRED has been renamed to PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_A_ERROR_OCCURRED
    - PCIE_EXERCISERSTATUS_DATAMEMCMP_ERROR_COUNT has been renamed to PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_A_ERROR_COUNT

The following new enumerations have been added:

- EPCIEExerciser
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_B_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_C_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_D_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_E_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_BVF1_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_BVF2_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_CVF1_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_CVF2_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_DVF1_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_DVF2_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_EVF1_COMPLETIONS_ENABLE
  - PCIE_EXERCISER_DATACMP_HWCHANNEL_EVF2_COMPLETIONS_ENABLE
- EPCIEExerciserStatus
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_B_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_C_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_D_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_E_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF1_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF2_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF1_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF2_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF1_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF2_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF1_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF2_INTADDR
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_B_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_C_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_D_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_E_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF1_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF2_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF1_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF2_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF1_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF2_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF1_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF2_DATA_EXPECTED
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_B_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_C_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_D_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_E_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF1_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF2_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF1_DATA_REAL
  - PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF2_DATA_REAL

- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF1_DATA_REAL
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF2_DATA_REAL
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF1_DATA_REAL
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF2_DATA_REAL
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_B_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_C_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_D_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_E_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF1_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF2_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF1_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF2_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF1_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF2_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF1_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF2_ERROR_OCCURRED
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_B_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_C_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_D_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_E_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF1_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_BVF2_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF1_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_CVF2_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF1_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_DVF2_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF1_ERROR_COUNT
- PCIE_EXERCISERSTATUS_DATAMEMCMP_HWCHANNEL_EVF2_ERROR_COUNT

The following new APIs have been added:

- void ExerciserStatusFunctionReset(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEHwChannelFunction functionNum)

  Resets the exerciser status values for specified function.

- void ExerciserStatusFunctionSnapshot(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEHwChannelFunction functionNum)

  Takes a snapshot of the Exerciser status properties for specified function.

- void ExerciserPhyStatusRead(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEExerciserPhyStatus status,
  /* [in] */ AgtSizeT lane_number,
  /* [retval][out] */ AgtValueT *val);

Read the Phy status of the exerciser based on status type and lane number

EPCIEExerciserPhyStatus

- PCIE_EXERCISERPHYSTATUS_TRANSMITTER_SETTING_RECEIVED_PRESET
- PCIE_EXERCISERPHYSTATUS_TRANSMITTER_SETTING_RECEIVED_C_MINUS_1
- PCIE_EXERCISERPHYSTATUS_TRANSMITTER_SETTING_RECEIVED_C_ZERO
- PCIE_EXERCISERPHYSTATUS_TRANSMITTER_SETTING_RECEIVED_C_PLUS_1
- PCIE_EXERCISERPHYSTATUS_TRANSMITTER_SETTING_RECEIVED_REJECT
- PCIE_EXERCISERPHYSTATUS_COEFFICIENT_REQUEST_FROM_EXERCISER_PRESET
- PCIE_EXERCISERPHYSTATUS_COEFFICIENT_REQUEST_FROM_EXERCISER_C_MINUS_1
- PCIE_EXERCISERPHYSTATUS_COEFFICIENT_REQUEST_FROM_EXERCISER_C_ZERO
- PCIE_EXERCISERPHYSTATUS_COEFFICIENT_REQUEST_FROM_EXERCISER_C_PLUS_1
- PCIE_EXERCISERPHYSTATUS_COEFFICIENT_REQUEST_FROM_EXERCISER_REJECT

- void CAgtPCIEExerciser::ExerciserChannelFunctionSet(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEHwChannelFunction functionNum,
  /* [in] */ EPCIEExerciserHwChannelStatus status,
  /* [in] */ AgtValueT val
)

- void CAgtPCIEExerciser::ExerciserChannelFunctionGet(
  /* [in] */ AgtPortHandleT portHandle,
  /* [in] */ EPCIEHwChannelFunction functionNum,
  /* [in] */ EPCIEExerciserHwChannelStatus status,
  /* [retval][out] */ AgtValueT* val
)

EPCIEExerciserHwChannelStatus

- PCIE_EXERCISER_GENERATOR_PREFIX,
- PCIE_EXERCISERSTATUS_DATAGENCMP_HEADER_PREFIX,
- PCIE_EXERCISERSTATUS_DATAGENCMP_HEADER0,
- PCIE_EXERCISERSTATUS_DATAGENCMP_HEADER1,
- PCIE_EXERCISERSTATUS_DATAGENCMP_HEADER2,
- PCIE_EXERCISERSTATUS_DATAGENCMP_HEADER3,
- PCIE_EXERCISERSTATUS_DATAGENCMP_DWOFFSET,
- PCIE_EXERCISERSTATUS_DATAGENCMP_DATA_EXPECTED,
- PCIE_EXERCISERSTATUS_DATAGENCMP_DATA_REAL,
- PCIE_EXERCISERSTATUS_DATAGENCMP_BYTEN,
- PCIE_EXERCISERSTATUS_DATAGENCMP_ERROR_OCCURRED

- Support for enabling / disabling individual pattern terms has been added. The following values have been added in the EPCIEExerciser enumeration for this purpose:

- PCIE_EXERCISER_PATTERN_TERM_0_ENABLE
- PCIE_EXERCISER_PATTERN_TERM_1_ENABLE
- PCIE_EXERCISER_PATTERN_TERM_2_ENABLE
- PCIE_EXERCISER_PATTERN_TERM_3_ENABLE

- Support for different transceiver settings at Gen 1, Gen 2 speeds have been added. These are available through the following new values added to the EPCIEDllPhy enumeration:
  - PCIE_DLLPHY_DEEMPHASIS_LEVEL_GEN1
  - PCIE_DLLPHY_DEEMPHASIS_LEVEL_GEN2

Directory structure:

The directory structure is as follows:

```
…\PCIe Port API
    +---Bin
        |  +---Win32                 : Dll files for 32 bit
        |  +---x64                   : Dll files for AMD 64 bit
    +---Doc                        : Readme files etc.
    +---Inc                        : Header files.
    +---Lib                        :
        |  +---Win32                 : Library files for 32 bit
        |  +---x64                   : Library files for AMD 64 bit
    +---Samples
        |  +---MRIOVMultiFunction    : MRIOV sample
        |  +---SimplePCIE            : Simple PCIE sample
```

Installation:

- Install the PCIe port API (e.g. into C:\Program Files\Agilent\SPT\PCIEExerciserGen3\8.70 Release\PCIe port API). This creates the PCIe port API subdirectory tree and installs the needed driver files.
- Shut down SUT.
- Plug a probe board into a PCI-Express slot of the SUT and open an exerciser session to it as usual from the client PC (via TCL, C++/COM or GUI). You can also use the TCL file "tclshrc.tcl" in the "C:\Program Files\Agilent\SPT\PCIEExerciserGen3\8.70 Release\Samples\PCIe Port" folder to start the session.
- Boot SUT.
- You may check that the link is up (from the client) and that the probe board is correctly recognized in the device manager (under Agilent Technologies Interface-Cards) on the SUT.
- To check successful installation on the SUT, open file ...\PCIe port API\Samples\SimplePCIE\SimplePCIE.sln with MS Visual Studio 2005 and compile. This example is a

good starting point and shows how to open and close a connection and how to access the exerciser functions.

More information:

- For a list of exerciser functions see ...\PCIe port API\src\AgtPCIEExerciser.h.
- For information about the connection functions see ...\PCIe port API\src\AgtPortSelector.h.
- MRIOV specific sample can be found at …\ PCIe port API\Samples\MRIOVMultiFunction. Please open MRIOVMultiFunction.sln in Microsoft Visual Studio 2005 to use this sample.
- The PCIe Port APIs are similar to the host APIs. Please refer to <Exerciser Install Folder>\8.70 Release\doc\PCIE_Exerciser_API_Online_Help.chm for information about individual API functions.

Common problems:

- If you cannot connect to U4305A (probe board) card, make sure the card shows up correctly in the device manager (under Agilent Technologies Interface-Cards). If not, make sure the driver files are installed correctly and reboot the PC. The driver files can be found at the following locations:
  32bit systems: <WindowsDir>\Inf\AgtPCIeEx_x86.inf and
  <WindowsDir>\SYSTEM32\DRIVERS\AgtPCIeEx_x86.sys
  64 bit Itanium systems: <WindowsDir>\INF\AgtPCIeEx_ia64.inf and
  <WindowsDir>\SYSTEM32\DRIVERS\AgtPCIeEx_ia64.sys
  64 bit AMD architecture systems: <WindowsDir>\INF\AgtPCIeEx_amd64.inf and
  <WindowsDir>\SYSTEM32\DRIVERS\AgtPCIeEx_amd64.sys

- When executing samples on Windows 7 and some other Windows operating systems, an error saying device not found may be displayed even though the device is visible in the Windows Device Manager. This may be due to insufficient privileges. Run command prompt as Administrator if launching the sample executable from command prompt or run Visual Studio as Administrator if launching the sample using Visual Studio.