Protocol Solutions Group

3385 Scott Blvd., Santa Clara, CA 95054

Tel: +1/408.727.6600

Fax: +1/408.727.6622

USB Power Delivery Exerciser Manual

Manual Version 1.46

For USB Protocol Suite Software Version 7.45 and above

February 2017

Version

This is version 1.46 of the *USB Power Delivery Exerciser Manual*.

This manual applies to USB Protocol Suite software version 7.45 and higher.

Contents

1 INTRODUCTION

Integrated in Teledyne LeCroy's Voyager M310C test platform, the Power Delivery exerciser supports traffic generation, including both provider and consumer device emulation. The Power Delivery exerciser continues to evolve with each software release. Be sure to check for updated software and firmware before getting started with the Exerciser.

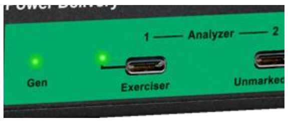
Important Licensing Note:

- Operating the PD Exerciser beta requires that the USB Power Delivery Exerciser option is enabled on the M310C base unit:

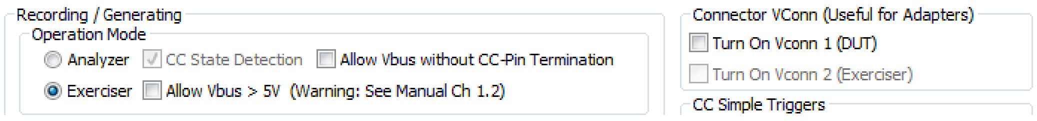
USB Power Delivery - Type C	Yes	USB Power Delivery Analysis - Type-C
USB Power Delivery - Exerciser	Yes	USB Power Delivery Exerciser

Getting Started:

- The "left" port of the Voyager should be used to connect DUT to the PD Exerciser. The PD exerciser also requires specific cable orientation (Red LED when connected wrong side-up).



- To enable the PD Trainer/Exerciser, use the PD Tab under *"Recording Options"* to select the Exerciser mode.

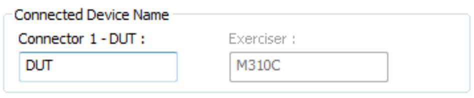


Recording / Generating
Operation Mode
○ Analyzer ☑ CC State Detection ☐ Allow Vbus without CC-Pin Termination
◉ Exerciser ☐ Allow Vbus > 5V (Warning: See Manual Ch 1.2)

Connector VConn (Useful for Adapters)
☐ Turn On Vconn 1 (DUT)
☐ Turn On Vconn 2 (Exerciser)
CC Simple Triggers

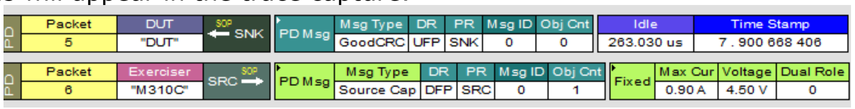
Note – *Allow VBUS > 5v* is a safety feature which prevents sourcing above 5V. When enabled, this mode will allow Voltage levels to be delivered to the DUT which may exceed their current carrying capabilities. While the M310C system is designed to tolerate higher current, these higher voltages may inadvertently cause damage to devices/cables under test.

- To set *devices port name*, use the General Tab under *"Recording Options"* to add "alias labels" for your DUTs.



Connected Device Name
Connector 1 - DUT : DUT
Exerciser : M310C

These labels will appear in the trace capture.

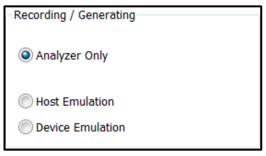


PD | Packet 5 | DUT "DUT" | SOP ← SNK | PD Msg | Msg Type GoodCRC | DR UFP | PR SNK | Msg ID 0 | Obj Cnt 0 | Idle 263.030 us | Time Stamp 7 . 900 668 406

PD | Packet 6 | Exerciser "M310C" | SRC → SOP | PD Msg | Msg Type Source Cap | DR DFP | PR SRC | Msg ID 0 | Obj Cnt 1 | Fixed | Max Cur 0.90 A | Voltage 4.50 V | Dual Role 0

The Alias name is primarily for use in analyzer mode and requires that device names are added before recording traffic. The device naming can also be used in Exerciser mode; however message frames from the Voyager M310C will be always be labeled "M310C".

- Within the USB 3.1 tab – "*Recording/Generating*" option - leave in 'Analyzer Only' mode unless you also want to run 3.1 traffic.


Recording / Generating

[✓ Analyzer Only] Host Emulation Device Emulation

- Use the example PD Exerciser scripts to begin testing:

C:\Users\Public\Documents\LeCroy\USB Protocol Suite\Examples\Power Delivery Exerciser

Example Script	**Behavior**
Source Power Negotiate VDM.updg	Voyager as Source negotiates default Provider 900mA@4.5V then sends Discover-Id. Using Basic Commands.
High Level Negotiate with dynamic change cap.updg	Voyager as Source negotiates default Provider 1A@5V then broadcasts lower PDO 900mA@4.5V and re-negotiates. Using High Level Commands.
Discover Cable.updg	Voyager as Source programmatically turns on VCONN and performs Discovery Process for cable. Using High Level Commands.
Sink Power Negotiate.updg	Voyager as Sink Waits to receive Source cap then negotiates as Sink - 900mA@5V. Using Basic Commands.
Apple VGA multiple Adaptor.updg	Voyager as Source enables VCONN and Sends Discover Id; Discover Mode for Apple SVID (0x05AC); Enter Mode (PD_DISPLAY_PORT_SVID) then Exit Mode; turns off VCONN. Using Basic Commands.
High Level Device Discovery.updg	Voyager as Source sends Discover Id; Discover SVIDs; Discover Modes for Display Port SVID (0xFF01); Enter Mode (0xFF01); Exit Mode (0xFF01); Discover Modes for Apple SVID (0x05AC); Enter Mode(0x5AC mode 1); Exit Mode(0x5AC mode 1); Enter Mode(0x5AC mode 2); Exit Mode(0x5AC mode 2); Using High Level Commands.
NegotiationSample_WithSwapPowerRole.updg	Voyager as Source sends SwapPowerRole; and negotiates as a Sink after power role swap. 1.5A@5V. Using High Level Commands.
Sink Auto Response.updg	Voyager as Sink will response to all incoming PD messages within 100s. Using Auto Response Command.

- To Run Sample Script – Connect Cable to Exerciser port; Click *Record*, wait a few seconds and Click *Run.* The PD Exerciser uses the sequence below at the beginning of each example script to simulate a re-connect event.

```
call PD_Disconnect()

call PD_SetResistorRp( PD_ON, CC_RP_CUR_1_5, CC_LINE_1 )
call PD_SetVBus( PD_ON )
```

Note- it's also possible to execute the example scripts before the cable is connected to M310C then performing "hot-plug" (It's possible some issues may be seen with some devices not responding to exerciser in this case).
Note – some latency may be observed when activating/downloading PD exerciser scripts (Run button) This will be improved in a future release.

2 Packet Templates

Following Packet Templates can be used in Basic or High-Level commands as data containers. All of these messages inherited from `PD_Packet` packet template except those which are used as containers for Data Objects.

2.1 PD_ControlMessage

Available fields for `PD_ControlMessage` packet template are:

Field Name	Description
MessageType	Default: 0
Reserved1	Default: 0 Rev2.0 only
PortDataRole_Reserved2	Default: 0
SpecificationRevision	Default: `PD_SPEC_REVISION_2` (Rev2.0) Default: `PD_SPEC_REVISION_3` (Rev3.0)
PortPowerRole_CablePlug	Default: 0
MessageId	Default: 0
NumberOfDataObjects	Default: 0
Reserved2	Default: 0 Rev2.0 only
Extended	Default: 0 Rev3.0 only

2.2 PD_GoodCrcMessage

`PD_GoodCrcMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `1`.

2.3 PD_GotoMinMessage

`PD_GotoMinMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `2`.

2.4 PD_AcceptMessage

`PD_AcceptMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `3`.

2.5 PD_RejectMessage

`PD_RejectMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `4`.

2.6 PD_PingMessage

`PD_PingMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `5`.

2.7 PD_PsRdyMessage

`PD_PsRdyMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `6`.

2.8 PD_GetSourceCapMessage

`PD_GetSourceCapMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `7`.

2.9 PD_GetSinkCapMessage

`PD_GetSinkCapMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `8`.

2.10 PD_DataRoleSwapMessage

`PD_DataRoleSwapMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `9`.

2.11 PD_PowerRoleSwapMessage

`PD_PowerRoleSwapMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `10`.

2.12 PD_VConnSwapMessage

`PD_VConnSwapMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `11`.

2.13 PD_WaitMessage

`PD_WaitMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `12`.

2.14 PD_SoftResetMessage

`PD_SoftResetMessage` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `13`.

2.15 PD_NotSupportedMsg

`PD_NotSupportedMsg` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `16`. Applied to Power Delivery Rev3.0.

2.16 PD_GetSourceCapExtendedMsg

`PD_GetSourceCapExtendedMsg` packet template has same fields as `PD_ControlMessage` but default value for `MessageType` is `17`. Applied to Power Delivery Rev3.0.

2.17 PD_GetStatusMsg

PD_GetStatusMsg packet template has same fields as PD_ControlMessage but default value for MessageType is 18. Applied to Power Delivery Rev3.0.

2.18 PD_FRSwapMsg

PD_FRSwapMsg packet template has same fields as PD_ControlMessage but default value for MessageType is 19. Applied to Power Delivery Rev3.0.

2.19 PD_SourceCapabilitiesMessage

PD_SourceCapabilitiesMessage packet template contains all the fields of PD_ControlMessage but default value for MessageType is 1. Following are additional data fields for PD_SourceCapabilitiesMessage packet template:

Field Name	Description
SourceCapabilitiesData	This field can contain one or more(up-to 7 according to PD Spec) PDO packet variables. PDO types which can assign to this field are: PD_PowerDataObjectFixedSupply_Source, PD_PDOFixedSupplyNotVSafe5V_Source, PD_PowerDataObjectVariableSupply_Source, PD_PowerDataObjectBatterySupply_Source

2.19.1 PD_PowerDataObjectFixedSupply_Source

Used as SourceCapabilitiesData for PD_SourceCapabilitiesMessage. Available fields for this packet template are:

Field Name	Description
MaxCurrent_10mAUnits	Default: 100
Voltage_50mVUnits	Default: 100
PeakCurrent	Default: 0
Reserved	Default: 0
UnchunkedExtMsgSupported	Default: 0 Rev3.0 only
DataRoleSwap	Default: 0
UsbCommunicationsCapable	Default: 0
ExternallyPowered	Default: 1
UsbSuspendSupported	Default: 0
DualRolePower	Default: 0
PowerDataType	Default: 0

2.19.2 PD_PDOFixedSupplyNotVSafe5V_Source

Used as SourceCapabilitiesData for PD_SourceCapabilitiesMessage. Available fields for this packet template are:

Field Name	Description
MaxCurrent_10mAUnits	Default: 0
Voltage_50mVUnits	Default: 0
PeakCurrent	Default: 0
Reserved	Default: 0
PowerDataType	Default: 0

2.19.3 PD_PowerDataObjectVariableSupply_Source

Used as `SourceCapabilitiesData` for `PD_SourceCapabilitiesMessage`. Available fields for this packet template are:

Field Name	Description
MaxCurrent_10mAUnits	Default: 0
MinVoltage_50mVUnits	Default: 0
MaxVoltage_50mVUnits	Default: 0
PowerDataType	Default: 2

2.19.4 PD_PowerDataObjectBatterySupply_Source

Used as `SourceCapabilitiesData` for `PD_SourceCapabilitiesMessage`. Available fields for this packet template are:

Field Name	Description
MaxAllowablePower_250mWUnits	Default: 0
MinVoltage_50mVUnits	Default: 0
MaxVoltage_50mVUnits	Default: 0
PowerDataType	Default: 1

2.20 PD_SinkCapabilitiesMessage

`PD_SinkCapabilitiesMessage` packet template contains all the fields of `PD_ControlMessage` but default value for `MessageType` is `4`. Following are additional data fields for `PD_SinkCapabilitiesMessage` packet template:

Field Name	Description
SinkCapabilitiesData	This field can contain one or more(up-to 7 according to PD Spec) PDO packet variables. PDO types which can assign to this field are: `PD_PowerDataObjectFixedSupply_Sink`, `PD_PowerDataObjectVariableSupply_Sink`, `PD_PowerDataObjectBatterySupply_Sink`

2.20.1 PD_PowerDataObjectFixedSupply_Sink

Used as `SinkCapabilitiesData` for `PD_SinkCapabilitiesMessage`. Available fields for this packet template are:

Field Name	Description
OperationalCurrent_10mAUnits	Default: 100
Voltage_50mVUnits	Default: 100
Reserved	Default: 0
FRSwapTypeCCurrent	Default: 0 Rev3.0 only
DataRoleSwap	Default: 0
UsbCommunicationsCapable	Default: 0
ExternallyPowered	Default: 1
HigherCapability	Default: 0
DualRolePower	Default: 0
PowerDataType	Default: 0

2.20.2 PD_PowerDataObjectVariableSupply_Sink

Used as `SinkCapabilitiesData` for `PD_SinkCapabilitiesMessage`. Available fields for this packet template are:

Field Name	Description
OperationalCurrent_10mAUnits	Default: 0
MinVoltage_50mVUnits	Default: 0
MaxVoltage_50mVUnits	Default: 0
PowerDataType	Default: 2

2.20.3 PD_PowerDataObjectBatterySupply_Sink

Used as `SinkCapabilitiesData` for `PD_SinkCapabilitiesMessage`. Available fields for this packet template are:

Field Name	Description
OperationalPower_250mWUnits	Default: 0
MinVoltage_50mVUnits	Default: 0
MaxVoltage_50mVUnits	Default: 0
PowerDataType	Default: 1

2.21 PD_RequestPacket

`PD_RequestPacket` packet template contains all the fields of `PD_ControlMessage` but default value for `MessageType` is `2` and default value for `NumberOfDataObjects` field is `1`. Following are additional data fields for `PD_RequestPacket` packet template:

Field Name	Description
Data	This field can contain only one RDO packet variables. RDO types which can assign to this field are: `PD_RequestDataObject_Fixed_Variable_NoGiveBack`, `PD_RequestDataObject_Fixed_Variable_GiveBack`, `PD_RequestDataObject_Battery_NoGiveBack`, `PD_RequestDataObject_Battery_GiveBack`

2.21.1 PD_RequestDataObject_Fixed_Variable_NoGiveBack

Used as `data` for `PD_RequestPacket`. Available fields for this packet template are:

Field Name	Description
MaxOperatingCurrent_10mAUnits	Default: 0
OperatingCurrent_10mAUnits	Default: 0
Rsvd1	Default: 0
UnchunkedExtMsgSupported	Default: 0 Rev3.0 only
NoUsbSuspend	Default: 0
UsbCommunicationsCapable	Default: 0
CapabilityMismatch	Default: 0
GiveBackFlag	Default: 0
ObjectPosition	Default: 1
Rsvd2	Default: 0

2.21.2 PD_RequestDataObject_Fixed_Variable_GiveBack

Used as data for PD_RequestPacket. This packet template has same fields as PD_RequestDataObject_Fixed_Variable_NoGiveBack packet template, but default value for GiveBackFlag field is 1.

2.21.3 PD_RequestDataObject_Battery_NoGiveBack

Used as data for PD_RequestPacket. Available fields for this packet template are:

Field Name	Description
MaxOperatingPower_250mWUnits	Default: 0
OperatingPower_250mWUnits	Default: 0
Rsvd1	Default: 0
UnchunkedExtMsgSupported	Default: 0 Rev3.0 only
NoUsbSuspend	Default: 0
UsbCommunicationsCapable	Default: 0
CapabilityMismatch	Default: 0
GiveBackFlag	Default: 0
ObjectPosition	Default: 1
Rsvd2	Default: 0

2.21.4 PD_RequestDataObject_Battery_GiveBack

Used as data for PD_RequestPacket. This packet template has same fields as PD_RequestDataObject_Battery_NoGiveBack packet template, but default value for GiveBackFlag field is 1.

2.22 PD_BISTCarrierModeMessage

PD_BISTCarrierModeMessage packet template contains all the fields of PD_ControlMessage but default value for MessageType is 3 and default value for NumberOfDataObjects field is 1. Following are additional data fields for PD_BISTCarrierModeMessage packet template:

Field Name	Description
Reserved	Default: 0
BISTRequestType	Default: BIST_REQUEST_CARRIER_MODE

2.23 PD_BISTTestDataMessage

PD_BISTTestDataMessage packet template contains all the fields of PD_ControlMessage but default value for MessageType is 3 and default value for NumberOfDataObjects field is 7. Following are additional data fields for PD_BISTTestDataMessage packet template:

Field Name	Description
Reserved	Default: 0
BISTRequestType	Default: BIST_REQUEST_TEST_DATA
TestData	Default: {AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA}

2.24 PD_BatteryStatusMsg

Applied to Rev3.0. PD_BatteryStatusMsg packet template contains all the fields of PD_ControlMessage but default value for MessageType is 5 and default value for

`NumberOfDataObjects` field is `1`. Following are additional data fields(fields of `PD_BatteryStatusDataObject` packet template) for `PD_BatteryStatusMsg` packet template:

Field Name	Description
Reserved_1	Default: 0x00
InvalidBatteryReference	Default: 0x00
BatteryIsPresent	Default: 0x00
BatteryChargingStatus	Default: 0x00
Reserved_2	Default: 0x00
BatteryPC	Default: 0xFFFF

2.25 PD_AlertMsg

Applied to Rev3.0. `PD_AlertMsg` packet template contains all the fields of `PD_ControlMessage` but default value for `MessageType` is `6` and default value for `NumberOfDataObjects` field is `1`. Following are additional data fields(fields of `PD_AlertDataObject` packet template) for `PD_AlertMsg` packet template:

Field Name	Description
Reserved_1	Default:0x00
HotSwappableBatteries	Default:0x00
FixedBatteries	Default:0x00
Reserved_2	Default:0x00
BatteryStatusChange	Default:0x00
OverCurProtection	Default:0x00
OverTempProtection	Default:0x00
OperatingConditionChange	Default:0x00
SourceInputChange	Default:0x00
OverVoltageProtection	Default:0x00
Reserved_3	Default:0x00

2.26 PD_VDM_Unstructured_Header

Used as Header for Unstructured VDM messages. `PD_VDM_Unstructured_Header` packet template contains all the fields of `PD_ControlMessage` but default value for `MessageType` is `0x0F` and default value for `NumberOfDataObjects` field is `1`. Following are additional data fields for `PD_VDM_Unstructured_Header` packet template:

Field Name	Description
VDMCustom	Default: 0x00
VDMType	Default: PD_VDM_TYPE_UNSTRUCTURED_VDM
VDMSVID	Default: 0x00

2.27 PD_VDM_Structured_Header

Used as Header for all Structured VDM messages. `PD_VDM_Structured_Header` packet template contains all the fields of `PD_ControlMessage` but default value for `MessageType` is `0x0F` and default value for `NumberOfDataObjects` field is `1`. Following are additional data fields for `PD_VDM_Structured_Header` packet template:

Field Name	Description
VDMCommand	Default: PD_VDM_COMMAND_RESERVED_0
VDMReserved1	Default: 0x00

VDMCommandType	Default: PD_VDM_COMMAND_TYPE_INITIATOR
VDMObjectPosition	Default: 0x00
VDMReserved2	Default: 0x00
VDMStructuredVdmVersion	Default: PD_VDM_STRUCTURED_VERSION_2 (Rev 3.0) Default: PD_VDM_STRUCTURED_VERSION_1 (Rev 2.0)
VDMType	Default: PD_VDM_TYPE_STRUCTURED_VDM
VDMSVID	Default: 0x00

2.28 PD_VDM_Discover_Identity_Message

PD_VDM_Discover_Identity_Message packet template contains all the fields of PD_VDM_Structured_Header but default value for VDMCommand field is PD_VDM_COMMAND_DISCOVER_IDENTITY and default value for VDMSVID field is PD_VDM_SID.

2.29 PD_VDM_Discover_Identity_Response

PD_VDM_Discover_Identity_Response packet template contains all the fields of PD_VDM_Discover_Identity_Message but default value for VDMCommandType field is PD_VDM_COMMAND_TYPE_RESPONDER_ACK. Following are additional data fields for PD_VDM_Discover_Identity_Response packet template:

Field Name	Description
VDOs	This field can contain up-to 6 VDOs, but should contain at least 3 VDOs (according to PD Spec). VDO types which can assign to this field are: PD_VDM_Discover_Identity_ID_Header_VDO, PD_VDM_Discover_Identity_Cert_Stat_VDO, PD_VDM_Discover_Identity_Product_VDO, PD_VDM_Discover_Identity_Cable_VDO (Rev 2.0 only), PD_DiscoverIdPassiveCableVdo (Rev 3.0 only), PD_DiscoverIdActiveCableVdo (Rev 3.0 only), PD_VDM_Discover_Identity_Alternate_Mode_Adapter_VDO

2.29.1 PD_VDM_Discover_Identity_ID_Header_VDO

Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are varies from Revision 2.0 to higher revisions:

2.29.1.1 Revision 2.0

Field Name	Description
IDHeaderVDO_USBVendorID	Default: 0x00
IDHeaderVDO_Reserved	Default: 0x00
IDHeaderVDO_ModalOperationSupported	Default: 0x00
IDHeaderVDO_ProductType	Default: PD_VDM_ID_HEADER_VDO_PRODUCT_TYPE_UNDEFINED
IDHeaderVDO_DataCapableAsUSBDevice	Default: 0x00
IDHeaderVDO_DataCapableAsUSBHost	Default: 0x00

2.29.1.2 Revision 3.0

Field Name	Description
USBVendorID	Default: 0x00
Reserved	Default: 0x00
ProductType_DFP	Default: PD_PRODUCT_TYPE_UNDEFINED

ModalOperationSupported	Default: 0x00
ProductType_UFP_Cable	Default: PD_PRODUCT_TYPE_UNDEFINED
DataCapableAsUSBDevice	Default: 0x00
DataCapableAsUSBHost	Default: 0x00

2.29.2 PD_VDM_Discover_Identity_Cert_Stat_VDO

Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are:

Field Name	Description
CertStatVDO_XID	Default: 0x00
Rsvd	Default: 0x00 Rev2.0 only

2.29.3 PD_VDM_Discover_Identity_Product_VDO

Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are:

Field Name	Description
ProductVDO_USBProductId	Default: 0x00
ProductVDO_BCDDevice	Default: 0x00

2.29.4 PD_VDM_Discover_Identity_Cable_VDO

Applied to Revision 2.0 only. Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are:

Field Name	Description
CableVDO_USBSuperSpeedSignalingSupport	Default: PD_VDM_CABLE_VDO_USB31_GEN1_SIGNALING_SUPPORT
CableVDO_SOPDPrimeControllerPresent	Default: 0x00
CableVDO_VBusThroughCable	Default: 0x00
CableVDO_VBusCurrentHandlingCapability	Default: PD_VDM_CABLE_VDO_VBUS_HANDLING_NO_VBUS
CableVDO_SSRX2DirectionalitySupport	Default: 0x00
CableVDO_SSRX1DirectionalitySupport	Default: 0x00
CableVDO_SSTX2DirectionalitySupport	Default: 0x00
CableVDO_SSTX1DirectionalitySupport	Default: 0x00
CableVDO_CableTerminationType	Default: 0x00
CableVDO_CableLatency	Default: PD_VDM_CABLE_VDO_CABLE_LATENCY_UPTO_10ns
CableVDO_TypeCPlugToPlugOrReceptacle	Default: PD_VDM_CABLE_VDO_TYPEC_PLUG_TO_PLUG
CableVDO_TypeCPlugToTypeA_B_C_Captive	Default: PD_VDM_CABLE_VDO_TYPEC_PLUGTO_TYPEC
CableVDO_Reserved	Default: 0x00
CableVDO_FirmwareVersion	Default: 0x00
CableVDO_HardwareVersion	Default: 0x00

2.29.5 PD_DiscoverIdPassiveCableVdo

Applied to Rev3.0 only. Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are:

Field Name	Description
USBSsSignaling	Default: PD_CABLE_USB31_GEN1_SIGNALING
Reserved_1	Default: 0x00
VBusCurHandlingCap	Default: PD_CABLE_CUR_HANDLING_CAP_3A

Reserved_2	Default: 0x00
MaxVBusVoltage	Default: PD_CABLE_MAX_VBUS_20V
CableTerminationType	Default: PD_CABLE_VCONN_NOT_REQUIRED
CableLatency	Default: PD_CABLE_LATENCY_MAX_10ns
Reserved_3	Default: 0x00
TypeCtoTypeC_Captive	Default: PD_CABLE_TYPEC_TO_TYPEC
Reserved_4	Default: 0x00
Version	Default: PD_CABLE_PASSIVE_VDO_VERSION_1
FirmwareVersion	Default: 0x00
HardwareVersion	Default: 0x00

2.29.6 PD_DiscoverIdActiveCableVdo

Applied to Rev3.0 only. Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are:

Field Name	Description
USBSsSignaling	Default: PD_CABLE_USB31_GEN1_SIGNALING
SOPDoublePrimeController	Default: 0x00
VBusThrough	Default: 0x01
VBusCurHandlingCap	Default: PD_CABLE_CUR_HANDLING_CAP_3A
Reserved_1	Default: 0x00
MaxVBusVoltage	Default: PD_CABLE_MAX_VBUS_20V
CableTerminationType	Default: PD_CABLE_TERM_ACTIVE_ACTIVE
CableLatency	Default: PD_CABLE_LATENCY_MAX_10ns
Reserved_2	Default: 0x00
TypeCtoTypeC_Captive	Default: PD_CABLE_TYPEC_TO_TYPEC
Reserved_3	Default: 0x00
Version	Default: PD_CABLE_ACTIVE_VDO_VERSION_1
FirmwareVersion	Default: 0x00
HardwareVersion	Default: 0x00

2.29.7 PD_VDM_Discover_Identity_Alternate_Mode_Adapter_VDO

Used as VDOs for PD_VDM_Discover_Identity_Response packet template. Available fields of this packet template are varies from Revision 2.0 to higher revisions:

2.29.7.1 Revision 2.0

Field Name	Description
AMDVDO_USBSuperSpeedSignalingSupport	Default: 0x01
AMDVDO_VBusRequired	Default: 0x00
AMDVDO_VConnRequired	Default: 0x00
AMDVDO_VConnPower	Default: 0x00
AMDVDO_SSRX2DirectionalitySupport	Default: 0x00
AMDVDO_SSRX1DirectionalitySupport	Default: 0x00
AMDVDO_SSTX2DirectionalitySupport	Default: 0x00
AMDVDO_SSTX1DirectionalitySupport	Default: 0x00
AMDVDO_Reserved	Default: 0x00
AMDVDO_FirmwareVersion	Default: 0x00
AMDVDO_HardwareVersion	Default: 0x00

2.29.7.2 Revision 3.0

Field Name	Description
USBSsSignaling	Default: 0x01
VBusRequired	Default: 0x00
VConnRequired	Default: 0x00
VConnPower	Default: PD_AMA_VCONN_POWER_1
Reserved	Default: 0x00
Version	Default: PD_AMA_VDO_VERSION_1
FirmwareVersion	Default: 0x00
HardwareVersion	Default: 0x00

2.30 PD_VDM_Discover_Svids_Message

`PD_VDM_Discover_Svids_Message` packet template contains all the fields of `PD_VDM_Structured_Header` but default value for `VDMCommand` field is `PD_VDM_COMMAND_DISCOVER_SVIDS` and default value for `VDMSVID` field is `PD_VDM_SID`.

2.31 PD_VDM_Discover_Svids_Response

`PD_VDM_Discover_Svids_Response` packet template contains all the fields of `PD_VDM_Discover_Svids_Message` but default value for `VDMCommandType` field is `PD_VDM_COMMAND_TYPE_RESPONDER_ACK`. Following are additional data fields for `PD_VDM_Discover_Svids_Response` packet template:

Field Name	Description
DiscoverSVIDsResponderVDOs	Contains one or more VDOs. The only VDO type which can assign to this field is: `Discover_SVIDs_Responder_VDO`

2.31.1 Discover_SVIDs_Responder_VDO

Used as `DiscoverSVIDsResponderVDOs` for `PD_VDM_Discover_Svids_Response` packet template. Available fields of this packet template are:

Field Name	Description
SVID1	Default: 0x00
SVID2	Default: 0x00

2.32 PD_VDM_Discover_Modes_Message

`PD_VDM_Discover_Modes_Message` packet template contains all the fields of `PD_VDM_Structured_Header` but default value for `VDMCommand` field is `PD_VDM_COMMAND_DISCOVER_MODES` and default value for `VDMSVID` field is `PD_VDM_SID`.

2.33 PD_VDM_Discover_Modes_Response

`PD_VDM_Discover_Modes_Response` packet template contains all the fields of `PD_VDM_Discover_Modes_Message` but default value for `VDMCommandType` field is `PD_VDM_COMMAND_TYPE_RESPONDER_ACK`. Following are additional data fields for `PD_VDM_Discover_Modes_Response` packet template:

Field Name	Description
DiscoverModes	This field should contain one or more VDOs(modes). Each Mode

	can be a PD_VDO packet variable which has 32bits data length.

2.33.1 PD_VDO

Can be use as DiscoverModes for PD_VDM_Discover_Modes_Response packet template. Available fields of this packet template are:

Field Name	Description
Data	Default: 0x00

2.33.2 PD_VDM_DisplayPort_DiscoverMode_Vdo

In response to a Discover Mode Command, the VDO assigned to the DiscoverModes field can be in type of PD_VDM_DisplayPort_DiscoverMode_Vdo packet template, if the requested VDMSVID is PD_DISPLAY_PORT_SVID(0xFF01). Available data fields for PD_VDM_DisplayPort_DiscoverMode_Vdo packet template are:

Field Name	Description
PortCapability	Default: PD_DISPLAYPORT_UFPD_CAPABLE
Signaling	Default: 0x01
ReceptacleIndication	Default: 0x00
Usb2SignalingNotUsed	Default: 0x01
DFPDPinAssignmentSupported	Default: 0x00
UFPDPinAssignmentSupported	Default: 0x00
Reserved_DMV	Default: 0x00

2.34 PD_VDM_Enter_Mode_Message

PD_VDM_Enter_Mode_Message packet template contains all the fields of PD_VDM_Structured_Header but default value for VDMCommand field is PD_VDM_COMMAND_ENTER_MODE and default value for VDMSVID field is PD_VDM_SID. Following are additional data fields for PD_VDM_Enter_Mode_Message packet template:

Field Name	Description
VDO	This field may contain one VDO. The VDO can be a PD_VDO packet variable which has 32bits data length.

2.35 PD_VDM_Enter_Mode_Response

PD_VDM_Enter_Mode_Response packet template contains all the fields of PD_VDM_Structured_Header but default value for VDMCommand field is PD_VDM_COMMAND_ENTER_MODE and default value for VDMSVID field is PD_VDM_SID and default value for VDMCommandType field is PD_VDM_COMMAND_TYPE_RESPONDER_ACK.

2.36 PD_VDM_Exit_Mode_Message

PD_VDM_Exit_Mode_Message packet template contains all the fields of PD_VDM_Structured_Header but default value for VDMCommand field is PD_VDM_COMMAND_EXIT_MODE and default value for VDMSVID field is PD_VDM_SID.

2.37 PD_VDM_Exit_Mode_Response

PD_VDM_Exit_Mode_Response packet template contains all the fields of PD_VDM_Exit_Mode_Message but default value for VDMCommandType field is PD_VDM_COMMAND_TYPE_RESPONDER_ACK.

2.38 PD_VDM_Attention_Message

PD_VDM_Attention_Message packet template contains all the fields of PD_VDM_Structured_Header but default value for VDMCommand field is PD_VDM_COMMAND_ATTENTION and default value for VDMSVID field is PD_VDM_SID. Following are additional data fields for PD_VDM_Attention_Message packet template:

Field Name	Description
VDO	This field may contain one VDO. The VDO can be a PD_VDO packet variable which has 32bits data length.

2.39 PD_VDM_DisplayPort_UpdateStatus_Message

PD_VDM_DisplayPort_UpdateStatus_Message packet template contains all the fields of PD_VDM_Structured_Header but default value for NumberOfDataObjects field is 2 and default value for VDMCommand field is PD_VDM_COMMAND_DISPLAYPORT_STATUS_UPDATE and default value for VDMSVID field is PD_DISPLAY_PORT_SVID. Following are additional data fields for PD_VDM_DisplayPort_UpdateStatus_Message packet template:

Field Name	Description
StatusVdo	Contains only one VDO in type of PD_VDM_DisplayPort_Status_VDO packet template.

2.39.1 PD_VDM_DisplayPort_Status_VDO

Used as StatusVdo for PD_VDM_DisplayPort_UpdateStatus_Message and PD_VDM_DisplayPort_UpdateStatus_Response packet templates. Following are available data fields for this packet template:

Field Name	Description
DFPD_UFPD_Connected	Default: PD_DISPLAYPORT_DISCONNECTED
PowerLow	Default: 0x00
AdaptorEnabled	Default: 0x00
MultiFunctionPreferred	Default: 0x00
UsbConfigurationRequest	Default: 0x00
ExitDisplayModeRequest	Default: 0x00
HPD_State	Default: 0x00
IRQ_HPD	Default: 0x00
Reserved_DPS_1	Default: 0x00

2.40 PD_VDM_DisplayPort_UpdateStatus_Response

PD_VDM_DisplayPort_UpdateStatus_Response packet template contains all the fields of PD_VDM_DisplayPort_UpdateStatus_Message but default value for VDMCommandType field is PD_VDM_COMMAND_TYPE_RESPONDER_ACK.

2.41 PD_VDM_DisplayPort_Configure_Message

PD_VDM_DisplayPort_Configure_Message packet template contains all the fields of PD_VDM_Structured_Header but default value for NumberOfDataObjects field is 2 and default value for VDMCommand field is PD_VDM_COMMAND_DISPLAYPORT_CONFIGURE and default value for VDMSVID field is PD_DISPLAY_PORT_SVID. Following are additional data fields for this packet template:

Field Name	Description
ConfigureVdo	Contains only one VDO in type of PD_VDM_DisplayPort_Configure_VDO packet template.

2.41.1 PD_VDM_DisplayPort_Configure_VDO

Used as ConfigureVdo for PD_VDM_DisplayPort_Configure_Message packet template. Available data fields for this packet template are:

Field Name	Description
SelectConfiguration	Default: PD_DISPLAYPORT_CONFIGURATION_USB
Signaling	Default: 0x00
Reserved_DPC_1	Default: 0x00
UFPU_PinAssignment	Default: 0x00
Reserved_DPC_2	Default: 0x00

2.42 PD_VDM_DisplayPort_Configure_Response

PD_VDM_DisplayPort_Configure_Response packet template contains all the fields of PD_VDM_Structured_Header but default value for VDMCommand field is PD_VDM_COMMAND_DISPLAYPORT_CONFIGURE and default value for VDMSVID field is PD_DISPLAY_PORT_SVID and default value for VDMCommandType is PD_VDM_COMMAND_TYPE_RESPONDER_ACK.

2.43 PD_ExtMsgHeaders

Applied to Rev3.0. PD_ExtMsgHeaders packet template contains all the fields of PD_ControlMessage but the default value of Extended field is 1. Following are the additional fields for this packet template:

Field Name	Description
DataSize	Default: 0x00
Reserved	Default: 0x00
RequestChunk	Default: 0x00
ChunkNumber	Default: 0x00
Chunked	Default: 0x00

2.44 PD_SourceCapExtendedMsg

Applied to Rev3.0. PD_SourceCapExtendedMsg packet template contains all the fields of PD_ExtMsgHeaders but the default value of MessageType field is 1 and the default value of DataSize field is 0x17. Following are the additional fields for this packet template(these additional fields blong to PD_SourceCapExtDataBlock packet template):

Field Name	Description

VendorId	Default: 0x00
ProductId	Default: 0x00
XId	Default: 0x00
FirmwareVersion	Default: 0x00
HardwareVersion	Default: 0x00
LoadStep	Default: 0x00
IOC	Default: 0x00
Reserved_1	Default: 0x00
HoldupTime	Default: 0x00
LPSCompliant	Default: 0x00
PS1Compliant	Default: 0x00
PS2Compliant	Default: 0x00
Reserved_2	Default: 0x00
LowTouchCurEPS	Default: 0x00
GroundPinSupport	Default: 0x00
GrndPinForProtectiveEarth	Default: 0x00
Reserved_3	Default: 0x00
PeakCur1_PercentOverload	Default: 0x00
PeakCur1_OverloadPeriod	Default: 0x00
PeakCur1_DutyCycle	Default: 0x00
PeakCur1_VBusVoltageDroop	Default: 0x00
PeakCur2_PercentOverload	Default: 0x00
PeakCur2_OverloadPeriod	Default: 0x00
PeakCur2_DutyCycle	Default: 0x00
PeakCur2_VBusVoltageDroop	Default: 0x00
PeakCur3_PercentOverload	Default: 0x00
PeakCur3_OverloadPeriod	Default: 0x00
PeakCur3_DutyCycle	Default: 0x00
PeakCur3_VBusVoltageDroop	Default: 0x00
TouchTemp	Default: 0x00
ExternalSupplyIsPresent	Default: 0x00
ExternalSupplyCondition	Default: 0x00
InternalBatteryIsPresent	Default: 0x00
Reserved_4	Default: 0x00
NumberOfFixedBatteries	Default: 0x00
NumberOfHotSwappableBatteries	Default: 0x00

2.45 PD_StatusMsg

Applied to Rev3.0. `PD_StatusMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `2` and the default value of `DataSize` field is `0x03`. Following are the additional fields(fields of `PD_StatusDataBlock` packet template) for this packet template:

Field Name	Description
InternalTemp	Default: 0x00
Reserved_1	Default: 0x00
ExternalPowerIsPresent	Default: 0x00
ExternalPower_AC_DC	Default: 0x00
InternalPowerBattery	Default: 0x00
InternalPowerNonBattery	Default: 0x00
Reserved_2	Default: 0x00
FixedBattery	Default: 0x00

HotSwappableBattery	Default: 0x00

2.46 PD_GetBatteryCapMsg

Applied to Rev3.0. `PD_GetBatteryCapMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `3` and the default value of `DataSize` field is `0x01`. Following are the additional fields(fields of `PD_GetBatteryCapDataBlock` packet template) for this packet template:

Field Name	Description
BatteryCapRef	Default: 0x00

2.47 PD_GetBatteryStatusMsg

Applied to Rev3.0. `PD_GetBatteryStatusMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `4` and the default value of `DataSize` field is `0x01`. Following are the additional fields(fields of `PD_GetBatteryStatusDataBlock` packet template) for this packet template:

Field Name	Description
BatteryStatusRef	Default: 0x00

2.48 PD_BatteryCapabilitiesMsg

Applied to Rev3.0. `PD_BatteryCapabilitiesMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `5` and the default value of `DataSize` field is `0x09`. Following are the additional fields(fields of `PD_BatteryCapDataBlock` packet template) for this packet template:

Field Name	Description
VendorId	Default: 0x00
ProductId	Default: 0x00
DesignCapacity	Default: 0x00
LastFullChargeCapacity	Default: 0x00
BatteryType	Default: 0x00

2.49 PD_GetManufacturerInfoMsg

Applied to Rev3.0. `PD_GetManufacturerInfoMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `6` and the default value of `DataSize` field is `0x02`. Following are the additional fields(fields of `PD_GetManufacturerInfoDataBlock` packet template) for this packet template:

Field Name	Description
Target	Default: 0x00
ManufacturerInfoRef	Default: 0x00

2.50 PD_ManufacturerInfoMsg

Applied to Rev3.0. `PD_ManufacturerInfoMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `7` and the default value of `DataSize` field is `0x04`. Following are the additional fields for this packet template:

Field Name	Description
VendorId	Default: 0x00
ProductId	Default: 0x00
ManufacturerString	Default: null Can be initialized using a byte stream

2.51 PD_SecurityRequestMsg

Applied to Rev3.0. PD_SecurityRequestMsg packet template contains all the fields of PD_ExtMsgHeaders but the default value of MessageType field is 8. Following are the additional fields for this packet template:

Field Name	Description
SecurityRequestDB	Can contain only one Security Request Data Block. Available SRDB types are: PD_SRQDB_GetDigests, PD_SRQDB_GetCertificate, PD_SRQDB_Challenge

2.51.1 PD_SRQDB_GetDigests

Used as SecurityRequestDB for PD_SecurityRequestMsg packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: PD_AUTH_PROT_VER_1
AuthMessageType	Default: PD_AUTH_TYPE_GET_DIGESTS
AuthParam1	Default: 0x00
AuthParam2	Default: 0x00

2.51.2 PD_SRQDB_GetCertificate

Used as SecurityRequestDB for PD_SecurityRequestMsg packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: PD_AUTH_PROT_VER_1
AuthMessageType	Default: PD_AUTH_TYPE_GET_CERTIFICATE
AuthParam1	Default: 0x00
AuthParam2	Default: 0x00
Offset	Default: 0x00
Length	Default: 0x00

2.51.3 PD_SRQDB_Challenge

Used as SecurityRequestDB for PD_SecurityRequestMsg packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: PD_AUTH_PROT_VER_1
AuthMessageType	Default: PD_AUTH_TYPE_GET_CHALLENGE
AuthParam1	Default: 0x00
AuthParam2	Default: 0x00
Nonce	Default: { 00 00 00 00 }

2.52 PD_SecurityResponseMsg

Applied to Rev3.0. `PD_SecurityResponseMsg` packet template contains all the fields of `PD_ExtMsgHeaders` but the default value of `MessageType` field is `9`. Following are the additional fields for this packet template:

Field Name	Description
SecurityResponseDB	Can contain only one Security Response Data Block. Available SRPDB types are: `PD_SRPDB_Digests`, `PD_SRPDB_Certificate`, `PD_SRPDB_ChallengeAuth`, `PD_SRPDB_Error`

2.52.1 PD_SRPDB_Digests

Used as `SecurityResponseDB` for `PD_SecurityResponseMsg` packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: `PD_AUTH_PROT_VER_1`
AuthMessageType	Default: `PD_AUTH_TYPE_DIGESTS`
AuthParam1	Default: `0x01`
AuthParam2	Default: `0x00`
DigestArray	Max len is 256 bytes, each digest is 32 bytes. Packet variables of `PD_Security_Digest` type, can be assigned to this field.

2.52.1.1 PD_Security_Digest

Used as `DigestArray` of `PD_SRPDB_Digests` packet template. Available fields of this packet template are:

Field Name	Description
Digest	Default: `0x00`

2.52.2 PD_SRPDB_Certificate

Used as `SecurityResponseDB` for `PD_SecurityResponseMsg` packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: `PD_AUTH_PROT_VER_1`
AuthMessageType	Default: `PD_AUTH_TYPE_CERTIFICATE`
AuthParam1	Default: `0x00`
AuthParam2	Default: `0x00`
Certificate	Default: `null` Can be initialized using a byte stream.

2.52.3 PD_SRPDB_ChallengeAuth

Used as `SecurityResponseDB` for `PD_SecurityResponseMsg` packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: `PD_AUTH_PROT_VER_1`
AuthMessageType	Default: `PD_AUTH_TYPE_CHALLENGE_AUTH`
AuthParam1	Default: `0x00`
AuthParam2	Default: `0x00`

MinProtVer	Default: 0x00
MaxProtVer	Default: 0x00
Capabilities	Default: 0x01
Rsvd	Default: 0x00
CertChainHash	Default: { 00 00 00 00 }
Salt	Default: { 00 00 00 00 }
ContextHash	Default: { 00 00 00 00 }
Signature	Default: { 00 00 00 00 }

2.52.4 PD_SRPDB_Error

Used as `SecurityResponseDB` for `PD_SecurityResponseMsg` packet template. Available data fields for this packet template are:

Field Name	Description
AuthProtocolVersion	Default: PD_AUTH_PROT_VER_1
AuthMessageType	Default: PD_AUTH_TYPE_ERROR
AuthParam1	Default: 0x01
AuthParam2	Default: 0x00

3 Type-C Commands

In addition to Power Delivery commands, PD Exerciser also provides a command set to manage USB Type-C connection . It includes some low level commands for manipulating voltages, capacitors and resistors as well as some high level commands that let you have SINK, SINKAS, SOURCE and DRP state machines, described in Type-C specification, with the facilities to customize different behaviors and characteristics. Note that at the present, Type-C state machines are just followed when related commands are running. In other words, Type-C state machines are not followed in parallel to other Power Delivery commands execution.

3.1 PD_SetResistorRp

Sets resistor Rp On/Off.

Format

```
Call PD_SetResistorRp( state, current, line )
```

Parameters

`state`

Possible values: PD_ON, PD_OFF

`current`

Possible values:
```
CC_RP_CUR_DEFAULT
CC_RP_CUR_1_5
CC_RP_CUR_3_0
```

`line`

Possible values:
```
CC_LINE_1
CC_LINE_2
CC_LINE_ALL
```

Examples

```
Call PD_SetResistorRP( PD_ON, CC_RP_CUR_1_5, CC_LINE_2 )
```

3.2 PD_SetResistorRd

Sets resistor Rd On/Off.

Format

```
Call PD_SetResistorRd( state, line )
```

Parameters

`state`

Possible values:
```
PD_ON
PD_OFF
```

`line`

Possible values:
```
CC_LINE_1
CC_LINE_2
```

`CC_LINE_ALL`

Examples

```
Call PD_SetResistorRd( PD_ON, CC_LINE_1 )
```

3.3 PD_SetResistorRa

Sets resistor Ra On/Off.

Format

```
Call PD_SetResistorRa( state, line )
```

Parameters

`state`

Possible values:

```
PD_ON
PD_OFF
```

`line`

Possible values:

```
CC_LINE_1
CC_LINE_2
CC_LINE_ALL
```

Examples

```
Call PD_SetResistorRa( PD_ON, CC_LINE_2 )
```

3.4 PD_SetVBusCap10MicroFarad

Sets the VBus Capacitor(10 Micro Farad) On/Off.

Format

```
Call PD_SetVBusCap10MicroFarad( state )
```

Parameters

`state`

Possible values:

```
PD_ON
PD_OFF
```

Examples

```
Call PD_SetVBusCap10MicroFarad( PD_ON )
```

3.5 PD_SetVBusCap1MicroFarad

Sets the VBus Capacitor(1 Micro Farad) On/Off.

Format

```
Call PD_SetVBusCap1MicroFarad( state )
```

Parameters

`state`

Possible values:

```
PD_ON
```

```
PD_OFF
```

Examples

```
Call PD_SetVBusCap1MicroFarad( PD_ON )
```

3.6 PD_SetVBus

Sets VBus On/Off.

Format

```
Call PD_SetVBus( state, voltage_milli_volt )
```

Parameters

`state`

Possible values:

```
PD_ON
PD_OFF
```

`voltage_milli_volt`

The voltage which applied on VBus. Voltage should be in range of 5000 to 20500 mV. In order to apply voltages greater than 5V, the corresponding check box should be set in recording options.

Examples

```
Call PD_SetVBus( PD_ON, 5000 )
```

3.7 PD_SetVConn

Sets VConn On/Off.

Format

```
Call PD_SetVConn( state )
```

Parameters

`state`

Possible values:

```
PD_ON
PD_OFF
```

Examples

```
Call PD_SetVConn( PD_ON )
```

3.8 PD_SetLoadOnVBus

Enables/Disables load on VBus.

Format

```
Call PD_SetLoadOnVBus( state )
```

Parameters

`state`

Possible values:

```
PD_ON
PD_OFF
```

Examples

```
Call PD_SetLoadOnVBus( PD_ON )
```

3.9 PD_TerminateCCLines

Terminates CC lines with the specified resistors in parameters.

Format

```
Call PD_TerminateCCLines( CC1_Resistor, CC2_Resistor )
```

Parameters

`CC1_Resistor`

Possible values:
CC_OPEN
CC_RP
CC_RP_1_5
CC_RP_3_0
CC_RD
CC_RA

`CC2_Resistor`

Possible values:
CC_OPEN
CC_RP
CC_RP_1_5
CC_RP_3_0
CC_RD
CC_RA

Result

None

Examples

```
Call PD_TerminateCCLines( CC_RP_1_5, CC_OPEN )
```

3.10 PD_SetStartDRPSetting

It is used to customize behavior and characteristics of the Exerciser when acts as a DRP device. Settings are applied to `PD_StartDRP` command.

Format

```
Call PD_SetStartDRPSetting( PD_Start_DRP_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_Start_DRP_Settings`. This type contains following data fields:

Field Name	Description
Timeout	Indicates the timeout in micro second for connecting as a SINK or SOURCE. Should be greater than 5000us. Default: PD_DEFAULT_TIMEOUT_INFINIT
WithRa	Indicates whether to provide Ra on CC2 line or not. Default: PD_FALSE
AdvertizedCurrent	Indicates advertised current level on Rp. Default: CC_RP_CUR_1_5
WithVConn	Indicates whether to turn on the VConn or not. Default: PD_FALSE

StartWithSNK	If is set to PD_TRUE, DRP state machine starts from Unattached.SNK state instead of Unattached.SRC state. Default: PD_FALSE
WithTrySRC	If is set to PD_TRUE, Exerciser supports Try.SRC state machine. Default: PD_FALSE
WithTrySNK	If is set to PD_TRUE, Exerciser supports Try.SNK state machine. Default: PD_FALSE

Result

None

Examples

```
$startdrp_setting = PD_Start_DRP_Settings
{
   WithTrySRC = PD_TRUE
}
Call PD_SetStartDRPSetting( $startdrp_setting )
```

3.11 PD_StartDRP

It starts DRP state machine for connecting to a Type-C device. The command quits if timeouts or Exerciser transitions to Attached.Src or Attached.SNK.

Format

```
Call PD_StartDRP()
```

Parameters

None

Result

Result Values	Description
PD_RESULT_OK	
PD_RESULT_FAILED	

Examples

```
Call PD_StartDRP()
```

3.12 PD_SetStartSourceSetting

It is used to customize behavior and characteristics of the Exerciser when acts as a SOURCE device. Settings are applied to `PD_StartSource` command.

Format

```
Call PD_SetStartSourceSetting( PD_Start_Source_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_Start_Source_Settings`. Available fields for this type are:

Field Names	Description
Timeout	Indicates the timeout (micro seconds) for connecting as SOURCE. Should be greater than 5000us. Default: PD_DEFAULT_TIMEOUT_INFINIT
WithRa	Indicates whether to provide Ra on CC2 line or not. Default: PD_FALSE

AdvertizedCurrent	Indicates advertised current level on Rp. Default: `CC_RP_CUR_1_5`
WithVConn	Indicates whether to turn on VConn or not. Default: `PD_FALSE`

Result

None

Examples

```
$startsrc_setting = PD_Start_Source_Settings
{
   WithRa = PD_TRUE
}
Call PD_SetStartSourceSetting( $startsrc_setting )
```

3.13 PD_StartSource

It starts SOURCE state machine for connecting to a Type-C device. The command is terminated if timeout occurs or the Exerciser transitions to Attached.SRC state.

Format

```
Call PD_StartSource()
```

Parameters

None

Result

Result Values	Description
PD_RESULT_OK	
PD_RESULT_FAILED	

Examples

```
Call PD_StartSource()
```

3.14 PD_SetStartSinkSetting

It is used to customize behavior and characteristics of the Exerciser when acts as a SINK device. Settings are applied to `PD_StartSink` command.

Format

```
Call PD_SetStartSinkSetting( PD_Start_Sink_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_Start_Sink_Settings`. Available fields of this type are:

Field Name	Description
Timeout	Indicates the timeout (micro second) for connecting as a SINK. Should be greater than `5000us`. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
WithRa	Indicates whether to provide Ra on CC2 line or not. Default: `PD_FALSE`
WithAccessory	Indicates whether to support SINKAS state machine or not. Default: `PD_FALSE`

AdvertizedCurrent	When `WithAccessory` setting is `PD_TRUE`: indicates advertised current level on Rp. Default: `CC_RP_CUR_1_5`
StartWithSNK	Applies when `WithAccessory` setting is `PD_TRUE`. If is set to PD_FALSE, SINKAS state machine starts from Unattached.Accessory state instead of Unattached.SNK state. Default: `PD_TRUE`
AccessoryStateDuration	When `WithAccessory` setting is `PD_TRUE`: indicates the time that Execiser stays in Powered.Accessory or Audio.Accessory states. Default: `1000000 us`
PoweredAccessoryExitState	When `WithAccessory` setting is `PD_TRUE`: indicates the exit state from Powered.Accessory state. Default: `PD_TYPE_C_STATE_NONE`

Result

None

Examples

```
$startsnk_setting = PD_Start_Sink_Settings
{
   WithAccessory = PD_TRUE
}
Call PD_SetStartSinkSetting( $startsnk_setting )
```

3.15 PD_StartSink

It starts SINK or SINKAS state machine for connecting to a Type-C device. The command is terminated if timeout occurs or Exerciser transitions to Attached.SNK. When Exerciser acts as SINKAS, with no exit state for Powered.Accessory state, that state will be the last state and command is terminated after specified time for this state duration.

Format

```
Call PD_StartSink()
```

Parameters

None

Result

Result Values	Description
PD_RESULT_OK	
PD_RESULT_FAILED	

Examples

```
Call PD_StartSink()
```

4 Basic Commands

4.1 PD_SendPacket

Sends the data payload towards the device. You can customize its behavior using provided settings.

Format

```
Call PD_SendPacket(PD_Packet $send_packet, PD_SendPacketSettings $settings)
```

Parameters

`$send_packet`

Defines the payload. Refer to `Packet Templates` for available packet templates.

`$settings`

Settings for sending packet. It should be inherited from `PD_SendPacketSettings` template. Table below shows `PD_SendPacketSettings` structure in detail:

Field Name	Description

OrderedSetType	Defines Ordered set type. Possible values:
	`PD_ORDERED_SET_TYPE_SOP`(default)
	`PD_ORDERED_SET_TYPE_SOP_PRIME`
	`PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME`
	`PD_ORDERED_SET_TYPE_HARDRESET`
	`PD_ORDERED_SET_TYPE_CABLERESET`

WaitForGoodCrc	If the command should wait for peer GoodCrc message. Possible values: `PD_TRUE`(default) `PD_FALSE`
ResetOnError	Send Soft Reset if relative GoodCrc has not been received, in case of sending SoftReset failure, HardReset will be sent. Possible values: `PD_TRUE`(default) `PD_FALSE`
RetryCount	Indicates the Retry Count. Default: `PD_DEFAULT_RETRY_COUNT_REV_2`(Rev2.0 only) Default: `PD_DEFAULT_RETRY_COUNT_REV_3`(Rev3.0)
RetryDelayTime	Delay time between two consecutive retries. Default: 0
AutoMessageId	To increase MessageId automatically. Possible values: `PD_TRUE`(default) `PD_FALSE`

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed
PD_SUBRESULT_NO_GOODCRC	Subresult - No GoodCRC received for sent packet
PD_SUBRESULT_HARDRESET	Subresult - HardReset occurred.
PD_SUBRESULT_SOFTRESET	Subresult - SoftReset occurred.

Examples

```
#send a discover identity command
#################################
$send_setting = PD_SendPacketSettings
{
   # could be PD_ORDERED_SET_TYPE_SOP_PRIME for cables
   OrderedSetType = PD_ORDERED_SET_TYPE_SOP
}
```

```
$discover_identity = PD_VDM_Discover_Identity_Message
Call PD_SendPacket( $discover_identity, $send_setting )

# Send Request message
######################
$request_data = PD_RequestDataObject_Fixed_Variable_NoGiveBack
{
    MaxOperatingCurrent_10mAUnits = 90
    OperatingCurrent_10mAUnits = 90
}
$request_packet = PD_RequestPacket
{
    Data = $request_data
}
#calling PD_SendPacket() command using default settings
$send_packet_settings = PD_SendPacketSettings
Call PD_SendPacket($request_packet, $send_packet_settings)
```

4.2 PD_SendPacket_Cable

Sends a packet as a Marked Cable towards the device.

Format

```
Call PD_SendPacket_Cable( PD_Packet $send_packet,
                          PD_SendPacketSettings_Cable $settings )
```

Parameters

`$send_packet`

Defines the payload. Refer to `Packet Templates` for available packet templates.

`$settings`

Settings for sending packet. It should be derived from `PD_SendPacketSettings_Cable` template. `PD_SendPacketSettings_Cable` is derived from `PD_SendPacketSettings` template. Default values for some fields is changed as below:

```
OrderedSetType = PD_ORDERED_SET_TYPE_SOP_PRIME
ResetOnErrors = PD_FALSE
RetryCount = 0
```

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of possible result values:

Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed
PD_SUBRESULT_NO_GOODCRC	Subresult - No GoodCRC received for sent packet

Examples

```
#send a discover identity response
####################################
$send_setting = PD_SendPacketSettings_Cable

$header_vdo = PD_VDM_Discover_Identity_ID_Header_VDO
$stat_vdo = PD_VDM_Discover_Identity_Cert_Stat_VDO
$product_vdo = PD_VDM_Discover_Identity_Product_VDO
$cable_vdo = PD_VDM_Discover_Identity_Cable_VDO

$discover_identity_response = PD_VDM_Discover_Identity_Response
{
   VDOs = $header_vdo + $stat_vdo + $product_vdo + $cable_vdo
}
Call PD_SendPacket_Cable( $discover_identity_response, $send_setting )
```

4.3 PD_SendCorruptedPacket

Sends a packet towards the Unit Under Test which is corrupted intentionally.

Format

```
Call PD_SendCorruptedPacket( PD_Packet $send_payload,
    PD_SendCorruptedPacketSettings $send_settings )
```

Parameters

`$send_payload`

The payload to be sent. Refer to `Packet Templates` for available packet templates.

`$send_settings`

Settings for sending the corrupted payload. Setting type is `PD_SendCorruptedPacketSettings`:

Field Name	Description
PreambleBitLen	Indicates the length of Preamble in bit. Default: `0x40`
NoPreamble	Indicates whether to insert the Preamble or not. Default: `PD_FALSE`
OrderedsetType	Indicates the ordered-set type. Possible values: `PD_ORDERED_SET_TYPE_SOP`(default), `PD_ORDERED_SET_TYPE_SOP_PRIME`, `PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME`, `PD_ORDERED_SET_TYPE_SOP_PRIME_DEBUG`, `PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME_DEBUG`, `PD_ORDERED_SET_TYPE_HARDRESET`, `PD_ORDERED_SET_TYPE_CABLERESET`, `PD_ORDERED_SET_TYPE_INVALID`
CorruptedOrderedset	If `OrderedsetType` field is `PD_ORDERED_SET_TYPE_INVALID` then content of this field will be replaced with ordered-set in the sent packet. Default: `0x00`
NoCrc	Indicates whether to insert Crc in the packet or not. Default: `PD_FALSE`
CorruptCrc4b	Indicates whether to corrupt Crc before 5-bit encoding or not. Default: `PD_FALSE`
CorruptCrc5b	Indicates whether to corrupt Crc after 5-bit encoding or not. Default: `PD_FALSE`
CorruptCrc5bSymbolIndex	Indicates the symbol index (starting from `0`) of 5-bit encoded Crc data to be corrupted (e.g. index `0` will corrupt the first 5-bit symbol in Crc data). This field will be processed if `CorruptCrc5b` is `PD_TRUE`. Default: `0x00`
CorruptCrc5bSymbolValue	Indicates the corrupted 5-bit symbol to be replaced with the 5-bit symbol indicated by `CorruptCrc5bSymbolIndex`. Will be processed if `CorruptCrc5b` is `PD_TRUE`. Default: `0x00`
CorruptPayload4b	Indicates whether to corrupt Payload before 5-bit encoding or not. Default: `PD_FALSE`
CorruptPayload5b	Indicates whether to corrupt Payload after 5-bit encoding or not. Default: `PD_FALSE`
CorruptPayload4bBitOffset	Indicates the bit offset of Payload (before 5-bit encoding) to get as the first data bit being corrupted (e.g. bit offset `0x08` means: get the Payload data corrupted starting from offset `0x08`). This field will be processed if `CorruptPayload4b` is `PD_TRUE`. Default: `0x00`
CorruptPaylaod4bBitLen	Indicates the bit length of Payload (before 5-bit encoding) to get corrupted.(e.g. bit length `0x03` means: corrupt Payload(before 5-

	bit encoding) starting from `CorruptPayload4bBitOffset` and length of 0x03 bits). This field will be processed if `CorruptPayload4b` is PD_TRUE. Default: 0x00
CorruptPaylaod4bValue	Byte stream. Defines the value to be replaced with the Payload (before 5-bit encoding) data. The offset and length of replacing data should be defined using `CorruptPayload4bBitOffset` and `CorruptPaylaod4bBitLen` fields. This field will be processed if `CorruptPayload4b` field is PD_TRUE. Default: 0x00
CorruptPyload5bSymbolIndex	Indicates the symbol index (starting from 0) of 5-bit encoded Payload data to be corrupted (e.g. index 0 will corrupt the first 5-bit symbol in Payload data). This field will be processed if `CorruptPayload5b` field is PD_TRUE. Default: 0x00
CorruptPyload5bSymbolValue	Indicates the corrupted 5-bit symbol to be replaced with the 5-bit symbol indicated by `CorruptPyload5bSymbolIndex`. Will be processed if `CorruptPayload5b` field is PD_TRUE. Default: 0x00
NoEop	Indicates whether to insert EOP in the packet or not. Default: PD_FALSE
CorruptEop	Indicates whether to corrupt EOP in th packet or not. Default: PD_FALSE
CorruptedEopSymbol	COrrupted EOP symbol to be replaced with EOP in the packet. This field will be processed if `CorruptEop` field is PD_TRUE.

Result

None

Examples

```
$GetSinkCapsPacket = PD_GetSinkCapMessage
{
    PortPowerRole_CablePlug = 1
}
$corrupted_send_settings = PD_SendCorruptedPacketSettings
{
    CorruptCrc4b = PD_TRUE
}

call PD_SendCorruptedPacket($GetSinkCapsPacket, $corrupted_send_settings)
```

4.4 PD_ReceivePacket

Receives a packet from device. You can specify the packet type using its settings.

Format

```
Call PD_ReceivePacket( PD_ReceivePacketSettings $receive_Settings )
```

Parameters

`$receive_Settings`

Settings for receiving packet. The structure type should be `PD_ReceivePacketSettings`. Table below shows this structure in detail:

Field Name	Description
OrderedSetType	Ordered set type for receiving message. Possible values: PD_ORDERED_SET_TYPE_SOP(default) PD_ORDERED_SET_TYPE_SOP_PRIME PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME PD_ORDERED_SET_TYPE_HARDRESET PD_ORDERED_SET_TYPE_CABLERESET

PacketType	Message type to receive. Possible values: PD_MESSAGE_TYPE_ANY(default) PD_MESSAGE_TYPE_GOODCRC PD_MESSAGE_TYPE_GOTO_MIN PD_MESSAGE_TYPE_ACCEPT PD_MESSAGE_TYPE_REJECT PD_MESSAGE_TYPE_PING PD_MESSAGE_TYPE_PS_RDY PD_MESSAGE_TYPE_GET_SOURCE_CAP PD_MESSAGE_TYPE_GET_SINK_CAP PD_MESSAGE_TYPE_DR_SWAP PD_MESSAGE_TYPE_PR_SWAP PD_MESSAGE_TYPE_VCONN_SWAP PD_MESSAGE_TYPE_WAIT PD_MESSAGE_TYPE_SOFT_RESET PD_MESSAGE_TYPE_SOURCE_CAP PD_MESSAGE_TYPE_REQUEST PD_MESSAGE_TYPE_BIST PD_MESSAGE_TYPE_SINK_CAP PD_MESSAGE_TYPE_VDM PD_MESSAGE_TYPE_NOT_SUPPORTED PD_MESSAGE_TYPE_GET_SRC_CAP_EXT PD_MESSAGE_TYPE_GET_STATUS PD_MESSAGE_TYPE_FR_SWAP PD_MESSAGE_TYPE_BATTERY_STATUS PD_MESSAGE_TYPE_ALERT PD_MESSAGE_TYPE_SRC_CAP_EXT PD_MESSAGE_TYPE_STATUS PD_MESSAGE_TYPE_GET_BATTERY_CAP PD_MESSAGE_TYPE_GET_BATTERY_STATUS PD_MESSAGE_TYPE_BATTERY_CAP PD_MESSAGE_TYPE_GET_MANUFACTURER_INFO PD_MESSAGE_TYPE_MANUFACTURER_INFO PD_MESSAGE_TYPE_SECURITY_REQUEST PD_MESSAGE_TYPE_SECURITY_RESPONSE
VdmCommand	VDM command. Possible values: PD_VDM_COMMAND_ANY(default) PD_VDM_COMMAND_DISCOVER_IDENTITY PD_VDM_COMMAND_DISCOVER_SVIDS PD_VDM_COMMAND_DISCOVER_MODES PD_VDM_COMMAND_ENTER_MODE PD_VDM_COMMAND_EXIT_MODE PD_VDM_COMMAND_DISPLAYPORT_STATUS_UPDATE PD_VDM_COMMAND_DISPLAYPORT_CONFIGURE PD_VDM_COMMAND_ATTENTION
VdmCommandType	VDM command type. Possible values: PD_VDM_COMMAND_TYPE_INITIATOR(default) PD_VDM_COMMAND_TYPE_RESPONDER_ACK PD_VDM_COMMAND_TYPE_RESPONDER_NAK PD_VDM_COMMAND_TYPE_RESPONDER_BUSY PD_VDM_COMMAND_TYPE_ANY
AutoGoodCrc	Send GoodCrc on receiving a message, automatically. Possible values: PD_TRUE(default) PD_FALSE
DelayBeforeGoodCrc	Delay before sending GoodCrc message. Default: 0
WaitTimeOut	Receive timeout(micro second). Possible values: PD_DEFAULT_TIMEOUT_SENDER_RESPONSE(default) PD_DEFAULT_TIMEOUT_INFINIT Or other user defined value.
DiscardPrevReceived	Discards any (unprocessed) packet received before calling `PD_ReceivePacket` function. Possible values: PD_TRUE PD_FALSE(default)
ReturnOnUnexpectedPkt	If set to PD_TRUE, cause `PD_ReceivePacket()` function to return on receiving unexpected packet. Possible values: PD_TRUE PD_FALSE(default)

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description

PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed
PD_SUBRESULT_RECEIVE_TIMEOUT	Subresult - No packet received within specified time
PD_SUBRESULT_UNEXPECTED_MSG_RECEIVED	Subresult - Unexpected packet received
PD_SUBRESULT_HARDRESET	Subresult - HardReset received
PD_SUBRESULT_SOFTRESET	Subresult - SoftReset received

Examples

```
#Receive source caps
####################
# Wait to receive source capability. GoodCRC is sent automatically.
$recv_settings = PD_ReceivePacketSettings
{
    WaitTimeOut = PD_DEFAULT_TIMEOUT_INFINIT
    PacketType = PD_MESSAGE_TYPE_SOURCE_CAP
}
call PD_ReceivePacket($recv_settings)

#Receive VDM message
####################
$receive_settings = PD_ReceivePacketSettings
{
    PacketType = PD_MESSAGE_TYPE_VDM
}
call PD_ReceivePacket( $receive_settings )
```

4.5 PD_SendSoftReset

Sends Soft Reset and performs the reset according to the selected Ordered-Set Type.

Format

```
Call PD_SendSoftReset( orderedset_type )
```

Parameters

`orderedset_type`

List of possible ordered set types:

OrderedSet Type	Description
PD_ORDERED_SET_TYPE_SOP	
PD_ORDERED_SET_TYPE_SOP_PRIME	
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME	

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
If sending SoftReset succeeded the result is `PD_RESULT_OK`. In case of failure it may lead to Power Negotiation.

Examples

```
Call PD_SendSoftReset( PD_ORDERED_SET_TYPE_SOP )
```

4.6 PD_SendHardReset

Sends Hard Reset and performs the reset.

Format

```
Call PD_SendHardReset()
```

Parameters

None

Result

None

Examples

```
Call PD_SendHardReset()
```

4.7 PD_SendCableReset

Sends Cable Reset and resets all the cable related states in protocol layer.

Format

```
Call PD_SendCableReset()
```

Parameters

None

Result

None

Examples

```
Call PD_SendCableReset()
```

4.8 PD_Delay

Delays Exerciser execution for specified time.

Format

```
Call PD_Delay( delay_value )
```

Parameters

`delay_value`

Delay in micro seconds.

Result

None

Examples

```
#calling PD_Delay
Call PD_Delay(15000)
```

4.9 PD_SetRoles

Sets data role and power role of Exerciser.

Format

```
Call PD_SetRoles( DataRole, PowerRole )
```

Parameters

`DataRole`

Possible values:
`PD_PORT_DATA_ROLE_UFP`
`PD_PORT_DATA_ROLE_DFP`

`PowerRole`

Possible values:
`PD_PORT_POWER_ROLE_SINK`
`PD_PORT_POWER_ROLE_SOURCE`

Examples

```
Call PD_SetRoles( PD_PORT_DATA_ROLE_DFP, PD_PORT_POWER_ROLE_SOURCE )
```

4.10 PD_Set

Using this command you can change necessary settings or variables inside the Exerciser.

Format

```
PD_Set $PdGlobalSettings.<field_name> = <value>
PD_Set $PdTimers.<field_name> = <value>
```

Parameters

List of `$PdGlobalSettings` fields:

Field Name	Description
PortDataRole	Defines port data role. Possible values: `PD_PORT_DATA_ROLE_DFP` `PD_PORT_DATA_ROLE_UFP`(default)
PortPowerRole	Defines port power role. Possible values: `PD_PORT_POWER_ROLE_SINK`(default) `PD_PORT_POWER_ROLE_SOURCE`
CheckMessageId	Enables/Disables received packet message id verification. Possible values: `PD_FALSE`(Default) `PD_TRUE`

SpecificationRevision	Changes the `SpecificationRevision` of all messages sent by the Exerciser.
	Possible values:
	`PD_SPEC_REVISION_1`
	`PD_SPEC_REVISION_2`(Default)
	`PD_SPEC_REVISION_3`
	Or any user defined value.

EnableCableEmulator	Enables/Disables Cable Emulator engine in Exerciser. If enabled, the Exerciser simulates a Marked Cable as well as source or sink PD Device. *It should be set only once in the target Exerciser Script*. Possible values: `PD_FALSE`(Default) `PD_TRUE`
EnableDeviceEmulator	If disabled, Device Emulator AutoResponse will be disabled (in case of Exerciser acting as a Sink or Source device). Possible values: `PD_FALSE`, `PD_TRUE`(default)
NegotiateAfterReset	If set to `PD_TRUE`, then the Exerciser will run Negotiation after receiving/sending SoftReset or HardReset. Possible values: `PD_FALSE`, `PD_TRUE` (default)
VConnPassThrough	Indicates whether the Exerciser is connected to the DUT using a VConn

	Pass Through cable or not. Possible values: PD_FALSE(default), PD_TRUE
PDWorkingRevision	Sets the Power Delivery working revision. *It should be set only once in the target Exerciser Script*. Its recommended to change this setting using PD_SetWorkingRevision high-level function. Possible values: PD_SPEC_REVISION_2(Default) PD_SPEC_REVISION_3
UnchunkedSupport	Indicates whether to support sending un-chunked messages or not. Possible values: PD_FALSE, PD_TRUE (default)
StructuredVDMVersion	Indicates the VDM version of structured VDM messages. If the value is PD_INVALID_VALUE then the Exerciser puts the proper value for structured VDM version according to current operational Power Delivery Revision. Possible values: PD_INVALID_VALUE(default) Or any user defined value.

List of `$PdTimers` fields(for detailed description refer to Power Delivery Specification):

Field Name	Description
tTypeCSinkWaitCap	Default: 620000 us
tTypeCSendSourceCap	Default: 150000 us
tPSTransition	Default: 550000 us
tPSSourceOff	Default: 900000 us
tPSSourceOn	Default: 450000 us
tSrcTransition	Default: 30000 us
tDiscoverIdentity	Default: 45000 us
tSafe0V	Default: 650000 us
tSafe5V	Default: 275000 us
tPSHardResetMin	Default: 25000 us
tPSHardResetMax	Default: 35000 us
tPSHardReset	Default: 30000 us
tSrcRecoverMin	Default: 660000 us
tSrcRecoverMax	Default: 1000000 us
tSrcRecover	Default: 1000000 us
tReceive	Default: 1100 us
tVCONNStable	Default: 50000 us
tVCONNSourceOff	Default: 25000 us
tVCONNSourceOn	Default: 100000 us
tSenderResponse	Default: 30000 us
tBISTContMode	Default: 60000 us
tVDMBusy	Default: 50000 us
tVDMWaitModeEntry	Default: 50000 us
tVDMWaitModeExit	Default: 50000 us
tSwapSourceStart	Default: 20000 us
tDRP	Default: 80000 us
dcSRC_DRP	Default: 50(time percent)
tCCDebounce	Default: 150000 us
tCCDebounceMin	Default: 100000 us
tCCDebounceMax	Default: 200000 us
tDRPTry	Default: 150000 us
tDRPTryMin	Default: 75000 us
tDRPTryMax	Default: 150000 us
tDRPTryWait	Default: 600000 us
tDRPTryWaitMin	Default: 400000 us
tDRPTryWaitMax	Default: 800000 us
tPDDebounce	Default: 15000 us

tPDDebounceMin	Default: 10000 us
tPDDebounceMax	Default: 20000 us
tSinkTx	Default: 18000 us
tFRSwapTx	Default: 110 us
tFRSwapInit	Default: 15000 us
tErrorRecovery	Default: 25000 us

Result

None

Examples

```
# Enables cable emulator
PD_Set $PdGLOBALSETTINGS.EnableCableEmulator = PD_TRUE

Main
{
   # Sets GoodCRC timeout
   PD_Set $PdTimers.tReceive = 950

   Call PD_WaitForDiscoverIdentity_Cable()
}
```

4.11 IfMatched/ElseMatched

Compares Exerciser settings, Received Packet Fields and Command Results to a desired value.

Using this command you can compare Exerciser settings or variables to other Exerciser settings or variables or to a constant.

Format

```
Ifmatched(<1st_operand>, <2nd_operand>, <operator>)
{
    #command list
}
[
ElseMatched(<1st_operand>, <2nd_operand>, <operator>)
{
 #command list
}
 #more optional ElseMatched(<1st_operand>, <2nd_operand>, <operator>) here
  .
  .
  .
ElseMatched
{
     #command list
}
]
IfMatchedEnd

* ElseMatched clause is optional
```

Parameters

1st_operand

1st operand should be in one of the following formats:
```
   $PdGlobalSettings.<field_name>
   $PdResult.<field_name>
   $<packet_variable>.<field_name>
```

List of `$PdResult` fields:

Field Name	Description
Result	Last executed command result
Subresult	Last executed command subresult (in case of failure, this field describes the reason)
LastReceivedPacketOrderedSet	Last received packet ordered set type
LastReceivedPacketType	Last received packet type
LastReceivedPacketPowerRole	Last received packet power role field value
LastReceivedPacketDataRole	Last received packet data role field value
LastReceivedPacketSentToCable	Indicates whether the last received packet has been sent to cable(packet towards the cable) or not
LastReceivedPacketMsgID	Last received packet `MessageId` field value
LastReceivedPacketVdmCommand	Last received packet VDM command value, if the packet is VDM packet
LastReceivedPacketVdmCommandType	Last received packet VDM command type value, if the packet is VDM packet
LastReceivedPacketVdmSVID	Last received packet SVID, if the packet is a VDM packet
LastReceivedPacketVdmObjPos	Last received packet `ObjetctPosition`, if the packet is a VDM packet
LastSelectedCapIndex	Last received packet selected capability index, if the packet is Request message
LastRequestHasMismatch	Last received packet `HasMismatch` field value, if the packet is Request message
ExplicitContract	Indicates whether explicit contract is established or not.

For available `$PdGlobalSettings` fields refer to `PD_Set`.

2nd_operand

It could be as `<1st_operand>` or a constant `<value>`.

operator

List of possible values for operator:

Operator	Description
PD_COMPARE_EQUAL	Equal
PD_COMPARE_GREATER	Greater than
PD_COMPARE_LESS	Less than
PD_COMPARE_NOT_EQUAL	Not equal

Result

None

Examples

```
$send_setting = PD_SendPacketSettings
{
   ResetOnError = PD_FALSE
    OrderedSetType = PD_ORDERED_SET_TYPE_SOP
}
$receive_settings = PD_ReceivePacketSettings
{
    PacketType = PD_MESSAGE_TYPE_VDM
}
#send the packet
$discover_identity = PD_VDM_Discover_Identity_Message
Call PD_SendPacket( $discover_identity, $send_setting )
```

```
#check for result
IfMatched( $PdResult.Result, PD_RESULT_OK, PD_COMPARE_EQUAL )
{
   Call PD_ReceivePacket( $receive_settings )
}
ElseMatched( $PdResult.Result, PD_RESULT_FAILED, PD_COMPARE_EQUAL )
{
   Call PD_SendHardReset()
}
ElseMatched
{
   $ping_msg = PD_PingMessage
   Call PD_SendPacket( $ping_msg, $send_setting )
}
IfMatchedEnd
```

4.12 PD_Loop

Using this command you can create a loop containing other Exerciser commands.

Note - The limit for using nested `PD_Loop()` commands is 8.

Format

```
PD_Loop(count)
{
    #command list
}
```

Parameters

`count`

Loop count

Result

None

Examples

```
$send_setting = PD_SendPacketSettings
{
    OrderedSetType = PD_ORDERED_SET_TYPE_SOP
}
$ping_msg = PD_PingMessage

PD_Loop(3)
{
   call PD_SendPacket( $ping_msg, $send_setting )
}
```

4.13 PD_TimerLoop

Using this command you can create a loop(containing other Exerciser commands) which is bound to a predefined timer. On timer timeout, the loop will exit.

Note - The limit for using nested `PD_TimerLoop()` commands is 8.

Format

```
PD_TimerLoop(timeout)
{
    #command list
}
```

Parameters

`timeout`

Loop duration in Micro Seconds.

Result

None

Examples

```
$send_setting = PD_SendPacketSettings
{
    OrderedSetType = PD_ORDERED_SET_TYPE_SOP
}
$ping_msg = PD_PingMessage

# Sending Ping message for 200ms
PD_TimerLoop(200000)
{
   call PD_SendPacket( $ping_msg, $send_setting )
}
```

4.14 PD_BreakLoop

Breaks the `PD_Loop` and `PD_TimerLoop` commands.

Format

`PD_BreakLoop()`

Parameters

None

Result

None

Examples

```
$send_setting = PD_SendPacketSettings
{
    OrderedSetType = PD_ORDERED_SET_TYPE_SOP
}

PD_Loop(3)
{
    PD_Loop(2)
    {
        $accept_msg = PD_AcceptMessage
        call PD_SendPacket( $accept_msg, $send_setting )
    }

    $ping_msg = PD_PingMessage
    call PD_SendPacket( $ping_msg, $send_setting )

    IfMatched( $PdResult.Result, PD_RESULT_OK, PD_COMPARE_EQUAL )
    {
        PD_BreakLoop()
    }
    IfMatchedEnd
}
```

4.15 PD_ContinueLoop

Continue command for `PD_Loop` and `PD_TimerLoop` comamnds.

Format

`PD_ContinueLoop()`

Parameters

None

Result

None

Examples

```
$send_setting = PD_SendPacketSettings
{
    OrderedSetType = PD_ORDERED_SET_TYPE_SOP
}

PD_Loop(3)
{
   $ping_msg = PD_PingMessage
   call PD_SendPacket( $ping_msg, $send_setting )

   IfMatched( $PdResult.Result, PD_RESULT_OK, PD_COMPARE_EQUAL )
   {
           PD_ContinueLoop()
   }
   IfMatchedEnd

   Call PD_SendSoftReset( PD_ORDERED_SET_TYPE_SOP )
}
```

4.16 PD_Stop

Stops the Exerciser.

Format

```
Call PD_Stop( return_value )
```

Parameters

`return_value`

Value returned to Exerciser.

Result

None

Examples

```
Call PD_Stop(0)
```

4.17 PD_Disconnect

Simulates cable detach.

Format

```
Call PD_Disconnect()
```

Parameters

None

Result

None

Examples

```
Call PD_Disconnect()
```

4.18 PD_ResumeUSB2Exerciser

Resumes USB2 Exerciser execution. **Not intended or supported for Customer Use.**

Format

```
Call PD_ResumeUSB2Exerciser()
```

Parameters

None

Result

None

Examples

```
Call PD_ResumeUSB2Exerciser()
```

4.19 PD_ReportUSB3TermStatus

Reports USB3 TermStatus. **Not intended or supported for Customer Use.**

Format

```
Call PD_ReportUSB3TermStatus()
```

Parameters

None

Result

None

Examples

```
Call PD_ReportUSB3TermStatus()
```

4.20 PD_IncreaseMsgId

Increase Message ID(Exerciser mode: DFP/UFP).

Format

```
Call PD_IncreaseMsgId(OrderedSetType)
```

Parameters

`OrderedSetType`

Indicates the OrderedSet type. Possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

None

Examples

```
Call PD_IncreaseMsgId(PD_ORDERED_SET_TYPE_SOP)
```

4.21 PD_DecreaseMsgId

Decrease Message ID(Exerciser mode: DFP/UFP).

Format

```
Call PD_DecreaseMsgId(OrderedSetType)
```

Parameters

`OrderedSetType`

Indicates the OrderedSet type. Possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

None

Examples

```
Call PD_DecreaseMsgId(PD_ORDERED_SET_TYPE_SOP)
```

4.22 PD_IncreaseMsgId_Cable

Increase Message ID(Exerciser mode: Cable Emulator).

Format

```
Call PD_IncreaseMsgId_Cable(OrderedSetType)
```

Parameters

`OrderedSetType`

Indicates the OrderedSet type. Possible values:

```
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

None

Examples

```
Call PD_IncreaseMsgId_Cable(PD_ORDERED_SET_TYPE_SOP_PRIME)
```

4.23 PD_DecreaseMsgId_Cable

Decrease Message ID(Exerciser mode: Cable Emulator).

Format

```
Call PD_DecreaseMsgId_Cable(OrderedSetType)
```

Parameters

`OrderedSetType`

Indicates the OrderedSet type. Possible values:

```
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

None

Examples

```
Call PD_DecreaseMsgId_Cable(PD_ORDERED_SET_TYPE_SOP_PRIME)
```

5 Transaction Engine™

Power Delivery Transaction Engine™ includes high level commands and auto response capability.

5.1 High Level Commands

5.1.1 PD_SetWorkingRevision

Sets the Exerciser working revision along with Specification Revision. It should call once in whole Exerciser script. The default working revision is `PD_SPEC_REVISION_2`.

Format

```
Call PD_SetWorkingRevision( revision )
```

Parameters

`revision`

Indicates the target revision.
Possible values:
`PD_SPEC_REVISION_2`(default),
`PD_SPEC_REVISION_3`

Result

None

Examples

```
Call PD_SetWorkingRevision( PD_SPEC_REVISION_3 )
```

5.1.2 PD_SetNegotiationSetting_Source

Applies settings to power negotiation related commands as Source in PD Exerciser. If the user wants to change default settings for Source Power Negotiation, must call this command before `PD_NegotiatePower_Source` or `PD_NegotiatePower` or `PD_WaitForNegotiatePower` commands to take effect.

Format

```
Call PD_SetNegotiationSetting_Source( PD_Negotiation_Source_Settings $settings )
```

Parameters

`$settings`

Defines negotiation settings for source. Should be in type of PD_Negotiation_Source_Settings template. Table below shows all available fields of PD_Negotiation_Source_Settings template:

Field Name	Description

NegotiationResponse	Indicates the response type. Possible values:
	`PD_NEGOTIATION_ACCEPT`(default)
	`PD_NEGOTIATION_WAIT`
	`PD_NEGOTIATION_REJECT`

SourceCapsRetryCount	Source capabilities retry count.
VBusVoltage_mv	VBus voltage in millivolt.
SourceCapMsgSpecRev	If the value is not `PD_INVALID_VALUE` then SourceCap

	Message in Negotiation sequence will be transferred using this Specification Revision. Possible values: `PD_INVALID_VALUE`(default) Or other user defined value.
AutoSpecRev	Rev3.0 only. Indicates whether the Exerciser should detect the Specification Revision automatically from Negotiation sequence or not. Possible values: `PD_FALSE`, `PD_TRUE`(default)
AutoUnchunkedSupport	Rev3.0 only. Indicates whether the Exerciser should detect the Un-chunked Support automatically from Negotiation sequence or not. Possible values: `PD_FALSE`, `PD_TRUE`(default)

Note - If user sets the `VBusVoltage_mv`, then the PD Exerciser will set `VBusVoltage_mv` on the `VBus` regardless the actual voltage value which UUT selected during the negotiation process, otherwise the Exerciser will set the `VBus` using the voltage which UUT selected during the negotiation process.

Note - In order to apply voltages greater than 5V, the corresponding check box should be set in recording options (*Allow VBUS > 5v*).

Result

None

Examples

```
#set negotiation using default values
$settings = PD_Negotiation_Source_Settings
call PD_SetNegotiationSetting_Source( $settings )

#set negotiation using reject as response
$settings
{
   NegotiationResponse = PD_NEGOTIATION_REJECT
}
call PD_SetNegotiationSetting_Source( $settings )
```

5.1.3 PD_AddSourceCap

Adds a specified Source Capability to the PD Exerciser. Before adding a group of source caps make sure that there is no unwanted source cap in the list by calling `PD_ResetSourceCaps` command. This command must be called before `PD_NegotiatePower_Source` or `PD_NegotiatePower` or `PD_WaitForNegotiatePower` commands to take effect.

Note - By default there is one pre-defined source cap(vSafe5V) in the list.

Format

```
Call PD_AddSourceCap(PD_PowerDataObject $PowerDataObject)
```

Parameters

`$PowerDataObject`

Parameter type is `PD_PowerDataObject`. Refer to `PD_SourceCapabilitiesMessage` for available source power data objects.

Result

None

Examples

```
local $power_data_object = PD_PowerDataObjectFixedSupply_Source
{
   MaxCurrent_10mAUnits = 20
   Voltage_50mVUnits = 250
}
call PD_AddSourceCap($power_data_object)
```

5.1.4 PD_ResetSourceCaps

Clears all Source Capabilities defined in PD Exerciser. Should be called before adding one or more source capabilities.

Format

```
Call PD_ResetSourceCaps()
```

Parameters

None

Result

None

Examples

```
call PD_ResetSourceCaps()
```

5.1.5 PD_NegotiatePower_Source

This command tries to establish an explicit contract as Source.

Format

```
Call PD_NegotiatePower_Source()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_REQUEST_MSG_INVALID_INDEX	Subresult - Invalid index in request message
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait has been sent as request message response
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been sent as request message response

Examples

```
call PD_NegotiatePower_Source()
```

5.1.6 PD_SetNegotiationSetting_Sink

Applies power negotiation settings as Sink. If the user wants to change default settings for Sink Power Negotiation, must call this command before `PD_NegotiatePower_Sink` or `PD_NegotiatePower` or `PD_WaitForNegotiatePower` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetNegotiationSetting_Sink( PD_Negotiation_Sink_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_Negotiation_Sink_Settings` type.
Table below shows all available fields of PD_Negotiation_Sink_Settings template:

Field Name	Description
WaitTimeout	Indicates the wait timeout(micro second) to receive SourceCapabilities Message. (default = PD_DEFAULT_TIMEOUT_INFINIT)
SinkRequestData	Defines data object of Request message.
AutoSinkRequest	Builds the `SinkRequestData` automatically according to current sink capabilities and received source capabilities. Possible values: PD_TRUE(Default) PD_FALSE
RetryCountOnWait	Indicates the retry count upon receiving Wait Message after sending the Request. Default: 2
RetryDelayOnWait	Indicates the delay time before retrying the Request, upon receiving Wait Message. Default: `100000 us`
RequestMsgSpecRev	If the value is not PD_INVALID_VALUE then Request Message in Negotiation sequence will be transferred using this Specification Revision. Possible values: PD_INVALID_VALUE(default) Or other user defined value.
ExTriggerOnAccept	Indicates whether to notify PD Exerciser through External Trigger on receiving Accept message or not. Possible values: PD_FALSE(default) PD_TRUE
ExTriggerOnPSRDY	Indicates whether to notify PD Exerciser through External Trigger on receiving PS_RDY message or not. Possible values: PD_FALSE(default) PD_TRUE
AutoSpecRev	Rev3.0 only. Indicates whether the Exerciser should detect the Specification Revision automatically from Negotiation sequence or not. Possible values: PD_FALSE, PD_TRUE(default)
AutoUnchunkedSupport	Rev3.0 only. Indicates whether the Exerciser should detect the Un-chunked Support automatically from Negotiation sequence or not.

	Possible values: PD_FALSE, PD_TRUE(default)

Result

None

Examples

```
#Set sink negotiation settings as default
$settings = PD_Negotiation_Sink_Settings
call PD_SetNegotiationSetting_Sink( $settings )
```

5.1.7 PD_AddSinkCap

Adds Sink Capabilities to PD Exerciser. Before adding a group of sink caps make sure that there is no unwanted sink cap in the list by calling `PD_ResetSinkCaps` command. This command must be called before `PD_NegotiatePower_Sink` or `PD_NegotiatePower` or `PD_WaitForNegotiatePower` or `PD_DelayAutoResponse` commands to take effect.

Note - By default there is one pre-defined sink cap in the list.

Format

```
Call PD_AddSinkCap(PD_PowerDataObject $PowerDataObject)
```

Parameters

`$PowerDataObject`

Parameter type is `PD_PowerDataObject`. Refer to `PD_SinkCapabilitiesMessage` for available sink power data objects.

Result

None

Examples

```
local $power_data_object = PD_PowerDataObjectFixedSupply_Sink
{
   OperationalCurrent_10mAUnits = 50
   Voltage_50mVUnits = 100
}
call PD_AddSinkCap($power_data_object)
```

5.1.8 PD_ResetSinkCaps

Clears all Sink Capabilities defined for PD Exercise. Should be called before adding one or more sink capabilities.

Format

```
Call PD_ResetSinkCaps()
```

Parameters

None

Result

None

Examples

```
call PD_ResetSinkCaps()
```

5.1.9 PD_NegotiatePower_Sink

Tries to establish explicit contract as Sink by sending Request message.

Format

```
Call PD_NegotiatePower_Sink()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject received as the response
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait received as the response
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - PS_RDY message not received.

Examples

```
call PD_NegotiatePower_Sink()
```

5.1.10 PD_WaitForNegotiatePower

Tries to establish explicit contract either as Source or Sink according to the current PD Exerciser power role. If the current power role of PD Exerciser is Source, this command will wait to receive Request message and if the current power role of PD Exerciser is Sink, it will wait to receive Source_Capabilities message.

Format

```
Call PD_WaitForNegotiatePower()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid

	also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject received as the response
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait received as the response
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Request(PD Exerciser as Source)/Source_Capabilities(PD Exerciser as Sink) message not received or PS_RDY message not received(PD Exerciser as Sink)

Examples

```
call Pd PD_WaitForNegotiatePower()
```

5.1.11 PD_NegotiatePower

Negotiates power with the peer port according to PD Exerciser current power role. If PD Exerciser operates as Source, this function starts power negotiation as Source and if the PD Exerciser operates as Sink, this function starts power negotiation as Sink(will wait to receive Request message).

Note - Both power negotiation settings can be applied to this function (by calling `PD_SetNegotiationSetting_Source` or `PD_SetNegotiationSetting_Sink` functions).

Format

```
Call PD_NegotiatePower()
```

Parameters

None

Result

If PD Exerciser operates as Source this function returns same sub-results as `PD_NegotiatePower_Source` function. If PD Exerciser operates as Sink this function returns same sub-results as `PD_NegotiatePower_Sink` function.

Examples

```
Call PD_NegotiatePower()
```

5.1.12 PD_SetSwapPowerRoleSetting

Applies settings to Swap Power Role related commands in PD Exerciser. It must be called before `PD_SwapPowerRole` or `PD_WaitForSwapPowerRole` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetSwapPowerRoleSetting(PD_SwapResponse_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_SwapResponse_Settings` type.
List of `SwapResponse_Settings` fields:

Field Name	Description
SwapResponse	Defines the response type. Possible values: PD_SWAPPOWERROLE_ACCEPT(default) PD_SWAPPOWERROLE_WAIT PD_SWAPPOWERROLE_REJECT
SkipSendingPSRDY	If set to PD_TRUE, PD_SwapPowerRole will not send the PS_RDY message. Possible values: PD_TRUE PD_FALSE(Default)
SkipSwap	If set to PD_TRUE, PD_SwapPowerRole will not swap the power role. Possible values: PD_TRUE PD_FALSE(Default)
WaitTimeout	Timeout(micro second) to wait in order to receive the PR_SWAP message Default: PD_DEFAULT_TIMEOUT_INFINIT
RetryCountOnWait	Indicates the retry count after receiving Wait Message in response to sent PR_Swap Message. Default: 2
RetryDelayOnWait	Indicates the retry delay time upon receiving Wait Message in response to sent PR_Swap Message. Default: 20000

Result

None

Examples

```
#Set swap power role settings as default
$settings = PD_SwapResponse_Settings
call PD_SetSwapPowerRoleSetting( $settings )
```

5.1.13 PD_SwapPowerRole

Tries to swap power role. It will start Swap Power Role AMS.

Format

```
Call PD_SwapPowerRole()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using IfMatched/ElseMatched command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for PD_SendPacket and PD_ReceivePacket are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - Response not received
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject received as response
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait received as response
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - PS_RDY not received(PD Exerciser as Sink)
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported received as response

Examples

```
call PD_SwapPowerRole()
```

5.1.14 PD_WaitForSwapPowerRole

Waits to receive PR_Swap message and will respond to incoming messages as part of the Swap Power Role AMS.

Format

```
Call PD_WaitForSwapPowerRole()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait has been sent as response
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been sent as response
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - PR_Swap message not received or PS_RDY message not received(PD Exerciser as Sink)
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported has been sent as response

Examples

```
call PD_WaitForSwapPowerRole()
```

5.1.15 PD_FastRoleSwap

Sends the FastRoleSwap Signal and handles Fast Role Swap AMS.

Note- Received FastRoleSwap Signal will handle by FastRoleSwap event handler automatically.

Format

```
Call PD_FastRoleSwap()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded.
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).

Examples

```
Call PD_FastRoleSwap()
```

5.1.16 PD_SetSwapDataRoleSetting

Applies settings to Swap Data Role related commands in PD Exerciser. It must be called before `PD_SwapDataRole` or `PD_WaitForSwapDataRole` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetSwapDataRoleSetting( PD_SwapResponse_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_SwapResponse_Settings` type. Table below describes the `PD_SwapResponse_Settings` template and settings related to Data Role Swap:

Field Name	Description
SwapResponse	Response type. Possible values: PD_MESSAGE_TYPE_ACCEPT(default) PD_MESSAGE_TYPE_REJECT PD_MESSAGE_TYPE_WAIT
WaitTimeout	Timeout(micro second) to wait in order to receive DR_SWAP message Default: PD_DEFAULT_TIMEOUT_INFINIT
RetryCountOnWait	Indicates the retry count after receiving Wait Message in response to sent DR_Swap Message. Default: 2
RetryDelayOnWait	Indicates the retry delay time upon receiving Wait Message in response to sent DR_Swap Message. Default: 20000

Result

None

Examples

```
$settings = PD_SwapResponse_Settings
{
   SwapResponse = PD_MESSAGE_TYPE_REJECT
}
call PD_SetSwapDataRoleSetting( $settings )
```

5.1.17 PD_SwapDataRole

Tries to swap the data role. It will start the Swap Data Role AMS.

Format

```
Call PD_SwapDataRole()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - Response not received
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been received
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait has been received
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported received as response

Examples

```
call PD_SwapDataRole()
```

5.1.18 PD_WaitForSwapDataRole

Waits for user-defined time-out to receive DR_Swap message and will respond to incoming messages as part of the Swap Data Role AMS.

Format

```
Call PD_WaitForSwapDataRole()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been sent as response
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DR_Swap not received
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported has been sent as response

Examples

```
call PD_WaitForSwapDataRole()
```

5.1.19 PD_SetSwapVConnSetting

Applies settings to Swap VConn related commands in PD Exerciser. It must be called before PD_SwapVConn or PD_WaitForSwapVConn or PD_DelayAutoResponse commands to take effect.

Format

```
Call PD_SetSwapVConnSetting( PD_SwapResponse_Settings $settings )
```

Parameters

`$settings`

Should be from PD_SwapResponse_Settings type. Table below describes the PD_SwapResponse_Settings template and specific settings related to Swap VConn:

Field Name	Description
SwapResponse	Response type. Possible values: PD_MESSAGE_TYPE_ACCEPT(default) PD_MESSAGE_TYPE_REJECT PD_MESSAGE_TYPE_WAIT
WaitTimeout	Timeout(micro second) to wait in order to receive VCONN_SWAP message Default: PD_DEFAULT_TIMEOUT_INFINIT
SkipSwap	If set to PD_TRUE, the command skips VConn swap. Possible values: PD_TRUE PD_FALSE(Default)
RetryCountOnWait	Indicates the retry count after receiving Wait Message in response to sent VConnSwap Message. Default: 2
RetryDelayOnWait	Indicates the retry delay time upon receiving Wait Message in response to sent VConnSwap Message. Default: 20000

Result

None

Examples

```
#Using default settings
$settings = PD_SwapResponse_Settings
call PD_SetSwapVConnSetting( $settings )
```

5.1.20 PD_SwapVConn

Tries to swap VConn. It will start the Swap VConn AMS.

Format

```
Call PD_SwapVConn()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using IfMatched/ElseMatched command.
List of result values:

Result Value	Description

PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - Response not received
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been received
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait has been received
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - PS_RDY not received(PD Exerciser as VCONN Source)
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported message has been received

Examples

```
call PD_SwapVConn()
```

5.1.21 PD_WaitForSwapVConn

Waits for user-defined time-out to receive VCONN_Swap message and will respond to incoming messages as part of Swap VConn AMS..

Format

```
Call PD_WaitForSwapVConn()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - VCONN Swap not received
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait has been sent as response
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been sent as response
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported has been sent as response

Examples

```
call PD_WaitForSwapVConn()
```

5.1.22 PD_SetGotoMinSetting

Applies settings to GotoMin related commands in PD Exerciser. It must be called before `PD_WaitForGotoMin` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetGotoMinSetting( PD_GotoMin_Settings $settings )
```

Parameters

`$settings`

Setting type is `PD_GotoMin_Settings`. Available fields of this type are:

Field Name	Description
WaitTimeout	Wait time-out(micro second) for receiving GotoMin message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates the response type upon receiving GotoMin message. Possible values: `PD_RESPONSE_UNSPECIFIED`(default), `PD_RESPONSE_NOT_SUPPORTED`

Result

None

Examples

```
$gotomin_setting = PD_GotoMin_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGotoMinSetting( $gotomin_setting )
```

5.1.23 PD_GotoMin

Starts the GotoMin AMS.

Format

```
Call PD_GotoMin()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` are valid also (depends on the error type which has been occurred during sending data).
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported message has been received

Examples

```
call PD_GotoMin()
```

5.1.24 PD_WaitForGotoMin

Waits for user-defined time-out to receive GotoMin message and will respond to incoming messages as part of GotoMin AMS.

Format

```
Call PD_WaitForGotoMin()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - GotoMin or PS_RDY message not received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported has been sent as response

Examples

```
call PD_WaitForGotoMin()
```

5.1.25 PD_SetGetSourceCapSetting

Applies settings to GetSourceCap related commands in PD Exerciser. It must be called before `PD_WaitForGetSourceCapabilities` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetGetSourceCapSetting( PD_GetCapability_Settings $settings )
```

Parameters

`$settings`

Setting type is `PD_GetCapability_Settings`. Available fields of this type are:

Field Name	Description
WaitTimeout	Wait time-out(micro second) for receiving GetSourceCap message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates the response type upon receiving GetSourceCap message. Possible values: `PD_RESPONSE_UNSPECIFIED`(default), `PD_RESPONSE_NOT_SUPPORTED`

Result

None

Examples

```
$getsrccap_setting = PD_GetCapability_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGetSourceCapSetting( $getsrccap_setting )
```

5.1.26 PD_GetSourceCapabilities

Starts GetSourceCapabilities AMS.

Format

```
Call PD_GetSourceCapabilities()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject received as response
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No messages received as response
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported received as response

Examples

```
Call PD_GetSourceCapabilities()
```

5.1.27 PD_WaitForGetSourceCapabilities

Waits for user-defined time-out to receive Get_Source_Cap message. It will respond to incoming messages as part of the Get_Source_Cap AMS.

Format

```
Call PD_WaitForGetSourceCapabilities()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Source_Cap message not received
PD_SUBRESULT_REQUEST_MSG_INVALID_INDEX	Subresult - Invalid index in request message
PD_SUBRESULT_RESPONSE_WAIT	Subresult - Wait has been sent as request message response

PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been sent as request message response
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported has been sent as response

Examples

```
Call PD_WaitForGetSourceCapabilities()
```

5.1.28 PD_SetGetSinkCapSetting

Applies settings to GetSinkCap related commands in PD Exerciser. It must be called before `PD_WaitForGetSinkCapabilities` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetGetSinkCapSetting( PD_GetCapability_Settings $settings )
```

Parameters

`$settings`

Setting type is PD_GetCapability_Settings. For available fields of this type refer to `PD_SetGetSourceCapSetting`.

Result

None

Examples

```
$getsnkcap_setting = PD_GetCapability_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGetSinkCapSetting( $getsnkcap_setting )
```

5.1.29 PD_GetSinkCapabilities

Starts the GetSinkCapabilities AMS.

Format

```
Call PD_GetSinkCapabilities()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject received as response
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No messages received as response
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported received as

	response

Examples

```
Call PD_GetSinkCapabilities()
```

5.1.30 PD_WaitForGetSinkCapabilities

Waits for user-defined timeout to receive Get_Sink_Cap message. It will respond to incoming messages as part of GetSinkCapabilities AMS.

Format

```
Call PD_WaitForGetSinkCapabilities()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Sink_Cap not received.
PD_SUBRESULT_RESPONSE_REJECT	Subresult - Reject has been sent as response.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Rev3.0 only. Subresult - Not_Supported has been sent as response

Examples

```
Call PD_WaitForGetSinkCapabilities()
```

5.1.31 PD_SendBISTCarrierMode

Starts BISTCarrierMode AMS.

Format

```
Call PD_SendBISTCarrierMode(OrderedSetType)
```

Parameters

`OrderedSetType`

Indicates the Ordered Set type
possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed

Examples

```
Call PD_SendBISTCarrierMode(PD_ORDERED_SET_TYPE_SOP)
```

5.1.32 PD_SendBISTTestData

Starts BISTTestData AMS.

Format

```
Call PD_SendBISTTestData( OrderedSetType, PD_BISTTestData $test_data )
```

Parameters

`OrderedSetType`

Indicates the Ordered Set type
possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`$test_data`

Defines the Test Data to be sent to the UUT

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` are valid also (depends on the error type which has been occurred during sending data).

Examples

```
$test_data = PD_BISTTestData
{
    TestData = { 00 00 00 00
                 AA AA AA AA
                 AA AA 00 00
                 AA AA AA AA
                 00 00 AA AA
                 AA AA AA AA }
}

Call PD_SendBISTTestData( PD_ORDERED_SET_TYPE_SOP_PRIME, $test_data )
```

5.1.33 PD_GetSourceCapExtended

Starts GetSourceCapExtended AMS.

Format

```
Call PD_GetSourceCapExtended()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message received.

Examples

```
Call PD_GetSourceCapExtended()
```

5.1.34 PD_SetGetSrcCapExtSetting

Applies settings to GetSourceCapExt related commands in PD Exerciser. It must be called before `PD_WaitForGetSrcCapExtended` or `PD_DelayAutoResponse` commands.

Format

```
Call PD_SetGetSrcCapExtSetting( PD_GetSourceCapExtented_Settings $settings )
```

Parameters

`$settings`

Setting type is `PD_GetSourceCapExtented_Settings`. Available fields for this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(micro second) to receive GetSourceCapExtended message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates response upon receiving the GetSourceCapExtended message. Possible values: `PD_RESPONSE_NOT_SUPPORTED`, `PD_RESPONSE_UNSPECIFIED`(default)
SkipSrcCapExt	Indicates whether to skip sending Source_Capabilities_Extended message or not. Possible values: `PD_TRUE`, `PD_FALSE`(default)
SendSrcCapExtDelay	Defines the delay before sending Source_Capabilities_Extended message. Default: `0`

Result

None

Examples

```
$getsrccapext_setting = PD_GetSourceCapExtented_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGetSrcCapExtSetting( $getsrccapext_setting )
```

5.1.35 PD_WaitForGetSrcCapExtended

Wait for user-defined time-out to receive Get_Source_Cap_Extended message. It will respond to incoming messages as part of GetSourceCapExtended AMS.

Format

```
Call PD_WaitForGetSrcCapExtended()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Source_Cap_Extended not received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message sent as response.

Examples

```
Call PD_WaitForGetSrcCapExtended()
```

5.1.36 PD_SetSrcCapExtDataBlock

Sets source capabilities extended Data Block in PD Exerciser. It must be called before `PD_WaitForGetSrcCapExtended` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetSrcCapExtDataBlock( PD_SourceCapExtDataBlock $src_cap_ext )
```

Parameters

`$src_cap_ext`

parameter type is `PD_SourceCapExtDataBlock`. Refer to `PD_SourceCapExtendedMsg` for available data fields.

Result

None

Examples

```
$src_cap_ext = PD_SourceCapExtDataBlock
Call PD_SetSrcCapExtDataBlock( $src_cap_ext )
```

5.1.37 PD_ResetSrcCapExtDataBlock

Clears the source capabilities extended Data Block in PD Exerciser. Should be called before calling `PD_SetSrcCapExtDataBlock`.

Format

```
Call PD_ResetSrcCapExtDataBlock()
```

Parameters

None

Result

None

Examples

```
Call PD_ResetSrcCapExtDataBlock()
```

5.1.38 PD_GetStatus

Starts the GetStatus AMS.

Format

```
Call PD_GetStatus()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message received.

Examples

```
Call PD_GetStatus()
```

5.1.39 PD_SetGetStatusSetting

Applies settings to GetStatus related commands in PD Exerciser. It must be called before `PD_WaitForGetStatus` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetGetStatusSetting( PD_GetStatus_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_GetStatus_Settings`. Available fields for this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(micro second) to receive GetStatus message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates response upon receiving the GetStatus message. Possible values: `PD_RESPONSE_NOT_SUPPORTED`, `PD_RESPONSE_UNSPECIFIED`(default)

Result

None

Examples

```
$getstatus_setting = PD_GetStatus_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGetStatusSetting( $getstatus_setting )
```

5.1.40 PD_WaitForGetStatus

Waits for user-defined time-out to receive Get_Status message. It will respond to incoming messages as part of GetStatus AMS.

Format

```
Call PD_WaitForGetStatus( )
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Status message not received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message sent as response.

Examples

```
Call PD_WaitForGetStatus()
```

5.1.41 PD_SetStatusDataBlock

Sets the Status Data Block in PD Exerciser. It must be called before `PD_WaitForGetStatus` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetStatusDataBlock( PD_StatusDataBlock $status_db )
```

Parameters

`$status_db`

Parameter type is `PD_StatusDataBlock`. Refer to `PD_StatusMsg` for available fields.

Result

None

Examples

```
$status_db = PD_StatusDataBlock
Call PD_SetStatusDataBlock( $status_db )
```

5.1.42 PD_ResetStatusDataBlock

Clears the Status Data Block in PD Exerciser. Should be called before calling `PD_SetStatusDataBlock` command.

Format

```
Call PD_ResetStatusDataBlock()
```

Parameters

None

Result

None

Examples

```
Call PD_ResetStatusDataBlock()
```

5.1.43 PD_GetBatteryStatus

Starts the GetBatteryStatus AMS.

Format

```
Call PD_GetBatteryStatus()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for PD_SendPacket and PD_ReceivePacket are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message received.

Examples

```
Call PD_GetBatteryStatus()
```

5.1.44 PD_SetGetBatteryStatusDataBlock

Sets the GetBatteryStatus Data Block in PD Exerciser. It must be called before `PD_GetBatteryStatus` command to take effect.

Format

```
Call PD_SetGetBatteryStatusDataBlock( PD_GetBatteryStatusDataBlock
$get_battery_stat_db )
```

Parameters

`$get_battery_stat_db`

Parameter type is `PD_GetBatteryStatusDataBlock`. Refer to `PD_GetBatteryStatusMsg` for available fields.

Result

None

Examples

```
$get_battery_stat_db = PD_GetBatteryStatusDataBlock
Call PD_SetGetBatteryStatusDataBlock( $get_battery_stat_db )
```

5.1.45 PD_SetGetBatteryStatusSetting

Applies settings to GetBatteryStatus related commands in PD Exerciser. It must be called before `PD_WaitForGetBatteryStatus` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetGetBatteryStatusSetting( PD_GetBatteryStatus_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_GetBatteryStatus_Settings`. Available fields of this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(micro second) to receive GetBatteryStatus message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates response upon receiving the GetBatteryStatus message. Possible values: `PD_RESPONSE_NOT_SUPPORTED`, `PD_RESPONSE_UNSPECIFIED`(default)

Result

None

Examples

```
$getbattstatus_setting = PD_GetBatteryStatus_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGetBatteryStatusSetting( $getbattstatus_setting )
```

5.1.46 PD_WaitForGetBatteryStatus

Waits for user-defined time-out to receive Get_Battery_Status message. It will respond to incoming messages as part of GetBatteryStatus AMS.

Format

```
Call PD_WaitForGetBatteryStatus()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Battery_Status message not received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message sent as response.

Examples

```
Call PD_WaitForGetBatteryStatus()
```

5.1.47 PD_SetBatteryStatusDO

Sets the BatteryStatus Data Object in PD Exerciser. It must be called before `PD_WaitForGetBatteryStatus` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetBatteryStatusDO( PD_BatteryStatusDataObject $battery_status )
```

Parameters

`$battery_status`

Parameter type is `PD_BatteryStatusDataObject`. Refer to `PD_BatteryStatusMsg` for available fields of this type.

Result

None

Examples

```
$battery_status = PD_BatteryStatusDataObject
Call PD_SetBatteryStatusDO( $battery_status )
```

5.1.48 PD_ResetBatteryStatusDO

Clears the BatteryStatus Data Object in PD Exerciser. Should be called before calling `PD_SetBatteryStatusDO` command.

Format

```
Call PD_ResetBatteryStatusDO()
```

Parameters

None

Result

None

Examples

```
Call PD_ResetBatteryStatusDO()
```

5.1.49 PD_Alert

Starts Alert AMS.

Format

```
Call PD_Alert()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` are valid also (depends on the error type which has been occurred during sending data).

Examples

```
Call PD_Alert()
```

5.1.50 PD_SetAlertDO

Sets Alert Data Object in PD Exerciser. It must be called before `PD_Alert` command to take effect.

Format

```
Call PD_SetAlertDO( PD_AlertDataObject $alert_do )
```

Parameters

`$alert_do`

Parameter type is `PD_AlertDataObject`. Refer to `PD_AlertMsg` for available fields of this type.

Result

None

Examples

```
$alert_do = PD_AlertDataObject
Call PD_SetAlertDO( $alert_do )
```

5.1.51 PD_SetAlertSetting

Applies settings to Alert related commands in PD Exerciser. It must be called before `PD_WaitForAlert` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetAlertSetting( PD_Alert_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_Alert_Settings.` Available fields for this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(micro second) to receive Alert message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates response upon receiving the Alert message. Possible values: `PD_RESPONSE_NOT_SUPPORTED`, `PD_RESPONSE_UNSPECIFIED`(default)

Result

None

Examples

```
$alert_setting = PD_Alert_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetAlertSetting( $alert_setting )
```

5.1.52 PD_WaitForAlert

Wiats for a user-defined time-out to receive Alert message. It will respond to incoming messages as part of Alert AMS.

Format

```
Call PD_WaitForAlert()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Alert message not received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message sent as response.

Examples

```
Call PD_WaitForAlert()
```

5.1.53 PD_GetBatteryCap

Starts the GetBatteryCap AMS.

Format

```
Call PD_GetBatteryCap()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message received.

Examples

```
Call PD_GetBatteryCap()
```

5.1.54 PD_SetGetBatteryCapDataBlock

Sets the GetBatteryCap Data Block in PD Exerciser. It must be called before `PD_GetBatteryCap` command to take effect.

Format

```
Call PD_SetGetBatteryCapDataBlock( PD_GetBatteryCapDataBlock $get_battery_cap_db
)
```

Parameters

`$get_battery_cap_db`

Parameter type is PD_GetBatteryCapDataBlock. Refer to `PD_GetBatteryCapMsg` for available fields of this type.

Result

None

Examples

```
$get_battery_cap_db = PD_GetBatteryCapDataBlock
Call PD_SetGetBatteryCapDataBlock( $get_battery_cap_db )
```

5.1.55 PD_SetGetBatteryCapSetting

Applies settings to GetBatteryCap related commands in PD Exerciser. It must be called before `PD_WaitForGetBatteryCap` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetGetBatteryCapSetting( PD_GetBatteryCap_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_GetBatteryCap_Settings`. Available fields for this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(micro second) to receive GetBatteryCap message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`
ResponseType	Indicates response upon receiving the GetBatteryCap message. Possible values: `PD_RESPONSE_NOT_SUPPORTED`, `PD_RESPONSE_UNSPECIFIED`(default)

Result

None

Examples

```
$getbattcap_setting = PD_GetBatteryCap_Settings
{
   ResponseType = PD_RESPONSE_NOT_SUPPORTED
}
Call PD_SetGetBatteryCapSetting( $getbattcap_setting )
```

5.1.56 PD_WaitForGetBatteryCap

Waits for user-defined time-out to receive Get_Battery_Cap message. It will respond to incoming messages as part of GetBatteryCap AMS.

Format

```
Call PD_WaitForGetBatteryCap()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Battery_Cap not received.
PD_SUBRESULT_RESPONSE_NOT_SUPPORTED	Subresult - Not_Supported message sent as response.

Examples

```
Call PD_WaitForGetBatteryCap()
```

5.1.57 PD_SetBatteryCapDataBlock

Sets the BatteryCap Data Block in PD Exerciser. It must be called before `PD_WaitForGetBatteryCap` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetBatteryCapDataBlock( PD_BatteryCapDataBlock $battery_cap_db )
```

Parameters

`$battery_cap_db`

Parameter type is PD_BatteryCapDataBlock. Refer to `PD_BatteryCapabilitiesMsg` for available fields of this type.

Result

None

Examples

```
$battery_cap_db = PD_BatteryCapDataBlock
Call PD_SetBatteryCapDataBlock( $battery_cap_db )
```

5.1.58 PD_ResetBatteryCapDataBlock

Clears the BatteryCap Data Block in PD Exerciser. Should be called before calling `PD_SetBatteryCapDataBlock` command.

Format

```
Call PD_ResetBatteryCapDataBlock()
```

Parameters

None

Result

None

Examples

```
Call PD_ResetBatteryCapDataBlock()
```

5.1.59 PD_GetManufacturerInfo

Starts GetManufacturerInfo AMS.

Format

```
Call PD_GetManufacturerInfo( OrderedSetType )
```

Parameters

`OrderedSetType`

possible values:
```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received.

Examples

```
Call PD_GetManufacturerInfo( PD_ORDERED_SET_TYPE_SOP )
```

5.1.60 PD_SetGetManufacturerInfoDataBlock

Sets GetManufacturerInfo Data Block in PD Exerciser. It must be called before `PD_GetManufacturerInfo` command to take effect.

Format

```
Call PD_SetGetManufacturerInfoDataBlock( PD_GetManufacturerInfoDataBlock
$get_manuf_info_db )
```

Parameters

`$get_manuf_info_db`

Parameter type is `PD_GetManufacturerInfoDataBlock`. Refer to `PD_GetManufacturerInfoMsg` for available fields of this type.

Result

None

Examples

```
$get_manuf_info_db = PD_GetManufacturerInfoDataBlock
Call PD_SetGetManufacturerInfoDataBlock( $get_manuf_info_db )
```

5.1.61 PD_SetGetManufacturerInfoSetting

Applies setting to GetManufacturerInfo related commands in PD Exerciser. It must be called before `PD_WaitForGetManufacturerInfo` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetGetManufacturerInfoSetting( PD_GetManufacturerInfo_Settings $settings
)
```

Parameters

`$settings`

Parameter type is PD_GetManufacturerInfo_Settings. Available fields of this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(in micro seconds) to receive GetManufacturerInfo message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`

Result

None

Examples

```
$getmaninfo_setting = PD_GetManufacturerInfo_Settings
{
   WaitTimeout = 50000
}
Call PD_SetGetManufacturerInfoSetting( $getmaninfo_setting )
```

5.1.62 PD_WaitForGetManufacturerInfo

Waits for user-defined time-out to receive Manufacturer_Info message. It will respond to incoming messages as part of GetManufacturerInfo AMS.

Format

```
Call PD_WaitForGetManufacturerInfo()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Manufacturer_Info message not received.

Examples

```
Call PD_WaitForGetManufacturerInfo()
```

5.1.63 PD_SetManufacturerInfoDataBlock

Sets ManufacurerInfo Data Block in PD Exerciser. It must be called before `PD_WaitForGetManufacturerInfo` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetManufacturerInfoDataBlock( PD_ManufacturerInfoDataBlock
$manufacturer_info_db )
```

Parameters

`$manufacturer_info_db`

Parameter type is `PD_ManufacturerInfoDataBlock`. Refer to `PD_ManufacturerInfoMsg` for available fields of this type.

Result

None

Examples

```
$manufacturer_info_db = PD_ManufacturerInfoDataBlock
Call PD_SetManufacturerInfoDataBlock( $manufacturer_info_db )
```

5.1.64 PD_SetSecurityRequestSetting

Applies setting to SecurityRequest related commands in PD Exerciser. It must be called before `PD_WaitForSecurityRequest` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetSecurityRequestSetting( PD_SecurityRequest_Settings $settings )
```

Parameters

`$settings`

Parameter type is `PD_SecurityRequest_Settings`. Available fields for this type are:

Field Names	Description
WaitTimeout	Wait TimeOut(in micro seconds) to receive SecurityRequest message. Default: `PD_DEFAULT_TIMEOUT_INFINIT`

Result

None

Examples

```
$secreq_settings = PD_SecurityRequest_Settings
{
   WaitTimeout = 50000
}
Call PD_SetSecurityRequestSetting( $secreq_settings )
```

5.1.65 PD_SecurityRequest

Starts the SecurityRequest AMS.

Format

```
Call PD_SecurityRequest( OrderedSetType )
```

Parameters

`OrderedSetType`

Possible values:
```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received.

Examples

```
Call PD_SecurityRequest( PD_ORDERED_SET_TYPE_SOP )
```

5.1.66 PD_SetSecurityRequestDataBlock

Sets the SecurityRequest Data Block in PD Exerciser. It must be called before `PD_SecurityRequest` command to take effect.

Format

```
Call PD_SetSecurityRequestDataBlock( PD_SecurityRequestDB $security_req_db )
```

Parameters

`$security_req_db`

Parameter type is PD_SecurityRequestDB. Refer to `PD_SecurityRequestMsg` for available types which are derived from this type.

Result

None

Examples

```
$security_req_db = PD_SRQDB_GetDigests
Call PD_SetSecurityRequestDataBlock( $security_req_db )
```

5.1.67 PD_WaitForSecurityRequest

Waits for user-defined time-out to receive Security_Request message. It will respond to incoming messages as part of SecurityRequest AMS.

Format

```
Call PD_WaitForSecurityRequest()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Security_Request message not received.

Examples

```
Call PD_WaitForSecurityRequest()
```

5.1.68 PD_SetSecurityResponseDataBlock

Sets the SecurityResponse Data Block in PD Exerciser. It must be called before `PD_WaitForSecurityRequest` or `PD_DelayAutoResponse` to take effect.

Format

```
Call PD_SetSecurityResponseDataBlock( PD_SecurityResponseDB $security_resp_db )
```

Parameters

`$security_resp_db`

Parameter type is `PD_SecurityResponseDB`. Refer to `PD_SecurityResponseMsg` for available types which are derived from this type.

Result

None

Examples

```
$security_resp_db = PD_SRPDB_Certificate
Call PD_SetSecurityResponseDataBlock( $security_resp_db )
```

5.1.69 PD_SetDiscoverIdentitySetting

Applies setting to DiscoverIdentity related commands in PD Exerciser. It must be called before `PD_DiscoverIdentity` or `PD_WaitForDiscoverIdentity` or `PD_PerformDiscoveryProcess` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetDiscoverIdentitySetting( PD_DiscoverIdentity_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_DiscoverIdentity_Settings` type. Table below shows the available fields of `PD_DiscoverIdentity_Settings` template:

Field Name	Description
DiscoverIdentityResponse	Indicates the response type. possible values are: PD_DISCOVERIDENTITY_ACK(default) PD_DISCOVERIDENTITY_BUSY PD_DISCOVERIDENTITY_NAK
WaitTimeout	Timeout(micro second) to wait for receiving Discover Identity Command Default: PD_DEFAULT_TIMEOUT_INFINIT
RetryCountOnWait	Indicates the retry count if Wait message received as response. Default: 4
RetryDelayOnWait	Indicates the retry delay time(micro second) if Wait message received as response. Default: 50000
AutoSpecRevCable	Indicates whether to detect SpecRev of messages towards the cable automatically or not. Default: PD_TRUE

Result

None

Examples

```
#Using default settings
$settings = PD_DiscoverIdentity_Settings
call PD_SetDiscoverIdentitySetting( $settings )
```

5.1.70 PD_AddDiscoverIdentityVDO

Adds DiscoverIdentity VDO in PD Exerciser. It must be called before `PD_WaitForDiscoverIdentity` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_AddDiscoverIdentityVDO( PD_DiscoverIdentity_VDO $vdo )
```

Parameters

$vdo

Parameter type is PD_DiscoverIdentity_VDO. Refer to `PD_VDM_Discover_Identity_Response` for available DiscoverID VDOs.

Result

None

Examples

```
#In this example, PD working revision is PD_SPEC_REVISION_2
#Add a ID Header VDO
$vdo = PD_VDM_Discover_Identity_ID_Header_VDO
{
   IDHeaderVDO_USBVendorID = 0xFF01
   IDHeaderVDO_ModalOperationSupported = 1
   IDHeaderVDO_ProductType = PD_VDM_ID_HEADER_VDO_PRODUCT_TYPE_PERIPHERAL
   IDHeaderVDO_DataCapableAsUSBDevice = 1
}
call PD_AddDiscoverIdentityVDO( $vdo )
```

5.1.71 PD_ResetDiscoverIdentityVDO

Clears DiscoverIdentity VDOs in PD Exerciser. Should be called before adding one or more DicoverIdentity VDO.

Format

```
Call PD_ResetDiscoverIdentityVDO()
```

Parameters

None

Result

None

Examples

```
call PD_ResetDiscoverIdentityVDO()
```

5.1.72 PD_DiscoverIdentity

Starts DiscoverIdentity AMS.

Format

```
Call PD_DiscoverIdentity( OrderedSetType )
```

Parameters

`OrderedSetType`

possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY received as response

Examples

```
call PD_DiscoverIdentity(PD_ORDERED_SET_TYPE_SOP)
```

5.1.73 PD_WaitForDiscoverIdentity

Waits for user-defined time-out to receive DISCOVERIDENTITY command. It will respond to incoming messages as part of DiscoverIdentity AMS.

Format

```
Call PD_WaitForDiscoverIdentity()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DISCOVERIDENTITY command not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY has been sent as response

Examples

```
call PD_WaitForDiscoverIdentity()
```

5.1.74 PD_SetDiscoverSVIDSetting

Applies settings to DiscoverSVID related commands in PD Exerciser. It must be called before `PD_DiscoverSvids` or `PD_WaitForDiscoverSvids` or `PD_PerformDiscoveryProcess` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetDiscoverSVIDSetting( PD_DiscoverSvids_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_DiscoverSvids_Settings` type. Table below shows the available fields of `PD_DiscoverSvids_Settings` template:

Field Name	Description

DiscoverSvidsResponse	Indicates the response type. possible values are:
	`PD_DISCOVERSVIDS_ACK`(default)
	`PD_DISCOVERSVIDS_BUSY`
	`PD_DISCOVERSVIDS_NAK`

WaitTimeout	Timeout(micro second) to wait for receiving Discover SVID Command Default: `PD_DEFAULT_TIMEOUT_INFINIT`
RetryCountOnWait	Indicates the retry count if Wait message received as response. Default: 4
RetryDelayOnWait	Indicates the retry delay time(micro second) if Wait message received as response. Default: 50000

Result

None

Examples

```
#Using default settings
$settings = PD_DiscoverSvids_Settings
call PD_SetDiscoverSVIDSetting( $settings )
```

5.1.75 PD_AddSvid

Adds SVIDs to PD Exerciser. It must be called before `PD_DiscoverSvids` or `PD_WaitForDiscoverSvids` or `PD_PerformDiscoveryProcess` or `PD_DelayAutoResponse` commands to take effect.

Note - Up to 11 SVIDs can be added using this command.

Format

```
Call PD_AddSvid(value)
```

Parameters

`value`

SVID value to add

Result

None

Examples

```
call PD_AddSvid(0xFF01)
```

5.1.76 PD_ResetSvids

Clears SVIDs which is added to PD Exerciser. Should be called before adding one or more SVID.

Format

```
Call PD_ResetSvids()
```

Parameters

None

Result

None

Examples

```
call PD_ResetSvids()
```

5.1.77 PD_DiscoverSvids

Starts DiscoverSVID AMS.

Note - PD Exerciser supports only one(first) DiscoverSVIDs Ack message.

Format

```
Call PD_DiscoverSvids(OrderedSetType)
```

Parameters

`OrderedSetType`

possible values:
```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY received as response

Examples

```
call PD_DiscoverSvids(PD_ORDERED_SET_TYPE_SOP)
```

5.1.78 PD_WaitForDiscoverSvids

Waits for user-defined time-out to receive DISCOVERSVID command. It will respond to incoming messages as part of DiscoverSVIDs AMS.

Format

```
Call PD_WaitForDiscoverSvids()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DISCOVERSVIDS message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY has been sent as response

Examples

```
call PD_WaitForDiscoverSvids()
```

5.1.79 PD_SetDiscoverModeSetting

Applies settings to DiscoverModes related commands in PD Exerciser. It must be called before `PD_DiscoverModes` or `PD_WaitForDiscoverModes` or `PD_PerformDiscoveryProcess` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetDiscoverModeSetting( PD_DiscoverModes_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_DiscoverModes_Settings` type. Table below describes the `PD_DiscoverModes_Settings` template:

Field Name	Description
DiscoverModesResponse	Response type. Possible values are PD_DISCOVERMODES_ACK(default) PD_DISCOVERMODES_BUSY PD_DISCOVERMODES_NAK
WaitTimeout	Timeout(micro second) to wait for receiving Discover Modes command Default: PD_DEFAULT_TIMEOUT_INFINIT
RetryCountOnWait	Indicates the retry count if Wait message received as response. Default: 4
RetryDelayOnWait	Indicates the retry delay time(micro second) if Wait message received as response. Default: 50000

Result

None

Examples

```
#Using default settings
$settings = PD_DiscoverModes_Settings
call PD_SetDiscoverModeSetting( $settings )
```

5.1.80 PD_AddMode

Adds Mode in PD Exerciser. It must be called before `PD_DiscoverModes` or `PD_WaitForDiscoverModes` or `PD_PerformDiscoveryProcess` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_AddMode(Mode)
```

Parameters

`Mode`

Mode to add

Result

None

Examples

```
call PD_AddMode(0x00000001)
```

5.1.81 PD_AddModeVDO

Adds Mode with VDO in PD Exerciser. It must be called before `PD_DiscoverModes` or `PD_WaitForDiscoverModes` or `PD_PerformDiscoveryProcess` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_AddModeVDO(PD_Generic_VDO $ModeVdo)
```

Parameters

`$ModeVdo`

Parameter type is PD_Generic_VDO. Refer to `PD_VDM_Discover_Modes_Response` for available VDOs which can be use as this parameter.

Result

None

Examples

```
local $vdo_1 = PD_VDO
{
   Data = 0x01
}
call PD_AddModeVDO($vdo_1)
```

5.1.82 PD_ResetModes

Clears Modes which are added to PD Exerciser. Could be called before adding one or more Mode.

Format

```
Call PD_ResetModes()
```

Parameters

None

Result

None

Examples

```
call PD_ResetModes()
```

5.1.83 PD_DiscoverModes

Starts DicoverModes AMS.

Format

```
Call PD_DiscoverModes(OrderedSetType, selectedSvid)
```

Parameters

`OrderedSetType`

possible values:
```
   PD_ORDERED_SET_TYPE_SOP
```

```
    PD_ORDERED_SET_TYPE_SOP_PRIME
    PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`selectedSvid`

Indicates the SVID value

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY received as response

Examples

```
call PD_DiscoverModes(PD_ORDERED_SET_TYPE_SOP,0xFF00)
```

5.1.84 PD_WaitForDiscoverModes

Waits for user-defined time-out to receive DISCOVERMODE command. It will respond to incoming messages as part of DiscoverModes AMS.

Format

```
Call PD_WaitForDiscoverModes()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DISCOVERMODES message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY has been sent as response

Examples

```
call PD_WaitForDiscoverModes()
```

5.1.85 PD_SetEnterModeSetting

Applies settings to EnterMode related commands in PD Exerciser. It must be called before `PD_WaitForEnterMode` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetEnterModeSetting( PD_EnterMode_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_EnterMode_Settings` type. Table below describes the `PD_EnterMode_Settings` template:

Field Name	Description
EnterModeResponse	Response type. Possible values : `PD_ENTERMODE_ACK`(default) `PD_ENTERMODE_NAK`
WaitTimeout	Timeout(micro second) to wait for receiving Enter Mode command Default: `PD_DEFAULT_TIMEOUT_INFINIT`

Result

None

Examples

```
#Using default setting
$settings = PD_EnterMode_Settings
call PD_SetEnterModeSetting( $settings )
```

5.1.86 PD_EnterMode

Starts EnterMode AMS.

Format

```
Call PD_EnterMode(OrderedSetType, selectedSvid, modeIndex)
```

Parameters

`OrderedSetType`

possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`selectedSvid`

Indicates the SVID

`modeIndex`

Indicates the mode index for the specified SVID

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded

PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response

Examples

```
call PD_EnterMode(PD_ORDERED_SET_TYPE_SOP,0xFF00, 1)
```

5.1.87 PD_EnterModeVdo

Starts EnterMode AMS.

Format

```
Call PD_EnterModeVdo( OrderedSetType, selectedSvid, modeId, PD_Generic_VDO $Vdo
)
```

Parameters

`OrderedSetType`

Indicates the ordered set type. Possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`selectedSvid`

Indicates the SVID

`modeId`

Indicates the mode index related to the specified SVID

`$Vdo`

Vendor defined data object. Should be from `PD_VDO`(Inherited from `PD_Generic_VDO`) type.

Field Name	Description
Data	VDO data

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response

Examples

```
$vdo = PD_VDO
{
    Data = 0x00
}
call PD_EnterModeVdo(PD_ORDERED_SET_TYPE_SOP, 0xFF01, 1, $vdo)
```

5.1.88 PD_WaitForEnterMode

Waits for user-defined time-out to receive ENTERMODE command. It will respond to incoming messages as part of EnterMode AMS.

Format

```
Call PD_WaitForEnterMode()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - ENERMODE message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response

Examples

```
call PD_WaitForEnterMode()
```

5.1.89 PD_SetExitModeSetting

Applies settings to ExitMode related commands in PD Exerciser. It must be called before `PD_WaitForExitMode` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetExitModeSetting( PD_ExitMode_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_ExitMode_Settings` type. Table below describes the `PD_ExitMode_Settings` template:

Field Name	Description
ExitModeResponse	Indicates the response type. Possible values : `PD_EXITMODE_ACK`(default) `PD_EXITMODE_NAK`
WaitTimeout	Timeout(micro second) to wait for receiving the Exit Mode command Default: `PD_DEFAULT_TIMEOUT_INFINIT`

Result

None

Examples

```
#Using default settings
$settings = PD_ExitMode_Settings
call PD_SetExitModeSetting( $settings )
```

5.1.90 PD_ExitMode

Starts ExitMode AMS.

Format

```
Call PD_ExitMode(OrderedSetType, selectedSvid, modeIndex)
```

Parameters

`OrderedSetType`

possible values:

```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`selectedSvid`

Indicates the SVID

`modeIndex`

Indicates the mode index related to the specified SVID

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response

Examples

```
call PD_ExitMode(PD_ORDERED_SET_TYPE_SOP, 0xFF00, 1)
```

5.1.91 PD_WaitForExitMode

Waits for user-defined time-out to receive EXITMODE command. It will respond to incoming messages as part of ExitMode AMS.

Format

```
Call PD_WaitForExitMode()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for PD_SendPacket and PD_ReceivePacket are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - EXITMODE message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response

Examples

```
call PD_WaitForExitMode()
```

5.1.92 PD_Attention

Starts Attention AMS.

Format

```
Call PD_Attention( OrderedSetType, selectedSvid, modeIndex )
```

Parameters

`OrderedSetType`

Indicates the ordered set type. possible values:
```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`selectedSvid`

Indicates the SVID

`modeIndex`

Indicates the mode index related to the specified SVID

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for PD_SendPacket are valid also (depends on the error type which has been occurred during sending data).

Examples

```
call PD_Attention(PD_ORDERED_SET_TYPE_SOP, 0xFF01, 1 )
```

5.1.93 PD_AttentionVdo

Starts Attention AMS.

Format

```
Call PD_AttentionVdo( OrderedSetType, selectedSvid, modeIndex, PD_Generic_VDO
$Vdo )
```

Parameters

`rderedSetType`

Indicates the ordered set type. Possible values:
```
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME
```

`selectedSvid`

Indicates the SVID

`modeId`

Indicates the mode index related to the specified SVID

`$vdo`

Vendor defined data object. Should be from `PD_VDO`(Inherited from `PD_Generic_VDO`) type.

Field Name	Description
Data	VDO data

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` are valid also (depends on the error type which has been occurred during sending data).

Examples

```
$vdo = PD_VDO
{
   Data = 0x00
}
call PD_AttentionVdo(PD_ORDERED_SET_TYPE_SOP, 0xFF01, 1, $vdo)
```

5.1.94 PD_SetDiscoveryProcessSetting

Applies settings to DiscoveryProcess command. It must be called before `PD_PerformDiscoveryProcess` to take effect.

Format

```
Call PD_SetDiscoveryProcessSetting(PD_DiscoveryProcess_Settings $settings)
```

Parameters

`$settings`

Should be from `PD_DiscoveryProcess_Settings` type. Table below describes the `PD_DiscoveryProcess_Settings` template:

Field Name	Description
Discover_SOP_PP_During_SOP_P	Indicates whether perform SOP Double Prime discovery during SOP Prime discovery process or not. Possible Values: `PD_TRUE` `PD_FALSE`(Default)
SkipEnterMode	Indicates whether to skip the EnterMode phase or not. Default: `PD_FALSE`

Result

None

Examples

```
#Using default settings
$settings = PD_DiscoveryProcess_Settings
call PD_SetDiscoveryProcessSetting( $settings )
```

5.1.95 PD_PerformDiscoveryProcess

Performs full discovery process.

Note - PD Exerciser supports only one(first) DiscoverSVIDs Ack message (up to 12 SVIDs).

Format

```
Call PD_PerformDiscoveryProcess( OrderedSetType )
```

Parameters

`OrderedSetType`

Indicates the ordered set type. Possible values:
PD_ORDERED_SET_TYPE_SOP
PD_ORDERED_SET_TYPE_SOP_PRIME
PD_ORDERED_SET_TYPE_SOP_DOUBLE_PRIME

Result

None

Examples

```
call PD_PerformDiscoveryProcess(PD_ORDERED_SET_TYPE_SOP)
```

5.1.96 PD_SetDisplayPortSetting

Applies settings to DisplayPort related commands in PD Exerciser. It must be called before `PD_DisplayPort_UpdateStatus` or `PD_DisplayPort_Configure` or `PD_WaitForDisplayPortStatus` or `PD_WaitForDisplayPortConfigure` or `PD_DelayAutoResponse` commands to take effect.

Format

```
Call PD_SetDisplayPortSetting( PD_DisplayPort_Settings $settings )
```

Parameters

`$settings`

Should be from `PD_DisplayPort_Settings` type. Table below describes the `PD_DisplayPort_Settings` template:

Field Name	Description
ConfigureResponse	Indicates the response for incoming Display Port Configure command. Possible values: PD_DISPLAYPORT_ACK(Default) PD_DISPLAYPORT_NAK
DisplayPortModeIndex	Mode index related to the Display Port SVID. (Default: 0x01).
StatusVdo	Indicates the Display Port Status Vendor Defined Data Object which can be used in Display Port Update Status initiator or responder messages.
ConfigureVdo	Indicates the Display Port Configure Vendor Defined Data

	Object which can be used in Display Port Configure initiator message.
WaitTimeout	Timeout(micro second) to wait for receiving Display Port Update Status or Configure command. Default: `PD_DEFAULT_TIMEOUT_INFINIT`

Result

None

Examples

```
#Using default settings
##################################
$settings = PD_DisplayPort_Settings
call PD_SetDisplayPortSetting($settings)

#Set the StatusVdo
##################################
$update_status = PD_VDM_DisplayPort_Status_VDO
{
    DFPD_UFPD_Connected     = PD_DISPLAYPORT_DFPD_CONNECTED
    PowerLow                = 0x00
    AdaptorEnabled          = 0x01
    MultiFunctionPreferred  = 0x01
    UsbConfigurationRequest = 0x00
    ExitDisplayModeRequest  = 0x00
    HPD_State               = 0x00
    IRQ_HPD                 = 0x00
    Reserved_DPS_1          = 0x00
}
$settings
{
    StatusVdo = $update_status
}
Call PD_SetDisplayPortSetting($settings)

#Set the ConfigureVdo to default
##################################
$config = PD_VDM_DisplayPort_Configure_VDO
$settings
{
    ConfigureVdo = $config
}
Call PD_SetDisplayPortSetting($settings)
```

5.1.97 PD_DisplayPort_UpdateStatus

Starts DisplayPortUpdateStatus(Structured VDM) AMS.

Format

```
Call PD_DisplayPort_UpdateStatus()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received

Examples

```
call PD_DisplayPort_UpdateStatus()
```

5.1.98 PD_DisplayPort_Configure

Starts DisplayPortConfigure(Structured VDM) AMS.

Format

```
Call PD_DisplayPort_Configure()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_RESPONSE_TIMEOUT	Subresult - No response received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK received as response

Examples

```
call PD_DisplayPort_Configure()
```

5.1.99 PD_WaitForDisplayPortStatus

Waits for user-defined time-out to receive DisplayPort STATUS command. It will respond to incoming messages as part of DisplayPortStatus(Structured VDM) AMS.

Format

```
Call PD_WaitForDisplayPortStatus()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - UPDATE_STATUS message not received

Examples

```
call PD_WaitForDisplayPortStatus()
```

5.1.100 PD_WaitForDisplayPortConfigure

Waits for user-defined time-out to receive DisplayPort CONFIGURE command. It will respond to incoming messages as part of DisplayPortConfigure(Structured VDM) AMS.

Format

```
Call PD_WaitForDisplayPortConfigure()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Configure message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response

Examples

```
call PD_WaitForDisplayPortConfigure()
```

5.1.101 PD_SetDiscoverIdentitySetting_Cable

Applies setting to DiscoverIdentity_Cable related commands in PD Exerciser. It must be called before `PD_WaitForDiscoverIdentity_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetDiscoverIdentitySetting_Cable( PD_DiscoverIdentity_Settings $settings
)
```

Parameters

`$settings`

Refer to `PD_SetDiscoverIdentitySetting` for more details. Only `DiscoverIdentityResponse` and `WaitTimeout` settings applied.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
#Using default settings
$settings = PD_DiscoverIdentity_Settings
call PD_SetDiscoverIdentitySetting_Cable( $settings )
```

5.1.102 PD_WaitForDiscoverIdentity_Cable

Waits for user-defined time-out to receive DISCOVERIDENTITY command. It will respond to incoming messages as part of DiscoverIdentity AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForDiscoverIdentity_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DISCOVER_IDENTITY message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY has been sent as response

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_WaitForDiscoverIdentity_Cable()
```

5.1.103 PD_AddDiscoverIdentityVDO_Cable

Adds DiscoverIdentity VDO(for cable) in PD Exerciser. It must be called before `PD_WaitForDiscoverIdentity_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_AddDiscoverIdentityVDO_Cable( PD_DiscoverIdentity_VDO $vdo )
```

Parameters

`$vdo`

Parameter type is PD_DiscoverIdentity_VDO. Refer to `PD_VDM_Discover_Identity_Response` for available DiscoverID VDOs.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
#Add a Cable VDO
$vdo = PD_VDM_Discover_Identity_Cable_VDO
call PD_AddDiscoverIdentityVDO_Cable( $vdo )
```

5.1.104 PD_ResetDiscoverIdentityVDO_Cable

Clears DiscoverIdentity VDOs(for cable) in PD Exerciser. Should be called before adding one or more DicoverIdentity VDO.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
call PD_ResetDiscoverIdentityVDO_Cable()
```

Parameters

None

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
call PD_ResetDiscoverIdentityVDO_Cable()
```

5.1.105 PD_SetDiscoverSVIDSetting_Cable

Applies settings to DiscoverSVID_Cable related commands in PD Exerciser. It must be called before `PD_WaitForDiscoverSvids_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetDiscoverSVIDSetting_Cable( PD_DiscoverSvids_Settings $settings )
```

Parameters

`$settings`

Refer to `PD_SetDiscoverSVIDSetting` for more details. Only `DiscoverSvidsResponse` and `WaitTimeout` settings applied.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
#Using default settings
$settings = PD_DiscoverSvids_Settings
call PD_SetDiscoverSVIDSetting_Cable( $settings )
```

5.1.106 PD_WaitForDiscoverSvids_Cable

Waits for user-defined time-out to receive DISCOVERSVID command. It will respond to incoming messages as part of DiscoverSVIDs AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForDiscoverSvids_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DISCOVER_SVIDS message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY has been sent as response

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_WaitForDiscoverSvids_Cable()
```

5.1.107 PD_AddSvid_Cable

Adds SVIDs to PD Exerciser. It must be called before `PD_WaitForDiscoverSvids_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_AddSvid_Cable(value)
```

Parameters

`value`

SVID value to add

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
call PD_AddSvid_Cable(0xFF81)
```

5.1.108 PD_ResetSvids_Cable

Clears SVIDs(for cable) which is added to PD Exerciser. Should be called before adding one or more SVID.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_ResetSvids_Cable()
```

Parameters

None

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
call PD_ResetSvids_Cable()
```

5.1.109 PD_SetDiscoverModeSetting_Cable

Applies settings to DiscoverModes_Cable related commands in PD Exerciser. It must be called before `PD_WaitForDiscoverModes_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetDiscoverModeSetting_Cable( PD_DiscoverModes_Settings $settings )
```

Parameters

`$settings`

Refer to `PD_SetDiscoverModeSetting` for more details. Only `DiscoverModesResponse` and `WaitTimeout` settings applied.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
#Using default settings
$settings = PD_DiscoverModes_Settings
call PD_SetDiscoverModeSetting_Cable( $settings )
```

5.1.110 PD_WaitForDiscoverModes_Cable

Waits for user-defined time-out to receive DISCOVERMODE command. It will respond to incoming messages as part of DiscoverModes AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForDiscoverModes_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - DISCOVER_MODES message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response
PD_SUBRESULT_RESPONSE_BUSY	Subresult - BUSY has been sent as response

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_WaitForDiscoverModes_Cable()
```

5.1.111 PD_AddModeVDO_Cable

Adds Mode(for cable) with VDO in PD Exerciser. It must be called before `PD_WaitForDiscoverModes_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_AddModeVDO_Cable(PD_Vdo $ModeVdo)
```

Parameters

`$ModeVdo`

Should be from `PD_Vdo` type. Table below describes the `PD_VDO` template that can be use as ModeVdo:

Field Name	Description
Data	

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
local $vdo_1 = PD_VDO
{
   Data = 0x01
}
call PD_AddModeVDO_Cable($vdo_1)
```

5.1.112 PD_AddMode_Cable

Adds Mode in PD Exerciser. It must be called before `PD_WaitForDiscoverModes_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

`Call PD_AddMode_Cable(Mode)`

Parameters

`Mode`

Mode to add

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
call PD_AddMode_Cable(0x00000001)
```

5.1.113 PD_ResetModes_Cable

Clears Modes(for cable) which are added to PD Exerciser. Could be called before adding one or more Mode.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

`Call PD_ResetModes_Cable()`

Parameters

None

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_ResetModes_Cable()
```

5.1.114 PD_SetEnterModeSetting_Cable

Applies settings to EnterMode_Cable related commands in PD Exerciser. It must be called before `PD_WaitForEnterMode_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetEnterModeSetting_Cable( PD_EnterMode_Settings $settings )
```

Parameters

`$settings`

Refer to `PD_SetEnterModeSetting` for more details.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
#Using default setting
$settings = PD_EnterMode_Settings
call PD_SetEnterModeSetting_Cable( $settings )
```

5.1.115 PD_WaitForEnterMode_Cable

Waits for user-defined time-out to receive ENTERMODE command. It will respond to incoming messages as part of EnterMode AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForEnterMode_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - ENTER_MODE message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_WaitForEnterMode_Cable()
```

5.1.116 PD_SetExitModeSetting_Cable

Applies settings to ExitMode_Cable related commands in PD Exerciser. It must be called before `PD_WaitForExitMode_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetExitModeSetting_Cable( PD_ExitMode_Settings $settings )
```

Parameters

`$settings`

Refer to `PD_SetExitModeSetting` for more details.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
#Using default settings
$settings = PD_ExitMode_Settings
call PD_SetExitModeSetting_Cable( $settings )
```

5.1.117 PD_WaitForExitMode_Cable

Waits for user-defined time-out to receive EXITMODE command. It will respond to incoming messages as part of ExitMode AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForExitMode_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - EXIT_MODE message not received
PD_SUBRESULT_RESPONSE_NAK	Subresult - NAK has been sent as response

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_WaitForExitMode_Cable()
```

5.1.118 PD_SetManufacturerInfoDataBlock_Cable

Sets ManufacurerInfo Data Block(for cable) in PD Exerciser. It must be called before `PD_WaitForGetManufacturerInfo_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetManufacturerInfoDataBlock_Cable( PD_ManufacturerInfoDataBlock
$manufacturer_info_db )
```

Parameters

`$manufacturer_info_db`

Parameter type is `PD_ManufacturerInfoDataBlock`. Refer to `PD_ManufacturerInfoMsg` for available fields of this type.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
$manufacturer_info_db = PD_ManufacturerInfoDataBlock
Call PD_SetManufacturerInfoDataBlock_Cable( $manufacturer_info_db )
```

5.1.119 PD_SetGetManufacturerInfoSetting_Cable

Applies setting to GetManufacturerInfo_Cable related commands in PD Exerciser. It must be called before `PD_WaitForGetManufacturerInfo_Cable` or `PD_DelayAutoResponse` commands to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetGetManufacturerInfoSetting_Cable( PD_GetManufacturerInfo_Settings
$settings )
```

Parameters

`$settings`

Refer to `PD_SetGetManufacturerInfoSetting` for more details.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
$getmaninfo_setting = PD_GetManufacturerInfo_Settings
{
   WaitTimeout = 50000
}
Call PD_SetGetManufacturerInfoSetting_Cable( $getmaninfo_setting )
```

5.1.120 PD_WaitForGetManufacturerInfo_Cable

Waits for user-defined time-out to receive Manufacturer_Info message. It will respond to incoming messages as part of GetManufacturerInfo AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForGetManufacturerInfo_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Get_Manufacturer_Info message not received.

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
```

```
Call PD_WaitForGetManufacturerInfo_Cable()
```

5.1.121 PD_SetSecurityResponseDataBlock_Cable

Sets the SecurityResponse Data Block(for cable) in PD Exerciser. It must be called before `PD_WaitForSecurityRequest_Cable` or `PD_DelayAutoResponse` to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetSecurityResponseDataBlock_Cable( PD_SecurityResponseDB
$security_resp_db )
```

Parameters

`$security_resp_db`

Parameter type is `PD_SecurityResponseDB`. Refer to `PD_SecurityResponseMsg` for available types which are derived from this type.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
$security_resp_db = PD_SRPDB_Certificate
Call PD_SetSecurityResponseDataBlock_Cable( $security_resp_db )
```

5.1.122 PD_SetSecurityRequestSetting_Cable

Applies setting to SecurityRequest_Cable related commands in PD Exerciser. It must be called before `PD_WaitForSecurityRequest_Cable` or `PD_DelayAutoResponse` to take effect.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_SetSecurityRequestSetting_Cable( PD_SecurityRequest_Settings $settings )
```

Parameters

`$settings`

Refer to `PD_SetSecurityRequestSetting` for more details.

Result

None

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
$secreq_settings = PD_SecurityRequest_Settings
{
   WaitTimeout = 50000
}
Call PD_SetSecurityRequestSetting( $secreq_settings )
```

5.1.123 PD_WaitForSecurityRequest_Cable

Waits for user-defined time-out to receive Security_Request message. It will respond to incoming messages as part of SecurityRequest AMS.

Note - PD Exerciser should also act as Cable Plug to be able to process this command.

Format

```
Call PD_WaitForSecurityRequest_Cable()
```

Parameters

None

Result

User can evaluate the command results(including sub-results) using `IfMatched/ElseMatched` command.
List of result values:

Result Value	Description
PD_RESULT_OK	Command succeeded
PD_RESULT_FAILED	Command failed. In this case corresponding sub results for `PD_SendPacket` and `PD_ReceivePacket` are valid also (depends on the error type which has been occurred during sending or receiving data).
PD_SUBRESULT_MSG_NOT_RECEIVED	Subresult - Security_Request message not received.

Examples

```
PD_Set $PdGlobalSettings.EnableCableEmulator = PD_TRUE
.
.
Call PD_WaitForSecurityRequest_Cable()
```

5.2 Auto Responses Capability

To gain auto response capability, use below command. This command will respond to any incoming Power Delivery messages according to current operational settings. In addition to this command, at the start of each High-Level command Auto-Response is activated.

5.2.1 PD_DelayAutoResponse

Format

```
Call PD_DelayAutoResponse( duration_micro_Sec )
```

Parameters

`duration_micro_Sec`

Command waits for maximum specified duration and responses to received packet automatically.

Examples

```
call PD_DelayAutoResponse( 1000 )
```