MCUXpresso SDK USB Type-C PD Stack Reference Manual

NXP Semiconductors

Document Number: MCUXUSBPDAPIRM
Rev. 1
Dec 2021

Contents

Contents

Contents

Contents

Chapter 1
Overview

USB has evolved from a data interface capable of supplying limited power to a primary provider of power with a data interface. Today many devices charge or get their power from USB ports contained in laptops, cars, aircraft or even wall sockets. USB has become a ubiquitous power socket for many small devices such as cell phones, MP3 players and other hand-held devices. Users need USB to fulfill their requirements not only in terms of data but also to provide power to, or charge their devices. There are however, still many devices which either require an additional power connection to the wall, or exceed the USB rated current in order to operate. Increasingly, international regulations require better energy management due to ecological and practical concerns relating to the availability of power. Regulations limit the amount of power available from the wall which has led to a pressing need to optimize power usage. The USB Power Delivery Specification has the potential to minimize waste as it becomes a standard for charging devices.

This USB Type-C PD stack implements the Type-C spec and PD3.0 spec basic functions, such as Type-C connect/disconnect state machine, PD message function. The stack provides API interface and configuration way for user to initialize, user can configure the stack as self requirement. The architecture and components of the USB Type-C PD stack are shown as below picture:

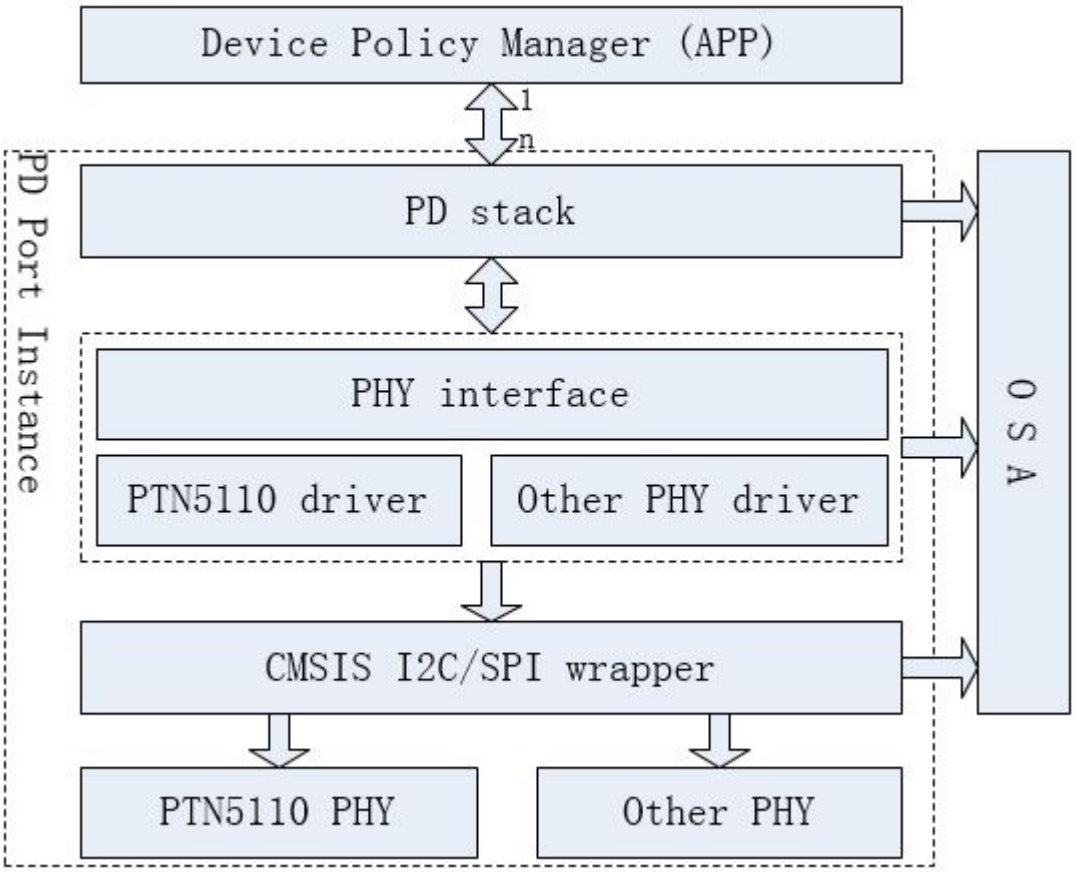

Figure 1: pd architecture

- Device Policy Manager: The device policy manager is the application's function. It implement the device policy manager function of PD spec. It manage power and negotiate the request (for example: decide accept pr_swap or reject it.)
- PD Stack: The PD stack implement the policy engine, protocol of PD spec and Type-c connect/disconnect state machine of Type-C spec.
- PHY interface and PHY driver: PHY interface is one common interface for different PHY, PD stack use this interface to operate the PHY. Different PHY implement the same interface, so one PD stack implementation can work with different PHY drivers.
- CMSIS I2C/SPI wrapper: This wrapper provide same API to PHY driver for CMSIS I2C and CMSIS SPI. PHY driver can call the same wrapper and don't need care much about the difference of I2C and SPI.
- PHY (PTN5110 or other PHY): the PHY IC.
- OSA: To support different RTOSes with the same code base, the OSA is used inside the PD stack to wrap the differences between RTOSes.
  Note that the OSA is not supported for use in the PD application. Therefore, from the PD application's view point, the OSA is invisible.
- One PD port instance contain one instance of the PD stack, one instance of the PHY driver, one

instance of the CMSIS I2C/SPI wrapper and one PHY. In one system there can be many PD port instances. The device policy manager can contain many PD port instances.

Note: The interface between the PHY Driver and the PD stack is internal and is simplified in this document.

1.1 USB PD Initialization flow

The PD stack initialization flow is as follow:

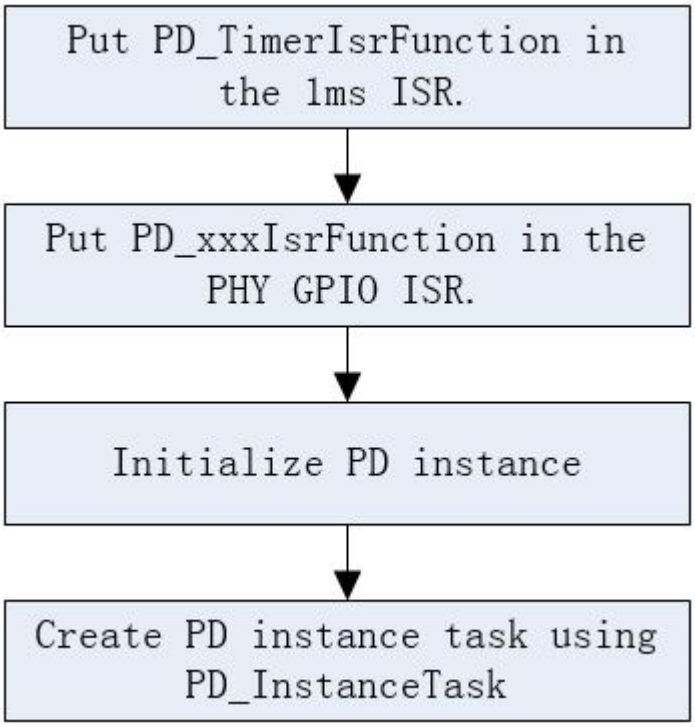

Figure 1.1.1: pd init flow

- PD_TimerIsrFunction and PD_xxxIsrFunction need be put in the corresponding ISR.
  Note: xxx means PHY, for example: PD_PTN5110IsrFunction
- Initialize PD instance: call PD_InstanceInit to initialize PD instance, The callback function, configuration need be passed to this API. This API return one handle repensent the PD instance.
- The PD instance task is important, it accomplish the PD commands or control function. PD commands and control are introduced in other sections.

1.2 USB PD connect/disconnect flow

The PD stack's connect/disconnect flow is as follow:

USB PD connect/disconnect flow

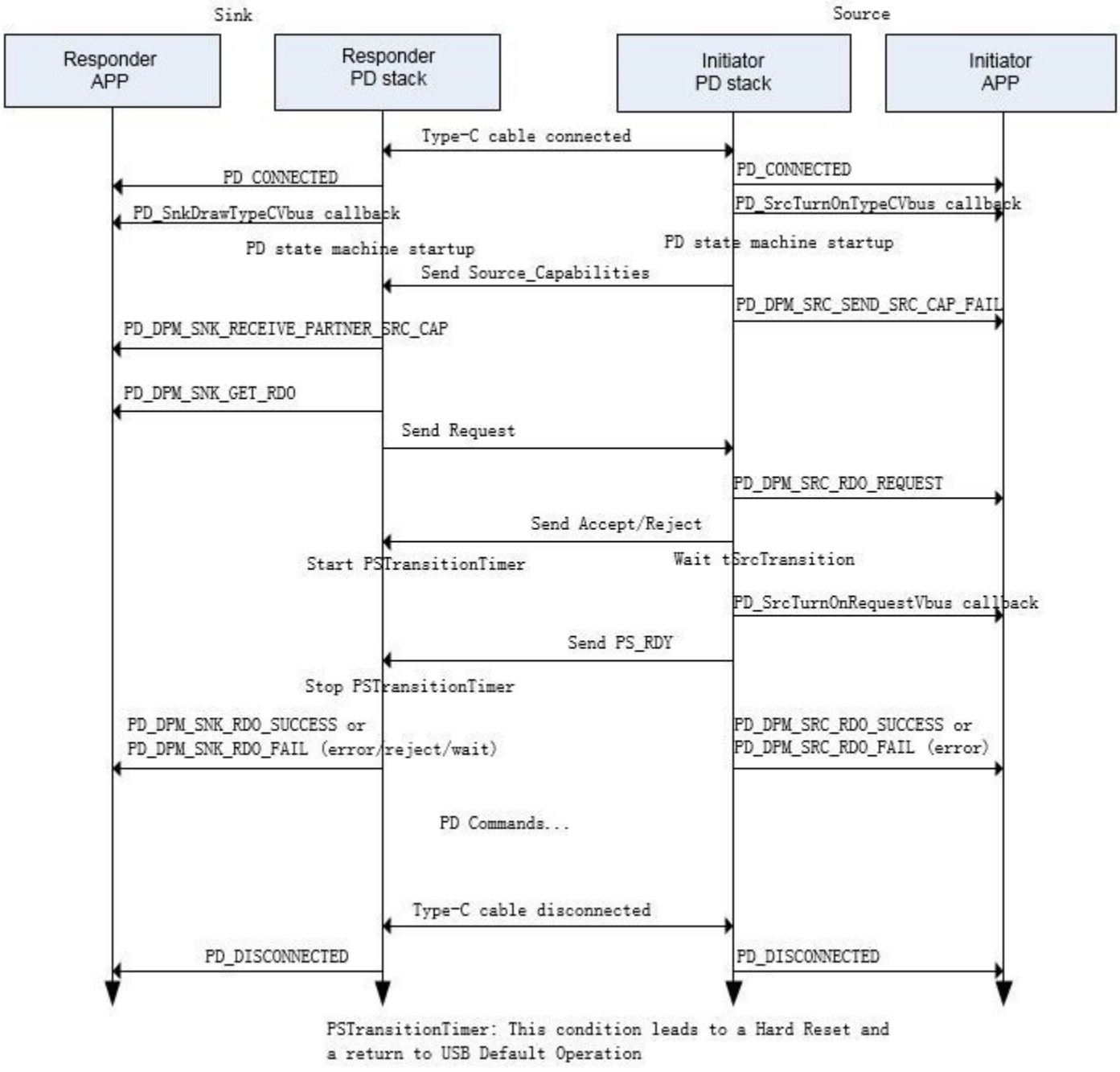

Figure 1.2.1: pd connect

- When connect, the PD_CONNECT event will callback to application. If the port is source, it should provide the vSafe5V to Vbus in this callback.
- Then the PD stack of source will send source_cap at the start-up of the state machine.
- When sink receive the source_cap, the PD_DPM_SNK_RECEIVE_PARTNER_SRC_CAP event will callback to application.

- Sink will callback PD_DPM_SNK_GET_RDO to get RDO from application, then sink start to request the power.
- Source will callback PD_DPM_SRC_RDO_REQUEST, application detemine to accept the request or reject.
- Source will call the power related callback function to provide the power.
- At last, the success or fail event will callback to application.
  When disconnect, the PD_DISCONNECTED event will callback to application.

1.3 USB PD control function

The PD_Control API provide the control function, for example: get self power role. Please reference to the PD_Control function description.

1.4 USB PD common task

One feature is provided as PD_CONFIG_COMMON_TASK in the USB PD stack to reduce the RAM size consumption.

- when PD_CONFIG_COMMON_TASK is enable, all the PD intances use one task. Application use the follow API to create task. This can reduce the RAM size reuirement.
  void PD_Task(void);
- when PD_CONFIG_COMMON_TASK is disable, every PD intances use one task. Application use the follow API to create taskes for every instance and pass the PD instance handle to the API. This is more flexible, customer can configure different priority for different PD instance.
  void PD_InstanceTask(pd_handle pdHandle);

1.5 USB PD alternate mode

The PD stack alternate mode structure is as follow (The alternate mode is one part of PD stack):

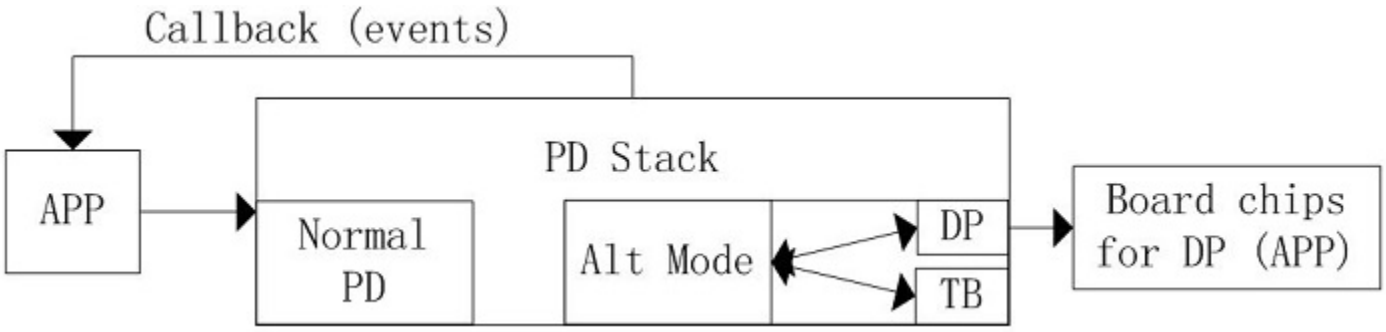

Figure 1.5.1: pd alternate mode

- Initialize PD Alternate Mode
  There is one parameter (pd_instance_config_t *config) to initialize PD instance when calling PD_InstanceInit. PD alternate mode releated parameters are configured as follow in this parameter:

PD alternate mode host configuration parameter.

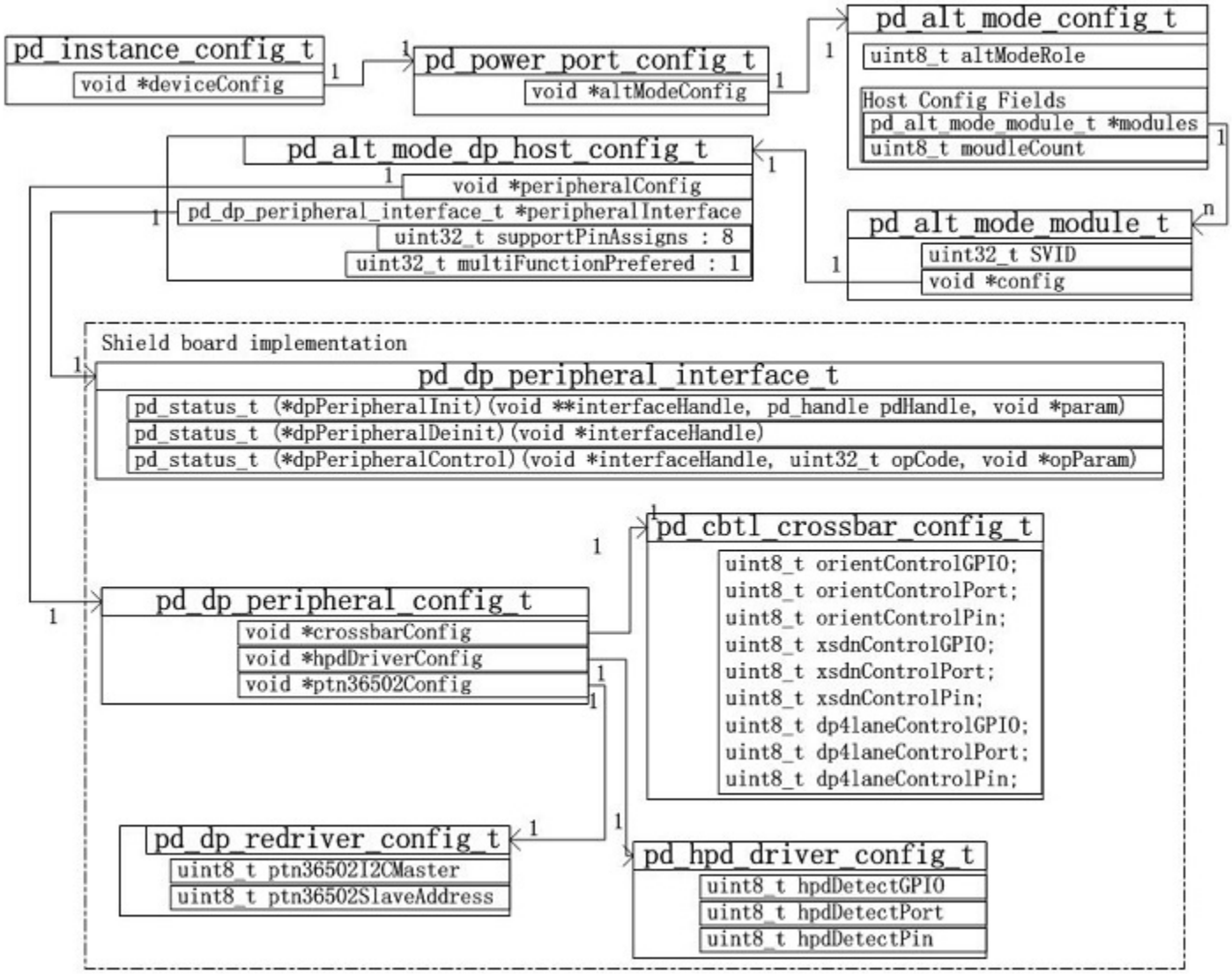

Figure 1.5.2: pd alternate mode host parameter

PD alternate mode slave configuration parameter.

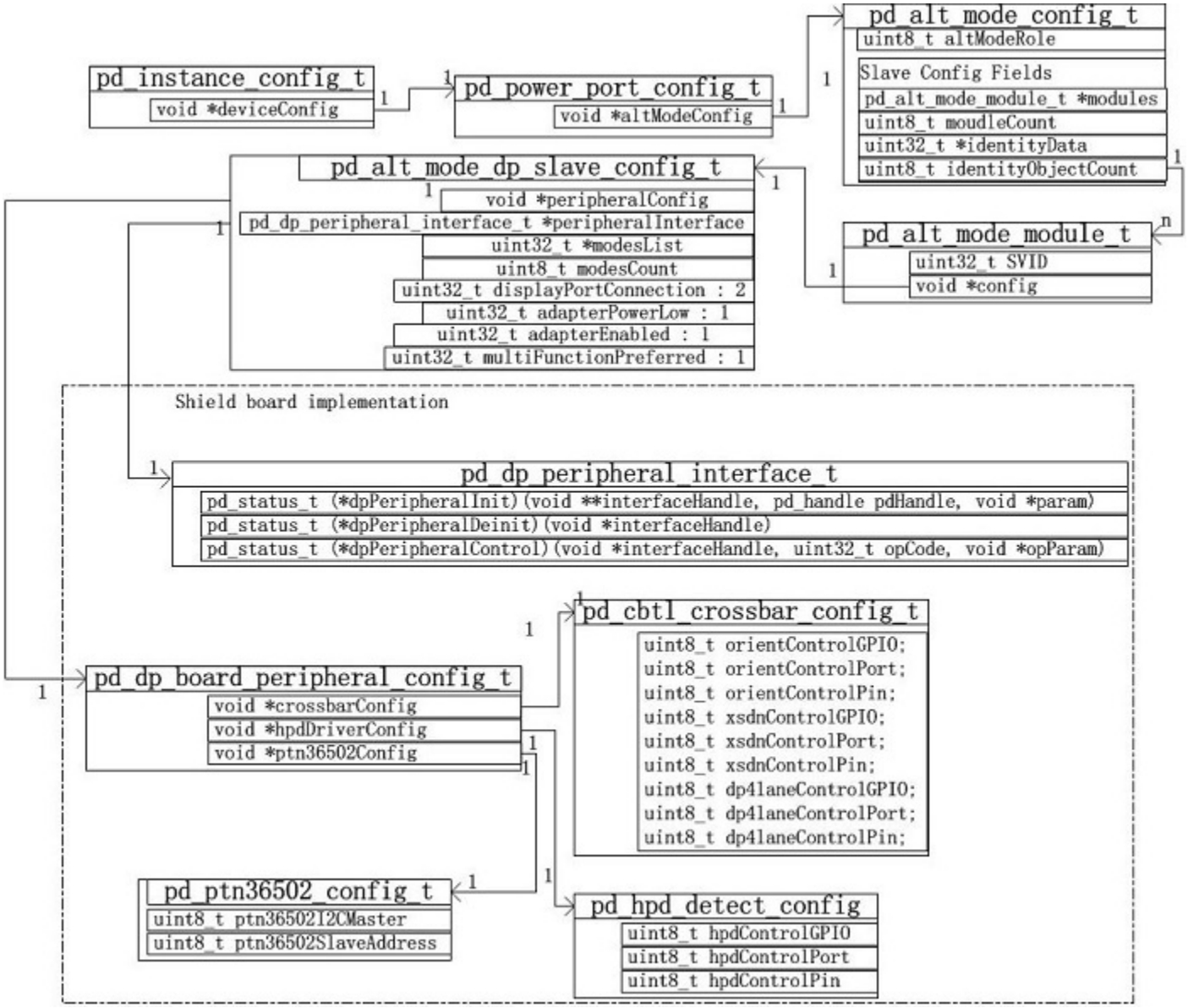

Figure 1.5.3: pd alternate mode slave parameter

- PD_CONFIG_ALT_MODE_SUPPORT configures alternate mode enable or not in usb_pd_config.h file.
- If altModeConifg is NULL, the PD instance doesn't support alternate mode function; If altModeConfig is not NULL, the PD instance supports alternate mode function.
- Alternate mode related parameters are defined by the strucutre pd_alt_mode_config_t.
- pd_alt_mode_module_t defines the supported modules (DisplayPort, ThunderBolt etc).
- pd_alt_mode_dp_host_config_t defines the DisplayPort host module parameter.
- pd_alt_mode_dp_slave_config_t defines the DisplayPort slave module parameter.
- For the shield host board, pd_dp_peripheral_config_t, pd_cbtl_crossbar_config_t, pd_-

ptn36502_config_t and pd_hpd_driver_config_t define the board peripherals' parameters to implement the DisplayPort function. pd_dp_peripheral_interface_t defines the function table that drives DP board related peripherals. pd_dp_peripheral_config_t defines the parameter for DP board related peripherals. The function table is as follow:

  - dpPeripheralInit: PD stack will call this function automatically and pass the boardChipConfig as parameter when initialize PD stack.
  - dpPeripheralDeinit: PD stack will call this function automatically when de-initialize PD stack.
  - dpPeripheralControl: implement the DP function.

- PD Alternate Mode Task
  Aternate mode application need create one task using the follow API.
  void PD_AltModeTask(void);
  - For BM, the API need be called periodically.
  - For FreeRTOS, application need create one task using the follow similar codes.
    void PD_PortAltModeTask(void *arg) { while (1) { PD_AltModeTask(); } }
- PD Alternate Mode Run State
  PD alternate mode run as follow (take DisplayPort as example):
  - After device attach. Alternate mode will start the disvover identity/SVIDs sequence if data role is DFP.
  - Search modules configured by pd_alt_mode_module_t *modules. if there is one module's SVID is matched with the result of the discover SVIDs, enter next step; if there is no module matched, the steps are done.
  - Get the supported SVID's modes by discover modes.
  - If customer enables PD_CONFIG_ALT_MODE_DP_AUTO_SELECT_MODE, the modes will be passed to application, and application will detemine which mode is supported or there is no supported mode. If there is no supported mode and pin assign, the steps are done.
  - If customer doesn't enable PD_CONFIG_ALT_MODE_DP_AUTO_SELECT_MODE (PD_CONFIG_ALT_MODE_DP_AUTO_SELECT_MODE's default value is disabled), PD stack will detemine which mode to support by the configure parameter (uint8_t supportPinAssigns) as follow:
    - Get attached dvice supported pin assigns from the mode value. If there is supported pin assigns that matches supportPinAssigns parameter, enter next step. otherwise done.
    - If multiFunctionPrefered parameter is set, PD stack will prefer to select the pin assign (kPinAssign_B and kPinAssign_D);
    - If multiFunctionPrefered parameter is not set or the previous step doesn't select one pin assign, PD stack will prefer to select the 4 lane pin assignment (kPinAssign_C, kPinAssign_E and kPinAssign_A);
    - If previous step doesn't select one pin assign, PD stack will select the first supported pin assign configured by supportPinAssigns parameter.
    - If there is no selected pin assign, the steps are done.
  - PD stack will do the enter mode, status update and DP configure to enable the attached device's DisplayPort function.
- PD Alternate Mode Events
  PD alternate mode notify application through the PD intance callback. the callback is registered to PD stack by PD_InstanceInit API. There are follow three events currently:

- PD_DPM_ALTMODE_DP_DFP_SELECT_MODE_AND_PINASSIGN: application need select the mode and pin assign in this callback, if PD_CONFIG_ALT_MODE_DP_AUTO_SELECT_MODE is enable. PD_CONFIG_ALT_MODE_DP_AUTO_SELECT_MODE is disable defaultly.
- PD_DPM_ALTMODE_DP_DFP_MODE_CONFIGURED: DisplayPort alternate mode is entered and pin assign is configured, displayport video data can be transfered.
- PD_DPM_ALTMODE_DP_DFP_MODE_UNCONFIGURED: DisplayPort's pin assign is configured as safe mode. DisplayPort video data cannot be transfered.

1.6 USB PD auto policy

The PD stack supports auto policy function. It can be configured by pd_auto_policy_t. PD_InstanceInit has one parameter called pd_instance_config_t, pd_instance_config_t has one field called deviceConfig, this field's type is pd_power_port_config_t, pd_power_port_config_t has one field called autoPolicyConfig, this field's type is pd_auto_policy_t, it is used to configure the auto policy function.

function	description
autoRequestPRSwapAsSource	It is valid when power role is source. -0: don't request power role swap automatically -1: request power role swap automatically when self is not external powered and partner is external powered, it only try one time, it will not retry if partner reply reject.
autoRequestPRSwapAsSink	It is valid when power role is sink. -0: don't request power role swap automatically -1: request power role swap automatically when self is external powered and partner is not external powered, it only try one time, it will not retry if partner reply reject.
autoAcceptPRSwapAsSource	It is valid when power role is source. -kAutoRequestProcess_NotSupport: this instance doesn't support this function -kAutoRequestProcess_Accept: accept pr_swap request if self is not external powered or partner is external powered -kAutoRequestProcess_Reject: reject pr_swap request

USB PD auto policy

function	description
autoAcceptPRSwapAsSink	It is valid when power role is sink. -kAutoRequestProcess_NotSupport: this instance doesn't support this function -kAutoRequestProcess_Accept: accept pr_swap request if self is external powered or partner is not external powered -kAutoRequestProcess_Reject: reject pr_swap request
autoRequestDRSwap	It is valid when self is not in alternating mode and self is DRD. -kPD_DataRoleUFP: auto request swap to UFP when self is DFP, it only try one time, it will not retry if partner reply reject -kPD_DataRoleDFP: auto request swap to DFP when self is UFP, it only try one time, it will not retry if partner reply reject -kPD_DataRoleNone: this instance doesn't support this function
autoAcceptDRSwapToDFP	It is valid when data role is UFP. -kAutoRequestProcess_NotSupport: this instance doesn't support this function -kAutoRequestProcess_Accept: accept dr_swap request -kAutoRequestProcess_Reject: reject dr_swap request
autoAcceptDRSwapToUFP	It is valid when data role is DFP. -kAutoRequestProcess_NotSupport: this instance doesn't support this function -kAutoRequestProcess_Accept: accept dr_swap request -kAutoRequestProcess_Reject: reject dr_swap request
autoRequestVConnSwap	It is valid when self support Vconn. -kPD_NotVconnSource: auto request swap to turn off Vconn when self is Vconn source, it only try one time, it will not retry if partner reply reject -kPD_IsVconnSource: auto request swap to turn on Vconn when self is not Vconn source, it only try one time, it will not retry if partner reply reject -kPD_VconnNone: this instance doesn't support this function

function	description
autoAcceptVconnSwapToOn	It is valid when Vconn is off. -kAutoRequestProcess_NotSupport: this instance doesn't support this function -kAutoRequestProcess_Accept: accept vconn_swap request -kAutoRequestProcess_Reject: reject vconn_swap request
autoAcceptVconnSwapToOff	It is valid when Vconn is on. -kAutoRequestProcess_NotSupport: this instance doesn't support this function -kAutoRequestProcess_Accept: accept vconn_swap request -kAutoRequestProcess_Reject: reject vconn_swap request
autoSinkNegotiation	It is valid when power role is sink. -0: this instance doesn't support this function -1: calculate the highest power request based on self's sink capabilities and partner's source capabilities

1.7 USB PD command function

The PD_Command API provide the command that are defined in the PD3.0 spec, in the spec these command are called AMS. For example: pr_swap AMS is called PD_DPM_CONTROL_PR_SWAP as PD_Command's parameter.

- PD_DPM_CONTROL_POWER_NEGOTIATION
  it is only used in source when source power change, the work flow is as follow:

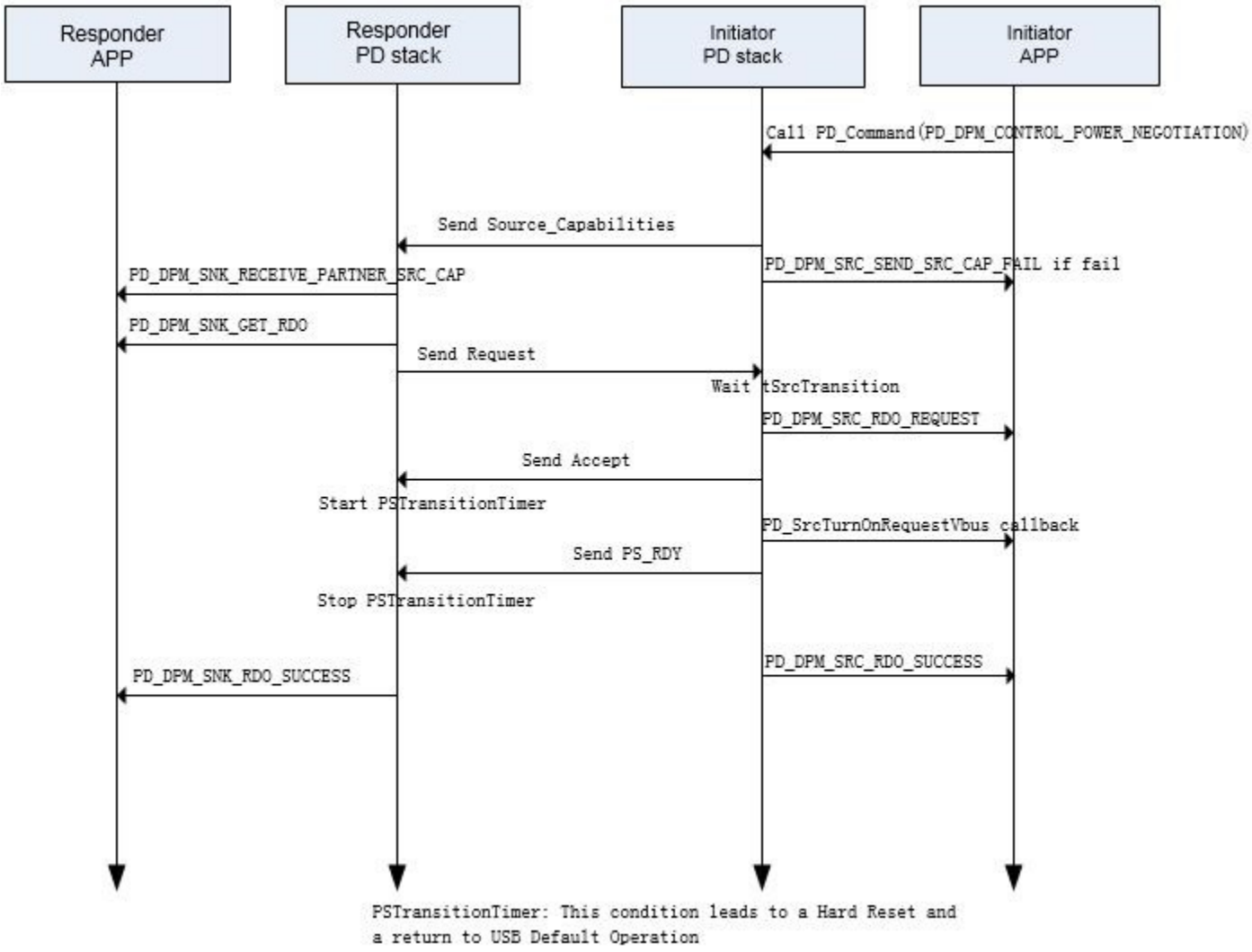

Figure 1.7.1: power negotiation 1

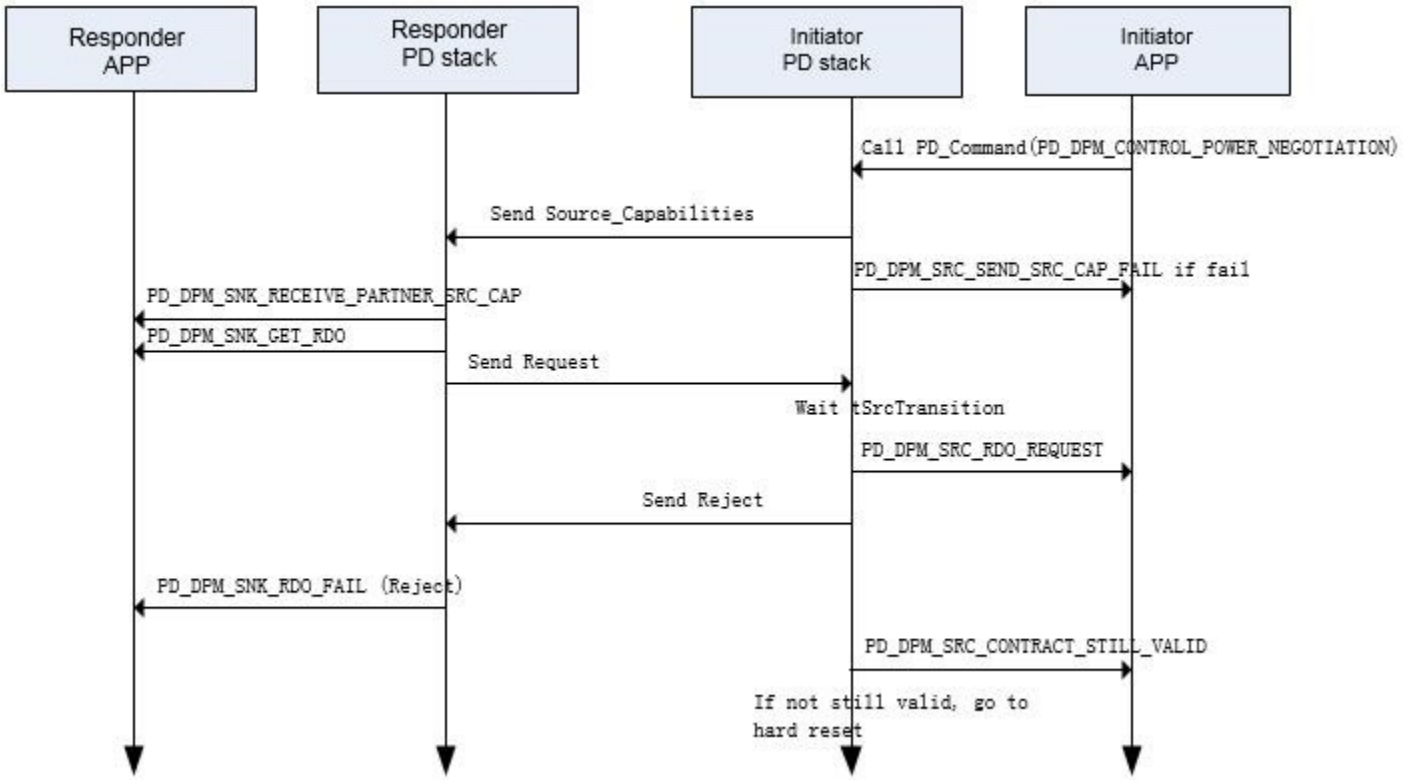

Figure 1.7.2: power negotiation 2

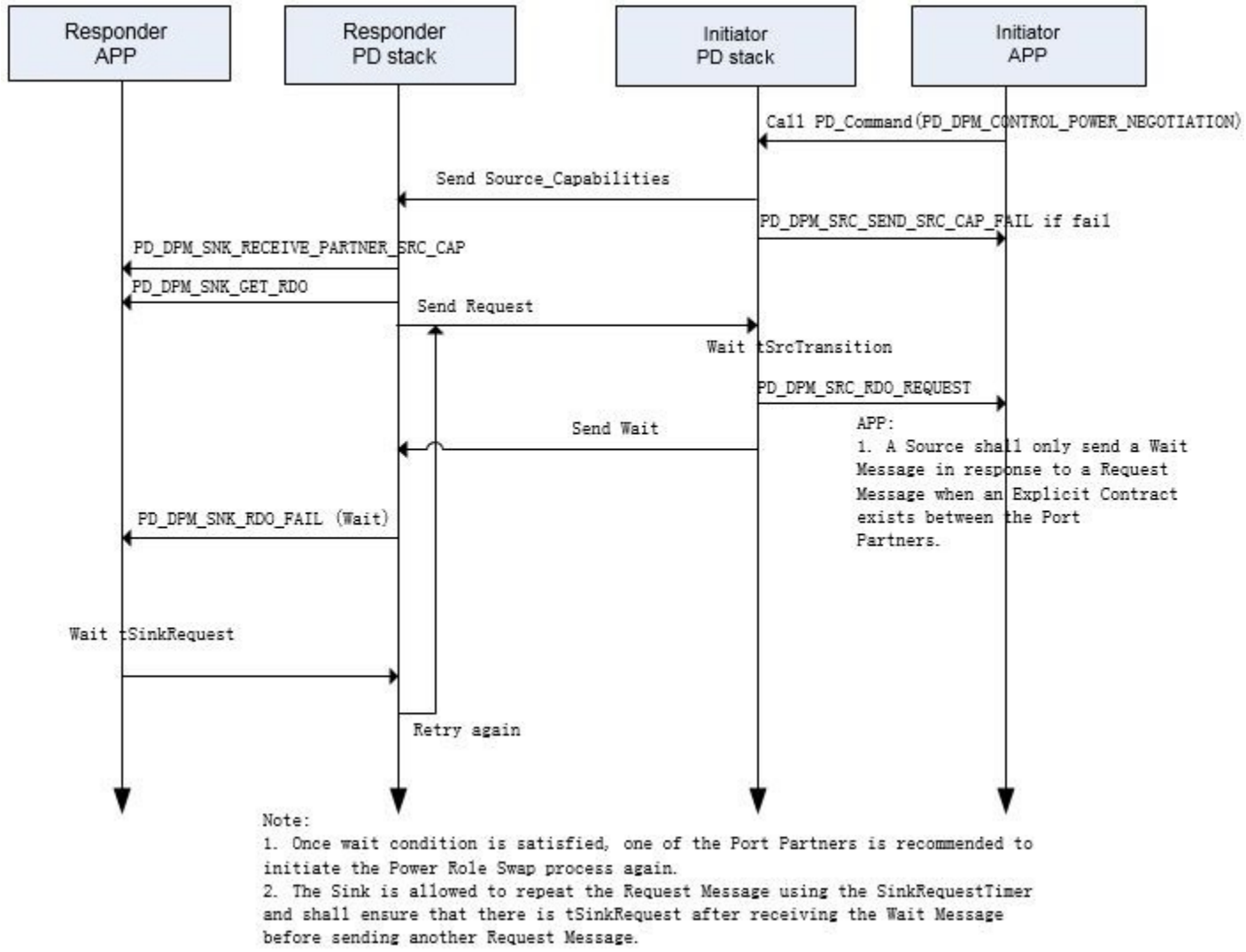

Figure 1.7.3: power negotiation 3

- PD_DPM_CONTROL_REQUEST
  it is only used in sink, the work flow is as follow:

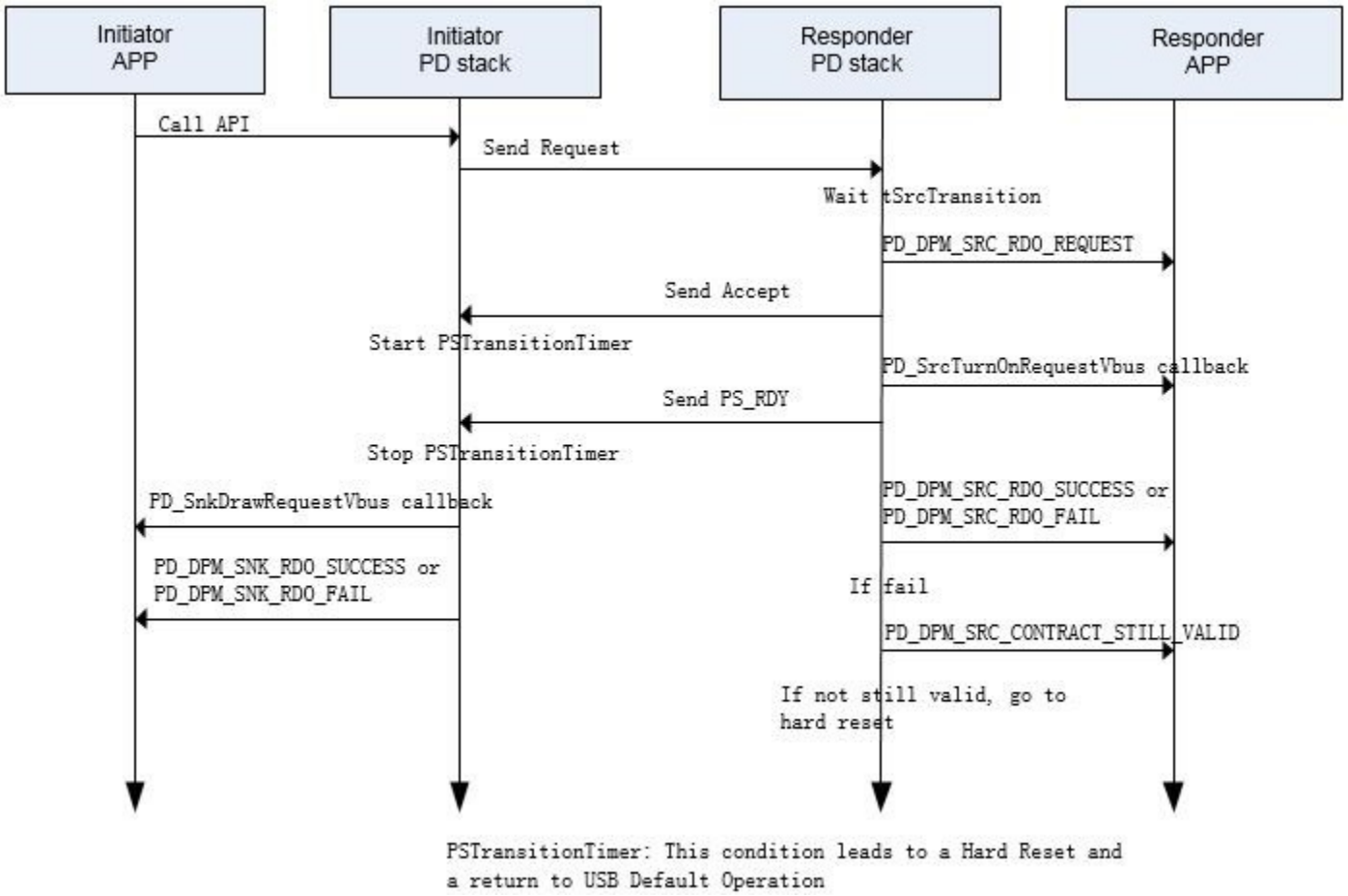

Figure 1.7.4: rdo request 1

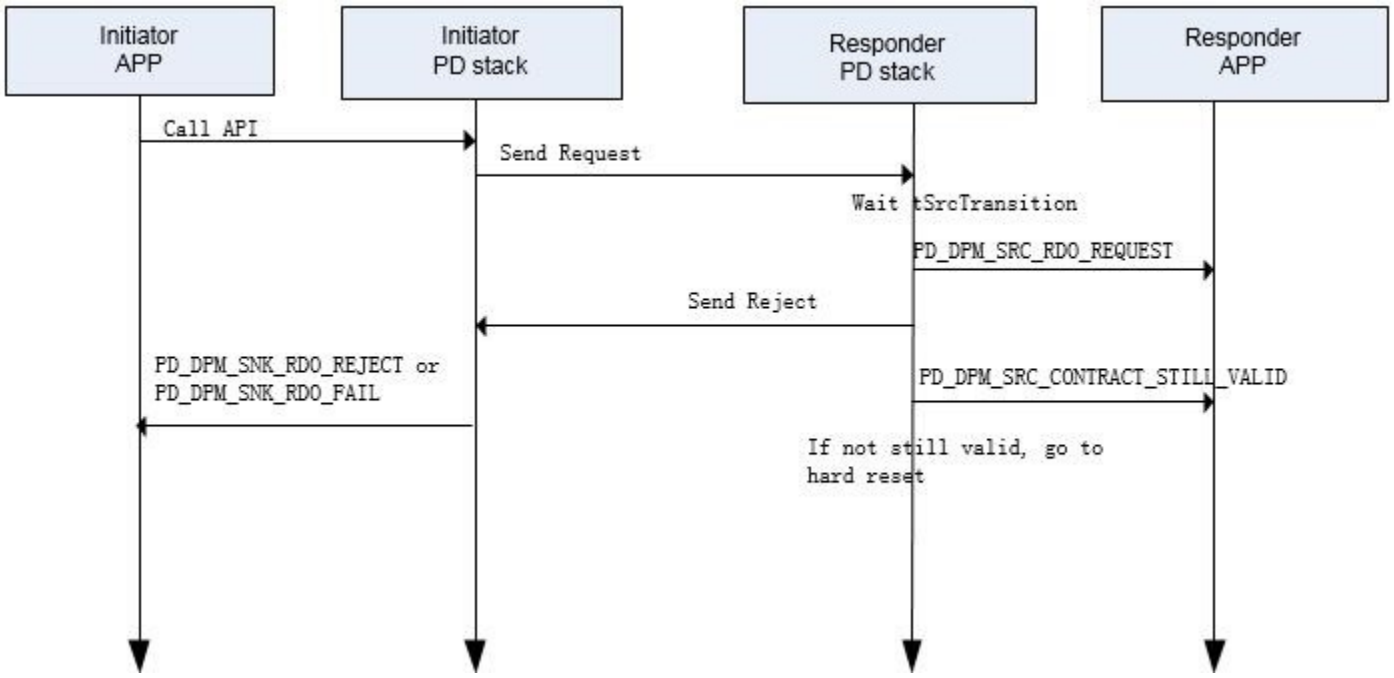

Figure 1.7.5: rdo request 2

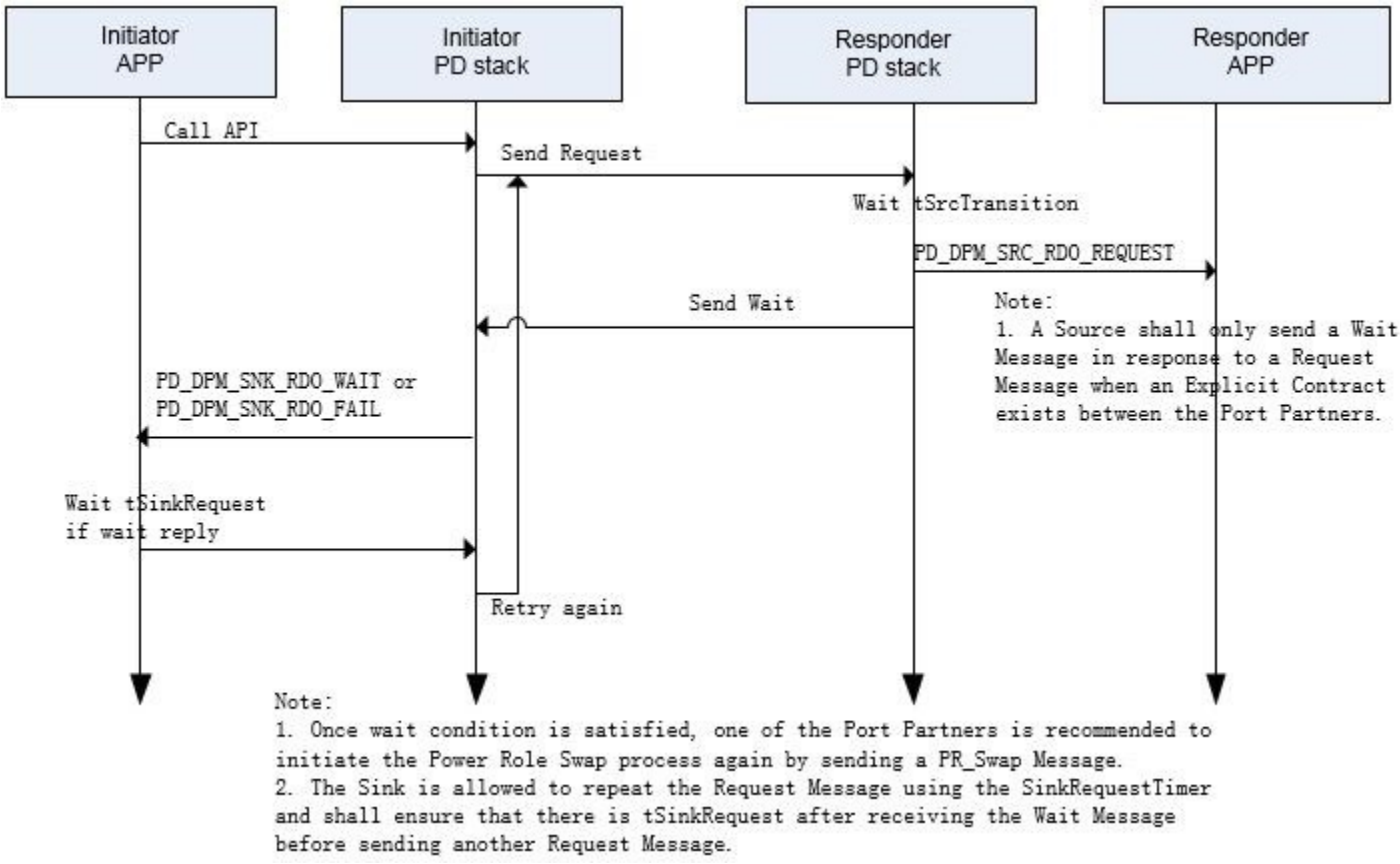

Figure 1.7.6: rdo request 3

- PD_DPM_CONTROL_GOTO_MIN
  goto min request, it is only used in source:

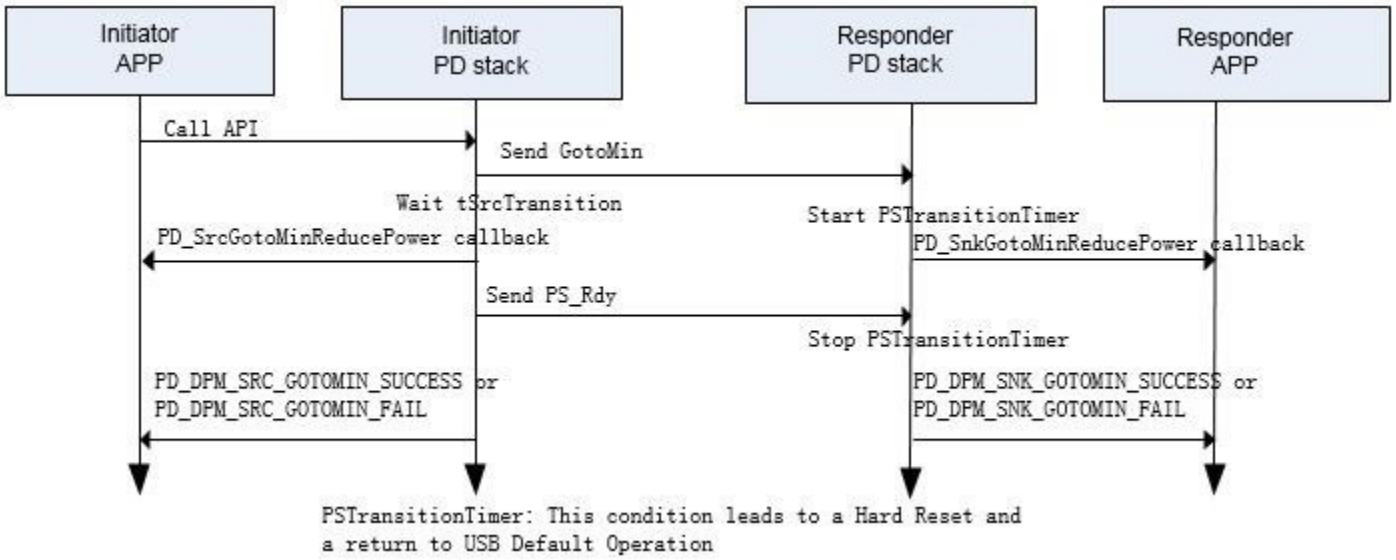

Figure 1.7.7: goto min

- PD_DPM_CONTROL_GET_PARTNER_SOURCE_CAPABILITIES
  get partner source capabilities

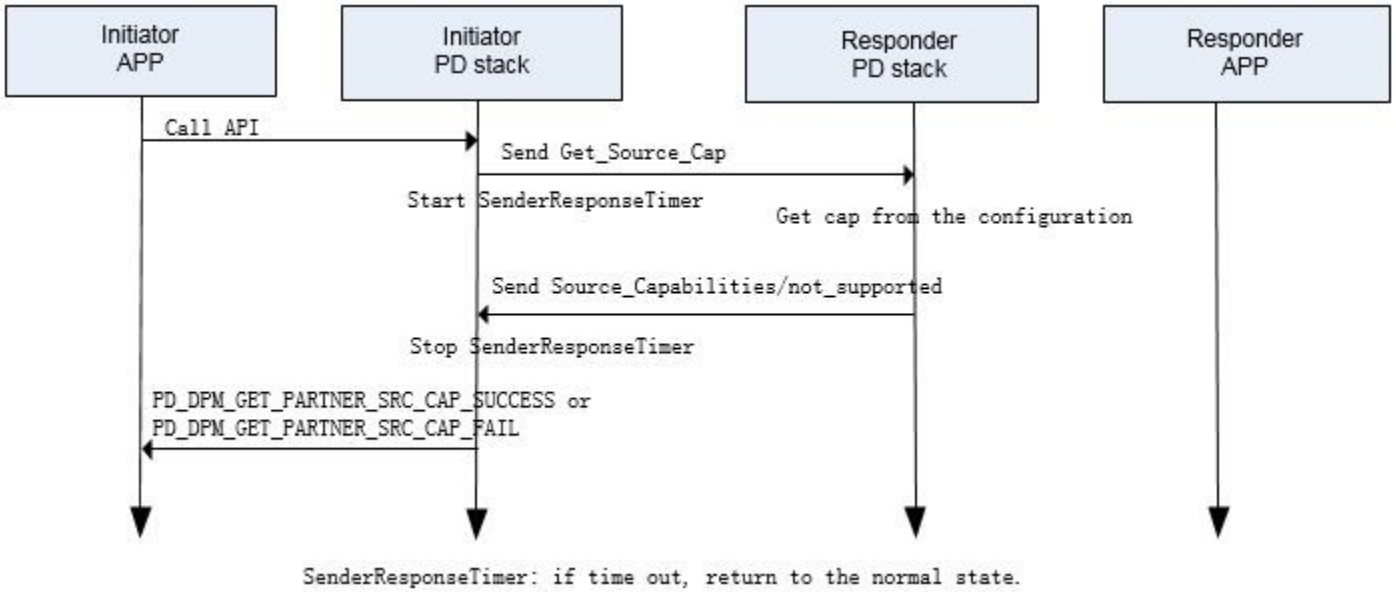

Figure 1.7.8: get partner source cap

- PD_DPM_CONTROL_GET_PARTNER_SINK_CAPABILITIES
  get partner sink capabilities

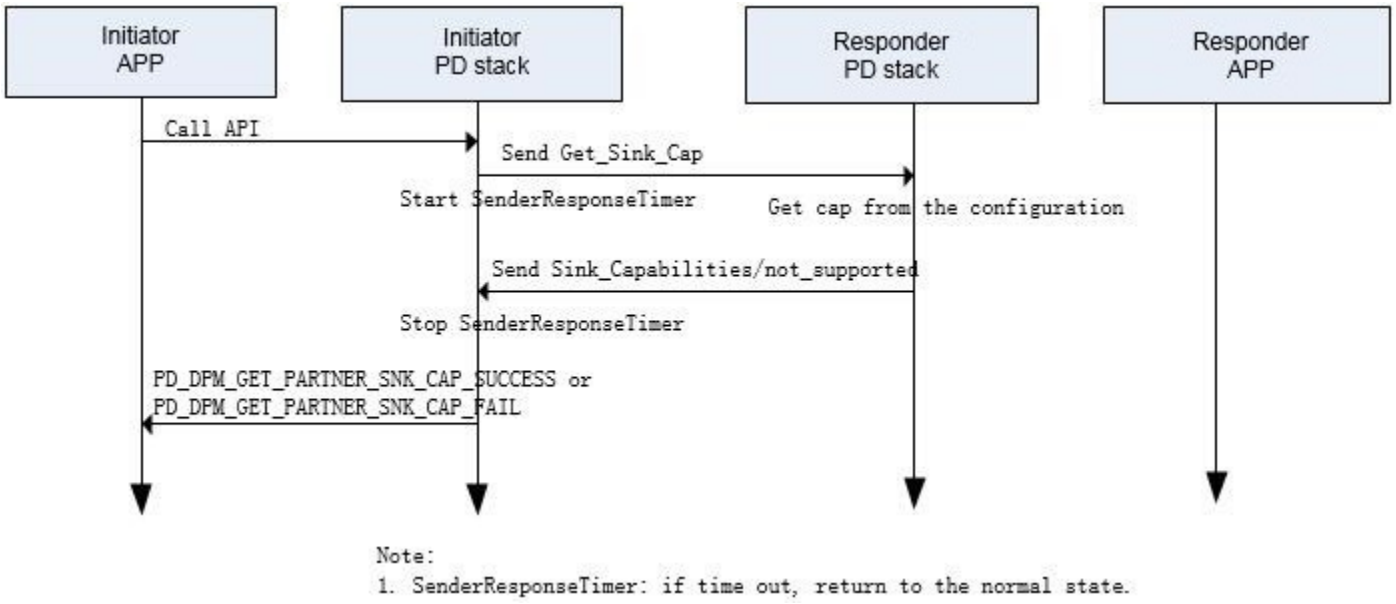

Figure 1.7.9: get partner sink cap

- PD_DPM_CONTROL_PR_SWAP
  power role swap

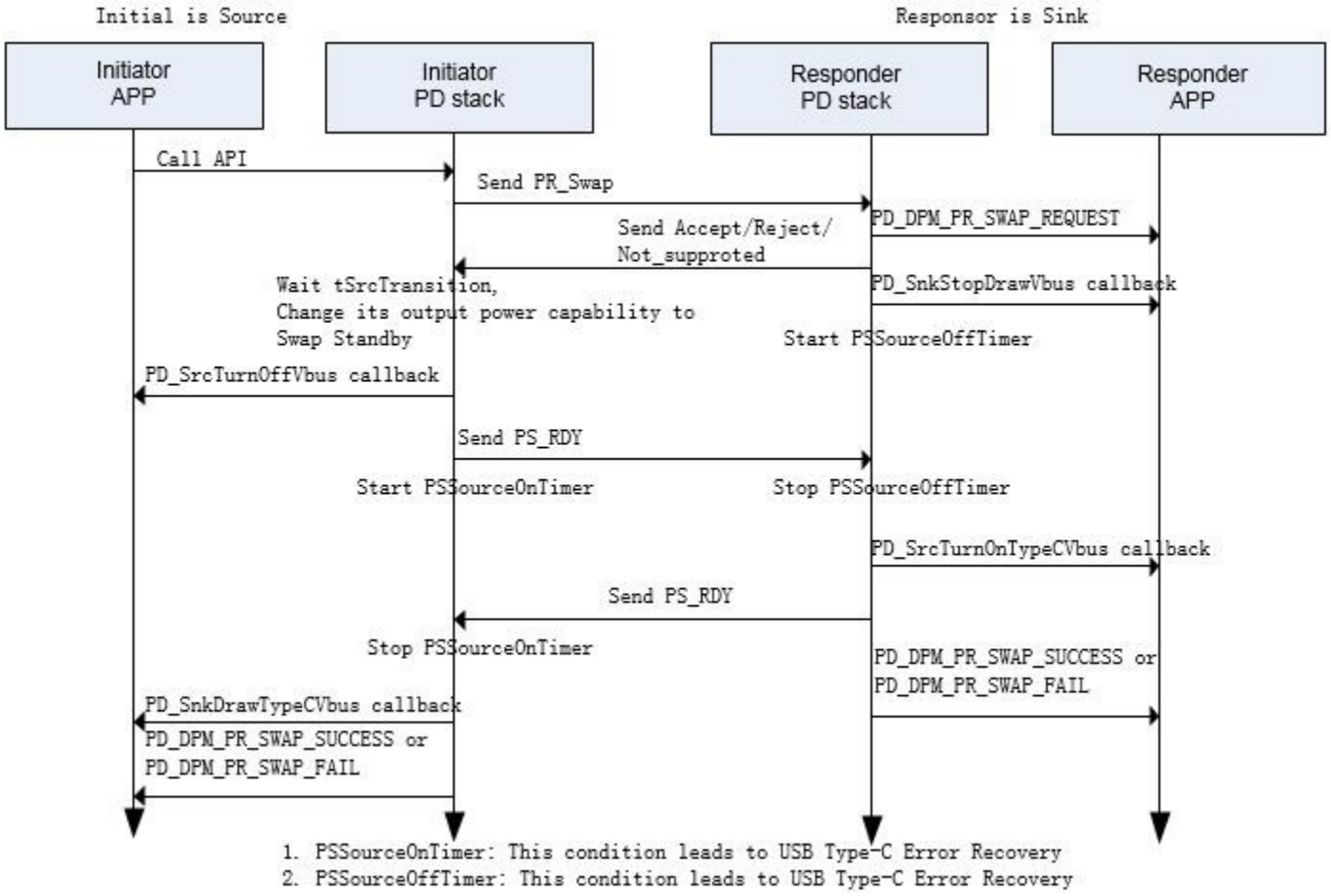

Figure 1.7.10: pr swap (source start)

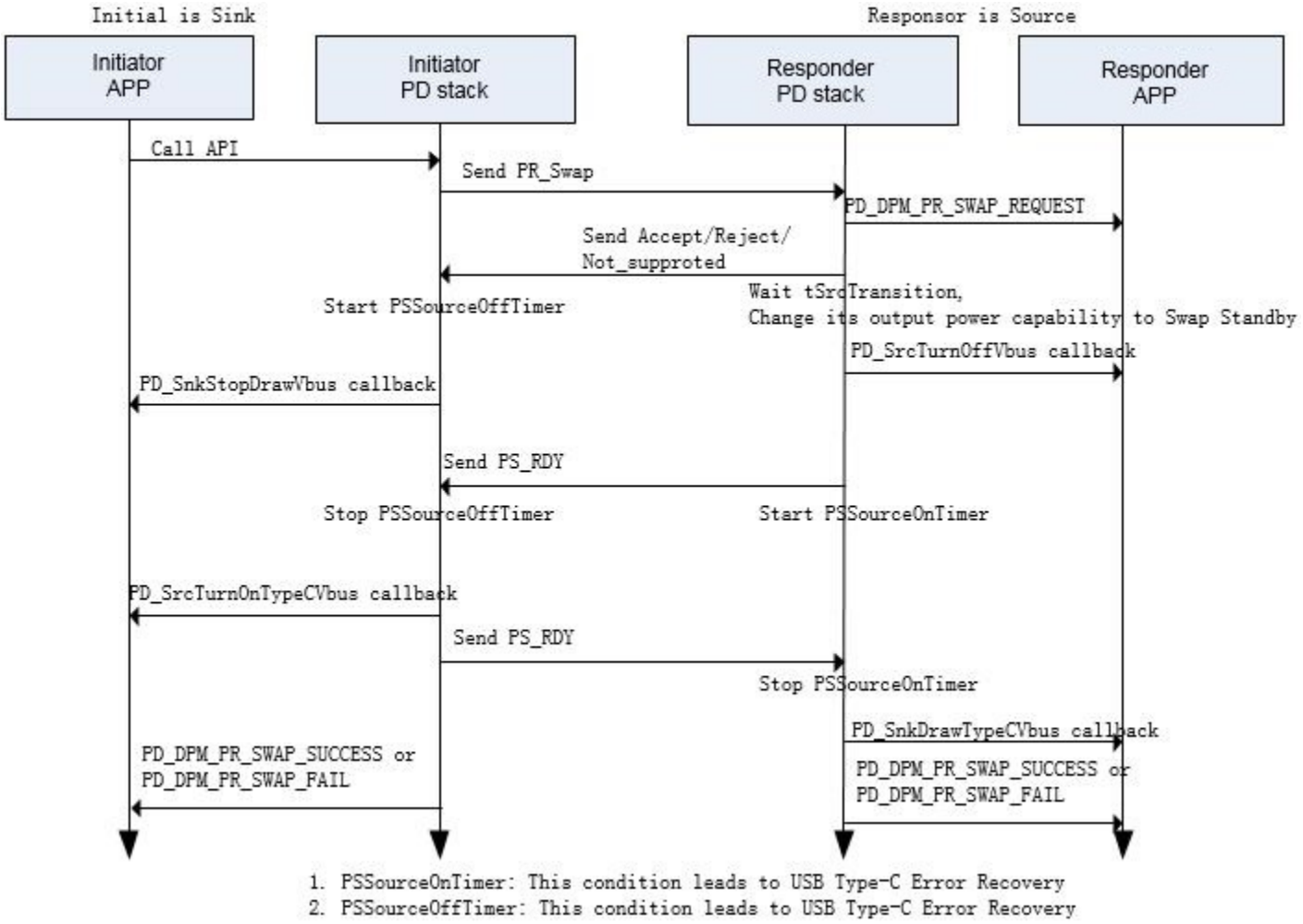

Figure 1.7.11: pr swap (sink start)

- PD_DPM_CONTROL_DR_SWAP
  data role swap

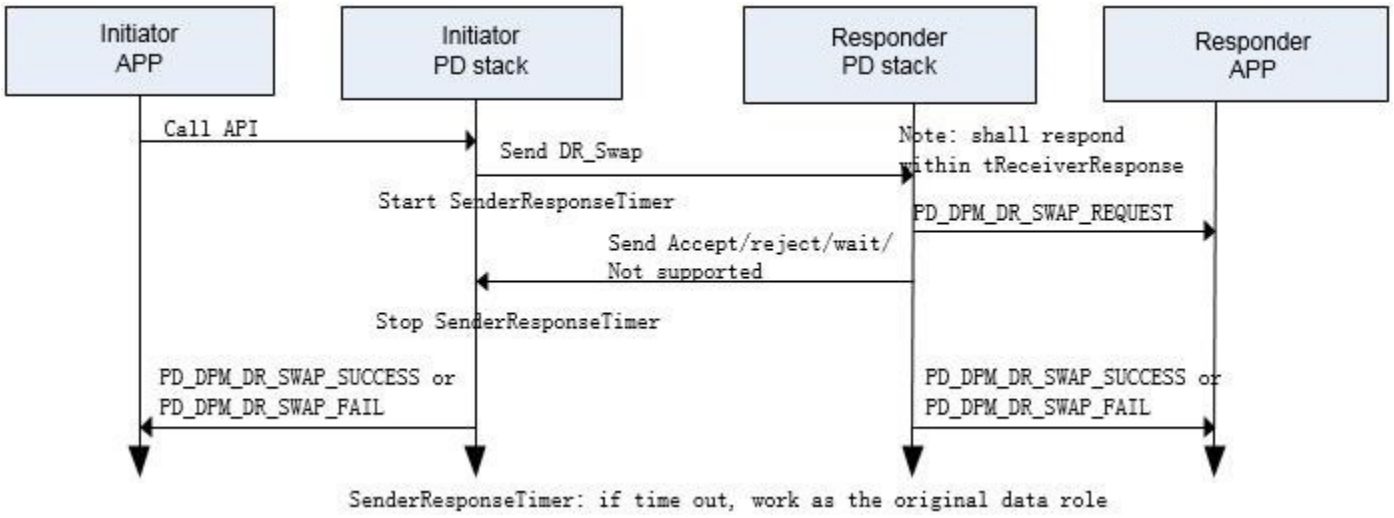

Figure 1.7.12: data role swap

- PD_DPM_CONTROL_VCONN_SWAP
  vconn role swap:

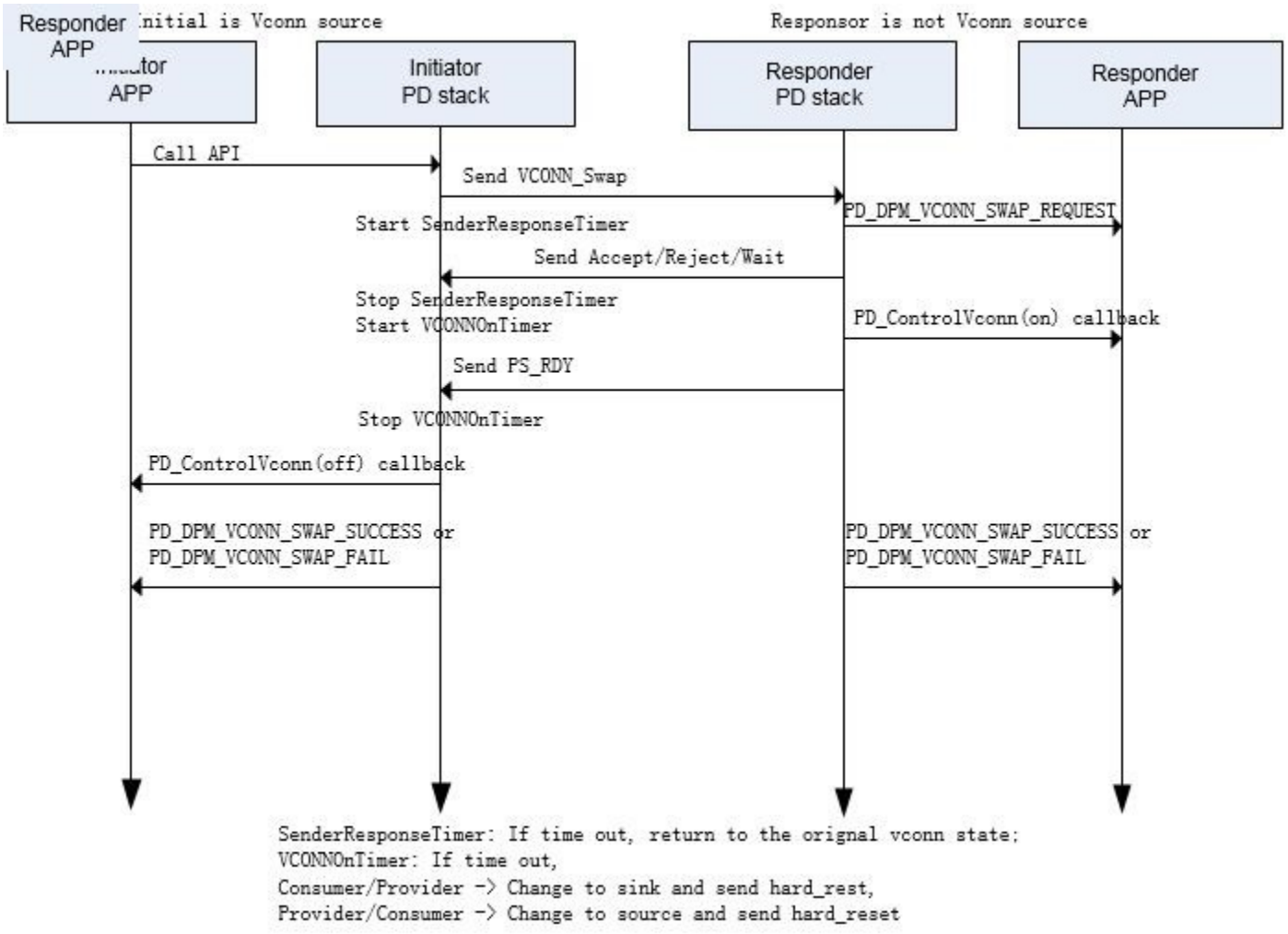

Figure 1.7.13: vconn swap 1

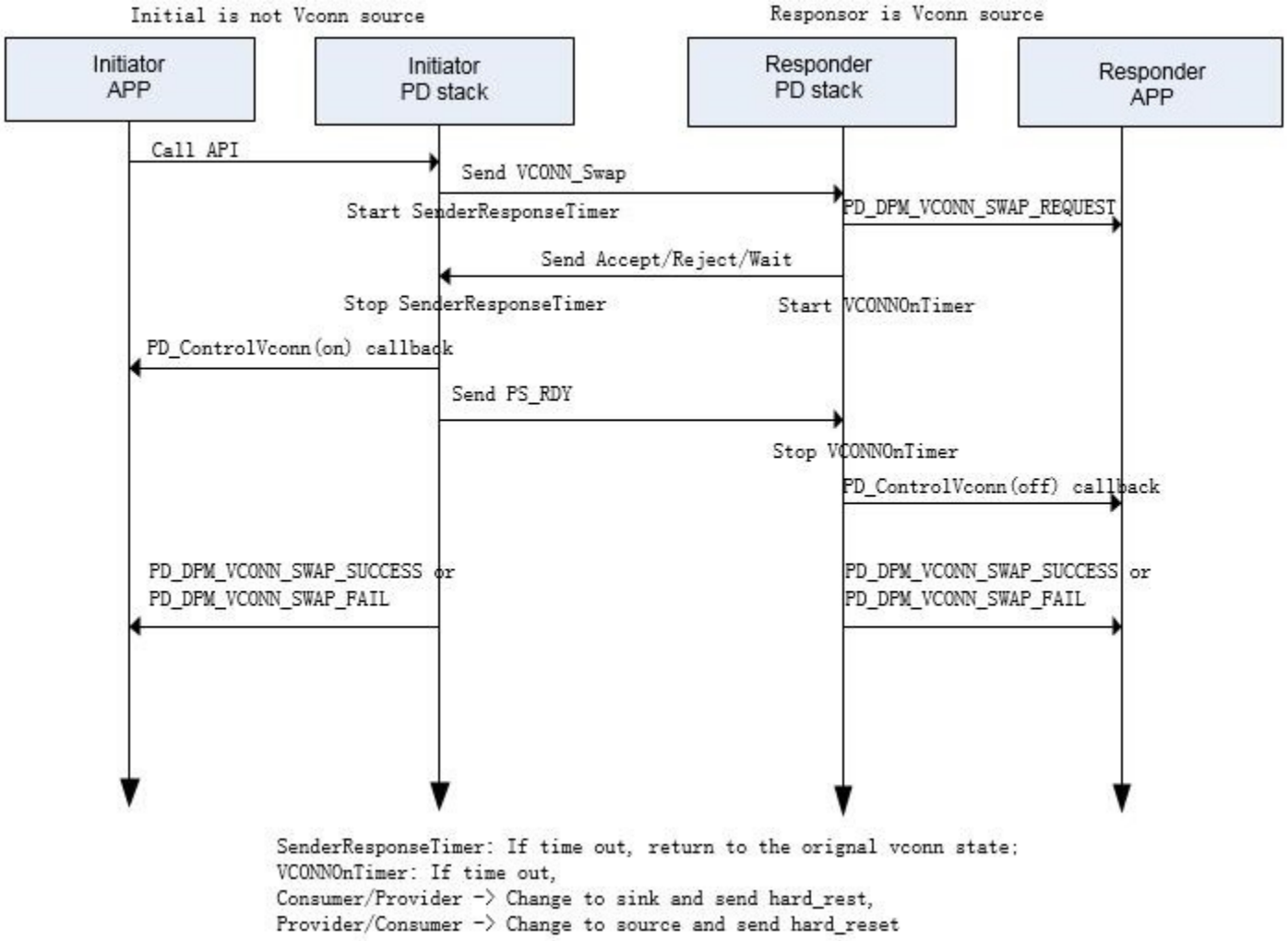

Figure 1.7.14: vconn swap 2

- PD_DPM_CONTROL_SOFT_RESET
  application can send soft_reset actively:

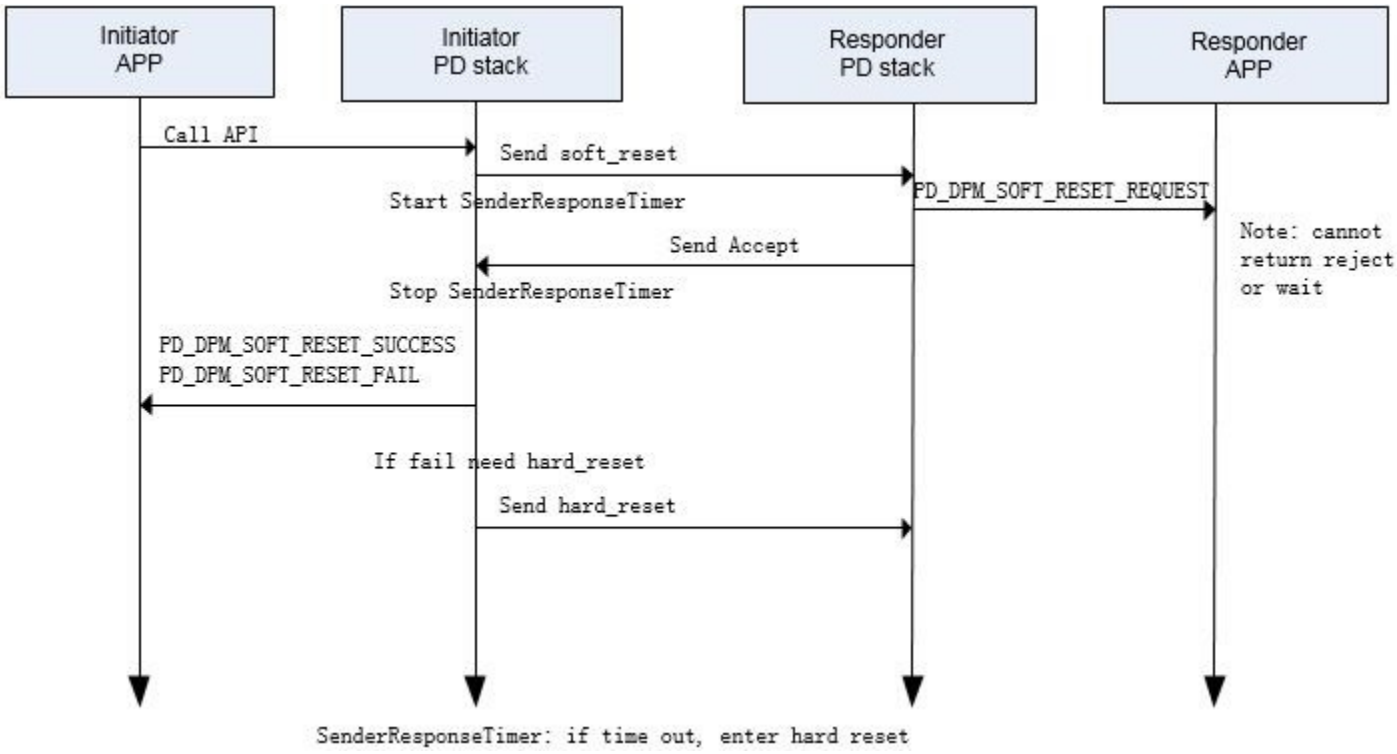

Figure 1.7.15: soft reset (app start)

PD stack may send soft_reset when there is error.

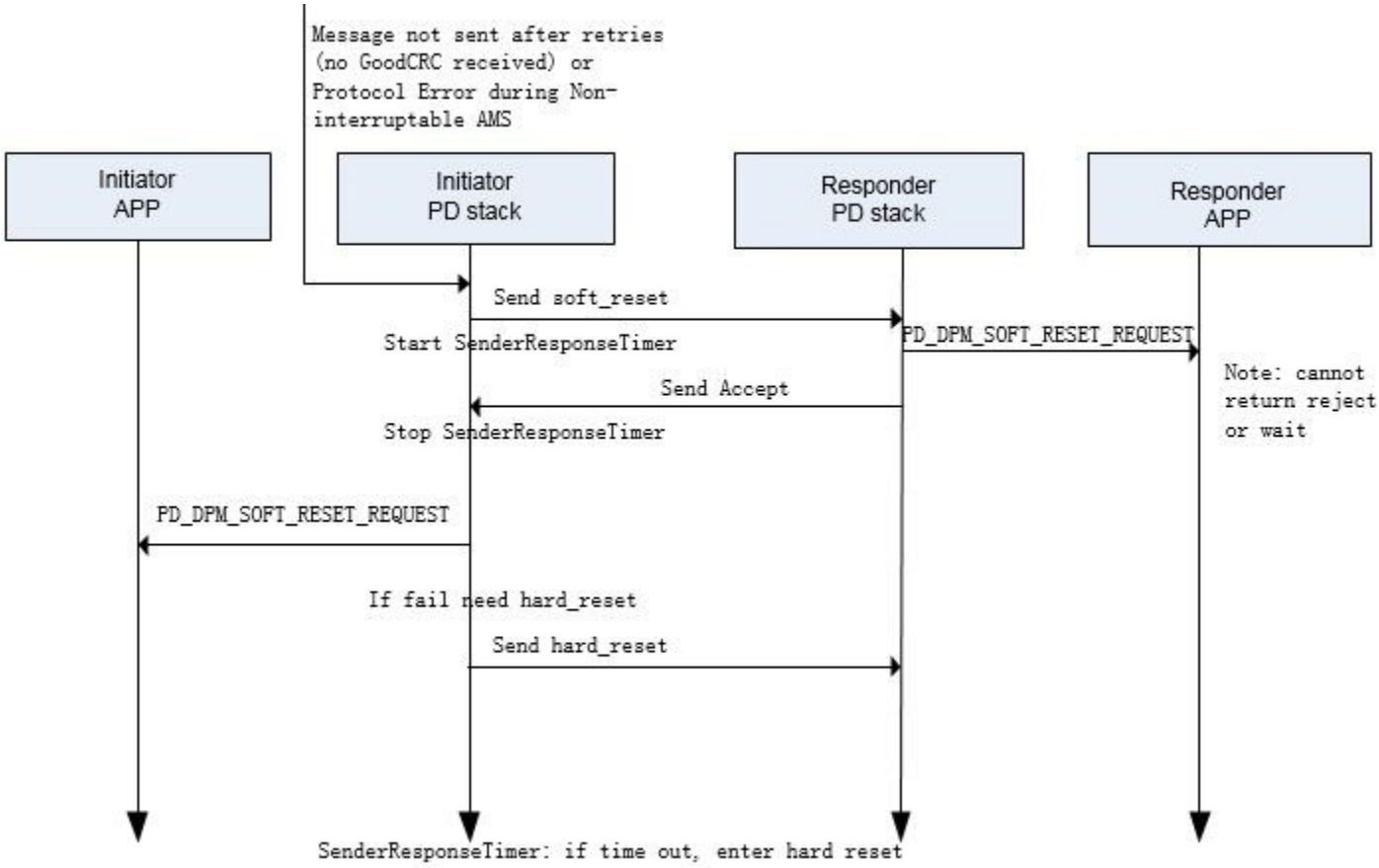

Figure 1.7.16: soft reset (stack start)

- PD_DPM_CONTROL_HARD_RESET
  application can send hard_reset actively, and PD stack may send hard_reset when there is error.

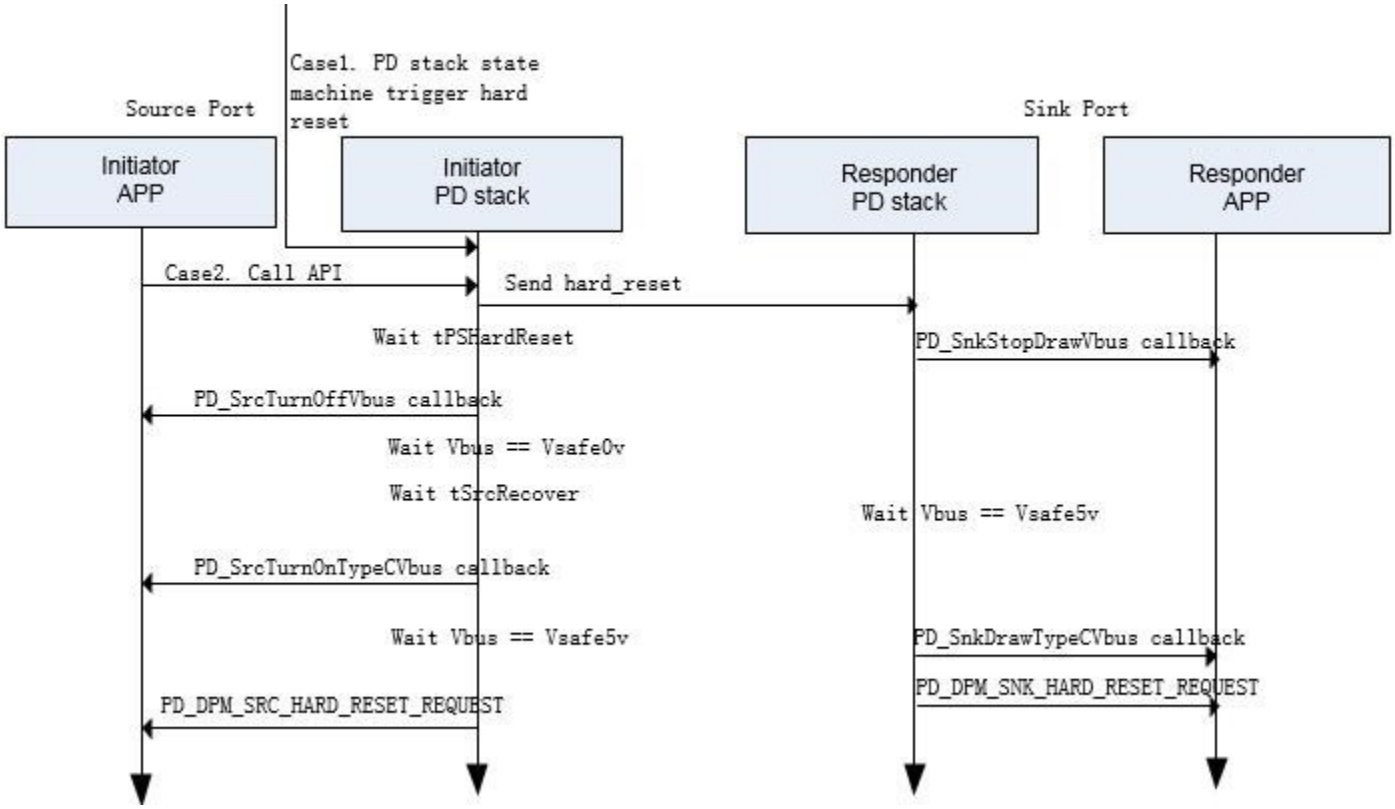

Figure 1.7.17: hard reset (source start)

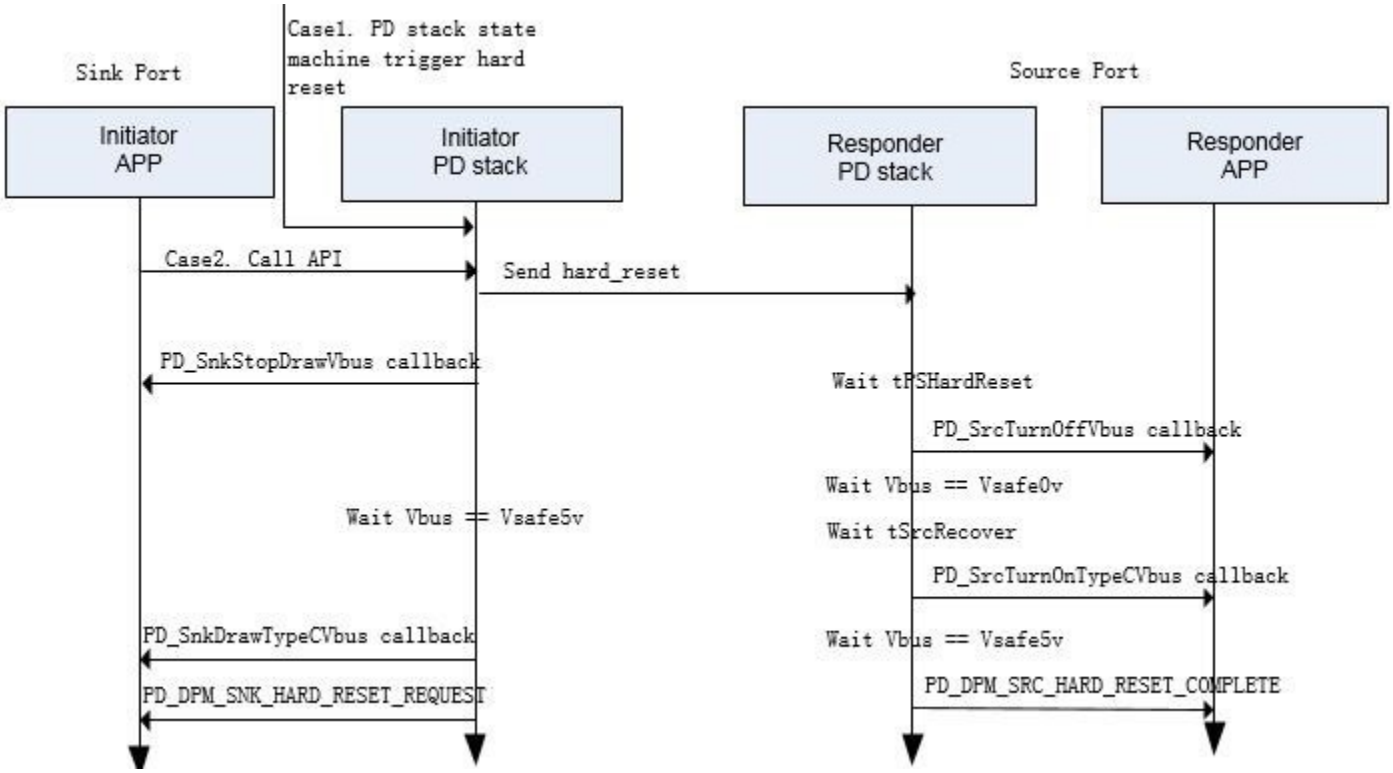

Figure 1.7.18: hard reset (sink start)

- PD_DPM_CONTROL_DISCOVERY_IDENTITY
  discovery identity
- PD_DPM_CONTROL_DISCOVERY_SVIDS
  discovery SVIDs
- PD_DPM_CONTROL_DISCOVERY_MODES
  discovery Modes
- PD_DPM_CONTROL_ENTER_MODE
  enter mode
- PD_DPM_CONTROL_EXIT_MODE
  exit mode

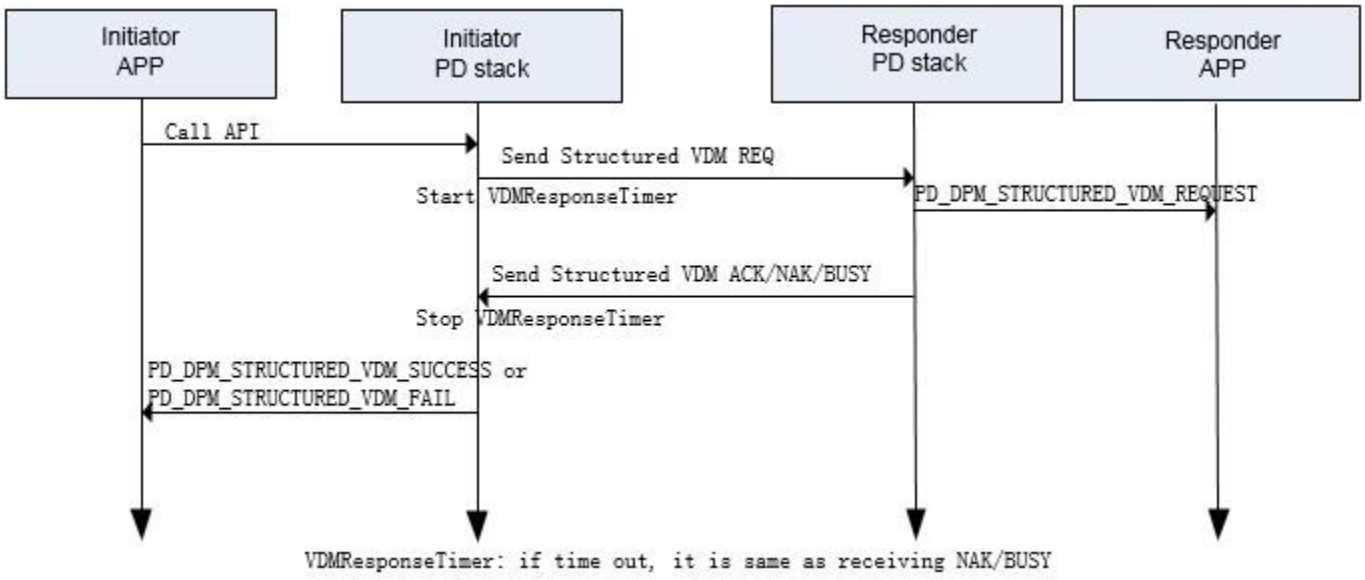

Figure 1.7.19: standard structured vdm

Note: For exit_mode command, if VDMModeExitTimer time out, work as receiving busy.

- PD_DPM_CONTROL_SEND_ATTENTION
  attention

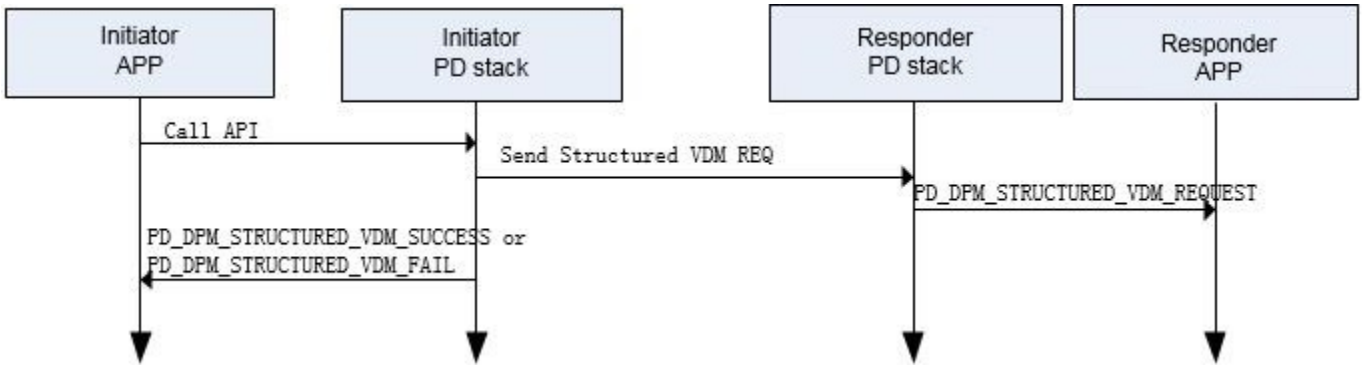

Figure 1.7.20: attention

- PD_DPM_SEND_VENDOR_STRUCTURED_VDM
  send vendor defined structured vdm the structured vdm has reply, the flow is as follow:

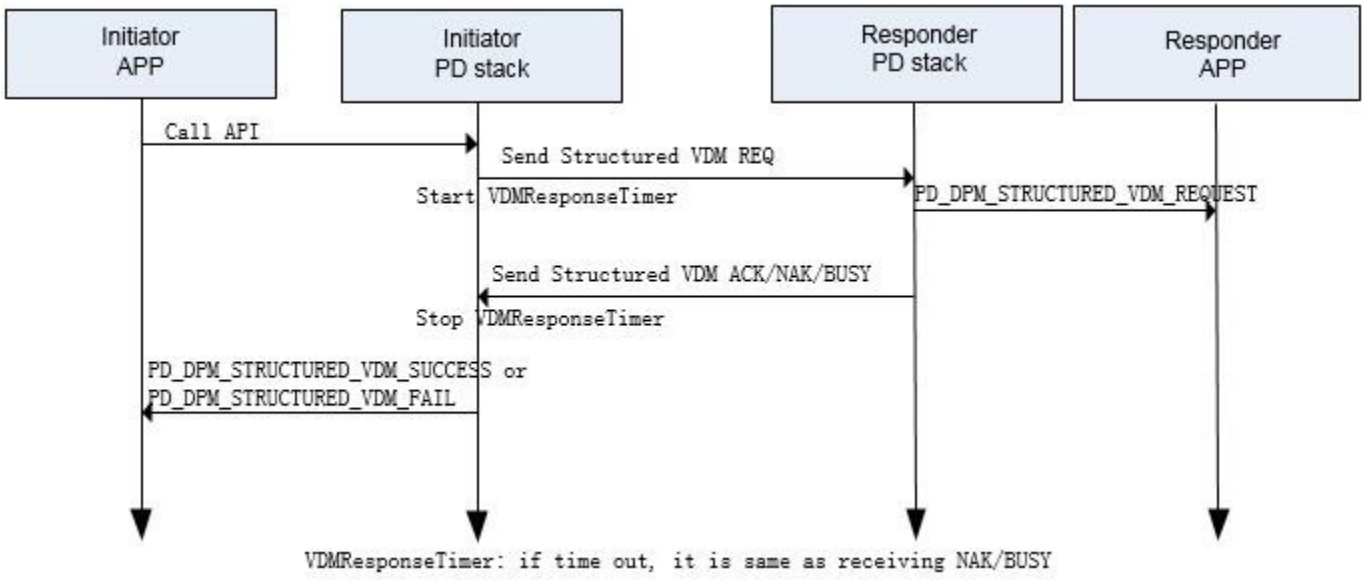

Figure 1.7.21: vendor structured vdm (with reply)

If the structured vdm doesn't have reply, the flow is as follow:

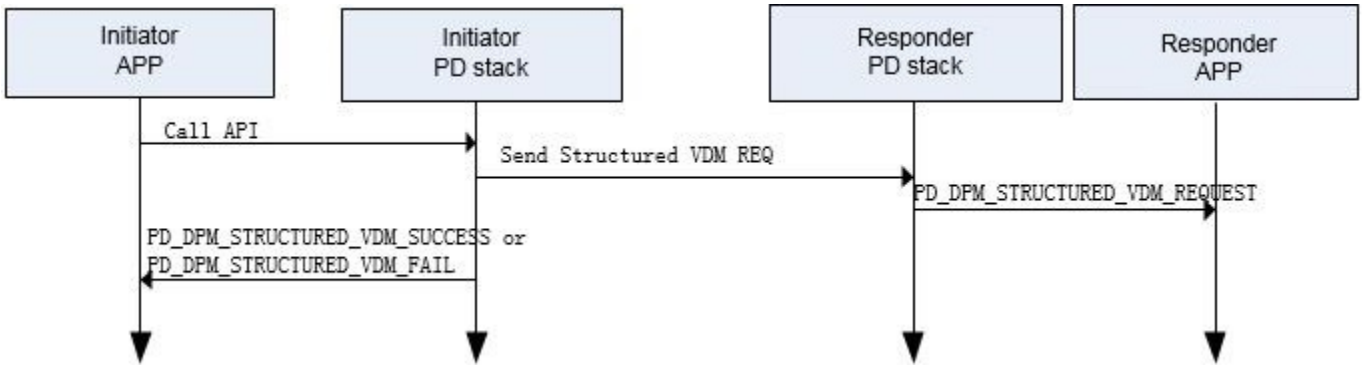

Figure 1.7.22: vendor structured vdm (without reply)

- PD_DPM_SEND_UNSTRUCTURED_VDM
  send unstructured vdm

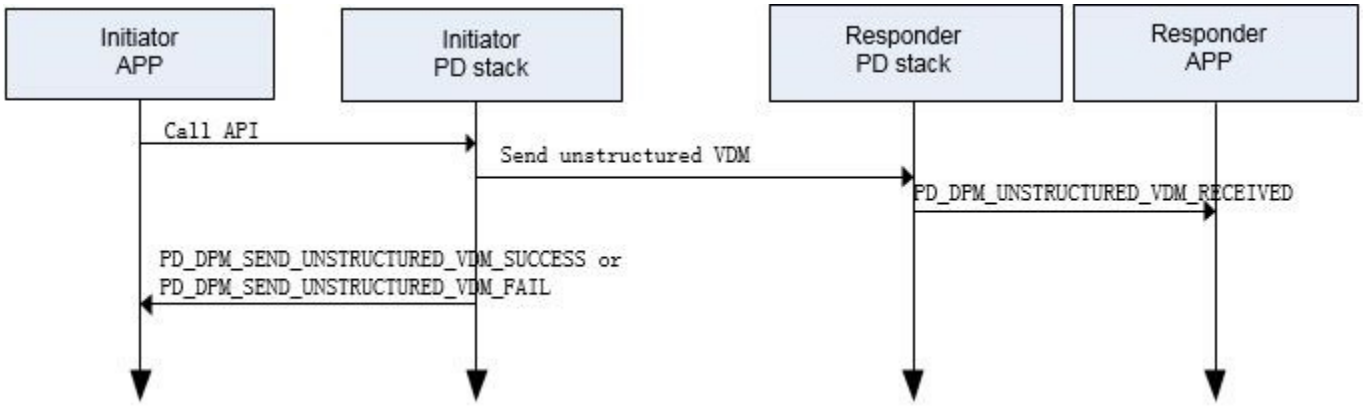

Figure 1.7.23: unstructured vdm

- PD_DPM_FAST_ROLE_SWAP
  fast role swap

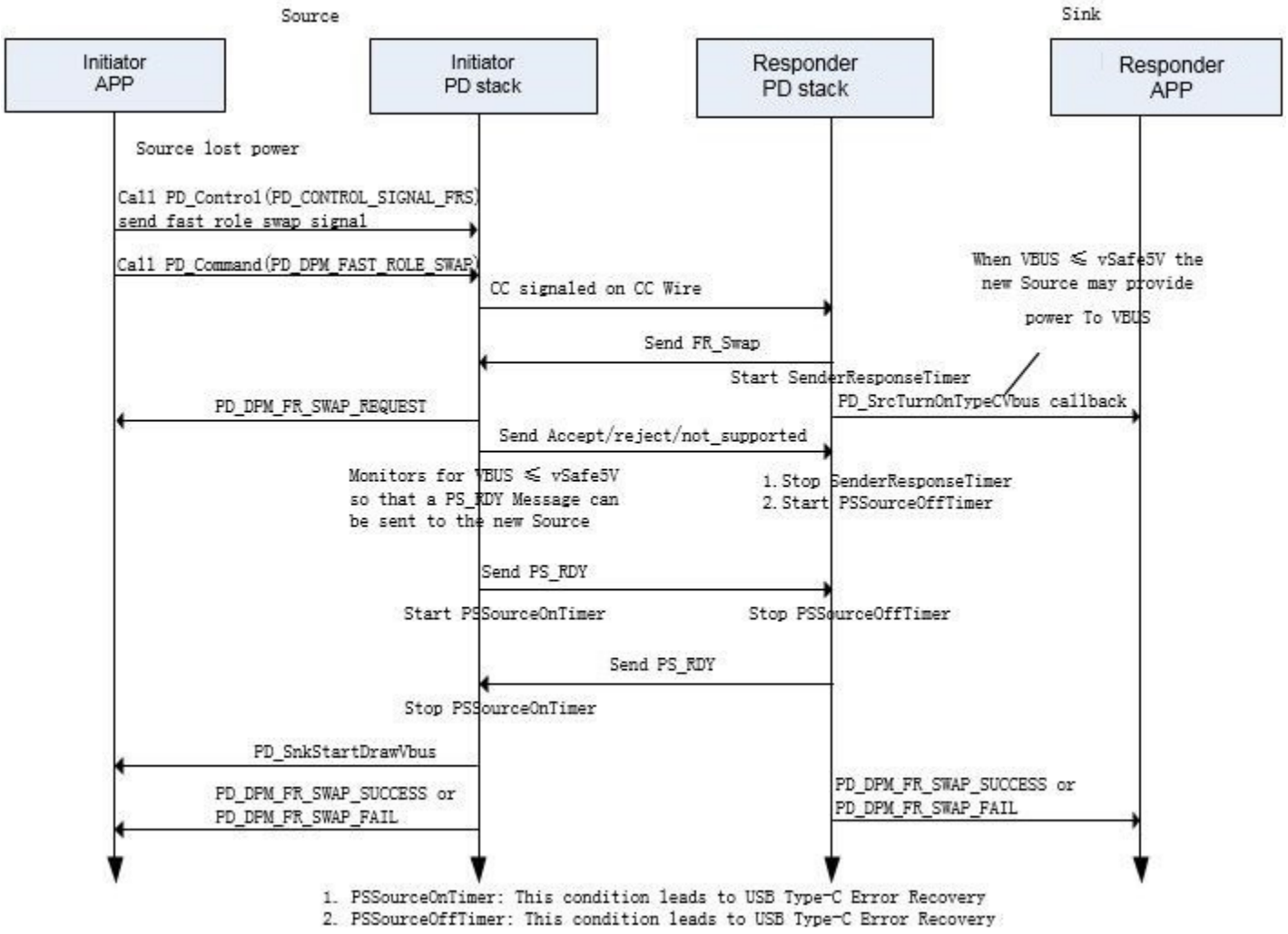

Figure 1.7.24: fast role swap

Note: the PD_SrcTurnOnTypeCVbus may not be called as the chart's flow. After receiving the fast role swap, new source need provide power to VBUS when VBUS <= vSafe5V.

- PD_DPM_GET_SRC_EXT_CAP
  get partner's source extended capabilities

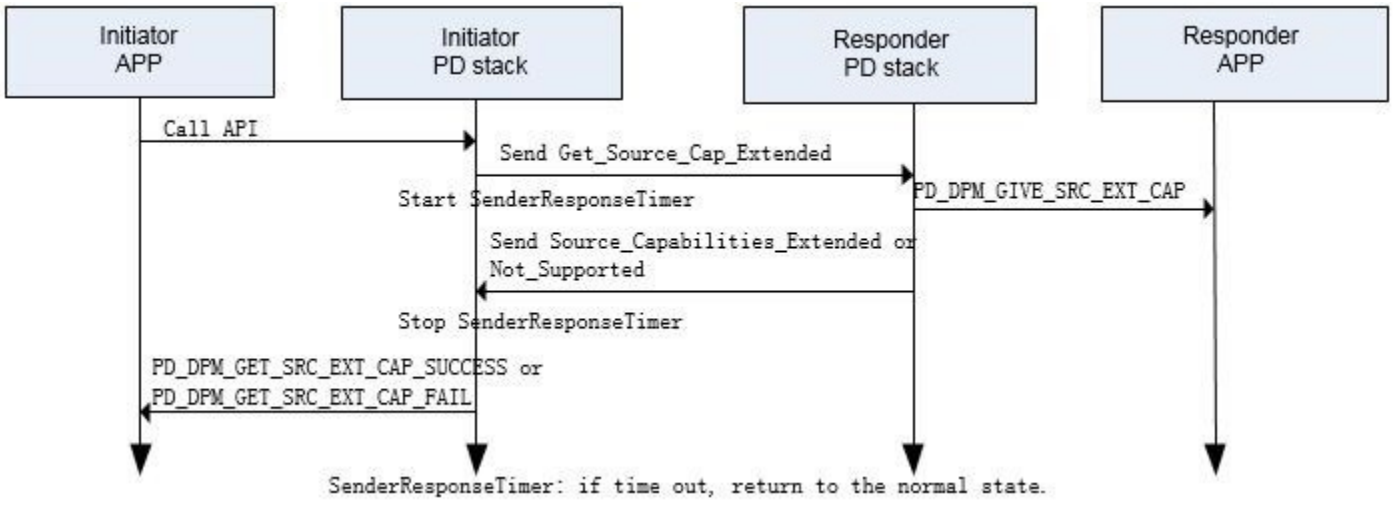

Figure 1.7.25: get source cap ext

- PD_DPM_GET_STATUS
  get partner status.

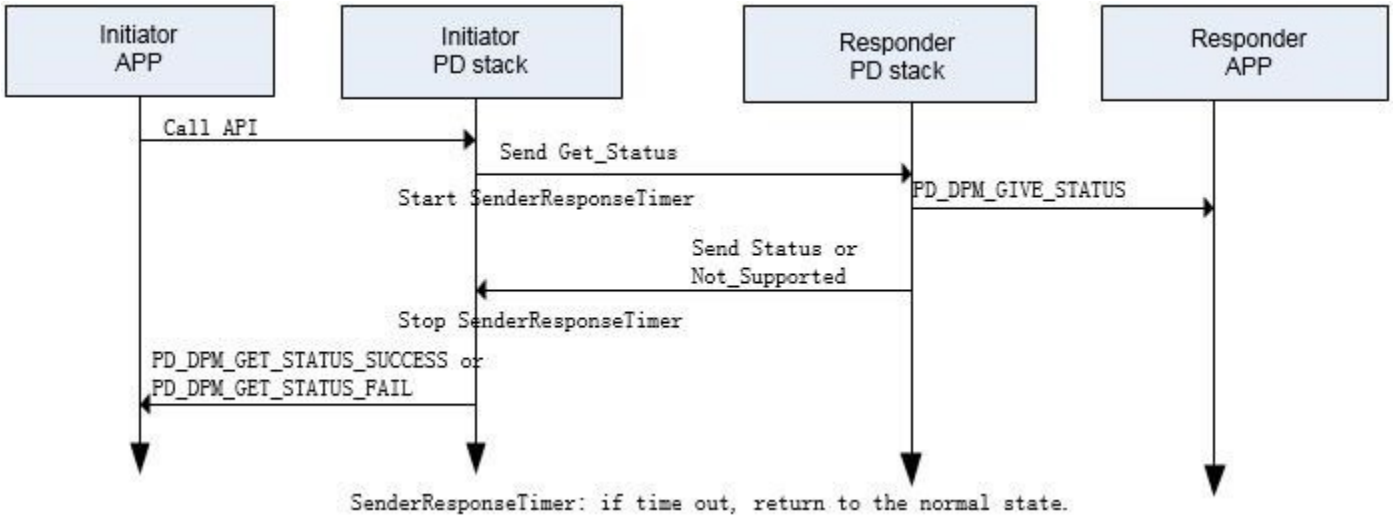

Figure 1.7.26: get status

- PD_DPM_GET_BATTERY_CAP
  get partner battery cap

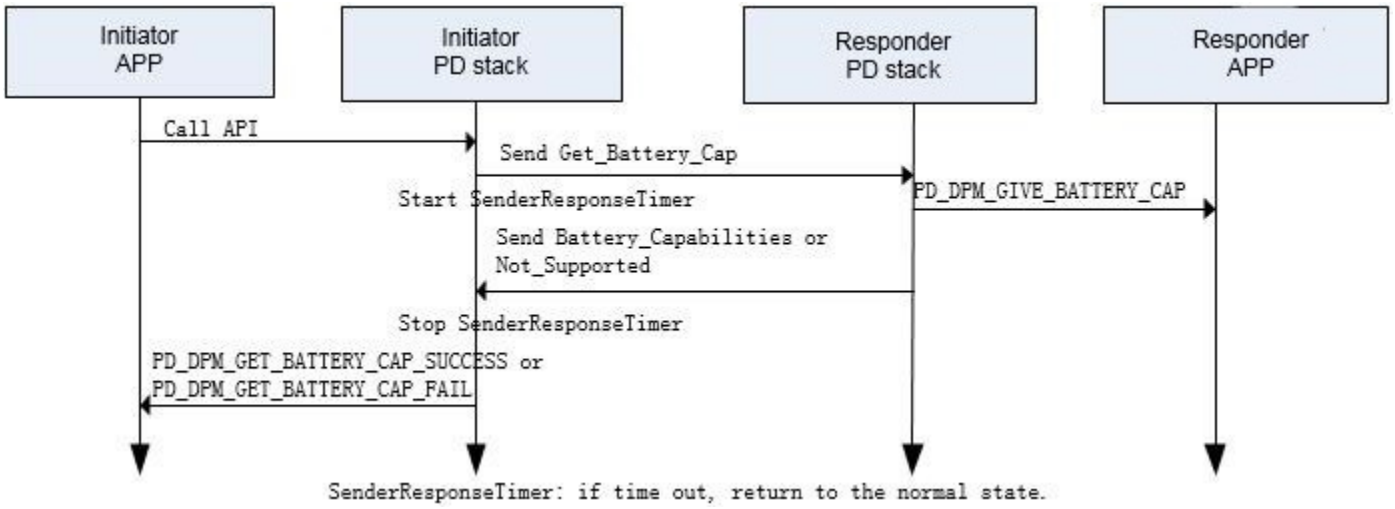

Figure 1.7.27: get battery cap

- PD_DPM_GET_BATTERY_STATUS
  get partner battery status

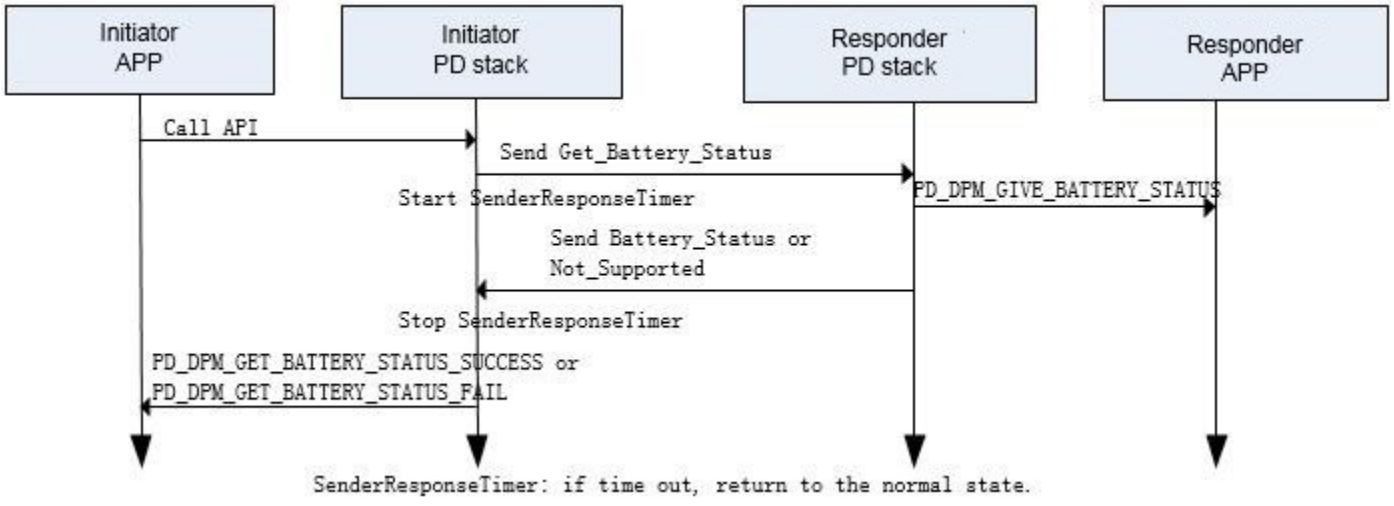

Figure 1.7.28: get battery status

- PD_DPM_GET_MANUFACTURER_INFO
  get partner's manufacturer info

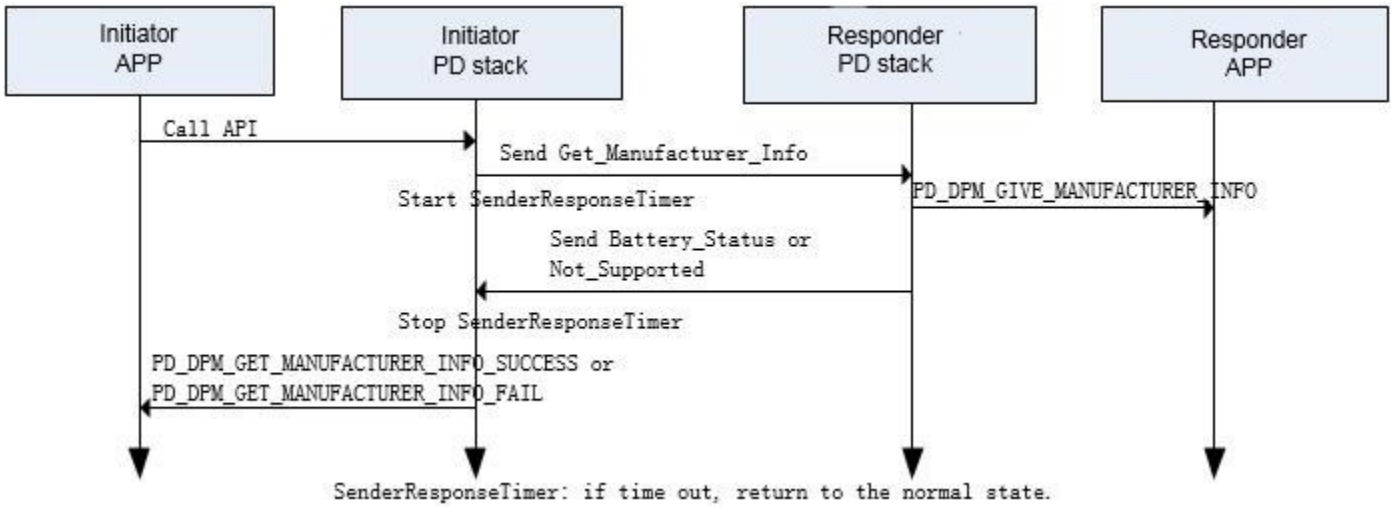

Figure 1.7.29: get manufacturer info

- PD_DPM_ALERT
  alert

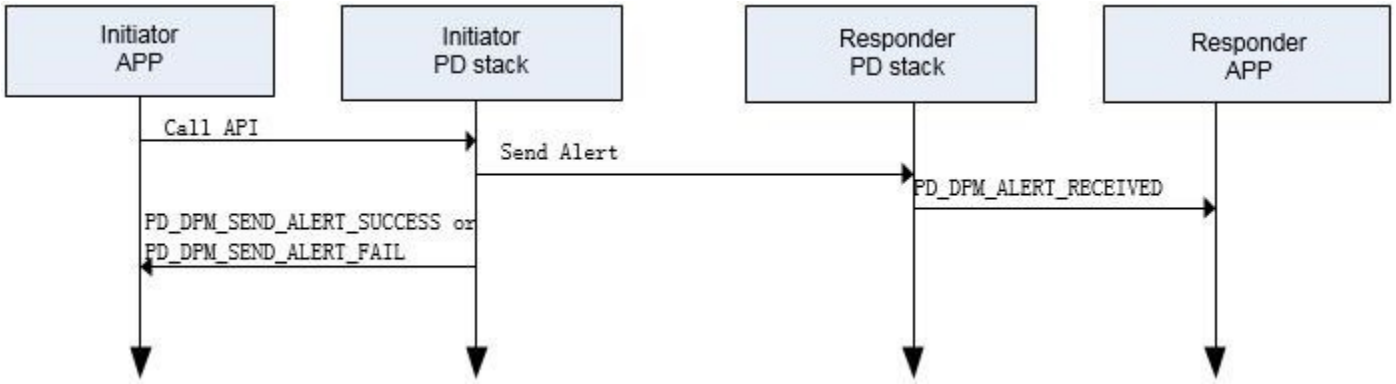

Figure 1.7.30: alert

- PD_DPM_CONTROL_CABLE_RESET
  cable reset

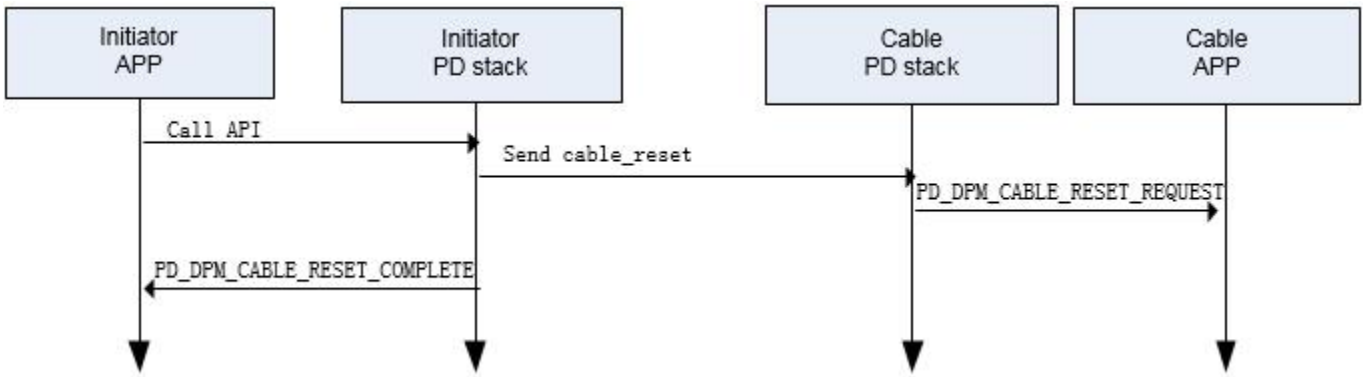

Figure 1.7.31: cable reset

1.8 USB PD low power

The PD stack supports low power function. It can be configured by the macro of PD_CONFIG_PHY_-LOW_POWER_LEVEL. The macro of PD_CONFIG_PHY_LOW_POWER_LEVEL can be configured as 0, 1 and 2. Users can choose different low power levels according to their own needs.

- 0, disable low power function. PTN5110 PHY will never enter low power state.
- 1, enter low power state when the port is in the detached state.
- 2, enter low power state when the port is in the detached state or there is not AMS in progress in the attached state. When entering low power state, PTN5110 PHY will keep a low power consumption and meanwhile will disable the following functional blocks. For more information about the power consumption parameters, please refer to the PTN5110 datasheet.
- Disable VBUS detection
- Disable VBUS monitoring
- Disable VBUS voltage alarm
- Disable VBUS auto discharge
- Disable fast role swap
- Disable fault status reporting
- Enable I2C clock stretching
  Note: The low power function is only for PTN5110 PHY. The low power function for MCU needs users to implement acorrding to their own needs.

Chapter 2
USB Type-C PD Stack

2.1 Overview

Data Structures

- struct pd_alt_mode_module_t
  PD alternating mode's modules config. More...
- struct pd_alt_mode_config_t
  PD alternating mode config. More...
- struct pd_alt_mode_control_t
  PD alternating mode control used by PD_Control(PD_CONTROL_ALT_MODE) More...
- struct pd_dp_mode_obj_t
  DisplayPort Capabilities (VDO in Responder Discover Modes VDM) More...
- struct pd_dp_status_obj_t
  DisplayPort Status. More...
- struct pd_dp_configure_obj_t
  DisplayPort Configurations. More...
- struct pd_altmode_dp_modes_sel_t
  DisplayPort mode select callback parameter. More...
- struct pd_dp_peripheral_interface_t
  DisplayPort mode board peripheral interface. More...
- struct pd_alt_mode_dp_config_t
  DisplayPort mode config. More...
- struct pd_power_port_config_t
  Normal power port's configuration. More...
- struct pd_phy_config_t
  PD phy interface parameter. More...
- struct pd_instance_config_t
  PD instance config. More...
- struct pd_source_fixed_pdo_t
  fixed source PDO More...
- struct pd_sink_fixed_pdo_t
  fixed sink PDO More...
- struct pd_source_variable_pdo_t
  variable source PDO More...
- struct pd_source_apdo_pdo_t
  APDO PDO. More...
- struct pd_sink_variable_pdo_t
  variable sink PDO More...
- struct pd_source_battery_pdo_t
  battery source PDO More...
- struct pd_sink_battery_pdo_t
  battery sink PDO More...
- struct pd_sink_apdo_pdo_t
  APDO PDO. More...
- struct pd_pdo_common_t

common PDO for union More...

- struct pd_source_pdo_t
  source PDO union More...
- struct pd_sink_pdo_t
  sink PDO union More...
- struct pd_structured_vdm_header_t
  PD structured VDM header. More...
- struct pd_unstructured_vdm_header_t
  PD unstructured VDM header. More...
- struct pd_extended_msg_header_t
  PD extended message header. More...
- struct pd_msg_header_t
  PD message header. More...
- struct pd_svdm_command_request_t
  PD structured vdm command request (negotiation) More...
- struct pd_svdm_command_result_t
  PD structured vdm command result. More...
- struct pd_svdm_command_param_t
  PD structured vdm command parameter. More...
- struct pd_unstructured_vdm_command_param_t
  PD unstructured vdm command parameter. More...
- struct pd_command_data_param_t
  For some command with data use this structure as parameter. More...
- struct pd_rdo_t
  PD request data object. More...
- struct pd_capabilities_t
  PD capabilities info. More...
- struct pd_negotiate_power_request_t
  negotiate power RDO request More...
- struct pd_bist_object_t
  BIST data object. More...
- struct pd_ptn5110_ctrl_pin_t
  control PTN5110 pin for power control More...
- struct pd_id_header_vdo_vdm10_t
  ID Header VDO (VDM1.0) More...
- struct pd_id_header_vdo_t
  ID Header VDO (VDM2.0) More...
- struct pd_passive_cable_vdo_vdm20_t
  Passive Cable VDO (VDM2.0) More...
- struct pd_passive_cable_vdo_vdm10_t
  Passive Cable VDO (VDM1.0) More...
- struct pd_active_cable_vdo_vdm20_t
  Active Cable VDO (VDM2.0) More...
- struct pd_active_cable_vdo_vdm10_t
  Active Cable VDO (VDM1.0) More...
- struct pd_cable_plug_info_t
  cable information More...
- struct pd_ama_vdo_t
  Alternate Mode Adapter VDO, has been depreacted in the latest PD specification. More...
- struct pd_ufp_vdo_t
  UFP VDO. More...

- struct pd_power_handle_callback_t
  power control interface. More...
- struct pd_auto_policy_t
  pd auto policy configuration. More...

Macros

- #define PD_VENDOR_ID_NXP (0x1FC9U)
  NXP's USB vendor id.
- #define PD_SPEC_REVISION_10 (0x00U)
  The Specification Revision 1.0 value.
- #define PD_SPEC_REVISION_20 (0x01U)
  The Specification Revision 2.0 value.
- #define PD_SPEC_REVISION_30 (0x02U)
  The Specification Revision 3.0 value.
- #define PD_SPEC_STRUCTURED_VDM_VERSION_10 (0x00U)
  The structured VDM version 1.0 value.
- #define PD_SPEC_STRUCTURED_VDM_VERSION_20 (0x01U)
  The structured VDM version 2.0 value.
- #define PD_STANDARD_ID (0xFF00U)
  PD standard ID.
- #define PD_VDO_VERSION_10 (0x00U)
  PD VDO version 1.0 value.
- #define PD_VDO_VERSION_11 (0x01U)
  PD VDO version 1.1 value.
- #define PD_VDO_VERSION_12 (0x02U)
  PD VDO version 1.2 value.
- #define PD_PDO_VOLTAGE_UNIT (50U)
  source/sink capability's voltage unit is 50mV
- #define PD_PDO_CURRENT_UNIT (10U)
  source/sink capability's current unit is 10mA
- #define PD_PDO_POWER_UNIT (250U)
  source/sink capability's power unit is 250mW
- #define PD_APDO_VOLTAGE_UNIT (100U)
  source/sink APDO capability's voltage unit is 100mV
- #define PD_APDO_CURRENT_UNIT (50U)
  source/sink APDO capability's current unit is 50mA
- #define PD_PRDO_VOLTAGE_UNIT (20U)
  Programmable Request Data Object's voltage unit is 20mV.
- #define PD_PRDO_CURRENT_UNIT (50U)
  Programmable Request Data Object's current unit is 50mA.
- #define PD_SINK_DETACH_ON_VBUS_ABSENT (1)
  check detach based on vbus absent
- #define PD_SINK_DETACH_ON_CC_OPEN (2)
  check detach based CC line open

Typedefs

- typedef void * pd_handle
  pd instance handle, return by PD_InstanceInit
- typedef void * pd_phy_handle

pd phy instance handle

- typedef pd_status_t(∗ pd_stack_callback_t)(void ∗callbackParam, uint32_t event, void ∗param)
  pd instance callback function.

Enumerations

- enum pd_alt_mode_control_code_t { ,
  kAltMode_TriggerEnterMode,
  kAltMode_TriggerExitMode,
  kAltMode_GetModeState,
  kDPControl_HPDDetectEvent }
  PD alternating mode control.
- enum pd_dp_hpd_driver_t {
  kDPHPDDriver_None = 0,
  kDPHPDDriver_IRQ,
  kDPHPDDriver_Low,
  kDPHPDDriver_High,
  kDPHPDDriver_Waiting }
  PD DisplayPort drive the HPD.
- enum pd_displayport_vdm_command_t {
  kDPVDM_StatusUpdate = 0x10,
  kDPVDM_Configure }
  PD DisplayPort VDM command.
- enum pd_status_connected_val_t {
  kDFP_D_NonConnected = 0,
  kDFP_D_Connected = 1,
  kUFP_D_Connected = 2,
  kUFP_D_BothConnected = 3 }
  PD DisplayPort connect state.
- enum pd_configure_set_config_val_t {
  kDPConfig_USB = 0,
  kDPConfig_DFPD = 1,
  kDPConfig_UFPD = 2 }
  PD DisplayPort select configuration.
- enum dp_mode_port_cap_val_t {
  kDPPortCap_Reserved = 0,
  kDPPortCap_UFPD = 1,
  kDPPortCap_DFPD = 2,
  kDPPortCap_Both = 3 }
  PD DisplayPort port capability.
- enum dp_mode_signal_t {
  kDPSignal_Unspecified = 0,
  kDPSignal_DP = 1,
  kDPSignal_USBGEN2 = 2 }
  Signaling for Transport of DisplayPort Protocol.
- enum dp_mode_pin_assign_val_t {

kPinAssign_DeSelect = 0,
kPinAssign_A = 0x01u,
kPinAssign_B = 0x02u,
kPinAssign_C = 0x04u,
kPinAssign_D = 0x08u,
kPinAssign_E = 0x10u,
kPinAssign_F = 0x20u }
Configure Pin Assignment.

- enum dp_mode_receptacel_indication_t {
kReceptacle_TypeCPlug = 0,
kReceptacle_TypeCReceptacle = 1 }
Receptacle Indication.
- enum dp_mode_usb20_signal_t {
kUSB2_Required = 0,
kUSB2_NotRequired = 1 }
USB r2.0 Signaling Not Used.
- enum pd_dp_peripheral_control_t {
kDPPeripheal_ControlHPDValue,
kDPPeripheal_ControlHPDSetLow,
kDPPeripheal_ControlHPDReleaseLow,
kDPPeripheal_ControlSetMux,
kDPPeripheal_ControlSetMuxSaftMode,
kDPPeripheal_ControlSetMuxUSB3Only,
kDPPeripheal_ControlSetMuxShutDown,
kDPPeripheal_ControlSetMuxDisable,
kDPPeripheal_ControlSetMuxDP4LANE,
kDPPeripheal_ControlSetMuxDP2LANEUSB3,
kDPPeripheal_ControlSetMuxDP2LANENOUSB,
kDPPeripheal_ControlHPDDetectStart,
kDPPeripheal_ControlHPDDetectStop,
kDPPeripheal_ControlHPDQueueEnable,
kDPPeripheal_ControlHPDQueueDisable,
kDPPeripheal_ControlHPDGetCurrentState }
DisplayPort peripheral control.
- enum pd_status_t {
kStatus_PD_Error = 0x00U,
kStatus_PD_Success,
kStatus_PD_Abort,
kStatus_PD_Cancel,
kStatus_PD_Busy }
pd error code used for function return value or error status.
- enum pd_phy_type_t {
kPD_PhyPTN5110,
kPD_PhyPTN5100 }
PD phy type, used in the pd_instance_config_t.
- enum pd_device_type_t {

kDeviceType_NormalPowerPort,
kDeviceType_Cable,
kDeviceType_AudioAccDevice,
kDeviceType_DebugAccDevice,
kDeviceType_AlternateModeProduct }
The device's type, used in the pd_instance_config_t.

- enum typec_power_role_config_t {
kPowerConfig_SinkOnly,
kPowerConfig_SinkDefault,
kPowerConfig_SourceOnly,
kPowerConfig_SourceDefault,
kPowerConfig_DRPToggling,
kPowerConfig_DRPSourcingDevice,
kPowerConfig_DRPSinkingHost }
configure device's type, used in the pd_instance_config_t.
- enum typec_sink_role_config_t {
kSinkConfig_Normal,
kSinkConfig_AudioAcc,
kSinkConfig_DebugAcc }
config typec sink role
- enum typec_try_t {
kTypecTry_None,
kTypecTry_Src,
kTypecTry_Snk }
configure try type function, used in the pd_instance_config_t.
- enum typec_data_function_config_t {
kDataConfig_None,
kDataConfig_DFP,
kDataConfig_UFP,
kDataConfig_DRD }
configure device's data function, used in the pd_instance_config_t.
- enum pd_phy_interface_t {
kInterface_i2c0 = 0u,
kInterface_i2c1 = 1U,
kInterface_i2c2 = 2U,
kInterface_i2c3 = 3U,
kInterface_i2c4 = 4U,
kInterface_i2c5 = 5U,
kInterface_i2c6 = 6U,
kInterface_spi0 = 0x10u,
kInterface_spi1 = 0x11u,
kInterface_spi2 = 0x12u,
kInterface_spi3 = 0x13u,
kInterface_spi4 = 0x15u,
kInterface_spi5 = 0x16u,
kInterface_spi6 = 0x17u }

PHY interface.

- enum start_of_packet_t {
  kPD_MsgSOP = 0,
  kPD_MsgSOPp = 1,
  kPD_MsgSOPpp = 2,
  kPD_MsgSOPDbg = 3,
  kPD_MsgSOPpDbg = 4,
  kPD_MsgSOPInvalid = 0xFFu,
  kPD_MsgSOPMask = 0x01u,
  kPD_MsgSOPpMask = 0x02u,
  kPD_MsgSOPppMask = 0x04u,
  kPD_MsgSOPDbgMask = 0x08u,
  kPD_MsgSOPpDbgMask = 0x10u }

  start of packet's type.
- enum pd_command_result_t {
  kCommandResult_None = 0x00U,
  kCommandResult_Accept = 0x01U,
  kCommandResult_Success = kCommandResult_Accept,
  kCommandResult_Reject = 0x02U,
  kCommandResult_Wait = 0x03U,
  kCommandResult_Error = 0x04U,
  kCommandResult_NotSupported = 0x05U,
  kCommandResult_VDMACK = 0x06U,
  kCommandResult_VDMNAK = 0x07U,
  kCommandResult_VDMBUSY = 0x08U,
  kCommandResult_Timeout = 0x09U }

  pd command error code.
- enum typec_current_val_t {
  kCurrent_Invalid = 0,
  kCurrent_StdUSB = 1,
  kCurrent_1A5 = 2,
  kCurrent_3A = 3 }

  Typec current level related to Rp value.
- enum pd_power_role_t {
  kPD_PowerRoleSink,
  kPD_PowerRoleSource,
  kPD_PowerRoleNone }

  PD running power role type.
- enum pd_data_role_t {
  kPD_DataRoleUFP,
  kPD_DataRoleDFP,
  kPD_DataRoleNone }

  PD running data role type.
- enum pd_vconn_role_t {
  kPD_NotVconnSource,
  kPD_IsVconnSource,

kPD_VconnNone }
Vconn role type (Vconn source or not).

- enum typec_port_connect_state_t {
  kTYPEC_ConnectNone = 0,
  kTYPEC_ConnectSource = 1,
  kTYPEC_ConnectSink = 2,
  kTYPEC_ConnectPoweredCable = 3,
  kTYPEC_ConnectPoweredCableWithSink = 4,
  kTYPEC_ConnectVconnPoweredAccessory = 4,
  kTYPEC_ConnectAudioAccessory = 5,
  kTYPEC_ConnectDebugAccessory = 6 }
  TypeC connection state, self or partner's device type.
- enum pd_vdm_command_t {
  kVDM_DiscoverIdentity = 1,
  kVDM_DiscoverSVIDs,
  kVDM_DiscoverModes,
  kVDM_EnterMode,
  kVDM_ExitMode,
  kVDM_Attention }
  Structured VDM command values.
- enum pd_vdm_command_type_t {
  kVDM_Initiator,
  kVDM_ResponderACK,
  kVDM_ResponderNAK,
  kVDM_ResponderBUSY,
  kVDM_ComandTypeInvalid = 0xFFU }
  Structured VDM command type values.
- enum pd_dpm_callback_event_t { ,

PD_DPM_SNK_HARD_RESET_REQUEST,
PD_DPM_SRC_HARD_RESET_REQUEST,
PD_DPM_PR_SWAP_REQUEST,
PD_DPM_PR_SWAP_SUCCESS,
PD_DPM_PR_SWAP_FAIL,
PD_DPM_FR_SWAP_REQUEST,
PD_DPM_FR_SWAP_SUCCESS,
PD_DPM_FR_SWAP_FAIL,
PD_DPM_SRC_RDO_REQUEST,
PD_DPM_SRC_CONTRACT_STILL_VALID,
PD_DPM_SRC_SEND_SRC_CAP_FAIL,
PD_DPM_SRC_RDO_SUCCESS,
PD_DPM_SRC_RDO_FAIL,
PD_DPM_SNK_RECEIVE_PARTNER_SRC_CAP,
PD_DPM_SNK_GET_RDO,
PD_DPM_SNK_RDO_SUCCESS,
PD_DPM_SNK_RDO_FAIL,
PD_DPM_GET_PARTNER_SRC_CAP_FAIL,
PD_DPM_GET_PARTNER_SRC_CAP_SUCCESS,
PD_DPM_SRC_GOTOMIN_SUCCESS,
PD_DPM_SNK_GOTOMIN_SUCCESS,
PD_DPM_SRC_GOTOMIN_FAIL,
PD_DPM_SNK_GOTOMIN_FAIL,
PD_DPM_GET_PARTNER_SNK_CAP_SUCCESS,
PD_DPM_GET_PARTNER_SNK_CAP_FAIL,
PD_DPM_DR_SWAP_REQUEST,
PD_DPM_DR_SWAP_SUCCESS,
PD_DPM_DR_SWAP_FAIL,
PD_DPM_VCONN_SWAP_REQUEST,
PD_DPM_VCONN_SWAP_SUCCESS,
PD_DPM_VCONN_SWAP_FAIL,
PD_DPM_SOFT_RESET_SUCCESS,
PD_DPM_SOFT_RESET_REQUEST,
PD_DPM_SOFT_RESET_FAIL,
PD_DPM_STRUCTURED_VDM_REQUEST,
PD_DPM_STRUCTURED_VDM_SUCCESS,
PD_DPM_STRUCTURED_VDM_FAIL,
PD_DPM_UNSTRUCTURED_VDM_RECEIVED,
PD_DPM_SEND_UNSTRUCTURED_VDM_SUCCESS,
PD_DPM_SEND_UNSTRUCTURED_VDM_FAIL,
PD_DPM_GIVE_SRC_EXT_CAP,
PD_DPM_GET_SRC_EXT_CAP_SUCCESS,
PD_DPM_GET_SRC_EXT_CAP_FAIL,
PD_DPM_GIVE_STATUS,
PD_DPM_GET_STATUS_SUCCESS,
PD_DPM_GET_STATUS_FAIL,
PD_DPM_GIVE_PPS_STATUS,
PD_DPM_GET_PPS_STATUS_SUCCESS,

PD_DPM_VBUS_ALARM }
The callback events.

- enum pd_command_t {
  PD_DPM_INVALID = 0,
  PD_DPM_CONTROL_POWER_NEGOTIATION = 1,
  PD_DPM_CONTROL_REQUEST = 2,
  PD_DPM_CONTROL_GOTO_MIN = 3,
  PD_DPM_CONTROL_SOFT_RESET = 4,
  PD_DPM_CONTROL_HARD_RESET = 5,
  PD_DPM_CONTROL_PR_SWAP = 6,
  PD_DPM_CONTROL_DR_SWAP = 7,
  PD_DPM_CONTROL_VCONN_SWAP = 8,
  PD_DPM_CONTROL_GET_PARTNER_SOURCE_CAPABILITIES = 9,
  PD_DPM_CONTROL_GET_PARTNER_SINK_CAPABILITIES = 10,
  PD_DPM_GET_SRC_EXT_CAP = 11,
  PD_DPM_GET_STATUS = 12,
  PD_DPM_GET_BATTERY_CAP = 13,
  PD_DPM_GET_BATTERY_STATUS = 14,
  PD_DPM_GET_MANUFACTURER_INFO = 15,
  PD_DPM_FAST_ROLE_SWAP = 16,
  PD_DPM_GET_PPS_STATUS = 17,
  PD_DPM_ALERT = 18,
  PD_DPM_CONTROL_DISCOVERY_IDENTITY = 19,
  PD_DPM_CONTROL_DISCOVERY_SVIDS = 20,
  PD_DPM_CONTROL_DISCOVERY_MODES = 21,
  PD_DPM_CONTROL_ENTER_MODE = 22,
  PD_DPM_CONTROL_EXIT_MODE = 23,
  PD_DPM_CONTROL_SEND_ATTENTION = 24,
  PD_DPM_CONTROL_CABLE_RESET = 25,
  PD_DPM_SEND_VENDOR_STRUCTURED_VDM = 26,
  PD_DPM_SEND_UNSTRUCTURED_VDM = 27 }
  The command used in PD_Command.
- enum pd_control_t {

PD_CONTROL_GET_POWER_ROLE,
PD_CONTROL_GET_DATA_ROLE,
PD_CONTROL_GET_VCONN_ROLE,
PD_CONTROL_GET_TYPEC_CONNECT_STATE,
PD_CONTROL_GET_SNK_TYPEC_CURRENT_CAP,
PD_CONTROL_GET_TYPEC_CURRENT_VALUE,
PD_CONTROL_PHY_POWER_PIN,
PD_CONTROL_DISCHARGE_VBUS,
PD_CONTROL_SET_TYPEC_CURRENT_ILIM,
PD_CONTROL_VCONN,
PD_CONTROL_GET_TYPEC_ORIENTATION,
PD_CONTROL_INFORM_VBUS_VOLTAGE_RANGE,
PD_CONTROL_GET_PD_STATE,
PD_CONTROL_GET_CABLE_INFO,
PD_CONTROL_ALT_MODE,
PD_CONTROL_INFORM_EXTERNAL_POWER_STATE,
PD_CONTROL_ENTER_LOW_POWER,
PD_CONTROL_EXIT_LOW_POWER,
PD_CONTROL_GET_PD_LOW_POWER_STATE }
PD control code.

- enum pd_fr_swap_current_t {
  kFRSwap_NotSupported = 0,
  kFRSwap_CurrentDefaultUSB = 1,
  kFRSwap_Current15A = 2,
  kFRSwap_Current3A = 3 }
  fast role swap current in the fixed supply PDO.
- enum pd_pdo_type_t {
  kPDO_Fixed = 0u,
  kPDO_Battery = 1u,
  kPDO_Variable = 2u,
  kPDO_APDO = 3u }
  pdo type
- enum pd_apdo_type_t { kAPDO_PPS = 0u }
  apdo type
- enum pd_vbus_power_progress_t {
  kVbusPower_Invalid,
  kVbusPower_Stable,
  kVbusPower_InHardReset,
  kVbusPower_InPRSwap,
  kVbusPower_InFRSwap,
  kVbusPower_ChangeInProgress }
  vbus power change progress
- enum vbus_discharge_t {
  kVbus_NoDischarge = 0,
  kVbus_ApplyTypecDischarge,
  kVbus_TypecDischarge }

vbus power dischange progress

- enum pd_usb_communication_capable_as_host_t
  ID Header Field, used in the pd_id_header_vdo_t.
- enum pd_usb_communication_capable_as_device_t
  ID Header Field, used in the pd_id_header_vdo_t.
- enum pd_product_type_ufp_t
  ID Header Field, used in the pd_id_header_vdo_t.
- enum pd_product_type_cable_t
  ID Header Field, used in the pd_id_header_vdo_t.
- enum pd_modal_operation_t
  ID Header Field, used in the pd_id_header_vdo_t.
- enum pd_product_type_dfp_t
  ID Header Field, used in the pd_id_header_vdo_t.
- enum pd_connector_type_t
  ID Header Field or UFP VDO Field, used in the pd_id_header_vdo_t or pd_ufp_vdo_t.
- enum pd_device_capability_t
  UFP VDO Field, used in the pd_ufp_vdo_t.
- enum pd_vconn_power_t
  UFP VDO Field, used in the pd_ufp_vdo_t.
- enum pd_vconn_required_t
  UFP VDO Field, used in the pd_ufp_vdo_t.
- enum pd_vbus_required_t
  UFP VDO Field, used in the pd_ufp_vdo_t.
- enum pd_alternate_mode_t
  UFP VDO Field, used in the pd_ufp_vdo_t.
- enum pd_usb_highest_speed_t
  UFP VDO Field, used in the pd_ufp_vdo_t.
- enum usb_pd_auto_accept_value_t {
  kAutoRequestProcess_NotSupport = 0x00u,
  kAutoRequestProcess_Accept = 0x01u,
  kAutoRequestProcess_Reject = 0x02u,
  kAutoRequestProcess_Wait = 0x03u }
  pd auto policy accept or reject values.

Functions

- void PD_AltModeTask (void)
  PD stack Alternating Mode task function.
- pd_status_t PD_InstanceInit (pd_handle *pdHandle, pd_stack_callback_t callbackFn, pd_power_handle_callback_t *callbackFunctions, void *callbackParam, pd_instance_config_t *config)
  Initialize PD stack instance.
- pd_status_t PD_InstanceDeinit (pd_handle pdHandle)
  de-initialize PD stack instance.
- pd_status_t PD_Command (pd_handle pdHandle, uint32_t command, void *param)
  start up PD command.
- pd_status_t PD_Control (pd_handle pdHandle, uint32_t controlCode, void *param)
  Control PD stack and get info from PD stack.
- void PD_Task (void)
  PD stack's task for all instances.
- void PD_InstanceTask (pd_handle pdHandle)
  PD stack's instance task.

- void PD_PTN5110IsrFunction (pd_handle pdHandle)
  PD PTN5110 PHY ISR function.
- void PD_TimerIsrFunction (pd_handle pdHandle)
  PD stack timer function.

2.2 Data Structure Documentation

2.2.1 struct pd_alt_mode_module_t

Data Fields

- uint32_t SVID
  module's SVID
- const void * config
  module specific configure parameter, for example: pd_alt_mode_dp_config_t for displayport host or slave

2.2.2 struct pd_alt_mode_config_t

Data Fields

- const pd_alt_mode_module_t * modules
  module's list, the first module has high priority
- uint32_t moduleCount
  module count for modules
- uint32_t * identityData
  identity data buffer
- uint8_t identityObjectCount
  identity data size (unit is 4 bytes)
- uint8_t altModeRole
  typec_data_function_config_t values kDataConfig_UFP: support alt mode and support UFP.

2.2.2.0.0.1 Field Documentation

2.2.2.0.0.1.1 uint8_t pd_alt_mode_config_t::altModeRole

kDataConfig_DFP: support alt mode and support DFP. kDataConfig_DRD: support alt mode and support DFP and UFP. kDataConfig_None: don't support alt mode.

2.2.3 struct pd_alt_mode_control_t

Data Fields

- uint8_t controlCode
  pd_alt_mode_control_code_t values
- uint8_t altModeModuleIndex

conresponding to pd_alt_mode_config_t's modules' index; 0 - the common alternating mode function; 1 - the first moddule pd_alt_mode_config_t's modules; n - the n module in pd_alt_mode_config_t's modules.

- void * controlParam
  the control parameter

2.2.4 struct pd_dp_mode_obj_t

2.2.5 struct pd_dp_status_obj_t

2.2.6 struct pd_dp_configure_obj_t

2.2.7 struct pd_altmode_dp_modes_sel_t

Data Fields

- pd_dp_mode_obj_t * modes
  The all modes buffer.
- uint8_t modesCount
  The all modes count.
- uint8_t selectIndex
  application return this value to stack, which one application select 0 - all modes are not supported and select none.
- uint8_t selectPinAssign
  the pin assign selected by application.

2.2.7.0.0.2 Field Documentation

2.2.7.0.0.2.1 uint8_t pd_altmode_dp_modes_sel_t::selectIndex

1 - modesCount: the mode that is selected.

2.2.7.0.0.2.2 uint8_t pd_altmode_dp_modes_sel_t::selectPinAssign

it is based on the selected mode.

2.2.8 struct pd_dp_peripheral_interface_t

Data Fields

- pd_status_t(* dpPeripheralInit)(void **interfaceHandle, pd_handle pdHandle, const void *param)
  displayport peripheral init
- pd_status_t(* dpPeripheralDeinit)(void *interfaceHandle)
  displayport peripheral de-init
- pd_status_t(* dpPeripheralControl)(void *interfaceHandle, uint32_t opCode, void *opParam)
  displayport peripheral control

2.2.9 struct pd_alt_mode_dp_config_t

Data Fields

- const void * peripheralConfig
  customer can define this structure by self For example: hpd/crossbar driver can differ for different customers' solution.
- const pd_dp_peripheral_interface_t * peripheralInterface
  pd_dp_peripheral_interface_t
- uint32_t * modesList
  modes list
- uint32_t modesCount
  modes count
- pd_dp_status_obj_t dpStatusConfig
  DisplayPort status update config, used by slave.
- pd_dp_configure_obj_t dpConfigurationsConfig
  DisplayPort configurations config, used by host.

2.2.10 struct pd_power_port_config_t

Data Fields

- uint32_t * sourceCaps
  source caps, if don't support set as NULL
- uint32_t * sinkCaps
  sink caps, if don't support set as NULL
- uint8_t sourceCapCount
  source caps count, if don't support set as 0
- uint8_t sinkCapCount
  sink caps count, if don't support set as 0
- uint8_t typecRole
  device's typec role, for example: sink, source or DRP.
- uint8_t typecSrcCurrent
  It only is valid for source, this value set the TypeC Rp current level it's value is typec_current_val_t.
- uint8_t drpTryFunction
  (not supported yet) It only is valid for DRP, this value configure try-sink, try-source or no try function.
- uint8_t dataFunction
  It configure data role function of the device, it is DFP, UFP, DRD.
- uint8_t vconnSupported
  value is 1: support vconn swap and vconn power; value is 0: don't support vconn swap or vconn power.
- void * altModeConfig
  alt mode configuration, NULL means don't support Alt mode
- void * autoPolicyConfig
  PD auto policy configuration.
- void * reserved2Config
  (For feature extension) reserved

2.2.10.0.0.3 Field Documentation

2.2.10.0.0.3.1 uint8_t pd_power_port_config_t::typecRole

it's value is typec_power_role_config_t

2.2.10.0.0.3.2 uint8_t pd_power_port_config_t::drpTryFunction

it's value is typec_try_t

2.2.10.0.0.3.3 uint8_t pd_power_port_config_t::dataFunction

For USB function, DFP correspond to Host; UFP correspond to Device; DRD correspond to OTG. it's value is typec_data_function_config_t

2.2.11 struct pd_phy_config_t

Data Fields

- uint16_t i2cInstance
  I2C instance, the value is pd_phy_interface_t.
- uint16_t slaveAddress
  I2C slave address.
- uint32_t i2cSrcClock
  I2C clock frequency.
- void(∗ i2cReleaseBus)(void)
  I2C release bus function.
- uint8_t alertPort
  Alert Port.
- uint8_t alertPin
  Alert pin.
- uint8_t alertPriority
  Alert pin interrupt priority.

2.2.11.0.0.4 Field Documentation

2.2.11.0.0.4.1 uint16_t pd_phy_config_t::i2cInstance

2.2.11.0.0.4.2 void(∗ pd_phy_config_t::i2cReleaseBus)(void)

It will be called when I2C transmission fails

2.2.12 struct pd_instance_config_t

used in PD_InstanceInit function

Data Fields

- uint8_t deviceType
  The device type, for example: normal power port or cable, the value is pd_device_type_t.
- uint8_t phyType
  The PHY type, the value is pd_phy_type_t.
- void * phyConfig
  The PHY interface parameter, the value is pd_phy_config_t.
- void * portConfig
  The type is based on the deviceType value.

2.2.12.0.0.5 Field Documentation

2.2.12.0.0.5.1 void* pd_instance_config_t::portConfig

- kDeviceType_NormalPowerPort: pd_power_port_config_t
- kDeviceType_Cable: not supported yet.
- kDeviceType_AudioAccDevice: not supported yet.
- kDeviceType_DebugAccDevice: not supported yet.
- kDeviceType_AlternateModeProduct: not supported yet.

2.2.13 struct pd_source_fixed_pdo_t

Data Fields

- uint32_t maxCurrent: 10
  max current
- uint32_t voltage: 10
  voltage value, unit is 50mV
- uint32_t peakCurrent: 2
  peak current
- uint32_t reserved: 2
  reserved field
- uint32_t unchunkedSupported: 1
  unchunked is supported or not
- uint32_t dualRoleData: 1
  dual data role
- uint32_t usbCommunicationsCapable: 1
  usb communication capable or not
- uint32_t externalPowered: 1
  external powered
- uint32_t usbSuspendSupported: 1
  usb suspend supported or not
- uint32_t dualRolePower: 1
  dual power role
- uint32_t fixedSupply: 2
  pdo type

2.2.14 struct pd_sink_fixed_pdo_t

Data Fields

- uint32_t operateCurrent: 10
  operate current
- uint32_t voltage: 10
  voltage
- uint32_t reserved: 3
  reserved
- uint32_t frSwapRequiredCurrent: 2
  The value is pd_fr_swap_current_t value.
- uint32_t dualRoleData: 1
  dual data role
- uint32_t usbCommunicationsCapable: 1
  usb communication capable or not
- uint32_t externalPowered: 1
  external powered
- uint32_t higherCapability: 1
  higher capability
- uint32_t dualRolePower: 1
  dual power role
- uint32_t fixedSupply: 2
  pdo type

2.2.15 struct pd_source_variable_pdo_t

Data Fields

- uint32_t maxCurrent: 10
  max current
- uint32_t minVoltage: 10
  min voltage
- uint32_t maxVoltage: 10
  max voltage
- uint32_t variableSupply: 2
  pdo type

2.2.16 struct pd_source_apdo_pdo_t

Data Fields

- uint32_t maxCurrent: 7
  max current (50mA increments)
- uint32_t minVoltage: 8
  min voltage (100mV increments)

- uint32_t maxVoltage: 8
  max voltage (100mV increments)
- uint32_t APDOType: 2
  APDO type.
- uint32_t APDOSupply: 2
  pdo type

2.2.17 struct pd_sink_variable_pdo_t

Data Fields

- uint32_t operateCurrent: 10
  operate current
- uint32_t minVoltage: 10
  min voltage
- uint32_t maxVoltage: 10
  max voltage
- uint32_t variableSupply: 2
  pdo type

2.2.18 struct pd_source_battery_pdo_t

Data Fields

- uint32_t maxAllowPower: 10
  max power
- uint32_t minVoltage: 10
  min voltage
- uint32_t maxVoltage: 10
  max voltage
- uint32_t battery: 2
  pdo type

2.2.19 struct pd_sink_battery_pdo_t

Data Fields

- uint32_t operatePower: 10
  operate power
- uint32_t minVoltage: 10
  min voltage
- uint32_t maxVoltage: 10
  max voltage
- uint32_t battery: 2
  pdo type

2.2.20 struct pd_sink_apdo_pdo_t

Data Fields

- uint32_t maxCurrent: 7
  max current
- uint32_t minVoltage: 8
  min voltage (100mV increments)
- uint32_t maxVoltage: 8
  max voltage (100mV increments)
- uint32_t APDOType: 2
  APDO type.
- uint32_t APDOSupply: 2
  pdo type

2.2.21 struct pd_pdo_common_t

Data Fields

- uint32_t reserved: 30
  reserved
- uint32_t pdoType: 2
  pdo type, the value is pd_pdo_type_t

2.2.22 struct pd_source_pdo_t

2.2.23 struct pd_sink_pdo_t

2.2.24 struct pd_structured_vdm_header_t

2.2.25 struct pd_unstructured_vdm_header_t

2.2.26 struct pd_extended_msg_header_t

2.2.27 struct pd_msg_header_t

2.2.28 struct pd_svdm_command_request_t

It is used in PD_DPM_STRUCTURED_VDM_REQUEST event callback. It provide vdm message information to application, application need reply ACK (contain data), NAK or BUSY.

Data Fields

- uint32_t * vdoData
  vdm data buffer address
- pd_structured_vdm_header_t vdmHeader
  vdm header
- uint8_t vdoCount
  vdm data length (unit is 4 bytes)
- uint8_t vdoSop
  vdm message's sop type
- uint8_t requestResultStatus
  application need return the negotiation result to PD stack, the value is pd_command_result_t

2.2.29 struct pd_svdm_command_result_t

It is used in PD_DPM_STRUCTURED_VDM_SUCCESS and PD_DPM_STRUCTURED_VDM_FAIL events callback. It provide vdm command reply message information to application, it may be ACK (contain data), NAK or BUSY.

Data Fields

- uint32_t * vdoData
  vdm data buffer address
- pd_structured_vdm_header_t vdmHeader
  vdm header, even structured vdm don't have reply or fail, the header still have the sent message's SVID info
- uint16_t vdmSVID
  vdm's VID
- uint8_t vdoCount
  vdm data length (unit is 4 bytes)
- uint8_t vdoSop
  vdm message's sop type
- uint8_t vdmCommand
  vdm command, the value is pd_vdm_command_t
- uint8_t vdmCommandResult
  vdm command's result: success with data or fail.

2.2.29.0.0.6 Field Documentation

2.2.29.0.0.6.1 uint8_t pd_svdm_command_result_t::vdmCommandResult

The value is pd_command_result_t

2.2.30 struct pd_svdm_command_param_t

It is used in PD_Command for PD_DPM_CONTROL_DISCOVERY_IDENTITY, PD_DPM_CONTROL_DISCOVERY_SVIDS, PD_DPM_CONTROL_DISCOVERY_MODES, PD_DPM_CONTROL_ENTER_MODE, PD_DPM_CONTROL_EXIT_MODE or PD_DPM_CONTROL_SEND_ATTENTION. it provide vdm command information to PD stack, PD stack will start the command with the information.

Data Fields

- uint32_t * vdoData
  vdm data buffer address
- pd_structured_vdm_header_t vdmHeader
  vdm header
- uint8_t vdoCount
  vdm data length (unit is 4 bytes)
- uint8_t vdmSop
  vdm message's sop type
- uint8_t vendorVDMNeedResponse
  discovery_identity need response, but attention don't need.

2.2.30.0.0.7 Field Documentation

2.2.30.0.0.7.1 uint8_t pd_svdm_command_param_t::vendorVDMNeedResponse

Vendor defined structured VDM may need or not.

2.2.31 struct pd_unstructured_vdm_command_param_t

It is used in PD_Command for PD_DPM_SEND_UNSTRUCTURED_VDM and in callback for PD_DPM_UNSTRUCTURED_VDM_RECEIVED. For PD_DPM_SEND_UNSTRUCTURED_VDM: it provide vdm command information to PD stack, PD stack will start the command with the information. For PD_DPM_UNSTRUCTURED_VDM_RECEIVED: it provide information for the received unstructured vdm to APP.

Data Fields

- uint32_t * vdmHeaderAndVDOsData
  unstructured vdm message data
- uint8_t vdmSop
  message sop
- uint8_t vdmHeaderAndVDOsCount
  message length (unit is 4 bytes)
- uint8_t resultStatus
  command's result: success with data or fail.

2.2.31.0.0.8 Field Documentation

2.2.31.0.0.8.1 uint8_t pd_unstructured_vdm_command_param_t::resultStatus

The value is pd_command_result_t

2.2.32 struct pd_command_data_param_t

The callback events of src_ext_cap, status, battery_cap, battery_status, manufacturer_info and alert command will use this structure. it provide the command data.

Data Fields

- uint32_t dataLength
  data length, the max length is 260 (extended msg)
- uint8_t * dataBuffer
  data buffer
- uint8_t sop
  message sop
- uint8_t resultStatus
  command's result: success with data or fail.

2.2.32.0.0.9 Field Documentation

2.2.32.0.0.9.1 uint8_t pd_command_data_param_t::resultStatus

The value is pd_command_result_t

2.2.33 struct pd_rdo_t

It provides the request power information.

2.2.33.0.0.10 Field Documentation

2.2.33.0.0.10.1 uint32_t pd_rdo_t::maxOrMinOperateValue

- For fixed and variable request.
  - giveBack == 0: it is max operate current.
  - giveBack == 1: it is min operate current.
- For battery request.
  - giveBack == 0: it is max operate power.
  - giveBack == 1: it is min operate power.

2.2.34 struct pd_capabilities_t

Used in PD_DPM_GET_PARTNER_SRC_CAP_SUCCESS, PD_DPM_GET_PARTNER_SNK_CAP_-SUCCESS and PD_DPM_SNK_RECEIVE_PARTNER_SRC_CAP callback.

Data Fields

- uint32_t * capabilities
  capabilities buffer address
- uint8_t capabilitiesCount
  capabilities count (unit is 4 bytes)

2.2.35 struct pd_negotiate_power_request_t

used in PD_DPM_SRC_RDO_REQUEST callback.

Data Fields

- pd_rdo_t rdo
  request rdo
- uint8_t negotiateResult
  capabilities result, the value is pd_command_result_t

2.2.36 struct pd_bist_object_t

2.2.37 struct pd_ptn5110_ctrl_pin_t

2.2.38 struct pd_id_header_vdo_vdm10_t

2.2.39 struct pd_id_header_vdo_t

2.2.40 struct pd_passive_cable_vdo_vdm20_t

2.2.41 struct pd_passive_cable_vdo_vdm10_t

2.2.42 struct pd_active_cable_vdo_vdm20_t

2.2.43 struct pd_active_cable_vdo_vdm10_t

2.2.44 struct pd_cable_plug_info_t

2.2.45 struct pd_ama_vdo_t

2.2.46 struct pd_ufp_vdo_t

2.2.47 struct pd_power_handle_callback_t

The application needs implement this interface and pass to PD_InstanceInit

Data Fields

- pd_status_t(∗ PD_SrcTurnOnTypeCVbus)(void ∗callbackParam, uint8_t powerProgress)
  source provide default typec vbus power.
- pd_status_t(∗ PD_SrcTurnOnRequestVbus)(void ∗callbackParam, pd_rdo_t rdo)
  source provide RDO request vbus power.
- pd_status_t(∗ PD_SrcTurnOffVbus)(void ∗callbackParam, uint8_t powerProgress)
  source turn off vbus power.
- pd_status_t(∗ PD_SrcGotoMinReducePower)(void ∗callbackParam)
  source reduce power for goto min.
- pd_status_t(∗ PD_SnkDrawTypeCVbus)(void ∗callbackParam, uint8_t typecCurrentLevel, uint8_t powerProgress)
  sink can draw the default type-c vbus power.
- pd_status_t(∗ PD_SnkDrawRequestVbus)(void ∗callbackParam, pd_rdo_t rdo)
  sink can draw the request rdo vbus power.

- pd_status_t(∗ PD_SnkStopDrawVbus)(void ∗callbackParam, uint8_t powerProgress)
  sink stop draw vbus power.
- pd_status_t(∗ PD_SnkGotoMinReducePower)(void ∗callbackParam)
  sink reduce power for goto min.
- pd_status_t(∗ PD_ControlVconn)(void ∗callbackParam, uint8_t on)
  control vconn

2.2.47.0.0.11 Field Documentation

2.2.47.0.0.11.1 pd_status_t(∗ pd_power_handle_callback_t::PD_SrcTurnOnTypeCVbus)(void ∗callbackParam, uint8_t powerProgress)

Parameters

callbackParam	
powerProgress	hard_reset, pr_swap or fr_swap. Normally it is useless, just in case the application needs this info to provide power.

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.2 pd_status_t(∗ pd_power_handle_callback_t::PD_SrcTurnOnRequestVbus)(void ∗callbackParam, pd_rdo_t rdo)

Parameters

callbackParam	
rdo	request RDO from partner

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.3 pd_status_t(∗ pd_power_handle_callback_t::PD_SrcTurnOffVbus)(void ∗callbackParam, uint8_t powerProgress)

Parameters

callbackParam	
powerProgress	hard_reset, pr_swap or fr_swap. Normally it is useless, just in case the application needs this info to provide power.

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.4 pd_status_t(∗ pd_power_handle_callback_t::PD_SrcGotoMinReducePower)(void ∗callbackParam)

Parameters

callbackParam	

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.5 pd_status_t(∗ pd_power_handle_callback_t::PD_SnkDrawTypeCVbus)(void ∗callbackParam, uint8_t typecCurrentLevel, uint8_t powerProgress)

Parameters

callbackParam	
typecCurrent-Level	the value is typec_current_val_t
powerProgress	hard_reset, pr_swap or fr_swap. Normally it is useless, just in case the application needs this info to provide power.

Return values

kStatus_PD_Success	success

kStatus_PD_Error	error

2.2.47.0.0.11.6 pd_status_t(∗ pd_power_handle_callback_t::PD_SnkDrawRequestVbus)(void ∗callbackParam, pd_rdo_t rdo)

Parameters

callbackParam	
rdo	request RDO from partner

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.7 pd_status_t(∗ pd_power_handle_callback_t::PD_SnkStopDrawVbus)(void ∗callbackParam, uint8_t powerProgress)

Parameters

callbackParam	
powerProgress	hard_reset, pr_swap or fr_swap. Normally it is useless, just in case the application needs this info to provide power.

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.8 pd_status_t(∗ pd_power_handle_callback_t::PD_SnkGotoMinReducePower)(void ∗callbackParam)

Parameters

callbackParam	

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.47.0.0.11.9 pd_status_t(* pd_power_handle_callback_t::PD_ControlVconn)(void *callbackParam, uint8_t on)

Parameters

callbackParam	
on	0 - turn off vconn; 1 - turn on vconn.

Return values

kStatus_PD_Success	success
kStatus_PD_Error	error

2.2.48 struct pd_auto_policy_t

Data Fields

- uint32_t autoRequestPRSwapAsSource: 1
  0 - don't support; 1 - auto request pr_swap when current power role is source.
- uint32_t autoRequestPRSwapAsSink: 1
  0 - don't support; 1 - auto request pr_swap when current power role is sink.
- uint32_t autoAcceptPRSwapAsSource: 2
  accept swap or not when current role is source kAutoRequestProcess_Accept or kAutoRequestProcess_Reject
- uint32_t autoAcceptPRSwapAsSink: 2
  accept swap or not when current role is sink kAutoRequestProcess_Accept or kAutoRequestProcess_Reject
- uint32_t autoRequestDRSwap: 2
  pd_data_role_t values kPD_DataRoleUFP : auto request to UFP.
- uint32_t autoAcceptDRSwapToDFP: 2
  accept swap to DFP or not
- uint32_t autoAcceptDRSwapToUFP: 2
  accept swap to UFP or not
- uint32_t autoRequestVConnSwap: 2
  pd_vconn_role_t values kPD_NotVconnSource : auto request to turn off vconn.
- uint32_t autoAcceptVconnSwapToOn: 2
  accept swap to trun on Vconn or not
- uint32_t autoAcceptVconnSwapToOff: 2
  accept swap to trun off Vconn or not
- uint32_t autoSinkNegotiation: 1
  sink request the max power that statify self's sink caps; 1 - enable, 0 - don't enable
- uint32_t reserved: 13
  reserved bits

2.2.48.0.0.12 Field Documentation

2.2.48.0.0.12.1 uint32_t pd_auto_policy_t::autoRequestDRSwap

kPD_DataRoleDFP : auto request to DFP. kPD_DataRoleNone : don't support auto request.

2.2.48.0.0.12.2 uint32_t pd_auto_policy_t::autoRequestVConnSwap

kPD_IsVconnSource : auto request to turn on vconn. kPD_VconnNone : don't support auto request.

2.3 Typedef Documentation

2.3.1 typedef pd_status_t(∗ pd_stack_callback_t)(void ∗callbackParam, uint32_t event, void ∗param)

It is one parameter of PD_InstanceInit. PD stack notify application command flow and connect/disconnect state by this callback.

2.4 Enumeration Type Documentation

2.4.1 enum pd_alt_mode_control_code_t

Enumerator

kAltMode_TriggerEnterMode start do the enter mode steps
kAltMode_TriggerExitMode start do the exit mode steps
kAltMode_GetModeState get the enter mode state
kDPControl_HPDDetectEvent there is new HPD detect evetns (event value is #pd_hpd_detect_type_t)

2.4.2 enum pd_dp_hpd_driver_t

Enumerator

kDPHPDDriver_None invalid value
kDPHPDDriver_IRQ IRQ.
kDPHPDDriver_Low LOW.
kDPHPDDriver_High HIGH.
kDPHPDDriver_Waiting wait

2.4.3 enum pd_displayport_vdm_command_t

Enumerator

kDPVDM_StatusUpdate status update command

kDPVDM_Configure dp configure

2.4.4 enum pd_status_connected_val_t

Enumerator

kDFP_D_NonConnected Neither DFP_D nor UFP_D is connected, or Adaptor is disabled.
kDFP_D_Connected DFP_D is connected.
kUFP_D_Connected UFP_D is connected.
kUFP_D_BothConnected Both DFP_D and UFP_D are connected.

2.4.5 enum pd_configure_set_config_val_t

Enumerator

kDPConfig_USB Set configuration for USB.
kDPConfig_DFPD Set configuration for UFP_U as DFP_D.
kDPConfig_UFPD Set configuration for UFP_U as UFP_D.

2.4.6 enum dp_mode_port_cap_val_t

Enumerator

kDPPortCap_Reserved RESERVED.
kDPPortCap_UFPD UFP_D-capable (including Branch device.
kDPPortCap_DFPD DFP_D-capable (including Branch device)
kDPPortCap_Both Both DFP_D and UFP_D-capable.

2.4.7 enum dp_mode_signal_t

Enumerator

kDPSignal_Unspecified Signaling unspecified (used only when Select Configuration field is set for USB Configuration)
kDPSignal_DP Select DP v1.3 signaling rates and electrical settings.
kDPSignal_USBGEN2 Select Gen 2 signaling rates and electrical specifications.

2.4.8 enum dp_mode_pin_assign_val_t

Enumerator

kPinAssign_DeSelect De-select pin assignment.
kPinAssign_A GEN2_BR signal, 4 lanes.
kPinAssign_B GEN2_BR signal, 2 lanes.
kPinAssign_C DP_BR signal, 4 lanes.
kPinAssign_D DP_BR signal, 2 lanes.
kPinAssign_E DP_BR signal, 4 lanes.
kPinAssign_F Select Pin Assignment F.

2.4.9 enum dp_mode_receptacel_indication_t

Enumerator

kReceptacle_TypeCPlug DisplayPort interface is presented on a USB Type-C Plug.
kReceptacle_TypeCReceptacle DisplayPort interface is presented on a USB Type-C Receptacle.

2.4.10 enum dp_mode_usb20_signal_t

Enumerator

kUSB2_Required USB r2.0 signaling may be required on A6 C A7 or B6 C B7 while in DisplayPort Configuration.
kUSB2_NotRequired USB r2.0 signaling is not required on A6 C A7 or B6 C B7 while in DisplayPort Configuration.

2.4.11 enum pd_dp_peripheral_control_t

Enumerator

kDPPeripheal_ControlHPDValue control HPD
kDPPeripheal_ControlHPDSetLow set HPD low
kDPPeripheal_ControlHPDReleaseLow release HPD
kDPPeripheal_ControlSetMux set mux
kDPPeripheal_ControlSetMuxSaftMode set mux safe mode
kDPPeripheal_ControlSetMuxUSB3Only set mux usb3 only
kDPPeripheal_ControlSetMuxShutDown shut mux
kDPPeripheal_ControlSetMuxDisable disable mux
kDPPeripheal_ControlSetMuxDP4LANE set mux dp4lane
kDPPeripheal_ControlSetMuxDP2LANEUSB3 set mux dp2land and usb3

kDPPeripheal_ControlSetMuxDP2LANENOUSB set mux dp2lane no usb
kDPPeripheal_ControlHPDDetectStart start HPD detect
kDPPeripheal_ControlHPDDetectStop stop HPD detect
kDPPeripheal_ControlHPDQueueEnable enable HPD queue
kDPPeripheal_ControlHPDQueueDisable disable HPD queue
kDPPeripheal_ControlHPDGetCurrentState get HPD detect state value

2.4.12 enum pd_status_t

Enumerator

kStatus_PD_Error Failed.
kStatus_PD_Success Success.
kStatus_PD_Abort abort operation
kStatus_PD_Cancel cancel operation
kStatus_PD_Busy PD stack busy.

2.4.13 enum pd_phy_type_t

Enumerator

kPD_PhyPTN5110 PHY is PTN5110.
kPD_PhyPTN5100 PHY is PTN5100.

2.4.14 enum pd_device_type_t

Enumerator

kDeviceType_NormalPowerPort The device works as normal power port, for example: source, sink and DRP.
kDeviceType_Cable The device works as cable.
kDeviceType_AudioAccDevice The device works as audio accessory.
kDeviceType_DebugAccDevice The device works as debug accessory.
kDeviceType_AlternateModeProduct The device works as one type alternate mode, for example; displayport.

2.4.15 enum typec_power_role_config_t

Enumerator

kPowerConfig_SinkOnly only work as sink

kPowerConfig_SinkDefault work as default sink, but have ability to swap as source. only can connect with source

kPowerConfig_SourceOnly only work as source

kPowerConfig_SourceDefault work as default source, but have ability to swap as sink. only can connect with sink

kPowerConfig_DRPToggling DRP toggling, please reference to TypeC spec can connect with source or sink.

kPowerConfig_DRPSourcingDevice DRP, can work as USB device but cannot work as USB host, please reference to TypeC spec can connect with source or sink.

kPowerConfig_DRPSinkingHost DRP, can work as USB host but cannot work as USB device, please reference to TypeC spec can connect with source or sink.

2.4.16 enum typec_sink_role_config_t

configure sink typec role (normal, audio accessory or debug accessory), used in the pd_instance_config_t.

Enumerator

kSinkConfig_Normal normal sink add two Rd in two CC, the cable's one CC must be Ra or cut-off

kSinkConfig_AudioAcc audio accessory add two Ra in two CC, the cable's one CC must be Ra or line connected

kSinkConfig_DebugAcc debug accessory, add two Rd in two CC, the cable's two CC must be line connected.

2.4.17 enum typec_try_t

Enumerator

kTypecTry_None invalid value

kTypecTry_Src Try-SRC.

kTypecTry_Snk Try-SNK.

2.4.18 enum typec_data_function_config_t

Enumerator

kDataConfig_None invalid value

kDataConfig_DFP downstream facing port. If supporting USB, it correspond to USB host function

kDataConfig_UFP upstream facing port, If supporting USB, it correspond to USB device function

kDataConfig_DRD dual role datam, it support dr swap. If supporting USB, it can swap between Host and Device function.

2.4.19 enum pd_phy_interface_t

configure PHY's communication interface (I2C or SPI etc), used in the pd_instance_config_t.

Enumerator

kInterface_i2c0 i2c instance 0
kInterface_i2c1 i2c instance 1
kInterface_i2c2 i2c instance 2
kInterface_i2c3 i2c instance 3
kInterface_i2c4 i2c instance 4
kInterface_i2c5 i2c instance 5
kInterface_i2c6 i2c instance 6
kInterface_spi0 spi instance 0
kInterface_spi1 spi instance 1
kInterface_spi2 spi instance 2
kInterface_spi3 spi instance 3
kInterface_spi4 spi instance 4
kInterface_spi5 spi instance 5
kInterface_spi6 spi instance 6

2.4.20 enum start_of_packet_t

Enumerator

kPD_MsgSOP sop
kPD_MsgSOPp sop'
kPD_MsgSOPpp sop"
kPD_MsgSOPDbg sop debug
kPD_MsgSOPpDbg sop' debug
kPD_MsgSOPInvalid invalid value
kPD_MsgSOPMask sop mask for msg receive function
kPD_MsgSOPpMask sop' mask for msg receive function
kPD_MsgSOPppMask sop" mask for msg receive function
kPD_MsgSOPDbgMask sop debug mask for msg receive function
kPD_MsgSOPpDbgMask sop' debug mask for msg receive function

2.4.21 enum pd_command_result_t

For example: if partner reply reject for pr_swap, the error code kCommandResult_Reject will return to APP.

Enumerator

kCommandResult_None The default invalid value.
kCommandResult_Accept partner accept the command or the command success
kCommandResult_Success partner accept the command or the command success
kCommandResult_Reject partner reply reject for this command
kCommandResult_Wait partner reply wait for this command
kCommandResult_Error msg send error
kCommandResult_NotSupported partner reply Not_Supported for this command
kCommandResult_VDMACK partner reply ACK for this VDM command
kCommandResult_VDMNAK partner reply NAK for this VDM command
kCommandResult_VDMBUSY partner reply BUSY for this VDM command
kCommandResult_Timeout msg send time out without msg received

2.4.22 enum typec_current_val_t

Enumerator

kCurrent_Invalid invalid value
kCurrent_StdUSB standard USB current (900mA)
kCurrent_1A5 1.5A
kCurrent_3A 3.0A

2.4.23 enum pd_power_role_t

Enumerator

kPD_PowerRoleSink sink
kPD_PowerRoleSource source
kPD_PowerRoleNone invalid value

2.4.24 enum pd_data_role_t

Enumerator

kPD_DataRoleUFP UFP.
kPD_DataRoleDFP DFP.
kPD_DataRoleNone invalid value

2.4.25 enum pd_vconn_role_t

Enumerator

kPD_NotVconnSource is vconn source
kPD_IsVconnSource is not vconn source
kPD_VconnNone invalid value

2.4.26 enum typec_port_connect_state_t

Enumerator

kTYPEC_ConnectNone invalid value, or not connected
kTYPEC_ConnectSource Self is normal source port.
kTYPEC_ConnectSink Self is normal sink port.
kTYPEC_ConnectPoweredCable If typec connection role is this value, there are two cases:
- Self is source port, the connected cable is active.
- Self is active cable.

kTYPEC_ConnectPoweredCableWithSink If typec connection role is this value, there are three cases:
- Self is source port, the connected partner is normal sink device and the cable is active cable;
- Self is source port, the connected partner is vconn powered accessory;
- Self is vconn powered accessory.

kTYPEC_ConnectVconnPoweredAccessory Same as kTYPEC_ConnectPoweredCableWithSink.
kTYPEC_ConnectAudioAccessory If typec connection role is this value, there are two cases:
- Self is source port, the connected partner is audio accessory device;
- Self is audio accessory.

kTYPEC_ConnectDebugAccessory If typec connection role is this value, there are two cases:
- Self is source port, the connected partner is debug accessory device;
- Self is debug accessory.

2.4.27 enum pd_vdm_command_t

Enumerator

kVDM_DiscoverIdentity discovery identity
kVDM_DiscoverSVIDs discovery SVIDs
kVDM_DiscoverModes discovery Modes
kVDM_EnterMode enter mode
kVDM_ExitMode exit mode
kVDM_Attention attention

2.4.28 enum pd_vdm_command_type_t

Enumerator

kVDM_Initiator initiator
kVDM_ResponderACK ACK.
kVDM_ResponderNAK NAK.
kVDM_ResponderBUSY BUSY.
kVDM_ComandTypeInvalid Invalid code.

2.4.29 enum pd_dpm_callback_event_t

There are two types events here:

- command flow events. For example: PD_DPM_SNK_HARD_RESET_REQUEST and PD_DPM-_SRC_HARD_RESET_REQUEST are for hard reset command.
- device state callback events. For example: PD_CONNECTED indicate connection

Enumerator

PD_DPM_SNK_HARD_RESET_REQUEST hard reset sent or received. (only sink receive this event)
PD_DPM_SRC_HARD_RESET_REQUEST hard reset sent or received. (only source receive this event)
PD_DPM_PR_SWAP_REQUEST pr swap request, need return accept or reject negotiation result
PD_DPM_PR_SWAP_SUCCESS pr swap success, indicate power role is changed
PD_DPM_PR_SWAP_FAIL pr swap fail because reject, error etc
PD_DPM_FR_SWAP_REQUEST fast role swap request, need return accept or reject negotiation result
PD_DPM_FR_SWAP_SUCCESS fast role swap success, indicate power role is changed
PD_DPM_FR_SWAP_FAIL fast role swap fail because reject, error etc
PD_DPM_SRC_RDO_REQUEST RDO request, need return accept or reject negotiation result (only source receive this event)
PD_DPM_SRC_CONTRACT_STILL_VALID check the prev contract valid or not (only source receive this event)
PD_DPM_SRC_SEND_SRC_CAP_FAIL send source cap fail (only source receive this event)
PD_DPM_SRC_RDO_SUCCESS RDO request command success (only source receive this event)
PD_DPM_SRC_RDO_FAIL RDO request fail (only source receive this event)
PD_DPM_SNK_RECEIVE_PARTNER_SRC_CAP sink receive partner's source capabilities (only sink receive this event)
PD_DPM_SNK_GET_RDO sink get request RDO from application (only sink receive this event)
PD_DPM_SNK_RDO_SUCCESS sink RDO request command success, can use the requested power (only sink receive this event)
PD_DPM_SNK_RDO_FAIL sink RDO request fail (only sink receive this event)

PD_DPM_GET_PARTNER_SRC_CAP_FAIL get partner's source capabilities fail, partner don't support or message transfer fail

PD_DPM_GET_PARTNER_SRC_CAP_SUCCESS receive partner's source capabilities

PD_DPM_SRC_GOTOMIN_SUCCESS source goto min command success (only source receive this event)

PD_DPM_SNK_GOTOMIN_SUCCESS sink goto min command success (only sink receive this event)

PD_DPM_SRC_GOTOMIN_FAIL source goto min command fail, message transfer fail (only source receive this event)

PD_DPM_SNK_GOTOMIN_FAIL sink goto min command success, message transfer fail (only sink receive this event)

PD_DPM_GET_PARTNER_SNK_CAP_SUCCESS receive partner's sink capabilities

PD_DPM_GET_PARTNER_SNK_CAP_FAIL get partner's sink capabilities fail, partner don't support or message transfer fail

PD_DPM_DR_SWAP_REQUEST data role swap request, need return accept or reject negotiation result

PD_DPM_DR_SWAP_SUCCESS data role swap success, indicate data role is changed

PD_DPM_DR_SWAP_FAIL data role swap fail because reject, error etc

PD_DPM_VCONN_SWAP_REQUEST vconn swap request, need return accept or reject negotiation result

PD_DPM_VCONN_SWAP_SUCCESS vconn swap success, indicate vconn role is changed

PD_DPM_VCONN_SWAP_FAIL data role swap fail because reject, error etc

PD_DPM_SOFT_RESET_SUCCESS If application call PD_Command to send soft_reset, this event indicate the command success.

PD_DPM_SOFT_RESET_REQUEST receive soft reset or PD stack send soft reset actively because stack error and send success

PD_DPM_SOFT_RESET_FAIL If application call PD_Command to send soft_reset, this event indicate the command fail.

PD_DPM_STRUCTURED_VDM_REQUEST structured VDM command received, application need return reply in this event

PD_DPM_STRUCTURED_VDM_SUCCESS structured VDM command success, and receive reply or don't need reply

PD_DPM_STRUCTURED_VDM_FAIL structured VDM command fail, reply with NAK or BUSY or doesn't receive reply

PD_DPM_UNSTRUCTURED_VDM_RECEIVED unstructured VDM message received

PD_DPM_SEND_UNSTRUCTURED_VDM_SUCCESS send unstructured VDM message successfully

PD_DPM_SEND_UNSTRUCTURED_VDM_FAIL send unstructured VDM message fail

PD_DPM_GIVE_SRC_EXT_CAP receive partner's request for source extended capabilities, needs return capabilities data or not supported

PD_DPM_GET_SRC_EXT_CAP_SUCCESS receive partner's reply data for request

PD_DPM_GET_SRC_EXT_CAP_FAIL request fail because receiving not supported or message error

PD_DPM_GIVE_STATUS receive partner's request for status, need return status data or not supported

PD_DPM_GET_STATUS_SUCCESS receive partner's reply data for request
PD_DPM_GET_STATUS_FAIL request fail because receiving not supported or message error
PD_DPM_GIVE_PPS_STATUS receive sink's request for pps status, source needs return pps status message
PD_DPM_GET_PPS_STATUS_SUCCESS receive source's reply data for request
PD_DPM_GET_PPS_STATUS_FAIL request fail because receiving not supported or message error
PD_DPM_GIVE_BATTERY_CAP receive partner's request for battery cap, needs return cap data or not supported
PD_DPM_GET_BATTERY_CAP_SUCCESS receive partner's reply data for request
PD_DPM_GET_BATTERY_CAP_FAIL request fail because receiving not supported or message error
PD_DPM_GIVE_BATTERY_STATUS receive partner's request for battery status, needs return status data or not supported
PD_DPM_GET_BATTERY_STATUS_SUCCESS receive partner's reply data for request
PD_DPM_GET_BATTERY_STATUS_FAIL request fail because receiving not supported or message error
PD_DPM_GIVE_MANUFACTURER_INFO receive partner's request for manufacturer info, needs return info data or not supported
PD_DPM_GET_MANUFACTURER_INFO_SUCCESS receive partner's reply data for request
PD_DPM_GET_MANUFACTURER_INFO_FAIL request fail because receiving not supported or message error
PD_DPM_ALERT_RECEIVED receive partner's alert message
PD_DPM_SEND_ALERT_SUCCESS send alert message success
PD_DPM_SEND_ALERT_FAIL send alert message fail
PD_DPM_CABLE_RESET_REQUEST receive cable reset
PD_DPM_CABLE_RESET_COMPLETE DFP send cable reset successfully.
PD_DPM_ALTMODE_DP_DFP_SELECT_MODE_AND_PINASSIGN application get modes and select one mode and pin assign in this callback When PD_CONFIG_ALT_MODE_DP_AUTO_SELECT_MODE is disable, this callback event is valid.
PD_DPM_ALTMODE_DP_DFP_MODE_CONFIGURED dp alt mode has enter and cnfigured
PD_DPM_ALTMODE_DP_DFP_MODE_UNCONFIGURED dp alt mode has enter and cnfigured
PD_DPM_ALTMODE_DP_UFP_MODE_CONFIGURED dp alt mode has enter and cnfigured
PD_FUNCTION_DISABLED PD stack enter the disabled state, needs application restart.
PD_CONNECTED partner is connected
PD_CONNECT_ROLE_CHANGE connect role change, for example Try.SRC
PD_DISCONNECTED partner is disconnected
PD_ALTERNATE_MODE_ENTER_TIME_OUT the product is alternate mode device, and time out before enter alternate mode as typec spec define
PD_DPM_OVP_OCP_FAULT there are OCP or OVP fault, return current fault status register value.
PD_DPM_GET_EXTERNAL_POWER_STATE get external power state
PD_DPM_VBUS_ALARM vbus voltage alarm high or low asserted, needs to take care of this. parameter 0 represents low voltage alarm, parameter 1 represents high voltage alarm.

2.4.30 enum pd_command_t

The are related to PD3.0 stack's AMS. The command's result is return by the callback events.

Enumerator

PD_DPM_INVALID invalid command
PD_DPM_CONTROL_POWER_NEGOTIATION power negotation, it is only used in source when source power change
PD_DPM_CONTROL_REQUEST RDO request, it is only used in sink.
PD_DPM_CONTROL_GOTO_MIN goto min request, it is only used in source
PD_DPM_CONTROL_SOFT_RESET application can send soft_reset actively
PD_DPM_CONTROL_HARD_RESET application can send hard_reset actively
PD_DPM_CONTROL_PR_SWAP power role swap
PD_DPM_CONTROL_DR_SWAP data role swap
PD_DPM_CONTROL_VCONN_SWAP vconn role swap
PD_DPM_CONTROL_GET_PARTNER_SOURCE_CAPABILITIES get partner source capabilities
PD_DPM_CONTROL_GET_PARTNER_SINK_CAPABILITIES get partner sink capabilities
PD_DPM_GET_SRC_EXT_CAP get partner source extended capabilities
PD_DPM_GET_STATUS get partner status
PD_DPM_GET_BATTERY_CAP get partner battery cap
PD_DPM_GET_BATTERY_STATUS get partner battery status
PD_DPM_GET_MANUFACTURER_INFO get partner manufacturer info
PD_DPM_FAST_ROLE_SWAP fast role swap
PD_DPM_GET_PPS_STATUS get partner source pps status
PD_DPM_ALERT alert
PD_DPM_CONTROL_DISCOVERY_IDENTITY discovery identity
PD_DPM_CONTROL_DISCOVERY_SVIDS discovery SVIDs
PD_DPM_CONTROL_DISCOVERY_MODES discovery Modes
PD_DPM_CONTROL_ENTER_MODE enter mode
PD_DPM_CONTROL_EXIT_MODE exit mode
PD_DPM_CONTROL_SEND_ATTENTION attention
PD_DPM_CONTROL_CABLE_RESET cable reset
PD_DPM_SEND_VENDOR_STRUCTURED_VDM send vendor defined structured vdm
PD_DPM_SEND_UNSTRUCTURED_VDM send standard/vendor unstructured vdm

2.4.31 enum pd_control_t

PD_Control implement these control functions.

Enumerator

PD_CONTROL_GET_POWER_ROLE get the power role, return value is pd_power_role_t

PD_CONTROL_GET_DATA_ROLE get data role, return value is pd_data_role_t
PD_CONTROL_GET_VCONN_ROLE get vconn role, return value is pd_vconn_role_t
PD_CONTROL_GET_TYPEC_CONNECT_STATE get typec connection role, return value is typec_port_connect_state_t
PD_CONTROL_GET_SNK_TYPEC_CURRENT_CAP get typec current level, return value is typec_current_val_t. Deprecated!!! Please replace PD_CONTROL_GET_SNK_TYPEC_CURRENT_CAP with PD_CONTROL_GET_TYPEC_CURRENT_VALUE.
PD_CONTROL_GET_TYPEC_CURRENT_VALUE get Type-C Rp current value, return value is typec_current_val_t
PD_CONTROL_PHY_POWER_PIN control phy switches for vbus power, if PHY has this function this control code is valid
PD_CONTROL_DISCHARGE_VBUS discharge vbus, if PHY has this function this control code is valid
PD_CONTROL_SET_TYPEC_CURRENT_ILIM control vbus to set the type-c 5V vbus current ilim (OCP current)
PD_CONTROL_VCONN control vconn, if PHY has this function this control code is valid
PD_CONTROL_GET_TYPEC_ORIENTATION get the typec orientation info
PD_CONTROL_INFORM_VBUS_VOLTAGE_RANGE inform vbus, let the PHY knows the actual vbus voltage, voltage unit is 1mV
PD_CONTROL_GET_PD_STATE #pd_stack_state_t
PD_CONTROL_GET_CABLE_INFO get cable information
PD_CONTROL_ALT_MODE alt mode related control
PD_CONTROL_INFORM_EXTERNAL_POWER_STATE external power state
PD_CONTROL_ENTER_LOW_POWER enter low power
PD_CONTROL_EXIT_LOW_POWER exit low power
PD_CONTROL_GET_PD_LOW_POWER_STATE get pd low power state

2.4.32 enum pd_fr_swap_current_t

Enumerator

kFRSwap_NotSupported fast role swap not supported
kFRSwap_CurrentDefaultUSB default usb current
kFRSwap_Current15A 1.5A@5V
kFRSwap_Current3A 3.0A@5V

2.4.33 enum pd_pdo_type_t

Enumerator

kPDO_Fixed Fixed pdo.
kPDO_Battery Battery pdo.
kPDO_Variable Variable pdo.

kPDO_APDO Augmented Power Data Object pdo.

2.4.34 enum pd_apdo_type_t

Enumerator

kAPDO_PPS Programmable Power Supply.

2.4.35 enum pd_vbus_power_progress_t

Enumerator

kVbusPower_Invalid invalid value
kVbusPower_Stable vbus is stable
kVbusPower_InHardReset vubs is changing in hard reset
kVbusPower_InPRSwap vbus is changing in power role swap
kVbusPower_InFRSwap vbus is changing in fast role swap
kVbusPower_ChangeInProgress vbus is changing positive or negative

2.4.36 enum vbus_discharge_t

Enumerator

kVbus_NoDischarge vbus is not discharging.
kVbus_ApplyTypecDischarge vbus needs to discharge.
kVbus_TypecDischarge vbus is discharging.

2.4.37 enum usb_pd_auto_accept_value_t

Enumerator

kAutoRequestProcess_NotSupport don't support
kAutoRequestProcess_Accept auto accept request
kAutoRequestProcess_Reject auto reject request
kAutoRequestProcess_Wait auto reply wait for request

2.5 Function Documentation

2.5.1 void PD_AltModeTask (void)

User need keep calling this function endlessly in the PD application using one task.

2.5.2 pd_status_t PD_InstanceInit (pd_handle * *pdHandle,* pd_stack_callback_t *callbackFn,* pd_power_handle_callback_t * *callbackFunctions,* void * *callbackParam,* pd_instance_config_t * *config*)

This function initializes the PD stack module specified by the parameter, it will associate with the PD PHY that is specified by parameter.

Parameters

out	*pdHandle*	Returns the PD stack instance handle, other API will use this handle as parameter;
in	*callbackFn*	PD stack callback function, it will notify the connect/disconnect and PD command's process flow and result;
in	*callbackParam*	the callbackFn's parameter.
in	*config*	The PD instance configuration table, see the struct pd_instance_config_t

Return values

kStatus_USB_Success	initialization success.
other	value error code.

2.5.3 pd_status_t PD_InstanceDeinit (pd_handle *pdHandle*)

This function de-initializes the PD stack module specified by the pdHandle.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.

Return values

kStatus_USB_Success	success.
other	value error code.

2.5.4 pd_status_t PD_Command (pd_handle *pdHandle,* uint32_t *command,* void * *param*)

This function trigger the AMS command functions that are defined in the Section 8.3.2 in PD3.0 spec, you can see the AMS summary in the Table 8-4. This function only start the command, the command need communicate the partner, so the command flow is asynchronous. The command process flow and result are notified by the callback function that is one parameter in the PD_InstanceInit API.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.
in	*command*	the AMS command enumeration, see the pd_command_t.
in	*param*	the command's parameter.

Return values

kStatus_USB_Success	command start success.
other	value error code.

2.5.5 pd_status_t PD_Control (pd_handle *pdHandle,* uint32_t *controlCode,* void * *param*)

This function is blocking and synchronous, it implement some controls that will used in the PD application.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.
in	*controlCode*	see the pd_control_t
in	*param*	the param is different for different control.

Return values

kStatus_USB_Success	function success.
other	value error code.

2.5.6 void PD_Task (void)

User need keep calling this function endlessly in the PD application using one task.

2.5.7 void PD_InstanceTask (pd_handle *pdHandle*)

User need keep calling this function endlessly in the PD application using one task.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.

2.5.8 void PD_PTN5110IsrFunction (pd_handle *pdHandle*)

User need call this function in the PHY GPIO ISR.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.

2.5.9 void PD_TimerIsrFunction (pd_handle *pdHandle*)

User need call this function in the 1ms Timer ISR.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.

Chapter 3
USB PD PHY driver interface

3.1 Overview

The USB Type-C PD stack provide the same PHY interface for different PHY. The PHY driver needs implement the interface functions. The same USB Type-C PD stack can adapter to different PHY.

Modules

- USB PD I2C driver wrapper
- USB PD PHY PTN5110 driver

Data Structures

- struct pd_phy_rx_result_t
  PD PHY receive result information. More...
- struct pd_detach_detection_param_t
  detect detach parameter used in the PD_PHY_CONFIG_DETACH_DETECTION control. More...
- struct pd_attach_detection_param_t
  detect attach parameter used in the PD_PHY_CONFIG_ATTACH_DETECTION control. More...
- struct pd_phy_vendor_info_t
  PHY's vendor information. More...
- struct pd_phy_msg_header_info_t
  configure PHY's data role, power role or goodCRC's msg header. More...
- struct pd_phy_get_cc_state_t
  Get PHY's CC line state. More...
- struct pd_phy_api_interface_t
  PD PHY interface table structure. More...

Macros

- #define PD_VBUS_POWER_STATE_VSAFE5V_MASK (0x01u)
  In the return value of PHY control (PD_PHY_GET_VBUS_POWER_STATE), this bit set means vbus is vsafe5v.
- #define PD_VBUS_POWER_STATE_VSAFE0V_MASK (0x02u)
  In the return value of PHY control (PD_PHY_GET_VBUS_POWER_STATE), this bit set means vbus is vsafe0v.
- #define PD_VBUS_POWER_STATE_VBUS_MASK (0x04u)
  In the return value of PHY control (PD_PHY_GET_VBUS_POWER_STATE), this bit set means vbus exist.
- #define PD_VBUS_POWER_STATE_VSYS_MASK (0x08u)
  In the return value of PHY control (PD_PHY_GET_VBUS_POWER_STATE), this bit set means phy vsys exist.
- #define PD_VBUS_POWER_STATE_SINK_DISCONNECT (0x10u)
  In the return value of PHY control (PD_PHY_GET_VBUS_POWER_STATE), this bit set means vbus is above vsinkDisconnect.

Enumerations

- enum pd_cc_type_t {
  kPD_CCInvalid,
  kPD_CC1,
  kPD_CC2 }
  The CC type.
- enum pd_phy_cc_state_t {
  kCCState_SrcOpen,
  kCCState_SrcRd,
  kCCState_SrcRa,
  kCCState_SnkOpen,
  kCCState_SnkRp,
  kCCState_SnkRpDefault,
  kCCState_SnkRp1_5,
  kCCState_SnkRp3_0,
  kCCState_Unknown,
  kCCState_Unstable }
  The CC state value.
- enum pd_phy_control_t {

PD_PHY_UPDATE_STATE,
PD_PHY_CONTROL_ALERT_INTERRUPT,
PD_PHY_ENTER_LOW_POWER_STATE,
PD_PHY_EXIT_LOW_POWER_STATE,
PD_PHY_GET_PHY_VENDOR_INFO,
PD_PHY_CONTROL_POWER_PIN,
PD_PHY_CONTROL_VBUS_DETECT,
PD_PHY_SET_VBUS_TRANSFORM_STATE,
PD_PHY_DISCHARGE_VBUS,
PD_PHY_CONTROL_VBUS_AUTO_DISCHARGE,
PD_PHY_CONTROL_VBUS_ALARM,
PD_PHY_SET_TYPEC_VBUS_ILIM,
PD_PHY_GET_VBUS_POWER_STATE,
PD_PHY_FR_SWAP_CHECK_VBUS_APPLIED,
PD_PHY_INFORM_VBUS_VOLTAGE_RANGE,
PD_PHY_CONTROL_VCONN,
PD_PHY_DISCHARGE_VCONN,
PD_PHY_CONFIG_ATTACH_DETECTION,
PD_PHY_CONFIG_DETACH_DETECTION,
PD_PHY_RESET_CONNECT_DETECTION,
PD_PHY_SIGNAL_FR_SWAP,
PD_PHY_CONTROL_FR_SWAP,
PD_PHY_SRC_SET_TYPEC_CURRENT_CAP,
PD_PHY_GET_TYPEC_CURRENT_CAP,
PD_PHY_GET_CC_LINE_STATE,
PD_PHY_GET_LOOK4_CONNECTION_STATE,
PD_PHY_CONNECT_SET_CC_ORIENTATION,
PD_PHY_RESET_MSG_FUNCTION,
PD_PHY_DISABLE_MSG_RX,
PD_PHY_DISABLE_MSG_TX,
PD_PHY_CANCEL_MSG_TX,
PD_PHY_CANCEL_MSG_RX,
PD_PHY_SEND_HARD_RESET,
PD_PHY_SEND_CABLE_RESET,
PD_PHY_SET_MSG_HEADER_INFO,
PD_PHY_RESET_BIST,
PD_PHY_ENTER_BIST,
PD_PHY_EXIT_BIST,
PD_PHY_GET_BIST_MODE,
PD_PHY_GET_BIST_ERR_COUNT,
PD_PHY_SET_VBUS_SINK_DISCONNECT }

PD PHY interface table's control function implement these control codes.

- enum pd_phy_notify_event_t {

PD_PHY_EVENT_STATE_CHANGE,
PD_PHY_EVENT_SEND_COMPLETE,
PD_PHY_EVENT_RECEIVE_COMPLETE,
PD_PHY_EVENT_HARD_RESET_RECEIVED,
PD_PHY_EVENT_VBUS_STATE_CHANGE,
PD_PHY_EVENT_VBUS_SINK_DISCONNECT,
PD_PHY_EVENT_FR_SWAP_SINGAL_RECEIVED,
PD_PHY_EVENT_VBUS_ALARM,
PD_PHY_EVENT_REQUEST_STACK_RESET,
PD_PHY_EVENT_VCONN_PROTECTION_FAULT,
PD_PHY_EVENT_TYPEC_OVP_OCP_FAULT,
PD_PHY_EVENT_FAULT_RECOVERY }
PD PHY driver notify PD stack the PHY state through these events.

- enum pd_bist_mst_t
  BIST message type (not supported yet)
- enum pd_phy_look4connect_state_t { kLook4ConnState_Looking = 0 }
  Get DRP is looking for connection state.

Functions

- void PD_Notify (pd_handle pdHandle, pd_phy_notify_event_t event, void ∗param)
  Notify PD stack the PHY status.

3.2 Data Structure Documentation

3.2.1 struct pd_phy_rx_result_t

PD_PHY_EVENT_RECEIVE_COMPLETE event use this structure as parameter.

Data Fields

- pd_status_t rxResultStatus
  receive result
- start_of_packet_t rxSop
  message's sop
- uint16_t rxLength
  message's length

3.2.2 struct pd_detach_detection_param_t

Data Fields

- uint8_t powerRole
  power role
- uint8_t usedCC

used CC line after attach
- uint8_t typecConnectState
  the typec connect state, the value is typec_port_connect_state_t
- uint8_t srcRpCurrent
  Rp value (Typec current level)
- uint8_t snkDetachDetectCCOpen
  if Ture: sink detect detach through CC open; if False: sink detect detach through Vbus
- uint8_t debugUnoriented
  (not supported yet) debug accessory info
- uint8_t debugDTS
  (not supported yet) debug accessory info

3.2.3 struct pd_attach_detection_param_t

Data Fields

- uint8_t isDRP
  if True: DRP; if False: normal source, sink or accessory
- uint8_t powerRole
- uint8_t deviceType
  device type, it's value is pd_device_type_t.
- uint8_t srcRpCurrent
  if it is source, it configure Rp current level

3.2.3.0.0.13 Field Documentation

3.2.3.0.0.13.1 uint8_t pd_attach_detection_param_t::powerRole

- isDRP is True:
  - source: the initial power role is source and toggling between source with sink.
  - sink: the initial power role is sink and toggling between source with sink.
- isDRP is False:
  - source: it is source role.
  - sink: it is sink role or accessory.

3.2.4 struct pd_phy_vendor_info_t

Data Fields

- uint16_t vendorID
  vendor ID
- uint16_t productID
  product ID
- uint16_t deviceID
  device ID

3.2.5 struct pd_phy_msg_header_info_t

Used as the PD_PHY_SET_MSG_HEADER_INFO control code parameter.

Data Fields

- uint8_t dataRole
  data role
- uint8_t powerRole
  power role
- uint8_t cablePlug
  self is cable plug
- uint8_t revision
  spec revision

3.2.6 struct pd_phy_get_cc_state_t

Used as the PD_PHY_GET_CC_LINE_STATE control code parameter.

Data Fields

- pd_phy_cc_state_t cc1State
  CC1 line state, the value is pd_phy_cc_state_t.
- pd_phy_cc_state_t cc2State
  CC2 line state, the value is pd_phy_cc_state_t.

3.2.7 struct pd_phy_api_interface_t

Data Fields

- pd_status_t(∗ pdPhyInit)(pd_handle pdHandle)
  Initializes PD PHY module.
- pd_status_t(∗ pdPhyDeinit)(pd_handle pdHandle)
  de-initialize phy instance
- pd_status_t(∗ pdPhySend)(pd_handle pdHandle, uint8_t startOfPacket, uint8_t ∗buffer, uint32_t length)
  send message
- pd_status_t(∗ pdPhyReceive)(pd_handle pdHandle, uint8_t startOfPacketMask, uint8_t ∗buffer)
  receive message
- pd_status_t(∗ pdPhyControl)(pd_handle pdHandle, pd_phy_control_t control, void ∗param)
  control phy

3.2.7.0.0.14 Field Documentation

3.2.7.0.0.14.1 pd_status_t(∗ pd_phy_api_interface_t::pdPhyInit)(pd_handle pdHandle)

This function initializes one PD PHY module instance specified by the parameters.

Parameters

in	*pdHandle*	rPD instance handle.

Return values

kStatus_USB_Success	initialization success.
other	value error code.

3.2.7.0.0.14.2 pd_status_t(∗ pd_phy_api_interface_t::pdPhyDeinit)(pd_handle pdHandle)

This function de-initializes the PD PHY module specified by the pdHandle.

Parameters

in	*pdHandle*	PD phy instance handle.

Return values

kStatus_USB_Success	initialization success.
other	value error code.

3.2.7.0.0.14.3 pd_status_t(∗ pd_phy_api_interface_t::pdPhySend)(pd_handle pdHandle, uint8_t startOfPacket, uint8_t ∗buffer, uint32_t length)

This function send one message, this function is asynchronous, the message send result return to PD stack by the PD_Notify(PD_PHY_EVENT_SEND_COMPLETE);

Parameters

in	*pdHandle*	PD phy instance handle.
in	*startOfPacket*	message's sop
in	*buffer*	data buffer.
in	*length*	data length.

Return values

kStatus_USB_Success	initialization success.

other	value error code.

3.2.7.0.0.14.4 pd_status_t(∗ pd_phy_api_interface_t::pdPhyReceive)(pd_handle pdHandle, uint8_t startOfPacketMask, uint8_t ∗buffer)

This function pend to receive message, this function is asynchronous, the message receive result return to PD stack by the PD_Notify(PD_PHY_EVENT_RECEIVE_COMPLETE);

Parameters

in	*pdHandle*	PD phy instance handle.
in	*startOfPacket-Mask*	message's SOPs that will receive.
in	*buffer*	data buffer.
in	*length*	data length.

Return values

kStatus_USB_Success	initialization success.
other	value error code.

3.2.7.0.0.14.5 pd_status_t(∗ pd_phy_api_interface_t::pdPhyControl)(pd_handle pdHandle, pd_phy_control_t control, void ∗param)

This function control PHY operatin. The control codes are defined at enumeration pd_phy_control_t

Parameters

in	*pdHandle*	PD phy instance handle.
in	*control*	The control code.
in	*param*	The control parameter.

Return values

kStatus_USB_Success	initialization success.
other	value error code.

3.3 Macro Definition Documentation

3.3.1 #define PD_VBUS_POWER_STATE_VSAFE5V_MASK (0x01u)

3.3.2 #define PD_VBUS_POWER_STATE_VSAFE0V_MASK (0x02u)

3.3.3 #define PD_VBUS_POWER_STATE_VBUS_MASK (0x04u)

3.3.4 #define PD_VBUS_POWER_STATE_VSYS_MASK (0x08u)

3.3.5 #define PD_VBUS_POWER_STATE_SINK_DISCONNECT (0x10u)

3.4 Enumeration Type Documentation

3.4.1 enum pd_cc_type_t

Enumerator

kPD_CCInvalid Disable CC communication function.
kPD_CC1 Enable CC1 communication function.
kPD_CC2 Enable CC2 communication function.

3.4.2 enum pd_phy_cc_state_t

Enumerator

kCCState_SrcOpen as source, detect no Rd or Ra
kCCState_SrcRd as source, detect partner's Rd
kCCState_SrcRa as source, detect partner's Ra
kCCState_SnkOpen as sink, detect no Rp
kCCState_SnkRp as sink, detect partner's Rp, PHY driver please use kCCState_SnkRpDefault,kCCState_SnkRp15,kCCState_SnkRp3
kCCState_SnkRpDefault as sink, detect partner's Rp (default current)
kCCState_SnkRp1_5 as sink, detect partner's Rp (1.5A current)
kCCState_SnkRp3_0 as sink, detect partner's Rp (3.0A current)
kCCState_Unknown looking for connection (not connected) or state is unknown
kCCState_Unstable CC state is not stable, CC state cannot stay as this value, need return stable state a little time later. PD stack cannot change state machine relying on this state , need wait the stable state.

3.4.3 enum pd_phy_control_t

Enumerator

PD_PHY_UPDATE_STATE when PHY state change, PHY driver will call PD_Notify(PD_PHY_EVENT_STATE_CHANGE), PD stack will call this control code to update PHY state. This method can keep all PHY operations are in the same task and there are no nested PHY operations. So the PHY operations are steady.

PD_PHY_CONTROL_ALERT_INTERRUPT Enable or Disable alert pin interrupt function.

PD_PHY_ENTER_LOW_POWER_STATE control PHY enter low power

PD_PHY_EXIT_LOW_POWER_STATE control PHY exit low power

PD_PHY_GET_PHY_VENDOR_INFO Get PHY vendor info, the return value is pd_phy_vendor_info_t.

PD_PHY_CONTROL_POWER_PIN control the PHY's pins output 1 or 0. these pins are used to control board's vbus power.

PD_PHY_CONTROL_VBUS_DETECT (vbus control) enable/disable vbus detect function

PD_PHY_SET_VBUS_TRANSFORM_STATE (vbus control) Tell PHY driver vbus is changing or stable. (for example: pr swap)

PD_PHY_DISCHARGE_VBUS (vbus control) discharge VBus

PD_PHY_CONTROL_VBUS_AUTO_DISCHARGE (vbus control) control auto discharge

PD_PHY_CONTROL_VBUS_ALARM (vbus control) control voltage alarm

PD_PHY_SET_TYPEC_VBUS_ILIM (vbus control) set Type-C VBus current ilim

PD_PHY_GET_VBUS_POWER_STATE (vbus info) get vbus state, bit0 - vsafe5v, bit1 - vsafe0v, bit2 - vbus, bit3 - vsys the parameter: when the bit is set, it means the vbus state is requested. For example: bit0 and bit2 is set, it means vsafe5v and vbus states is requested, and don't care the other vbus states.

PD_PHY_FR_SWAP_CHECK_VBUS_APPLIED (vbus info) get the vbus supply state in fast role swap

PD_PHY_INFORM_VBUS_VOLTAGE_RANGE (vbus info) inform the vbus voltage, high 16 bits - high voltage, low 16 bits - low voltage.

PD_PHY_CONTROL_VCONN (vconn control) enable/disable vconn power

PD_PHY_DISCHARGE_VCONN (vconn control) discharge vconn

PD_PHY_CONFIG_ATTACH_DETECTION (CC line control) configure PHY to detect attach, currently state is detached.

PD_PHY_CONFIG_DETACH_DETECTION (CC line control) configure PHY to detect detach, currently state is attached.

PD_PHY_RESET_CONNECT_DETECTION (CC line control) reset the PHY connect/disconnect detection function

PD_PHY_SIGNAL_FR_SWAP (CC line control) signal fast role swap

PD_PHY_CONTROL_FR_SWAP (CC line control) enable/disable fast role swap function (detect the fast role swap signal)

PD_PHY_SRC_SET_TYPEC_CURRENT_CAP (CC line control) Set the Rp value to relect the typec current level

PD_PHY_GET_TYPEC_CURRENT_CAP (CC line info) get typec current level

PD_PHY_GET_CC_LINE_STATE (CC line info) get CC line state, used for connect/disconnect

judgement

PD_PHY_GET_LOOK4_CONNECTION_STATE (CC line info) get connection state, pd_phy_look4connect_state_t

PD_PHY_CONNECT_SET_CC_ORIENTATION (message control) set the PD communication CC

PD_PHY_RESET_MSG_FUNCTION (message control) reset the message function.

PD_PHY_DISABLE_MSG_RX (message control, not supported yet) disable message RX function

PD_PHY_DISABLE_MSG_TX (message control, not supported yet) disable message TX function

PD_PHY_CANCEL_MSG_TX (message control, not supported yet) cancel the sending message

PD_PHY_CANCEL_MSG_RX (message control, not supported yet) cancel the receiving message

PD_PHY_SEND_HARD_RESET (message control) send hard reset

PD_PHY_SEND_CABLE_RESET (message control) send cable reset

PD_PHY_SET_MSG_HEADER_INFO (message control) set message header info, goodCRC use this info. the parameter is pd_phy_msg_header_info_t

PD_PHY_RESET_BIST not supported yet

PD_PHY_ENTER_BIST Enter BIST Mode.

PD_PHY_EXIT_BIST Exit BIST Mode.

PD_PHY_GET_BIST_MODE not supported yet

PD_PHY_GET_BIST_ERR_COUNT not supported yet

PD_PHY_SET_VBUS_SINK_DISCONNECT Set register VBUS_SINK_DISCONNECT.

3.4.4 enum pd_phy_notify_event_t

Enumerator

PD_PHY_EVENT_STATE_CHANGE PHY state change, When PD stack receive this event, it will call PD_PHY_UPDATE_STATE control interface to update PHY state. see PD_PHY_UPDATE_STATE discription.

PD_PHY_EVENT_SEND_COMPLETE message sent complete

PD_PHY_EVENT_RECEIVE_COMPLETE message receive complete

PD_PHY_EVENT_HARD_RESET_RECEIVED receive hard reset

PD_PHY_EVENT_VBUS_STATE_CHANGE vbus state change.

PD_PHY_EVENT_VBUS_SINK_DISCONNECT sink disconnect.

PD_PHY_EVENT_FR_SWAP_SINGAL_RECEIVED receive fast role swap signal

PD_PHY_EVENT_VBUS_ALARM vbus alarm high or low asserted

PD_PHY_EVENT_REQUEST_STACK_RESET PHY run into the unrecovered error state, need PD stack to reset PHY.

PD_PHY_EVENT_VCONN_PROTECTION_FAULT vconn protection fault

PD_PHY_EVENT_TYPEC_OVP_OCP_FAULT there are TYPE-C OVP or OCP fault

PD_PHY_EVENT_FAULT_RECOVERY there are fault

3.4.5 enum pd_phy_look4connect_state_t

Enumerator

kLook4ConnState_Looking is looking for connection

3.5 Function Documentation

3.5.1 void PD_Notify (pd_handle *pdHandle*, pd_phy_notify_event_t *event*, void * *param*)

This function is implemented in the PD stack not PHY driver. PHY driver will call this function to notify PD stack that changes and process flow.

Parameters

in	*pdHandle*	the pdHandle that is got through PD_InstanceInit.
in	*event*	see the pd_phy_notify_event_t
in	*param*	the param is different for different event.

3.6 USB PD PHY PTN5110 driver

3.6.1 Overview

The PTN5110 driver implements the PD send/recive message function and connect/disconnect detection function through PTN5110 PHY.

Functions

- pd_status_t PDPTN5110_Init (pd_handle pdHandle)
  Implement the init function interface.
- pd_status_t PDPTN5110_Deinit (pd_handle pdHandle)
  Implement the de-init function interface.
- pd_status_t PDPTN5110_Control (pd_handle pdHandle, pd_phy_control_t control, void ∗param)
  Implement the control function interface.
- pd_status_t PDPTN5110_Send (pd_handle pdHandle, uint8_t startOfPacket, uint8_t ∗buffer, uint32_t length)
  Implement the message send function interface.
- pd_status_t PDPTN5110_Receive (pd_handle pdHandle, uint8_t startOfPacketMask, uint8_t ∗buffer)
  Implement the message receive function interface.

3.7 USB PD I2C driver wrapper

3.7.1 Overview

Provides the same API interface for different I2C.

Macros

- #define USB_PD_I2C_INSTANCE_COUNT (4U)
  I2C instance count.
- #define I2C_TRANSFER_RETRY_COUNT (5U)
  I2C transfer retry count.

Typedefs

- typedef void * usb_pd_i2c_handle
  usb pd I2C driver handle

Functions

- pd_status_t PD_I2cInit (usb_pd_i2c_handle *i2cHandle, uint8_t i2cInstance, uint32_t i2cSrcFreq, PD_I2cReleaseBus i2cReleaseBus)
  Initialize I2C driver adapter instance.
- pd_status_t PD_I2cDeinit (usb_pd_i2c_handle i2cHandle)
  De-initialize I2C driver adapter instance.
- pd_status_t PD_I2cReadBlocking (usb_pd_i2c_handle i2cHandle, uint32_t slave, uint32_t registerAddr, uint8_t registerLen, uint8_t *data, uint32_t num)
  Read data from slave.
- pd_status_t PD_I2cWriteBlocking (usb_pd_i2c_handle i2cHandle, uint32_t slave, uint32_t registerAddr, uint8_t registerLen, uint8_t *data, uint32_t num)
  Write data to slave.

3.7.2 Function Documentation

3.7.2.1 pd_status_t PD_I2cInit (usb_pd_i2c_handle * *i2cHandle,* uint8_t *i2cInstance,* uint32_t *i2cSrcFreq,* PD_I2cReleaseBus *i2cReleaseBus*)

This function return the #i2c_driver_adpter_t instance, the other API use this as the parameter.

Parameters

out	*i2cHandle*	Return the handle.
in	*i2cInstance*	the I2C instance index, see pd_phy_interface_t

Return values

kStatus_PD_Success	initialization success.
kStatus_PD_Error	error code.

3.7.2.2 pd_status_t PD_I2cDeinit (usb_pd_i2c_handle *i2cHandle*)

Parameters

in	*i2cHandle*	The handle from PD_I2cInit

Return values

kStatus_PD_Success	initialization success.
kStatus_PD_Error	error code.

3.7.2.3 pd_status_t PD_I2cReadBlocking (usb_pd_i2c_handle *i2cHandle,* uint32_t *slave,* uint32_t *registerAddr,* uint8_t *registerLen,* uint8_t * *data,* uint32_t *num*)

Parameters

in	*i2cHandle*	The handle from PD_I2cInit.
in	*slave*	For I2C it is slave address.
in	*registerAddr*	The access register address. Transferred MSB(most significant byte) first.
in	*registerLen*	The register addreess's length, normally it is one byte or two bytes.
in	*data*	The data buffer.
in	*num*	The data length.

Return values

kStatus_PD_Success	success.
kStatus_PD_Error	error code.

3.7.2.4 pd_status_t PD_I2cWriteBlocking (usb_pd_i2c_handle *i2cHandle,* uint32_t *slave,* uint32_t *registerAddr,* uint8_t *registerLen,* uint8_t * *data,* uint32_t *num*)

Parameters

in	*i2cHandle*	The handle from PD_I2cInit.
in	*slave*	For I2C it is slave address.
in	*registerAddr*	The access register address. Transferred MSB(most significant byte) first.
in	*registerLen*	The register addreess's length, normally it is one byte or two bytes.
in	*data*	The data buffer.
in	*num*	The data length.

Return values

kStatus_PD_Success	success.
kStatus_PD_Error	error code.

Chapter 4
USB OS Adapter

Please reference to MCUXpresso SDK API Reference Manual.

Chapter 5
Appendix

5.1 Overview

Modules

- Appendix.A
- Appendix.B

5.2 Appendix.A

This chapter introduce few typical use cases.

In these cases, the source's capabilities are: 5V/2A, 12V/2A and 20V/1A, sink's expected power is 12-V/1A.

Case1 Request expected power in connection.

The event and API callback flow is as follow:

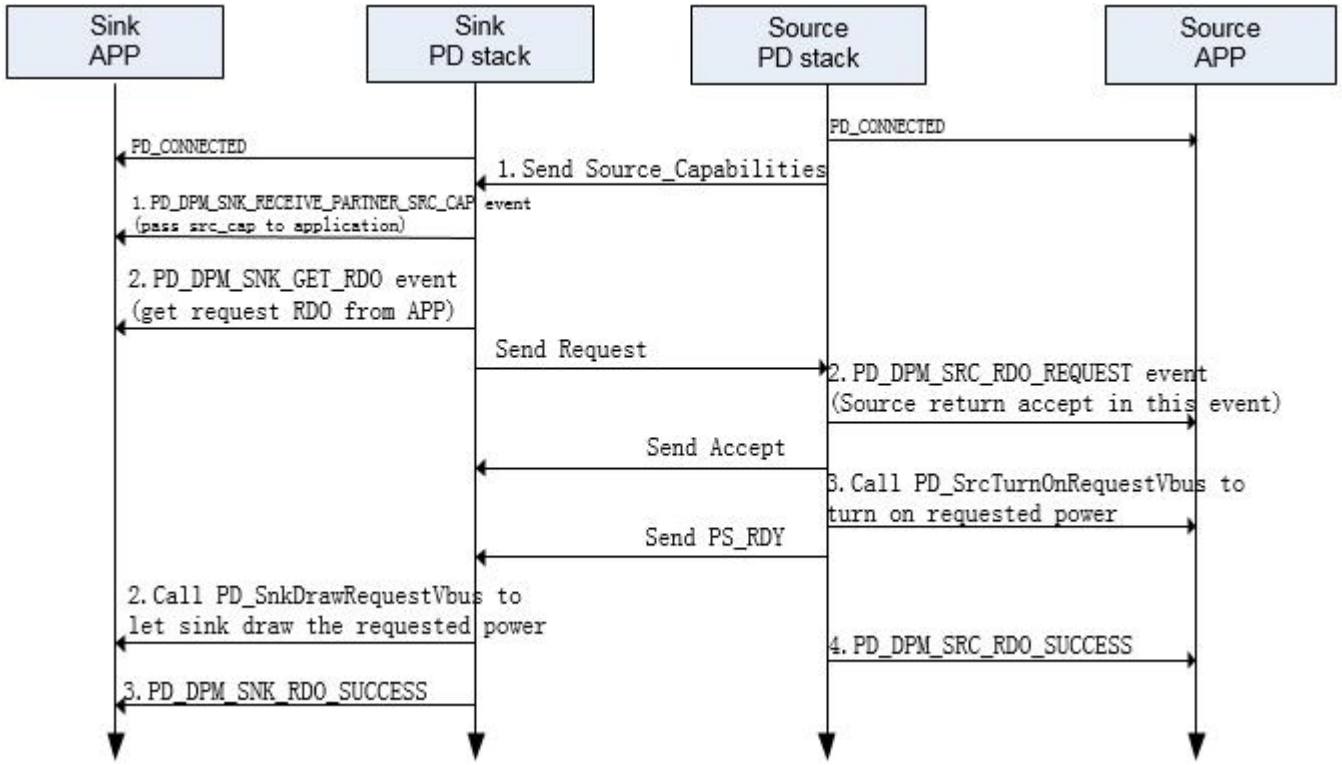

Figure 5.2.1: request power in connection

- Source
  - Step1, send the source_capabilities (two PDOs: 5V2A and 12V/2A).
  - Step2, source receive the request, the request power is 12V/1A source will return accept for the request.
  - Step3, source call the power API to turn on the requested power.
  - Step4, this event tell application the result.
- Sink
  - Step1, sink receive partner's source_capabilities, pass the capabilities to applicaion.
  - Step2, sink get the requested RDO, application parse the source capabilities and return the RDO (12V/1A).

- Step3, sink can draw the requested 12V/1A power.
- Step4, this event tell application the result.

Case2 Request high voltage after connection.

After case1, sink request 20V/1A in this case.

The event and API callback flow is as follow:

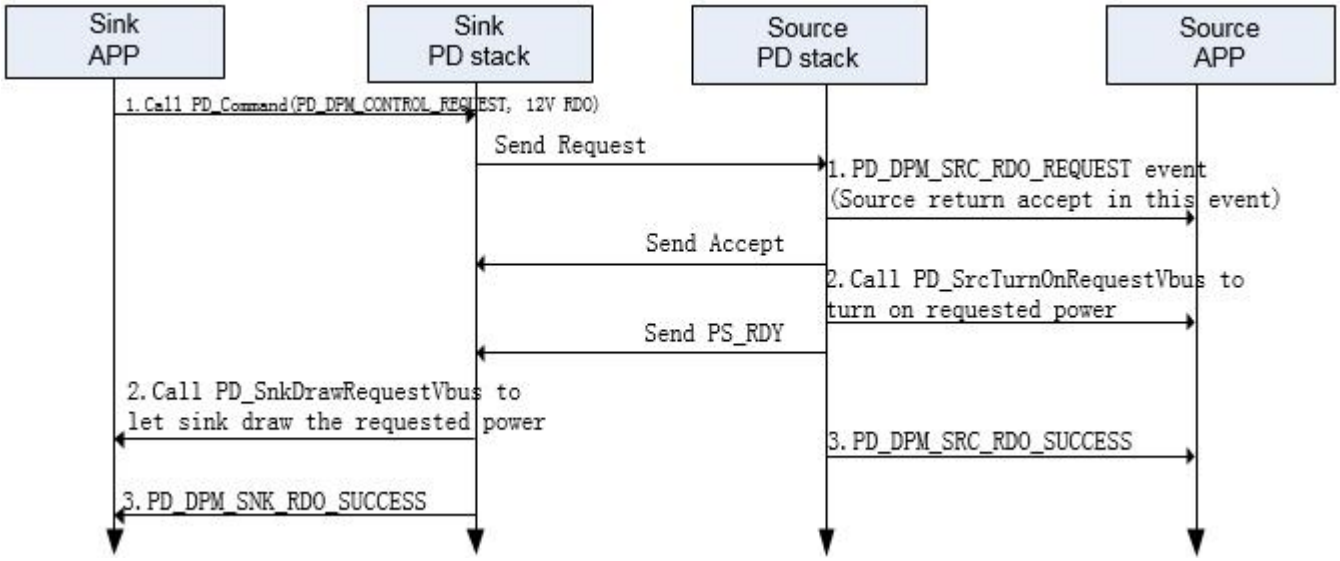

Figure 5.2.2: request power in connection

- Source
  - Step1, source receive the request, the request power is 20V/1A source will return accept for the request.
  - Step2, source call the power API to turn on the requested power.
  - Step3, this event tell application the result.
- Sink
  - Step1, sink application call the PD_Command(PD_DPM_CONTROL_REQUEST, 20V RDO) to trigger the request flow. This API is asynchronous will return immediately.
  - Step2, sink can draw the requested 20V/1A power.
  - Step3, this event tell application the result.

Case3 Request power role swap.

Assume source's external power is removed and it request power role swap.

The event and API callback flow is as follow:

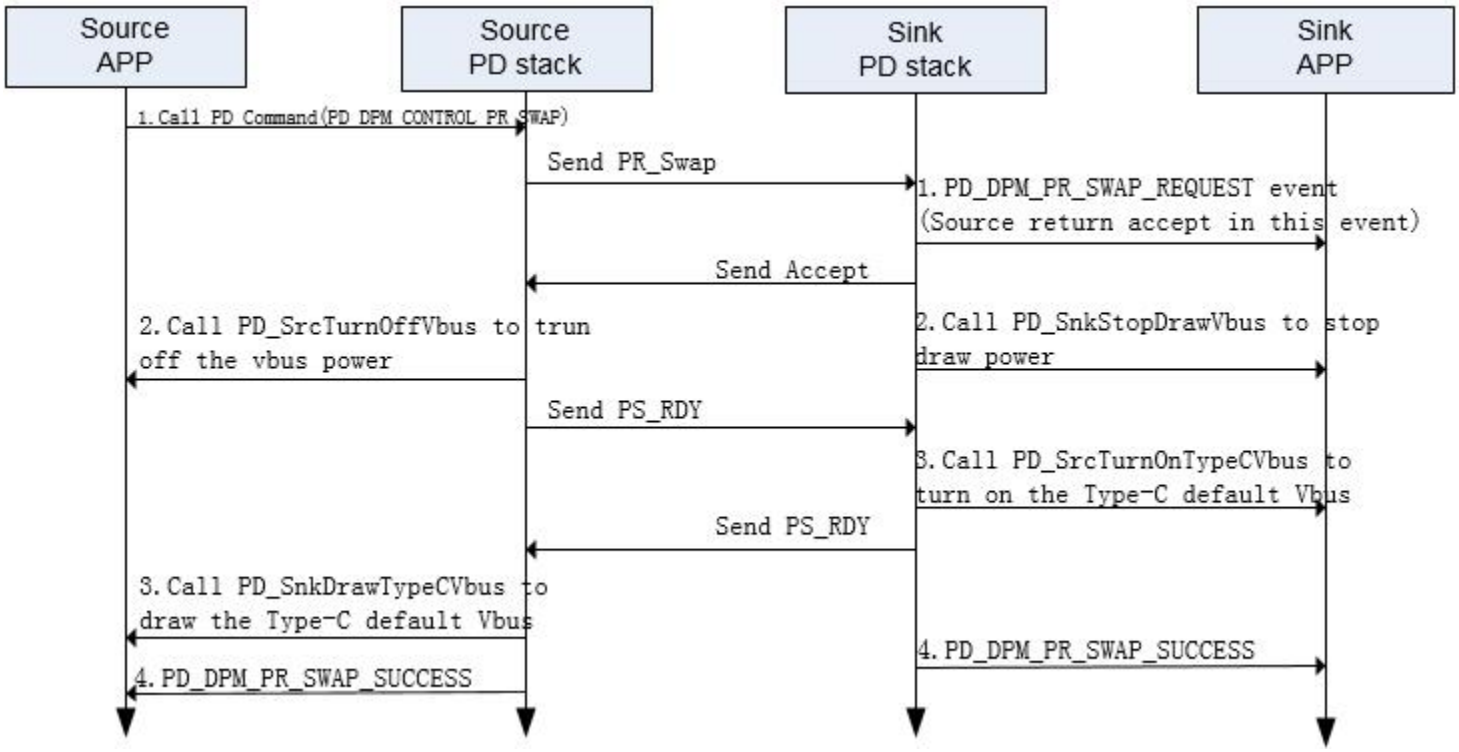

Figure 5.2.3: request power in connection

- Source
  - Step1, source application call the PD_Command(PD_DPM_CONTROL_PR_SWAP) to trigger the swap request flow.
  - Step2, source call the power API to turn off Vbus power supply.
  - Step3, source can draw Type-C default Vbus.
  - Step3, this event tell application the result.
- Sink
  - Step1, sink receive the request, sink will return accept for the request.
  - Step2, sink call the power API to stop draw Vbus power.
  - Step3, sink call the power API to turn on the Type-C default Vbus power.
  - Step4, this event tell application the result.

5.3 Appendix.B

This chapter introduce the PD stack functions configure MACROs that are defined in the usb_pd_config.h, application need provide this file.

Note:

- The configure MACROs are used to configure whether one function is supported or not. For example: if extended message is not supported, PD_CONFIG_EXTENDED_MSG_SUPPORT should be set as (0) to decrease code size.
- Even if the MACRO is set as (1), it only mean the stack has this function, doesn't mean the stack will do this function. For example: PD_CONFIG_TRY_SRC_SUPPORT is set as (1), it doesn't mean PD stack will do Try.SRC, user need do Try.SRC through the pd_instance_config_t.

The MACROs are as follow:

- PD_CONFIG_PTN5110_PORT
  PD PTN5110 PHY driver instance count, meantime it indicates PTN5110 PHY driver enable or disable.
  - if 0, PTN5110 driver is disable.
  - if greater than 0, PTN5110 driver is enable.
- PD_CONFIG_CMSIS_I2C_INTERFACE
  Enable CMSIS I2C driver
- PD_CONFIG_CMSIS_SPI_INTERFACE
  Enable CMSIS SPI driver.
- PD_CONFIG_SOURCE_ROLE_ENABLE
  Enable PD stack source role function.
- PD_CONFIG_SINK_ROLE_ENABLE
  Enable PD stack sink role function.
- PD_CONFIG_DUAL_POWER_ROLE_ENABLE
  Enable PD stack dual power role function.
- PD_CONFIG_DUAL_DATA_ROLE_ENABLE
  Enable PD stack dual data role function.
- PD_CONFIG_VCONN_SUPPORT
  Enable Vconn support (vconn_swap, vconn supply).
- USBPD_ENABLE_VCONN_DISCHARGE
  Enable Vconn discharge function.
- PD_CONFIG_VENDOR_DEFINED_MESSAGE_ENABLE
  Enable vendor defined message function.
- PD_CONFIG_ALTERNATE_MODE_SUPPORT.
  Enable alternate mode function.
- PD_CONFIG_SRC_AUTO_DISCOVER_CABLE_PLUG
  Enable auto discovery cable plug function when connection.
- PD_CONFIG_CABLE_COMMUNICATION_ENABLE
  Enable cable communication function.
- PD_CONFIG_SINK_DETACH_DETECT_WAY
  Config the detach detect way
  - PD_SINK_DETACH_ON_VBUS_ABSENT: detach is detected when vubs absent.

  - PD_SINK_DETACH_ON_CC_OPEN: detach is detected when CC is open.
- PD_CONFIG_EXTENDED_MSG_SUPPORT
  Enable PD3.0 extended message function.
- PD_CONFIG_PD3_FAST_ROLE_SWAP_ENABLE
  Enable fast role swap function.
- PD_CONFIG_PD3_AMS_COLLISION_AVOID_ENABLE
  Enable PD3.0 AMS collision avoid function.
- PD_CONFIG_MIN_DISCHARGE_TIME_ENABLE
  Set the discharge time.
  - 1: use the 14ms.
  - 0: use the 650ms.
- PD_CONFIG_TRY_SNK_SUPPORT
  Enable Try.SNK function.
- PD_CONFIG_TRY_SRC_SUPPORT
  Enable Try.SRC function.
- PD_CONFIG_SINK_ACCESSORY_SUPPORT
  Enable sink accessory function.
- PD_CONFIG_AUDIO_ACCESSORY_SUPPORT
  Enable audio accessory function.
- PD_CONFIG_DEBUG_ACCESSORY_SUPPORT
  Enable debug accessory function.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.