

# **USB Power Delivery Software Framework(PSF) User Guide**

Version 1.09

## Table of Contents

<b>1 Revision History</b>	<b>1-1</b>
<b>2 Introduction</b>	<b>2-2</b>
2.1 Feature Overview	2-2
2.2 References	2-3
2.3 Terms and Abbreviations	2-3
<b>3 Legal Information</b>	<b>3-4</b>
<b>4 PSF Architecture Overview</b>	<b>4-5</b>
<b>5 Supported/Non Supported PD Features</b>	<b>5-8</b>
<b>6 Supported/Non Supported PD Messages</b>	<b>6-9</b>
<b>7 Directory Structure</b>	<b>7-11</b>
7.1 Source folder	7-11
7.2 SOC_Portable folder	7-13
7.3 Demo folder	7-13
<b>8 PSF Configuration Options</b>	<b>8-14</b>
8.1 Include/Exclude Features	8-14
8.1.1 INCLUDE_PD_3_0	8-15
8.1.2 INCLUDE_PD_SOURCE	8-15
8.1.3 INCLUDE_PD_SINK	8-15
8.1.4 INCLUDE_POWER_FAULT_HANDLING	8-16
8.1.5 INCLUDE_UPD_PIO_OVERRIDE_SUPPORT	8-16
8.1.6 INCLUDE_POWER_MANAGEMENT_CTRL	8-16
8.1.7 INCLUDE_PDFU	8-17
8.1.8 INCLUDE_POWER_BALANCING	8-17
8.1.9 INCLUDE_POWER_THROTTLING	8-18
8.1.10 INCLUDE_PD_SOURCE_PPS	8-18
8.1.11 INCLUDE_PD_VCONN_SWAP	8-18
8.1.12 INCLUDE_PD_DR_SWAP	8-19

8.1.13 INCLUDE_PD_DRP	8-19
8.1.14 INCLUDE_PD_PR_SWAP	8-19
8.1.15 INCLUDE_PD_VDM	8-20
8.1.16 INCLUDE_PD_ALT_MODE	8-20
8.1.17 INCLUDE_UPD_HPDP	8-20
8.1.18 INCLUDE_PD_FR_SWAP	8-21
<b>8.2 System Level Configuration</b>	<b>8-21</b>
8.2.1 CONFIG_PD_PORT_COUNT	8-21
8.2.2 CONFIG_DEFINE_UPD350_HW_INTF_SEL	8-22
<b>8.3 Configuration and Status Registers</b>	<b>8-22</b>
8.3.1 _GlobalCfgStatusData Structure	8-23
8.3.2 _PortCfgStatus Structure	8-27
8.3.3 _PBPortCfgStatus Structure	8-46
8.3.4 _PPSPortCfgStatus Structure	8-47
8.3.5 _VDMPortCfgStatus Structure	8-48
8.3.6 _AltModePortCfgStatus Structure	8-49
8.3.7 gasCfgStatusData Variable	8-50
8.3.8 INCLUDE_CFG_STRUCT_MEMORY_PAD_REGION	8-50
<b>8.4 DC-DC Buck Boost Controller Configuration</b>	<b>8-51</b>
8.4.1 Power Control GPIO Enumeration constants	8-51
8.4.1.1 eUPD_OUTPUT_PIN_MODES_TYPE Enumeration	8-51
<b>8.5 Debug Hook Configuration</b>	<b>8-52</b>
8.5.1 CONFIG_HOOK_DEBUG_MSG	8-52
<b>8.6 PDFU Configuration</b>	<b>8-52</b>
8.6.1 CONFIG_PDFU_SUPPORTED	8-53
8.6.2 CONFIG_PDFU_VIA_USBDPD_SUPPORTED	8-53
8.6.3 CONFIG_MAX_FIRMWARE_IMAGESIZE	8-53
8.6.4 CONFIG_RECONFIG_PHASE_WAITTIME	8-54
8.6.5 CONFIG_TRANSFER_PHASE_WAITTIME	8-54
8.6.6 CONFIG_UPDATABLE_IMAGEBANK_INDEX	8-54
8.6.7 CONFIG_VALIDATION_PHASE_WAITTIME	8-55
8.6.8 CONFIG_HWMajor_VERSION	8-55
8.6.9 CONFIG_HWMinor_VERSION	8-55
<b>9 System level integration of PSF</b>	<b>9-57</b>
<b>9.1 Hardware Requirements</b>	<b>9-57</b>
9.1.1 UPD350	9-57
9.1.1.1 Hardware Communication Interface	9-57
9.1.1.2 PIOs for UPD350 IRQs	9-58
9.1.1.3 PIO for UPD350 Reset	9-59

9.1.2 Hardware Timer	9-59
9.1.3 DC-DC Buck Boost Controller	9-59
<b>9.2 Software Requirements</b>	<b>9-60</b>
9.2.1 Memory Requirement	9-60
9.2.2 Multi-Port Support	9-61
9.2.3 Endianness	9-61
<b>9.3 Steps to integrate PSF</b>	<b>9-61</b>
 <b>10 APIs Implementation required for SW integration</b>	 <b>10-62</b>
<b>10.1 Data types</b>	<b>10-62</b>
<b>10.2 APIs to be implemented by the User Application</b>	<b>10-62</b>
10.2.1 UPD350 Hardware Interface Configurations	10-63
10.2.1.1 MCHP_PSF_HOOK_UPDHW_INTF_INIT	10-63
10.2.1.2 MCHP_PSF_HOOK_UPD_READ	10-63
10.2.1.3 MCHP_PSF_HOOK_UPD_WRITE	10-65
10.2.2 PD Timer Configuration	10-66
10.2.2.1 MCHP_PSF_PDTIMER_INTERRUPT_RATE	10-66
10.2.2.2 MCHP_PSF_HOOK_HW_PDTIMER_INIT	10-66
10.2.2.3 MCHP_PSF_CONFIG_16BIT_PDTIMER_COUNTER	10-67
10.2.3 SoC Interrupt Enable/Disable	10-67
10.2.3.1 MCHP_PSF_HOOK_ENABLE_GLOBAL_INTERRUPT	10-67
10.2.3.2 MCHP_PSF_HOOK_DISABLE_GLOBAL_INTERRUPT	10-68
10.2.4 Memory Compare and Copy	10-68
10.2.4.1 MCHP_PSF_HOOK_MEMCMP	10-68
10.2.4.2 MCHP_PSF_HOOK_MEMCPY	10-69
10.2.5 Structure Packing	10-70
10.2.5.1 MCHP_PSF_STRUCT_PACKED_START	10-70
10.2.5.2 MCHP_PSF_STRUCT_PACKED_END	10-70
10.2.6 Port Power Control	10-70
10.2.6.1 MCHP_PSF_HOOK_HW_PORTPWR_INIT	10-71
10.2.6.2 MCHP_PSF_HOOK_PORTPWR_DRIVE_VBUS	10-71
10.2.6.3 MCHP_PSF_HOOK_PORTPWR_CONFIG_SINK_HW	10-72
10.2.6.4 MCHP_PSF_HOOK_DRIVE_DAC_I	10-73
10.2.7 Boot time Configuration	10-74
10.2.7.1 MCHP_PSF_HOOK_BOOT_TIME_CONFIG	10-74
10.2.8 GPIO Configuration	10-75
10.2.8.1 GPIO Init Function	10-75
MCHP_PSF_HOOK_GPIO_FUNC_INIT	10-75
10.2.8.2 GPIO Drive Function	10-76
MCHP_PSF_HOOK_GPIO_FUNC_DRIVE	10-76

10.2.8.3 GPIO Control enum Constants	10-77
eMCHP_PSF_GPIO_DriveValue Enumeration	10-77
eMCHP_PSF_GPIO_Functionality Enumeration	10-77
10.2.9 Hooks for Policy Manager	10-80
10.2.9.1 MCHP_PSF_HOOK_DPM_PRE_PROCESS	10-80
10.2.10 Debug Hook Functions	10-81
10.2.10.1 MCHP_PSF_HOOK_DEBUG_INIT	10-81
10.2.10.2 MCHP_PSF_HOOK_PRINT_CHAR	10-81
10.2.10.3 MCHP_PSF_HOOK_PRINT_INTEGER	10-82
10.2.10.4 MCHP_PSF_HOOK_PRINT_TRACE	10-82
10.2.11 PD Firmware Upgrade	10-83
10.2.11.1 MCHP_PSF_HOOK_BOOT_FIXED_APP	10-83
10.2.11.2 MCHP_PSF_HOOK_BOOT_UPDATABLE_APP	10-84
10.2.11.3 MCHP_PSF_HOOK_GETCURRENT_IMAGEBANK	10-84
10.2.11.4 MCHP_PSF_HOOK_IS_PDFU_ALLOWED_NOW	10-85
10.2.11.5 MCHP_PSF_HOOK_PROGRAM_FWBLOCK	10-85
10.2.11.6 MCHP_PSF_HOOK_VALIDATE_FIRMWARE	10-86
10.2.12 Hooks to Read Output Voltage and Current	10-87
10.2.12.1 MCHP_PSF_HOOK_GET_OUTPUT_CURRENT_IN_mA	10-87
10.2.12.2 MCHP_PSF_HOOK_GET_OUTPUT_VOLTAGE_IN_mV	10-88
<b>10.3 APIs to be called by the User application</b>	<b>10-88</b>
10.3.1 MchpPSF_Init	10-89
10.3.2 MchpPSF_PDTimerHandler	10-89
10.3.3 MchpPSF_UPDIrqHandler	10-89
10.3.4 MchpPSF_RUN	10-90
 <b>11 Notification callback from PSF</b>	 <b>11-91</b>
11.1 MCHP_PSF_NOTIFICATION Enumeration	11-91
11.2 MCHP_PSF_NOTIFY_CALL_BACK	11-96
11.3 eMCHP_PSF_NOTIFY_IDLE Enumeration	11-97
11.4 MCHP_PSF_HOOK_NOTIFY_IDLE	11-97
11.5 MCHP_PSF_HOOK_PDTIMER_EVENT	11-98

# 1 Revision History

REV	DATE	DESCRIPTION OF CHANGE
0.92	11-Dec-19	Initial Revision
0.95	1-Jan-20	Revised multiple sections to improve document flow
1.00	26-Feb-20	Updated document version to align with v1.00 release
1.01	19-Mar-20	Updated the document to align with recent PSF_Config header file updates where various PSF Configuration Options are moved to global configuration structure and Type C and Policy Engine Timeouts are moved inside PSF stack.
1.02	7-Apr-20	Sink functionality added. GotoMin support, Sink related configuration registers updated; <u>MCHP_PSF_HOOK_DRIVE_DAC_I()</u> added.
1.03	21-Apr-20	Added EN_SINK functionality. Updated comments for elements of PortCfgStatus Structure accordingly.
1.04	19-Jun-20	Added PB, PT and PPS features for source. PIO configurability updated to have only mandatory PIOs configurable as part of stack
1.05	24-Jul-20	Added PSF Idle notification hooks in section 11 and updated typos
1.06	20-Aug-2020	Added swap related notification hooks in section 11 and updated section 8 to include swap policy configuration
1.07	11-Sep-2020	Added VDM include/exclude macro in section 8.1, VDM notifications in section 11.1 and removed CONFIG_PORT_UPD_IDLE_TIMEOUT_MS
1.08	11-Nov-2020	Updated various sections to include Alt Mode support
1.09	24-Dec-2020	Updated various sections to include Fast Role Swap support and corrected the typos throughout the document

## 2 Introduction

USB Power Delivery Software Framework (PSF) with USB-PD Port Controller [UPD350](#) is a full-featured, configurable USB Power Delivery(USB-PD) software stack that is compliant to the USB-PD 3.0 Specification. PSF is MCU-agnostic. A system designer may choose from Microchip's wide catalog of Micro Controller Units(MCU) or System on Chips(SoC) to meet specific system requirements while optimizing cost and PCB space. Versatility is achieved through flexible configuration and hardware abstraction. Most PD functions, like power roles and capabilities, data role and capabilities and product/vendor IDs are achieved through configuration registers where each port can be configured in a manner independent of other ports.

This document is divided into the following sections:

- [PSF Architecture Overview](#) - An overview of how an SoC could be interfaced with [UPD350](#) to enable PD support
- [Supported/Non Supported PD Features](#) - List of supported and non-supported USB-PD features
- [Supported/Non Supported PD Messages](#) - List of supported and non-supported USB-PD messages
- [Directory Structure](#) - Directory organization of USB-PD source files and user configurable files
- [PSF Configuration Options](#) - A range of compile and run time configuration options for USB-PD and other special features
- [System level integration of PSF](#) - Steps for integrating PSF with an existing application or system
- [APIs Implementation required for SW integration](#) - List of APIs needed to integrate PSF with an existing application or a new SoC
- [Notification callback from PSF](#) - Notifications given by PSF through its callback to USER\_APPLICATION

### 2.1 Feature Overview

Following are the key features of PSF:

- Compliant to USB Power Delivery 3.0 Specification Revision 1.2 and USB Type-C Specification Revision 1.3
- Multi-port support upto 4 ports, where each port can be configured independent of other ports
- USB-PD Source-only or Sink-only or Dual Role Power (DRP) port specific configurability
- I2C and GPIO mode DC-DC control
- Power Balancing
- Power Throttling
- Source only Programmable Power Supply(PPS) support
- Multiple Sink PDO selection algorithms
- Power Role Swap, Data Role Swap and VCONN Swap support
- Fast Role Swap support
- Vendor Defined Message(VDM) tunneling to enable Alternate Mode
- Runtime Client requests to initiate Renegotiation, Power Role Swap, Data Role Swap, VCONN Swap, VDM and handle port enable/disable and externally detected power fault conditions
- FW update through CC support compliant to PD FW Update Specification Revision 1.0 is available but not tested
- Multiple compile time and run time configuration options available to tailor the stack to meet the needs of USER\_APPLICATION

Refer section [Supported/Non Supported PD Features](#) for further details.

## 2.2 References

- Microchip [UPD350](#) Datasheet
- USB Power Delivery 3.0 Specification Revision 1.2
- USB Type-C Specification Revision 1.3
- PD FW Update Specification Revision 1.0

## 2.3 Terms and Abbreviations

Term	Definition
USB-PD	USB Power Delivery
SPI	Serial Peripheral Interface bus - Full-duplex bus utilizing a single-master/multi-slave architecture. SPI is a 4-wire bus consisting of SPI CLK, SPI MOSI, SPI MISO and SPI SS
<a href="#">UPD350</a>	Microchip USB Type-C™ Power Delivery 3.0 Port Controller
Sink Role	USB Type-C Port which sinks power from its Port Partner
Source Role	USB Type-C Port which sources power to its Port Partner
USER_APPLICATION	The software platform that runs on SoC where the PSF is integrated
SoC	System on Chip
FW	Firmware
DFP	Downstream Facing Port
UFP	Upstream Facing Port
PDFU	Power Delivery Firmware Update
PDO	Power Data Object
APDO	Augmented Power Data Object
FRS	Fast Role Swap
PPS	Programmable Power Supply
PB	Power Balancing
PT	Power Throttling
API	Application Programming Interface
DRP	Dual Role Power
DRD	Dual Role Data
VDM	Vendor Defined Message
Port Partner	A Contract is negotiated between a Port Pair connected by a USB cable. These ports are known as Port Partners.



# 3 Legal Information

## Software License Agreement

Copyright © [2019-2020] Microchip Technology Inc. and its subsidiaries.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

# 4 PSF Architecture Overview

PSF is a lightweight configurable software framework that helps realize a power delivery solution with any suitable SoC and Microchip's [UPD350](#) Type-C PD controller. The software architecture of the PSF stack is shown in figure 4.1 and described below:

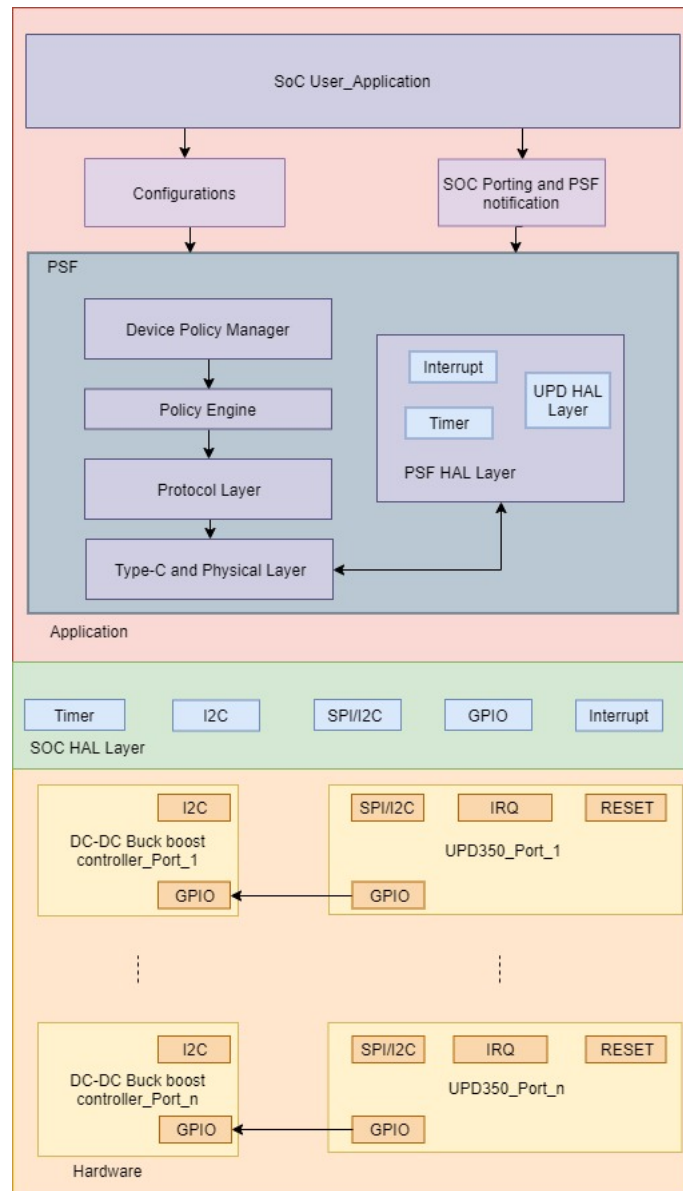


Figure 4.1

## SoC User Application Layer:

SoC User\_Application layer performs the following:

- Configures PSF through PSF\_Config.h and PSF<Demo>\_BootCfg.c and .h files.
- Drives appropriate PIO pins based on PIO events from PSF and handles interaction of USER\_APPLICATION with PSF through PSF\_APIHook.h, PSF<Demo>\_App.c and .h files.

**SoC HAL Layer:**

- SoC HAL layer has the drivers for SoC's modules required for PSF functionalities like Timer, Interrupt, SPI/I2C Interface and GPIO.
- These drivers are generated using MPLABX Harmony.
- The files present under usb-pd-software-framework-public\PSF\Demo\<PSF\_EVB\_<demo>\firmware\src\config make up this layer.

**Device Policy Manager (DPM) Layer:**

The Device Policy Manager is responsible for the following,

- The DPM role is to maintain the Local Policy of the device.
- Controls the Source/Sink in the device.
- For a USB-PD Source, it monitors Source's present capabilities & in case of any change in the capability it triggers notification to Policy Engine.
- For a USB-PD Sink, it evaluates & respond to capability related requests from the Policy Engine for a given port.
- Controls the USB-C Port Control module for each Port.
- It communicates with Policy Engine Layer and provides it with necessary information such as Cable/Device detection as Source/Sink and the PDO to be negotiated etc.
- Takes care of any VBUS or VCONN power fault

The files that make up this layer are policy\_manger.c and .h and policy\_manager\_sm.c.

**Policy Engine Layer:**

There is one Policy Engine instance per Port that interacts with the Device Policy Manager to implement present Local Policy for that Port.

- Policy Engine drives message sequences for various operations.
- It is responsible for establishing Explicit Contract by negotiating power based on Local Policy.

The files that constitute this layer are policy\_engine.c and .h, policy\_engine\_fwup.c and .h, policy\_engine\_snk.c, policy\_engine\_src.c, policy\_engine\_swap.c and policy\_engine\_vdm.c

**Protocol Layer:**

- Protocol Layer handles message construction, transmission, & reception, reset operation, message error handling.
- It sends/receives messages through [UPD350](#) using SPI wrapper functions.
- It acts as an interface between Policy Engine & [UPD350](#).
- The files that constitute this layer are protocol\_layer.c and .h.

**Type-C Connector Management:**

Type-C Management includes following operation as defined in the USB Type-C v1.3 specification,

- Source-to-Sink attach/detach detection
- Plug orientation/cable twist detection
- Initial power (Source-to-Sink) detection and enabling PD communication
- VBUS Detection & Type C Error Recovery Mechanism

The files that constitute this layer are typeC\_control.c and .h.

**Port Power Management:**

Port power management as a Source/Sink for multiple ports

- Configurable Fixed PDOs per port up to 100W
- Over current sense on ports

The files that make up this layer are portpower\_control.c and .h.

#### **PSF HAL Layer:**

- Apart from the layers specified by USB-PD Specification for PSF, PSF has HAL layer to interact and access [UPD350](#)'s hardware peripherals (example - GPIO, DRP offload and USB PD MAC etc) and software peripherals (example - timer).
- The files that constitute this layer are upd\_hw.c and .h, upd\_interrupts.c and .h and pd\_timer.c and .h.
- upd\_hw.c and .h files handle [UPD350](#)'s peripherals.
- upd\_interrupts.c and .h files handle all interrupts from [UPD350](#).
- pd\_timer.c and .h files handles software timeouts based on interrupts from hardware timer.

#### **Hardware:**

This layer supports the following:

- DC-DC buck-boost controller, which is external to [UPD350](#), is controlled to drive the desired voltage at VBUS. The VBUS voltage can either be driven by [UPD350](#) PIOs or SoC's I2C controller.
- SoC communication with [UPD350](#) through SPI/I2C or GPI, using SoC HAL layer, to control [UPD350](#)'s IRQ and RESET lines and other peripherals.

The files that constitute this layer are i2c\_dc\_dc\_driver..c and .h and i2c\_dc\_dc\_ung8198.c and .h.

# 5 Supported/Non Supported PD Features

List of supported and non supported USB-PD features in this release of PSF.

NOTE: This section will be updated frequently as the PSF feature set is extended in the future.

Features	Supported/Not Supported
USB-PD Source only	Supported
USB-PD Sink only	Supported
Power negotiation up to 100 watts	Supported
PDFU support through CC	Support is available in PSF stack, but there is no sample application example
VCONN Power support	Supported
Extended message support via Chunking	Supported
Fixed PDOs	Supported
Dynamic Power Balancing	Supported
Power Throttling	Supported
PPS	Supported
Dual-Role Power(DRP)	Supported
Dual-Role Data(DRD)	Supported
Alternate Mode	Supported
Runtime Client Requests	Supported
Fast Role Swap(FRS)	Supported
Unchunked Extended message	Not Supported
UVDM & USB Type-C Bridging	Not Supported

# 6 Supported/Non Supported PD Messages

## Control Messages

USB-PD Control messages	Supported/Not Supported
GoodCRC	Supported
Accept	Supported
Reject	Supported
Ping	Supported
PS_RDY	Supported
Get_Source_Cap (For Sink Role only)	Supported
Get_Sink_Cap (For Source Role only)	Supported
VCONN_Swap	Supported
Wait	Supported
Soft_Reset	Supported
Not Supported	Supported
GotoMin	Supported
DR_SWAP	Supported
PR_SWAP	Supported
Get_Source_Cap_Extended	Not Supported
Get_Status (For Source Role only)	Supported
FR_Swap	Supported
Get_PPS_Status	Not Supported
Get_Country_Codes	Not Supported
Get_Sink_Cap_Extended	Not Supported

## Data Messages

USB-PD Data messages	Supported/Not Supported
Source_Capabilities (Source and DRP Roles only)	Supported
Request	Supported
BIST	Supported
Sink_Capabilities (Sink and DRP Roles only)	Supported
Battery_Status	Not Supported
Alert (Source Role only)	Supported
Get_Country_Info	Not Supported
Vendor_Defined	Supported

**Extended Messages**

USB-PD Extended Message	Supported/Not Supported
Firmware_Update_Request	Supported
Firmware_Update_Response	Supported
Source_Capabilities_Extended	Not Supported
Status (Source Role only)	Supported
Get_Battery_Cap	Not Supported
Get_Battery_Status	Not Supported
Battery_Capabilities	Not Supported
Manufacturer_Info	Not Supported
Security_Request	Not Supported
Security_Response	Not Supported
PPS_Status	Supported
Country_Info	Not Supported
Country_Codes	Not Supported
Sink_Capabilities_Extended	Not Supported

# 7 Directory Structure

The directory structure of PSF is shown in Figure 7.1 below and is organized as follows:

- ../Demo - Contains all the power delivery application, PSF configuration files of Source Lite, Source Pro, Sink and DRP sample applications.
- ../SOC\_Portable - Contains SoC specific integration, PSF APIs porting and drivers files
- ../Source - Contains PSF source and header file

PSF has two user configurable files for configurability and porting:

- PSF\_Config.h -> Contains all the PSF PD features and functions related configuration options
- PSF\_APIHook.h -> Contains all the APIs needed to port PSF to a new SOC

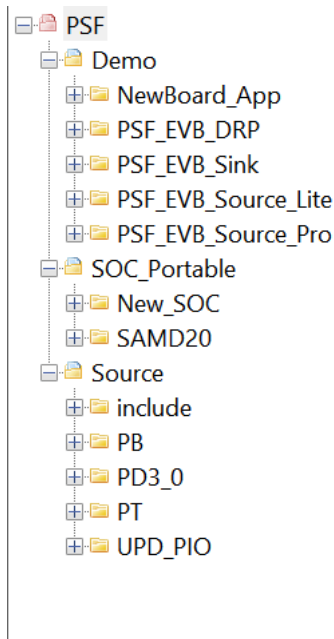


Figure 7.1

## 7.1 Source folder

This folder comprises of all the source files that implement the core USB PD functionality as defined in the USB-PD 3.0 specification.

The sub directory structure is as follows:

- ../PSF\Source\include - contains all the header files for PD3.0 functionality
- ../PSF\Source\PD3\_0 - contains all the source file for PD3.0 functionality
- ../PSF\Source\PB - contains all the source and header files for Power Balancing functionality
- ../PSF\Source\PT - contains all the source and header files for Power Throttling functionality
- ../PSF\Source\UPD\_PIO - contains all the source and header files for UPD PIO functionality

File descriptions of all the files present under include and PD3\_0



File Name	PSF Stack Layer	File description
policy_manager.c policy_manager.h	Device Policy Manager (DPM)	Handles PD across multiple ports and decides how to allocate power for each port.
policy_engine.h policy_engine.c policy_engine_src.c policy_engine_snk.c policy_engine_swap.c policy_engine_vdm.c	Policy Engine	<ul style="list-style-type: none"> <li>Policy Engine (PE) drives Atomic Message Sequences (AMS).</li> <li>policy_engine.c maintains PE functionality common to both Source &amp; Sink whereas policy_engine_src.c and policy_engine_snk.c maintains functionality specific to Source and Sink respectively, policy_engine_swap.c maintains swap functionalities namely VCONN swap, Power Role Swap and Data Role Swap and policy_engine_vdm.c maintains VDM state machine.</li> <li>policy_engine.h is single header file for all the PE related source files.</li> </ul>
protocol_layer.c protocol_layer.h	Protocol Layer	Handles message construction, chunking and retries.
typeC_control.c typeC_control.h	Type-C Connector Management	Type-C control functionality & connector management
pd_timer.c pd_timer.h	Timer management	Contains APIs to manage the different software timers used in the stack.
upd_interrupts.c upd_interrupts.h	Interrupts Management	Handles all interrupts from <a href="#">UPD350</a> for each port.
debug.h	Debug support	Debug interface support APIs file
upd_hw.c upd_hw.h	<a href="#">UPD350</a> Hardware	APIs to access <a href="#">UPD350</a> Hardware GPIO, Registers, PIO override
int_globals.c int_globals.h	Internal globals	Declaration and definition of global variables and data structures used in PSF.
generic_defs.h	Generic Defines	Contains generic type defines for data types
timer_defs.h	Timer Defines	Contains Type C and Policy Engine Timeout configurations
psf_stdinc.h	Standard include file	Standard include file for PSF
pd_main.c	PSF main source file	contains APIs to be called by User_Application
policy_engine_fwup.c policy_engine_fwup.h	Power Deliver Firmware update	Maintains PDFU functionality
portpower_control.c portpower_control.h	Port Power control	Maintains PSF default port power control support
pb_helper.c pb_mgr.c pb_mgr.h	Power Balancing	Maintains Power Balancing related APIs that are used to handle Port Policy Management (PPM) events and helper functions needed for the same.
pt_mgr.c pt_mgr.h	Power Throttling	Maintains Power Throttling related APIs to allow PSF to actively switch between up to three different power capability banks.
upd350_pio_drivers.c upd350_pio_drivers.h	<a href="#">UPD350</a> PIO driver	Maintains <a href="#">UPD350</a> PIO Driver Functionality
ProjectVersion.h	PSF version	Maintains PSF release version

Table 7.1

## 7.2 SOC\_Portable folder

SOC\_Portable contains all the driver porting files required for PSF functionality from SoC layer. It varies depending on the SoC and user application. The list of APIs required for PSF integration and porting are available in \PSF\SOC\_Portable\New\_SOC\PSF\_APIHook.h, Drivers.c and .h files

## 7.3 Demo folder

This folder contains sample applications for different hardware platforms supported in PSF namely:

- Source Lite - PSF Source only sample application with no add-on features
- Source Pro - PSF Source sample application with value-add features such as Power Balancing, PPS, etc.
- Sink - PSF Sink sample application
- DRP - PSF DRP application with Power Role Swap(PRS), Data Role Swap(DRS), Fast Role Swap(FRS) and Alternate Mode support

File descriptions of all the files present under Demo folder are given in the table below.

File Name	File Description
Files under PSF_EVB_<demo>.X folder	MPLABX Project files specific to sample application.
Files under firmware\src folder	Files containing driver APIs to control the peripherals of SoC, which are required for the sample application.
PSF_Config.h	File used to configure PSF.
PSF_<demo>BootCfg.c and .h	Files used to configure PSF's data structures at boot time.
PSF<demo>_App.c and .h	Files used to interact with USER_APPLICATION and PSF.
PSF <demo> Demo Read me.pdf	File to guide a user on setting up the PSF-EVB to work properly with the demo application along with a demonstration of a PD device attached to the PSF-EVB.
i2c_dc_dc_driver.c and .h and i2c_dc_dc_ung8198.c and .h	Files used to control I2C DC DC buck-boost controller.

Table 7.2

With these files the user can build and exercise an application on the PSF evaluation platform or other custom-designed hardware platform. PD functionality can be configured with the "PSF\_Config.h" header file located at "..\Demo\NewBoard\_App\PSF\_Config.h". This file is used to configure the PD application to meet specific system requirements and feature-set. In Boot time configuration file, user must write a function to initialize all the needed configuration parameters.

## 8 PSF Configuration Options

PSF has a range of compile time and runtime configuration options for the various USB-PD features and other special features.



















This release of PSF allows configuration of the following:

- Number of ports
- Power Role - Source only or Sink only or DRP
- Data Role - DFP or UFP
- Port Power control management
- Source & Sink PDOs
- Power Balancing Parameters
- Power Throttling Bank and Power values
- [UPD350](#) PIO Configuration
- Role Swap Policies for VCONN\_Swap, DR\_Swap and PR\_Swap
- Compile time Code & Data RAM inclusion/exclusion based on USB-PD Specification Revision & Power Role

All these configurable options are present in PSF\_Config.h. header file of PSF. PSF\_Config.h shall be customized as per the PD application and included as part of User Application Directory. It is recommended to include this file path as one of preprocessor include paths. The template of PSF\_Config.h for new demo application is available at `..\PSF\Demo\NewBoard_App\`

### 8.1 Include/Exclude Features

#### Macros

	Name	Description
	<a href="#">INCLUDE_PD_3_0</a>	USB-PD V3.0 support code inclusion.
	<a href="#">INCLUDE_PD_SOURCE</a>	Source support code inclusion.
	<a href="#">INCLUDE_PD_SINK</a>	Sink support code inclusion.
	<a href="#">INCLUDE_POWER_FAULT_HANDLING</a>	Power Fault Handling code inclusion.
	<a href="#">INCLUDE_UPD_PIO_OVERRIDE_SUPPORT</a>	PIO Override Feature code inclusion.
	<a href="#">INCLUDE_POWER_MANAGEMENT_CTRL</a>	Power Management Control Support code inclusion.
	<a href="#">INCLUDE_PDFU</a>	PD Firmware update code inclusion.
	<a href="#">INCLUDE_POWER_BALANCING</a>	Power Balancing support code inclusion.
	<a href="#">INCLUDE_POWER_THROTTLING</a>	Power Throttling support code inclusion.
	<a href="#">INCLUDE_PD_SOURCE_PPS</a>	Source PPS support code inclusion.
	<a href="#">INCLUDE_PD_VCONN_SWAP</a>	VCONN_Swap Support code inclusion.
	<a href="#">INCLUDE_PD_DR_SWAP</a>	Data Role Swap support code inclusion.
	<a href="#">INCLUDE_PD_DRP</a>	DRP support code inclusion.
	<a href="#">INCLUDE_PD_PR_SWAP</a>	Power Role Swap support code inclusion.
	<a href="#">INCLUDE_PD_VDM</a>	Vendor Defined Message support code inclusion.
	<a href="#">INCLUDE_PD_ALT_MODE</a>	Alternate Mode support code inclusion.
	<a href="#">INCLUDE_UPD_HPD</a>	Hot Plug Detect support code inclusion.
	<a href="#">INCLUDE_PD_FR_SWAP</a>	Fast Role Swap support code inclusion.

**Description**

The table above lists all the compile time configuration options offered by PSF. These options are implemented using compile time macros (#define) or preprocessor directives. They allow the user to add or remove features as needed and reduce the code size.

---

## 8.1.1 INCLUDE\_PD\_3\_0

**C**

```
#define INCLUDE_PD_3_0 1
```

**Description**

Setting the INCLUDE\_PD\_3\_0 as 1, enables PSF to include USB Power delivery 3.0 specification features (collision avoidance and extended message support via chunking) along with PD 2.0 features at compile time. Users can set this define to 0 to reduce code size if none of the PD enabled ports require PD 3.0 specific features.

**Remarks**

Default value is '1'.

**Example**

```
#define INCLUDE_PD_3_0 1 (Include USB PD 3.0 specific features to PSF)
#define INCLUDE_PD_3_0 0 (Exclude USB PD 3.0 specific features from PSF)
```

---

## 8.1.2 INCLUDE\_PD\_SOURCE

**C**

```
#define INCLUDE_PD_SOURCE 1
```

**Description**

Setting the INCLUDE\_PD\_SOURCE as 1 enables PSF to include the USB PD Source functionality at the compile time. Users can set this define to 0 to reduce code size if none of the PD enabled ports in the system are configured for Source operation.

**Remarks**

Default value is '1' for Source and DRP Applications.

**Example**

```
#define INCLUDE_PD_SOURCE 1 (Include USB PD Source functionality in PSF)
#define INCLUDE_PD_SOURCE 0 (Exclude USB PD Source functionality from PSF)
```

---

## 8.1.3 INCLUDE\_PD\_SINK

**C**

```
#define INCLUDE_PD_SINK 1
```

**Description**

Setting the INCLUDE\_PD\_SINK as 1 enables PSF to include USB PD Sink functionality at the compile time. Users can set this define to 0 to reduce code size if none of the PD enabled ports are configured for Sink operation.

**Remarks**

Default value is '1' for Sink and DRP Applications.

**Example**

```
#define INCLUDE_PD_SINK 1(Include USB PD Sink functionality in PSF)
#define INCLUDE_PD_SINK 0(Exclude USB PD Sink functionality from PSF)
```

---

## 8.1.4 INCLUDE\_POWER\_FAULT\_HANDLING

**C**

```
#define INCLUDE_POWER_FAULT_HANDLING 1
```

**Description**

Setting the INCLUDE\_POWER\_FAULT\_HANDLING as 1 enables PSF to handle power faults (Source and Sink over voltage, Source OCS, Sink under voltage) as per Power Delivery specification Rev3.0 as applicable. Users can set this define to 0 to reduce code size if PSF based power fault handling is not required.

**Remarks**

Default value is 1.

**Example**

```
#define INCLUDE_POWER_FAULT_HANDLING 1(Include Power Fault handling to PSF)
#define INCLUDE_POWER_FAULT_HANDLING 0(Exclude Power Fault handling from PSF )
```

---

## 8.1.5 INCLUDE\_UPD\_PIO\_OVERRIDE\_SUPPORT

**C**

```
#define INCLUDE_UPD_PIO_OVERRIDE_SUPPORT 1
```

**Description**

PIO override is **UPD350** specific feature which changes the state of a PIO without software intervention. PSF uses this feature to disable EN\_VBUS(in case of source operation) or EN\_SINK(in case of sink operation) instantly on detection of a Power Fault or Fast Role Swap Condition. Setting the INCLUDE\_UPD\_PIO\_OVERRIDE\_SUPPORT as 1 enables this feature. Users can set this define to 0 to reduce code size of PSF if PIO override based power fault handling is not required. It is recommended that this define shall be set when Fast Role Swap feature is enabled since Fast Role Swap is tightly coupled with PIO Override feature.

**Remarks**

To use this feature, EN\_VBUS or EN\_SINK, FAULT\_IN and EN\_FRS pins of the system should be **UPD350** PIOs. It is also confined to **INCLUDE\_POWER\_FAULT\_HANDLING** define, thus **INCLUDE\_POWER\_FAULT\_HANDLING** should be declared as 1 for INCLUDE\_UPD\_PIO\_OVERRIDE\_SUPPORT define to be effective. Recommended default value is 1 if **UPD350** PIOs are used for EN\_VBUS, EN\_SINK and FAULT\_IN.

**Example**

```
#define INCLUDE_UPD_PIO_OVERRIDE_SUPPORT 1(Include UPD350 PIO Override support for Power
                                         fault to PSF)
#define INCLUDE_UPD_PIO_OVERRIDE_SUPPORT 0(Exclude UPD350 PIO Override support for Power
                                         fault from PSF)
```

---

## 8.1.6 INCLUDE\_POWER\_MANAGEMENT\_CTRL

**C**

```
#define INCLUDE_POWER_MANAGEMENT_CTRL 1
```

**Description**

Setting the `INCLUDE_POWER_MANAGEMENT_CTRL` as 1 enables PSF to include the functionality that puts the [UPD350](#) into low power mode if [UPD350](#) is inactive for 15 seconds time and PSF notifies the same via the call back [MCHP\\_PSF\\_NOTIFY\\_CALL\\_BACK](#). Users can set this define to 0 to reduce code size of the PSF if low power mode operation of [UPD350](#) is not required for the application.

**Remarks**

Default value is 1.

**Example**

```
#define INCLUDE_POWER_MANAGEMENT_CTRL 1(Include power management feature)
#define INCLUDE_POWER_MANAGEMENT_CTRL 0(Exclude power management feature)
```

---

## 8.1.7 INCLUDE\_PDFU

**C**

```
#define INCLUDE_PDFU 0
```

**Description**

Setting the `INCLUDE_PDFU` as 1 includes the state machine code for PD Firmware Update feature as per USB Power Delivery FW Update Specification v1.0. Users can set this define to 0 to reduce code size if the PSF application does not use Firmware update feature.

**Remarks**

Set to 0 by default. [INCLUDE\\_PD\\_3\\_0](#) should be set to 1 as a prerequisite when `INCLUDE_PDFU` is set to 1.

**Example**

```
#define INCLUDE_PDFU 1(Include PDFU feature)
#define INCLUDE_PDFU 0(Exclude PDFU feature)
```

---

## 8.1.8 INCLUDE\_POWER\_BALANCING

**C**

```
#define INCLUDE_POWER_BALANCING 1
```

**Description**

Setting the `INCLUDE_POWER_BALANCING` as 1 enables PSF to include the PD Power Balancing functionality at the compile time. Users can set this define to 0 to reduce code size if none of the PD enabled Source ports in the system require Power Balancing functionality.

**Remarks**

Default value is 1 for Source application. [INCLUDE\\_PD\\_SOURCE](#) should be set to 1 as a prerequisite when `INCLUDE_POWER_BALANCING` is set to 1.

**Example**

```
#define INCLUDE_POWER_BALANCING 1(Include Power Balancing functionality in PSF)
#define INCLUDE_POWER_BALANCING 0(Exclude Power Balancing functionality from PSF)
```

## 8.1.9 INCLUDE\_POWER\_THROTTLING

C

```
#define INCLUDE_POWER_THROTTLING 1
```

### Description

Setting the INCLUDE\_POWER\_THROTTLING as 1 enables PSF to include the Power Throttling(PT) feature at the compile time. Users can set this define to 0 to reduce code size if none of the Source ports in the system require PT functionality.

### Remarks

Default value is 1 for Source application. [INCLUDE\\_PD\\_SOURCE](#) should be set to 1 as a prerequisite when INCLUDE\_POWER\_THROTTLING is set to 1.

### Example

```
#define INCLUDE_POWER_THROTTLING 1(Include PT functionality in PSF)
#define INCLUDE_POWER_THROTTLING 0(Exclude PT functionality from PSF)
```

## 8.1.10 INCLUDE\_PD\_SOURCE\_PPS

C

```
#define INCLUDE_PD_SOURCE_PPS 1
```

### Description

Setting the INCLUDE\_PD\_SOURCE\_PPS as 1 enables PSF to include the Source Programmable Power Supply(PPS) feature at the compile time. Users can set this define to 0 to reduce code size if none of the Source ports in the system require PPS functionality.

### Remarks

Default value is 1 for Source application. [INCLUDE\\_PD\\_SOURCE](#) and [INCLUDE\\_PD\\_3\\_0](#) should be set to 1 as a prerequisite when INCLUDE\_PD\_SOURCE\_PPS is set to 1.

### Example

```
#define INCLUDE_PD_SOURCE_PPS 1(Include Source PPS functionality in PSF)
#define INCLUDE_PD_SOURCE_PPS 0(Exclude Source PPS functionality from PSF)
```

## 8.1.11 INCLUDE\_PD\_VCONN\_SWAP

C

```
#define INCLUDE_PD_VCONN_SWAP 1
```

### Description

Setting the INCLUDE\_PD\_VCONN\_SWAP as 1 enables PSF to include the VCONN Swap functionality at compile time. Users can set this define to 0 to reduce code size if none of the PD enabled ports require VCONN Swap functionality.

### Remarks

Default value is 1. For Source and DRP Operation, it is mandatory to define this macro as '1'. When [INCLUDE\\_PD\\_SOURCE](#) is defined as '1', define this macro as '1'.

### Example

```
#define INCLUDE_PD_VCONN_SWAP 1(Include VCONN Swap functionality in PSF)
#define INCLUDE_PD_VCONN_SWAP 0(Exclude VCONN Swap functionality from PSF)
```

## 8.1.12 INCLUDE\_PD\_DR\_SWAP

C

```
#define INCLUDE_PD_DR_SWAP 1
```

### Description

Setting the INCLUDE\_PD\_DR\_SWAP as 1 enables PSF to include the Data Role Swap (DR\_SWAP) feature at the compile time. Users can set this define to 0 to reduce code size if none of the ports in the system require DR\_SWAP functionality.

### Remarks

Default value is 1. Users can configure it based on the application.

### Example

```
#define INCLUDE_PD_DR_SWAP 1(Include DR_SWAP functionality in PSF)
#define INCLUDE_PD_DR_SWAP 0(Exclude DR_SWAP functionality from PSF)
```

## 8.1.13 INCLUDE\_PD\_DRP

C

```
#define INCLUDE_PD_DRP 1
```

### Description

Setting the INCLUDE\_PD\_DRP as 1 enables PSF to include USB PD DRP functionality at the compile time. Users can set this define to 0 to reduce code size if none of the PD enabled ports are configured for DRP operation.

### Remarks

Default value is '1' for DRP Application. [INCLUDE\\_PD\\_SOURCE](#) and [INCLUDE\\_PD\\_SINK](#) should be set to 1 as a prerequisite when INCLUDE\_PD\_DRP is set to 1.

### Example

```
#define INCLUDE_PD_DRP 1(Include USB PD DRP functionality in PSF)
#define INCLUDE_PD_DRP 0(Exclude USB PD DRP functionality from PSF)
```

## 8.1.14 INCLUDE\_PD\_PR\_SWAP

C

```
#define INCLUDE_PD_PR_SWAP 1
```

### Description

Setting the INCLUDE\_PD\_PR\_SWAP as 1 enables PSF to include the Power Role Swap (PR\_SWAP) feature at the compile time. PR\_Swap is applicable only for the ports that are configured for DRP operation. Users can set this define to 0 to reduce the code size if none of the DRP ports in the system require Power Role Swap functionality.

### Remarks

Default value is 1 for DRP application. [INCLUDE\\_PD\\_DRP](#) should be set to 1 as a prerequisite when INCLUDE\_PD\_PR\_SWAP is set to 1.

### Example

```
#define INCLUDE_PD_PR_SWAP 1(Include PR_SWAP functionality in PSF)
#define INCLUDE_PD_PR_SWAP 0(Exclude PR_SWAP functionality from PSF)
```



---

## 8.1.15 INCLUDE\_PD\_VDM

C

```
#define INCLUDE_PD_VDM 1
```

### Description

Setting the INCLUDE\_PD\_VDM as 1 enables PSF to include the Structured Vendor Defined Message(VDM) feature at the compile time. Users can set this define to 0 to reduce the code size if none of the ports in the system require Structured VDM support.

### Remarks

Default value is 1.

### Example

```
#define INCLUDE_PD_VDM 1(Include Structured VDM support in PSF)
#define INCLUDE_PD_VDM 0(Exclude Structured VDM support from PSF)
```

---

## 8.1.16 INCLUDE\_PD\_ALT\_MODE

C

```
#define INCLUDE_PD_ALT_MODE 1
```

### Description

Setting the INCLUDE\_PD\_ALT\_MODE as 1 enables PSF to include the Alternate Mode feature at the compile time. Users can set this define to 0 to reduce the code size if none of the ports in the system require Alternate Mode support.

### Remarks

Default value is 1. **INCLUDE\_PD\_VDM** should be set to 1 as a prerequisite when INCLUDE\_PD\_ALT\_MODE is set to 1.

### Example

```
#define INCLUDE_PD_ALT_MODE 1(Include Alternate Mode support in PSF)
#define INCLUDE_PD_ALT_MODE 0(Exclude Alternate Mode support from PSF)
```

---

## 8.1.17 INCLUDE\_UPD\_HPD

C

```
#define INCLUDE_UPD_HPD 1
```

### Description

Setting the INCLUDE\_UPD\_HPD as 1 enables PSF to include the Hot Plug Detect(HPD) feature at the compile time. Users can set this define to 0 to reduce the code size if none of the ports in the system require HPD support.

### Remarks

Default value is 1. **INCLUDE\_PD\_ALT\_MODE** should be set to 1 as a prerequisite when INCLUDE\_UPD\_HPD is set to 1.

### Example

```
#define INCLUDE_UPD_HPD 1(Include Hot Plug Detect support in PSF)
#define INCLUDE_UPD_HPD 0(Exclude Hot Plug Detect support from PSF)
```

## 8.1.18 INCLUDE\_PD\_FR\_SWAP

C

```
#define INCLUDE_PD_FR_SWAP 1
```

### Description

Setting the INCLUDE\_PD\_FR\_SWAP as 1 enables PSF to include the Fast Role Swap (FR\_SWAP) feature at the compile time. FR\_Swap is applicable only for the ports that are configured for DRP operation. Users can set this define to 0 to reduce the code size if none of the DRP ports in the system require Fast Role Swap functionality.

### Remarks



Default value is 1 for DRP application. [INCLUDE\\_PD\\_DRP](#), [INCLUDE\\_PD\\_3\\_0](#), [INCLUDE\\_PD\\_VCONN\\_SWAP](#), [INCLUDE\\_PD\\_DR\\_SWAP](#), [INCLUDE\\_PD\\_PR\\_SWAP](#) and [INCLUDE\\_UPD\\_PIO\\_OVERRIDE\\_SUPPORT](#) should be set to 1 as a prerequisite when INCLUDE\_PD\_FR\_SWAP is set to 1.

### Example

```
#define INCLUDE_PD_FR_SWAP 1(Include FR_SWAP functionality in PSF)
#define INCLUDE_PD_FR_SWAP 0(Exclude FR_SWAP functionality from PSF)
```

## 8.2 System Level Configuration

### Macros

	Name	Description
	<a href="#">CONFIG_PD_PORT_COUNT</a>	Power Delivery Enabled ports count.
	<a href="#">CONFIG_DEFINE_UPD350_HW_INTF_SEL</a>	HW Communication interface between SOC and <a href="#">UPD350</a> .

### Description

The table above lists the overall system level configuration options available at compile time.

### 8.2.1 CONFIG\_PD\_PORT\_COUNT

C

```
#define CONFIG_PD_PORT_COUNT 2
```

### Description

CONFIG\_PD\_PORT\_COUNT defines the number of Power delivery enabled ports. The maximum number of ports PSF can support is '4'.

### Remarks

The max and min value for CONFIG\_PD\_PORT\_COUNT is '4' and '1' respectively. PSF refers the Port number in the call backs as 0 to (CONFIG\_PD\_PORT\_COUNT - 1). The default value is 2 and it can be defined based on the user application. For each port defined by this macro approximately 500Bytes of Data RAM is consumed.

### Example

```
#define CONFIG_PD_PORT_COUNT 2 (Number of PD ports enabled in PSF Stack is 2)
```

## 8.2.2 CONFIG\_DEFINE\_UPD350\_HW\_INTF\_SEL

### C

```
#define CONFIG_DEFINE_UPD350_HW_INTF_SEL CONFIG_UPD350_SPI
```

#### Description

CONFIG\_DEFINE\_UPD350\_HW\_INTF\_SEL defines the Hardware interface for communication between the SOC and [UPD350](#). It can take either CONFIG\_UPD350\_SPI or CONFIG\_UPD350\_I2C as input value.

**CONFIG\_UPD350\_SPI** - SPI is the communication interface between SOC and [UPD350](#). SPI interface is supported by [UPD350](#) B and D parts alone.

**CONFIG\_UPD350\_I2C** - I2C is the communication interface between SOC and [UPD350](#). I2C interface is supported by [UPD350](#) A and C parts alone.

#### Remarks

CONFIG\_DEFINE\_UPD350\_HW\_INTF\_SEL should be defined based on [UPD350](#) silicon part used for the application. All the ports in a system should use either I2C supported or SPI supported [UPD350](#) part. Using mixed interfaces for individual ports is not supported (i.e.: SPI for Port 1 and I2C for Port 2).

If the target for PSF is a UPD301C device, SPI must always be selected. I2C is not an option for UPD301C due to the physical bonding of the UPD301C.

#### Example







```
#define CONFIG_DEFINE_UPD350_HW_INTF_SEL CONFIG_UPD350_SPI
#define CONFIG_DEFINE_UPD350_HW_INTF_SEL CONFIG_UPD350_I2C
```

## 8.3 Configuration and Status Registers


### Macros

	Name	Description
	<a href="#">INCLUDE_CFG_STRUCT_MEMORY_PAD_REGION</a>	Includes reserved bytes in Configuration and Status structure.

### Structures

	Name	Description
	<a href="#">_GlobalCfgStatusData</a>	This structure contains the system level, Port specific configuration and status parameters of PSF. <a href="#">gasCfgStatusData</a> is the defined variable of this structure.
	<a href="#">_PortCfgStatus</a>	This structure contains port specific Type C and PD configuration and status parameters. sPerPortData is referred from <a href="#">_GlobalCfgStatusData</a> .
	<a href="#">_PBPortCfgStatus</a>	This structure contains port specific Power Balancing Configuration parameters. sPBPerPortData is referred from <a href="#">_GlobalCfgStatusData</a> .
	<a href="#">_PPSPortCfgStatus</a>	This structure contains port specific status parameters. sPPSPerPortData is referred from <a href="#">_GlobalCfgStatusData</a> .
	<a href="#">_VDMPortCfgStatus</a>	This structure contains port specific VDM Configuration and Status parameters. sVDMPerPortData is referred from <a href="#">_GlobalCfgStatusData</a> .
	<a href="#">_AltModePortCfgStatus</a>	This structure contains port specific Alternate Mode Configuration and Status parameters. sAltModePerPortData is referred from <a href="#">_GlobalCfgStatusData</a> .

**Variables**

	Name	Description
	<a href="#">gasCfgStatusData</a>	This is an instance of <a href="#">_GlobalCfgStatusData</a> .

**Description**

Configuration and Status registers help users to control the operation of the PSF ports and to know the run time status of all the ports in the system as defined by [CONFIG\\_PD\\_PORT\\_COUNT](#).

## 8.3.1 \_GlobalCfgStatusData Structure

**C**

```

struct _GlobalCfgStatusData {
    UINT8 u8MinorVersion;
    UINT8 u8MajorVersion;
    UINT8 u8HWVersion;
    UINT8 u8SiVersion;
    UINT8 u8aManfString[21];
    UINT8 u8ManfStringLength;
    UINT8 u8PSFMajorVersion;
    UINT8 u8PSFMinorVersion;
    UINT8 u8PwrThrottleCfg;
    UINT8 u8aReserved1[3];
    UINT16 u16ProductID;
    UINT16 u16VendorID;
    PORT_CFG_STATUS sPerPortData[CONFIG_PD_PORT_COUNT];
    UINT16 u16SharedPwrCapacityIn250mW;
    UINT8 u8PBEnableSelect;
    UINT8 u8Reserved1;
    UINT16 u16SystemPowerBankAIn250mW;
    UINT16 u16MinPowerBankAIn250mW;
    UINT16 u16SystemPowerBankBIn250mW;
    UINT16 u16MinPowerBankBIn250mW;
    UINT16 u16SystemPowerBankCIn250mW;
    UINT16 u16MinPowerBankCIn250mW;
    PB_PORT_CFG_STATUS sPBPerPortData[CONFIG_PD_PORT_COUNT];
    PPS_PORT_CFG_STATUS sPPSPerPortData[CONFIG_PD_PORT_COUNT];
    VDM_PORT_CFG_STATUS sVDMPerPortData[CONFIG_PD_PORT_COUNT];
    ALT_MODE_PORT_CFG_STATUS sAltModePerPortData[CONFIG_PD_PORT_COUNT];
    UINT8 u8aReservedPadBytes[16];
};

```

**Description**

This structure contains global configuration and status parameters that are either Integer data types, Bit-Mapped bytes or other data structure.

**1. Members that are Integer data types:**

Name	Size in Bytes	R/W Config time	R/W Run time	Description
u8MinorVersion	1	R/W	R	Defines Structure Minor Version
u8MajorVersion	1	R/W	R	Defines Structure Major Version
u8HWVersion	1	R/W	R	Defines Hardware Version
u8SiVersion	1	R/W	R	Defines Silicon Version
u8aManfString[8]	1	R/W	R	Defines Manufacturer String
u8PSFMajorVersion	1	R/W	R	Defines PSF Stack Major Version

u8PSFMinorVersion	1	R/W	R	Defines PSF Stack Minor Version
u16ProductID	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Product Identifier Value.</li> <li>It is used by the PD Firmware Update state machine during Enumeration phase. This information is shared with the PDFU initiator as part of GET_FW_ID command's response.</li> <li>User definition of this macro is mandatory when <code>INCLUDE_PDFU</code> is defined as 1. It should always be two byte wide.</li> </ul>
u16VendorID	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Vendor Identifier Value.</li> <li>It is used by the PD Firmware Update state machine during Enumeration phase. This information is shared with the PDFU initiator as part of GET_FW_ID command's response.</li> <li>User definition of this macro is mandatory when <code>INCLUDE_PDFU</code> is defined as 1. It should always be two byte wide.</li> </ul>
u16SystemPowerBankAIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Total System Power of Bank A. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x01E0 = 120W</li> <li>0x0320 = 200W</li> </ol> </li> <li>A setting of 0x0000 and 0x0321-0xFFFF is invalid.</li> <li>This variable is used only when either of <code>INCLUDE_POWER_BALANCING</code> or <code>INCLUDE_POWER_THROTTLING</code> is set to '1'.</li> </ul>
u16MinPowerBankAIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Guaranteed minimum Power of Bank A. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x003C = 15W</li> <li>0x0190 = 100W</li> </ol> </li> <li>A setting of 0x0000 and 0x0191-0xFFFF is invalid.</li> <li>This variable is used only when either of <code>INCLUDE_POWER_BALANCING</code> or <code>INCLUDE_POWER_THROTTLING</code> is set to '1'.</li> </ul>
u16SystemPowerBankBIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Total System Power of Bank B. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x01E0 = 120W</li> <li>0x0320 = 200W</li> </ol> </li> <li>A setting of 0x0000 and 0x0321-0xFFFF is invalid.</li> <li>This variable is used only when either of <code>INCLUDE_POWER_BALANCING</code> or <code>INCLUDE_POWER_THROTTLING</code> is set to '1'.</li> </ul>

u16MinPowerBankBIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Guaranteed minimum Power of Bank B. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x003C = 15W</li> <li>0x0190 = 100W</li> </ol> </li> <li>A setting of 0x0000 and 0x0191-0xFFFF is invalid.</li> <li>This variable is used only when either of <u>INCLUDE_POWER_BALANCING</u> or <u>INCLUDE_POWER_THROTTLING</u> is set to '1'.</li> </ul>
u16SystemPowerBankCIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Total System Power of Bank C. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x01E0 = 120W</li> <li>0x0320 = 200W</li> </ol> </li> <li>A setting of 0x0000 and 0x0321-0xFFFF is invalid.</li> <li>This variable is used only when either of <u>INCLUDE_POWER_BALANCING</u> or <u>INCLUDE_POWER_THROTTLING</u> is set to '1'.</li> </ul>
u16MinPowerBankCIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the Guaranteed minimum Power of Bank C. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x003C = 15W</li> <li>0x0190 = 100W</li> </ol> </li> <li>A setting of 0x0000 and 0x0191-0xFFFF is invalid.</li> <li>This variable is used only when either of <u>INCLUDE_POWER_BALANCING</u> or <u>INCLUDE_POWER_THROTTLING</u> is set to '1'.</li> </ul>
u16SharedPwrCapacityIn250mW	2	R	R	<ul style="list-style-type: none"> <li>Defines the currently available shared power capacity from which power is allocated to ports that are not yet in a valid negotiated contract.</li> <li>The shared power capacity is dynamically adjusted as ports are connected and disconnected. Each unit is 0.25W</li> <li>Sample values are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x003C = 15W</li> <li>0x0190 = 100W</li> </ol> </li> <li>This variable is used only when either of <u>INCLUDE_POWER_BALANCING</u> or <u>INCLUDE_POWER_THROTTLING</u> is set to '1'.</li> </ul>
u8aReserved1	3			Reserved
u8Reserved1	1			Reserved

u8aReservedPadBytes[16]	16			<ul style="list-style-type: none"> <li>Reserved bytes included based on configuration macro <a href="#">INCLUDE_CFG_STRUCT_MEMORY_PAD_REGION</a></li> </ul>
-------------------------	----	--	--	---

## 2. Members that are Bit-Mapped bytes:

### a. u8PBEnableSelect:

u8PBEnableSelect defines PB Enable/Disable for the system and also the Power Balancing algorithm. It's size is 1 byte.

Bit	R/W Config time	R/W Run time	Description
3:0	R/W	R	Selection of Power Balancing Algorithm <ul style="list-style-type: none"> <li>0000 = First Come First Serve</li> <li>0001 = Last Plugged Gets Priority</li> <li>0010-1111 = Reserved</li> </ul>
4	R/W	R	Power balancing Enable/Disable for the system <ul style="list-style-type: none"> <li>0 = PD Balancing is disabled</li> <li>1 = PD Balancing is enabled</li> </ul>
7:5			Reserved

### b. u8PwrThrottleCfg:

u8PwrThrottleCfg defines the Power Throttle Enable/Disable configuration and currently selected Power Bank. It's size is 1 byte.

Bit	R/W Config time	R/W Run time	Description
0	R/W	R	Power Throttle Enable/Disable for the system <ul style="list-style-type: none"> <li>0 = Disable Power Throttling</li> <li>1 = Enable Power Throttling</li> </ul>
2:1	R/W	R/W	Selection of Power Throttling Bank <ul style="list-style-type: none"> <li>00 = Bank A</li> <li>01 = Bank B</li> <li>10 = Bank C</li> <li>11 = Shutdown mode</li> </ul>
7:3			Reserved

## 3. Members that are another structures:

Structure	Description
sPerPortData	Includes Type C and PD parameters of a port, like default Source PDOs, default Sink PDOs, currently negotiated voltage and current values, under voltage and over voltage threshold values, etc., Tag for this structure is <a href="#">_PortCfgStatus</a> .
sPPSPerPortData	Includes PPS parameters of a port, like partner alert and status. Tag for this structure is <a href="#">_PPSPortCfgStatus</a> .
sPBPerPortData	Includes Power Balancing parameters of a port, like maximum power and maximum current. Tag for this structure is <a href="#">_PBPortCfgStatus</a> .

**Remarks**

None

## 8.3.2 \_PortCfgStatus Structure

**C**

```

struct _PortCfgStatus {
    UINT32  u32CfgData;
    UINT32  u32aAdvertisedPDO[7];
    UINT32  u32aPartnerSourcePDO[7];
    UINT32  u32aPartnerSinkPDO[7];
    UINT32  u32RDO;
    UINT32  u32PortConnectStatus;
    UINT32  u32PortStatusChange;
    UINT32  u32PortIOStatus;
    UINT32  u32ClientRequest;
    UINT16  u16AllocatedPowerIn250mW;
    UINT16  u16NegoVoltageInmV;
    UINT16  u16NegoCurrentInmA;
    UINT16  u16MaxSrcPrtCurrentIn10mA;
    UINT16  u16PortIntrMask;
    UINT16  u16PowerGoodTimerInms;
    UINT16  u16FeatureSelect;
    UINT16  u16SwapPolicy;
    UINT16  u16aMinPDOPreferredCurInmA[7];
    UINT16  u16SnkMaxOperatingCurInmA;
    UINT16  u16SnkMinOperatingCurInmA;
    UINT16  u16DAC_I_MaxOutVoltInmV;
    UINT16  u16DAC_I_MinOutVoltInmV;
    UINT16  u16DAC_I_CurrentInd_MaxInA;
    UINT8   u8AdvertisedPDOCnt;
    UINT8   u8PartnerSinkPDOCnt;
    UINT8   u8PartnerSourcePDOCnt;
    UINT8   u8SinkConfigSel;
    UINT8   u8FaultInDebounceInms;
    UINT8   u8OCSThresholdPercentage;
    UINT8   u8OVThresholdPercentage;
    UINT8   u8UVThresholdPercentage;
    UINT8   u8VCONNOCSDebounceInms;
    UINT8   u8VBUSMaxFaultCnt;
    UINT8   u8VCONNMaxFaultCnt;
    UINT8   u8Pio_FAULT_IN;
    UINT8   u8Mode_FAULT_IN;
    UINT8   u8aReserved2[3];
    UINT32  u32aSourcePDO[7];
    UINT32  u32aNewSourcePDO[7];
    UINT32  u32aCableIdentity[6];
    UINT8   u8SourcePDOCnt;
    UINT8   u8NewSourcePDOCnt;
    UINT8   u8CableIdentityCnt;
    UINT8   u8Pio_EN_VBUS;
    UINT8   u8Mode_EN_VBUS;
    UINT8   u8aReserved3[3];
    UINT32  u32aSinkPDO[7];
    UINT32  u32aNewSinkPDO[7];
    UINT8   u8SinkPDOCnt;
    UINT8   u8NewSinkPDOCnt;
    UINT8   u8Pio_EN_SINK;
    UINT8   u8Mode_EN_SINK;
    UINT8   u8DAC_I_Direction;
    UINT8   u8aSinkCapsExtd[21];
    UINT8   u8aReserved4[2];
    UINT8   u8PIO_HPD;
    UINT8   u8aReserved5[3];
    UINT8   u8Pio_EN_FRS;

```



```

UINT8 u8Mode_EN_FRS;
UINT8 u8aReserved8[2];
UINT8 u8aReservedPortPadBytes[32];
};

```

### Description

This structure contains global configuration and status parameters that are either Integer data types, Bit-Mapped bytes or other data structure.

#### 1. Members that are Integer data types:

Name	Size in Bytes	R/W Config time	R/W Run time	Description
u32aSourcePDO[7]	28	R/W	R	<ul style="list-style-type: none"> <li>Source Capabilities array holding maximum of 7 Data Objects including Fixed PDOs and PPS APDOs.</li> <li>This array should be used only for Source Operation.</li> </ul>
u32aSinkPDO[7]	28	R/W	R	<ul style="list-style-type: none"> <li>Upto 7 fixed Sink PDOs where Voltage is specified in mV and Current is specified in mA.</li> <li>This array should be used only when the port is configured as Sink.</li> </ul>
u32aNewSourcePDO[7]	28	R/W	R/W	<ul style="list-style-type: none"> <li>New Source Capabilities array holding maximum of 7 Data Objects including Fixed PDOs and PPS APDOs.</li> <li>This array is applicable only when the port acts as Source.</li> </ul>
u32aNewSinkPDO[7]	28	R/W	R/W	<ul style="list-style-type: none"> <li>New Sink Capabilities array holding maximum of 7 fixed Sink PDOs where voltage is specified in mV and Current is specified in mA.</li> <li>This array is applicable only when the port acts as Sink.</li> </ul>
u32aAdvertisedPDO[7]	28	R	R	<ul style="list-style-type: none"> <li>Upto 7 PDOs that are advertised to Port Partner.</li> <li>During run time, when the port acts as source, this array holds the value of current u32aNewSourcePDO[7] if Bit 10 of u32CfgData is enabled else holds the value of current u32aSourcePDO[7].</li> <li>During run time, when the port acts as sink, this array holds the value of current u32aNewSinkPDO[7] if Bit 10 of u32CfgData is enabled else holds the value of current u32aSinkPDO[7].</li> </ul>
u32aPartnerSourcePDO[7]	28	R	R	<ul style="list-style-type: none"> <li>Upto 7 fixed Partner's Source PDOs where Voltage is specified in mV and Current is specified in mA</li> <li>This array is specific for Sink functionality.</li> </ul>
u32aPartnerSinkPDO[7]	28	R	R	<ul style="list-style-type: none"> <li>Upto 7 fixed Partner's Sink PDOs where Voltage is specified in mV and Current is specified in mA</li> <li>This array is common for Source and Sink functionality.</li> </ul>
u32aCableIdentity[6]	24	R	R	<ul style="list-style-type: none"> <li>Cable Identity array holding the VDM Data Objects received from cable where Index 0 corresponds to ID Header VDO, Index 1 being Cert Stat VDO, Index 2 being Product VDO and indices 3-5 correspond to Product Type VDO(s)</li> </ul>

u32RDO	4	R	R	<ul style="list-style-type: none"> <li>Complete raw RDO Data as sent to the port partner when acting as Sink and Complete raw RDO Data as requested by connected port partner when acting as Source.</li> <li>Will be blank if no RDO has been issued/received.</li> <li>This variable is common for Source and Sink.</li> </ul>
u16AllocatedPowerIn250mW	2	R	R	<ul style="list-style-type: none"> <li>Allocated Power for the Port PD contract in 0.25W steps</li> </ul>
u16NegoVoltageInmV	2	R	R	<ul style="list-style-type: none"> <li>Negotiated Voltage in mV steps.</li> <li>When acting as Source and in Fixed power supply contract, it holds the value of bits 19:10 of PDO in mV.</li> <li>When acting as Source and in Programmable Power Supply contract, it holds the value of bits 19:9 of RDO in mV.</li> <li>When acting as Sink, it holds the value of bits 19:10 of PDO in mV.</li> <li>This variable is common for both Source and Sink.</li> </ul>
u16NegoCurrentInmA	2	R	R	<ul style="list-style-type: none"> <li>Negotiated Current in mA steps.</li> <li>When acting as Source and in Fixed power supply contract, it holds the value of bits 9:0 of PDO in mV.</li> <li>When acting as Source and in Programmable Power Supply contract, it holds the value of bits 6:0 of RDO in mV.</li> <li>When acting as Sink, it holds the value of bits 9:0 of PDO in mV.</li> <li>This variable is common for both Source and Sink.</li> </ul>
u16MaxSrcPrtCurrentIn10mA	2	R/W	R	<ul style="list-style-type: none"> <li>Maximum allowable current for ports in 10mA steps</li> <li>Sample values this variable can take are,               <ol style="list-style-type: none"> <li>0x0032 = 0.5A</li> <li>0x012C = 3A</li> <li>0x01F4 = 5A</li> </ol> </li> <li>Note : Values above 5A (0x01F5 - 0x0FFF) are not supported</li> </ul>
u16SnkMaxOperatingCurlInmA	2	R/W	R	<ul style="list-style-type: none"> <li>Maximum allowable current or system's maximum operating current in terms of mA.</li> <li>This variable is applicable only when acting as Sink.</li> </ul>
u16aMinPDOPreferredCurlInmA[7]	14	R/W	R	<ul style="list-style-type: none"> <li>Preferred minimum current range for the PDO by which the Sink may select without setting Capability Mismatch Bit with highest current preferred.</li> <li>This array is applicable only for Sink Operation.</li> </ul>
u16SnkMinOperatingCurlInmA	2	R/W	R	<ul style="list-style-type: none"> <li>Minimum current required by the sink hardware to be operational.</li> <li>This variable is applicable only for Sink Operation.</li> <li>When a Gotomin message is issued by source, sink reduces its operating current to the value provided in this variable.</li> </ul>

u16DAC_I_MaxOutVoltInmV	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the maximum voltage on DAC_I with a maximum of 2.5V in terms of mV</li> <li>This is applicable only for Sink operation.</li> </ul>
u16DAC_I_MinOutVoltInmV	2	R/W	R	<ul style="list-style-type: none"> <li>Defines the minimum voltage on DAC_I with a minimum of 0V in terms of mV</li> <li>This is applicable only for Sink operation.</li> </ul>
u16DAC_I_CurrentInd_MaxInA	2	R/W	R	<ul style="list-style-type: none"> <li>Defines which current in terms of mA corresponding to maximum output voltage</li> <li>It can take either 3A or 5A value.</li> <li>If it is 5A and maximum output voltage is 2.5V and if direction mentioned in u8DAC_I_Direction is High Amperage - Max Voltage, then               <ol style="list-style-type: none"> <li>0.5A &gt; DAC_I = 0.25V</li> <li>1.5A &gt; DAC_I = 0.75V</li> <li>2.0A &gt; DAC_I = 1V</li> <li>3.0A &gt; DAC_I = 1.5V</li> <li>4.0A &gt; DAC_I = 2.0V</li> <li>5.0A &gt; DAC_I = 2.5V</li> </ol> </li> <li>If it is 3A and maximum output voltage is 2.5V, then               <ol style="list-style-type: none"> <li>0.5A &gt; DAC_I = 0.42V</li> <li>1.5A &gt; DAC_I = 1.25V</li> <li>2.0A &gt; DAC_I = 1.67V</li> <li>3.0A &gt; DAC_I = 2.5V</li> <li>4.0A &gt; DAC_I = 2.5V</li> <li>5.0A &gt; DAC_I = 2.5V</li> </ol> </li> <li>This is applicable only for Sink operation.</li> </ul>
u16PowerGoodTimerInms	2	R/W	R	<ul style="list-style-type: none"> <li>After an automatic fault recovery, u16PowerGoodTimerInms is run to determine whether power remains in a good state for the duration of the timer, then the Fault Counter is reset. If another fault occurs before the Power Good Timer expires, then the Fault Counter is incremented.</li> <li>For power Source, it is the time a power source must consistently provide power without a fault to determine the power is good and a fault condition does not exist. For power Sink, it is the time after the sink established a contract and its consistently drawing power from VBUS without a power fault to determine that power is good and a fault condition does not exist.</li> </ul>
u8SourcePDOCnt	1	R/W	R	<ul style="list-style-type: none"> <li>Number of Default Source PDOs supported.</li> <li>This variable is applicable only when the port is configured as Source.</li> </ul>
u8SinkPDOCnt	1	R/W	R	<ul style="list-style-type: none"> <li>Number of Default Sink PDOs supported.</li> <li>This variable is applicable only when the port is configured as Sink.</li> </ul>

u8NewSourcePDOCnt	1	R/W	R/W	<ul style="list-style-type: none"> <li>Number of New Source PDOs Supported.</li> <li>This variable is applicable only when the port acts as Source.</li> </ul>
u8NewSinkPDOCnt	1	R/W	R/W	<ul style="list-style-type: none"> <li>Number of New Sink PDOs Supported.</li> <li>This variable is applicable only when the port acts as Sink.</li> </ul>
u8AdvertisedPDOCnt	1	R	R	<ul style="list-style-type: none"> <li>Number of PDOs advertised to port partner.</li> </ul>
u8PartnerSourcePDOCnt	1	R	R	<ul style="list-style-type: none"> <li>Number of Source PDOs received from port partner.</li> <li>This variable is specific for Sink functionality.</li> </ul>
u8PartnerSinkPDOCnt	1	R	R	<ul style="list-style-type: none"> <li>Number of Sink PDOs received from port partner.</li> <li>This variable is common for Source and Sink functionalities.</li> </ul>
u8CableIdentityCnt	1	R	R	<ul style="list-style-type: none"> <li>Number of VDM Data Objects received from cable.</li> </ul>
u8SinkConfigSel	1	R/W	R	<ul style="list-style-type: none"> <li>BIT[1:0] - Sink Selection mode for operation.               <ol style="list-style-type: none"> <li>'0x00' Mode A: Prefer Higher Voltage and Wattage</li> <li>'0x01' Mode B: Prefer Lower Voltage and Wattage</li> </ol> </li> <li>BIT2 - No USB Suspend Flag               <ol style="list-style-type: none"> <li>'1':Set the flag to '1' in RDO request</li> <li>'0':Set the flag to '0' in RDO request</li> </ol> </li> <li>BIT3 - GiveBackFlag               <ol style="list-style-type: none"> <li>'1':Set the flag to '1' in RDO request enabling GotoMin feature</li> <li>Set the flag to '0' in RDO request disabling GotoMin Feature</li> </ol> </li> </ul>
u8FaultInDebounceInms	1	R/W	R	<ul style="list-style-type: none"> <li>Debounce timer value in terms of milliseconds for VBUS overcurrent fault conditions before reacting and entering fault recovery routine.</li> <li>It is applicable only for OCS detection via FAULT_IN configured <a href="#">UPD350</a> pin.</li> </ul>
u8OCSThresholdPercentage	1	R/W	R	<ul style="list-style-type: none"> <li>OCS Threshold. Reserved for future use. This variable is not currently used by PSF.</li> </ul>

u8OVThresholdPercentage	1	R/W	R	<ul style="list-style-type: none"> <li>Percentage of PDO voltage to be considered as Over Voltage for that PDO. As per PD specification, desired range for fixed PDO voltage is <math>(0.95 * \text{PDO Voltage})</math> to <math>(1.05 * \text{PDO Voltage})</math>, So u8OVThresholdPercentage should be greater than the desired range.</li> <li>If 115% of the PDO voltage has to be considered as overvoltage for that PDO voltage, then define u8OVThresholdPercentage as 115.</li> <li>It is mandatory to define u8OVThresholdPercentage when <b>INCLUDE_POWER_FAULT_HANDLING</b> is defined as '1'.</li> </ul>
u8UVThresholdPercentage	1	R/W	R	<ul style="list-style-type: none"> <li>Percentage of PDO voltage to be considered as under Voltage for that PDO. As per PD specification, desired range for fixed PDO voltage is <math>(0.95 * \text{PDO Voltage})</math> to <math>(1.05 * \text{PDO Voltage})</math>, So u8UVThresholdPercentage should be less than the desired range.</li> <li>If 85% of the PDO voltage has to be considered as under voltage for that PDO voltage, then define u8UVThresholdPercentage as 85.</li> <li>u8UVThresholdPercentage must be defined when <b>INCLUDE_POWER_FAULT_HANDLING</b> is defined as '1'. As an exceptional case this factor is not considered for VSafe5V.</li> </ul>
u8VCONNOCSDebounceInms	1	R/W	R	<ul style="list-style-type: none"> <li>Debounce timer value in terms of milliseconds for VCONN overcurrent fault conditions before reacting and entering fault recovery routine.</li> </ul>
u8VBUSMaxFaultCnt	1	R/W	R	<ul style="list-style-type: none"> <li>The maximum number of back-to-back VBUS faults allowed before permanent shut down of the port. A back-to-back fault is a second fault which occurs within the u16PowerGoodTimerInms after a port is automatically reenabled from a previous fault condition.</li> <li>During port shutdown due to over current fault, the device removes its CC termination and wait for port partner to get detached physically from the port to resume its normal operation.</li> </ul>
u8VCONNMaxFaultCnt	1	R/W	R	<ul style="list-style-type: none"> <li>The maximum number of back-to-back VCONN faults allowed before it permanently disables the VCONN. A back-to-back fault is a second fault which occurs within the u16PowerGoodTimerInms after a port is automatically reenabled from a previous fault condition.</li> <li>If VCONN is disabled due to over current VCONN power fault, VCONN will be enabled only after a physical detach and reattach.</li> </ul>

u8Pio_EN_VBUS	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the <a href="#">UPD350</a> PIO number used for EN_VBUS pin functionality for the Port.</li> <li>This variable is applicable only for source operation.</li> <li>EN_VBUS is to enable VBUS drive out of DC/DC controller. EN_VBUS pin connects to a load switch device such as a power FET or load switch IC. It is driven as per u8Mode_EN_VBUS configuration mode whenever stack requires VBUS to driven high as well as low.</li> <li>The range of valid values is 0 to 9 which correspond to <a href="#">UPD350</a> PIO0 to PIO9.</li> <li>To disable the pin functionality from the stack, the user can define a value of 0xFF.</li> <li>By defining <a href="#">INCLUDE_UPD_PIO_OVERRIDE_SUPPORT</a> as '1', the PIO Override feature of the <a href="#">UPD350</a> shall be utilized in this pin to ensure that fast and autonomous action is taken by the <a href="#">UPD350</a> in a fault condition.</li> </ul>
u8Mode_EN_VBUS	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the PIO mode of the <a href="#">UPD350</a> PIO EN_VBUS defined in u8Pio_EN_VBUS.</li> <li>This variable is applicable only for source operation.</li> <li>It takes values only from enum <a href="#">eUPD_OUTPUT_PIN_MODES_TYPE</a>.</li> </ul>
u8Pio_FAULT_IN	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the UPD PIO used as FAULT_IN pin.</li> <li>FAULT_IN pin detects over current fault or under/over voltage fault from external sensing device based on its configuration u8Mode_FAULT_IN.</li> <li>It can take values only from 0 to 15 and to disable the pin functionality from stack, user can define it as 0xFF.</li> <li>It is applicable only when <a href="#">INCLUDE_POWER_FAULT_HANDLING</a> defined as '1'.</li> </ul>
u8Mode_FAULT_IN	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the PIO mode of the <a href="#">UPD350</a> PIO FAULT_IN defined in u8Pio_FAULT_IN.</li> <li>It takes values only from enum <a href="#">eUPD_INPUT_PIN_MODES_TYPE</a></li> </ul>

u8Pio_EN_SINK	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the <a href="#">UPD350</a> PIO number used for EN_SINK pin.</li> <li>This is applicable only for Sink operation.</li> <li>This pin is asserted in the following conditions: <ol style="list-style-type: none"> <li>If the source supports Power delivery, the PD negotiated current should be greater than or equal to the current mentioned under u16SnkMinOperatingCurlnmA variable.</li> <li>If the source does not support Power delivery and is Type-C only, the Rp value in source partner should be greater than or equal to the current mentioned under u16SnkMinOperatingCurlnmA variable.</li> </ol> </li> <li>This pin is de-asserted during a hard reset, a power fault recovery or a detach.</li> <li>The range of valid values is 0 to 9 which correspond to <a href="#">UPD350</a> PIO0 to PIO9.</li> <li>By defining <a href="#">INCLUDE_UPD_PIO_OVERRIDE_SUPPORT</a> as '1', the PIO Override feature of the <a href="#">UPD350</a> shall be utilized in this pin to ensure that fast and autonomous action is taken by the <a href="#">UPD350</a> in a fault condition.</li> </ul>
u8Mode_EN_SINK	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the PIO mode for EN_SINK pin</li> <li>This is applicable only for Sink operation.</li> <li>It takes values only from enum <a href="#">eUPD_OUTPUT_PIN_MODES_TYPE</a>.</li> </ul>
u8DAC_I_Direction	1	R/W	R	<ul style="list-style-type: none"> <li>Specifies the direction of DAC_I to allow user invert direction of DAC_I if required <ol style="list-style-type: none"> <li>0 - High Amperage - Max Voltage</li> <li>1- High Amperage - Min Voltage</li> </ol> </li> <li>This is applicable only for Sink operation.</li> </ul>
u8aSinkCapsExtd[21]	21	R/W	R	<ul style="list-style-type: none"> <li>21-byte Sink Capabilities Extended Data Block that needs to be sent in response to a Get_Sink_Cap_Extended Message</li> <li>The contents of the array shall comply with Table 6-60 Sink Capabilities Extended Data Block (SKEDB) of USB PD 3.0 Specification</li> <li>This array shall be used only when the port is configured as Sink or DRP</li> </ul>
u8PIO_HPD	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the <a href="#">UPD350</a> PIO number used for HPD IO pin.</li> <li>The state of this pin is tracked in u16HPDStatus variable.</li> <li>This is applicable only when <a href="#">INCLUDE_UPD_HPD</a> is enabled.</li> </ul>

u8Pio_EN_FRS	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the <a href="#">UPD350</a> PIO number used for FRS Enable pin functionality for the port</li> <li>When the power/data state of the port is configured as Power Source/Data Device, this PIO acts an input pin to <a href="#">UPD350</a> and is used to trigger FRS request signaling upon detection of loss of power in the port</li> <li>When the power/data state of the port is configured as Power Sink/Data Host, this PIO acts an output pin to <a href="#">UPD350</a> and is used to arm external circuitry for FRS operation (such as a smart load switch with FRS capability)</li> <li>This variable is applicable only when <a href="#">INCLUDE_PD_FR_SWAP</a> is enabled</li> <li>The range of valid values is 0 to 9 which correspond to <a href="#">UPD350</a> PIO0 to PIO9 and to disable the pin functionality, user can define it as 0xFF</li> <li>By defining <a href="#">INCLUDE_UPD_PIO_OVERRIDE_SUPPORT</a> as '1', the PIO Override feature of the <a href="#">UPD350</a> shall be utilized in this pin to ensure that fast and autonomous action is taken by the <a href="#">UPD350</a> in a Fast Role Swap condition.</li> </ul>
u8Mode_EN_FRS	1	R/W	R	<ul style="list-style-type: none"> <li>Defines the PIO mode of the <a href="#">UPD350</a> PIO EN_FRS defined in u8Pio_EN_FRS</li> <li>This variable is applicable only when <a href="#">INCLUDE_PD_FR_SWAP</a> is enabled</li> <li>When u8Pio_EN_FRS is configured as an input pin, it takes values from enum <a href="#">eUPD_INPUT_PIN_MODES_TYPE</a></li> <li>When u8Pio_EN_FRS is configured as an output pin, it takes values from enum <a href="#">eUPD_OUTPUT_PIN_MODES_TYPE</a></li> </ul>
u8aReserved2	3			Reserved
u8aReserved3	3			Reserved
u8aReserved4	2			Reserved
u8aReserved5	3			Reserved
u8aReserved8	2			Reserved
u8aReservedPortPadBytes[32]	32			<ul style="list-style-type: none"> <li>Reserved bytes included based on configuration macro <a href="#">INCLUDE_CFG_STRUCT_MEMORY_PAD_REGION</a></li> </ul>

## 2. Members that are Bit-Mapped bytes:

### a. u32CfgData:

u32CfgData variable holds the Port Configuration Data. It's size is 4 bytes.

Bit	R/W Config time	R/W Run time	Description
1:0	RW	R	Port Power Role <ul style="list-style-type: none"> <li>'00' Sink</li> <li>'01' Source</li> <li>'10' DRP</li> </ul>



2	RW	R	Dual Role Data Capability <ul style="list-style-type: none"> <li>• '0' No Dual Role Data</li> <li>• '1' Dual Role Data supported</li> </ul>
4:3	RW	R	Rp Selection <ul style="list-style-type: none"> <li>• '00' Disabled</li> <li>• '01' USB Power</li> <li>• '10' 1.5A</li> <li>• '11' 3.0A</li> </ul>
5	RW	R	Port Enable/Disable <ul style="list-style-type: none"> <li>• '0' Disabled</li> <li>• '1' Enabled</li> </ul>
8:6	RW	R	USB Data <ul style="list-style-type: none"> <li>• '000' No Data</li> <li>• '001' USB2</li> <li>• '010' USB3.1 Gen1</li> <li>• '011' USB3.1 Gen2</li> </ul>
9	RW	R	VCONN OCS Enable <ul style="list-style-type: none"> <li>• '0' Disable</li> <li>• '1' Enable</li> </ul>
10	RW	R	Use New PDOs for negotiation <ul style="list-style-type: none"> <li>• '0' Default PDOs provided in <code>gasCfgStatusData.sPerPortData[u8PortNum].u32aSourcePDO</code> or <code>gasCfgStatusData.sPerPortData[u8PortNum].u32aSinkPDO</code> will be used depending on the current power role for power negotiation.</li> <li>• '1' New PDOs provided in <code>gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSourcePDO</code> or <code>gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSinkPDO</code> will be used depending on the current power role for power negotiation.</li> </ul> <p>The first PD negotiation will take place with default PDOs. After the first power negotiation, if user wants PD negotiation to happen with new PDOs, the user must ensure that new PDOs (<code>gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSourcePDO</code> or <code>gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSinkPDO</code> arrays) are configured depending on the current power role and then set this bit. Then, further power negotiations will happen based on new PDOs. After power negotiation with new PDOs, if user wants further PD negotiations to happen with default PDOs (<code>gasCfgStatusData.sPerPortData[u8PortNum].u32aSourcePDO</code> or <code>gasCfgStatusData.sPerPortData[u8PortNum].u32aSinkPDO</code>), this bit can be cleared.</p>

12:11	RW	R	Power/Data State for initiating FRS <ul style="list-style-type: none"> <li>'00' FRS is disabled for the port</li> <li>'01' FRS will be initiated only when the Power/Data state of the port is Power Sink/Data Host</li> <li>'10' FRS will be initiated only when the Power/Data state of the port is Power Source/Data Device</li> <li>'11' Reserved</li> <li>Note: These bits are applicable only when the Port Power Role is set to DRP and <b>INCLUDE_PD_FR_SWAP</b> is set to 1</li> <li>User Application shall ensure that the 'Fast Role Swap required USB Type-C current' field (Bits 23 and 24) is not set to 0 in gasCfgStatusData.sPerPortData[u8PortNum].u32aSinkPDO[0] and VCONN Swap, PR Swap DR Swap Policy bits in gasCfgStatusData.sPerPortData[u8PortNum].u16SwapPolicy variable are configured in alignment with the FRS Power/Data State bits</li> </ul>
32:13			Reserved

**b. u32PortConnectStatus:** u32PortConnectStatus variable holds the connection status of the port. It's size is 4 bytes.

Bit	R/W Config time	R/W Run time	Description
0	R	R	Attached <ul style="list-style-type: none"> <li>'0' Detached</li> <li>'1' Attached</li> </ul>
1	R	R	Orientation <ul style="list-style-type: none"> <li>'0' Unflipped - Port Partner attached in CC1 pin</li> <li>'1' Flipped - Port Partner attached in CC2 pin</li> </ul>
3:2	R	R	Data Role <ul style="list-style-type: none"> <li>'00' UFP</li> <li>'01' DFP</li> <li>'10' Toggling</li> <li>'11' Reserved</li> </ul>
5:4	R	R	Power Role <ul style="list-style-type: none"> <li>'00' Sink</li> <li>'01' Source</li> <li>'10' DRP</li> <li>'11' Reserved</li> </ul>
6	R	R	VCONN Status <ul style="list-style-type: none"> <li>'0' Disabled</li> <li>'1' Enabled</li> </ul>
7	R	R	Cable Reduced Source Capabilities <ul style="list-style-type: none"> <li>'0' Attached USB-C cable supports the locally-defined Source PDOs</li> <li>'1' Attached USB-C cable does not support the locally defined Source PDOs</li> </ul>

8	R	R	<p>Reduced Source Capabilities</p> <ul style="list-style-type: none"> <li>'0' The advertised PDOs are equivalent to the default configured values</li> <li>'1' The advertised PDOs have been reduced from default configured values</li> </ul>
9	R	R	<p>Source Capability Mismatch</p> <ul style="list-style-type: none"> <li>'0' De-asserted by Source port when there is capability mismatch with sink partner</li> <li>'1' Asserted by Source port when sink port indicates capability mismatch in RDO</li> </ul>
10	R	R	<p>As Source PD Contract Good</p> <ul style="list-style-type: none"> <li>'0' As Source: USB-C Connection Only (No Request Made Yet)</li> <li>'1' As Source; USB PD connection established, Power request has been made, accepted and PS_RDY message sent.</li> <li>This bit will always remain 0 when acting as sink.</li> </ul>
11	R	R	<p>As Source RDO Accepted</p> <ul style="list-style-type: none"> <li>'0' As Source: No RDO Accept message has been sent to last Request made by attached Sink or no Request has yet been made during connection.</li> <li>'1' As Source: RDO Accept message has been sent to last Request made by attached Sink</li> <li>This bit will always remain 0 when acting as sink</li> </ul>
12	R	R	<p>As Source RDO Rejected</p> <ul style="list-style-type: none"> <li>'0' As source; No RDO reject message has been sent to last request made by attached Sink or no Request has yet been made during connection</li> <li>'1' As Source: RDO Reject message has been sent to last Request made by attached Sink</li> <li>This bit will always remain 0 when acting as Sink</li> </ul>
13	R	R	<p>As Sink Last Request Accept</p> <ul style="list-style-type: none"> <li>'0' As Sink: Last RDO Request was not Accepted or no request has yet been made.</li> <li>'1' As Sink: Last RDO Request was Accepted</li> <li>This bit will always remain 0 when acting as a source.</li> </ul>
14	R	R	<p>As Sink Last Request Reject</p> <ul style="list-style-type: none"> <li>'0' As Sink: Last RDO Request was not Rejected or no request has yet been made.</li> <li>'1' As Sink: Last RDO Request was Rejected</li> <li>This bit will always remain 0 when acting as a source.</li> </ul>
15	R	R	<p>As Sink Last Request PS_RDY</p> <ul style="list-style-type: none"> <li>'0' As Sink: PS_RDY not yet received for last RDO request</li> <li>'1' As Sink: PS_RDY received for last RDO request</li> <li>This bit will always remain 0 when acting as a source.</li> </ul>
16	R	R	<p>Sink Capability Mismatch</p> <ul style="list-style-type: none"> <li>'0' De-asserted by the Sink Port when there is no capability mismatch</li> <li>'1' Asserted by Sink Port when no Source capability was found</li> </ul>
18:17	R	R	<p>Rp Value detected by Sink</p> <ul style="list-style-type: none"> <li>'00' Disabled</li> <li>'01' USB Power</li> <li>'10' 1.5A</li> <li>'11' 3.0A</li> </ul>

20:19	R	R	Current Negotiated PD Specification Revision <ul style="list-style-type: none"> <li>• '01' PD2.0</li> <li>• '10' PD3.0</li> <li>• '00' &amp; '11' - Reserved</li> </ul>
31:21			Reserved

**c. u32PortIOStatus:** u32PortIOStatus variable holds the IO status of the port. It's size is 4 bytes.

Bit	R/W Config time	R/W Run time	Description
0	R	R	EN_DC_DC Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
1	R	R	VSEL0 Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
2	R	R	VSEL1 Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
3	R	R	VSEL2 Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
4	R	R	EN_VBUS Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
5	R	R	VBUS_DIS Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
6	R	R	EN_SINK Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
7	R	R	1.5_IND Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
8	R	R	3.0_IND Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>
9	R	R	Capability Mismatch Status <ul style="list-style-type: none"> <li>• '1' Asserted</li> <li>• '0' De-asserted</li> </ul>

10	R	R	Power role Status <ul style="list-style-type: none"> <li>'1' Asserted if Source</li> <li>'0' De-asserted if Sink</li> <li>Applicable only for DRP configuration</li> </ul>
11	R	R	Data role Status <ul style="list-style-type: none"> <li>'1' Asserted if DFP</li> <li>'0' De-asserted if UFP</li> <li>Applicable only for DRP configuration</li> </ul>
12	R	R	EN_FRS Status <ul style="list-style-type: none"> <li>'1' Asserted</li> <li>'0' De-asserted</li> <li>Applicable only if EN_FRS is configured as output</li> </ul>
31:13			Reserved

**d. u32PortStatusChange:** u32PortStatusChange variable defines the port connection status change bits. It's size is 4 bytes.

Bit	R/W Config time	R/W Run time	Description
0	R	R	Attach Event <ul style="list-style-type: none"> <li>'0' Since the last read of this register, PSF has not experienced a USB-C attach</li> <li>'1' Since the last read of this register, PSF has experienced a USB-C attach</li> </ul>
1	R	R	Detach Event <ul style="list-style-type: none"> <li>'0' Since the last read of this register, PSF has not experienced a USB-C detach</li> <li>'1' Since the last read of this register, PSF has experienced a USB-C detach</li> </ul>
2	R	R	As Source New Request <ul style="list-style-type: none"> <li>'0' As Source, since the last read of this register, PSF has not received any new PDO request from attached port partner</li> <li>'1' As Source: Received a new PDO request the attached Sink port partner, As Sink: Received an updated set of Source capabilities form the attached Source port partner</li> </ul>
3	R	R	As Sink New PDOs Received <ul style="list-style-type: none"> <li>'0' As Sink: Since the last read of this register, PSF has not received any changed source capabilities</li> <li>'1' As Sink: Received an updated set of Source capabilities form the attached Source port partner</li> </ul>
4	R	R	As Sink New Request Sent <ul style="list-style-type: none"> <li>'0' As Sink: Since the last read of this register, PSF has not sent any additional Sink RDOs</li> <li>'1' As Sink: Since the last read of this register, PSF has issued a new Sink RDO to the attached Source,</li> <li>This bit will always remain 0 when acting as a source</li> </ul>

5	R	R	As Sink Last Request Accept <ul style="list-style-type: none"> <li>'0' As Sink: Since the last read of this register, PSF has not received any new Request Accept messages from the attached Source</li> <li>'1' As Sink: Since the last read of this register, PSF has received a new Request Accept from the attached Source</li> </ul>
6	R	R	As Sink Last Request Reject <ul style="list-style-type: none"> <li>'0' As Sink: Since the last read of this register, PSF has not received any new Request Reject messages from the attached Source</li> <li>'1' As Sink: Since the last read of this register, PSF has received a new Request Reject message from the attached Source</li> </ul>
7	R	R	As Sink Last Sink PS Rdy <ul style="list-style-type: none"> <li>'0' As Sink: Since the last read of this register, PSF has not received any PS_RDY messages from the attached Source</li> <li>'1' As Sink: Since the last read of this register, PSF has received a PS_RDY message from the attached Source</li> </ul>
8	R	R	Hard Reset Event <ul style="list-style-type: none"> <li>'0' Since the last read of this register, PSF has not experienced a USB PD Hard Reset</li> <li>'1' Since the last read of this register, PSF has experienced a USB PD Hard Reset</li> </ul>
9	R	R	Pin Reset Event <ul style="list-style-type: none"> <li>'0' Since the last read of this register, PSF has not been reset via POR or pin</li> <li>'1' Since the last read of this register, PSF has been reset via POR or pin</li> </ul>
10	R	R	VBUS Fault <ul style="list-style-type: none"> <li>'0' Since the last read of this register, no VBUS faults have been detected</li> <li>'1' Since the last read of this register, 1 or more VBUS faults have been detected</li> </ul>
11	R	R	VCONN Fault <ul style="list-style-type: none"> <li>'0' Since the last read of this register, no VCONN faults have been detected</li> <li>'1' Since the last read of this register, 1 or more VCONN faults have been detected</li> </ul>
31:12			Reserved

**e. u32ClientRequest:** u32ClientRequest variable defines the client request mask bits. It's size is 4 bytes. USER\_APPLICATION can set the corresponding bit in this variable to request PSF to handle the client requests mentioned in the table below. Only one client request can be handled by PSF at a given time. If more than one client request bits are set at the same time, the requests will be queued internally and processed based on the priority of events where bit 0 takes the highest priority and bit 31 takes the least priority. If the request is accepted and processed, a response notification will be posted by PSF as mentioned in the below table.

Bit	R/W Config time	R/W Run time	Description
0	R/W	R/W	Port Disable Request <ul style="list-style-type: none"> <li>Set this bit to request PSF to disable a port.</li> <li>This client request will be processed by PSF irrespective of whether it is idle.</li> <li>Once a port is disabled successfully, eMCHP_PSF_PORT_DISABLED notification will be posted by PSF to user application.</li> </ul>

1	R/W	R/W	<p>Port Enable Request</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to enable a port.</li> <li>This client request will be processed by PSF irrespective of whether it is idle.</li> <li>Once a port is disabled successfully, eMCHP_PSF_PORT_ENABLED notification will be posted by PSF to user application.</li> </ul>
2	R/W	R/W	<p>Handle VBUS Power Fault Over voltage Request</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to process externally detected over voltage VBUS fault.</li> </ul>
3	R/W	R/W	<p>Handle VBUS Power Fault Over Current Request</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to process externally detected over current VBUS power fault or to inform PSF that Current Limit mode is entered by external DC-DC controller.</li> </ul>
4	R/W	R/W	<p>Handle VBUS Power Fault Over Current Exit Request</p> <ul style="list-style-type: none"> <li>Set this bit to inform PSF that externally detected over current VBUS power fault condition is exited or Constant Voltage mode is entered by external DC-DC controller.</li> </ul>
5	R/W	R/W	<p>Respond to a Vendor Defined Message</p> <ul style="list-style-type: none"> <li>Set this bit to respond to a VDM request received from port partner</li> <li>Reception of a VDM request from partner will be notified through eMCHP_PSF_VDM_REQUEST_RCVD notification. VDM Header, received from partner will be stored in u32PartnerVDMHeader variable of sVDMPerPortData structure, VDOs and number of VDOs will be stored in u32aPartnerVDO array and u8PartnerVDOCnt variable of sAltModePerPortData structure respectively.</li> <li>User Application shall ensure that the VDM Header is configured in u32VDMHeader variable of sVDMPerPortData structure and VDOs in the u32aVDO array and VDOs count in u8VDOCnt of sAltModePerPortData structure while raising this client request</li> <li>eMCHP_PSF_VDM_AMS_COMPLETE notification will be posted when Good CRC is received for the response sent</li> <li>This client request is supported only when <b>INCLUDE_PD_ALT_MODE</b> is defined as '1'.</li> </ul>
6	R/W	R/W	<p>Renegotiation Request</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to trigger renegotiation</li> <li>User application may request PSF to renegotiate based on default PDOs (gasCfgStatusData.sPerPortData[u8PortNum].u32aSourcePDO or gasCfgStatusData.sPerPortData[u8PortNum].u32aSinkPDO) or New PDOs (gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSourcePDO or gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSinkPDO). To renegotiate with default PDOs, user application must ensure that BIT(10) in gasCfgStatusData.sPerPortData[u8PortNum].u32CfgData variable is cleared and then BIT(6) in gasCfgStatusData.sPerPortData[u8PortNum].u32ClientRequest is set. To renegotiate with New PDOs, user application must ensure that the New PDOs (gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSourcePDO or gasCfgStatusData.sPerPortData[u8PortNum].u32aNewSinkPDO) are configured, BIT(10) in gasCfgStatusData.sPerPortData[u8PortNum].u32CfgData variable is set and BIT(6) in gasCfgStatusData.sPerPortData[u8PortNum].u32ClientRequest is set</li> <li>Once the request is processed by PSF, eMCHP_PSF_PD_CONTRACT_NEGOTIATED notification will be posted.</li> <li>Note: This client request is not applicable when Power Balancing or Power Throttling is enabled. Therefore, the user application should not trigger this client request when Power Balancing or Power Throttling is enabled.</li> </ul>

7	R/W	R/W	<p>Initiate a VCONN Swap</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to initiate a VCONN Swap</li> <li>User Application shall ensure that either of the VCONN Swap Policy bits (Bit 8 or Bit 9) is set in gasCfgStatusData.sPerPortData[u8PortNum].u16SwapPolicy variable while raising this client request</li> <li>eMCHP_PSF_VCONN_SWAP_COMPLETE notification will be posted for an Accept/Reject response, and eMCHP_PSF_VCONN_SWAP_NO_RESPONSE_RCVD will be posted when there is no response for VCONN Swap from the partner</li> <li>This client request is supported only when <b>INCLUDE_PD_VCONN_SWAP</b> is defined as '1'.</li> </ul>
8	R/W	R/W	<p>Initiate a Power Role Swap</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to initiate a PR_Swap</li> <li>User Application shall ensure that either of the PR_Swap Policy bits (Bit 4 or Bit 5) is set in gasCfgStatusData.sPerPortData[u8PortNum].u16SwapPolicy variable while raising this client request</li> <li>eMCHP_PSF_PR_SWAP_COMPLETE notification will be posted for an Accept/Reject response, and eMCHP_PSF_PR_SWAP_NO_RESPONSE_RCVD will be posted when there is no response for the PR_Swap from the partner</li> <li>This client request is supported only when <b>INCLUDE_PD_PR_SWAP</b> is defined as '1'.</li> </ul>
9	R/W	R/W	<p>Initiate a Data Role Swap</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to initiate a DR_Swap</li> <li>User Application shall ensure that either of the DR_Swap policy bits (Bit 0 or Bit 1) is set in gasCfgStatusData.sPerPortData[u8PortNum].u16SwapPolicy variable while raising this client request</li> <li>eMCHP_PSF_DR_SWAP_COMPLETE notification will be posted for an Accept/Reject response, and eMCHP_PSF_DR_SWAP_NO_RESPONSE_RCVD will be posted when there is no response for the DR_Swap from the partner</li> <li>This client request is supported only when <b>INCLUDE_PD_DR_SWAP</b> is defined as '1'.</li> </ul>
10	R/W	R/W	<p>Initiate a Structured Vendor Defined Message</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to initiate a VDM request to port partner</li> <li>User Application shall ensure that the VDM Header is configured in u32VDMHeader variable of sVDMPerPortData structure and VDOs in the u32aVDO array and VDOs count in u8VDOCnt of sAltModePerPortData structure while raising this client request</li> <li>eMCHP_PSF_VDM_RESPONSE_RCVD notification will be posted for an ACK/NAK response, and eMCHP_PSF_VDM_RESPONSE_NOT_RCVD will be posted when no response is received.</li> <li>eMCHP_PSF_VDM_AMS_COMPLETE notification will be posted when Good CRC is received for the VDM requests that does not consist of a command response, say Attention command</li> <li>This client request is supported only when <b>INCLUDE_PD_VDM</b> is defined as '1'.</li> </ul>
11	R/W	R/W	<p>Hot Plug Detect (HPD) disable request</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to disable HPD support.</li> <li>This client request is supported only when <b>INCLUDE_UPD_HPD</b> is defined as '1'.</li> <li>eMCHP_PSF_HPD_DISABLED notification will be posted once HPD peripheral is disabled.</li> </ul>
12	R/W	R/W	<p>Hot Plug Detect (HPD) enable request</p> <ul style="list-style-type: none"> <li>Set this bit to request PSF to enable HPD support.</li> <li>This client request is supported only when <b>INCLUDE_UPD_HPD</b> is defined as '1'.</li> <li>eMCHP_PSF_HPD_ENABLED notification will be posted once HPD peripheral is enabled.</li> </ul>
31:13			Reserved



**f. u16PortIntrMask:** u16PortIntrMask variable defines the port interrupt mask bits. It's size is 2 bytes.

Bit	R/W Config time	R/W Run time	Description
0	RC	RC	Attach Event Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
1	RC	RC	Detach Event Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
2	RC	RC	As Source New Request Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
3	RC	RC	As Sink New PDOs Received Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
4	RC	RC	As Sink New Request Sent Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
5	RC	RC	As Sink Last Request Accept Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
6	RC	RC	As Sink Last Request Reject Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
7	RC	RC	As Sink Last Sink PS RDY Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
8	RC	RC	Hard Reset Event Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
9	RC	RC	Pin Reset Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
10	RC	RC	VBUS Fault Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>
11	RC	RC	VCONN Fault Mask <ul style="list-style-type: none"> <li>'0' Do not mask interrupt pin toggle on changes to this event.</li> <li>'1' Mask this event from generating interrupt pin toggle.</li> </ul>

15:12			Reserved
-------	--	--	----------

**g. u16FeatureSelect:** u16FeatureSelect variable defines the enable/disable of various PSF features. It's size is 2 bytes.

Bit	R/W Config time	R/W Run time	Description
0	R/W	R	Power Balancing Enable/Disable <ul style="list-style-type: none"> <li>'0' Disable.</li> <li>'1' Enable.</li> </ul> This bit is applicable only for source operation.
1	R/W	R	Indicates if Vendor Defined Message is supported by the port. <ul style="list-style-type: none"> <li>'0' - Vendor Defined Messages are not supported. The port will respond with 'Not Supported' on reception of a Vendor Defined Message.</li> <li>'1' - Vendor Defined Messages are supported. The port will respond with relevant VDM data on reception of a Vendor Defined Message.</li> </ul>
2	R/W	R	AltMode Entry Monitoring Enable/Disable <ul style="list-style-type: none"> <li>'0' - The port will not monitor AltMode Entry</li> <li>'1' - The port will monitor AltMode Entry and indicate the failure when AMETimer times out</li> </ul>
15:3			Reserved

**h. u16SwapPolicy:** u16SwapPolicy defines the policy of a port whether to accept, request or reject Power Role Swap, Data Role Swap and VCONN Swap based on its power and data roles.

Bit	R/W Config time	R/W Run time	Description
0	R/W	R/W	EN_AUTO_DR_SWAP_REQUEST_AS_DFP <ul style="list-style-type: none"> <li>'0' Disable Auto Data Role Request When Data Role is DFP</li> <li>'1' Enable Auto Data Role Request when Data Role is DFP</li> </ul>
1	R/W	R/W	EN_AUTO_DR_SWAP_REQUEST_AS_UFP <ul style="list-style-type: none"> <li>'0' Disable Auto Data Role Request When Data Role is UFP</li> <li>'1' Enable Auto Data Role Request when Data Role is UFP</li> </ul>
2	R/W	R/W	EN_AUTO_DR_SWAP_ACCEPT_AS_DFP <ul style="list-style-type: none"> <li>'0' Disable Auto Data Role Accept When Data Role is DFP</li> <li>'1' Enable Auto Data Role Accept when Data Role is DFP</li> </ul>
3	R/W	R/W	EN_AUTO_DR_SWAP_ACCEPT_AS_UFP <ul style="list-style-type: none"> <li>'0' Disable Auto Data Role Accept When Data Role is UFP</li> <li>'1' Enable Auto Data Role Accept when Data Role is UFP</li> </ul>
4	R/W	R/W	EN_AUTO_PR_SWAP_REQUEST_AS_SOURCE <ul style="list-style-type: none"> <li>'0' Disable Auto Power Role Request When Power Role is Source</li> <li>'1' Enable Auto Power Role Request when Power Role is Source</li> </ul>
5	R/W	R/W	EN_AUTO_PR_SWAP_REQUEST_AS_SINK <ul style="list-style-type: none"> <li>'0' Disable Auto Power Role Request When Power Role is Sink</li> <li>'1' Enable Auto Power Role Request when Power Role is Sink</li> </ul>

6	R/W	R/W	EN_AUTO_PR_SWAP_ACCEPT_AS_SOURCE <ul style="list-style-type: none"> <li>'0' Disable Auto Power Role Accept When Power Role is Source</li> <li>'1' Enable Auto Power Role Accept when Power Role is Source</li> </ul>
7	R/W	R/W	EN_AUTO_PR_SWAP_ACCEPT_AS_SINK <ul style="list-style-type: none"> <li>'0' Disable Auto Power Role Accept When Power Role is Sink</li> <li>'1' Enable Auto Power Role Accept when Power Role is Sink</li> </ul>
8	R/W	R/W	EN_AUTO_VCONN_SWAP_REQ_AS_VCONN_SRC <ul style="list-style-type: none"> <li>'0' Disable Auto VCONN Swap Request When acting as VCONN Source</li> <li>'1' Enable Auto VCONN Swap Request When acting as VCONN Source</li> <li>Note: This bit shall be set to 1 when Power/Data state of the FRS port is set to Power Source/Data UFP to enable PSF in initiating a VCONN Swap before an FRS</li> </ul>
9	R/W	R/W	EN_AUTO_VCONN_SWAP_REQ_AS_NOT_VCONN_SRC <ul style="list-style-type: none"> <li>'0' Disable Auto VCONN Swap Request When not acting as VCONN Source</li> <li>'1' Enable Auto VCONN Swap Request When not acting as VCONN Source</li> <li>Note: This bit shall be set to 1 when Power/Data state of the FRS port is set to Power Sink/Data DFP to enable PSF in initiating a VCONN Swap before an FRS</li> </ul>
10	R/W	R/W	EN_AUTO_VCONN_SWAP_ACCEPT_AS_VCONN_SRC <ul style="list-style-type: none"> <li>'0' Disable Auto VCONN Swap Accept When acting as VCONN Source</li> <li>'1' Enable Auto VCONN Swap Accept When acting as VCONN Source</li> <li>Note: This bit shall be set to 1 always to comply with the PD spec for VCONN Swap request</li> </ul>
11	R/W	R/W	EN_AUTO_VCONN_SWAP_ACCEPT_AS_NOT_VCONN_SRC <ul style="list-style-type: none"> <li>'0' Disable Auto VCONN Swap Accept When not acting as VCONN Source</li> <li>'1' Enable Auto VCONN Swap Accept When not acting as VCONN Source</li> <li>Note: This bit shall be set to 1 when Power/Data state of the FRS port is set to Power Sink/Data DFP to enable PSF in accepting a VCONN Swap before an FRS</li> </ul>
15:12			Reserved

**Remarks**

None

## 8.3.3 \_PBPortCfgStatus Structure

C

```

struct _PBPortCfgStatus {
    UINT16 u16MaxPrtPwrBankAIn250mW;
    UINT16 u16MaxPrtPwrBankBIn250mW;
    UINT16 u16MaxPrtPwrBankCIn250mW;
    UINT8  u8PortPriority;
    UINT8  u8Reserved2;
};

```

**Description**

This structure contains global configuration and status parameters that are either Integer data types, Bit-Mapped bytes or other data structure. This structure is used only when either of the macros `INCLUDE_POWER_BALANCING` or `INCLUDE_POWER_THROTTLING` is set to '1'.

**1. Members that are Integer data types:**

Name	Size in Bytes	R/W Config time	R/W Run time	Description
u16MaxPrtPwrBankAIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Maximum Port Power Bank A in 0.25W steps.</li> <li>Unit/LSB = 0.25W</li> <li>Sample values this variable can take are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x00F0 = 60W</li> <li>0x0190 = 100W</li> </ol> </li> <li>Note : A setting of 0x0000 and 0x191-0xFFFF is invalid.</li> </ul>
u16MaxPrtPwrBankBIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Maximum Port Power Bank B in 0.25W steps.</li> <li>Unit/LSB = 0.25W</li> <li>Sample values this variable can take are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x00F0 = 60W</li> <li>0x0190 = 100W</li> </ol> </li> <li>Note : A setting of 0x0000 and 0x191-0xFFFF is invalid.</li> </ul>
u16MaxPrtPwrBankCIn250mW	2	R/W	R	<ul style="list-style-type: none"> <li>Maximum Port Power Bank A in 0.25W steps.</li> <li>Unit/LSB = 0.25W</li> <li>Sample values this variable can take are,               <ol style="list-style-type: none"> <li>0x0001 = 0.25W</li> <li>0x00F0 = 60W</li> <li>0x0190 = 100W</li> </ol> </li> <li>Note : A setting of 0x0000 and 0x191-0xFFFF is invalid.</li> </ul>
u8PortPriority	1	R/W	R	<ul style="list-style-type: none"> <li>Selects the port priority</li> <li>000b is the highest priority</li> </ul>
u8Reserved2	1			Reserved

**Remarks**

None

**8.3.4 \_PPSPortCfgStatus Structure****C**

```

struct _PPSPortCfgStatus {
    UINT32 u32PartnerAlert;
    UINT8 u8aPartnerStatus[6];
    UINT8 u8aReserved6[2];
};

```

**Description**

This structure contains the following status parameters that are either Integer Datatypes or Bit-Mapped bytes. This structure is used only when **INCLUDE\_PD\_SOURCE\_PPS** is set to '1'.

**1. Members that are Integer Datatypes:**

Name	Size in Bytes	R/W Config time	R/W Run time	Description
u32PartnerAlert	4	R	R	<ul style="list-style-type: none"> <li>Complete Alert information received from Partner, Will be blank if no Alert has been received.</li> <li>This variable is common for Source and Sink.</li> </ul>
u8aPartnerStatus[6]	6	R	R	<ul style="list-style-type: none"> <li>Store the Status information received from Port Partner.</li> <li>This array would hold a valid value if eMCHP_PSF_SINK_STATUS_RCVD notification is posted. It would be 0 when eMCHP_PSF_SINK_STATUS_NOT_RCVD notification is posted.</li> </ul>
u8aReserved6[2]	2			Reserved

**Remarks**

None

## 8.3.5 \_VDMPortCfgStatus Structure

**C**

```

struct _VDMPortCfgStatus {
    UINT32 u32VDMHeader;
    UINT32 u32PartnerVDMHeader;
    UINT32 u32aPartnerPDIdentity[6];
    UINT32 u32aPDIdentity[6];
    UINT8 u8PDIdentityCnt;
    UINT8 u8PartnerPDIdentityCnt;
    UINT8 u8aReserved7[2];
};

```

**Description**

This structure contains the following parameters that are either Integer Datatypes or Bit-Mapped bytes. This structure is used only when **INCLUDE\_PD\_VDM** is set to '1'.

**1. Members that are Integer Datatypes:**

Name	Size in Bytes	R/W Config time	R/W Run time	Description
u32VDMHeader	4	R/W	R/W	<ul style="list-style-type: none"> <li>VDM Header used while sending a VDM to port partner</li> <li>The fields of this variable shall comply with Table 6-25: Structured VDM Header of PD Specification</li> </ul>
u32PartnerVDMHeader	4	R	R	<ul style="list-style-type: none"> <li>VDM Header sent by partner</li> </ul>

u32aPartnerPDIdentity[6]	24	R	R	<ul style="list-style-type: none"> <li>Partner Identities received in response to a Discover Identity request. This array can hold upto 6 VDM Data Objects where Index 0 corresponds to ID Header VDO, Index 1 being Cert Stat VDO, Index 2 being Product VDO and indices 3-5 correspond to 0-3 Product Type VDO(s)</li> </ul>
u32aPDIdentity[6]	24	R/W	R	<ul style="list-style-type: none"> <li>Port PD Identities to be sent in Discover Identity response. This array can hold upto 6 VDM Data Objects where Index 0 corresponds to ID Header VDO, Index 1 being Cert Stat VDO, Index 2 being Product VDO and indices 3-5 correspond to 0-3 Product Type VDO(s)</li> </ul>
u8PartnerPDIdentityCnt	1	R	R	<ul style="list-style-type: none"> <li>Number of Identities received from partner in response to a Discover Identity request</li> </ul>
u8PDIdentityCnt	1	R/W	R	<ul style="list-style-type: none"> <li>Number of PD Identities of the port that needs to be sent in response to a Discover Identity request</li> </ul>
u8aReserved7	2			Reserved

**Remarks**

None

## 8.3.6 \_AltModePortCfgStatus Structure

**C**

```

struct _AltModePortCfgStatus {
    UINT32 u32aModesTable[16];
    UINT32 u32aVDO[6];
    UINT32 u32aPartnerVDO[6];
    UINT16 u16aSVIDsTable[16];
    UINT8 u8aSVIDEntryTable[16];
    UINT8 u8SVIDsCnt;
    UINT8 u8VDOCnt;
    UINT8 u8PartnerVDOCnt;
    UINT8 u8Reserved3;
};

```

**Description**

This structure contains the following parameters that are either Integer Datatypes or Bit-Mapped bytes. This structure is used only when `INCLUDE_PD_ALT_MODE` is set to '1'.

**1. Members that are Integer Datatypes:**

Name	Size in Bytes	R/W Config time	R/W Run time	Description
u32aModesTable	64	R/W	R	<ul style="list-style-type: none"> <li>List of Modes corresponding to each supported SVID</li> </ul>
u32aVDO	24	R/W	R/W	<ul style="list-style-type: none"> <li>This array contains VDOs received from partner during an Enter Mode request and other SVID specific commands.</li> <li>Application can make use of this array to send the VDOs to partner while initiating or responding to Enter Mode and other SVID specific commands</li> </ul>
u32aPartnerVDO	24	R	R	<ul style="list-style-type: none"> <li>This array contains VDOs received from partner during an Enter Mode request and other SVID specific commands.</li> </ul>

u16aSVIDsTable	32	R/W	R	<ul style="list-style-type: none"> <li>List of SVIDs supported by the port</li> </ul>
u8aSVIDEntryTable	16	R/W	R	<ul style="list-style-type: none"> <li>SVID Entry table where in every index,</li> <li>Bits 2:0 - No of Modes for an SVID</li> <li>Bits 6:3 - Start Mode Index The index into the mode table for the first mode for this SVID. The allocation of modes in the table starts from this field up to the value in No of Modes.</li> <li>Bit 7 - Reserved</li> </ul>
u8SVIDsCnt	1	R/W	R	<ul style="list-style-type: none"> <li>Number of entries stored in u16aSVIDsTable</li> </ul>
u8VDOCnt	1	R/W	R/W	<ul style="list-style-type: none"> <li>Number of VDOs to be sent from u32aVDO</li> </ul>
u8PartnerVDOCnt	1	R	R	<ul style="list-style-type: none"> <li>Number of VDOs stored in u32aPartnerVDO which are received from partner</li> </ul>
u8Reserved3	1			Reserved

**Remarks**

None

## 8.3.7 gasCfgStatusData Variable

**C**

```
GLOBAL_CFG_STATUS_DATA gasCfgStatusData;
```

**Description**

gasCfgStatusData is the global structure which defines the overall system level configuration and status parameters of PSF. This structure contains the system level and port specific Configuration and Status parameters of PSF including Type C, PD, PB, PT and PPS parameters.

It is mandatory that the user has to initialize the configuration parameters for the PSF stack to function properly. This can be done through [MCHP\\_PSF\\_HOOK\\_BOOT\\_TIME\\_CONFIG](#) which initializes the parameters defined in gasCfgStatusData during compile time. For accessing the configuration registers and reading the status registers at run time, an I2C slave interface shall be used by the user application.

**Remarks**

None

## 8.3.8 INCLUDE\_CFG\_STRUCT\_MEMORY\_PAD\_REGION

**C**

```
#define INCLUDE_CFG_STRUCT_MEMORY_PAD_REGION 0
```

**Description**

INCLUDE\_CFG\_STRUCT\_MEMORY\_PAD\_REGION will define the reserved bytes in the config and status register structure, so that expansion of structure members in future can be handled without change in the address of the existing member elements.

**Remarks**

The default value is 0 and it can be defined to 1 based on the user application needs.

**Example**

```
#define INCLUDE_CFG_STRUCT_MEMORY_PAD_REGION 0
```

## 8.4 DC-DC Buck Boost Controller Configuration

This section describes all the configuration options available in PSF for controlling the DC-DC buck boost modules.

**NOTE:** The PSF controls EN\_VBUS, EN\_SINK and FAULT\_IN functions as these require autonomous operation. This is ensured by using the PIO override capability of [UPD350](#). All other PSF functions are available for configuration by the user.

### 8.4.1 Power Control GPIO Enumeration constants

**Enumerations**

	Name	Description
	<a href="#">eUPD_OUTPUT_PIN_MODES_TYPE</a>	<a href="#">UPD350</a> GPIO Output mode enum.

**Description**

This section describes the enumeration constants provided by PSF for configuring the various Port Power Control parameters such as Mode and PIO number of EN\_VBUS pin.

#### 8.4.1.1 eUPD\_OUTPUT\_PIN\_MODES\_TYPE Enumeration

**C**

```
typedef enum {
    ePUSH_PULL_ACTIVE_HIGH = 0x0CU,
    ePUSH_PULL_ACTIVE_LOW = 0x04U,
    eOPEN_DRAIN_ACTIVE_HIGH = 0x08U,
    eOPEN_DRAIN_ACTIVE_LOW = 0x00U,
    eOPEN_DRAIN_ACTIVE_HIGH_PU = 0x88U,
    eOPEN_DRAIN_ACTIVE_LOW_PU = 0x80U
} eUPD_OUTPUT_PIN_MODES_TYPE;
```

**Description**

eUPD\_OUTPUT\_PIN\_MODES\_TYPE enum defines the various combination modes applicable for [UPD350](#) GPIO in output mode. This is applicable only for EN\_SINK and EN\_VBUS functions.

**Members**

Members	Description
ePUSH_PULL_ACTIVE_HIGH = 0x0CU	Active High output signal
ePUSH_PULL_ACTIVE_LOW = 0x04U	Active low output signal
eOPEN_DRAIN_ACTIVE_HIGH = 0x08U	Active High Open Drain output signal
eOPEN_DRAIN_ACTIVE_LOW = 0x00U	Active Low Open Drain output signal
eOPEN_DRAIN_ACTIVE_HIGH_PU = 0x88U	Active High Open Drain output signal with internal pull up
eOPEN_DRAIN_ACTIVE_LOW_PU = 0x80U	Active Low Open Drain output signal with internal pull up



**Remarks**

None

## 8.5 Debug Hook Configuration

**Macros**

	Name	Description
	<a href="#"><u>CONFIG_HOOK_DEBUG_MSG</u></a>	Print status messages from PSF stack through UART interface

### 8.5.1 CONFIG\_HOOK\_DEBUG\_MSG

**C**

```
#define CONFIG_HOOK_DEBUG_MSG 0
```

**Description**

Setting CONFIG\_HOOK\_DEBUG\_MSG to 1, prints status messages from PSF stack through UART interface.

**Remarks**

The following hook APIs should be defined with appropriate UART functions to view status messages from PSF stack.

1. [MCHP\\_PSF\\_HOOK\\_DEBUG\\_INIT\(\)](#)
2. [MCHP\\_PSF\\_HOOK\\_PRINT\\_CHAR](#)(byData)
3. [MCHP\\_PSF\\_HOOK\\_PRINT\\_INTEGER](#)(dwWriteInt, byWidth)
4. [MCHP\\_PSF\\_HOOK\\_PRINT\\_TRACE](#)(pbyMessage)










None.

**Example**

```
#define CONFIG_HOOK_DEBUG_MSG 0
#define CONFIG_HOOK_DEBUG_MSG 1
```

## 8.6 PDFU Configuration

**Macros**

	Name	Description
	<a href="#"><u>CONFIG_PDFU_SUPPORTED</u></a>	Power Delivery Firmware Update Supported field.
	<a href="#"><u>CONFIG_PDFU_VIA_USBDP_SUPPORTED</u></a>	Power Delivery Firmware Update Supported via USB config.
	<a href="#"><u>CONFIG_MAX_FIRMWARE_IMAGESIZE</u></a>	Maximum Firmware image size.
	<a href="#"><u>CONFIG_RECONFIG_PHASE_WAITTIME</u></a>	Reconfig phase wait time value.
	<a href="#"><u>CONFIG_TRANSFER_PHASE_WAITTIME</u></a>	Transfer phase wait time value.
	<a href="#"><u>CONFIG_UPDATABLE_IMAGEBANK_INDEX</u></a>	Index of Updatable image.
	<a href="#"><u>CONFIG_VALIDATION_PHASE_WAITTIME</u></a>	Validation phase wait time value.
	<a href="#"><u>CONFIG_HWMAJOR_VERSION</u></a>	Hardware Major Version.
	<a href="#"><u>CONFIG_HWMINOR_VERSION</u></a>	Hardware Minor Version.

**Description**

This section explains the configurable options available for PDFU.

---

## 8.6.1 CONFIG\_PDFU\_SUPPORTED

**C**

```
#define CONFIG_PDFU_SUPPORTED 1
```

**Description**

CONFIG\_PDFU\_SUPPORTED is set to '0' if firmware is not updatable during Run time. Otherwise shall be set to 1. It is used by the PD Firmware Update state-machine during Enumeration phase. This information is shared with the PDFU Initiator as part of GET\_FW\_ID command's response.

**Remarks**

The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is '1'. By default, it is defined as '1'.

**Example**

```
#define CONFIG_PDFU_SUPPORTED 1
```

---

## 8.6.2 CONFIG\_PDFU\_VIA\_USBDPD\_SUPPORTED

**C**

```
#define CONFIG_PDFU_VIA_USBDPD_SUPPORTED 1
```

**Description**

CONFIG\_PDFU\_VIA\_USBDPD\_SUPPORTED Set to '1' to indicate support for PDFU via USB PD Firmware Update flow. Otherwise shall be set to '0'. It is used by the PD Firmware Update state-machine during Enumeration phase. This information is shared with the PDFU Initiator as part of GET\_FW\_ID command's response.

**Remarks**

The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is '1'. The default value is '1'.

**Example**

```
#define CONFIG_PDFU_VIA_USBDPD_SUPPORTED 1
```

---

## 8.6.3 CONFIG\_MAX\_FIRMWARE\_IMAGESIZE

**C**

```
#define CONFIG_MAX_FIRMWARE_IMAGESIZE 0x8800UL
```

**Description**

CONFIG\_MAX\_FIRMWARE\_IMAGESIZE defines the ROM size allocated for the Updatable application. PDFU Initiator shall flash entire size during every re-flash operation. Flashing lesser or more than this Size results in error response.

**Remarks**

Choose Firmware Image size in such a way that integral multiple of 256. The definition of this function is mandatory when [INCLUDE\\_PDFU](#) is '1' and shall expressed in terms of bytes. By default, the value is 0x8800UL(34KB).

**Example**

```
#define CONFIG_MAX_FIRMWARE_IMAGESIZE 0x9800UL (38*1024 bytes for 38KB Updatable
```

application).

## 8.6.4 CONFIG\_RECONFIG\_PHASE\_WAITTIME

C

```
#define CONFIG_RECONFIG_PHASE_WAITTIME 0x00u
```

### Description

CONFIG\_RECONFIG\_PHASE\_WAITTIME specifies the Wait time required for the Reconfigure state, i.e. the PDFU\_Initiate request processing takes "Wait time" millisecond, and next request can be issued by the PDFU\_Initiator after the specified wait time. This information is shared with the PDFU Initiator as part of PDFU\_INITIATE command's response.

### Remarks

The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is '1'. It can have values from 0x00 to 0xFF. By default, it is defined as '0x00'.

### Example

```
#define CONFIG_RECONFIG_PHASE_WAITTIME 0x03u //3ms wait time required
```

## 8.6.5 CONFIG\_TRANSFER\_PHASE\_WAITTIME

C

```
#define CONFIG_TRANSFER_PHASE_WAITTIME 0x64u
```

### Description

CONFIG\_TRANSFER\_PHASE\_WAITTIME Species the Wait time required during the Transfer state, i.e. the PDFU Data request processing takes "Wait time" millisecond, and next PDFU\_DATA request to be issued by the initiator after the specified wait time. This information is shared with the PDFU Initiator as part of PDFU\_DATA command's response.

### Remarks

The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is '1'. It can have values from 0x00 to 0xFF. By default, it is defined as '0x03'.

### Example

```
#define CONFIG_TRANSFER_PHASE_WAITTIME 0x03u //3ms required for processing PDFU_DATA request
```

## 8.6.6 CONFIG\_UPDATABLE\_IMAGEBANK\_INDEX

C

```
#define CONFIG_UPDATABLE_IMAGEBANK_INDEX 0x03u
```

### Description

CONFIG\_UPDATABLE\_IMAGEBANK\_INDEX specifies the Image bank index for which firmware upgrade is requested (or) in other words it corresponds to the image bank index of the Updatable application as mentioned by Architecture 2 of PD FW Update Specification.

This information is used during the Reconfiguration phase to determine what application is currently executing and whether application switching to Fixed Application is required or not.

### Remarks

The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is '1'. By default, this macro is defined as 0x03.

**Example**

```
#define CONFIG_UPDATABLE_IMAGEBANK_INDEX    0x03u (3 image banks are available)
```

---

## 8.6.7 CONFIG\_VALIDATION\_PHASE\_WAITTIME

**C**

```
#define CONFIG_VALIDATION_PHASE_WAITTIME 0x03u
```

**Description**

CONFIG\_VALIDATION\_PHASE\_WAITTIME specifies the wait time macro for the validation state, i.e. the PDFU\_Validate command's processing takes "Wait time" millisecond, and next request can be issued by the Initiator after the specified wait time.

**Remarks**

The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is '1'. It can have values from 0x00 to 0xFF. By default, it is defined as '0x03'.

**Example**

```
#define CONFIG_VALIDATION_PHASE_WAITTIME    0x03u
```

---

## 8.6.8 CONFIG\_HWMAJOR\_VERSION

**C**

```
#define CONFIG_HWMAJOR_VERSION
```

**Description**

CONFIG\_HWMAJOR\_VERSION defines Hardware Major Version details of the product. It is used by the PD Firmware Update state-machine during Enumeration phase. This information is shared with the PDFU Initiator as part of GET\_FW\_ID command's response.

**Remarks**

This is a 4-bit entity. (Valid values are 0x0 to 0xF). The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is defined as '1'.

**Example**

```
#define CONFIG_HWMAJOR_VERSION    0x1
```

---

## 8.6.9 CONFIG\_HWMINOR\_VERSION

**C**

```
#define CONFIG_HWMINOR_VERSION
```

**Description**

[CONFIG\\_HWMAJOR\\_VERSION](#) defines Hardware Minor Version details of the product. It is used by the PD Firmware Update state-machine during Enumeration phase. This information is shared with the PDFU Initiator as part of GET\_FW\_ID command's response.

**Remarks**

This is a 4-bit entity. (Valid values are 0x0 to 0xF). The user definition of this macro is mandatory when [INCLUDE\\_PDFU](#) is defined as '1'.

**Example**

```
#define CONFIG_HWMINOR_VERSION    0x0
```

## 9 System level integration of PSF

As PSF is hardware agnostic, it can be ported to many Microchip SoC platforms. This section covers:

- Hardware and Software requirements for system level integration of PSF
- Steps for integrating PSF with an existing application or system
- APIs to be implemented for integrating PSF
- Notification callback from PSF

All the APIs required for PSF porting and integration are available in PSF\_APIHooks.h header file. It is recommended to include this PSF\_APIHooks.h file path as one of preprocessor include path. The template of PSF\_APIHooks.h for new SoC integration is available under ..\PSF\SOC\_Portable\New\_SOC.

### 9.1 Hardware Requirements

This section describes the hardware requirements of the SoC for running PSF.

#### 9.1.1 UPD350

The UPD350 is a companion Power Delivery Port Controller that provides cable plug orientation and detection for a USB Type-C receptacle. It implements baseband communication with a partner Type-C device on the opposite side of the cable.

The UPD350 communicates to an external SoC using the integrated I2C/SPI interface. One UPD350 per port is required to achieve USB-PD functionality in each port.

PSF supports the following versions of UPD350 SKUs

Device	Hardware communication Interface
UPD350-A	I2C
UPD350-B	SPI
UPD350-C	I2C
UPD350-D	SPI

PSF requires the following hardware support from SoC to control UPD350:

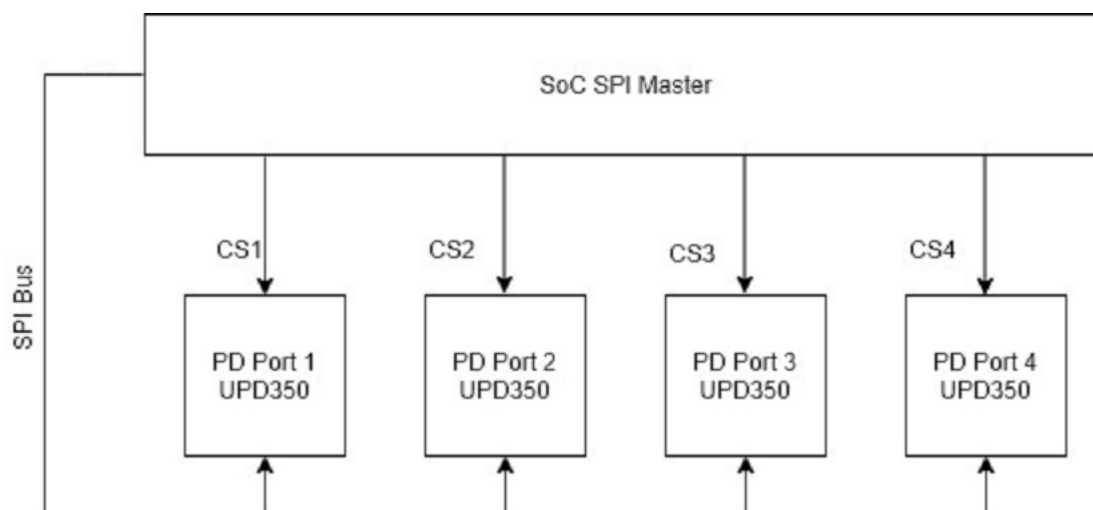
- Hardware communication interface either I2C or SPI to interact with UPD350
- PIOs for each port's UPD350 to detect IRQ alert
- PIO to reset the UPD350s

##### 9.1.1.1 Hardware Communication Interface

SoC shall use either SPI or I2C interface of [UPD350](#) to access all the on-chip Control and Status Registers.

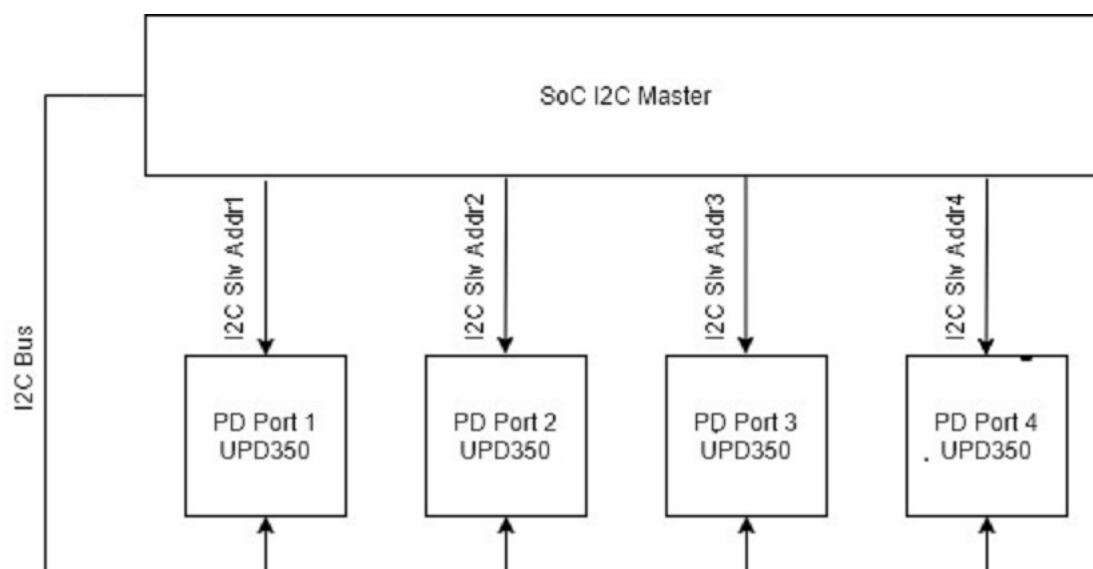
##### SPI Master:

The SoC needs to have a SPI master interface that can interact with the UPD350B/D SPI slave interfaces at SPI clock speeds up to 25 MHz. In addition to this, the SoC should also have a dedicated SPI Chip Select (CS) line for each UPD350B/D in the system.



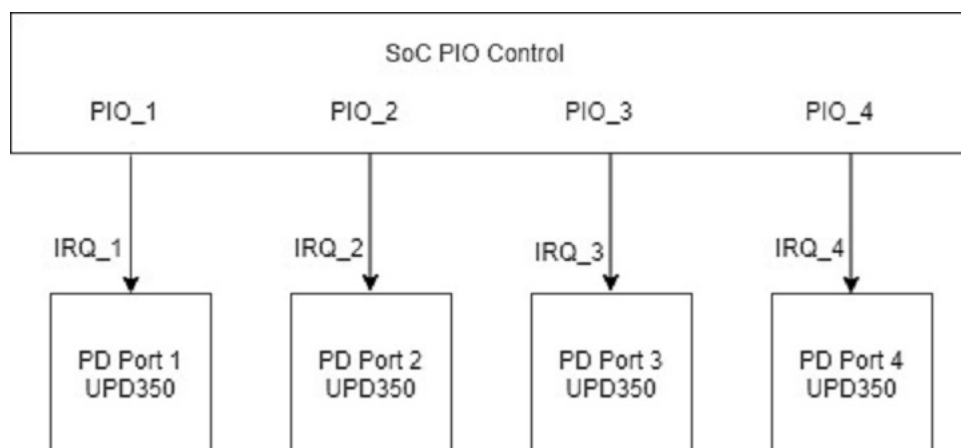
#### **I2C Master:**

SOC needs to have an I2C Master interface for interacting with the UPD350A/C in the system. The UPD350A/C's I2C slave interface supports Standard mode(100 kbps), Fast Mode(400 kbps) and Fast Mode Plus(1 Mbps) speeds. Each PD port having a [UPD350](#) is identified by a unique I2C slave address.



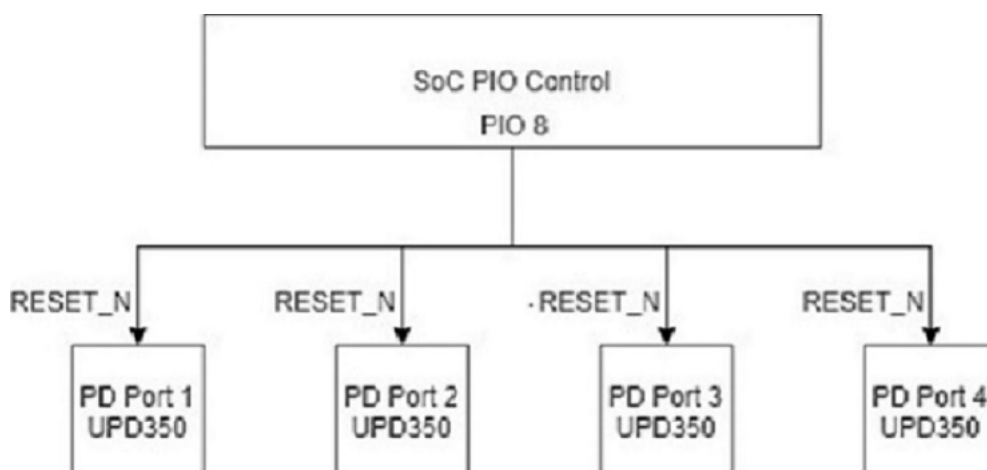
### **9.1.1.2 PIOs for UPD350 IRQs**

A PIO specific to each port's [UPD350](#) is required for [UPD350](#) IRQ interrupt detection. IRQ\_N of [UPD350](#) is an active low signal. Thus, SoC 's User\_Application shall configure the PIO for active low interrupt detection. The SOC is also required to inform PSF when the IRQ line is asserted by the [UPD350](#). This can be done by using the APIs provided in [APIs to be called by the User application](#) section.



### 9.1.1.3 PIO for UPD350 Reset

PSF needs a dedicated PIO from SoC to reset the UPD350s. It is recommended to use a single PIO for all the UPD350s in the system. It is connected to RESET\_N of [UPD350](#), an active low signal.



## 9.1.2 Hardware Timer

A Hardware timer with a minimum resolution of 1ms is required synchronizing the various state machines of PSF. PSF creates multiple software instance of this hardware timer. SoC User\_Application shall inform PSF about the occurrence of every Timer interrupt through APIs provided in [APIs to be called by the User application](#) section.

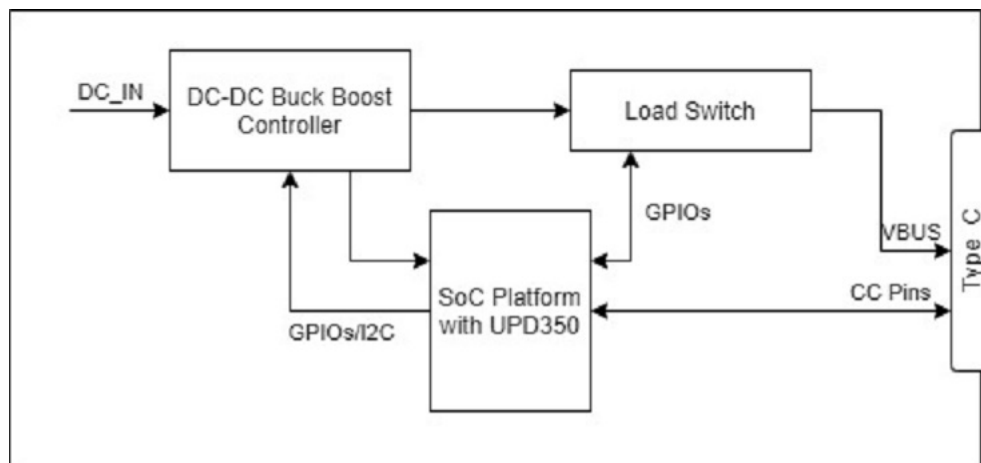
Note: PSF is tested with hardware timer resolution of 1ms for two port source configuration.

## 9.1.3 DC-DC Buck Boost Controller

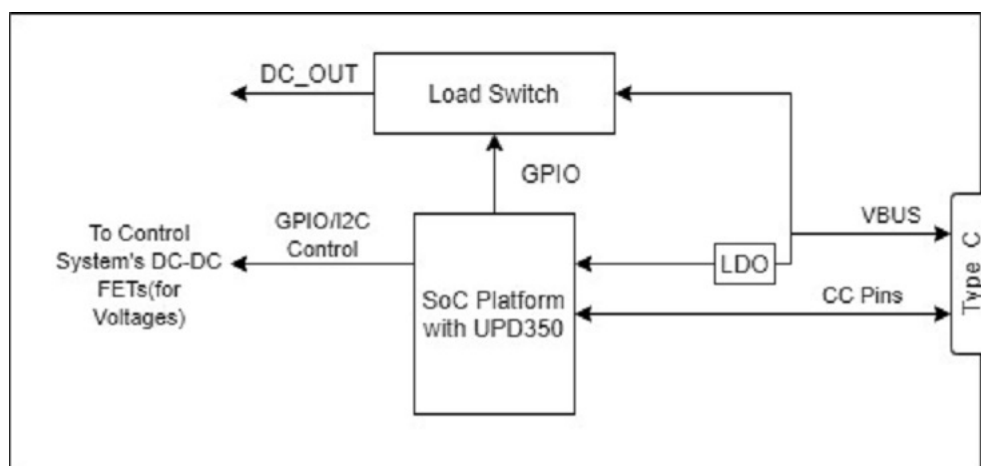
A [UPD350](#) is PD controller that negotiates a PD contract. It does not provide the PD voltage for the ports. The [UPD350](#) employs a DC-DC buck boost controller for this purpose. The DC-DC controller can be controlled from the [UPD350](#) using PIOs or I2C.



For Source-only port, the setup would be as follows,



For Sink-Only port, the set up would be as follows,



Note: PSF is configured by default to use GPIO mode DC-DC control. I2C mode DC-DC control is available for select DC-DC modules.

## 9.2 Software Requirements

This section lists the software requirements for PSF integration. This guide helps to determine whether PSF integration is possible with the available memory constraints in SoC.

### 9.2.1 Memory Requirement

- **32-bit**

With SAMD20 SoC and XC32 compiler whose optimization level is set to s(highest level), the code size for the sample applications is below

Sample	Functionality	Code Size	Data Size
Source Lite	Basic Source - 2 Port	31897 Byte	1540 Bytes
Source Pro	2 Port Source + PB +PT	35135 Bytes	1648 Bytes
Source Pro	2 Port Source +PPS	33717 Bytes	1588 Bytes
Sink	1 Port Sink	30269 Bytes	928 Bytes
DRP	2 Port DRP	42266 Bytes	1716 Bytes

Any additional port will increase total Data RAM size requirement by approximately 500 Bytes.

- **16-bit**

TBD

- **8-bit**

TBD

## 9.2.2 Multi-Port Support

PSF supports maximum of 4 ports.

Note: PSF has been tested for up to 2 PD ports.

## 9.2.3 Endianness

PSF supports and has been tested on little endian SOC.

## 9.3 Steps to integrate PSF

Steps to integrate to new SoC platform:

1. Select a compatible SoC based on [Hardware Requirements](#) and [Software Requirements](#)
2. Generate required SoC drivers. Port the mandatory APIs specified in [APIs to be implemented by the User application](#) in PSF\_APIHooks header file.
3. Integrate the PSF APIs listed in [APIs to be called by the User Application](#) with the driver APIs of the new SoC. These APIs are available in PSF\_APIHook header file.
4. Choose a PD functionality for the application (for example, Source only operation, Sink only operation) and configure the PSF\_Config.h file with the help of section [PSF Configuration Options](#).

# 10 APIs Implementation required for SW integration

This section provides the list of APIs needed for integrating PSF with an existing application or a new SoC.

These APIs can be categorized further as:

- APIs to be implemented by the SoC User\_Application
- APIs to be called by the SoC User\_Application

## 10.1 Data types

The following user data types need to be defined for ensuring compatibility with PSF.




Pre-processor definitions	Data types
TRUE	1
FALSE	0
BOOL	unsigned char
UINT8	unsigned char
UINT16	unsigned short
UINT32	unsigned long
INT8	char
INT16	short
INT32	long
CHAR	char
UCHAR	unsigned char

## 10.2 APIs to be implemented by the User Application

This section lists all the APIs that need to be implemented by the user application for ensuring compatibility with PSF.

## 10.2.1 UPD350 Hardware Interface Configurations

### Macros

	Name	Description
	<code>MCHP_PSF_HOOK_UPDHW_INTF_INIT</code>	Initialize the hardware interface(SPI/I2C) used for communicating with <a href="#">UPD350</a> part.
	<code>MCHP_PSF_HOOK_UPD_READ</code>	Initiates a read transfer to <a href="#">UPD350</a> via I2C/SPI
	<code>MCHP_PSF_HOOK_UPD_WRITE</code>	Initiates a write transfer to <a href="#">UPD350</a> via I2C/SPI

### 10.2.1.1 MCHP\_PSF\_HOOK\_UPDHW\_INTF\_INIT

#### C

```
#define MCHP_PSF_HOOK_UPDHW_INTF_INIT 0
```

#### Description

PSF requires a Hardware interface from SOC(either SPI or I2C) to communicate with [UPD350](#). [UPD350](#) supports either I2C or SPI interface depending on [UPD350](#) SKU used. [UPD350](#) A and C SKUs support I2C interface and [UPD350](#) B and D SKUs support SPI interface. This Hook is used to initialize the SOC's Hardware interface for communication. It is called during initialization of PSF. This hook is assigned to a function that takes no arguments and has a return type of UINT8 which indicates if the initialization was successful or not.

#### Preconditions

Use SPI interface for part [UPD350](#) B and D. Use I2C interface for part [UPD350](#) A and C.

#### Returns

UINT8 - Return TRUE if initialization was successful or else return FALSE. If hardware interface initialization(SPI/I2C) fails and the API returns FALSE, all the PD ports are disabled by PSF by default.

#### Remarks

User definition of this Hook function is mandatory.

#### Example

```
#define MCHP_PSF_HOOK_UPDHW_INTF_INIT()      hw_spi_init()
UINT8 hw_spi_init(void);
UINT8 hw_spi_init(void)
{
    //Intialise SOC's SPI master
    //Return TRUE if initialisation is successful else FALSE
}
#define MCHP_PSF_HOOK_UPDHW_INTF_INIT()      hw_i2c_init()
UINT8 hw_i2c_init(void);
UINT8 hw_i2c_init(void)
{
    //Initialise SOC's I2C master
    //Return TRUE if initialisation is successful else FALSE
}
```

### 10.2.1.2 MCHP\_PSF\_HOOK\_UPD\_READ

#### C

```
#define MCHP_PSF_HOOK_UPD_READ(u8PortNum,pu8WriteBuf,u8WriteLen,pu8ReadBuf, u8ReadLen) 0
```

#### Description

This hook is used to read the registers of the [UPD350](#) of a given port over the chosen communication interface defined by

**CONFIG\_DEFINE\_UPD350\_HW\_INTF\_SEL.** This hook is assigned to a function that takes arguments of type UINT8, UINT8 \*, UINT8, UINT8 \*, UINT8 and has a return type of UINT8.

#### Preconditions

None.

#### Parameters

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to ( <b>CONFIG_PD_PORT_COUNT</b> -1).
pu8WriteBuf	PSF shall pass the pointer to the buffer which has the data to be written on the SPI/I2C Hardware bus before read. It contains the Register address to be read. Data type of the pointer buffer must be UINT8*.
u8WriteLen	PSF shall pass the Number of bytes to be written on the SPI/I2C Hardware bus. Data type of this parameter must be UINT8.
pu8ReadBuf	PSF will pass the pointer to the buffer where data read from the SPI/I2C bus to be stored. Data type of the pointer buffer must be UINT8*.
u8ReadLen	PSF will pass the number of bytes to be read on the SPI/I2C bus. Data type of this parameter must be UINT8.

#### Returns

UINT8 - Return TRUE if read was successful or else return FALSE.

#### Remarks

User definition of this Hook function is mandatory.

#### Example

```
#define MCHP_PSF_HOOK_UPD_READ(u8PortNum,pu8WriteBuf,u8WriteLen,pu8ReadBuf,u8ReadLen)
    SPI_Read (u8PortNum,pu8WriteBuf,u8WriteLen,pu8ReadBuf,u8ReadLen)
void SPI_Read (UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength,
               UINT8 *pu8ReadBuffer, UINT8 u8Readlength);
void SPI_Read (UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength,
               UINT8 *pu8ReadBuffer, UINT16 u8Readlength)
{
    for(UINT8 u8Txcount = 0; u8Txcount < u16Writelength; u8Txcount++)
    {
        //Write data bytes to SPI bus
    }
    for(UINT8 u8Rxcount = 0; u8Rxcount< u8Readlength; u8Rxcount++)
    {
        //Read data from SPI bus
    }
    // Return TRUE if the read is successful; else FALSE
}
#define MCHP_PSF_HOOK_UPD_READ(u8PortNum,pu8WriteBuf,u8WriteLen,pu8ReadBuf,u8ReadLen)
    I2C_Read(u8PortNum,pu8WriteBuf,u8WriteLen,pu8ReadBuf,u8ReadLen)

void I2C_Read (UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength,
               UINT8 *pu8ReadBuffer, UINT16 u8Readlength);
void I2C_Read (UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength,
               UINT8 *pu8ReadBuffer, UINT16 u8Readlength)
{
    //Select I2C address for the UPD350 I2C slave using u8PortNum
    for(UINT8 u8Txcount = 0; u8Txcount < u16Writelength; u8Txcount++)
    {
        //Write data bytes to I2C bus
    }
    for(UINT8 u8Rxcount = 0; u8Rxcount< u8Readlength; u8Rxcount++)
    {
        //Read data from I2C bus
    }
    // Return TRUE if the read is successful else return FALSE
}
```

## 10.2.1.3 MCHP\_PSF\_HOOK\_UPD\_WRITE

### C

```
#define MCHP_PSF_HOOK_UPD_WRITE(u8PortNum,pu8WriteBuf,u8WriteLen) 0
```

#### Description

This hook is used to write to the registers of the [UPD350](#) of a given port over the chosen communication interface defined by [CONFIG\\_DEFINE\\_UPD350\\_HW\\_INTF\\_SEL](#). This hook is assigned to a function that takes arguments of type UINT8, UINT8 \*, UINT8, and has a return type of UINT8.

#### Preconditions

None.

#### Parameters

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to ( <a href="#">CONFIG_PD_PORT_COUNT</a> -1).
pu8WriteBuf	PSF shall pass the pointer to the buffer which has the data to be written on the SPI/I2C Hardware bus. Data type of the pointer buffer must be UINT8*.
u8WriteLen	PSF shall pass the Number of bytes to be written on the SPI/I2C Hardware bus. Data type of this parameter must be UINT8.

#### Returns

UINT8 - Return TRUE if write was successful or else return FALSE.

#### Remarks




User definition of this Hook function is mandatory

#### Example

```
#define MCHP_PSF_HOOK_UPD_WRITE(u8PortNum,pu8WriteBuf,u8WriteLen)
    SPI_Write (u8PortNum,pu8WriteBuf,u8WriteLen)
UINT8 SPI_Write(UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength);
UINT8 SPI_Write(UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength)
{
    for(UINT8 u8Txcount = 0; u8Txcount < u16Writelength; u8Txcount++)
    {
        //Write data bytes to SPI bus
    }
    // Return TRUE if the write is successful; else FALSE
}
#define MCHP_PSF_HOOK_UPD_WRITE(u8PortNum,pu8WriteBuf,u8WriteLen)
    I2C_Write (u8PortNum,pu8WriteBuf,u8WriteLen)
UINT8 I2C_Write(UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength);
UINT8 I2C_Write(UINT8 u8PortNum, UINT8 *pu8WriteBuffer, UINT8 u8Writelength)
{
    // Select I2C address for the UPD350 I2C slave using u8PortNum
    for(UINT8 u8Txcount = 0; u8Txcount < u16Writelength; u8Txcount++)
    {
        //Write data bytes to I2C bus
    }
    // Return TRUE if the write is successful; else FALSE
}
```

## 10.2.2 PD Timer Configuration

### Macros

	Name	Description
	<a href="#"><code>MCHP_PSF_PDTIMER_INTERRUPT_RATE</code></a>	Rate at which the Hardware PD timer generates an interrupt
	<a href="#"><code>MCHP_PSF_HOOK_HW_PDTIMER_INIT</code></a>	Hook to Initialize and start the hardware timer module.
	<a href="#"><code>MCHP_PSF_CONFIG_16BIT_PDTIMER_COUNTER</code></a>	Config 16-bit PD Timer Counter

### 10.2.2.1 MCHP\_PSF\_PDTIMER\_INTERRUPT\_RATE

#### C

```
#define MCHP_PSF_PDTIMER_INTERRUPT_RATE 1000
```

#### Description

MCHP\_PSF\_PDTIMER\_INTERRUPT\_RATE defines the frequency of interrupt set in the hardware timer dedicated for PSF. In other words, it is the resolution of the hardware timer. It can be configured depending on the resolution of the hardware timer available.

#### Remarks

Resolution of the hardware timer has to be at least 1ms. Tested resolution values of hardware timer is 1ms.(Corresponding MCHP\_PSF\_PDTIMER\_INTERRUPT\_RATE value is 1000).

#### Example

```
#define MCHP_PSF_PDTIMER_INTERRUPT_RATE      1000
(1000 interrupts per second, with interrupt interval or resolution of 1ms)
```

### 10.2.2.2 MCHP\_PSF\_HOOK\_HW\_PDTIMER\_INIT

#### C

```
#define MCHP_PSF_HOOK_HW_PDTIMER_INIT 0
```

#### Description

PSF requires a single dedicated hardware timer module for synchronizing the various state machines in the stack. This Hook initializes and starts the hardware timer module for [`MCHP\_PSF\_PDTIMER\_INTERRUPT\_RATE`](#) interrupt frequency. To inform PSF about the occurrence of hardware timer interrupt, [`MchPSF\_PDTimerHandler`](#) should be called by the SOC layer on every occurrence of the timer interrupt. This hook is assigned to a function that takes no arguments and has a return type of UINT8.

#### Preconditions

None.

#### Returns

UINT8 - Returns TRUE if initialization is successful else FALSE. If Timer initialization fails and the API returns FALSE, all the PD ports are disabled by PSF by default.

#### Remarks

User definition of this Hook function is mandatory

#### Example

```
#define MCHP_PSF_HOOK_HW_PDTIMER_INIT()      Timer_Init()
UINT8 Timer_Init(void);
```

```

UINT8 Timer_Init(void)
{
    //Initialize and start the SOC timer module for MCHP_PSF_PDTIMER_INTERRUPT_RATE
    //interrupt frequency & register MchpPSF_PDTimerHandler as callback
    //Return TRUE if Timer initialisation is successful else return FALSE
}

```

### 10.2.2.3 MCHP\_PSF\_CONFIG\_16BIT\_PDTIMER\_COUNTER

C

```
#define MCHP_PSF_CONFIG_16BIT_PDTIMER_COUNTER 0
```

#### Description

MCHP\_PSF\_CONFIG\_16BIT\_PDTIMER\_COUNTER can be defined as either 1 or 0 to set the timeout counter in PSF to unsigned 16bit or unsigned 32bit correspondingly. When set as 1, maximum timeout that can be set will be 65535 ticks.(Ticks = Resolution of the Hardware timer used). When set as 0, maximum timeout that can be set will be 4294967296 ticks. Default value of MCHP\_PSF\_CONFIG\_16BIT\_PDTIMER\_COUNTER is set as 1. With Hardware timer resolution set as 1ms, PSF will be capable of handling timeouts upto 65.535 Seconds.

#### Remarks

None

#### Example

```

#define MCHP_PSF_CONFIG_16BIT_PDTIMER_COUNTER 1 (Sets timeout variable inside the PSF as
unsigned 16bit)
#define MCHP_PSF_CONFIG_16BIT_PDTIMER_COUNTER 0 (Sets timeout variable inside the PSF as
unsigned 32bit)

```

## 10.2.3 SoC Interrupt Enable/Disable

#### Macros

	Name	Description
	<u>MCHP_PSF_HOOK_ENABLE_GLOBAL_INTERRUPT</u>	Enables the global interrupt.
	<u>MCHP_PSF_HOOK_DISABLE_GLOBAL_INTERRUPT</u>	Disables the global interrupt.

### 10.2.3.1 MCHP\_PSF\_HOOK\_ENABLE\_GLOBAL\_INTERRUPT

C

```
#define MCHP_PSF_HOOK_ENABLE_GLOBAL_INTERRUPT
```

#### Description

This hook is used to enable the global interrupts of the chosen SOC. It is usually invoked when PSF exits from a critical section in the code. This hook is assigned to a function that takes no arguments and has a return type of void. The assigned function should be very short to ensure interrupt response times are shorter and potential timing issues or race conditions can be prevented.

#### Preconditions

None.

#### Returns

None.

#### Remarks

User definition of this Hook function is mandatory



**Example**

```
#define MCHP_PSF_HOOK_ENABLE_GLOBAL_INTERRUPT()    CRITICAL_SECTION_EXIT()
void CRITICAL_SECTION_EXIT(void);
void CRITICAL_SECTION_EXIT()
{
    //Enable global interrupts
}
```

## 10.2.3.2 MCHP\_PSF\_HOOK\_DISABLE\_GLOBAL\_INTERRUPT

**C**

```
#define MCHP_PSF_HOOK_DISABLE_GLOBAL_INTERRUPT
```

**Description**

This hook is used to disable the global interrupts of the chosen SOC. It is usually invoked when PSF enters a critical section in the code. This hook is assigned to a function that takes no arguments and has a return type of void. The assigned function should be very short to ensure interrupt response times are shorter and potential timing issues or race conditions can be prevented.

**Preconditions**

None.

**Returns**

None.

**Remarks**



User definition of this Hook function is mandatory

**Example**

```
#define MCHP_PSF_HOOK_DISABLE_GLOBAL_INTERRUPT()    CRITICAL_SECTION_ENTER()
void CRITICAL_SECTION_ENTER(void);
void CRITICAL_SECTION_ENTER()
{
    //Disable SOC's global interrupts
}
```

## 10.2.4 Memory Compare and Copy

**Macros**

	Name	Description
	<a href="#">MCHP_PSF_HOOK_MEMCMP</a>	Compare two memory regions
	<a href="#">MCHP_PSF_HOOK_MEMCPY</a>	Copies one memory area to another memory area

**Description**

These hooks are used to leverage the in-built library functions in the some compilers that allow for copying and comparing memory sections in minimum number of CPU machine cycles. These hooks can be used to improve performance.

### 10.2.4.1 MCHP\_PSF\_HOOK\_MEMCMP

**C**

```
#define MCHP_PSF_HOOK_MEMCMP(pObj1, pObj2, iLength) 0
```

**Description**

This hook is used to compare two memory regions pObj1, pObj2 with specified length u8Length. User must define

this hook based on the compiler of SOC.

#### Preconditions

None.

#### Parameters

Parameters	Description
pObj1	This is the pointer to block of Memory region 1
pObj2	This is the pointer to block of Memory region 2
iLength	This is the number of bytes to be compared.

#### Returns

Return 0 if two memory regions are same else return number of bytes did not match.

#### Remarks

User definition of this Hook function is mandatory

#### Example

```
#define MCHP_PSF_HOOK_MEMCMP(pObj1, pObj2, iLength) memcmp(pObj1, pObj2, iLength)
//This hook definition can be compiler defined or user defined.
```

## 10.2.4.2 MCHP\_PSF\_HOOK\_MEMCPY

### C

```
#define MCHP_PSF_HOOK_MEMCPY(pDest, pSrc, iLen) 0
```

#### Description

This hook is used to copy iLen bytes from pSrc memory area to pDest memory area. User must define this hook based on the compiler of SOC. The memory areas cannot overlap.

#### Preconditions

None.

#### Parameters

Parameters	Description
pDest	This is the pointer to block of destination memory region
pSrc	This is the pointer to block of source memory region
iLen	This is the number of bytes to be copied.

#### Returns

Returns a pointer to pDest.

#### Remarks



User definition of this Hook function is mandatory

#### Example

```
#define MCHP_PSF_HOOK_MEMCPY(pDest, pSrc, iLen) memcpy(pDest, pSrc, iLen)
//This hook definition can be compiler defined or user defined.
```

## 10.2.5 Structure Packing

### Macros

	Name	Description
	<a href="#"><u>MCHP_PSF_STRUCT_PACKED_START</u></a>	Structure packing to align the bytes in data memory based on the compiler.
	<a href="#"><u>MCHP_PSF_STRUCT_PACKED_END</u></a>	Structure packing to align the bytes in data memory based on the compiler.

### 10.2.5.1 MCHP\_PSF\_STRUCT\_PACKED\_START

#### C

```
#define MCHP_PSF_STRUCT_PACKED_START
```

#### Description

Packed structures are used to save space & align the bytes in data memory based on the compiler. If this pre-processor is defined, then all the PSF's "C" structures will be replaced with this keyword prior to compilation. If this pre-processor is not defined, then structures will be compiled using the default rules of the compiler.

#### Remarks

Need to be packed always based on the type of SOC.

#### Example

```
#define MCHP_PSF_STRUCT_PACKED_START __attribute__((__packed__))
```

### 10.2.5.2 MCHP\_PSF\_STRUCT\_PACKED\_END

#### C

```
#define MCHP_PSF_STRUCT_PACKED_END
```

#### Description

Packed structures are used to save space & align the bytes in data memory based on the compiler. If this pre-processor is defined, then all the PSF's "C" structures will be replaced with this keyword prior to compilation. If this pre-processor is not defined, then structures will be compiled using the default rules of the compiler.

#### Remarks


Need to be packed always based on the type of SOC.




#### Example

```
#define CONFIG_STRUCT_PACKED_END _Pragma("pack()")
```

## 10.2.6 Port Power Control

### Macros

	Name	Description
	<a href="#"><u>MCHP_PSF_HOOK_HW_PORTPWR_INIT</u></a>	Initializes all the hardware modules related to port power functionality especially DC-DC buck boost controller and load switch. Additionally, in case of sink functionality, this hook may be defined with APIs to initialize a DAC.

	<a href="#">MCHP_PSF_HOOK_PORTPWR_DRIVE_VBUS</a>	Drives the VBUS line with the negotiated voltage and current
	<a href="#">MCHP_PSF_HOOK_PORTPWR_CONFIG_SINK_HW</a>	Enables or disables sink hardware circuitry and configures it to sink the VBUS voltage for a given port based on the sink requested voltage and current.
	<a href="#">MCHP_PSF_HOOK_DRIVE_DAC_I</a>	Indicates the implicit/explicit current capability of attached source partner.

### 10.2.6.1 MCHP\_PSF\_HOOK\_HW\_PORTPWR\_INIT

#### C

```
#define MCHP_PSF_HOOK_HW_PORTPWR_INIT(u8PortNum)
```

#### Description

This hook is used to initialize the hardware modules related to port power functionality. Implementation of this function depends on the type of DC-DC buck boost controller, load switch or DAC used. This hook is assigned to a function that takes an argument of type UINT8 and has a return type of UINT8.

#### Preconditions

API implementation must make sure that the Port Power(VBUS) of all ports is set to 0V.

#### Parameters

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to ( <a href="#">CONFIG_PD_PORT_COUNT</a> -1).

#### Returns

None.

#### Remarks

User definition of this Hook function is mandatory. A DAC may initialized under this hook if PSF is configured as SINK.

#### Example

```
#define MCHP_PSF_HOOK_HW_PORTPWR_INIT(u8PortNum)    hw_portpower_init(u8PortNum)
void hw_portpower_init(void);
void hw_portpower_init(void)
{
    //Initializes the hardware modules related to port power functionality
}
```

### 10.2.6.2 MCHP\_PSF\_HOOK\_PORTPWR\_DRIVE\_VBUS

#### C

```
#define MCHP_PSF_HOOK_PORTPWR_DRIVE_VBUS(u8PortNum,u16VBUSVolatge,u16Current) \
```

#### Description

If the user chose to implement their own DC-DC buck boost control, this hook must be implemented to drive VBUS as per the arguments passed with it. Implementation of this hook depends on the type of DC-DC buck boost controller and load switch used. This hook is assigned to a function that takes arguments of type UINT8, UINT16, UINT16 and no return type.

#### Preconditions

It is applicable only for Source operation.

**Parameters**

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to (CONFIG_PD_PORT_COUNT-1).
u16VBUSVoltage	VBUS Voltage level to be driven in VBUS expressed in terms of milliVolts.
u16Current	VBUS current level in terms of mA.

**Returns**

None.

**Remarks**

User definition of this Hook function is mandatory.

**Example**

```
#define MCHP_PSF_HOOK_PORTPWR_DRIVE_VBUS(u8PortNum, u16VBUSVoltage, u16Current)
    hw_portpower_driveVBUS(u8PortNum, u16VBUSVoltage, u16Current)
void hw_portpower_driveVBUS(UINT8 u8PortNum, UINT16 u16VBUSVoltage, UINT16 u16Current);
void hw_portpower_driveVBUS(UINT8 u8PortNum, UINT16 u16VBUSVoltage, UINT16 u16Current)
{
    // Configure DC-DC buck boost control to drive u16VBUSVoltage & u16Current in VBUS
}
```

### 10.2.6.3 MCHP\_PSF\_HOOK\_PORTPWR\_CONFIG\_SINK\_HW

**C**

```
#define MCHP_PSF_HOOK_PORTPWR_CONFIG_SINK_HW(u8PortNum,u16Voltage,u16Current)
```

**Description**

This hook is used to enable or disable sink hardware circuitry and configure it for Sink requested current and voltage. Implementation of this function depends on the type of Sink circuitry used. This hook is assigned to a function that takes arguments of type UINT8,UINT16,UINT16 without return type.

**Preconditions**

It is applicable only for Sink operation.

**Parameters**

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to (CONFIG_PD_PORT_COUNT-1).
u16voltage	Enable Sink HW Circuitry if the u16voltage is not vSafe0V to drain power. Disable sink HW circuitry if the u16voltage is vSafe0V. Configure the HW to requested u16voltage in mV.
u16Current	Configure the HW for the requested current passed in terms of mA.

**Returns**

None.

**Remarks**

User definition of this Hook function is mandatory if PSF is configured for Sink functionality.

**Example**

```
#define MCHP_PSF_HOOK_PORTPWR_CONFIG_SINK_HW(u8PortNum, u16Voltage, u16Current)
    hw_Configure_SinkCircuitary(u8PortNum, u16Voltage, u16Current)
void hw_Configure_SinkCircuitary(UINT8 u8PortNum,UINT16 u16Votlage,UINT16 u16Current);
void hw_Configure_SinkCircuitary(UINT8 u8PortNum,UINT16 u16Votlage,UINT16 u16Current)
{
    if(u16Voltage == Vsafe0V)
    {
```

```

        //Disable the Sink circuitary for "u8PortNum" Port
    }
    else
    {
        //Enable the Sink circuitary for "u8PortNum" Port and
        configure it to drain u16Voltage
    }
    //Conifgure Sink circuitary for u16Current current rating
}

```

## 10.2.6.4 MCHP\_PSF\_HOOK\_DRIVE\_DAC\_I

### C

```
#define MCHP_PSF_HOOK_DRIVE_DAC_I(u8PortNum, u16DACData)
```

#### Description

This hook is used by PSF to inform the sink about the implicit/explicit current capability of the attached source partner. The current capability is indicated through a voltage level on Digital to Analog Converter(DAC)'s output pin. The voltage level on DAC's output pin is calculated based on per port Configuration parameters, which were configured using [MCHP\\_PSF\\_HOOK\\_BOOT\\_TIME\\_CONFIG](#)(pasCfgStatusData) hook.

In [gasCfgStatusData](#) structure, if u16DAC\_I\_CurrentIhd\_MaxInA is 5000mA, u16DAC\_I\_MaxOutVoltInmV is 2500mV, u16DAC\_I\_MinOutVoltInmV is 0V and direction mentioned in u8DAC\_I\_Direction is High Amperage - Max Voltage, then

1. 0.5A > DAC\_I = 0.25V
2. 1.5A > DAC\_I = 0.75V
3. 2.0A > DAC\_I = 1V
4. 3.0A > DAC\_I = 1.5V
5. 4.0A > DAC\_I = 2.0V
6. 5.0A > DAC\_I = 2.5V

In [gasCfgStatusData](#) structure, if u16DAC\_I\_CurrentIhd\_MaxInA is 3000mA, u16DAC\_I\_MaxOutVoltInmV is 2500mV, u16DAC\_I\_MinOutVoltInmV is 0V and direction mentioned in u8DAC\_I\_Direction is High Amperage - Max Voltage, then \* If it is 3A and maximum

1. 0.5A > DAC\_I = 0.42V
2. 1.5A > DAC\_I = 1.25V
3. 2.0A > DAC\_I = 1.67V
4. 3.0A > DAC\_I = 2.5V
5. 4.0A > DAC\_I = 2.5V
6. 5.0A > DAC\_I = 2.5V

A suitable function that initializes DAC from SoC may be implemented in this hook.

#### Preconditions

SoC should support a DAC and the DAC should be initialized under [MCHP\\_PSF\\_HOOK\\_HW\\_PORTPWR\\_INIT](#)() hook.

#### Parameters

Parameters	Description
u8PortNum	Port number for which DAC_I needs to be driven
u16DACData	Analog voltage to be driven on DAC_I pin

#### Returns

None.

#### Remarks

This hook is applicable only if [INCLUDE\\_PD\\_SINK](#) macro is 1. Definition of this hook is not mandatory. This is applicable

only for Sink operation.

#### Example

```
#define MCHP_PSF_HOOK_DRIVE_DAC_I(u8PortNum, u16DACData)    HW_Drive_DAC_I(u8PortNum,
u16DACData)
void HW_Drive_DAC_I(UINT8 u8PortNum, UINT16 u16DACData);
void HW_Drive_DAC_I(UINT8 u8PortNum, UINT16 u16DACData)
{
    //Implement user specific application to output volatge provided under
    //u16DACData argument in DAC's output pin
}
```

## 10.2.7 Boot time Configuration

#### Macros

	Name	Description
	<b>MCHP_PSF_HOOK_BOOT_TIME_CONFIG</b>	Updates the global and per port Configuration parameters.

### 10.2.7.1 MCHP\_PSF\_HOOK\_BOOT\_TIME\_CONFIG

#### C

```
#define MCHP_PSF_HOOK_BOOT_TIME_CONFIG(pasCfgStatusData)
```

#### Description

This hook is used to update the configuration parameters of Type-C, PD, Power Balancing, Power throttling, PPS, VDM and AltMode contained in [gasCfgStatusData](#) structure. This hook must have an argument [gasCfgStatusData](#).

#### Preconditions

None.

#### Parameters

Parameters	Description
pasCfgStatusData	Holds the structure pointer of the structure <a href="#">_GlobalCfgStatusData</a>

#### Returns

None.

#### Remarks

User definition of this Hook function is mandatory

#### Example


```
#define MCHP_PSF_HOOK_BOOT_TIME_CONFIG(pasCfgStatusData)
PSF_LoadConfig(pasCfgStatusData)
void PSF_LoadConfig(pasCfgStatusData)
{
    // Configure the global parameters
    // Enable Power Throttling and Select Bank B
    pasCfgStatusData->u8PwrThrottleCfg = 0x03;
    // Set 120W as Total system Power of Bank A
    pasCfgStatusData->u16SystemPowerBankAIn250mW = 0x01E0U;
    // Configure per port parameters
    // Set Port 1's VBUS Maximum Fault Count as 3
    pasCfgStatusData->sPerPortData[0].u8VBUSMaxFaultCnt = 0x03; // 0 is the port number
    // Configure per port PB parameters
    // Set Port 2's maximum port power for Bank C as 60W
    pasCfgStatusData.sPBPerPortData[1]->u16MaxPrtPwrBankCIn250mW = 0x00F0U; // 1 is the
port number
}
```

## 10.2.8 GPIO Configuration

This section lists all the PIO configuration hooks and associated enums.

### 10.2.8.1 GPIO Init Function

#### Macros

	Name	Description
	<a href="#">MCHP_PSF_HOOK_GPIO_FUNC_INIT</a>	Hook to initialize all the GPIO functionality pins in application layer.

#### 10.2.8.1.1 MCHP\_PSF\_HOOK\_GPIO\_FUNC\_INIT

##### C

```
#define MCHP_PSF_HOOK_GPIO_FUNC_INIT(u8PortNum, eGPIOFunc)
```

#### Description

PSF calls this API to initialize the eMCHP\_PSF\_GPIO\_FUNCTIONALITY pins in the application layer. Users has to define an appropriate function with UINT8 and eMCHP\_PSF\_GPIO\_FUNCTIONALITY as argument. Users can assign any PIO either from UDP350 or MCU for any GPIO functionality defined. Drive of this API will be controlled by the hook [MCHP\\_PSF\\_HOOK\\_GPIO\\_FUNC\\_DRIVE](#).

#### Preconditions

None.

#### Parameters

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to <a href="#">(CONFIG_PD_PORT_COUNT-1)</a> .
eGPIOFunc	Passes the GPIO functionality type that has to be initialized by the application.

#### Returns

None.

#### Remarks

User definition of this Hook function is mandatory as well as it is mandatory to define functionality for ePSF\_GPIO\_Functionality.

#### Example

```
#define MCHP_PSF_HOOK_GPIO_FUNC_INIT(u8PortNum, eGPIO_Func)
App_GPIOControl_Initialisation(u8PortNum, eGPIO_Func)
void App_GPIOControl_Initialisation(UINT8 u8PortNum, eMCHP_PSF_GPIO_FUNCTIONALITY
eGPIOFunc)
{
    switch(eGPIOFunc)
    {
        case eDC_DC_EN_FUNC:
        {
            //Initialise the GPIO assigned for DC_DC
            // Drive the GPIO in default state
            break;
        }
        case eVBUS_DIS_FUNC:
        {
            //Initialise the GPIO assigned for VBUS_Discharge functionality
            //Drive the GPIO in default state
        }
    }
}
```




```

    }
}

```

## 10.2.8.2 GPIO Drive Function

### Macros

	Name	Description
	<code>MCHP_PSF_HOOK_GPIO_FUNC_DRIVE</code>	Hook to drive GPIOs assigned to GPIO functionality pins in application layer.

### 10.2.8.2.1 MCHP\_PSF\_HOOK\_GPIO\_FUNC\_DRIVE

#### C

```
#define MCHP_PSF_HOOK_GPIO_FUNC_DRIVE(u8PortNum, eGPIOFunc, eDriveVal)
```

#### Description

PSF calls this API to drive the ePSF\_GPIO\_Functionality pins in application layer as per drive value ePSF\_GPIO\_DriveVal. Users has to define an appropriate function with UINT8, eMCHP\_PSF\_GPIO\_FUNCTIONALITY, eMCHP\_PSF\_GPIO\_DRIVE\_VAL as argument. Users can assign any PIO either from UDP350 or MCU for any GPIO functionality defined.

#### Preconditions

None.

#### Parameters

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to ( <code>CONFIG_PD_PORT_COUNT</code> -1).
eGPIOFunc	Passes the GPIO functionality type that has to be initialized by the application.
eDriveVal	Drive value for the pin

#### Returns

None.

#### Remarks

User definition of this Hook function is mandatory as well as it is mandatory to define functionality for eMCHP\_PSF\_GPIO\_FUNCTIONALITY.

#### Example

```

#define MCHP_PSF_HOOK_GPIO_FUNC_DRIVE (u8PortNum, eGPIO_Func, eDriveVal)
App_GPIOControl_Drive(u8PortNum, eGPIO_Func, eDriveVal)
void App_GPIOControl_Drive(UINT8 u8PortNum,
                           eMCHP_PSF_GPIO_FUNCTIONALITY eGPIOFunc, eMCHP_PSF_GPIO_DRIVE_VAL
eDriveVal )
{
    switch(eGPIOFunc)
    {
        case eDC_DC_EN_FUNC:
        {
            if (eGPIO_Assert == eDriveVal)
            {
                // Assert the DC_DC pin
            }
            else
            {
                // De-assert the DC_DC pin
            }
            break;
        }
    }
}

```



```

    }
    case eVBUS_DIS_FUNC:
    {
        if (eGPIO_Assert == eDriveVal)
        {
            // Assert the VBUS Discharge pin
        }
        else
        {
            // De-assert the VBUS Discharge pin
        }
        break;
    }
}
}
}

```

### 10.2.8.3 GPIO Control enum Constants

#### Enumerations

	Name	Description
	<a href="#">eMCHP_PSF_GPIO_DriveValue</a>	GPIO Drive enum.
	<a href="#">eMCHP_PSF_GPIO_Functionality</a>	GPIO Functionality enum.

#### 10.2.8.3.1 eMCHP\_PSF\_GPIO\_DriveValue Enumeration

##### C

```

enum eMCHP_PSF_GPIO_DriveValue {
    eGPIO_DEASSERT,
    eGPIO_ASSERT
};

```

#### Description

eMCHP\_PSF\_GPIO\_DRIVE\_VAL enum defines Assert and Deassert drives of the various GPIO functionality Pins that are used in PSF.

#### Members

Members	Description
eGPIO_DEASSERT	Drive to De-Assert the GPIO
eGPIO_ASSERT	Drive to Assert the GPIO

#### Remarks

None

#### 10.2.8.3.2 eMCHP\_PSF\_GPIO\_Functionality Enumeration

##### C

```

enum eMCHP_PSF_GPIO_Functionality {
    eUPD350_ALERT_FUNC,
    eI2C_DC_DC_ALERT_FUNC,
    eUPD350_RESET_FUNC,
    eSPI_CHIP_SELECT_FUNC,
    eVBUS_DIS_FUNC,
    eDC_DC_EN_FUNC,
    eORIENTATION_FUNC,
    eSNK_CAPS_MISMATCH_FUNC,
    eSNK_1_5A_IND_FUNC,
    eSNK_3A_IND_FUNC,
    ePOWER_ROLE_FUNC,
    eDATA_ROLE_FUNC
};

```

**Description**

eMCHP\_PSF\_GPIO\_FUNCTIONALITY enum contains all the GPIO pin roles or functions that are used by PSF on a per port basis. Depending on the PD mode supported, each of these pin roles are assigned a unique GPIO from the MCU or the [UPD350](#).

Functionality	Input/Output	Description
eUPD350_ALERT_FUNC	Input	<ul style="list-style-type: none"> <li>PSF needs an SOC GPIO to be assigned to this role for detecting the status of the IRQ_N line of <a href="#">UPD350</a> associated with each port</li> <li>IRQ_N is an active low signal. This GPIO functionality shall initialize the SOC GPIOs connected to the IRQ_N lines of UPD350s in the system for interrupt notification. It is recommended to configure SOC GPIOs interrupt in edge level detection with internal pull up since the <a href="#">UPD350</a> keeps the IRQ_N line in low state until the interrupt is cleared.</li> <li>To notify PSF about the occurrence of <a href="#">UPD350</a> interrupt, the API <a href="#">MchpPSF_UPDIrqHandler</a> shall be called by SOC on interrupt detection of the specific port.</li> <li>GPIO connected to IRQ_N should be wakeup capable if <a href="#">INCLUDE_POWER_MANAGEMENT_CTRL</a> defined as 1. This is a mandatory functionality and configured only during initialization.</li> </ul>
eI2C_DC_DC_ALERT_FUNC	Input	<ul style="list-style-type: none"> <li>Configures the GPIO of the SOC for DC DC Alert functionality. This is not a mandatory functionality and can be configured based on the DC DC controller used.</li> </ul>
eUPD350_RESET_FUNC	Input	<ul style="list-style-type: none"> <li>This GPIO functionality is to control SOC GPIOs connected to the RESET_N lines of Port's <a href="#">UPD350</a>. It is mandatory to connect a single GPIO to the reset line of all UPD350s. The <a href="#">UPD350</a> RESET_N is an active low signal.</li> <li>The GPIO control for reset shall be handled as below <ul style="list-style-type: none"> <li>Initialization - Drive the GPIO high</li> <li>eGPIO_ASSERT - Drive the GPIO low and provide a sufficient delay for the <a href="#">UPD350</a> to reset</li> <li>eGPIO_DEASSERT - Drive the GPIO high</li> </ul> </li> <li>As single line is connected to all the <a href="#">UPD350</a> reset pin, the port number passed as argument to the hooks <a href="#">MCHP_PSF_HOOK_GPIO_FUNC_INIT</a> and <a href="#">MCHP_PSF_HOOK_GPIO_FUNC_DRIVE</a> shall be 0.</li> <li>This is a mandatory functionality to reset <a href="#">UPD350</a> and done only during initialization.</li> </ul>
eSPI_CHIP_SELECT_FUNC	Output	<ul style="list-style-type: none"> <li>The SOC GPIO assigned this pin role is used to enable/disable the SPI communication with the <a href="#">UPD350</a> of a given port. It is applicable only when <a href="#">CONFIG_DEFINE_UPD350_HW_INTF_SEL</a> is defined as CONFIG_UPD350_SPI. The eSPI_CHIP_SELECT_FUNC is active low signal.</li> <li>The GPIO control for CS shall be handled as below <ul style="list-style-type: none"> <li>Initialization - Drive the GPIO high</li> <li>eGPIO_ASSERT - Drive the GPIO low for the specific port to enable communication</li> <li>eGPIO_DEASSERT - Drive the GPIO high for the specific port to disable communication</li> </ul> </li> <li>It is mandatory to have a unique MCU pin for every port in the design, which is assigned this role.</li> </ul>

eVBUS_DIS_FUNC	Output	<ul style="list-style-type: none"> <li>VBUS Discharge mechanism is required to enable quick discharge of VBUS when VBUS transitions from higher to lower voltage. PSF requests the application layer to assert this pin whenever it requires a quick discharge of VBUS. PSF requests for de-assertion once the quick discharge is complete. The state of GPIO during initialization, assertion and de-assertion of this functionality is specific discharge circuitry used.</li> <li>It is mandatory to have a unique pin for every port in the design, which is assigned this role.</li> </ul>
eDC_DC_EN_FUNC	Output	<ul style="list-style-type: none"> <li>DC_DC_EN functionality is required to enable DC DC Controller. PSF requests the application layer to assert when PSF is initialized, ready to operate and CC pins are functional. It will be toggled during error condition say, on occurrence of fault to reset the DC DC controller. The state of GPIO during initialization, assertion and de-assertion of this functionality is specific to the DC DC Controller used.</li> <li>This is applicable only for Source functionality</li> </ul>
eORIENTATION_FUNC	Output	<ul style="list-style-type: none"> <li>Orientation functionality is used to indicate the orientation of the type-C cable connector. It can be used to control an external USB data multiplexer. PSF requests the application layer to initialize this functionality during device attach, assert during CC1 attach and de-assert during CC2 attach. The state of GPIO during initialization, assertion and de-assertion of this functionality is user specific.</li> <li>This is not mandatory, depends on user application.</li> </ul>
eSNK_CAPS_MISMATCH_FUNC	Output	<ul style="list-style-type: none"> <li>Sink Caps mismatch functionality is used to indicate any mismatch of capability during a PD negotiation. It is applicable only in Sink mode operation. PSF requests the application to assert when PD Sink negotiation is complete and if there was a capability mismatch with the selection. PSF requests the application to de-assert during port partner detach or during a new negotiation. The state of GPIO during initialization, assertion and de-assertion of this functionality is user specific.</li> <li>This is not mandatory, depends on the user application.</li> </ul>
eSNK_1_5A_IND_FUNC	Output	<ul style="list-style-type: none"> <li>This functionality indicates if the current capability of the sink is more than 1.5A. PSF requests the application to assert when the current capability or negotiated current is 1.5A or more and less than 3A. PSF requests the application to de-assert on detach event or during renegotiation. The state of GPIO during initialization, assertion and de-assertion of this functionality is user specific.</li> <li>This is applicable only in sink mode of operation and it is not mandatory, depends on user application.</li> </ul>
eSNK_3A_IND_FUNC	Output	<ul style="list-style-type: none"> <li>3A indicator functionality indicates if the current capability of the sink is more than 3A. PSF requests the application to assert when current capability or negotiated current is 3A or more. PSF requests the application to de-assert on detach event or during renegotiation. The state of GPIO during initialization, assertion and de-assertion of this functionality is user specific.</li> <li>This is applicable only for sink mode operation and it is not mandatory, depends on the user application.</li> </ul>


ePOWER_ROLE_FUNC	Output	<ul style="list-style-type: none"> <li>Power role indicator functionality is used to indicate the current power role of a port. PSF requests the application to assert this pin when the current power role is source. PSF requests the application to de-assert this pin when the current power role is sink or when the port partner is detached. The state of GPIO during initialization, assertion and de-assertion of this functionality is user specific.</li> <li>This is applicable only for DRP mode operation and it is not mandatory, depends on user application.</li> </ul>
eDATA_ROLE_FUNC	Output	<ul style="list-style-type: none"> <li>Data role indicator functionality is to indicate that the current data role is Host/Hub DFP. PSF requests the application to assert the pin when current data role is DFP. PSF requests the application to de-assert the pin when the current data role is UFP or when the port partner is detached. The state of GPIO during initialization, assertion and de-assertion of this functionality is user specific.</li> <li>This is applicable only for DRP mode operation and it is not mandatory, depends on user application.</li> </ul>

**Remarks**

None

## 10.2.9 Hooks for Policy Manager

**Macros**

	Name	Description
	<u>MCHP_PSF_HOOK_DPM_PRE_PROCESS</u>	This hook is called inside the DPM_RunStateMachine() API.

### 10.2.9.1 MCHP\_PSF\_HOOK\_DPM\_PRE\_PROCESS

**C**

```
#define MCHP_PSF_HOOK_DPM_PRE_PROCESS(u8PortNum)
```

**Description**

This hook is called before executing the Type C state machine and policy engine state machine. USER\_APPLICATION can define this function if a change is required in default device policy manager or add a user defined state machine. This hook is assigned to a function that takes an argument of type UNIT8 and has no return type.

**Preconditions**

None.

**Parameters**

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to ( <u>CONFIG_PD_PORT_COUNT</u> -1).

**Returns**

None.

**Remarks**





User definition of this Hook function is optional

**Example**

```
#define MCHP_PSF_HOOK_DPM_PRE_PROCESS(u8PortNum)
HookDevicePolicyManagerPreProcess(u8PortNum)
void HookDevicePolicyManagerPreProcess(UINT8 u8PortNum);
void HookDevicePolicyManagerPreProcess(UINT8 u8PortNum)
{
    //any application related change or enhancement or user defined state machine
}
```

## 10.2.10 Debug Hook Functions

**Macros**

	Name	Description
	<a href="#"><u>MCHP_PSF_HOOK_DEBUG_INIT</u></a>	Initialization of debug module interface
	<a href="#"><u>MCHP_PSF_HOOK_PRINT_CHAR</u></a>	Outputs a character via UART interface
	<a href="#"><u>MCHP_PSF_HOOK_PRINT_INTEGER</u></a>	Outputs an integer via UART interface
	<a href="#"><u>MCHP_PSF_HOOK_PRINT_TRACE</u></a>	Outputs a string via UART interface

### 10.2.10.1 MCHP\_PSF\_HOOK\_DEBUG\_INIT

**C**

```
#define MCHP_PSF_HOOK_DEBUG_INIT
```

**Description**

This hook is used to initialize the debug interface used by the SOC. It is called during initialization of PSF if [CONFIG\\_HOOK\\_DEBUG\\_MSG](#) is set to 1. This hook is assigned to a function that takes no arguments and has no return type

**Preconditions**

None.

**Returns**

None.

**Remarks**

User definition of this Hook function is mandatory when [CONFIG\\_HOOK\\_DEBUG\\_MSG](#) is declared as '1'.

**Example**

```
#define MCHP_PSF_HOOK_DEBUG_INIT()          uart_init()
void uart_init();
void uart_init()
{
    //Initializes the uart module to send and receive a character
}
```

### 10.2.10.2 MCHP\_PSF\_HOOK\_PRINT\_CHAR

**C**

```
#define MCHP_PSF_HOOK_PRINT_CHAR(byData)
```

**Description**

This hook is used to output a character via UART interface to help the user in debugging. This hook is assigned to a function that takes an argument of type char and has no return type.

**Preconditions**

`MCHP_PSF_HOOK_DEBUG_INIT()` should have been called before using this API.

**Returns**

None.

**Remarks**

This hook API can be used only if `CONFIG_HOOK_DEBUG_MSG` is 1.

**Example**

```
#define MCHP_PSF_HOOK_PRINT_CHAR(byData)          uart_print_char(byData)
void uart_print_char(char);
void uart_print_char(char byData)
{
    //Print a character through UART terminal
}
```

## 10.2.10.3 MCHP\_PSF\_HOOK\_PRINT\_INTEGER

**C**

```
#define MCHP_PSF_HOOK_PRINT_INTEGER(dwData, byWidth)
```

**Description**

This hook is used to output an integer via UART interface to help the user in debugging. The size of integer is specified in `byWidth`. `dwWriteInt` is of type unsigned long. `byWidth` is of type unsigned char. This hook is assigned to a function that takes arguments of type unsigned long and unsigned char and has no return type.

**Preconditions**

`MCHP_PSF_HOOK_DEBUG_INIT()` should have been called before using this API.

**Returns**

None.

**Remarks**

This hook API can be used only if `CONFIG_HOOK_DEBUG_MSG` is 1.

**Example**

```
#define MCHP_PSF_HOOK_PRINT_INTEGER(dwWriteInt, byWidth)
uart_print_char(dwWriteInt, byWidth)
void uart_print_int(UINT32 dwWriteInt, UINT8 byWidth);
void uart_print_int(UINT32 dwWriteInt, UINT8 byWidth)
{
    //Print byWidth no of bytes from dwWriteInt through UART terminal.
}
```

## 10.2.10.4 MCHP\_PSF\_HOOK\_PRINT\_TRACE

**C**

```
#define MCHP_PSF_HOOK_PRINT_TRACE(pbyMessage)
```

**Description**

This hook is used to output a string via UART interface to help the user in debugging. `pbyMessage` is of type `char *`. This hook is assigned to a function that takes an argument of type `char *` and has no return type.

**Preconditions**

`MCHP_PSF_HOOK_DEBUG_INIT()` should have been called before using this API.

**Returns**

None.

**Remarks**







This hook API can be used only if `CONFIG_HOOK_DEBUG_MSG` is 1.

**Example**

```
#define MCHP_PSF_HOOK_PRINT_TRACE(pbyMessage)          uart_print_string(pbyMessage)
void uart_print_string(char *);
void uart_print_string(char * pbyMessage)
{
    //Print a string in pbyMessage through UART terminal
}
```

## 10.2.11 PD Firmware Upgrade

**Macros**

	Name	Description
	<code>MCHP_PSF_HOOK_BOOT_FIXED_APP</code>	MCHP_PSF_HOOK_BOOT_FIXED_APP shall perform necessary operation to switch from Updatable application to Fixed application.
	<code>MCHP_PSF_HOOK_BOOT_UPDATABLE_APP</code>	MCHP_PSF_HOOK_BOOT_UPDATABLE_APP shall perform necessary operation to boot from the updatable application after a PDFU is successfully completed.
	<code>MCHP_PSF_HOOK_GETCURRENT_IMAGEBANK</code>	Returns the Index of the Image Bank which is currently executing.
	<code>MCHP_PSF_HOOK_IS_PDFU_ALLOWED_NOW</code>	<code>MCHP_PSF_HOOK_BOOT_UPDATABLE_APP</code> specifies if PD Firmware Update can be currently allowed, based on the priority of the application tasks currently executing.
	<code>MCHP_PSF_HOOK_PROGRAM_FWBLOCK</code>	Validate the Data block and program the data to the Memory, and return the status of the Programming Operation.
	<code>MCHP_PSF_HOOK_VALIDATE_FIRMWARE</code>	Validates the Flashed Binary using a user defined validation method and returns the status of the Firmware Validation Operation.

### 10.2.11.1 MCHP\_PSF\_HOOK\_BOOT\_FIXED\_APP

**C**

```
#define MCHP_PSF_HOOK_BOOT_FIXED_APP
```

**Description**

Re-flash of the Updatable\_Application image bank while currently executing in the Updatable Application image bank, requires switching to the fixed application for performing the upgrade. The application switching shall include invalidating the Updatable\_Application signatures and/or jump/reset for fresh boot from the fixed application.

**Preconditions**

This hook is invoked by the PD Firmware Update State-machine during the Reconfiguration phase(On reception PDFU\_INITIATE Command), when the Updatable application is currently running.

**Returns**

No Return Value. During execution of this function the control shall be switched to the Fixed application.

**Remarks**

User definition of this Hook function is mandatory in the Updatable application when `INCLUDE_PDFU` is TRUE



**Example**

```
#define MCHP_PSF_HOOK_BOOT_FIXED_APP()    Boot_Fixed_Application()
void Boot_Fixed_Application(void)
{
    EraseUpdatableAppSignature(); //Invalidate the Updatable app sign
    Reset(); //Reset to boot from Fixed app
}
```

## 10.2.11.2 MCHP\_PSF\_HOOK\_BOOT\_UPDATABLE\_APP

**C**

```
#define MCHP_PSF_HOOK_BOOT_UPDATABLE_APP
```

**Description**

As the flashing operation is executed from the Fixed application, once the PDFU process is complete it is necessary to switch to the newly upgraded updatable application. This hook definition shall implement necessary operations to safely exit the fixed application and boot from the updatable application. The application switching may include setting the valid Updatable\_Application signatures (and) jump/reset for fresh boot from Updatable application.

**Preconditions**

This function is invoked by the PD Firmware Update State-machine during the Manifestation phase (On reception PDFU\_INITIATE Command), when the Fixed application is currently running.

**Returns**

No Return Value. During execution of this function the control shall be switched to the Updatable application.

**Remarks**

User definition of this Hook function is mandatory in the Fixed application when [INCLUDE\\_PDFU](#) is TRUE.

**Example**

```
#define MCHP_PSF_HOOK_BOOT_UPDATABLE_APP()    Boot_Updatable_Application()
void Boot_Updatable_Application(void)
{
    Reset(); //Reset to boot from Updatable app
}
```

## 10.2.11.3 MCHP\_PSF\_HOOK\_GETCURRENT\_IMAGEBANK

**C**

```
#define MCHP_PSF_HOOK_GETCURRENT_IMAGEBANK 0x0
```

**Description**

This hook is used to get the Index of the image bank which is currently executing in the application. PSF follows "Architecture 2 - Fixed base application with updatable application image". In which the Fixed Application is Image Bank 1 and updatable Application is Image Bank 2.

**Preconditions**

This function is invoked by the PD Firmware Update State-machine during the Enumeration Phase (On reception PDFU\_GET\_FW\_ID Command).

**Returns**

Returns UINT8 - the index of the Image Bank. It can take following values: 0x01 - IMAGE\_BANK\_BOOTLOADER 0x02 - IMAGE\_BANK\_FIXED\_APP 0x03 - IMAGE\_BANK\_UPDATABLE\_APP

**Remarks**

The User definition of the function is mandatory in both Fixed and Updatable application when [INCLUDE\\_PDFU](#) is TRUE.

**Example 1**

1. 0x01 - Corresponds to Bootloader Application
2. 0x02 - Corresponds to Fixed Application
3. 0x03 - Corresponds to Updatable Application

**Example 2**

```
#define MCHP_PSF_HOOK_GETCURRENT_IMAGEBANK()  getCurrentImageBank()
UINT8 getCurrentImageBank(void)
{
    return u8ImageBankIndex;
}
```

## 10.2.11.4 MCHP\_PSF\_HOOK\_IS\_PDFU\_ALLOWED\_NOW

**C**

```
#define MCHP_PSF_HOOK_IS_PDFU_ALLOWED_NOW 0
```

**Description**

1. When the PD Firmware Update is allowed, the PDFU Initiator can perform firmware upgrade by the PDFU Sequence
2. When the PD Firmware Update is not allowed, the PDFU Initiator is responded with the error code during the Reconfiguration phase.

Example scenario of When PDFU cannot be allowed: Assuming a product with firmware update capability through CC and I2C as well. In an example, if the firmware upgrade through I2C is already in progress, then PDFU through CC interface shall not be allowed. To handle such situations, this macro shall return the current status of allow-ability of firmware upgrade.

**Preconditions**

This function is invoked by the PD Firmware Update State-machine during the Reconfiguration Phase (On reception PDFU\_INITIATE Command).

**Returns**

UINT8 value - Shall return the run time status whether PDFU via CC is allowed now or not. 0x00 - PDFU Not Allowed. 0x01 - PDFU Allowed.

**Remarks**

User definition of this Hook function is mandatory in fixed as well as updatable when **INCLUDE\_PDFU** is TRUE.

**Example**

```
#define MCHP_PSF_HOOK_IS_PDFU_ALLOWED_NOW  isPdfuAllowedNow()
UINT8 isPdfuAllowedNow(void)
{
    return u8PdfuAllow;
}
```

## 10.2.11.5 MCHP\_PSF\_HOOK\_PROGRAM\_FWBLOCK

**C**

```
#define MCHP_PSF_HOOK_PROGRAM_FWBLOCK(u8pObj,u16Len) 0
```

**Description**

This hook is invoked during the Transfer Phase on the successful reception event of every PDFU\_DATA packet. It is responsible for updating the Firmware data to the memory and identifying any errors during the Firmware flash.

**Preconditions**

Only during the Policy Engine State-Reconfigure Phase or Transfer phase this function hook will be invoked.

**Parameters**

Parameters	Description
u8pObj	UINT8 Pointer to PDFU_DATA packet payload Buffer.
u8pObj[0]	Reserved field Contains PD FW Version.
u8pObj[1]	Reserved field Contains Msg Type which is PDFU_DATA 0x83.
u8pObj[2]	LSB of Data Block Index.
u8pObj[3]	MSB of Data Block
Index u8pObj[4..260]	Firmware Image data upto 256 bytes where the Data block index is used to calculate the Physical memory address to which the current data block corresponds to 16 bit parameter.
u16Len	Indicates the length of the Firmware data contained in the packet.

**Returns**

Returns ePE\_PDFU\_RESPONSE\_CODE Type Return Value - The Status of the Program Operation.

1. ePE\_FWUP\_OK - Upon successful flash operation.
2. ePE\_FWUP\_errVERIFY - When verification of the flash operation failed.
3. ePE\_FWUP\_errADDRESS - When data block index is out of range.

**Remarks**

User definition of this Hook function is mandatory when **INCLUDE\_PDFU** is TRUE.

**Example**

```
#define MCHP_PSF_HOOK_PROGRAM_FWBLOCK(u8pObj, u16Len)
    PDFW_ProcessPDFUDataRequest(u8pObj, u16Len)
ePE_PDFU_RESPONSE_CODE PDFW_ProcessPDFUDataRequest( UINT8 u8RequestBuffer,
                                                    UINT16 u16RequestLength)
{
    UINT16 u16DataBlockIndex = *((UINT16*)&u8RequestBuffer[2]);
    u32ProgAddr = CalculateAddress(u16DataBlockIndex);
    if( u32ProgAddr < 0xFFFFFu )
    {
        ProgramMemoryCB(u32ProgAddr, &u8RequestBuffer[4u],u16RequestLength);
        ReadMemoryCB(u32ProgAddr, &u8ResponseBuffer[0u],u16RequestLength);
        //Compare data written and read
        if (0 == MCHP_PSF_HOOK_MEMCMP(&u8ResponseBuffer[0],
                                     &u8RequestBuffer[4], u16RequestLength))
        {
            //Set the status as OK
            u8Status = ePE_FWUP_OK;
        }
        else
        {
            //Verification Stage failure
            u8Status = ePE_FWUP_errVERIFY;
        }
    }
    else
    {
        u8Status = ePE_FWUP_errADDRESS;
    }

    return u8Status;
}
```

**10.2.11.6 MCHP\_PSF\_HOOK\_VALIDATE\_FIRMWARE****C**

```
#define MCHP_PSF_HOOK_VALIDATE_FIRMWARE 0
```

**Description**

This hook is invoked during the validation phase on reception of every PDFU Validation Request. It is responsible for validating the Firmware data in the memory. It shall return the progress status of the Validation on every invocation. If the Status indicates "On going" then the Validation command will be responded with the configured Wait time `CONFIG_VALIDATION_PHASE_WAITTIME`. Validation Method can be any custom method as required by the User.

**Preconditions**

Multiple invocations of the function hook is possible from PDFU Validation phase. Until the Validation process is complete, for every request of PDFU\_VALIDATION command this function will be invoked. The definition of this function shall include 1) Starting the Validation process on the First call, 2) Returning the Status of the Validation process during subsequent invocation of the function.

**Returns**

Returns the UINT8 Status of the Validation Operation. It take following values

0x00u - PE\_FWUP\_VALIDATION\_SUCCESSFUL

0x01u - PE\_FWUP\_VALIDATION\_INPROGRESS

0x02u - PE\_FWUP\_VALIDATION\_FAILURE

**Remarks**

User definition of this Hook function is mandatory when `INCLUDE_PDFU` is TRUE

**Example**

```
#define MCHP_PSF_HOOK_VALIDATE_FIRMWARE() PDFW_ProcessPDFUDataRequest()
UINT8 PDFW_ProcessPDFUValidateRequest(void)
{
    The definition of this function shall include
    1) Starting the Validation process on the First call,
    2) Returning the Status of the Validation process during subsequent invocation of
the function.
}
```

## 10.2.12 Hooks to Read Output Voltage and Current

**Macros**

	Name	Description
	<code>MCHP_PSF_HOOK_GET_OUTPUT_CURRENT_IN_mA</code>	Gets the output current.
	<code>MCHP_PSF_HOOK_GET_OUTPUT_VOLTAGE_IN_mV</code>	Gets the current output voltage driven in the VBUS.

**Description**

This section contains the hooks that are needed to read the Voltage and Current parameters from DC DC Controller.

### 10.2.12.1 MCHP\_PSF\_HOOK\_GET\_OUTPUT\_CURRENT\_IN\_mA

**C**

```
#define MCHP_PSF_HOOK_GET_OUTPUT_CURRENT_IN_mA 0xFFFFFFFF
```

**Description**

This hook is used when PSF needs to know about the current drawn from external DC\_DC controller. The function should be defined with return type UINT32 and UINT8 type as input parameter. If the DC\_DC controller does not have the capability to measure output current, return 0xFFFFFFFF to denote the feature is not supported.

**Preconditions**

Output Current shall be returned in terms of mA.

**Returns**

None.

**Remarks**

User definition of this Hook function is mandatory when `INCLUDE_PD_SOURCE_PPS` is defined as '1'.

**Example**

```
#define MCHP_PSF_HOOK_GET_OUTPUT_CURRENT_IN_mV(u8PortNum)
DCDC_GetOutCurrent(u8PortNum)
UINT32 DCDC_GetOutCurrent(UINT8 u8PortNum)
{
    // return Output current driven by the external DC_DC controller
    // in terms of mA.
}
```

## 10.2.12.2 MCHP\_PSF\_HOOK\_GET\_OUTPUT\_VOLTAGE\_IN\_mV

**C**

```
#define MCHP_PSF_HOOK_GET_OUTPUT_VOLTAGE_IN_mV 0xFFFFFFFF
```

**Description**

This hook is used when PSF needs to know about the present voltage driven by external DC\_DC controller. The function should be defined with return type UINT32 and UINT8 type as input parameter. If the DC\_DC controller does not have the capability to measure output voltage, return 0xFFFFFFFF to denote the feature is not supported.

**Preconditions**

Output voltage shall be returned in terms of milliVolts.

**Returns**

None.

**Remarks**

User definition of this Hook function is mandatory when `INCLUDE_PD_SOURCE_PPS` is defined as '1'.

**Example**

```
#define MCHP_PSF_HOOK_GET_OUTPUT_VOLTAGE_IN_mV(u8PortNum)
DCDC_GetOutVoltage(u8PortNum)
UINT32 DCDC_GetOutVoltage(UINT8 u8PortNum)
{
    // return Output voltage driven by the external DC_DC controller
    // in terms of milliVolts
}
```

## 10.3 APIs to be called by the User application

**Functions**

	Name	Description
≡	<a href="#">MchpPSF_Init</a>	PSF initialization API
≡	<a href="#">MchpPSF_PDTimerHandler</a>	PD Timer Interrupt Handler
≡	<a href="#">MchpPSF_UPDIrqHandler</a>	<a href="#">UPD350</a> IRQ Interrupt Handler
≡	<a href="#">MchpPSF_RUN</a>	PSF state machine run API

**Description**

PSF APIs have to be called by the SoC User\_Application for PSF to run as well as to get to know about the Timer and IRQ\_N interrupt from the [UPD350](#). This section explains them in detail

---

## 10.3.1 MchpPSF\_Init

**C**

```
UINT8 MchpPSF_Init();
```

**Description**

This API should be called by the SOC layer to initialize the PSF stack and [UPD350](#) Hardware.

**Preconditions**

API should be called before [MchpPSF\\_RUN\(\)](#).

**Returns**

TRUE - Stack and [UPD350](#) HW successfully initialized. FALSE - Stack and [UPD350](#) HW initialization failed.

**Remarks**

For the current PSF implementation, return value is not used.

**With SAMD20 SOC environment configured for CPU frequency of 48MHZ, this API took maximum of 3.488ms and 6.182ms execution time for 1 and 2 port Source only solution respectively.**

---

## 10.3.2 MchpPSF\_PDTimerHandler

**C**

```
void MchpPSF_PDTimerHandler();
```

**Description**

This API is used to handle the PD Timer (Software timer) Interrupt, User should call this API whenever the hardware timer interrupt triggered.

**Preconditions**

This API should be called inside the Hardware timer ISR.

**Returns**

None

**Remarks**

With SAMD20 environment configured for CPU frequency 48MHZ, this API took maximum of 262us execution time for both 1 and 2 port solution.

---

## 10.3.3 MchpPSF\_UPDIrqHandler

**C**

```
void MchpPSF_UPDIrqHandler(  
    UINT8 u8PortNum  
);
```

**Description**

This API handles the [UPD350](#) IRQ\_N Interrupt, User should call this API when the IRQ line interrupt triggered to the SOC. This API will services and then clear the Alert interrupt for corresponding port.

**Preconditions**

This API should be called inside the GPIO ISR for IRQ interrupt

**Parameters**

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to ( <a href="#">CONFIG_PD_PORT_COUNT</a> -1).

**Returns**

None

**Remarks**

With SAMD20 environment configured for CPU frequency 48MHZ, this API took maximum of 3.98us and 5.8us execution time for 1 and 2 port solution respectively.

---

## 10.3.4 MchpPSF\_RUN

**C**

```
void MchpPSF_RUN( ) ;
```

**Description**

This API is used to run the PSF state machine. For single threaded environment, it should be called repeatedly within a while(1).

**Preconditions**

API should be called only after [MchpPSF\\_Init\(\)](#).

**Returns**




None

**Remarks**




In SAMD20 environment where CPU frequency is configured for 48MHZ and single threaded environment where MchpPSF\_RUN is called in a while(1) loop, it took maximum of 74.57us and 0.1439 ms execution time for 1 port and 2 source port solution respectively. In Multi threaded SAMD20 configured for 48MHz environment, MchpPSF\_RUN can be called for every 1ms to 5ms for Successful 2-Port Source only operation.

# 11 Notification callback from PSF

## Enumerations

	Name	Description
	<u>MCHP_PSF_NOTIFICATION</u>	PSF notification enum
	<u>eMCHP_PSF_NOTIFY_IDLE</u>	PSF Notify Idle enum.
	<u>ePSF_NOTIFY_IDLE</u>	PSF Notify Idle enum.

## Macros

	Name	Description
	<u>MCHP_PSF_NOTIFY_CALL_BACK</u>	Notifies the USER_APPLICATION about various PD events from PSF.
	<u>MCHP_PSF_HOOK_NOTIFY_IDLE</u>	Hook to notify entry of Policy Engine and Type C State Machine into idle state
	<u>MCHP_PSF_HOOK_PDTIMER_EVENT</u>	Hook for PD timer timeout event

## Description

This section gives details on all the notifications given by PSF through its callback.

## 11.1 MCHP\_PSF\_NOTIFICATION Enumeration

### C

```
enum MCHP_PSF_NOTIFICATION {
    eMCHP_PSF_TYPEC_DETACH_EVENT = 1,
    eMCHP_PSF_TYPEC_CC1_ATTACH,
    eMCHP_PSF_TYPEC_CC2_ATTACH,
    eMCHP_PSF_TYPEC_ERROR_RECOVERY,
    eMCHP_PSF_UPDS_IN_IDLE,
    eMCHP_PSF_VCONN_PWR_FAULT,
    eMCHP_PSF_VBUS_PWR_FAULT,
    eMCHP_PSF_PORT_POWERED_OFF,
    eMCHP_PSF_RECOVERED_FRM_VCONN_PWR_FAULT,
    eMCHP_PSF_RECOVERED_FRM_VBUS_PWR_FAULT,
    eMCHP_PSF_PE_SRC_DISABLED,
    eMCHP_PSF_PD_CONTRACT_NEGOTIATED,
    eMCHP_PSF_HARD_RESET_COMPLETE,
    eMCHP_PSF_SINK_CAPS_RCVD,
    eMCHP_PSF_SINK_CAPS_NOT_RCVD,
    eMCHP_PSF_CAPS_MISMATCH,
    eMCHP_PSF_NEW_SRC_CAPS_RCVD,
    eMCHP_PSF_SINK_ALERT_RCVD,
    eMCHP_PSF_SINK_STATUS_RCVD,
    eMCHP_PSF_SINK_STATUS_NOT_RCVD,
    eMCHP_PSF_VCONN_SWAP_COMPLETE,
    eMCHP_PSF_VCONN_SWAP_RCVD,
    eMCHP_PSF_VCONN_SWAP_NO_RESPONSE_RCVD,
    eMCHP_PSF_FR_SWAP_COMPLETE,
    eMCHP_PSF_PR_SWAP_COMPLETE,
    eMCHP_PSF_PR_SWAP_RCVD,
    eMCHP_PSF_PR_SWAP_NO_RESPONSE_RCVD,
    eMCHP_PSF_DR_SWAP_COMPLETE,
    eMCHP_PSF_DR_SWAP_RCVD,
    eMCHP_PSF_DR_SWAP_NO_RESPONSE_RCVD,
    eMCHP_PSF_CABLE_IDENTITY_DISCOVERED,
    eMCHP_PSF_CABLE_IDENTITY_NAKED,
```



```

eMCHP_PSF_VDM_RESPONSE_RCVD,
eMCHP_PSF_VDM_RESPONSE_NOT_RCVD,
eMCHP_PSF_VDM_REQUEST_RCVD,
eMCHP_PSF_VDM_AMS_COMPLETE,
eMCHP_PSF_HPD_ENABLED,
eMCHP_PSF_HPD_EVENT_HIGH,
eMCHP_PSF_HPD_EVENT_LOW,
eMCHP_PSF_HPD_EVENT_IRQ_HPD,
eMCHP_PSF_HPD_DISABLED,
eMCHP_PSF_ALT_MODE_ENTRY_FAILED,
eMCHP_PSF_PORT_DISABLED,
eMCHP_PSF_PORT_ENABLED
};

```

## Description

eMCHP\_PSF\_NOTIFICATION enum defines the all the notifications PSF can notify via [MCHP\\_PSF\\_NOTIFY\\_CALL\\_BACK](#).

**eMCHP\_PSF\_TYPEC\_DETACH\_EVENT:** This event is posted by PSF Type C state machine when port partner Detach event is detected.

**eMCHP\_PSF\_TYPEC\_CC1\_ATTACH:** This event is posted by PSF Type C state machine when port partner Type C attach is detected in CC1 pin.

**eMCHP\_PSF\_TYPEC\_CC2\_ATTACH:** This event is posted by PSF Type C state machine when port partner Type C attach is detected in CC2 pin.

**eMCHP\_PSF\_TYPEC\_ERROR\_RECOVERY:** PSF notifies Type-C Error Recovery condition via this notification. For this notification, PSF expects a return value to decide whether to handle Error Recovery. When user application returns TRUE, PSF enters Type-C Error Recovery state and resolves the error condition. The user application may also return FALSE, in which case, the user application will itself handle Error Recovery condition by raising a Port disable client request (BIT(0) of gasCfgStatusData.sPerPortData[u8PortNum].u32ClientRequest variable).

**eMCHP\_PSF\_UPDS\_IN\_IDLE:** This event is posted by Power management control. PSF runs an algorithm in the background for Power management control. The Idle timeout value maintained by PSF is 15 seconds. If there is no activity in [UPD350](#) for 15 seconds, the corresponding [UPD350](#) is put in low power mode. When all the UPD350s present in the system enter low power mode, eMCHP\_PSF\_UPDS\_IN\_IDLE is notified. User can then choose to put the SOC into a low power state based on this notification. This notification occurs only when [INCLUDE\\_POWER\\_MANAGEMENT\\_CTRL](#) is defined as 1.

**eMCHP\_PSF\_VCONN\_PWR\_FAULT:** [UPD350](#) has VCONN comparators to detect VCONN OCS faults. This event is notified when VCONN OCS fault is detected by [UPD350](#). For this notification, PSF expects a return value to decide whether to handle the fault occurred. When user returns TRUE for VCONN power fault, Incase of explicit contract, if VCONN power fault count is less than u8VCONNMaxFaultCnt, PSF DPM power fault manager handles it by sending Hard Reset. If the count exceeds max fault count, VCONN is powered off until physical detach of port partner. Incase of implicit contract, PSF handles by entering TypeC Error Recovery. This notification occurs only when [INCLUDE\\_POWER\\_FAULT\\_HANDLING](#) is defined as 1.

**eMCHP\_PSF\_VBUS\_PWR\_FAULT :** PSF notifies all VBUS power fault VBUS Over voltage, VBUS under voltage, VBUS OCS via this notification. For this notification, PSF expects a return value to decide whether to handle the fault occurred. When user returns TRUE for power fault, Incase of explicit contract, if power fault count is less than u8VBUSMaxFaultCnt, PSF DPM power fault manager handles it by sending Hard Reset. When the power fault count exceeds the max fault count, CC termination on the port is removed until the physical detach of the port partner. Incase of implicit contract, PSF handles by entering TypeC Error Recovery. This notification occurs only when [INCLUDE\\_POWER\\_FAULT\\_HANDLING](#) is defined as 1.

**eMCHP\_PSF\_RECOVERED\_FRM\_VCONN\_PWR\_FAULT :** This event is notified when a port is recovered from VCONN power fault if the VCONN is stable for u16PowerGoodTimerInms without any further fault occurrence. PSF clears VCONN fault recorded with this notification.

**eMCHP\_PSF\_RECOVERED\_FRM\_VBUS\_PWR\_FAULT :** This event is notified when a port recovers from VBUS power fault when the VBUS is stable for u16PowerGoodTimerInms without any further VBUS fault occurrence. PSF clears the VBUS fault count it recorded with this notification.

**eMCHP\_PSF\_PORT\_POWERED\_OFF** : This event is used by PSF to notify application when the port has been powered off as a result of VBUS or VCONN fault count exceeding the values present in u8VBUSMaxFaultCnt or u8VCONNMaxFaultCnt variables respectively within u16PowerGoodTimerInms time period. When a port is powered off, it stops sourcing/sinking power and waits for a detach. The port can be revived only when the partner is detached and attached again.

**eMCHP\_PSF\_PD\_CONTRACT\_NEGOTIATED** : PSF notifies when PD contract is established with the Port partner.

**eMCHP\_PSF\_HARD\_RESET\_COMPLETE** : This event is posted when a Hard Reset is sent or received by the port and as a result of which the port has entered into an implicit contract and is going to reestablish the explicit contract with the port partner.

**eMCHP\_PSF\_PE\_SRC\_DISABLED** : This event is posted when Source policy engine enters PE\_SRC\_DISABLED indicating Power Delivery capable port partner is not present and only Implicit contract is established with Sink port partner.

**eMCHP\_PSF\_SINK\_CAPS\_RCVD** : This event is used by PSF to notify application when Sink capabilities has been received from Port Partner in response to the Get\_Sink\_Cap message initiated by PSF. Application can read the Sink Capabilities by accessing gasCfgStatusData.sPerPortData[u8PortNum].u32aPartnerSinkPDO array. This event is applicable only when the port is configured for Source only or DRP configuration.

**eMCHP\_PSF\_SINK\_CAPS\_NOT\_RCVD** : This event is used by PSF to notify application when Sink capabilities has not been received from Port Partner within tSenderResponseTimer as a response to the Get\_Sink\_Cap message initiated by PSF. This event is applicable only when the port is configured for Source only or DRP configuration.

**eMCHP\_PSF\_CAPS\_MISMATCH** : It is notified by PSF when there is a capability mismatch with Source partner PDOs in a PD negotiation. This event is applicable only when PSF is operating as Sink or the power role of the port is resolved as Sink during DRP operation.

**eMCHP\_PSF\_NEW\_SRC\_CAPS\_RCVD** : It is notified by PSF when new source capability message is received from the Source Partner. This event is applicable only when PSF is operating as Sink or the power role of the port is resolved as Sink during DRP operation.

**eMCHP\_PSF\_SINK\_ALERT\_RCVD** : This event is used by PSF to notify application when PD Alert message has been received from Sink Partner. Application can read the alert data by accessing gasCfgStatusData.sPPSPerPortData[u8PortNum].u32PartnerAlert. This event is applicable only when PSF is operating as Source or the power role of the port is resolved as Source during DRP operation.

**eMCHP\_PSF\_SINK\_STATUS\_RCVD** : This event is used by PSF to notify application when Sink Status has been received from Port Partner in response to the Get\_Status message initiated by PSF. Application can read the Sink Status by accessing gasCfgStatusData.sPPSPerPortData[u8PortNum].u8aPartnerStatus array. This event is applicable only when PSF is operating as Source or the power role of the port is resolved as Source during DRP operation.

**eMCHP\_PSF\_SINK\_STATUS\_NOT\_RCVD** : This event is used by PSF to notify application when Sink Status has not been received from Port Partner within tSenderResponseTimer as a response to the Get\_Status message initiated by PSF. gasCfgStatusData.sPPSPerPortData[u8PortNum].u8aPartnerStatus[6] would have 0 when this notification is posted. This event is applicable only when PSF is operating as Source or the power role of the port is resolved as Source during DRP operation.

**eMCHP\_PSF\_VCONN\_SWAP\_COMPLETE** : This notification will be posted when the VCONN Swap is completed and the VCONN roles of the both the partners are changed successfully. This notification will also be posted when the VCONN Swap initiated by either of the partners is rejected by the other. Users can know the current status of VCONN through u32PortConnectStatus.

**eMCHP\_PSF\_VCONN\_SWAP\_RCVD** : This notification will be posted by PSF when a VCONN Swap message is received from port partner. Application can make use of this event to dynamically update the VCONN Swap Policy Configuration through u16SwapPolicy based on which the VCONN Swap request would be accepted or rejected by PSF.

**eMCHP\_PSF\_VCONN\_SWAP\_NO\_RESPONSE\_RCVD** : This notification will be posted by PSF when a VCONN Swap message has been initiated by PSF and no response has been received from the port partner.

**eMCHP\_PSF\_FR\_SWAP\_COMPLETE** : This notification will be posted when the FR Swap is completed and the power roles of the both the partners are changed successfully. Users can know the current power role status through u32PortConnectStatus. This event is applicable only when PSF is operating as DRP and FR\_Swap is supported.

**eMCHP\_PSF\_PR\_SWAP\_COMPLETE** : This notification will be posted when the PR Swap is completed and the power roles of the both the partners are changed successfully. This notification will also be posted when the PR\_Swap initiated by either of the partners is rejected by the other. Users can know the current power role status through `u32PortConnectStatus`. This event is applicable only when PSF is operating as DRP.

**eMCHP\_PSF\_PR\_SWAP\_RCVD** : This notification will be posted by PSF when a Power Role Swap message is received from port partner. Application can make use of this event to dynamically update the Power Role Swap Policy Configuration through `u16SwapPolicy` based on which the Power Role Swap request would be accepted or rejected by PSF. This event is applicable only when PSF is operating as DRP.

**eMCHP\_PSF\_PR\_SWAP\_NO\_RESPONSE\_RCVD** : This notification will be posted by PSF when a Power Role Swap message has been initiated by PSF and no response has been received from the port partner. This event is applicable only when PSF is operating as DRP.

**eMCHP\_PSF\_DR\_SWAP\_COMPLETE**: This notification will be posted when the DR Swap is completed and the data roles of the both the partners are changed successfully. This notification will also be posted when the DR\_Swap initiated by either of the partners is rejected by the other. Users can know the current data role status through `u32PortConnectStatus`.

**eMCHP\_PSF\_DR\_SWAP\_RCVD**: This occurs whenever there is a DR\_SWAP message from port partner. User can modify `u16SwapPolicy` dynamically to Accept or Reject DR\_SWAP. Note for dynamic DR\_SWAP initiation by PSF, client request has to be raised apart from dynamic change of `u16SwapPolicy` configuration.

**eMCHP\_PSF\_DR\_SWAP\_NO\_RESPONSE\_RCVD** : This is posted by PSF when Response Timer times out and there is no response from the port partner for the DR\_SWAP initiated.

**eMCHP\_PSF\_CABLE\_IDENTITY\_DISCOVERED** : This event is used by PSF to notify the application when the Cable Identity has been received from Cable in response to the VDM:Discover Identity request initiated by PSF. Application can read the cable identity by accessing the `gasCfgStatusData.sPerPortData[u8PortNum].u32aCableIdentity` array. This event is applicable only when the port is operating as both VBUS Source and VCONN Source.

**eMCHP\_PSF\_CABLE\_IDENTITY\_NAKED** : This event is used by PSF to notify the application of the scenarios Good\_CRC not received, no response, NAK response and BUSY response from cable for SOP' VDM:Discover Identity request initiated by PSF. This event is applicable only when the port is operating as both VBUS Source and VCONN Source.

**eMCHP\_PSF\_VDM\_RESPONSE\_RCVD** : This notification will be posted by PSF when the partner has sent an ACK/NAK response for the VDM request initiated by PSF within VDM Response time. This notification will not be posted for a BUSY response. Instead, PSF will re-initiate the request after `PE_VDM_BUSY_TIMEOUT_MS`. This will be done for a maximum Busy count of 5 times.

**eMCHP\_PSF\_VDM\_RESPONSE\_NOT\_RCVD** : This notification will be posted by PSF when the VDM response timer has timed out due to no response from the partner for the VDM request sent or when the partner has responded BUSY for a maximum of 5 times and the VDM request will not be initiated by PSF anymore or when the partner has sent Not Supported for the VDM request.

**eMCHP\_PSF\_VDM\_REQUEST\_RCVD** : This notification will be posted by PSF when a VDM request received from the partner is evaluated and determined to be ACKed or when the request is a SVID specific command i.e it needs evaluation by the user application. In case of Structured VDM commands like Discover Identity, Discover SVIDs, Discover Modes, Enter Mode, and Exit Mode, PSF expects a return value from the application to send ACK/NAK response. If the application returns TRUE, PSF will send ACK response. The application may also return FALSE in which case PSF will send NAK response. If after the evaluation by PSF, it is determined that these commands are to be NAKed, then PSF will send NAK response without notifying the user application. For SVID specific commands, PSF will not send any response since it is expected that the user application will send the response by raising a VDM client request within the VDM response time specified by USB PD Specification.

**eMCHP\_PSF\_VDM\_AMS\_COMPLETE** : This notification will be posted by PSF when Good CRC is received for the VDM request initiated by PSF that consists of no command response or when Good CRC is received for the VDM response sent in response to the VDM request received from partner.

**eMCHP\_PSF\_HPD\_ENABLED** : This notification will be posted by PSF when an Hot Plug Detect (HPD) support is enabled.

**eMCHP\_PSF\_HPD\_EVENT\_HIGH** : This notification will be posted by PSF when an HPD\_HIGH event has occurred.

**eMCHP\_PSF\_HPD\_EVENT\_LOW** : This notification will be posted by PSF when an HPD\_LOW event has occurred.

**eMCHP\_PSF\_HPD\_EVENT\_IRQ\_HPD** : This notification will be posted by PSF when an IRQ\_HPD event has occurred.

**eMCHP\_PSF\_HPD\_DISABLED** : This notification will be posted by PSF when an Hot Plug Detect (HPD) support is disabled.

**eMCHP\_PSF\_ALT\_MODE\_ENTRY\_FAILED** : This notification will be posted by PSF when AltModeEntry timer has timed out because of No 'Enter Mode' command from the port partner

**eMCHP\_PSF\_PORT\_DISABLED** : This event is used by PSF to notify the application that a port has been disabled as a result of port disable client request (BIT[0] in gasCfgStatusData.sPerPortData[u8PortNum].u32ClientRequest variable). When a port is disabled, any connection to the port is prevented by removing all terminations from the CC pins. A disabled port will continue to be in disabled state even after the port partner is physically detached and attached. The port can only be revived by a client request to enable the port (BIT[1] in gasCfgStatusData.sPerPortData[u8PortNum].u32ClientRequest variable).

**eMCHP\_PSF\_PORT\_ENABLED** : This event is used by PSF to notify the application that a port has been enabled as a result of port enable client request (BIT[1] in gasCfgStatusData.sPerPortData[u8PortNum].u32ClientRequest variable). When a port is enabled, Type-C attach and PD negotiation sequence happens again.

## Members

Members	Description
eMCHP_PSF_TYPEC_DETACH_EVENT = 1	Detach event has occurred
eMCHP_PSF_TYPEC_CC1_ATTACH	Port partner attached at CC1 orientation
eMCHP_PSF_TYPEC_CC2_ATTACH	Port partner attached at CC2 orientation
eMCHP_PSF_TYPEC_ERROR_RECOVERY	Error recovery condition has occurred and it will be handled by PSF if user application returns TRUE
eMCHP_PSF_UPDS_IN_IDLE	All the UPD350s are Idle and in low power mode
eMCHP_PSF_VCONN_PWR_FAULT	VCONN Power Fault has occurred
eMCHP_PSF_VBUS_PWR_FAULT	VBUS Power Fault has occurred
eMCHP_PSF_PORT_POWERED_OFF	Port powered off since fault count exceeded maximum fault count
eMCHP_PSF_RECOVERED_FRM_VCONN_PWR_FAULT	Port Recovered from VCONN power fault
eMCHP_PSF_RECOVERED_FRM_VBUS_PWR_FAULT	Port Recovered from VBUS Power fault
eMCHP_PSF_PE_SRC_DISABLED	Only Type C device is present, Partner does not support PD
eMCHP_PSF_PD_CONTRACT_NEGOTIATED	PD Contract established with port partner
eMCHP_PSF_HARD_RESET_COMPLETE	Hard Reset is sent or received by the port
eMCHP_PSF_SINK_CAPS_RCVD	Sink Caps received from Port Partner
eMCHP_PSF_SINK_CAPS_NOT_RCVD	Sink Caps not received from Port Partner
eMCHP_PSF_CAPS_MISMATCH	Capability mismatch with Source Port Partner
eMCHP_PSF_NEW_SRC_CAPS_RCVD	New source capability message is received from Source Partner
eMCHP_PSF_SINK_ALERT_RCVD	Alert message received from Sink Partner
eMCHP_PSF_SINK_STATUS_RCVD	Sink Status received from Sink Partner
eMCHP_PSF_SINK_STATUS_NOT_RCVD	Sink Status not received from Sink Partner
eMCHP_PSF_VCONN_SWAP_COMPLETE	VCONN Swap completed
eMCHP_PSF_VCONN_SWAP_RCVD	VCONN Swap Received from port partner
eMCHP_PSF_VCONN_SWAP_NO_RESPONSE_RCVD	No response from port partner for VCONN Swap sent
eMCHP_PSF_FR_SWAP_COMPLETE	Fast Role Swap completed
eMCHP_PSF_PR_SWAP_COMPLETE	Power Role Swap completed
eMCHP_PSF_PR_SWAP_RCVD	Power Role Swap Received from port partner
eMCHP_PSF_PR_SWAP_NO_RESPONSE_RCVD	No response from port partner for Power Role Swap sent
eMCHP_PSF_DR_SWAP_COMPLETE	Data Role Swap completed

eMCHP_PSF_DR_SWAP_RCVD	Data Role swap received from port partner
eMCHP_PSF_DR_SWAP_NO_RESPONSE_RCVD	No response from port partner for the DR_SWAP initiated
eMCHP_PSF_CABLE_IDENTITY_DISCOVERED	ACK received from cable for Discover Identity sent to SOP
eMCHP_PSF_CABLE_IDENTITY_NAKED	NAK received from cable for Discover Identity sent to SOP
eMCHP_PSF_VDM_RESPONSE_RCVD	Response received from partner for VDM request sent to SOP
eMCHP_PSF_VDM_RESPONSE_NOT_RCVD	No response from partner for VDM request sent to SOP
eMCHP_PSF_VDM_REQUEST_RCVD	VDM Request received from partner
eMCHP_PSF_VDM_AMS_COMPLETE	VDM AMS Completed
eMCHP_PSF_HPD_ENABLED	Indicates that HPD module is enabled
eMCHP_PSF_HPD_EVENT_HIGH	Indicates that HPD_HIGH event has occurred
eMCHP_PSF_HPD_EVENT_LOW	Indicates that HPD_LOW event has occurred
eMCHP_PSF_HPD_EVENT_IRQ_HPD	Indicates that IRQ_HPD event has occurred
eMCHP_PSF_HPD_DISABLED	Indicates that HPD module is disabled
eMCHP_PSF_ALT_MODE_ENTRY_FAILED	Alt Mode Entry Failed
eMCHP_PSF_PORT_DISABLED	Indicates that port is disabled successfully
eMCHP_PSF_PORT_ENABLED	Indicates that the port is enabled successfully

**Remarks**

None

## 11.2 MCHP\_PSF\_NOTIFY\_CALL\_BACK

**C**

```
#define MCHP_PSF_NOTIFY_CALL_BACK(u8PortNum, ePSFNotification)
```

**Description**

This hook is used by the various modules of PSF to notify the USER\_APPLICATION about different PD events such as Type-C Attach and Detach , Type-C Orientation. USER\_APPLICATION can define this hook function if it wants external handling of the PD events. Define relevant function that has UINT8, eMCHP\_PSF\_NOTIFICATION argument without return type.

**Preconditions**

None.

**Parameters**

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to (CONFIG_PD_PORT_COUNT-1).
ePSFNotification	Type of Notification occurred inside the stack. This argument can take one of the values from enum eMCHP_PSF_NOTIFICATION.

**Returns**

UINT8 - Except for eMCHP\_PSF\_VCONN\_PWR\_FAULT and eMCHP\_PSF\_VBUS\_PWR\_FAULT the return value is ignored by PSF. For eMCHP\_PSF\_VCONN\_PWR\_FAULT and eMCHP\_PSF\_VBUS\_PWR\_FAULT event, user can return TRUE - if the Power fault shall be handled by PSF FALSE - if the Power fault occurrence is ignored.

**Remarks**

User definition of this Hook function is mandatory

**Example**

```
#define MCHP_PSF_NOTIFY_CALL_BACK(u8PortNum, ePSFNotification)
```

```

        HookNotifyPDEvents(u8PortNum, ePSFNotification)
void HookNotifyPDEvents(UINT8 u8PortNum, eMCHP_PSF_NOTIFICATION ePSFNotification);
void HookNotifyPDEvents(UINT8 u8PortNum, eMCHP_PSF_NOTIFICATION ePSFNotification)
{
    // Return for Power fault notification
    // Implement user specific application as required
}

```

## 11.3 eMCHP\_PSF\_NOTIFY\_IDLE Enumeration

C

```

typedef enum ePSF_NOTIFY_IDLE {
    eIDLE_PE_NOTIFY,
    eIDLE_TYPEC_NOTIFY
} eMCHP_PSF_NOTIFY_IDLE;

```

### Description

eMCHP\_PSF\_NOTIFY\_IDLE enum notifies the Idle state in PSF

### Members

Members	Description
eIDLE_PE_NOTIFY	Notify Policy Engine Idle State
eIDLE_TYPEC_NOTIFY	Notify Type C Idle State

### Remarks

None

## 11.4 MCHP\_PSF\_HOOK\_NOTIFY\_IDLE

C

```

#define MCHP_PSF_HOOK_NOTIFY_IDLE(u8PortNum, eIDLESubState)

```

### Description

PSF calls this API to notify the entry of Policy Engine and Type C State Machine into idle state. The entry into idle state refers to when the state machine waits for an event from the partner or for the activated timer to get expired. The entry into Idle state of Policy Engine or Type C state machine is differentiated based on the enum argument passed

### Preconditions

None.

### Parameters

Parameters	Description
u8PortNum	Port number of the device. It takes value between 0 to (CONFIG_PD_PORT_COUNT-1).
eIDLESubState	Defines the idle notification of Policy Engine(eIDLE_PE_NOTIFY) or Type C State machine(eIDLE_TYPEC_NOTIFY)

### Returns

None.

### Remarks

User definition of this Hook function is not mandatory and would be useful in an RTOS environment

**Example**

```
#define MCHP_PSF_HOOK_NOTIFY_IDLE(u8PortNum, eIDLESubState)
PSF_IDLENotification(u8PortNum, eIDLESubState)
void PSF_IDLENotification(u8PortNum, eIDLESubState)
{
    if(eIDLESubState == eIDLE_PE_NOTIFY)
    {
        gu8PEIDLEFlag[u8PortNum] = TRUE;
    }
    else if (eIDLESubState == eIDLE_TYPEC_NOTIFY)
    {
        gu8TypeCIDLEFlag[u8PortNum] = TRUE;
    }
    else
    {
        //Do Nothing
    }
}
```

---

## 11.5 MCHP\_PSF\_HOOK\_PDTIMER\_EVENT

**C**

```
#define MCHP_PSF_HOOK_PDTIMER_EVENT
```

**Description**

This hook is used when PD timer expires for the given event to call the callback function.

**Preconditions**

None

**Returns**

None.

**Remarks**

User definition of this Hook function is not mandatory and would be useful in an RTOS environment

**Example**

```
#define MCHP_PSF_HOOK_PDTIMER_EVENT () PSF_IdleExit()
void PSF_IdleExit()
{
    gbyIdleFlag = FALSE;
}
```