

Getting started with PSF

Information contained in this publication regarding device applications and the like is provided only for your convenience and can be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE**.**

Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

|  |  |  |  |
| --- | --- | --- | --- |
| Microchip Technology, Inc. | | | Microchip Technology, Incorporated  2355 W. Chandler Boulevard  Chandler, Arizona 85224  480/792-7416 |
| REV | DATE | ORIGINATOR | DESCRIPTION OF CHANGE |
| 0.1 | 26-Aug-19 | Poornima R | Initial revision |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc23860225)

[1.1 Features Overview 5](#_Toc23860226)

[1.2 Terms and abbreviations 5](#_Toc23860227)

[1.3 References 7](#_Toc23860228)

[2 PSF FW Architecture Overview 7](#_Toc23860229)

[2.1 Device Policy Manager(DPM) 8](#_Toc23860230)

[2.2 Policy Engine 8](#_Toc23860231)

[2.3 Protocol Layer 8](#_Toc23860232)

[2.4 Type-C Connector Management 8](#_Toc23860233)

[2.5 Interrupts/Timer Management 8](#_Toc23860234)

[2.6 Port Power Management 8](#_Toc23860235)

[2.7 Power Delivery Firmware Update (PDFU) 9](#_Toc23860236)

[3 Hardware Requirements for PSF 9](#_Toc23860237)

[3.1 UPD 350 9](#_Toc23860238)

[3.2 Hardware Communication Interface 9](#_Toc23860239)

[3.2.1 SPI Master 9](#_Toc23860240)

[3.3 PIOs for UPD350 Alert 10](#_Toc23860241)

[3.4 PIO for UPD350 Reset 10](#_Toc23860242)

[3.5 Hardware Timer 10](#_Toc23860243)

[3.6 DC-DC convertor 10](#_Toc23860244)

[4 Memory Requirement 11](#_Toc23860245)

[4.1 32-bit MCU 11](#_Toc23860246)

[4.2 16-bit MCU 11](#_Toc23860247)

[4.3 8-bit MCU 12](#_Toc23860248)

[5 Multi-Port Support Requirement 12](#_Toc23860249)

[6 Endianness 12](#_Toc23860250)

[7 Supported/Not Supported PD Features and Messages 12](#_Toc23860251)

[7.1 Supported/Not Supported PD features 12](#_Toc23860252)

[7.1.1 Supported features 12](#_Toc23860253)

[7.1.2 Not supported features 12](#_Toc23860254)

[7.2 Supported/Not Supported PD messages 12](#_Toc23860255)

[7.2.1 Supported PD Messages 12](#_Toc23860256)

[7.2.2 Unsupported PD Messages 13](#_Toc23860257)

[8 PSF Configurability TBD – To be renamed 14](#_Toc23860258)

[9 Directory structure 14](#_Toc23860259)

[9.1 PSF Source 14](#_Toc23860260)

[9.2 SOC Portable 15](#_Toc23860261)

[9.3 Demo Applications 16](#_Toc23860262)

[10 Frequently Asked Questions (FAQ) 16](#_Toc23860263)

# Introduction

USB Power Delivery Software Framework (PSF) with USB-PD Port Controller UPD350 is an effective USB-PD solution compliant to USB-PD 3.0 Specification.

PSF stack is designed to run on different MCU Hardware platform. Versatility towards different HW platform is achieved through flexibility towards configurability of PSF stack.

This section describes PSF stack configurability features & directory structure.

This document is a user guide to PSF and it covers the following topics,

* PSF overview and its architecture
* Memory & Hardware level requirements
* PSF stack directory structure
* Supported/Not supported features
* Requirements for expanding the PD-solution for multi-ports
* Supported/Not supported PD messages
* Reference links to related PSF documents

## Features Overview

Following are the key features of PSF stack,

* Compliant to USB Power Delivery 3.0 & Type-C specification V1.3
* Multiport support
* USB-PD Source-only or Sink-only port specific configurability
* FW update through CC support compliant to USB PD Firmware Update Specification R1.0.

## Terms and abbreviations

The following are the list of terms and abbreviations used in this document.

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| PSF | USB Power Delivery Software Framework |
| DPM | Device Policy Manager |
| FW | Firmware |
| PE | Policy Engine |
| USB-PD | USB Power Delivery |
| UPD 350 | Microchip USB Power Delivery port controller |
| UPD 301 | Microchip Stand-Alone Type-C/PD Port Controller |
| CC | Configuration Channel |
| IRQ | Interrupt Request Line |
| MCU | Microcontroller |
| USBPD3 | USB Power Delivery Specification 3.0 |
| Dynamic power balancing | For Sources with multiple port, monitoring and balancing power requirements across these ports. |

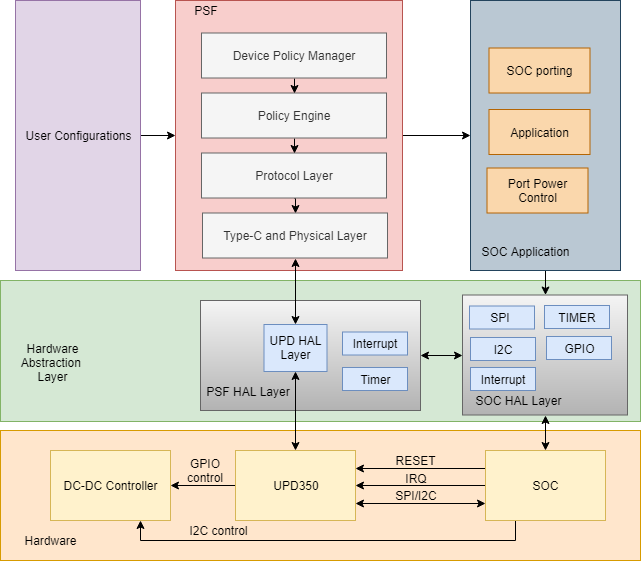
## References

Following are the list of documents for reference,

* Microchip UPD350 Datasheet
* USB Power Delivery 3.0 Specification Revision 1.1 + ECNs
* USB Type-C Specification Revision 1.3

# PSF FW Architecture Overview

PSF is a generic USB-PD solution with UPD350. The Architecture and configurability of the stack is designed in such a way that it doesn’t limit number of ports nor the port functionality. It also provides flexibility for future enhancements by the user.



## Device Policy Manager(DPM)

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 interfaces with Policy Engine Port specifically
* Informs Policy Engine Cable/Device detection as Source/Sink.

## Policy Engine

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.

## 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.

## 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

## Interrupts/Timer Management

* Interrupt Management handles all the external interrupt from UPD350 Silicon.
* Timer Management involves handling of all the active software timer’s timeouts based on the interrupt from periodic hardware timer.

## Port Power Management

Port power management as a Source/Sink for multiple ports supports,

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

## Power Delivery Firmware Update (PDFU)

FW update through CC as per USB PD Firmware Update Specification R1.0.

# System level integration of PSF

This section describes hardware requirements for Zeus Stack.

## UPD 350

UPD350 silicon is required for each port specifically. Zeus Stack supports UPD350 Rev A Silicon part with SPI Companion support i.e. UPD350/B, UPD350/D & UPD350/F parts of UPD350 Rev A.

## Hardware Communication Interface

### SPI Master

SPI master is required as Zeus stack interacts with UPD350 only through SPI interface. UPD350 SPI Slave supports maximum of 25MHz.

MCU SPI Master

CS 2

CS n

CS 1

SPI Bus

Port n UPD350

Port 2 UPD350

Port 1 UPD350

…..

*Note: ‘n’ denotes maximum number of ports.*

*cs denotes Chip select or slave select.*

2 Port Source and Sink solution is tested SPI Master with 8MHz.

## PIOs for UPD350 Alert

Alert line of each UPD350 must be mapped to individual GPIOs of MCU in the HW platform.

MCU GPIO Control

GPIO n

GPIO 2

GPIO 1

Port 1 UPD350 IRQ

Port n UPD350 IRQ

…..

Port 2 UPD350 IRQ

*Note: ‘n’ denotes maximum number of ports.*

## PIO for UPD350 Reset

## Hardware Timer

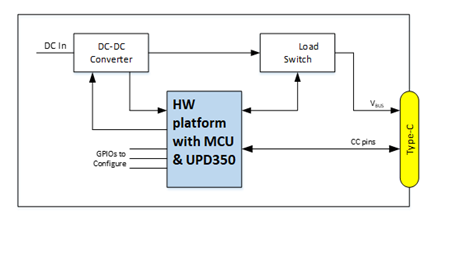
A Hardware timer with minimum resolution of 1ms is required for Zeus stack functionality.

The recommended resolution is 1ms.

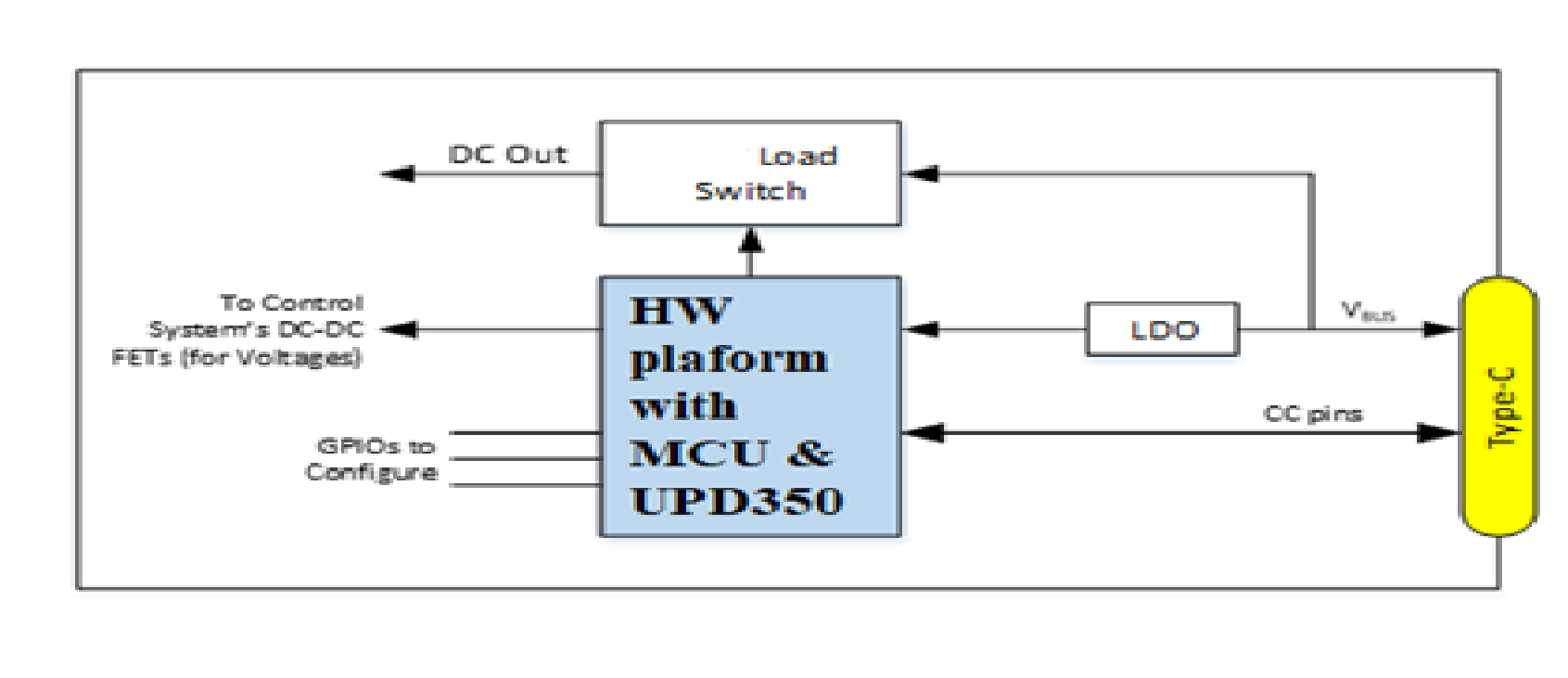
## DC-DC convertor

UPD350 cannot drive higher PD voltages directly. A DC-DC converter is required to drive higher PD voltages.

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



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



## Memory Requirement

### 32-bit MCU

* Estimated Code size of PSF Stack is 23KB for listed features in section [Supported features](#_Supported_features).
* Estimated Data RAM size is 1KB (RAM for Stack operation) + 1KB per port.

*Note: Above mentioned memory estimate is only approximate.*

### 16-bit MCU

TBD

### 8-bit MCU

TBD

## Multi-Port Support Requirement

PSF supports maximum of 4 Port.

## Endianness

PSF stack supports only Little-Endian format.

# Supported/Not Supported PD Features and Messages

## Supported/Not Supported PD features

This section details the features that are supported & not supported by Zeus stack.

### Supported features

* + USB-PD Source/Sink only support
  + Electronically Marked cable support
  + PD power negotiation up to 100 watts
  + PDFU support through CC
  + VCONN Power support
  + Extended message is supported via Chunking (Refer section [Supported & not Supported PD messages](#_Supported_&_not))
  + Supports only Fixed PDO’s

### Not supported features

* + FRS support
  + UVDM & USB Type-C Bridging Support
  + Dynamic power balancing
  + Dual-Role Power (DRP) functions.
  + Dual-Role Data (DRD) functions.
  + Alternate mode support.
  + Unchunked Extended Message support.

## Supported/Not Supported PD messages

This section details the supported & not supported PD messages by Zeus Stack.

### Supported PD Messages

The following [USBPD3] Control messages are supported:

* GoodCRC
* Accept
* Reject
* Ping
* PS\_RDY
* Get\_Source\_Cap (Sink Role only)
* Get\_Sink\_Cap (Source Role only)
* VCONN\_Swap
* Wait
* Soft\_Reset

The following [USBPD3] Data messages are supported:

* Source\_Capabilities (Source Role only)
* Request
* BIST
* Sink\_Capabilities (Sink Role only)

The following [USBPD3] Extended messages are supported:

* Firmware\_Update\_Request
* Firmware\_Update\_Response

### Unsupported PD Messages

The following [USBPD3] Control messages are not supported:

* GotoMin
* DR\_Swap
* PR\_Swap
* Get\_Source\_Cap\_Extended
* Get\_Status
* FR\_Swap
* Get\_PPS\_Status
* Get\_Country\_Codes

The following [USBPD3] Data messages are not supported:

* Battery\_Status
* Alert
* Get\_Country\_Info
* Vendor\_Defined

The following [USBPD3] Extended messages are not supported:

* Source\_Capabilities\_Extended
* Status
* Get\_Battery\_Cap
* Get\_Battery\_Status
* Battery\_Capabilities
* Manufacturer\_Info
* Security\_Request
* Security\_Response
* PPS\_Status
* Country\_Info
* Country\_Codes

# PSF Configurability TBD – To be renamed

The main advantage of PSF stack is configuration flexibility based on required USB-PD feature.

PSF allows to configure the following,

* Number of ports
* Power Role - Source only or Sink only
* Data Role - DFP or UFP
* Port Power control management
* Source & Sink PDOs
* HW Timer
* USB-PD & Type-C Timeouts
* MCU interrupts
* UPD 350 interrupts & Reset
* SPI Master
* Compile time Code & Data RAM inclusion/exclusion based on USB-PD Specification Revision & Power Role

PSF has three user configurable files to enable user various level of configurability and porting:

* PSF\_Config.h -> To configure PSF to different PD features and functionality in a PSF integrated SOC platform
* PSF\_BoardConfig.h -> To configure the Hardware board parameters in a PSF integrated SOC platform
* PSF\_Port.h -> To port and integrate to any new SOC platform

# Directory structure

PSF Stack is organised into three main folders:

* BSP\_Package
* Portable
* PSF\_Stack

## PSF Source

The Zeus Stack is organised as,

* .. /UPD350\_PSF\_Stack\_Package/PSF\_Stack/include/– Contains all the header files
* .. /UPD350\_PSF\_Stack\_Package/PSF\_Stack/pd/– Contains all the .c source files

TODO: Insert directory image

##### Zeus stack file description

|  |  |  |  |
| --- | --- | --- | --- |
| Application | File Name | File location | File Description |
| Device Policy Manager(DPM) | policy\_manager.c | ...\ZeusStack\pd\  policy\_manager.c | Maintains DPM functionality |
| policy\_manager.h | ...\ZeusStack\include\ policy\_manager.h |
| Policy Engine | policy\_engine.c | ...\ZeusStack\pd\ policy\_engine.c | PE functionality common to both Source & Sink are maintained in this file |
| policy\_engine\_src.c | ...\ZeusStack\pd\ policy\_engine\_src.c | PE functionality specific to Source operation |
| policy\_engine\_snk.c | ...\ZeusStack\pd\ policy\_engine\_snk.c | PE functionality specific to Sink operation |
| policy\_engine.h | ...\ZeusStack\include\ policy\_engine.h | Single header file corresponding to all PE related source file |
| Protocol Layer | protocol\_layer.c | ...\ZeusStack\pd\ protocol\_layer.c | APIs for Protocol layer functionality |
| protocol\_layer.h | ...\ZeusStack\include\ protocol\_layer.h |
| Type-C Connector Management | typeC\_control.c | ...\ZeusStack\pd\ typeC\_control.c | Type-C control functionality & connector management |
| typeC\_control.h | ...\ZeusStack\include\ typeC\_control.h |
| Interrupts/Timer Management | generic\_timer.c | ...\ZeusStack\pd\ generic\_timer.c | Contains APIs to manage multiple Software Timers |
| generic\_timer.h | ...\ZeusStack\include\ generic\_timer.h |
| upd\_interrupts.c | ...\ZeusStack\pd\ upd\_interrupts.c | UPD350 Alert Interrupt management (port specific) |
| upd\_interrupts.h | ...\ZeusStack\include\ upd\_interrupts.h |
| Debug support | debug.c | ...\ZeusStack\pd\ debug.c | Support file for debug through UART |
| debug.h | ...\ZeusStack\include\ debug.h |
| UPD HW | upd\_hw.c | ...\ZeusStack\pd\ upd\_hw.c | API Wrappers for SPI interface to communicate with UPD350 (n port specific) |
| upd\_hw.h | ...\ZeusStack\include\ upd\_hw.h |
| Globals | cfg\_globals.c | ...\ZeusStack\pd\ cfg\_globals.c | Maintains configurable globals. Globals in this file are configured based on ZeusStackConfig.h |
| cfg\_globals.h | ...\ZeusStack\include\ cfg\_globals.h |
| int\_globals.c | ...\ZeusStack\pd\ int\_globals.c | Maintains globals internal to the Zeus Stack. |
| int\_globals.h | ...\ZeusStack\include\  int\_globals.h |
| Generic Defines | generic\_defs.h | ...\ZeusStack\include\ generic\_defs.h | Generic defines for data types |
| Standard include file | stdinc.h | ...\ZeusStack\include\ stdinc.h | Standard include file to include Zeus stack |
| Stack main | pd\_main.c | ...\ZeusStack\pd\  pd\_main.c | Main function of Zeus stack |

## SOC Portable

TBD

## Demo Applications

TBD

# Frequently Asked Questions (FAQ)

1. Can Zeus stack be integrated to 8-bit Microcontroller?

* Yes, for integration with any HW platform refer document Microchip Integrating Zeus with any HW platform.

1. Is there any sample product with Zeus stack?

* Yes, Microchip UPD301A is USB-PD controller with Zeus stack integrated.

Refer Microchip UPD301A Datasheet for more details.